TEACHING FOR INNOVATION

A Workshop with Dr. William Bricken

presented for the

Software Engineering Institute
East China Normal University

March 2007

TOPICS

1) Teaching Practices
2) Learning Styles
3) Teaching Styles and Methods
4) Thesis Preparation and Guidance
5) Curriculum Design
6) Structuring Content
7) Projects and Assignments
8) Small Group Activities
9) Challenging the Students
10) Testing and Evaluation

TEACHING FOR INNOVATION

A Workshop with Dr. William Bricken

We have the opportunity to meet together for ten one hour sessions. Each session will consist of about one-third presentation, one-third discussion, and one-third exercises and activities. Content will be selected dynamically from these possibilities ("TP:" refers to articles from Tomorrow's Professors website:

1) TEACHING PRACTICES
   - Best Practices in Teaching and Learning
   - TP: Good Teaching: The Top Ten Requirements
   - TP: Seven Principles of Good Practice in Undergraduate Education
   - Teaching Styles
   - TP: Just-in-Time Teaching
   - TP: Problem-based Learning
   - TP: Learning by Doing
   - Instructor Control vs. Learner Control
   - TP: Silence and Structure in the Classroom
   - TP: Minimizing the Distances Between Teacher and Student

2) LEARNING STYLES
   - TP: Major Learning Theories of the Twentieth Century
   - TP: The Nature of Learning
   - TP: How Students Learn, How Teachers Teach, and What Goes Wrong
   - How People Learn
   - Learning Styles
   - Types of Learners
   - Meyers-Briggs Type Indicator
   - Multiple Intelligences
   - Talkers and Listeners

Teaching Examples (Bricken):
   - Management: Classification

3) TEACHING STYLES AND METHODS
   - TP: New Technologies in Teaching and Learning: Evolution of Lectures
   - TP: Powerpoint Debate
   - Teaching Large Classes: Strategies for Improving Student Learning
   - Activity Breaks: A Push for Participation
   - TP: Problem Solving Through Design
   - TP: Asking the Right Questions in Class
   - TP: Keeping Discussion Going Though Questioning, Listening, Responding
   - TP: Tactics for Effective Questioning

4) THESIS PREPARATION AND GUIDANCE
Small Piddly Projects, and Big Time Undertakings
TP: The Roles and Phases of Mentorship
TP: Combining Undergraduate Research and Learning
Teaching Examples (Bricken):
HCI: Project Ideas and Refinement
HCI: Design a Software Toolkit
Ethics: Local Expert

5) CURRICULUM DESIGN
Teaching and Facilitating Learning Syllabus
TP: The Function of the Course Syllabus
TP: The Value of Writing a Course Portfolio
Syllabus Elements
Course Structuring
Cognitive Taxonomy
Affective Domain Taxonomy
Psychomotor Domain Taxonomy
TP: 101 Things You Can Do the First Three Weeks of Class
Teaching Examples (Bricken):
Situated Curriculum
Curriculum Exercises
Just What is VR Anyway?
Wonderful Computer Science Books

6) STRUCTURING CONTENT
Lesson Plan Outline
How to Write Clear Objectives
Matching Objectives to Learning Styles
Teaching Examples (Bricken):
Formal: Proof Techniques, An Extended Example
Programming: A Small Interpreted Language
Programming: Pseudocode Assignment Package

7) PROJECTS AND ASSIGNMENTS
Teaching Examples (Bricken):
HCI: HCI Assignments
HCI: Interface Design Simulation
AI: LISP Program Modification Exercises
Management: Formal Model: Card Games
HCI: Complete Window System
VR: 3D Interactive Virtual Worlds
VR: Expandable Virtual Cube World

DS&A: Exam Package

8) SMALL GROUP ACTIVITIES
Managing Learner-Instructor Interaction and Feedback
TP: Group Presentations
TP: Establishing Ground Rules for Groups
TP: Integrating Team Exercises with Other Course Work
TP: Peer Instruction
TP: Difference Between Cooperative and Collaborative Learning
Teaching Examples (Bricken):
Management: simulation game
Management: archeologist, telephone drawing, consensus

9) CHALLENGING THE STUDENTS
Contests Motivate Top Students in Large Classes
Teaching Examples (Bricken):
Foundations: Chapter 0 and responses
DS&A: Versions of Factorial
Management: Measurement
Management: Critical Incidents
Formal: Formal Cube, Algebraic Specification
AI: Streams with Delayed Evaluation
AI: Knowledge Engineering
Ethics: Six Dilemmas
Applications to Teaching Mathematics (Bricken):
Foundations: Timeline
Foundations: Functions
Formal: Combinatorial Circuit Minimization
Spatial Math

10) TESTING AND EVALUATION
Nine Principles of Good Practice for Assessing Students
Assessment and Outcomes
Evaluating a Course
Teaching Examples (Bricken):
Foundations: Map of the territory
Ethics: Questions and text answers, course summary, content evaluation
Management: What you have learned

TEACHING FOR INNOVATION

TOPIC 1. TEACHING PRACTICES

Best Practices in Teaching and Learning

TP: Good Teaching: The Top Ten Requirements

TP: Seven Principles of Good Practice in Undergraduate Education

Teaching Styles

TP: Just-in-Time Teaching

TP: Problem-based Learning

TP: Learning by Doing

Instructor Control vs. Learner Control

TP: Silence and Structure in the Classroom

TP: Minimizing the Distances Between Teacher and Student

Best Practices in Teaching and Learning

The increasing focus on student learning as the central indicator of institutional excellence challenges many tacit assumptions about the respective roles of college students and faculty. In student-centered education, faculty take on less responsibility for being sources of knowledge, and take on greater responsibility as facilitators of a broad range of learning experiences. For their part, students are called on to take on more responsibility for their own learning.

As shown in the following table, the responsibilities of students and faculty and the relationships between them are quite different in the two models:

Domain	Teacher-centered	Learner-centered
Knowledge	Transmitted from instructor	Constructed by students
Student participation	Passive	Active
Role of professor	Leader/authority	Facilitator/partner in learning
Role of Assessment	Few tests, mainly for grading	Many tests, for ongoing feedback
Emphasis	Learning correct answers	Developing deeper understanding
Assessment method	Unidimensional testing	Multidimensional products
Academic culture	Competitive, individualistic	Collaborative, supportive

Most lists of important "best practices" include the following:

* Engage students in active learning experiences
* Set high, meaningful expectations
* Provide, receive, and use regular, timely, and specific feedback
* Become aware of values, beliefs, preconceptions; unlearn if necessary
* Recognize and stretch student styles and developmental levels
* Seek and present real-world applications
* Understand and value criteria and methods for student assessment
* Create opportunities for student-faculty interactions
* Create opportunities for student-student interactions
* Promote student involvement through engaged time and quality effort

GOOD TEACHING: THE TOP TEN REQUIREMENTS

By Richard Leblanc, York University, Ontario

This article appeared in The Teaching Professor after Professor Leblanc won a Seymous Schulich Award for Teaching Excellence including a $10,000 cash award. Reprinted here with permission of Professor Leblanc, October 8, 1998.

One. Good teaching is as much about passion as it is about reason. It's about not only motivating students to learn, but teaching them how to learn, and doing so in a manner that is relevant, meaningful, and memorable. It's about caring for your craft, having a passion for it, and conveying that passion to everyone, most importantly to your students.

Two. Good teaching is about substance and treating students as consumers of knowledge. It's about doing your best to keep on top of your field, reading sources, inside and outside of your areas of expertise, and being at the leading edge as often as possible. But knowledge is not confined to scholarly journals. Good teaching is also about bridging the gap between theory and practice. It's about leaving the ivory tower and immersing oneself in the field, talking to, consulting with, and assisting practitioners, and liaisoning with their communities.

Three. Good teaching is about listening, questioning, being responsive, and remembering that each student and class is different. It's about eliciting responses and developing the oral communication skills of the quiet students. It's about pushing students to excel; at the same time, it's about being human, respecting others, and being professional at all times.

Four. Good teaching is about not always having a fixed agenda and being rigid, but being flexible, fluid, experimenting, and having the confidence to react and adjust to changing circumstances. It's about getting only 10 percent of what you wanted to do in a class done and still feeling good. It's about deviating from the course syllabus or lecture schedule easily when there is more and better learning elsewhere. Good teaching is about the creative balance between being an authoritarian dictator on the one hand and a pushover on the other.

Five. Good teaching is also about style. Should good teaching be entertaining? You bet! Does this mean that it lacks in substance? Not a chance! Effective teaching is not about being locked with both hands glued to a podium or having your eyes fixated on a slide projector while you drone on. Good teachers work the room and every student in it. They realize that they are the conductors and the class is the orchestra. All students play different instruments and at varying proficiencies.

Six. This is very important -- good teaching is about humor. It's about being self-deprecating and not taking yourself too seriously. It's often about making innocuous jokes, mostly at your own expense, so that the ice breaks and students learn in a more relaxed atmosphere where you, like them, are human with your own share of faults and shortcomings.

Seven. Good teaching is about caring, nurturing, and developing minds and talents. It's about devoting time, often invisible, to every student. It's also about the thankless hours of grading, designing or redesigning courses, and preparing materials to still further enhance instruction.

Eight. Good teaching is supported by strong and visionary leadership, and very tangible institutional support -- resources, personnel, and funds. Good teaching is continually reinforced by an overarching vision that transcends the entire organization -- from full professors to part-time instructors -- and is reflected in what is said, but more importantly by what is done.

Nine. Good teaching is about mentoring between senior and junior faculty, teamwork, and being recognized and promoted by one's peers. Effective teaching should also be rewarded, and poor teaching needs to be remediated through training and development programs.

Ten. At the end of the day, good teaching is about having fun, experiencing pleasure and intrinsic rewards ... like locking eyes with a student in the back row and seeing the synapses and neurons connecting, thoughts being formed, the person becoming better, and a smile cracking across a face as learning all of a sudden happens. Good teachers practice their craft not for the money or because they have to, but because they truly enjoy it and because they want to. Good teachers couldn't imagine doing anything else.

STRATEGIES THAT IMPROVE UNDERGRADUATE EDUCATION

A variety of well-researched scholarly publications (for example, Association of American Colleges Task Group on General Education, 1988; Donovan, Bransford, & Pellegrino, 1999; Engelkemeyer & Brown, 1998; Study Group on the Conditions of Excellence in American Higher Education, 1984) spanning over fifteen years provide both faculty and academic administrators with a clear, consistent, and comprehensive description of instructional strategies for enhanced student learning. For illustrative purposes here, the findings and recommendations of three such reports will be mentioned briefly.

Seven Principles of Good Practice in Undergraduate Education

The single best known description of teaching practices that promote student learning is Chickering and Gamon's (1987, 1991, 1999) "Seven Principles of Good Practice in Undergraduate Education." First published in an article in the March 1987 AAHE Bulletin, the authors' provocative and pithy review of the research literature was later reproduced by the Johnson Foundation and over 150,000 copies were distributed. Subsequently, several articles and texts based on this landmark document, along with helpful instruments to assess instructor and institutional effectiveness in each of these seven areas, have been created (Gamson & Poulsen, 1989). These assessment inventories can be found in Chickering and Gamson (1991) and Hatfield (1995). The seven principles of good practice are these:

1. Encourages contact between students and faculty. Frequent student-faculty contact in and out of class is the most important factor in student motivation and involvement.

2. Develops reciprocity and cooperation among students. Learning is enhanced when it resembles a team effort rather than a solo race.

3. Encourages active learning. Learning is not a spectator sport. Students must talk about what they are learning, write about it, relate it to past experiences, and apply it to their daily lives. They must make what they learn part of themselves.

4. Gives prompt feedback. Knowing what you do and do not know focuses learning. Students need appropriate feedback on performance to benefit from courses.

5. Emphasizes time on task. Time plus energy equals learning. There is no substitute for time on task.

6. Communicates high expectations. If teachers expect more they will get more.

7. Respects diverse talents and ways of learning. There are many roads to learning. Students need the opportunity to show their talents and learn in ways that work for them.

Other Best Practices

Angelo (1993) similarly articulated for faculty and administrators a well-supported list "fourteen general research-based principles for improving higher learning."

1. Active learning is more effective than passive learning.

2. Learning is more effective and efficient when learners have explicit, reasonable, positive goals, and when their goals fit well with teachers' goals.

3. High expectations encourage high achievement.

4. Motivation to learn is alterable; it can be positively or negatively affected by the task, the environment, the teacher, and the learner.

5. Learning requires focused attention and awareness of the importance of what is to be learned.

6. To be remembered, new information must be meaningfully connected to prior knowledge, and it must first be remembered in order to be learned.

7. Unlearning what is already known is often more difficult than learning new information.

8. Information that is organized in personally meaningful ways is more likely to be remembered, learned, and used.

9. To be most effective, teachers need to balance levels of intellectual challenge and instructional support.

10. Mastering a complex skill or body of knowledge takes great amounts of time and effort.

11. Learning to transfer, to apply previous knowledge and skills to new contexts, requires a great deal of directed practice.

12. The ways in which learners are assessed and evaluated powerfully affect the ways they study and learn.

13. Interaction between teachers and learners is one of the most powerful factors in promoting learning; interaction among learners is another.

14. Learners need feedback on their learning, early and often, to learn well; to become independent learners, they need to become self-assessing and self-correcting.

Among the more recent analyses of how instructors can be most helpful in facilitating student learning is the report of the Joint Task Force on Student Learning, created by the American Association of Higher Education, the American College Personnel Association, and the National Association of Student Personnel Administrators. This document articulated ten principles of learning and identified a large number of actions and initiatives that have been used on various campuses to implement these principles (Engelkemeyer & Brown, 1998). The ten principles of learning are these:

1. Learning is fundamentally about making and maintaining connections: biologically through neural networks; mentally among concepts, ideas, and meanings; and experientially through interaction between the mind and the environment, self and other, generality and context, deliberation and action.

2. Learning is enhanced by taking place in the context of a compelling situation that balances challenge and opportunity, stimulating and using the brain's ability to conceptualize quickly ant its capacity and need for contemplation and reflection upon experiences.

3. Learning is an active search for meaning by the learner = constructing knowledge rather than passively receiving it, shaping as well as being shaped by experiences.

4. Learning is developmental, a cumulative process involving the whole person, relating past and present, integrating the new with the old, starting from but transcending personal concerns and interests.

5. Learning is done by individuals who are intrinsically tied to others as social beings, interacting as competitors or collaborators, constraining or supporting the learning process, and able to enhance learning through cooperation and sharing.

6. Learning is strongly affected by the educational climate in which it takes place; the settings and surroundings, the influences of others, and the values accorded to the life of the mind and to learning achievements.

7. Learning requires frequent feedback if it is to be sustained, practice if it is to be nourished, and opportunities to use what has been learned.

8. Much learning takes place informally and incidentally, beyond explicit teaching or the classroom, in contacts with faculty and staff, peers, campus

life, active social and community involvement, and unplanned but interesting, complex situations.

9. Learning is grounded in particular contexts and individual experiences, requiring effort to transfer specific knowledge and skills to other circumstances or to more general understandings and ability of individuals to monitor their own learning, to understand how knowledge is acquired to develop strategies for learning based on discerning their capacities and limitations, and to be aware of their own ways of knowing in approaching new bodies of knowledge and disciplinary frameworks.

10. Learning involves the ability of individuals to monitor their own learning, to understand how knowledge is acquired to develop strategies for learning based on discerning their capacities and limitations, and to be aware of their own ways of knowing in approaching new bodies of knowledge and disciplinary framework.

Teaching Styles

Felder & Soloman, 1992 – "When planning and developing instructional material, strive for a balance of teaching styles to match the various learning styles."

Rationale

Students will gain more knowledge, retain more information, and perform far better when teaching styles match learning styles (Lage, Platt, & Treglia, 2000). However, it is recognized that it is difficult to match with every learning style and therefore, a portfolio of teaching styles is recommended (Moallem, 2001).

Four Basic Teaching Styles

1. Formal Authority: A instructor-centered approach where the instructor feels responsible for providing and controlling the flow of content which the student is to receive and assimilate. The formal authority figure does not concern himself with creating a relationship with the student nor is it important if the students build relationships with each other.

2. Demonstrator or Personal Model: A instructor-centered approach where the instructor demonstrates and models what is expected (skills and processes) and then acts as a coach or guide to assist the students in applying the knowledge. This style encourages student participation and utilizes various learning styles.

3. Facilitator: A student centered approach where the instructor facilitates and focuses on activities. Responsibility is placed on the students to take initiative to achieve results for the various tasks. Students who are independent, active, collaborative learners learners thrive in this environment. Instructors typically design group activities which necessitate active learning, student-to-student collaboration and problem solving.

4. Delegator: A student-centered approach whereby the instructor delegates and places much control and responsibility for learning on individuals or groups of students. This type of instructor will often require students to design and implement a complex learning project and will act solely in a consultative role. Students are often asked to work independently or in groups and must be able to effectively work in group situations and manage various interpersonal roles.

Consider these questions on teaching style

* What teaching style do you mainly use?
* Does your style facilitate achievement of course goals?
* Should you consider new styles or continuations of teaching styles?

TEACHING STYLES

Grasha's Five Teaching Styles

1. Expert

Possesses knowledge and expertise that students need. Strives to maintain status as an expert among students by displaying detailed knowledge and by challenging students to enhance their competence. Concerned with transmitting information and insuring that students are well prepared.

Advantage: The information, knowledge, and skills such individuals possess.

Disadvantage: If overused, the display of knowledge can be intimidating to less experienced students. May not always show the underlying though processes that produced answers.

2. Formal Authority

Possesses status among students because of knowledge and role as a faculty member. Concerned with providing positive and negative feedback, establishing learning goals, expectations, and rules of conduct for students. Concerned with the correct, acceptable, and standard ways to do things and with providing students with the structure they need to learn.

Advantage: The focus on clear expectations and acceptable ways of doing things.

Disadvantages: A strong investment in this style can lead to rigid, standardized, and less flexible ways of managing students and their concerns.

3. Personal Model

Believes in "teaching by personal example" and establishes a prototype for how to think and behave. Oversees, guides, and directs by showing how to do things, and encouraging students to observe and then to emulate the instructor's approach.

Advantage: An emphasis on direct observation and following a role model.

Disadvantage: Some teachers may believe their approach is the best way leading some students to feel inadequate if they cannot live up to such expectations and standards.

4. Facilitator

Emphasizes the personal nature of teacher-student interactions. Guides and directs students by asking questions, exploring options, suggesting alternatives, and encouraging them to develop criteria to make informed choices. Overall goal is to develop in students the capacity for independent action, initiative, and responsibility. Works with students on projects in a consultative fashion and tries to provide as much support and encouragement as possible.

Advantage: The personal flexibility, the focus on students' needs and goals, and the willingness to explore options and alternative courses of action.

Disadvantage: Style is often time consuming and is sometimes employed in a positive and affirming manner.

5. Delegator

Concerned with developing students' capacity to function in an autonomous fashion. Students work independently on projects or as part of autonomous teams. The teacher is available at the request of students as a resource person.

Advantage: Helps students to perceive themselves as independent learners.

Disadvantage: May misread student's readiness for independent work. Some students may become anxious when given autonomy.

JUST-IN-TIME TEACHING
Blending Active Learning with Web Technology

PREFACE EXCERPT

Just-in-time Teaching (JiTT) is a pedagogical strategy that succeeds through a fusion of high-tech and low-tech elements. On the high-tech side, we use the World Wide Web to deliver multimedia curricular materials and to manage electronic communications among faculty and students. On the low-tech side, we maintain a classroom environment that emphasizes personal teacher-student and student-student interactions. We combine these disparate elements in several ways, and the interplay produces an educational setting that students find engaging and instructive. The underlying method is to use feedback between the Web and the classroom to increase interactivity and allow rapid response to students' problems.

We have based most of the discussion in this book on physics because it is our primary subject. However, there is nothing in our underlying method that is specific to physics. Interactivity and responsiveness are applicable to any instructional setting, and student achievement and motivation are important in any subject. While any course can benefit from JiTT, it is easy to describe those courses that can benefit the most: any course that students consider to be of secondary importance to their lives or their education. Courses taken to satisfy requirements and courses taken by part-time students meet these criteria. We use physics examples because we are familiar and experienced with. We hope that this does not put off instructors from other fields. We encourage others to adapt our ideas to their own subjects.

CHAPTER 1: WHAT IS JITT

Just-in-time-Teaching is a teaching and learning strategy comprised of two elements: classroom activities that promote active learning and World Wide Web resources that used to enhance the classroom component. Many industries use JiTT methods; they combine high-speed communications and rapid distribution systems to improve efficiency and flexibility. Our use of JiTT is analogous in many ways. We combine high-speed communications on the Web with our ability to rapidly adjust to our students' needs. The essential element is feedback between the Web-based and classroom activities.

WE have built the JiTT system around Web-used preparatory assignments that are due a few hours before class. The students complete these assignments individually, at their own pace, and submit them electronically. In turn, we adjust and organize the classroom lessons

in response to the student submissions "Just-in-Time." Thus, a feedback loop between the classroom and the Web is established. Each lecture is preceded and informed by an assignment on the Web. This cycle occurs several times each week, encouraging students to stay current and to do so by studying in several sessions that are short enough to avoid fatigue.

THE JiTT GOALS

We strive for both physics content mastery and acquisition of more general skills. We also design our courses to provide experiences in teamwork and opportunities to practice written and oral communication. Our goal is to help the whole spectrum of students advance and learn, rather than targeting the average students or either extreme. The JiTT strategy provides appropriate levels of support and feedback. JiTT provides remediation and encouragement to the weaker students while providing enrichment to the stronger students.

Students Enrolled in a Course that Successfully Implements JiTT Will:
* Gain both problem-solving skills and conceptual understanding
* Connect classroom physics to real-world phenomena and to their careers
* Be in control of their own learning processes
* Develop their critical thinking ability
* Develop their ability to frame and solve problems
* Develop their teamwork and communication skills

In addition to traditional homework assignments, students taking a JiTT-based course work in two interactive instructional environments. They work at their own pace in a virtual, Web-based setting that continually evolves with the progress of the class. They also collaborate with each other and instructors in a highly interactive classroom. Electronic communication among students and faculty provides a bridge between these two settings.

The JiTT Strategy Specifically Target Obstacles Facing Many of Today's Students:
* Motivation to learn physics
* Study habits and academic backgrounds
* Confidence in their ability to succeed
* Time constraints

These goals and difficulties are addressed by combining high-tech (Web-based) and low-tech (classroom) elements, which we will discuss throughout this book. The feedback between these elements and between the people involved is the most fundamental asset of the JiTT method.

THE JITT ENVIRONMENT

We have been student-testing this strategy for five semesters and are encouraged by the results, both attitudinal and cognitive. For details, visit the JiTT Web-site: http://webphysics.iupui.edu/jitt.html

In fact, working with the JiTT strategy has convinced us that the Web, combined with live teachers in the classroom, can humanize instruction for all students and make a real difference to the nontraditional student.

We have developed JiTT concurrently at three institutions: Indiana University Purdue University at Indianapolis (IUPUI), the United States Air Force Academy (USAFA) in Colorado Springs and Davidson College. The JiTT strategy is effective despite numerous differences among the three institutions (we will elaborate in Chapter 2). This suggests that JiTT is applicable in many other settings. The generality of the JiTT approach is also shown by our experiences with national JiTT workshops attended by faculty from a broad spectrum of institutions. For example, Daniel Kim-Shapiro, an assistant professor of physics at Wake Forest University, a private four-year liberal arts institution, has successfully employed the JiTT strategy in his calculus-based introductory physics course taken by approximately 50 students. most of whom were pre-med majors. His students gave the use of the strategy an overall rating of 8 out of 10 on an end-of-course survey. It is interesting to note that in similar surveys at IUPUI, USAFA, and Davidson, our students also gave the JiTT strategy a score of 8 out of 10.

WHAT JITT IS NOT

Despite our best efforts to explain what JiTT is, some readers may pick up false impressions. With this in mind, we would like to list a few things that JiTT is not:

* JiTT is Not a way to "process" more students
* JiTT is Not "Just-in-Time Training"
* JiTT is Not distance learning
* JiTT is Not computer-aided instruction
* JiTT is Not designed from student evaluations
* JiTT is Not market research

We pay attention to our students' comments and suggestions. We agree with some but disagree with others. The JiTT strategy was not designed with student evaluations as the motivation for change; it was designed to address pedagogical issues.

Why Change the Way We Teach?

What worked in the classroom a decade (or two or three) ago, however, will no longer suffice for the simple reason that past approaches fail to develop the full battery of skills and abilities desired in a contemporary college graduate. In June of 1994, a Wingspread Conference brought together state and federal policymakers, and leaders from the corporate, philanthropic, higher education, and accreditation communities to discuss quality in undergraduate education. This conference was sponsored by the Education Commission of the States (ECS), the Johnson Foundation, the National Governors' Association, and the National Conference of State Legislatures. The discussion that took place was based on the assertion that substantial improvement in American undergraduate education is needed to prepare students to function successfully in current business and industrial environments. The Conference developed the following list of important characteristics of quality performance of college and university graduates (Wingspread, 1994):

* High-level skills in communication, computation, technological literacy, and information retrieval to enable individuals to gain and apply new knowledge and skills as needed

* The ability to arrive at informed judgments-that is, to effectively define problems, gather and evaluate information related to those problems, and develop solutions

* The ability to function in a global community through the possession of a range of attitudes and dispositions including flexibility and adaptability, ease with diversity, motivation and persistence (for example, being a self-starter), ethical and civil behavior, creativity and resourcefulness, and the ability to work with others, especially in team settings

* Technical competence in a given field

* Demonstrated ability to deploy all of the previous characteristics to address specific problems in complex, real-world settings, in which the development of workable solutions is required

Survey results (Czujko, 1994) of all physics baccalaureates who were employed in either the private sector or government/national labs confirmed the Wingspread Conference conclusions. With approximately 80 percent response to the question, "What skills have you found to be the most useful in your work?", problem-solving, interpersonal skills, technical writing, and management skills were cited (greater than 60 percent) over physics knowledge. More recently, the Carnegie Foundation's report, Reinventing Undergraduate Education: A Blueprint

for America's Research Universities (1998) stated that "traditional lectures and note-taking were created for a time when books were scarce and costly and lecturing to large numbers of students was an efficient means of transferring knowledge." Lecturing is still efficient and has persisted as the traditional teaching method largely because it is familiar, easy, and how we learned. It does little, however, to foster the development of process skills to complement content knowledge.

There are teaching practices, however, that do foster such skill development without forsaking content. Quoting John Dewey's observation that "true learning is based on discovery guided by mentoring rather than the transmission of knowledge," (Boyer, 1998, p. 15) the Boyer report urged universities to Šfacilitate inquiry in such contexts as the library, the laboratory, the computer, and the studio, with the expectation that senior learners, that is, professors, will be students' companions and guidesŠ. The research university's ability to create such an integrated education will produce a particular kind of individual, one equipped with a spirit of inquiry and a zest for problem solving; one possessed of the skill in communication that is the hallmark of clear thinking as well as mastery of language; one informed by a rich and diverse experience. It is that kind of individual that will provide the scientific, technological, academic, political, and creative leadership for the next century. (Boyer, 1998)

Student-centered, inquiry-based instruction, particularly problem-based learning, falls right into line with this philosophy; indeed, the Boyer Commission pointed to the PBL efforts at the University of Delaware as one example of how to help students reach the important goals highlighted in the report.

What is Problem-based learning?

We believe that problem-based learning (PBL) provides a forum in which these essential skills will be developed. The basic principle supporting the concept of PBL is older than formal education itself; namely, learning is initiated by a posed problem, query, or puzzle that the learner wants to solve (Boud & Feletti, 1991). In the problem-based approach, complex, real-world problems are used to motivate students to identify and research the concepts and principles they need to know to work through those problems. Students work in small learning teams, bringing together collective skills at acquiring, communication, and integrating information. Problem-based instruction addresses directly many of the recommended and desirable outcomes of an undergraduate education: specifically, the ability to do the following:

* Think critically and be able to analyze and solve complex, real-world problems
* Find, evaluate, and use appropriate learning resources
* Work cooperatively in teams and small groups

* Demonstrate versatile and effective communication skills, both verbal and written
* Use content knowledge and intellectual skills acquired at the university to become continual learners

The PBL Cycle

PBL in the sciences traces its roots to the medical school setting where small groups of intellectually mature, highly motivated medical students work in small groups with a dedicated faculty tutor to learn basic science concepts in the context of actual clinical cases. The process of problem-based instruction (Boud & Feletti, 1997) follows:

* Students are presented with a problem (case, research paper, videotape, for example). Students working in permanent groups organize their ideas and previous knowledge related to the problem and attempt to define the broad nature of the problem.

* Throughout discussion, students pose questions called "learning issues" that delineate aspects of the problem that they do not understand. These learning issues are recorded by the group and help generate and focus discussion. Students are continually encourage to define what they know and-more importantly-what they don't know.

* Students rank, in order of importance, the learning issues generated in the session. They decide which questions will be followed up by the whole group and which issues can be assigned to individuals, who later teach the rest of the group. Students and instructor also discuss what resources will be needed to research the learning issues and where they could be found.

* When students reconvene, they explore the previous learning issues, integrating their new knowledge into the context of the problem. Students are also encouraged to summarize their knowledge and connect new concepts to old ones. They continue to define new learning issues as they progress through the problem. Students soon see that learning is an ongoing process and that there will always be (even for the teacher) learning issues to be explored.

PBL fosters the ability to identify the information needed for a particular application, where and how to seek that information, how to organize that information in a meaningful conceptual framework, and how to communicate that information to others. Use of cooperative working groups fosters the development of learning communities in all classrooms, enhancing student achievement (Johnson, Johnson, & Smith, 1991). Students who learn concepts in the context in which they will be used more likely to retain that knowledge and apply it appropriately (Albanese & Mitchell, 1993). They will also recognize that

knowledge transcends artificial boundaries since problem-based instruction highlights interconnections between disciplines and the integration of concepts.

References

Albanese, M.A. & Mitchell, S. (1993). Problem-based learning: A review of literature on its outcomes and implementation issues. Academic Medicine, 68, 52-81.
Boud, D., & Feletti, G. (1997) The challenge of problem-based learning (2nd ed.). London: Kogan Page.
Johnson, D. W., Johnson, R. T., & Smith, K. A. (1991). Cooperative learning: Increasing college faculty instructional productivity. (ASHE-ERIC Higher Education Report No. 4). Washington, DC: George Washington University.

Learning by Doing

Thanks to some excellent classroom and cognitive research in recent decades, we know a great deal about how learning happens and how little of it happens in lectures.1

There's no mistaking the catatonia that falls over classrooms after even just a few minutes of it. Numbed minds can't learn. The students who decide that their interests lie in cutting that 8 a.m. class and getting more sleep may be right on target.

You have roughly 40 contact hours in a typical course. If all you do in them is lecture, you might as well just hand out your notes and let the students find something more productive to do with all that time. The only way a skill is developed-skiing, cooking, writing, critical thinking, or solving thermodynamics problems-is practice: trying something, seeing how well or poorly it works, reflecting on how to do it differently, then trying it again and seeing if it works better. Why not help students develop some skills during those contact hours by giving them guided practice in the tasks they'll later be asked to perform on assignments and tests? Which is to say, why not use active learning? At several points during the class,

1. Give the students something to do (answer a question, sketch a flow chart or diagram or plot, outline a problem solution, solve all or part of a problem, carry out all or part of a formula derivation, predict a system response, interpret an observation or an experimental result, critique a design, troubleshoot, brainstorm, come up with a question,Š).

2. Tell them to work individually, in pairs, or in groups of three or four; tell them how long they'll have (anywhere from 10 seconds to two minutes); and turn them loose.

3. Stop them after the allotted time, call on a few individuals for responses, ask for additional volunteered responses, provide your own response if necessary, and continue teaching.

You may also occasionally do a think-pair-share, in which the students work on something individually and then pair up to compare and improve their responses before you call on them. As little as five minutes of that sort of thing in a 50-minute class session can produce a major boost in learning. For starters, it wakes students up: we have seen some of them elbowing their sleeping neighbors when an active learning task was assigned. Academically weak students get the benefit of being tutored by stronger classmates, and stronger students get the deep understanding that comes from teaching something to someone else. Students who successfully complete a task own the knowledge in a way they never would from just watching a lecturer do it. Students who are not successful are put on

notice that they don't know something they may need to know, so when the answer is provided shortly afterwards they are likely to pay attention in a way they never do in traditional lectures.

The number of possible active learning tasks is limitless.2 At a minimum, you can ask the same questions you would normally ask in your lectures, only now you'll get the whole class trying to answer them and not just the same two students who always answer the questions. You can also use any of the activities suggested in Item 1 of the list several paragraphs back, and you might occasionally run a TAPPS ("thinking-aloud pair problem solving") exercise, arguably the most powerful classroom instructional technique for promoting understanding.3 Have the students work in pairs through a complex derivation or worked-out problem solution in the text or on a handout, with one of them explaining the solution step-by-step and the other questioning anything unclear and giving hints when necessary. Periodically stop them, call on several of them for explanations, provide your own when necessary, and have the students reverse roles in their pairs and proceed from a common starting point. It may take most or all of a class period to work through the entire solution, but the students will end with a depth of understanding they would be unlikely to get any other way.

Here are several techniques to make active learning as effective as possible.

* At the beginning of the course, announce that you'll be assigning short exercises during class and explain why you're doing it (research shows students learn by doing, and the exercises will give them a head start on the homework and tests). The explanation can help defuse the resistance some students feel toward any teaching approach but the instructor telling them just what they need to know for the exam.

* After an active learning exercise, call on a few individuals for responses before opening the floor to volunteers. The knowledge that you might call on them gets active participation from students who would normally just sit passively and let others do the work.

* Go for variety. Vary the type of activity (answering questions, solving problems, brainstorming, etc.), the activity duration (10 seconds-2 minutes), the interval between activities (1-15 minutes), and the size of the groups (1-4 students). Mixing things up keeps active learning from becoming as stale as straight lecturing.

As many as half of the participants in our recent teaching workshops report using active learning in their classes, but nonusers often have concerns about the approach. (1) If I use active learning, will I still be able to cover my syllabus? (2) Can I do it in a really large class? (3) What should I do if some of my students refuse to participate?

We have offered detailed answers to the first two questions in another column4 and so will just give the short versions here. (1) Yes. (See Reference 5 for details on how.) (2) Yes, and in fact, the larger the class, the more important it is to use active learning. Try finding another way to get students actively engaged when there are 150 of them in the room.

What about (3)-students who refuse to participate? There may indeed be several who just sit staring straight ahead when groupwork is assigned, even after the awkwardness of the first few times has passed. We never see more than two or three of them in our classes, but for the sake of discussion let's say it's as many as 10% in yours. That means that while you're doing an active learning exercise, 90% of the students are actively engaged with the material and getting practice in the skills you're trying to teach them, and 10% are out to lunch. On the other hand, at any given moment in a traditional lecture, if as many as 10% of your students are actively involved with the lecture material you're doing very well. No instructional technique works for all students at all times: the best you can do is reach as many as possible, and 90% is more than 10%. If some students opt out, don't let it bother you-it's their loss, not yours.

In short, if you start using active learning in your classes, you can expect to see some initial hesitation among the students followed by a rapidly increasing comfort level, much higher levels of energy and participation, and above all, greater learning. See for yourself.

References

1. (a) W.J. McKeachie, P.R. Pintrich, Y-G Lin, D.A. Smith, & R. Sharma, Teaching and Learning in the College Classroom: A Review of the Research Literature (2nd Ed.), Ann Arbor, University of Michigan, 1990; (b) J.D. Bransford, A.L. Brown, and R.R. Cocking, Eds. How People Learn: Brain, Mind, Experience, and School, Washington, National Academy Press, 2000.

2. (a) D.W. Johnson, R.T. Johnson, & K.A. Smith, Active Learning: Cooperation in the College Classroom, 2nd Edn., Edina, MN, Interaction Book Company, 1998; (b) R.M. Felder and R. Brent, "Cooperative Learning in Technical Courses: Procedures, Pitfalls, and Payoffs," ERIC Document Reproduction Service, ED 377038 (1994), <www.ncsu.edu/felder-public/Papers/Coopreport.html>.

3. J. Lochhead & A. Whimbey, "Teaching Analytical Reasoning through Thinking Aloud Pair Problem Solving," in J.E. Stice (Ed.), Developing Critical Thinking and Problem-Solving Abilities, New Directions for Teaching and Learning, No. 30, San Francisco, Jossey-Bass, 1987.

4. R.M. Felder and R. Brent, FAQs-II. Chem. Engr. Education, 33(4), 276-277 (1999). <http://www.ncsu.edu/felder-public/Columns/FAQs-2.html

Instructor Control vs. Learner Control

Learner control refers to the ability of students to choose topics, assignments, project format or communication strategies according to their own interests and preferences.

Reigeluth & Stein 1983, p. 362 – "... instruction generally increases in effectiveness, efficiency, and appeal to the extent that it permits informed learner control by motivated learners."

Marlino, M.R. – "While some learner control can be motivating, too much can be confusing. The learner may not always be the best judge of the instruction required for effective learning.

Rationale

* Learner control strategies tend to increase students' involvement and achievement (Williams, 1996).

* "Learners given control over their instruction might be more likely to think about what they are doing as a result of having to make choices along the way" (Williams, 1996).

* "Adults desire to 'set my own learning pace,' 'use my own style of learning,' 'keep the learning strategy flexible,' and 'put my own structure on the learning project'." (Penland, 1979).

* "No open arguments exist against learner control, yet there are arguments on the degree of learner control, i.e., how much learner control should the students have (Schulmeister, 1997).

Definition of learner control

Williams, 1996 – '"[Learner control is when] learners make their own decisions regarding to some aspects of the 'path', 'flow', or 'events' of instruction."

In many classes, learner control can take the form of learner choice, i.e., alternatives that achieve the same learning objectives but give learners opportunities to select the best way to learn and the best way to show what they have learned. Consider the following situations:

* Students can choose to write one 50-page paper or three 12-page papers on 3 different topics.

* Students can choose alternative project formats (e.g. a poster, presentation, or a short paper).

* Students can choose to hand in a draft of the paper before it's due, it's not graded, but you give the students feedback.

* Students are given the option of additional practice exercises. They are available but not required.

* Students can choose to do an interview through video-conferencing or face-to-face.

* Students can choose to take a class in a traditional lecture format, an online format or a combined ("hybrid") online and traditional format.

Degree of Learner control

The degree of learner control should depend on following variables: (Depover and Quintin, 1992; Hannafin, 1984; Milheim & Martin, 1991; Steinberg, 1989):

* Previous knowledge
* Student strategy and ability
* Learning progress
* Complexity of material
* Familiarity with the subject

Instructor guidance vs. Learner control

Some questions to consider:

* Is it necessary to tell the students every step they should take?

* Do you provide resources and let the students select from among the resources?

* Should you let students choose their own research topic, will you assign a topic, or will you provide a list of topics from which they may choose?

SILENCE AND STRUCTURE IN THE CLASSROOM
From Seminar to Town Meeting via 'Post-it's

Chad M. Hanson, Ph.D.
Northcentral Tech
Wausau, WI

Like most, I started out teaching the way I was taught. My first inclination as a faculty member was to reproduce the format of the graduate course. I wanted my students to share the same feeling of excitement I had known as a student. I wanted their minds to sharpen and their pulses to quicken just as mine had in those vital forums.

Sociology is my subject so it's probably no surprise that I started teaching by selecting a textbook and several readings from within the field. Mindful of my students' level of preparation, I chose well-respected articles written for a general audience, and I assigned only four of them in my Introduction to Sociology classes. I explained to students early on in the semester that the articles would serve as a basis for in-class discussions.

When the first discussion date rolled around I walked into class with genuine enthusiasm. I welcomed the students, reminded them about the discussion, then I followed in the footsteps of one of my fondest mentors by issuing a familiar challenge. "OK," I said, "who would like to begin?" No one began. There were no hands in the air. I did not hear the cacophony of voices I had come to know so well in graduate school--everyone anxious to support or refute the claims of the author now up for discussion. Instead there was silence. This wasn't graduate school. Twenty-nine pairs of eyes pointed in my direction. So I began.

I continued, and eventually I finished the discussion myself. Meanwhile, students wrote in their tablets. They took what looked like detailed notes while I talked, and that was gratifying, but not part of my plan. Unfortunately, I repeated roughly the same series of events four more times the same week. By Friday afternoon, I had decided the approach that worked so well for my professors was not going to work for me.

The Pendulum Swings: Structured Cooperative Learning Activities

The first step in any process of redemption involves admitting you have a problem, which, obviously, I did. I needed help and I sought it out. The first place I found guidance was the literature on cooperative learning. Years before, I ran across a copy of Ken Bruffee's Collaborative Learning (1993). I revisited Bruffee first, because I remembered that he outlines a theoretical foundation for collaboration in the college classroom. For anyone experimenting with discussion leading or the grouping of students for

educational purposes, I recommend Bruffee's work.

For the nuts and bolts of getting students involved in conversation, I relied on the work of David and Roger Johnson, namely Active Learning (Johnson, Johnson, and Smith, 1991). Over the last several years I have had a great deal of success using the procedures described by these authors. Success with the cooperative learning approach described by Johnson and Johnson hinges on having a clear set of guidelines for students. In the Johnsons' model, each student must have a clearly defined role in the class. The instructor's job is to ensure that the students' roles and the objectives of the class are both well defined. I have found that when I take that initiative, the procedures outlined in Active Learning provide a formal structure for ensuring that students stay engaged with course material, and with one another, during the class periods I set aside for cooperative work.

Although I quickly became comfortable with the Active Learning techniques, I found that I still had a longing to create the excitement and spontaneity of the unstructured and free-ranging discussions that took place in my graduate courses. At the same time, I also began to feel a responsibility to create an environment where students could interact with one another in an exchange that would mirror that of a discussion held outside of the classroom in places where our democratic traditions are strongest (Beckman, 1990). I had in mind the New England town meeting as an ideal (Bellah, et al., 1985). Consequently, I set out to create a forum where I did not personally determine the nature of each student's contribution to in-class discussions. I did not want to prohibit the discussions from unfolding on their own, as they would in a town meeting or similarly democratic forum.

As I began to conceive the new format for my in-class discussions, I realized that citizens who attend town meetings are a self-selected group. The attendees are there because they have something to say. My students are also a self-selected group, but the primary reason for selecting one of my courses is that it fulfills a requirement for the degrees that they seek. Given the lack of inherent motivation, I needed a strategy that would ensure everyone's participation. The solution to my problem was as near as the pad of Post-it notes lying next to my office telephone.

Finding the Middle Ground: Required Participation

Today, I use a particular format to create an environment in the classroom that approximates a town hall meeting. The first step I take is to allow the students to decide the topics to be discussed. I begin by having students brainstorm a list of potential topics in small groups. After each group generates its own list, we compile all the topics on a chalkboard and hold a vote to determine the top ten to be discussed.

Once the topics are determined I select groups of two to four students, at random, to lead the discussions. I require discussion leaders to find at least two articles on their topic and I give them a list of things to consider when they analyze the articles, including a set of guidelines on how to prepare a set of talking points to use during the town hall meetings.

However, in the town hall format, the most important step is to ensure that all of the students have both the opportunity and the incentive to participate. In order to create that incentive I make each discussion worth two points. To earn the points, people have to take part.

I begin town hall meetings by giving two Post-it notes to every student in class. The Post-its are worth a point each, so I have them write their name on each note. After the discussion leaders are given the floor, all of the students are free to raise questions or to comment. Each time they add to the discussion, students stick one of their Post-it notes on the front of their desk for everyone to see. Once a person has participated twice and placed both Post-its on the front of their desk, they can no longer earn points but they may still contribute to the discussion.

I have found that Post-it notes, visible to all, serve two important roles in class. First, for students who might otherwise dominate discussions, the notes are visual reminders that they have already said their piece. I have found this to be a subtle, but important reminder in those cases. Second, the notes are a less than subtle reminder to those less likely to participate. In this case the notes serve as a reminder that you do not earn points if you do not contribute to the discussion. I realize that may seem like undue pressure to place on students who may not wish to participate. However, during the last three semesters I have found that students who participate quickly and place their notes out in front right away often go on to create opportunities for other students to answer questions or to comment. One of the most rewarding observations I have made during town hall meetings has been the tendency of outspoken members of class to encourage others to add their voices to the conversations. Each semester I watch students take steps to ensure that everyone has a chance to contribute.

Conclusion

During the time I've spent using Post-it notes and town hall meetings, I have felt very close to the format of the graduate seminar I enjoyed so much as a student. The discussions flow freely, they are full of excitement and they serve as a model for democratic participation. As an unintended consequence, I have also been pleased to find that Post-its have had the effect of producing an environment where students consistently demonstrate that they value each other's thoughts. When I use the notes in class I am

guaranteed not to face the silence that vexed me as a beginning teacher. At the same time, they provide a structure that is subtle enough to allow the freedom necessary for students to determine the nature of their own contribution to class. Today I can say that the unassuming stack of Post-its that sits next to my phone provides the means to create balance, equity and a model for democracy in the classroom.

References

* Beckman, M. 1990. "Collaborative Learning: Preparation for the Workplace and Democracy?" College Teaching 38/4: 128-133.
* Bellah, R., et al. 1985. Habits of the Heart: Individualism and Commitment in American Life. Berkeley, CA: University of California Press.
* Bruffee, K. 1993. Collaborative Learning: Higher Education, Interdependence and the Authority of Knowledge. Baltimore, MD: Johns Hopkins University Press.
* Johnson, D., Johnson, R., and Smith, K. 1991. Active Learning: Cooperation in the College Classroom. Edina, MN: Interaction Book Company.

MINIMIZING THE DISTANCES BETWEEN TEACHER AND STUDENT

(1) Let students know that they are not faces in an anonymous audience.
In large courses students often think that their classroom behavior (eating, talking, sleeping, reading the newspaper, arriving late, leaving early) goes unnoticed. Tell students that you are aware of what is happening in class and act accordingly.

(2) Ask students to refrain from sitting in certain rows of the classroom.
For example, one math professor asks students to sit only in rows 1 ,2, 4, 5, 7 ,8 and so on. With rows 3, 6, and 9 empty, he can walk thorough the audience between the rows, which is especially important while students are working at their seats. Of course this suggestion is only possible if your course is not maximally enrolled or oversubscribed and if your classroom is large enough.

(3) Recognize students' outside accomplishments.
Read your campus newspaper, scan the dean's list, pay attention to undergraduate awards and honors, and let students know you are aware of their achievements.

(4) Occasionally attend lab or discussion sections. Sections give you an opportunity to meet students and answer questions in a smaller setting.

(5) Capitalize on outside events or situations, as appropriate.
Relate major world events or events on campus both to topics in your class and to the fabric of your students' lives outside the classroom. Consider distributing a calendar or setting aside class time to mention community events and resources that will enhance their understanding of the subject matter: plays, lectures, performances, demonstrations and the like.

(6) Arrive early and chat with students.
Ask how the course is going, whether they are enjoying the readings, whether there is anything they want you to include in the lectures. Or ask students to walk back with you to your office after class.

(7) Read a sampling of assignments and exams.
If you have graduate student instructors who do most of the grading, let students know you will be reading and grading some of their assignemtns and exams.

(8) Seek out students who are doing poorly in the course.
Write "I know you can do better, see me during my office hours on all exams graded C or below. Offer early assistance to students having difficulty.

(9) Acknowledge students who are doing well in the course.

Write "Good job! See me after class on all exams graded A, or above. Take a moment after class to compliment students who are excelling. Some teachers send "A" students a letter of congratulations at the end of the semester.

(10) Schedule topics for office hours.
If students are reluctant to come, periodically schedule a "help session" on a particular topic rather than a free-form office hour. See "Holding Office Hours."

(11) Talk about questions students have asked the previous terms.
Mention specific questions former students have asked and explain why they were excellent questions. This lets students know that you take their questions seriously and that their questions will contribute to further offerings of the course. (Source: Gleason, 1986).

(12) Listen attentively to all questions and answer them directly.
If they answer to a question is contained in material you will cover during the remainder of the lecture, acknowledge the aptness of the question directly whey you arrive at that subject. See "Fielding Students' Questions."

(13) Try to empathize with beginners.
Remember that not all of your students are as highly motivated and interested in the discipline as you were when you were a student. Slow down when explaining complex ideas, and acknowledge the difficulty and importance of certain concepts or operations. Try to recall your first encounter with the concept - what examples, strategies, or techniques helped clarify it for you? By describing that encounter and its resolution to your students, you not only explain the concept but also convey the struggle and rewards of learning.
(Source: Gleason, 1986)
[Gleason, M. "Better Communication in Large Courses,"*College Teaching* 1986, 34(1), 20-24.]

TEACHING FOR INNOVATION

TOPIC 2. LEARNING STYLES

TP: Major Learning Theories of the Twentieth Century

TP: The Nature of Learning

TP: How Students Learn, How Teachers Teach, and What Goes Wrong

How People Learn

Learning Styles

Types of Learners

Meyers-Briggs Type Indicator

Multiple Intelligences

Talkers and Listeners

Teaching Examples (Bricken):

Management: Classification

MAJOR LEARNING THEORIES OF THE TWENTIETH CENTURY

Pantel, C. (1997). Educational Theory. A Framework for Comparing Web-Based Learning Environments.

Chapter 2 of Masters Thesis: Simon Fraser University.

Abstracted by Vaibhavi Gala

THE RESPONSE STRENGTHENING MODEL, which influenced the first half of this century, lays emphasis on the role of feedback to enhance learning. Knowledge is considered to be the associations people make between stimuli and responses. Drill and practice was the instructional method of choice by the proponents of this theory.

THE INFORMATION PROCESSING MODEL proposes that knowledge is a definite entity that can be transferred from one person to another. This assumption gave rise to didactic instruction and classical instructional design with lecturing as the prevalent instructional technique.

CONSTRUCTIVISM came into light in the early 1980s and proposes that knowledge is 'constructed' individually in a person's mind. Individuals have their own mental framework which is a function of their beliefs, past experiences and knowledge. When a person comes across new information, he understands and assimilates it in the context of his existing mental structures thereby constructing new knowledge. Hence, learning is seen as a process of internal negotiation of meaning.

Under constructivism the goal of instruction is to help learners 'develop learning and thinking strategies' and evaluation of learning outcomes consists of 'determining how the student structures and processes knowledge'. Constuctivism propagates creating a learning environment that facilitates higher-order thinking and metacognition (awareness of one's own cognitive abilities and the ability to apply them to the task at hand). It shifts cognitive labor such as analysis and synthesis of information from teachers to the learners. Constuctivists advocate that students be allowed and encouraged to take ownership of their learning thus ensuring that learning activities are more authentic and meaningful to them.

Within the constructivist community there seems to be agreement that constuctivist learning environments are good for advanced knowledge acquisition. There is no consensus however, on its appropriateness for lower levels of education, which involve introductory knowledge acquisition

SOCIOCULTURAL THEORIES are rooted in Constuctivism but they focus on the role of community and environment in the creation of knowledge as opposed to the constructivist focus on internal negotiation of meaning. They acquiesce that meaning can vary but contend that it is defined by the community of practitioners which uses it. Thus, knowledge resides in communities. Meaning-making is the result of active participation in socially, culturally, historically, and politically situated contexts. Socioculturalism is more extreme in its beliefs than situated learning in that it focuses on the development of the collective knowledge of a community as opposed to the development of individuals' knowledge within a community.

Adherents to the sociocultural theories of learning, like constuctivists, argue that it is important to reflect the complexity of the application domain in the learning environment. This would contribute to the authenticity of the learning activities.

Instructional Techniques based on the constuctivist and sociocultural theories include:

a) Scaffolding : Teachers support a learner's personal construction of knowledge by offering comments, suggestions, feedback or observation

b) Fading: Once the learner progresses towards mastery, teachers remove the supports they provided to make the learner self-sufficient.

c) Cognitive Apprenticeship: Learners learn by actually engaging in the activity they want to learn about with the support of knowledgeable others in the field. (similar to traditional apprenticeships: learning by doing)

d) Collaborative Learning: Learners develop their knowledge by sharing ideas, reflecting and interacting in learning groups.

RELEVANCE

The author provides a reasonable account of the contemporary educational theories of constructivism and socioculturalism (though he has not elaborated on situated learning, a variant of socioculturalism). Understanding the theoretical framework which describes the meaning of knowledge and the process of learning would enable the Learning Lab personnel to form their own informed opinions about the models, reflect on what a learning environment should support and articulate their reasoning for the basis of the various projects. It would also inform the design of the framework for future endeavors.

THE NATURE OF LEARNING

Again, this is a huge topic. There is a great deal of research into how students learn at the postsecondary level. We particularly recommend the report of the National Research Council, as commissioned by the U.S. Department of Education's Office of Educational Research and Improvement (OERI). This research is documented in How People Learn (Donovan, Bransford, & Pellegrino, 1999). One of the assumptions behind this work is that there is a significant gap between what is known about learning processes and how we teach; in other words, between research and
practice. The authors attempt, in a very practical way, to highlight just what is going on when people learn well. We share their view that an understanding of learning processes should inform how we teach, including how we use technology for teaching.

We will just very briefly introduce some key concepts that need to be explained in order to show the link between different understandings of how students learn and the design of technology-based teaching.

There are many different theories of learning. We will look first at three main categories of learning theory: behaviorism, cognitivism, and the social construction of knowledge. We also discuss some issues arising from these theories, such as cognitive development, student differences, and motivation and engagement in learning. These themes of how students learn and what influences their learning will, as with epistemology, recur throughout the book as we examine the role of media and technology in teaching and learning, and the planning, design, and delivery of technology-based courses.

Behaviorism

Behaviorist psychology arose in the 1920s and 1930s from an attempt to model the study of human behavior on the methods of the physical sciences. Therefore it concentrates attention on aspects of behavior that are capable of direct observation and measurement. At the heart of behaviorism is the idea that certain behavioral responses become associated in a mechanistic and invariant way with specific stimuli. Hence a certain stimulus will evoke a particular response. At its simplest, the response may be a purely physiological reflex action, like the contraction of an iris in the eye when stimulated by bright light.

However, most behavior is more complex. Nevertheless, according to behaviorists, it is possible to reinforce through reward or punishment the association between any particular stimulus or event and a particular response. The bond formed between a stimulus and a response will depend on the existence of an appropriate means of reinforcement at the time of association between stimulus and response.

Behavior therefore can be modified or controlled by appropriately reinforcing random behavior (trial and error) as it occurs.

This is essentially the concept of operant conditioning, a principle most clearly developed by Skinner (1969). He showed that pigeons could be trained in quite complex behavior by rewarding
particular, desired response, randomly occurring, with appropriate stimuli, such as the provision of food pellets. He also found that intervening stimuli to be present, thus linking an initially remote
stimulus with a complex behavior. Furthermore, inappropriate or previously learned behavior could be extinguished by withdrawing reinforcement. Skinner also claimed that rewarding behavior was more effective than punishment.

Underlying this approach is the belief that learning is governed by invariant principles, and these principles are independent of conscious control on the part of the learner. Behaviorists attempt to maintain a high degree of objectivity in how they view human activity, and they generally reject reference to unobservable states, such as feelings, attitudes, and consciousness. Human behavior is above all seen as predictable and controllable. Behaviorism stems from a strongly objectivist epistemological position.

Skinner's theory led to the development of teaching machines, measurable learning objectives, computer-assisted instruction, and multiple choice tests. There was also a tendency until recently to see technology, particularly computers, as being closely associated with behaviorist approaches to learning. Today there has been a strong movement away from behaviorist approaches to teaching in higher education, although its influence is still strong in corporate and military training and in some areas of science, engineering, and medical training.

Cognitivism

Behaviorism denies or ignores mental activity as the basis for learning. Learning for behaviorists is determined by external environmental structures that lead to reinforcement of behavior, rather than to mental processing or conscious thought on the part of the learner. Cognitivists, though, insist that there are mental processes "internal and conscious representations of the world" that are essential for human learning. Fontana (1981) summarizes the cognitive approach as follows:

"The cognitive approach holds that if we are to understand learning we cannot confine ourselves to observable behavior, but must also concern ourselves with the learner's ability mentally to reorganize his psychological field (i.e., his inner world of concepts, memories, etc.) in response to experience. This latter approach therefore lays stress not only on the environment, but upon the way in which the individual

interprets and tries to make sense of the environment. It sees the individual not as the somewhat mechanical product of his environment, but as an active agent in the learning process, deliberately trying to process and categorize the stream of information fed into him by the external world." (p. 148)

Thus the search for rules, principles, or relationships in processing new information, and the search for meaning and consistency in reconciling new information with previous knowledge, are key concepts in cognitive psychology. Cognitive psychology is concerned with identifying and describing mental processes. In some ways, basic mental processes are often considered genetic or hard-wired but can be programmed or modified by external factors, such as experience.

Cognitive approaches to learning cover a very wide range. On the one hand, attempts have been made through areas such as artificial intelligence to provide mechanical, electronic, and physical representations of mental processes, reflecting very much as objectivist epistemological position. On the other hand, teachers who place a strong emphasis on learners' developing personal meaning through reflection, analysis, and construction of knowledge through conscious mental processing would indicate much more of a constructivist epistemological position. Cognitive approaches to learning "with their focus on abstraction, generalization, and creative thinking" seem to fit much better in higher education.

The Social Construction of Knowledge

We have pulled together different theories of learning here under the common theme of the social construction of knowledge. Both behaviorist and some elements of cognitive theories of learning are deterministic, in the sense that behavior and learning are believed to be rule-based and operate under predictable and constant conditions over which the individual learner has no or little control.

However, the trend these days is to recognize the importance of consciousness, free will, and social influences on learning. Although constructivism has become the "flavor of the month" in higher education in recent years, the belief that humans are essentially active and free and strive for meaning in personal terms has been around for a long time. Carl Rogers (1969) stated that "every individual exists in a continually changing world of experience in which he is the center." The external world is interpreted within the context of that private world.

Individuals consciously strive for meaning to make sense of their environment in terms of past experience and their present state. It is an attempt to create order in their minds out of disorder, resolve incongruities, and reconcile

external realities with prior experience. The means by which this is done are complex and multifaceted, from engaging in personal reflection to seeking new information to testing ideas through social contact with others. Problems are resolved, and incongruities sorted out, through strategies such as seeking relationships between what was known and what is new, identifying similarities and differences, and testing hypotheses. Reality is always tentative and dynamic.

For many educators, the social context of learning is critical. Ideals are tested not just on the teacher but also with fellow students, friends, and colleagues. Furthermore, knowledge is mainly acquired through social processes or institutions that are socially constructed: schools, universities. What is taken to be "valued" knowledge is also socially constructed. Knowledge is thus not just about content but also about values.

One set of values comprises those around the concept of a liberal education. According to this notion, one of the principal aims of education is to develop a critical awareness of the values and ideologies that shape the form of received knowledge. This aim suggests a constant probing and criticism of received knowledge.

One consequence of theories of social construction of knowledge is that each individual is considered unique, because the interaction of each person's different experiences and the search for personal meaning result in each person being different from anyone else. Behavior is thus not predictable or deterministic, at least not at the individual level (although pollsters will argue that patterns of group behavior is predictable).

The key point here is that learning is seen as essentially a social process, requiring communication among learner, teacher, and others. This social process cannot effectively be replaced by technology, although technology may facilitate it.

References

Donovan, M., Bransford, J., & Pellegrino, J. (Eds.) (1999). How people learn: Bridging research and practice. Washington, DC: National Research Council, U.S. Department of Education's Office of Educational Research and Improvement.

Fontana, D. (1981). Psychology for teachers. London: Macmillan/British Psychological Society.

Rogers, C. (1969). Freedom to learn. Columbus, OH: Merrill.
Skinner, B. (1969). Contingencies of reinforcement. New York: Appleton-Century-Crofts.

HOW STUDENTS LEARN, HOW TEACHERS TEACH, AND WHAT GOES WRONG WITH THE PROCESS

Richard M. Felder
North Carolina State University
8/3/98

* Different students learn in different ways, that is, they have different learning styles.

* Different faculty also teach in different ways, that is, they have different teaching styles.

* Learning styles can be defined in large part by the answers to five questions:

(1) What type of information does the student preferentially perceive: sensory (external) - sights, sounds, physical sensations, or intuitive (internal) - possibilities, insights, hunches?

(2) Through which sensory channel is external information most effectively perceived: visual - pictures, diagrams, graphs, demonstrations, or auditory - words, sounds?*

(3) With which organization of information is the student most comfortable: inductive - facts and observations are given, underlying principles are inferred, or deductive - principles are given, consequences and applications are deduced?

(4) How does the student prefer to process information: actively - through engagement in physical activity or discussion, or reflectively - through introspection?

(5) How does the student progress toward understanding: sequentially - in continual steps, or globally - in large jumps, holistically?

* Teaching styles may also be defined in terms of the answers to five questions:

(1) What type of information is emphasized by the instructor: concrete - factual, or abstract - conceptual, theoretical?

(2) What mode of presentation is stressed: visual - pictures, diagrams, films, demonstrations, or verbal - lectures, readings, discussions?

(3) How is the presentation organized: inductively - phenomena leading to principles, or deductively - principles leading to phenomena?

(4) What mode of student participation is facilitated by the presentation: active - students talk, move, reflect, or passive - students watch and listen?

(5) What type of perspective is provided on the information presented: sequential - step-by-step progression (the trees), or global - context and relevance (the forest)? [6]

* Problems occur because there are often significant mismatches between the learning styles of most college students and the teaching styles of most college professors.

* The key to dealing with the above reality is "BALANCE." The goal is NOT to match each students preferred learning style with a corresponding teaching style, rather it is to present a variety of teaching styles to all learners.

* Professionals need to function as sensors (practical, methodical) and intuitors (interpretive, imaginative), visual and verbal learners, etc.

* Students taught only in their less preferred modes can't learn effectively.

* Students taught only in their preferred modes won't develop balanced strength.

* Solution: Teach to both sides of each dimension.

Felder offers the following recommendations to address various learning types:

* Establish relevance and provide applications for all course material. Before presenting theoretical material, provide graphic examples of phenomena that the theory describes or predicts. (sensing, inductive, global)

* Balance concrete information (facts, observations, data) (sensing) and abstract information (principles, theories, models) (intuitive) in all courses.

* Make extensive use of pictures, schematics, graphs, and simple sketches before, during, and after presenting verbal material. (sensing, visual)

* Use multimedia presentations. (sensing, visual) Provide demonstrations (sensing, visual), hands-on if possible.

* Use some numbers in illustrative examples, not just algebraic variables. (sensing)

* Give students time to think about what they have been told. Assign "one-minute papers" (Write the main point of this lecture and the muddiest point) or learning logs. (reflective)

* Give small-group exercises in class. (active, reflective)

* Use computer-assisted instruction (if you have software that allows for experimentation and provides feedback). (sensing, active)

* Assign some drill exercises in homework (sensing, active) but don't overdo it (intuitive, reflective).

* Assign some open-ended problems and exercises that call for creative thinking and critical judgment. (all styles)

* Have students cooperate on homework. (all styles)

* Limit new material, surprises, twists, etc., on timed tests and minimize speed as a critical factor. (sensing)

* Encourage creative solutions, even wrong ones. (all styles)

* Tell students about their learning styles or let them assess their own style. See the Index of Learning Styles at above web site.

* Try a few of these suggestions at a time. Adopt the ones that work. Then try a few more.

R.M. Felder & R. Brent, National Effective Teaching Institute, 1998

R. M. Felder and L.K. Silverman, "Learning and teachings styles in engineering education," Journal of Engineering Education, vol. 77, no. 2, April, 1988

How People Learn

* Learning occurs in context.
* Learning is active.
* Learning is social.
* Learning is reflective.

Driscoll (2002) proposes the following principles for how people learn:

* Learning occurs in context: Learning must happen within certain context. Without an appropriate setting, learning is unlikely to succeed.

* Learning is active. Learners have to be mentally active during learning activities, make connections between the new knowledge and existing knowledge, and construct meaning from their own experiences.

* Learning is social. Learners benefit from working collaboratively in groups so that they can hear different perspectives and accomplish the learning tasks with the help of their peers and experts.

* Learning is reflective. Learning is facilitated when learners are given chances to express and evaluate on their own thinking.

Different Learning Theories

There are three dominant learning theories which provides different perspectives on how learning occurs.

1. Behaviorism: Focuses on observable behavior rather than non-observable mental events. It suggests learning is a relatively permanent change in behavior due to experience (Ormrod, 1999). The learner must be engaged in the behavior in order to learn.

2. Cognitivism: Focuses on the internal mental events. Cognitivism considers how people perceive, interpret, remember and think about the environmental events they experience. It suggests learning occurs when information is mentally processed and the structure of learner's knowledge changes.

3. Constructivism: Constructivism is also internal oriented, asserting that one's knowledge, as well as the learning process itself, is constructed by the learners according to their interpretation of their own experiences.

What is learning?

Different learning theories give different definitions for what learning is.

1. Behaviorism Perspective of Learning

“Learning is a change in human disposition or capability that persists over a period of time and is not simply ascribable to processes of growth.” (Gagne, 1985, p. 2). It represents itself in a change in behavior.

2. Cognitivism Perspective of Learning

“Learning is a relatively permanent change in a person’s knowledge or behavior due to experience. This definition has three components: (1) the duration of the change is long-term rather than short-term; (2) the locus of the change is the content and structure of knowledge in memory or the behavior of the learner; (3) the cause of the change is the learner’s experience in the environment rather than fatigue, motivation, drugs, physical condition, or physiological intervention.” (Mayer, 1982, p. 1040).

“Learning is a process that takes place inside a person’s head.” This process “enables organisms to modify their behavior fairly rapidly in a more or less permanent way.” (Gagne & Driscoll, 1988).

3. Constructivism Perspective of Learning

Jonassen & Land (2000, p.v) suggested that constructivists believe that learning is “willful, intentional, active, conscious, constructive practice that includes reciprocal intention-action-reflection activities.” Therefore, learning is “conscious activity guided by intentions and reflections.”

Driscoll (1994) suggested that many learning theories do share some basic assumptions about learning:

* Learning is a persisting change in human performance or performance potential.

* To be considered learning, a change in performance must come about as a result of the learner’s interaction with the environment. Learning requires experience. How these experiences are presumed to bring about learning distinguishes different learning theories.

The Ways That People Learn

A research study, How people learn: Bridging research and practice (Donovan, Bransford, & Pellegrino, eds., 1999)), summarized three key findings of how people learn based on robust research. The findings have significant implications for teaching and learning:

1. Students have preconceptions about how the world works before they come to the classroom. Research suggests learners start to make sense of the world at a very young age. Many research experiments show the persistence of preexisting understandings. Therefore, teaching has to integrate their preexisting knowledge in order to be effective.

2. Research that compared the performance of experts and novices on learning and transfer suggest that in order to develop competence, students must have deep understanding of the factual knowledge, understand the facts/ideas in the context of a conceptual framework, and organize knowledge in certain ways which can facilitate retrieval and application.

3. Research on performance of experts and research on metacognition also suggests that learners can be taught to define their learning goals and monitor their learning progress.

How to Facilitate Learning

A lot of research studies have been conducted to investigate the effects of different instructional strategies on human learning. The following sample research studies investigate the effects of illustrated text and animations on human learning and provide some general guidelines on using illustrated text and animations in teaching and learning process.

1. Levie and Lentz (1982) conducted a metanalysis using the instructional treatments developed by Dwyer which was presented in a text format or programmed booklet. All studies included in the metanalysis included a text-only condition. Based on 41 comparisons of treatments with text plus prose vs. with text only using four criterion measures (drawing test, identification test, terminology test, comprehension test), Levie and Lentz (1982) reported that 36 comparisons favored illustrated text and 4 favored text alone.

2. Park and Hopkins (1993) summarized 25 studies investigating the effects of dynamic versus static visual displays. Fourteen of the studies found significant effects for dynamic visual displays. The findings suggest that dynamic visuals are effective under some circumstances.

3. Rieber (1990) stated “The power of animation … comes from the potential for creating a wide assortment of practice strategies. In 1989, Hannafin and Rieber (1989a, 1989b) conducted two research studies to compare a traditional questioning activity to an activity involving student control of an interactive dynamic in a structured simulation which taught Newton’s law of motion. When subjects were adults, the question practice group and interactive dynamic practice group performed equally well, however, the interactive dynamic practice group required significantly less time to answer posttest questions. It suggested that the interactive dynamic practice “supported encoding and retrieval tasks better than traditional questioning” (Rieber, 1990, p.83).

Principles of Human Learning

Ten tested principles of human learning have been summarized below ("Ten Tested Principles," 2003):

1. Students cannot recall and apply knowledge unless they practice retrieval and use.

2. Better learning (more easily recalled and applied) results when we vary the conditions of learning.

3. When learners integrate knowledge from both verbal and visual representations, they can recall it and apply it with greater ease.

4. Prior knowledge or belief determines what students will learn.

5. What instructors and learners believe about knowledge acquisition (epistemology) influences what will be learned.

6. Experience is a poor teacher because corrective feedback is rare.

7. Lectures fail to promote understanding because understanding is an interpretive process in which students must be mentally involved.

8. Remembering is a creative process that influences what learners will and will not be able to recall and apply.

9. In learning, less is more. Trying to cover large amounts of material and information reduces understanding and recall. If we teach toward future use, we should focus on in-depth understanding of principles.

10. What learners do in a course will determine what they will learn, how well they can recall it, and the conditions under which they can use it.

Learning Styles

"Information about learning styles can serve as a guide to the design of learning experiences that either match, or mismatch, students' style" ("Why is learning style important," no date).

Information about students' learning style is important to both the instructors and the students because:

* Instructors need to understand their students' learning styles in order to adapt their teaching methods accordingly.
* Students who knows their own learning style become better learners.
* Instructors will better understand the differences among the students.
* If an instructor's learning style differs from that of many of his or her students, the instructor may need to make adjustments in how material is presented.

Fleming and Mills (1992) suggested four categories that seemed to reflect the experiences of their students.

Visual

This preference includes the depiction of information in charts, graphs, flow charts, and all the symbolic arrows, circles, hierarchies and other devices that instructors use to represent what could have been presented in words.

Aural / Auditory

This perceptual mode describes a preference for information that is "heard." Students with this modality report that they learn best from lectures, tutorials, tapes, group discussion, speaking, web chat, talking things through.

Read/write

This preference is for information displayed as words. Not surprisingly, many academics have a strong preference for this modality. This preference emphasizes text-based input and output – reading and writing in all its forms.

Kinesthetic

By definition, this modality refers to the "perceptual preference related to the use of experience and practice (simulated or real)." Although such an experience may invoke other modalities, the key is that the student is connected to reality, "either through experience, example, practice or simulation."

Types of Learners

Summary

Active and reflective
Sequential and global
Sensing and intuitive
Visual and verbal

Active and reflective learners

* Active learners tend to retain and understand information best by doing something active with it—discussing or applying it or explaining it to others. Reflective learners prefer to think about it quietly first.

* "Let's try it out and see how it works" is an active learner's phrase; "Let's think it through first" is the reflective learner's response.

* Active learners tend to like group work more than reflective learners, who prefer working alone.

* Sitting through lectures without getting to do anything physical but take notes is hard for both learning types, but particularly hard for active learners

Sequential and global learners

* Sequential learners tend to gain understanding in linear steps, with each step following logically from the previous one. Global learners tend to learn in large jumps, absorbing material almost randomly without seeing connections, and then suddenly "getting it."

* Sequential learners tend to follow logical stepwise paths in finding solutions; global learners may be able to solve complex problems quickly or put things together in novel ways once they have grasped the big picture, but they may have difficulty explaining how they did it.

Sensing and intuitive learners

* Sensing learners tend to like learning facts; intuitive learners often prefer discovering possibilities and relationships.

* Sensors often like solving problems by well-established methods and dislike complications and surprises; intuitors like innovation and dislike repetition. Sensors are more likely than intuitors to resent being tested on material that has not been explicitly covered in class.

* Sensors tend to be patient with details and good at memorizing facts and doing hands-on (laboratory) work; intuitors may be better at grasping new concepts and are often more comfortable than sensors with abstractions and mathematical formulations.

* Sensors tend to be more practical and careful than intuitors; intuitors tend to work faster and to be more innovative than sensors.

* Sensors don't like courses that have no apparent connection to the real world; intuitors don't like "plug-and-chug" courses that involve a lot of memorization and routine calculations.

Visual and verbal learners

* Visual learners remember best what they see—pictures, diagrams, flow charts, time lines, films, and demonstrations.

* Verbal learners get more out of words—written and spoken explanations. Everyone learns more when information is presented both visually and verbally.

Meyers-Briggs Type Indicator

MBTI assigns four personality dimensions to individuals depending on how the perceive and interact with their environment.

Summary

Introvert vs. Extrovert
Thinking vs. Feeling
Judging vs. Perceptive
Sensing vs. Intuition

Introvert vs. Extrovert

Introverts tend to prefer to focus on inward thoughts and feelings and may prefer a quiet environment for learning and to listen rather than talk in class.

Extraverts often prefer to talk aloud and are more comfortable interacting with others. These learners may prefer collaborative learning, thinking aloud and/or class discussion.

Faculty vs. Students (Brightman, no date):

"The majority of undergraduate students are extraverts. Based on data from the Center for Applied Psychological Type (CAPT) between 56% and 58% of over 16,000 freshman students at three state universities were extraverts. Interestingly, over 83% of college student leaders were extraverts, while over 65% of honors students were introverts. Our own data base indicates that over 65% of business students are extraverts...The majority of university faculty are introverts."

Thinking vs. Feeling

Thinking students tend to prefer to use objective, impersonal facts to make decisions and form opinions. Thinking students may be more comfortable with personal conflicts than other students. Thinking students may prefer concrete language and working directly with data.

Feeling students tend to focus on emotions and personal values when making decisions and forming opinions and tend to value group harmony. Because students may form opinions based on emotional reactions or vague intuitions, them may need coaching to generate precise commentary or analysis.

Gender Differences and Student vs. Faculty (Brightman, no date):

"CAPT reports that on this dimension, the proportion of males and females differ. About 64% of all males have a preference for thinking, while only about 34% of all females have a preference for thinking...The majority of university faculty have a preference for thinking. CAPT reported that almost 54% of 2,282 faculty are thinking. Seventy percent of business faculty have a preference for thinking."

Judging vs. Perceptive

Judging students tend to prefer to make immediate decisions based on initial input and may be considered "decisive". A danger for these students is to make a premature conclusion before examining all the data.

Perceptive students may not make decisions until they process all the data and may be considered "indecisive" or "wandering" (as they begin more tasks). A danger for these students is procrastination as they collect more data.

Student vs. Faculty (Brightman, no date):

"The majority of undergraduate students are judging students. Based on data from the Center for Applied Psychological Type (CAPT) between 46% and 60% of over 16,000 freshmen at three state universities were judging students...The majority of university faculty also have a preference for judging."

Sensing vs. Intuition

These students prefer to focus on established facts, known procedures and linear presentations. These students tend to have stronger skills in memorizing details. However, concept maps may be recommended to help these students understand the "big picture."

These learners may see connections between seemingly random sets of data, but may not be as strong in remembering details. These students may prefer to see the entire framework first and fill in the details later.

Faculty vs. Students (Brightman, no date):

"The majority of undergraduate students are sensing students. Based on data from the Center for Applied Psychological Type (CAPT) between 56% and 72% of over 16,000 freshmen at three state universities were sensing students. Interestingly, almost 83% of national merit scholarship finalists and 92% of Rhodes Scholars were intuitive students. Our own data base indicates that over 65% of business majors are sensing students.....The majority of university faculty are intuitive. CAPT reported that almost 64% of 2,282 faculty are intuitive."

Multiple Intelligences

Howard Gardner's Multiple Intelligence Theory: different "ways of knowing".

Traditionally, instructional methods tend to favor verbal-linguistic and logical-mathematical intelligences, and don't focus on the arts, self-awareness, communication and physical education.

By employing role-playing, musical performance, cooperative learning, reflection, visualization, story telling, etc. as well as assessment methods that account for the diversity of intelligences, the learning experience can be richer for all students.

Summary of Types

Verbal/Linguistic -- Word Player
Logical/Mathematical -- Questioner
Visual/Spatial -- Visualizer
Musical/Rhythmic -- Music Lover
Body/Kinesthetic -- Mover
Interpersonal -- Socializer
Intrapersonal -- Individual

TEMPLATE
Type of Learner
-- Definition
Likes to
Is Good At
Is Best At

Verbal/Linguistic -- Word Player
-- the ability to use words and language

Read Write Tell Stories
Memorizing Names,places, trivia, dates
Saying, hearing and seeing words

Logical/Mathematical -- Questioner
- the capacity for inductive and deductive thinking and reasoning, as well as the use of numbers and the recognition of abstract patterns

Experiments Works with numbers Explores patterns
Math, logic, reasoning, & problem solving
Categorizing, classifying, & working with abstract patterns

Visual/Spatial -- Visualizer
-- the ability to visualize objects and spatial dimensions and create internal images and pictures

Draw, build, design. Create. Watch Movies. Play with machines
Imagination, Sensing Changes, Mazes & Puzzles, Map reading
Visualizing Dreaming Working with pictures

Musical/Rhythmic -- Music Lover
-- The ability to recognize tonal patterns and sounds, as well as a sensitivity to rhythms and beats

Sing, hum Play instruments Listen to music Respond to music
Picking up sounds Noticing rhythms Keeping time Melodies
Rhythm Melody Music

Body/Kinesthetic -- Mover
-- The wisdom of the body and the ability to control physical motion

Move around Touch and talk Use body language
Physical activities
Moving around Interacting with space Touching

Interpersonal -- Socializer
-- The capacity for person-to-person communications and relationships

Lots of friends Talk to people Join groups
Understanding people. Leading others. Organizing. Communicating
Sharing Comparing Cooperating Interviewing

Intrapersonal -- Individual
-- The spiritual, inner states of being, self-reflection, and awareness

Work along Pursue own interests
Understanding self. Following instincts. Originality. Pursuing goals
Working alone. Self-paced. Individual projects. Having own space.

TALKERS AND LISTENERS

When running seminar or discussion classes for undergraduates, the major issue instructors face is unbalanced participation, with some students dominating the discussion while others remain silent. While there are ways to force more widespread participation (such as calling upon people, basing grades on participation, using tokens, allowing people to speak only a limited number of times, etc.), all these techniques involve coercion to a greater or lesser degree. They run counter to the basic idea of the seminar/discussion as a continuing conversation, similar to the ones that one might have with friends and neighbors. One cannot imagine using coercion there. <http://www.ntlf.com/html/ti/images/v13n2a.jpg>

No Coercion

Since my own teaching philosophy has evolved to the point where I believe that the best learning occurs under conditions that aren't coercive, I tried a promising experiment this semester that focused on improving discussion without coercion. The course was on the "Evolution of Scientific Ideas." The class was comprised of 17 sophomore students. At the beginning of the very first meeting, after brief introductions all around, I spoke for a few minutes, saying that the class would function best if everyone participated in the discussions. Of course, all instructors say this, and it usually has little effect.

But then I said that in semi-formal groups such as this, each one of us had, over time, developed a preferred, or at least customary, role. We saw ourselves as either "talkers" (people who volunteered to speak and did so frequently) or "listeners" (people who preferred to stay silent and rarely, if ever, joined in the discussion unasked). I asked each person to self-identify, with me beginning and identifying myself as a talker. (This should be no surprise. McKeachie reports that the most common cause of unbalanced discussion is the instructor who typically talks about 70-80% of the time!)

Which Are You?

Six students identified themselves as talkers, while eleven said they were listeners. I then said that both talking and listening were essential skills and that we needed to develop both aspects of our personalities. I then asked all the talkers to sit together in one part of the room, the listeners to group in another part, and to discuss amongst themselves the following questions: What made me become a talker (listener)? How can I develop my listening (talking) skills? How can I help listeners (talkers) talk (listen) more?

The two groups spent about 20 minutes discussing these questions. The talkers group (which I naturally joined), although half the size of the listeners, made much more noise, talking and laughing as they discussed, with people jumping in

with ideas and comments. The listeners group was much quieter, with only one person speaking at a time, but even there the conversation never died down. The two groups then reported to each other at the end of the time period.

Listener Characteristics

The listeners said they listened and did not talk much because they felt that their ideas must already be obvious to everyone; that there was usually no pause in the discussion for them to insert their ideas; they liked to take in information; they took time to formulate their ideas and by the time that happened the discussion had moved on to something else; they did not feel themselves to be experts and did not want to waste other people's time with their unformed or poorly articulated views. To overcome these feelings, they felt that they should force themselves to talk more.

Talker Characteristics

On the other hand, the talkers said that they felt compelled to share whatever ideas they had; that they thought their ideas were good; felt compelled to correct ideas they believed were wrong; were uncomfortable with silence and felt obligated to break it; and sometimes felt they would explode if they kept silent. They also said that this behavior had developed over years as they realized that they liked the attention talkers received, they were noticed in class by teachers and hence did better, and were often expected by teachers to respond to questions. To overcome this, they felt they should force themselves to listen.

An important realization by the listeners was that the talkers did not need to think their ideas had to be very original or carefully phrased before they expressed them. Talkers said they often thought things through while they talked, rather than before. Listeners realized that their own ideas were not inferior to those of the talkers. In their private journals to me for that first week, students said they were totally surprised by the exercise, but that they enjoyed it because they had never before thought carefully about why they adopted their particular roles.

The whole class felt that we should try and create the conditions under which everyone got to participate. It was agreed that this responsibility should be shared and that the instructor should not have to play the role of arbitrator or be the focal point of the discussion. The class as a whole would try to develop good seminaring skills as we went along, monitoring the discussions so that they were not dominated by a few people.

Silent Running

I was apprehensive as to how this early discussion would influence subsequent classes. The next few classes were not promising, with low levels of participation and discussion. But what I then learned from their journals was that a few of the talkers (who are the kinds of students who keep discussions going) had decided to take a vow of complete silence in order not to dominate the discussions and to allow space for the listeners! They said they felt discouraged that the listeners had not immediately picked up the slack. I replied that they had to be patient, and that it is much harder for a listener to talk than for a talker to decide to listen. I suggested that they strive for a balance between domination and silence.

Conversation

The discussions got much better as the semester progressed, with the distinction between the talkers and listeners getting blurred but not eliminated. Almost all the listeners seemed to feel much more at ease in speaking and one or two of them even started talking to such an extent that they were accused (in good humor) of having "crossed over" to the talkers.

In a review discussion at the end of the semester, students said that this initial discussion had had a major impact on how they viewed their role in the seminar. It had made them more self-reflective and conscious of how their actions influenced that of others. They wished that it would be done in other classes as well.

CLASSIFICATION

Place the following items into groups. Describe each group.

tree	book	table	star
house	cup	fire	shirt
car	horse	foot	bird
string	door	heart	idea
water	road	fork	ball

TEACHING FOR INNOVATION

TOPIC 3. TEACHING STYLES AND METHODS

TP: New Technologies in Teaching and Learning: Evolution of Lectures

TP: Powerpoint Debate

Teaching Large Classes: Strategies for Improving Student Learning

Activity Breaks: A Push for Participation

TP: Problem Solving Through Design

TP: Asking the Right Questions in Class

TP: Keeping Discussion Going Though Questioning, Listening, Responding

TP: Tactics for Effective Questioning

NEW TECHNOLOGIES IN TEACHING AND LEARNING: EVOLUTION OF LECTURES

By Charles Kerns

As a learning activity designer, I explore how new technologies are likely to change specific teaching and learning problems and practices. For this article, I shall examine, in detail, one instructional practice: the lecture. It is important to look at the possibilities for change in the lecture because this mode of teaching is still the dominant practice in higher education. I do not mean to suggest that the traditional lecture will disappear, but that new models for oral presentations by instructors are appearing and are following normal innovation-adoption patterns.

The lecture has already been affected by technology, of course. During the past twenty-five years, the lecture was extended into distributed learning through analog video recordings. Given the opportunity, many students choose to view videos rather than attend lectures, even when doing so involves inconvenient visits to an audio-video center. Once recorded lectures are made available, it is difficult to constrain use only to certain students. Some students register for online courses while living on campus, simply to gain access to the recordings. In addition, faculty want to make lecture recordings available to all students, local or distant, for makeup and review.

Many technologies including streaming video, widespread high-bandwidth networks, recording whiteboards and rooms, automated indexing of audio and video, IP-based videoconferencing, and new types of computer-supported collaborative learning (CSCL) tools can affect how lectures will be given. With tools to digitize, index, summarize, link, and annotate video, we can create and distribute streaming-video recordings of lectures, including the slides and whiteboards that were presented. Handouts, alternative illustrations, animations, references, problem sets, and assessments can be indexed and tied to points in the audio-video recording. These clusters of resources and activities can be used as independent modules or learning objects, in some cases replacing the event of the lecture. Indexed recordings allow students to access specific moments in the lecture. Once the lecture recording has nonlinear access, students will move from sequential viewing (as must be done in the face-to-face lecture) to a combination of sequential (with and without pausing) and search-and-review viewings.

Another change is that online, lecture-based learning objects will be used with communication tools for discussion and annotation. New systems allow moments in the video to be annotated with students' questions, novice and expert explanations, drawings, and other representations of the content. Excerpts from lectures can be pasted into students' Web page projects and papers to elaborate on the original content. The students' works can then be linked back to the original learning object. Eventually, the recorded lecture can lose its

centrality in the learning object. The lecture thus evolves from a single event to a mediated,,"chunked" learning object to a dynamic set of resources. It evolves from a performance to an annotated recording of the performance to a new type of dynamic text.

Because of these possibilities, it is difficult to predict exactly how learning objects that contain lectures will be used by students. We do know that students do not like most lectures. Students often feel isolated, distant, and passive in the large lecture halls. They have trouble dealing with the continuous flow of information. With online lecture modules, students are able to decide when to"go" to a lecture, with whom to go, where to see it, and what to do while viewing it. With shared, network access, lectures can become distributed, informal group events (as homework has become for high school students with telephones and chat rooms). In both local and distributed informal study groups, students will dissect, review, and question the information in the lecture. Research has shown that for learning, facilitated group viewings of recorded lectures, both co-located and distributed, have been as effective as or more effective than simply attending lectures.1

Faculty, administrators, and academic technologists should support collaborative viewing. Planners and designers should be aware that students' study of lecture learning objects will lead to new types of behaviors determined by temporal constraints, learning styles, social supports, and other variables. Faculty need to monitor these new practices to identify those that are effective in helping students gain deep understanding. Academic computing groups should provide logistical and technical support for interaction, not simply distribute digital video recordings, in order to encourage the evolving collaborative learning practices.

The face-to-face lecture event, in which people physically meet, is an impetus to informal interactions: asking questions of instructors and friends in the hallway before class, carrying out discussions with other students, and developing trust and supportive friendships that start with the camaraderie resulting from facing common challenges. If students study from lecture-based learning objects, they will still need these informal interactions. CSCL tools that support casual discussion, trust building, and awareness are currently being researched. Collaborative activities will likely become part of the lecture-viewing practice. Buddy lists and other methods of maintaining awareness in informal groups have already become popular on some campuses and in some distributed learning environments.

As in most mediated learning interactions, the instructor will lose some level of control over students' behavior when lecture-based learning objects are used. Attending face-to-face lectures several times weekly provides external discipline for the student. When students can schedule their viewing and

discussions of an online lecture, they will need more support in planning their time and in developing meta-learning skills.

Finally, what happens to faculty as the lecture changes from being an event to being part of a learning object? Many faculty like to give lectures. Others are driven by the economic necessities of large classes. Many feel that the presentation of a long, sustained, oral argument is an important form of academic discourse. Lectures often form the skeletons of future books. In any case, faculty have become experts in organizing and preparing the content of lectures. They have gone through an apprenticeship in lecturing. They create lectures with little outside assistance. They consider the lecture their own independent activity. When lectures are part of a complex, online learning object, instructors must rely on technicians, producers, and, often, instructional designers, programmers, and other support staff. Learning objects that include lectures can force the faculty into new relationships. Some faculty may create lectures as they always did and leave the production to others; some may become producers; some may act only as content consultants in production groups.

How will faculty integrate learning objects into their teaching? Rather than providing basic coverage of facts (which learning objects can provide), will lecture periods consist of more complex discussions and arguments? Will they be periods of remediation based on monitoring student interactions with learning objects? Will more guest lectures delivered over IP-based videoconferencing offer different viewpoints? Will there be fewer, but intellectually more stimulating, lectures? Or will faculty simply be assigned more students per course?

Other instructional practices: seminars, laboratories, tutorials, problem-based instruction, peer tutoring, can be analyzed similarly to the lecture. These analyses need to look for constellations of interlocking, mutually supportive technologies that affect practice by providing rich interactions, access, effective learning, and efficiency.

THE PERILS OF POWERPOINT

Thomas R. McDaniel, Converse College, and Kathryn N. McDaniel, Marietta College

College professors everywhere are incorporating PowerPoint presentations into their classroom lectures. Faculty often pressure their deans to make every classroom a "smart classroom," and those fuddy-duddy faculty too slow to embrace this quickly-emerging technology are considered Luddites, resisters to change, out of step with modern student expectations. Technology can be a boon to pedagogy, but it is not without its perils. Before jumping headlong into the rushing tide of PowerPoint presentations, consider these cautions and criticisms about this popular teaching tool:

1) It's Inflexible.

When you use PowerPoint to convey information and ideas, it limits not only the content you can convey, but also the pace at which you present. If a student has a question (which the format of PowerPoint discourages anyway), the presenter may lose the flow of the PowerPoint in trying to answer it. If the student's question requires a quick jump ahead to a later point, the presenter will (if the program will allow it) have to scroll through upcoming points to address it. This can lead to confusion and a sense of disorder for both the presenter and the students. If the presenter has included too much information on the slides, students may delay the presentation by insisting that they "haven't finished" copying everything down. If you find out that your audience has a different level of knowledge than you expected (for example, if they didn't do the reading they were supposed to), the presentation cannot easily be adapted to fit the new situation. What all of these "ifs" demonstrate is that there's insufficient flexibility in the presentation form to allow for any surprises-those wonderful "teachable moments" that energize a lesson.

2) It's Risky.

How many times have you seen a PowerPoint presentation where some technical difficulty
a) made it impossible to start the presentation on time?
b) caused the presenter to lose the presentation entirely and end up fumbling halfheartedly through the presentation?
c) made it difficult to change the "slides," making every transition a long or clumsy process?
d) created a problem with the sequencing of points such that the presenter lost his or her place?
e) all of the above?

Technology is a wonderful thing, but its use also opens up all kinds of possible delays and technical difficulties. The real trouble with PowerPoint technology

is that the presenter becomes too dependent on it and often cannot simply abandon the technological "enhancement" to perform the lesson anyway when technical difficulties arise, as they invariably do.

3) It's a Crutch.

PowerPoint often serves as a crutch that prevents academics from developing real teaching skills. This is particularly a problem for academics who have spent most of their training in relatively isolated activities (researching in labs and libraries and then writing up their research) and who often have introverted tendencies. Instead of having to develop a pedagogy that engages the class at some level, instead of having to learn to communicate ideas to the individuals within the class, the professor can spend hours laying out a PowerPoint presentation that resembles a scholarly publication more than a lesson and that presents the information in a way that stifles communication between teacher and students. This is "presentation," not teaching.

4) It's Boring.

One of our students talks about PowerPoint classes as a "Zone-Out Zone." Not only is it easy for students to zone out during a presentation, it's often actually difficult for them to stay focused and attentive. This occurs for several reasons. First, a PowerPoint presentation seems to signal to students that they will not be necessary for the next 50 minutes or so, that their presence is purely as an audience, and as a result many students automatically disengage even at the very outset. (Having the lights out provides a cue.)

Second, presenters often put all of the salient points of the presentation on the slides, bullet-pointed for clarity-and sometimes they even distribute a handout of the information on each slide. Why does a student need to listen to the presenter read through each of these, even if there is a longer explanation? The pacing seems to slow down painfully; the students never have to figure out for themselves what the key ideas or points are; they have become merely scribes, copying down information. No matter how many "cool graphics" you have, if they don't relate to the material (and are just "frills"), students will tune out everything of substance.

5) It's Style without Substance.

The stylish presentation that PowerPoint offers often occurs at the expense of substance. Instead of spending time researching and studying the content, the presenter spends hours figuring out how to have the bullet points "fly" in. Examples end up watered down because of technological limitations or the lack of an appropriate graphic. Complex ideas are reduced to bullet points and clever images which don't allow for nuance, multiple perspectives or definitions, or

points of contention. Excessive stylish features slow the pace of the lesson and reduce the amount of material that can be conveyed effectively.

Even the best PowerPoint presentation is impressive not because of the insights and ideas conveyed, but because of the skilled use of technology it represents. In thinking about whether or not a PowerPoint presentation was effective, people will often focus on the technologies used, the frills and graphics, the smoothness with which the technology functioned. This is the last thing you want students to be getting out of your lesson-that you, the teacher, are good with technology.

Like Any Tool . . .

While PowerPoint can be a great addition to a teacher's pedagogical repertoire, it is no magic bullet guaranteed to make professors better (and more impressive) teachers. Like any tool, it can be misused or abused, and when that happens teaching effectiveness may be undermined instead of enriched. Effective pedagogy means knowing the benefits of any given teaching tool. Those who know the "perils of PowerPoint" are most likely to avoid its pitfalls.

==========

A PRUDENT PERSPECTIVE ON "THE PERILS OF POWERPOINT®

William R. Hamilton and Melissa F. Beery

PowerPoint® presentations are becoming the standard method to aid in lectures and college discussions. PowerPoint® is also frequently seen in financial, educational, and other professional institutions. The authors William R. Hamilton and Melissa F. Beery would like to respond to the concerns presented in "The Perils of PowerPoint®" by Thomas R. McDaniel, and Kathryn M McDaniel. PowerPoint® presentations are reviewed in terms of the format, technology, and style.

1. "It's Inflexible."

PowerPoint® presentations are inherently flexible because each presenter is different and can adjust the PowerPoint® slides accordingly. For example, if a question is built into the slide, then the presenter can pause for a group discussionŠ.When a person's lecture is flexible and accommodating, then the PowerPoint® presentation is a reflection of the author. In fact, PowerPoint® is only intended to be a framework for presentation of information and content of the lecture. Teachable moments can be created within that framework. The presenter need not be a slave to PowerPoint®, in that PowerPoint® presentations are as flexible as the author wants them to be.

2. "It's Risky."

Risk can be countered with options available outside of the technology. PowerPoint® presenters have a plethora of backup plans available if they chose to use them. A whole system failure is not common but slides can be printed in advance and used in the event the technology doesn't work at all. File management can be an issue but a CD, flash card, or the presenter can email the presentation to himself can all be options. Presenters need to be aware that a backup method is the key to any technological or other types of "failures," and then adjustments can be made. For example, a presenter that relies on a prepared report in his briefcase but leaves that briefcase on the subway would be in the same position as the PowerPoint® presenter that is faced with a computer crash. Do the presentations continue in either case? That's up to the individual person, not the briefcase or PowerPoint®. Backup plans are available almost without limit.

3. "It's a Crutch."

In high school speech, some students will use note cards and try to hide their face behind a 3 X 5 card. Some of these students will outgrow their introverted tendencies, while others will not. The same goes for those who use PowerPoint® presentations. Regardless of the tool used, introverted tendencies will occur because of the individual. There are rare instances when a PowerPoint® presenter will try and use their technology as a barrier. However, barriers can be overcome by involving the audience. Some of the best presentations involve PowerPoint® presentations that are paused to allow group work and enhance discussion if the presenter wishes it to do so.

4. "It's Boring"

It is generally agreed that the more a student is "engaged," or the more senses are utilized when involved in a discussion, the more the student will retain the information. PowerPoint® enables students to "see" in addition to hear a presentation. A student who wants or needs to "Zone-Out" probably can make that happen even if they were electrocuted by the PowerPoint® presentation as they begin to drift off. Simple entertainment techniques can be added to break up the monotony of a lecture. To blame the technology for "The Zone" is like blaming paper for a poor lecture.

5. "It's Style without Substance."

If presenters wish to be successful to any degree, they must learn how to have a bit of style to enhance their presentation. Even a novice can manipulate PowerPoint® to his or her liking with minimal training and experience. On the other hand, PowerPoint® presentations can be tailored to fit even complex discussions. >From graduate students to senior professors, many have used the

PowerPoint® format to defend dissertations and present research. Styles will vary depending on the use of PowerPoint®, but the substance is in the content of the lecture.

Like many technological aids for the classroom, PowerPoint® presentations have their advantages and disadvantages. However, PowerPoint® is the correct tool to most likely to be used. It would be difficult to go back to using the chalk board where poor handwriting and broken chalk are almost always an issue. With PowerPoint®, the professor also doesn't have to spend his time with his back to the students writing everything on the board. Instead, he or she can engage the students when a group discussion question flies in and sparks the intellectual thought process.

==========

DEATH BY POWERPOINT*

It's a rare professor who hasn't been tempted in recent years to put his or her lecture notes on transparencies or PowerPoint. It takes some effort to create the slides, but once they're done, teaching is easy. The course material is nicely organized, attractively formatted, and easy to present, and revising and updating the notes each year is trivial. You can put handouts of the slides on the Web so the students have convenient access to them, and if the students bring copies to class and so don't have to take notes, you can cover the material efficiently and effectively and maybe even get to some of that vitally important stuff that's always omitted because the semester runs out.

Or so the theory goes. The reality is somewhat different. At lunch the other day, George Roberts-a faculty colleague and an outstanding teacher-talked about his experience with this teaching model. We asked him to write it down so we could pass it on to you, which he kindly did.

* * * "About five years ago, I co-taught the senior reaction engineering course with another faculty member. That professor used transparencies extensively, about 15 per class. He also handed out hard copies of the transparencies before class so that the students could use them to take notes.

"Up to that point, my own approach to teaching had been very different. I used transparencies very rarely (only for very complicated pictures that might be difficult to capture with freehand drawing on a chalkboard). I also interacted extensively with the class, since I didn't feel the need to cover a certain number of transparencies. However, in an effort to be consistent, I decided to try out the approach of the other faculty member. Therefore, from Day 1, I used transparencies (usually about 8 -10 per class), and I handed out hard copies of the transparencies that I planned to use, before class.

"After a few weeks, I noticed something that I had not seen previously (or since)-attendance at my class sessions was down, to perhaps as low as 50% of the class. (I don't take attendance, but a significant portion of the class was not coming.) I also noticed that my interaction with the class was down. I still posed questions to the class and used them to start discussions, and I still introduced short problems to be solved in class. However, I was reluctant to let discussions run, since I wanted to cover the transparencies that I had planned to cover.

"After a few more weeks of this approach, two students approached me after class and said, in effect, `Dr. Roberts, this class is boring. All we do is go over the transparencies, which you have already handed out. It's really easy to just tune out.' After my ego recovered, I asked whether they thought they would get more out of the class and be more engaged if I scrapped the transparencies and used the chalkboard instead. Both said `yes.' For the rest of the semester, I went back to the chalkboard (no transparencies in or before class), attendance went back up to traditional levels, the class became more interactive, and my teaching evaluations at the end of the semester were consistent with the ones that I had received previously. Ever since that experience, I have never been tempted to structure my teaching around transparencies or PowerPoint."

* * * The point of this column is not to trash transparencies and PowerPoint. We use PowerPoint all the time-in conference presentations and invited seminars, short courses, and teaching workshops. We rarely use pre-prepared visuals for teaching, however-well, hardly ever-and strongly advise against relying on them as your main method of instruction.

Most classes we've seen that were little more than 50- or 75-minute slide shows seemed ineffective. The instructors flashed rapid and (if it was PowerPoint) colorful sequences of equations and text and tables and charts, sometimes asked if the students had questions (they usually didn't), and sometimes asked questions themselves and got either no response or responses from the same two or three students. We saw few signs of any learning taking place, but did see things similar to what George saw. If the students didn't have copies of the slides in front of them, some would frantically take notes in a futile effort to keep up with the slides, and the others would just sit passively and not even try. It was worse if they had copies or if they knew that the slides would be posted on the Web, in which case most of the students who even bothered to show up would glance sporadically at the screen, read other things, or doze. We've heard the term "Death by PowerPoint" used to describe classes like that. The numerous students who stay away from them reason (usually correctly) that they have better things to do than watch someone drone through material they could just as easily read for themselves at a more convenient time and at their own pace.

This is not to say that PowerPoint slides, transparencies, video clips, and computer animations and simulations can't add value to a course. They can and they do, but they should only be used for things that can't be done better in other ways. Here are some suggested dos and don'ts.

* Do show slides containing text outlines or (better) graphic organizers that preview material to be covered in class and/or summarize what was covered and put it in a broader context. It's also fine to show main points on a slide and amplify them at the board, in discussion, and with in-class activities, although it may be just as easy and effective to put the main points on the board too.

* Do show pictures and schematics of things too difficult or complex to conveniently draw on the board (e.g., large flow charts, pictures of process equipment, or three-dimensional surface plots). Don't show simple diagrams that you could just as easily draw on the board and explain as you draw them.

* Do show real or simulated experiments and video clips, but only if they help illustrate or clarify important course concepts and only if they are readily available. It takes a huge amount of expertise and time to produce high-quality videos and animations, but it's becoming increasingly easy to find good materials at Web sites such as SMETE, NEEDS, Merlot, Global Campus, and World Lecture Hall. (You can find them all with Google.)

* Don't show complete sentences and paragraphs, large tables, and equation after equation. There is no way most students can absorb such dense material from brief visual exposures on slides. Instead, present the text and tables in handouts and work out the derivations on the board or-more effectively-put partial derivations on the handouts as well, showing the routine parts and leaving gaps where the difficult or tricky parts go to be filled in by the students working in small groups.

If there's an overriding message here, it is that doing too much of anything in a class is probably a mistake, whether it's non-stop lectures, non-stop slide shows, non-stop activities, or anything else that falls into a predictable pattern. If a teacher lectures for ten minutes, does a two-minute pair activity, lectures another ten minutes and does another two-minute pair activity, and so on for the entire semester, the class is likely to become almost as boring as a straight lecture class. The key is to mix things up: do some board work, conduct some activities of varying lengths and formats at varying intervals, and when appropriate, show transparencies or PowerPoint slides or video clips or whatever else you've got that addresses your learning objectives. If the students never know what's coming next, it will probably be an effective course.

Teaching Large Classes: Strategies for Improving Student Learning

Dr. Graham Gibbs
Centre for Higher Education Practice
Open University, UK
g.p.gibbs@oper.ac.uk

4/24/98
Stanford University
(notes by R. Reis)

What research has found?

*Student ratings:
* Students dislike large classes
* They sometimes like very large classes, for dysfunctional reasons:- they can hide- tests are easier
* Students with unsophisticated conceptions of learning like teachers who have unsophisticated conceptions of teaching

* Student performance (USA):
* Performance on introductory large classes not worse, nor is it on subsequent courses (that build on the introductory courses) EXCEPT
*where assessment taps higher level outcomes
* where subsequent course had higher level goals
* This result has helped universities get away with very large introductory courses - if you keep testing with cheap/dirty methods, you wont' catch this.

*Classroom studies:
* Pattern of interaction changes as class size grows.- top 3-4 students who participate in a class of 8, still participate at same rate as size grows. - the remaining minimal interaction is just spread over the remainder of the class
* Quantity, quality of interaction changes
- % of teacher talking increases as size grows
- Students questions & answers get shorter
- Cognitive level of Q & A drops -- start to just ask/deliver facts not ideas

* UK Quantitative Studies
(Note on methods: external examiners review exams and set standards, a system not easily available in the U.S.)
* Correlation between enrollment & marks (grades) is as high as 0.5. Worst affected is social science, then humanities, then technical/engineering
* Decline of 1% avg. marks for ea. add'l 12 students!
* 50% more likely to gain C or F when enrollment over 70, than under 20
* Lots of studies across different institutions, given same systems

* Negative correlation between amt. of teaching & learning (The more teaching you do, the harder it is for students to prioritize what's important
* Long-term outcome research shows a dependence on amount of interaction with teachers
* Teaching and learning-centered descriptions of a course
*"teaching-centered" schedule at Oxford Brooks:

Lectures 24 hrs
Labs 36 hrs
Problem classes 12 hrs
* Lots of teacher misunderstanding of how much time students have available, or how much they actually spend outside of class
*"learning-centered description for same course
Budget for 120 hours of total student effort in the course
Example
4 hrs lectures (teaching)
3 hrs workshops (teaching)
6 hrs seminars (teaching)
56 hrs fieldwork (learning)
10 hrs workshops to present fieldwork (teaching)
18 hrs preparing fieldwork notes (learning)
21 hrs preparing reports (learning)
6 hrs on resource paper
15 hrs on group report
120 hrs total, t:l ratio = 1: 4

*You can brief students on this & track it.
Understanding total student learning time is the key indicator of learning!

* What they did to improve situation at minimum cost
* Course requirement to complete 50 of 70 problem sets
* Peer assessment in six additional "lecture" sessions (Students' assessment was more personal & direct, but less correct)
* Grades on these problem sets didn't count!
* Lectures, problem sets, classes, exams unchanged
* Result - Average exam increased from 45% to 80%
Note: More examples given in the overheads.

* Why did it work?
* Peer group is more influential
* Doing the grading made them engage in the solution, not just the problem...they had to use the solution
* Experience with correcting problems gave them an inside perspective on how to look for problems in the solution
* More time spent on task (your peers will see it)

* SUMMARY
* Focus on learning activity, not teaching
* Generate learning hours up to your limit
* Use assessment to lever hours and focus
* Get students to do for themselves and for each other what you previously did for them
* Use social mechanisms for peer support and peer pressure
* See your course as an integrated whole

ACTIVITY BREAKS - A PUSH FOR PARTICIPATION

By Phillip Wankat and Frank Oreovicz

Active learning makes lectures a more powerful classroom technique.

You've surely heard about active learning, cooperative groups, personalized systems of instruction and problem-based learning. But you were probably taught through lectures. What is best? Is a well-presented lecture or one of these other techniques the best learning tool?

It depends on your goals. If all you want to do is transmit information and assess the results with a multiple- choice test, then lectures do the job. The only teaching methods that statistically show that students learn better are the closely related techniques of mastery learning and the personalized system of instruction. But how many practicing engineers do you know who are paid to take multiple-choice tests? As soon as higher-order skills (designing, problem solving, communicating, working with people) are included in the assessment, teaching methods involving active learning and cooperative groups show a significant increase in student learning.

Still, lecturing does have advantages. Quite simply, it doesn't rock the boat. The professor stays in control and only has to be 50 minutes ahead of the students. And since lectures are face-to-face, developing rapport can be easier, although this advantage is lost in large classes. If the lecture format enabled students to learn higher-order skills, it would be quite a good technique.

We don't have to completely abandon lectures to gain many of the advantages that active learning and cooperative groups offer. If lecture classes are interactive so that students are not passive for long periods of time, they can be good learning experiences.

Since the attention span of almost all students is between 10 and 20 minutes, you can expect to lose most of your students if you lecture for 50 minutes straight. Even professionals fall victim to the "my eyes glaze over" syndrome. Not only do students tune out once that "dead" period is reached, the energy level of the class also flags. The solution might be to structure a 50-minute class something like this: a mini-lecture including an introduction, an activity break, a second mini-lecture, an activity break and finally a third mini-lecture, including a wrap-up. The mini-lectures contain an introduction, a body and a closing, similar to a straight lecture except they are shorter.

Activity breaks should incorporate active learning and the formation of cooperative groups. Both techniques practically force students to become involved. They can be very simple, like turning to a peer and comparing lecture notes. Alternatively, ask the student groups to solve a short problem. If the

problem is part of the homework assignment, they will be more motivated to do it. Or use technology to involve your students, such as student response systems like "clickers" to obtain immediate responses to multiple choice questions. Clickers, which allow students to respond anonymously to a multiple-choice question and allow the professor to display the responses in real time, involve the students and give the professor immediate feedback on student learning. After answering the questions, you might allow students to compare their answers with one another and change them if necessary.

Ensuring that all of the courses in the curriculum are lecture/active learning classes is not sufficient-students still need laboratory, design and computer simulation courses. However, it will go a long way toward satisfying the conditions necessary to becoming an engineer.

Phillip Wankat is director of undergraduate degree programs in the department of engineering education and the Clifton L. Lovell Distinguished Professor of chemical engineering at Purdue University. Frank Oreovicz is an education communications specialist at Purdue's chemical engineering school. They can be reached by e-mail at purdue@asee.org.

PROBLEM SOLVING THROUGH DESIGN

To design is to solve problems. The author describes a model of problem solving through design that can be used to restructure courses, programs of study, or entire institutions.

Although problem-based learning (PBL) can be successful in individual classrooms, I am advocating a broader and more sweeping implementation of PBL that can extend across courses, degree programs, or even institutions. Specifically, I advocate the notion of PBL through design. In this article, I begin with a description of the relationship between design and problem solving. Next I offer an example of how I implemented "problem solving through design" across three graduate-level courses. Finally, I offer considerations for implementing a model of problem solving through design.

Connection Between Design and Problem Solving

Designing is a problem-solving process, and design problems are usually described as open-ended, unstructured, or "wicked" (Rittel, 1984). Whether designing something highly technical, like a computer-based flight simulator to train future pilots, or something far less technical, like a centerpiece flower arrangement for a formal table setting, we cannot design without inherently thinking and working in a problem-solving mode. Through both design and problem solving, we are focused on "changing existing situations into preferred ones" (Simon, 1996, p. 130).

Across disciplines, designers tend to share a common problem-solving process that is an open-ended analogue of the scientific method (Newell and Simon, 1972). Designers solve problems by employing a cyclical process of problem identification and analysis, research, and inquiry that leads to the ranking of design priorities, testing multiple solutions through prototyping, and evaluating the design artifacts against performance criteria (Davis, 1998). To conceive this cyclical process in slightly different terms, we can note that design typically flows through major stages: naming (identifying main issues in the problem), framing (establishing the limits of the problem), moving (taking an experimental design action), and reflecting (evaluating and criticizing the move and the frame). Schön (1991) notes that designers reflect on moves in three ways: by judging the desirability and consequences of the move, by examining the implications of a move in terms of conformity or vio!
lation of earlier moves, and by understanding new problems or potentials the move has created. Regardless of how we describe the process, the point is that designing, like problem solving, is based on systematic processes and situational "rules of thumb" (Perez, 1995) that should lead to purposeful and practical outcomes.

Example of Classroom That Uses Problem Solving Through Design

In a recent semester, I incorporated a problem-solving-through-design method into three graduate courses in instructional technology-an instructional design class, a software development class, and a project management class.

In the past, I had taught these courses using traditional approaches, including the use of in-class exercises based on decontextualized examples, readings from texts and journals, and final projects as a basis of evaluation. In these courses, collaboration was minimal. I recognized a huge limitation of this traditional approach. Because courses are removed from practical and authentic contexts, students come to see the content of courses as isolated stages of a process, not as integrated activities within a single process.

To "transform" these classes using a method of problem solving through design. I compiled several problem scenarios that included possibilities for real and simulated interaction with clients. I also designed a set of performance expectations that established major deadlines and described my ideas of various working relationship among the three classes. As I introduced the various problem scenarios to students at the beginning of the semester, I invited each class member to volunteer for problem scenarios that were personally appealing, although I monitored the process to ensure that at least one student from each class was on each design team. Once all students had volunteered for a team, I distributed the performance expectations document, members of each team collaboratively worked to devise processes of design that would result in suitable artifacts.

Because each team was autonomous, no single description of the events that semester could fully capture each team's approach to design. In general, members of the project management class were in charge of the various projects. The project managers worked with the clients to establish project goals and tasks. Members of the design class assisted project managers in completing a needs assessment and analysis. Members of the design class also developed a design plan that members of the project management class presented to the client for approval. After the clients approved the various design plans, members of the software development class produced prototypes based on the plan created by the design class. The prototypes were tested with target audiences. The project management class then produced an evaluation report and held a culminating meeting with the design team to reflect on the process and outcomes of the design project.

Because students were enrolled in three different courses that met on three different nights, communication within each team was a potential problem. Project managers maintained Web sites for each problem scenario. These Web sites allowed all team members to view work schedules, drafts of design plans, and prototypes. Team members could communicate with each other and the client

through e-mail. An important feature was that, using the Web sites as guides, each group, for the most part, was self-directed and self-sufficient.

I served as a consultant to the teams at various points of difficulty, as a "client" when quick decisions were necessary regarding project goals or vision, and as a team member when production problems arose. By the end of the semester, students had successfully completed seven projects, and students remarked that the process, while arduous, was also meaningful, fun, and afforded them opportunities to learn in ways that were different from those in traditional graduate classes.

Recommendations for Implementing Problem Solving Through Design

So far in this article, I have made a connection between design and problem solving. I also have described my attempts with implementing a problem-solving-through-design model across three higher education courses. In this section, I offer a vision of an environment that would fully support such a model. To implement a problem-solving-through-design approach, professors should reconceptualize curriculum as problems, places students in the role of designers, and reconfigure classrooms as design studios.

Curriculum as Problems. In a problem-solving-through-design model, professors cannot preestablish a curriculum. Even the idea of teaching design sensibilities as a topic in the curriculum is problematic because design is not an object of study; design is a mode of inquiry and exploration (Davis, Hawley, McMullan, and Spilka, 1997). Instead of a contrived curriculum presented through an artificial context, design tasks are supported by learning on demand, where learning goals emerge from the situation at hand. In other words, because design problems are ill structured, professors cannot determine a standard curriculum until students actively devise methods for addressing the design problem.

Although a predesigned curriculum is irrelevant in a problem-solving-through-design model, professors are necessary and vital to students' success. Professors serve as facilitators and share their expertise as experienced designers. Facilitators can help participants establish individual and small-group goals through the use of performance contracts (Rieber, 2000). The facilitator also can moderate evaluations, helping and encouraging learners to offer feedback to their peers. Most important, however, professors must serve as experienced designers by helping students formulate alternatives to solutions as students design.

Students as Designers. In a problem-solving-through-design model students become designers. Designers work collaboratively and use conversation, argumentation, and persuasion to achieve consensus about perspectives and actions that might be take to solve a design problem (Bucciarelli, 2001). Conflicting viewpoints are debated, and differences of opinion are negotiated. In this way, dialogue

transforms individual thinking, creating collective thought and socially constructed knowledge within the team (Sherry and Myers, 1998). To further a shared understanding of the problem to be solved, designers create representations to solidify their design ideas (Hedberg and Sims, 2001).

Beyond working collaboratively, designers tend to be self-organized both individually and within their collaborative groups (Thomas and Harri-Augstein, 1985). Designers accept responsibility for their own learning by identifying their own purposes, setting goals for learning, implementing learning strategies, and identifying appropriate resources and tools (Fiedler, 1999).

Classrooms as Studios. To organize and manage design activities, professors can create an environment that is more akin to a studio than to a traditional classroom. Design studios are common in fine arts, architecture, and other fields that emphasize design (Orey, Rieber, King, and Matzko, 2000). Studios provide a learning environment in which participants use design tools and processes to complete real-world, and often self-selected, projects.

First, a design studio supports the use of appropriate design tools to craft models, drawings, narratives, and other representations of solutions. In many situations, professors may find that design activities provide excellent opportunities for the integration of computers into the classroom (for example, D'Ignazio, 1989; Liu and Pedersen, 1998). In other contexts, a consideration of communication tools can facilitate good design. As I note in my problem-solving-through-design example, students were officially registered for different courses, so a Web site became a valuable tool for promoting organization among students, and electronic communication tools became important tools for fostering clear communication.

Second, design studios support the use of processes that assist students in the design task. In general, students work independently and within teams to design a viable product that will solve their problem. For many students who have experience as designers, the idea of reflection may be natural and innate. But professors should consider building into the studio environment processes that will promote reflection among students. Professors need to scaffold reflection through concrete activities. For example, designers often maintain sketchbooks and diaries to support reflection (Cheng, 2000; Webster, 2001).

Also, professors can use numerous evaluation processes in design studios. They can conduct informal "desk critiques" on a regular basis. These desk critiques serve to provide students with cursory feedback about their work products. More formally, design studios imply the use of "juried" presentations of works in progress. In juried presentations, groups summarize their processes and showcase their products to professors and students who are working on other design projects. Juries provide an opportunity for formative peer review. In studios, summative evaluation often comes in the form of portfolios or formal

presentations to faculty committees, other students, and possible even real-world clients.

Conclusions

>From students learning through the design and production of multimedia (Kahn and Taber Ullah, 1998) to students learning science by designing and testing solutions to problems (Harel and Papert, 1991), problem-solving-through-design tasks have become an effective model for teaching and learning. For students and professors, the use of design in the classroom presents new challenges and fundamentally alters their roles. In accepting the challenges of incorporating design into the classroom, professors create new learning experiences that are more appropriate for students rather than relying on tradition exercises or lectures from a textbook.

References

Bucciarelli, L. "Design Knowing and Learning: A Socially Mediated Activity." In C.M. Eastman, W.M. McCracken, and W.C. Newstetter (eds.) Designing Knowing and Learning: Cognition in Design Education. Amsterdam, N.Y,: Elsevier, 2001, pp. 297-314.

Cheng, N. "Web-based Teamwork in Design Education." Paper presented at SiGraDi 2000: 4th Iberoamerican Congress of Digital Graphics, Rio de Janiero, Sept. 2000.

Davis, M. "Making a Case for Design-Based Learning." Arts Education Policy Review, 1998, 100(2), 7-14.

Davis, M., Hawley, P., McMullan, B., and Spilka, G. Design as a Catalyst for Learning. Alexandria, Va.: Association for Supervision and Curriculum Development, 1997.

D'Ignazio, F. "The Multimedia Sandbox: Creating a Publishing Center for Students." Classroom Computer Learning, 1989, 10(2), 22-23, 26-29.

Fiedler, S.H.D. "The Studio Experience: Challenges and Opportunities for Self-Organized Learning." Department of Instructional Technology, University of Georgia. [http://itech1.coe.uga.edu/studio/fiedler.html]. 1999.

Harel, I., and Papert, S. (eds.). Constructionism. Norwood, N.J.: Ablex, 1991.

Hedberg, J., and Sims, R. "Speculations on Design Team Interactions." Journal of Interactive Learning Research, 2001, 12(2-3), 193-208.

Kahn, T.M., and Taber Ullah, L.N. Learning by Design: Integrating Technology into the Curriculum Through Student Multimedia Design Projects. Tucson: Zephyr Press, 1998.

Liu, M., and Pedersen, S. "The Effect of Being Hypermedia Designers on Elementary School Students' Motivation and Learning of Design Knowledge." Journal of Interactive Learning Research, 1998, 9(2), 155-182.

Newell, A., and Simon, H.A. Human Problem Solving. Englewood Cliffs, N.J.: Prentice Hall, 1972.

Orey, M., Riever, L., King, J., and Matzko, M. "The Studio: Curriculum Reform in an Instructional Technology Graduate Program." Paper presented at the annual meeting of the American Educational Research Association, New Orleans, Apr. 2000.
Perez, R. "Instructional Design Expertise: A Cognitive Model of Design." Instructional Science, 1995, 23(5-6), 321-349.
Rieber, L.P. "The Studio Experience: Educational Reform in Instructional Technology." In D.G. Brown (ed.), Best Practices in Computer Enhanced Teaching and Learning. Winston-Salem, N.C.: Wake Forest Press, 2000, pp. 195-196.
Rittel, H.W. "Second-Generation Design Methods." In N. Cross (ed.), Developments in Design Methodolofies. Chichester, U.K.: Wiley, 1984.
Schön, D. The Reflective Practitioner: How Professionals Think in Action. New York: Teachers College Press, 1991.
Sherry, L., and Myers, K.M. "The Dynamics of Collaborative Design." IEEE (Institute of Electrical and Electronics Engineers) Transactions on Professional Communication, 1998, 41(2), 123-139.
Simon, H.A. The Sciences of the Artificial. (3rd ed.) Cambridge, Mass.: MIT Press, 1996.
Thomas, L., and Harri-Augstein, S. Self-Organised Learning. London, U.K.: Routledge, 1985.
Webster, H. "The Design Diary: Promoting Reflective Practice in the Design Studio." Paper presented at the Architectural Education Exchange, Cardiff, U.K., Sept. 2001.

WAYNE A. NELSON is a professor of instructional technology and chair of the department of educational leadership at Southern Illinois University Edwardsville.

ASKING THE RIGHT QUESTIONS IN CLASS

"Effective Teaching in Agriculture and Life Sciences," Unit 2 Improving Presentation/Classroom Skills

Module D -Questioning

Teachers should be liberal in their use of questions while teaching. Numerous research studies have found a correlation between questioning and student learning. Questions serve a variety of purposes:

* They can be used to ascertain what students know prior to teaching
* They can be used to determine if students have learned what has been taught
* They can be used to gain attention
* They provide variation while teaching
* They can be directed at problem students to get the student back on task
* They cause students to think

Using questions while teaching is a desirable behavior.

Levels of Questions: Questions are typically divided into two levels: Higher Order and Lower Order. The higher order questions call for responses from students that require synthesis, analysis and evaluation. Lower order questions require students to provide answers that demonstrate basic knowledge and comprehension (see Unit 1 Developing Higher-Order Thinking Skills for a review of the levels of the cognitive domain). It is desirable to ask both higher order and lower order questions. Research finds professors tend to ask only lower order questions.

Types of Questions: There are several systems for classifying questions. One system classifies questions as convergent or divergent. Convergent questions have a single or limited number of correct answers. Convergent questions typically involve the recall of facts or application of knowledge to a specific situation. Examples of convergent questions are:

    What is the chemical formula for photosynthesis?
    What are signs of nitrogen deficiency in plants?
    Which breeds of livestock would be best adapted for South Texas?

In some classification schemes, convergent questions are called closed questions.

Divergent questions have many correct answers or even unknown answers. They are often used to get students to think or solve problems. Examples of Divergent questions are:

What do you think will happen to family farms over the next ten years If you were the Secretary of Agriculture, what three things would you do first?

For an agribusiness to be successful, what business principles must be followed?

Teachers typically asked convergent questions five times more often than they ask divergent questions. Both types of questions are valuable in the classroom. In some classification schemes, divergent questions are called open questions.

A probing question is one in which the teacher asks the student to provide additional information, clarify a response or justify an answer. Teachers should get into the practice of asking probing questions as this causes students to develop higher order thinking skills. Even if a student response to a question is correct, it is appropriate to follow-up with a probing question.

One teaching skill not discussed in Module A of this lesson is cueing. When a student is asked a question and cannot respond, it is ok to provide a hint or clue to help the student. This is called cueing.

Steps in Asking Questions

There is a correct way and incorrect way to ask a question. A novice teacher may throw out a question or two to the class, get no response, and then decide not to use questioning as a part of the teaching repertoire. The problem was in the way the question was asked. In using questions the following sequence is recommended:

1.Ask the question. The question should be clearly stated and correctly phrased. If all you get are blank looks after asking a question, it may be because the question is poorly worded. When teachers come up with questions on the spur of the moment, they may not be the greatest example of precise wording. It isn't a bad idea to write down 2-3 questions you might want to ask and place those in your lecture notes.

2.Pause. After the question is asked, the teacher should pause for several seconds. This allows time for students to formulate a response. The longer the pause, the better the response will be. Research shows the average pause time after a question is asked is eight-tenths of a second. This is inadequate. Research shows the quality of the response is improved if more

time is allowed for students to think after a question is asked.

3.Call on a student by name. There are two things that generally happen when a teacher asks a question, but doesn't call on a specific student to respond.

A. No one will respond. Broadcast questions such as "Does anybody know..." or "Who knows..." rarely invoke a response; especially early in the semester. After rapport has been established, a professor may be able to ask this type of question. A specific student should be called on to answer the question.

B. One or two students may dominate the class if no one is called on to respond. Every time a question is posed, the same couple of students will answer. This is not desirable.

There are some people who are reluctant to call on a student by name because they might embarrass the student if the student doesn't answer the question correctly. As long as the professor doesn't lambaste the student for not knowing the answer and makes it a habit to call on all students in the class as a matter of course instead of singling out a few, this isn't a major problem.

The reason the questioning process starts with stating the question instead of identifying a student to answer is because this will cause all the students to have to think of the answer. If the teacher calls on a student and then asks the question, the other students tend to relax.

4.Acknowledge the answer, probe or redirect the question. The manner in which the teacher reacts to a student response to a question depends up the time available and the goals trying to be accomplished. The simplest response is to say "That is correct" or "That isn't quite right" or something to that effect. The student response should beacknowledged but a master teacher will build upon the student response whether it correct or incorrect. A master teacher will probe further (Why do you believe that to be true? Are you sure? Why did you respond that way?) or redirect the question to another student (Do you agree? What do you think?). The question may be redirected to 3-4 other students. Even if the original response was correct, it is not a bad idea to bounce the same question off of several students. Probing and redirecting the question promotes a deeper level of understanding and thinking.

5.State the correct response. Before a question is left, the teacher should emphasize the correct answer.

Keeping Discussion Going Though Questioning, Listening, and Responding

We emphasize throughout this book that democratic discussion is open and fluid, building on the diverse experiences and interpretations of its participants. Although teachers have some responsibility for guiding the discussion, no one person controls its direction entirely. Consequently, good discussions are unpredictable and surprising. They reveal things about the discussants and the topic under examination that are illuminating and eye-opening. At the same time, however, because democratic discussions have a life of their own, they can falter and even expire quite unexpectedly.

Even when discussions gets off to a good start and seem to have momentum, a variety of circumstances can intervene to bring group talk to a grinding halt. Sometime the teacher or one or two students assume too dominant a role. Sometimes the question or issue to be discussed just isn't controversial enough. Often the pace seems too slow, or the process for exploring the question lacks variety. In other cases, the students may not be ready to explore a topic in a large group setting or for some reason have lost their enthusiasm for the subject. Although it is frequently difficult to pinpoint the reasons why attention is wandering or commitment to the subject is waning, action needs to be taken to reinvigorate the conversation when these things happen. Part of the secret of dealing with these situations lies in refusing to panic or to berate oneself for allowing things to get off track. Fortunately, it is often possible to revive discussion and regain the sense of "controlled spontaneity" (Welty, 1989, p.47) characteristic of good conversation.

This is not to say, however, that we regard discussion as a panacea for tuning bored, disinterested, or hostile students into enthusiastic advocates for learning. Neither do we believe that simply talking about problems leads inevitably to students' deciding to take action to address pressing social concerns. As we argued in Chapters One and Two, discussions, in general tend to increase motivation, promote engagement with difficult material, and give people appreciation for what they can learn from one another and for what can be accomplished as a group. But we want to acknowledge that we have both been responsible for classes where discussion failed miserably, inducing boredom, resentment, and confusion. We have no magic formula to guarantee success, just some ideas that have proved useful to rejuvenate conversations that seem to be stuck.

Sometimes a discussion can be considered successful even if the original intentions of the leader go unrealized. When participants learn that a problem is more complex than they had thought or when their appreciation for existing differences is deepened, these can be counted as significant accomplishments, even though they might be different from the teacher's anticipated outcomes. We can say unequivocally, however, that discussion fails when participants avoid similar dialogical encounters in the future or when they lose interest in the

topics under consideration. If part of the point is to keep conversation going, to stimulate people to keep talking in the future, then discussion that inhibit this desire must be regarded as counterproductive and miseducational.

The question remains, what conditions inhibit dialogue and what measures can be taken to overcome them? This chapter and the next will focus on a variety of ways to make discussion a process of continuous discovery and mutual enlightenment. Getting students to view problems more critically and creatively helps keep discussion fresh. How teachers maintain the pace of the discussion, how they use questioning and listening to engage students in probing subject matter, and how they group students for instruction all affect how the discussion proceeds and how motivated the students are to participate in similar discussions in the future.

Questioning

To reiterate, an important focus of democratic discussion should be on getting as many people as possible deeply engaged in the conversation. Whatever the teacher says and does should facilitate and promote this level of engagement. As a number of commentators have pointed out, at the heart of sustaining an emerging discussion are the skills of questioning, listening, and responding (Christensen, 1991a, 1991b, Jacobson, 1984; Welty, 19898). Of the three learning to question takes the most practice and skill (Freire, 1993; Bateman, 1990). Although it is certainly true that the kinds of questions one asks to begin a discussion set an important tone, it is equally true that subsequent questions asked by both the teacher and the students can provide a powerful impetus for sustaining discussion. Indeed, as Palmer (1998) has noted, how we ask questions can make the difference between a discussion that goes nowhere and one that turns into a "complex communal dialogue that bounces all around the room" (p. 134).

Types of Questions

Once the discussion is moving along, several kinds of questions are particularly helpful in maintaining momentum.

Questions That Ask for More Evidence These questions are asked when participants state an opinion that seems unconnected to what's already been said or that someone else in the group thinks is erroneous, unsupported, or unjustified. The question should be asked as a simple request for more information, not as a challenge to the speaker's intelligence. Here are some examples:

How do you know that? What data is that claim based on? What does the author say that supports your argument? Where did you find that view expressed in text? What evidence would you give to someone who doubted your interpretation?

Questions That Ask for Clarification Clarifying questions give speakers the chance to expand on their ideas so that they are understood by others in the group. They should be an invitation to convey one's meaning in the most complete sense possible. Here are some examples:

Can you put that another way? What's a good example of what you are talking about? What do you mean by that? Can you explain the term you just used? Could you give a different illustration of your point?

Open Questions Questions that are open-ended, particularly those beginning with how and why, are more likely to provoke the students; thinking and problem-solving abilities and make the fullest use of discussion's potential for expanding intellectual and emotional horizons. Of course, using open questions obliges the teacher to take such responses seriously and to keep the discussion genuinely unrestricted. It is neither fair nor appropriate to ask an open-ended question and then to hold students accountable for failing to furnish one's preferred response. As Van Ments (1990) says, "The experienced teacher will accept the answer given to an open questions and build on it" (p.78). That is, as we all know, easier said than done. Here are some examples of open questions:

Sauvage says that when facing moral crises, people who agonize don't act, and people who act don't agonize. What does he mean by this? (Follow-up question: Can you think of an example that is consistent with Sauvage's maxim and another that conflicts with it?)

Racism pervaded American society throughout the twentieth century. What are some signs that things are as bad as ever? What are other signs that racism has abated significantly?

Why do you think many people devoted their lives to education despite the often low pay and poor working conditions?

Linking or Extension Questions An effective discussion leader tries to create a dialogical community in which new insights emerge from prior contributions of group members. Linking or extension questions actively engage students in building on one another's responses to questions. Here are some examples of such question:

Is there any connection between what you've just said an d what Rajiv was saying a moment ago? How does your comment fit in with Neng's earlier comment? How does your observation relate to what the group decided last week? Does your idea challenge or support what we seem to be saying? How does that contribution add to what has already been said?

These kinds of questions tend to prompt student-to-student conversation and help students see that discussion is a collaborative enterprise in which th e wisdom

and experience of each participant contributes something important to the whole. Too often discussion degenerates into a gathering of isolated heads, each saying things that bear no relationship to other comments. The circular response exercise (see Chapter Four), which requires students to ground their comments in the words of the previous speakers, gives students practice in creating discussions that are developmental and cooperative. Skillfully employing linking questions can also help participants practice discussion as "a connected series of spoken ideas" (Leonard, 1991, p. 145).

Hypothetical Questions Hypothetical questions ask students to consider how changing the circumstances of a case might alter the outcome. They require students to draw on their knowledge and experience to come up with plausible scenarios. Because such questions encourage highly creative responses, they can sometime cause learners to veer off into unfamiliar and seeming tangential realms. But with a group that is reluctant to take risks or that typically answers in a perfunctory, routinized manner, the hypothetical question can provoke flights of fancy that can take a group to a new level of engagement and understanding, Here are some examples of hypothetical questions:

How might World War II have turned out if Hitler had not decided to attack the Soviet Union in 1941? What might have happened to the career of Orson Welled, in RKO Studios had not tampered with his second film, The Magnificent Ambersons? In the video we just saw, how might the discussion have been different if the leader had refrained from lecturing the group? If Shakespeare had intended Iago to be a tragic or m ore sympathetic figure, how might he have changed the narrative of Othello?

Cause-and-Effect Questions Questions that provoke students to explore cause-and-effect linkages are fundamental to developing critical thought. Questions that ask students to consider the relationship between class size and academic achievement or to consider why downtown parking fees double on days when there's a game at the stadium encourage them to investigate conventional wisdom. Asking the class-size question might prompt other questions concerning the discussion method itself, for example:

What is likely to be the effect of raising the average class size from twenty to thirty on the ability of learners to conduct interesting and engaging discussions? How might halving our class affect our discussion?

Summary and Synthesis Questions Finally, one of the most valuable types of questions that teachers can ask invites students to summarize or synthesize what has been thought and said. These questions call on participations to identify important ideas and think about them in ways that will aid recall. For instance, the following questions are usually appropriate and illuminating:

What are the one or two most important ideas that emerged from this discussion? What remains unresolved or contentious about this topic? What do you understand better as a result of today's discussion? Based on our discussion today, what do we need to talk about next time if we're to understand this issue better? What key word or concept best captures out discussion today?

By skillfully mixing all the different kinds of questions outlined in this chapter, teachers can alter the pace and direction of conversation, keeping students alert and engaged. Although good teachers prepare questions beforehand to ensure variety and movement, they also readily change their plans as the actual discussion proceeds, abandoning prepared questions and formulating new ones on the spot.

References

Welty, W. "Discussion Method Teaching." Change, 1989, 21(4), 41-49.

Christensen, C. "The Discussion Leader in Action: Questioning, Listen- ing, and Response." In C. Christensen, D. Garvin, and A. Sweet (eds.), Education for Judgment: The Artistry of Discussion Leadership. Boston: Harvard Business School Press, 1991a.

Christensen, C. "Every Student Teaches and Every Student Learns: The Reciprocal Gift of Discussion Teaching." In C. Christensen, D. Garvin, and A. Sweet (eds.) Education for Judgment: The Artistry of Discussion Leadership. Boston: Harvard Business School, 1991b.

Jacobson, R. "Asking Questions Is the Key Skill Needed for Discussion." Chronicle of Higher Education, July 25, 1984, p. 20.

Welty, W. "Discussion Method Teaching." Change, 1989, 21(4), 41-49.

Ferrier, B., Marrin, M., and Seidman, J. "Student Autonomy in Learning Medicine: Some Participants' Experiences." In D. Boud (ed.), Devel- oping Student Autonomy in Learning. New York: Nichols, 1988.

Baetman, W.L. Open to Question: The Art of Teaching and Learning by Inquiry. San Francisco: Jossey-Bass, 1990.

Palmer, P.J. The Courage to Teach: Exploring the Inner Landscape of a Teacher's Life. San Francisco: Jossey-Bass, 1998.

Van Ments, M. Active Talk: The Effective Use of Discussion in Learning. New York: St. Martin's Press, 1990.

Leonard, H. "With Open Ears: Listening and the Art of Discussion Lead- ing." In C. Christensen, D. Garvin, and A. Sweet (eds.), Education for Judgment: The artistry of Discussion Leadership. Boston: Harvard Business School Press, 1991.

TACTICS FOR EFFECTIVE QUESTIONING
B.G. Davis, Tools for Teaching, San Francisco, CA: Jossey-Bass Inc., Publishers, 1995, pp. 85-88

ASK ONE QUESTION AT A TIME. Sometimes, in an effort to generate a response, instructors attempt to clarify a question by rephrasing it. But often the rephrasing constitutes an entirely new question. Keep your questions brief and clear. Lon complex questions may lose the class. For example, "How is the theory of Jacques Lacan similar to Freud's?" rather than "How are Lacan and Freud alike?" Are they alike in their view of the unconscious? How about their approach to psychoanalysis?" (Sources: Hyman, 1982; "Successful Participation Strategies," 1987)

AVOID YES/NO QUESTIONS. Ask "why" or "how" questions that lead students to try to figure out things for themselves. Not "Is radon considered a pollutant?" but "Why is radon considered to be a pollutant?" You cannot get a discussion going if you ask questions that only require a one-syllable or short-phrase response.

POSE QUESTIONS THAT LACK A SINGLE RIGHT ANSWER. A history professor includes questions for which a number of hypotheses are equally plausible-for example, "Why did the birthrate rise in mid-eighteenth century England?" or "Why did Napoleon III agree to Carvour's plans? She emphasizes to students that the answers to these questions are matters of controversy or puzzlement to scholars and asks the class to generate their own hypotheses. She embellishes what the students suggest by adding historians' theories and by showing how different answers to the questions lead in very different directions. She concludes by stressing that the answer to the question remains unsolved.

ASK FOCUSED QUESTIONS. An overly broad question such as "What about the fall of the Berlin Wall?" can lead your class far off the topic. Instead ask, "How did the reunification f Germany affect European economic conditions?"

AVOID LEADING QUESTIONS. A question such as "Don't you all think that global warming is the most serious environmental hazard we face?" will not lead to a free-ranging discussion of threats to the environment. Similarly, avoid answering your own question: "Why can't we use the chi-square test here? It is because the cells are too small?"

AFTER YOU ASK A QUESTION, WAIT SILENTLY FOR AN ANSWER. Do not be afraid of silence. Be patient. Waiting is a signal that you want thoughtful participation. Count to yourself while your students are thinking; the silence rarely lasts more than ten seconds. If you communicate an air of expectation, usually someone will break the silence, even if only to say, "I don't understand the question." If a prolonged silence continues, ask your students what the silence means: "Gee, everyone has been quiet for a while- why?" or encourage students by saying, "It's not easy to be the first one to talk, is it?" Someone will jump in with a comment or response. Don't feel like you have to call on the first person who volunteers. You might want to wait until several hands have been raised to let the students know that replies do not have to be formulated quickly to be considered. Consider choosing the student who has spoken least. After the first student is finished, call on the other students who had raised their hands, even if their hands are down. (Sources: Kasulis, 1984; Lowman, 1984; Swift , Gooding and Swift, 1988)

SEARCH FOR CONSENSUS ON CORRECT RESPONSES. If one student immediately gives a correct response, follow up by asking others what they think. "Do you agree, Hadley?" is a good way to get students involved in the discussion.

ASK QUESTIONS THAT REQUIRE STUDENTS TO DEMONSTRATE THEIR UNDERSTANDING. Instead of "Do you understand?" or "Do you have any question about evaluation utilization?" ask, "What are the considerations to keep in mind when you want your evaluation results to be used?" Instead of "Do you understand this computer software?" ask, "How would we change the instructions if we wanted to sort numbers in ascending order rather than descending order. Instead of "Does everybody see how I got this answer?" ask, "Why did I substitute the value of the delta in this equation?" If you want to ask, "Do you have any questions?" rephrase it to "What questions do you have?" The latter implies that you expect questions and are encouraging students to ask them.

STRUCTURE YOUR QUESTIONS TO ENCOURAGE STUDENT-TO-STUDENT INTERACTION. "Sam, could you relate that to what Molly said earlier?" Be prepared to help Sam recall what Molly said. Students become more attentive when you ask questions that require them to respond to each other. (Source: Kasulis, 1984)

DRAW OUT RESERVED OR RELUCTANT STUDENTS. Sometimes a question disguised as an instructor's musings will encourage students who are hesitant to speak.

For example, instead of "What is the essence or thesis of John Dewey's work?" saying, "I wonder if it's accurate to describe John Dewey's work as learning by doing?" gives a student a chance to comment without feeling put on the spot.

USE QUESTIONS TO CHANGE THE TEMPO AND DIRECTION OF THE DISCUSSION. Kasulis (1984) identifies several ways to use questions.
Ö To lay out perspectives: "If you had to pick just one factorä" or "In a few words, name the most important reasonä" This form of questioning can also be used to cap talkative students.
Ö To move from abstract to concrete, or general to specific: "If you were to generalizeä" or "Can you give some specific examples?"
Ö To acknowledge good points made previously: "Sandra, would you tend to agree with Francisco on this point?"
Ö To elicit a summary or give closure: "Beth if you had to pick two themes that recurred most often today, what would they be?"

USE PROBING STRATEGIES. Probes are follow-up questions that focus students' attention on ideas or assumptions implicit in their first answer. Probes can ask for specifics, clarifications, consequences, elaborations, parallel examples, relationship to other issues, or explanations. Probes are important because they help students explore and express what they know even when they aren't sure they know it (Hyman, 1980). Here are some examples of probing from Goodwin, Sharp, Cloutier, and Diamond (1985, pp. 15-17):

Instructor: What are some ways we might solve the energy crisis?

Student: Peak-load pricing by utility companies.

Instructor: What assumptions are you making about consumer behavior when you suggest that solution?

Instructor: What does it mean to devalue the dollar?

Student: I'm not really sure, but doesn't it mean that, um, like say last year the dollar could buy a certain amount of goods and this year it could buy less-does that mean devalued?

Instructor: Well, let's talk a little bit about another concept, and
this is inflation. Does inflation affect the dollar in

that way?

Instructor: What is neurosis?

Students: [no response]

Instructor: What are the characteristics of a neurotic person?

Instructor: How far has the ball fallen after three seconds, Christi?

Student: I have no idea.

Instructor: Well, Christi, how would we measure distance?MOVE AROUND THE ROOM TO INCLUDE STUDENTS IN THE DISCUSSION. When a student asks a question, it is natural for an instructor to move toward that student without realizing that this tends to exclude other students. To draw others into the conversation, look at the student who is speaking, but move away from that student.

Hyman, R. T. Improving Discussion Leadership, New York: Teachers College Press, 1980
Goodwin, S.S., Sharp, G. W., Cloutier, E.F., and Diamond, N.A., Effective Classroom Questioning, Urbana: Office of Instructional Resources, University of Illinois, 1985.

==========

QUESTIONING STRATEGIES

Questions should play an important role in every classroom--both student questions and teacher questions. Teachers can create an active learning environment by encouraging students to ask and answer questions.

STUDENT QUESTIONS

* Make it easy for students to ask questions
* Make time for questions
* Wait for students to formulate questions
* Ask other students to answer
* Have students formulate questions prior to class

TEACHER QUESTIONS

* Plan some questions as you prepare

    * Ask clear, specific questions
    * Use vocabulary students can understand
    * Ask questions in an evenly-paced, easily identifiable order
    * Ask questions from all levels of Bloom's Taxonomy of Educational Objectives
    * Use questions to help students connect important concepts
    * Use questions to give you feedback
    * Allow sufficient time for students to answer
    * Rephrase questions

Teacher Questions
PLAN SOME QUESTIONS AS YOU PREPARE your lesson plan.

Consider your instructional goals and emphasize questions that reinforce them. The questions you ask will help students see what topics you consider important.

ASK CLEAR, SPECIFIC QUESTIONS that require more than a yes or no answer. Avoid ambiguous or vague questions such as "What did you think of the short story?"

If a student does give you a yes/no or short answer, ask a follow up question that will encourage him/her to expand, clarify, or justify the answer.

USE VOCABULARY THAT STUDENTS CAN UNDERSTAND. Students cannot respond well to a question that contains unfamiliar terms.

ASK QUESTIONS IN AN EVENLY-PACED, EASILY IDENTIFIABLE ORDER. Students might be confused by random, rapid-fire questions. Use questions to signal a change of topic or direction in the lecture.

ASK QUESTIONS FROM ALL LEVELS of Bloom's Taxonomy of Educational Objectives. Mixing more difficult questions that require synthesis and evaluation with simple questions that require memory and comprehension keeps students actively switching gears. For a more complete description of the major categories in the cognitive domain, see the section on testing.

USE QUESTIONS TO HELP STUDENTS CONNECT IMPORTANT CONCEPTS. (e.g., Now that we've learned about conservation of energy, how does this knowledge help us relate the kinetic and potential energy of an object?)

USE QUESTIONS TO GIVE YOU FEEDBACK on whether students have understood the material. (e.g., "Which part of the experiment was most difficult for you and why?").

ALLOW SUFFICIENT TIME FOR STUDENTS TO ANSWER your questions (10-15 seconds). Students need time to think and organize an answer before responding. Learn to wait until you get a student response. The silence can be uncomfortable

sometimes, but it is necessary in order for students to know that you are serious about wanting an answer to your question. You can ask students to write down their response to a question, then call on several students to read their answers. This technique requires all students to become actively involved in thinking about your question.

REPHRASE QUESTIONS when students do not respond in the manner you expected. Admit that your original question might have been confusing.

Student Questions
MAKE IT EASY FOR STUDENTS TO ASK QUESTIONS

a. Make your classroom risk-free for asking questions. Banish the phrase "stupid question" from your vocabulary. Let students know the first day that you want and expect questions.

b. Solicit questions by asking:
- "What aspects of this material are unclear?"
- "Can I give another example to help you understand this topic?"
- "Can anyone add some examples to mine to help clarify this material?"

MAKE TIME FOR QUESTIONS throughout your class. Do not leave the question time until the last 2 or 3 minutes. Students will assume that "Are there any questions?" is a signal for class to end.

WAIT FOR STUDENTS TO FORMULATE QUESTIONS. Be sure to allow pause time (10-15 seconds) for students to review their notes for areas that are unclear.

Again, you may ask students to write their question and then call on several students to read what they have written.

ASK OTHER STUDENTS TO ANSWER student questions. This will encourage a discussion among the class.

HAVE STUDENTS FORMULATE QUESTIONS PRIOR TO CLASS.
Anytime you assign reading, math problems, experiments, case studies, journal writing, etc., ask your students to prepare three questions they had while they were completing the assignment. Also, you might ask them to write three questions they would expect to answer on a quiz covering the material they encountered. Begin class by having your students share their questions in small groups or as a whole. Their questions not only will stimulate discussion but also will allow you to determine confusing aspects of the material. In addition, being able to anticipate questions a teacher will ask on exams is an important study skill for students to develop.

PUTTING A SECOND LIFE "METAVERSE" SKIN ON LEARNING MANAGEMENT SYSTEMS

Jeremy Kemp,
eCampus,
San Jose State University
jkemp@cemail.sjsu.edu
SL: Jeremy Kabumpo

Daniel Livingstone,
School of Computing
University of Paisley
daniel.livingstone@paisley.ac.uk
SL: Buddy Sprocket

Abstract

This paper outlines the advantages and weaknesses of Multi-User Virtual Environments for teaching and explores the possible benefits of integrating them closely with traditional Learning Management Systems. We present survey findings of teachers interested in using the Second Life MUVE for teaching. The teachers gave us their opinions about integrating SL and LMS in their classrooms. We finally propose technical methods for creating hybrid systems combining elements of both MUVE and traditional LMS systems for use in teaching. The hybrid system uses the Moodle open source system and Second Life's connectivity features to mirror web-based classrooms with in-world learning spaces and interactive objects. We suggest that further work may help suggest the most suitable educational applications for these hybrid systems.

Introduction

Faculty who offer web-based instruction and resources have become very familiar with the likes of WebCT, Blackboard, Moodle and other Learning Management Systems, or LMS. Rather than wasting time learning the technical craft of Web design, they rely on templates and simple forms to create interactive web-based class environments.

These environments offer affordances beyond simple document repositories, by featuring discussion forums, online chatrooms, gradebooks and the ability to give automatically marked tests such as multiple choice questionnaires.

LMS often include a variety of means for communication between staff and students, but they are perhaps most commonly used as document repositories (Livingstone and Kemp 2006). This enables flexible access to course materials – on and off campus with the security of password-controlled access. More adept faculty employ the fuller range of communication tools including discussion forums, synchronous chat, assignment file drop-boxes, self scoring quizzes and grade books.

For the most part, the educational content is stored in static documents – copies of Powerpoint slides and Word documents. Assessment and interactive features are used more sparingly. It is clear that the full potential for interactive learning support is not being reached in the main. There is relatively little use of multi-media – and indeed these VLE's do not readily support the creation of multi-media content. But richer multi-media presentations supporting learning of 'hard' topics has long been known to have value in student learning (Laurillard, 1997).

Second Life overview

Teachers and university administrators are experimenting with a new form of virtual learning environment with some basic similarities to LMS but offering radically different affordances. The Second Life, SL, system by Linden Lab is a persistent 3D world, or "metaverse". Users access the online system with a proprietary client and interact with content and other "residents." Unique features include simple tools for constructing 3D objects and scripting tools for interactive content - including connectivity with external web-pages and internet resources. SL improves on its predecessors in several key ways.

First, the SL platform is completely free of a publisher-imposed narrative. Unlike thematic MMORPG games such as *World of Warcraft*, SL has no plotline or setting. Teachers have freedom to weave their own metaphors and build domain-specific settings in 3D environments. Currently, education designers in SL create all manner of classrooms, lecture halls and campus landmarks. For example, New York Law School created a "Democracy Island" complete with a Supreme Court building and miniature models of urban neighbourhoods. These cityscapes were proposed as a way to meet public review requirements for city planning (Democracy Design Workshop 2006).

Secondly, SL offers very simple tools for modifying or "modding" content. Users build items with a limited palette of primitive objects "prims" including cubes, spheres, cones, etc. Simple menus allow users to adjust the size of the objects and to map images on their surface. For-profit designers do a brisk business in virtual furniture and pre-fabricated structures such as one-room school houses, office desks, decorative seats and interactive bookshelves.

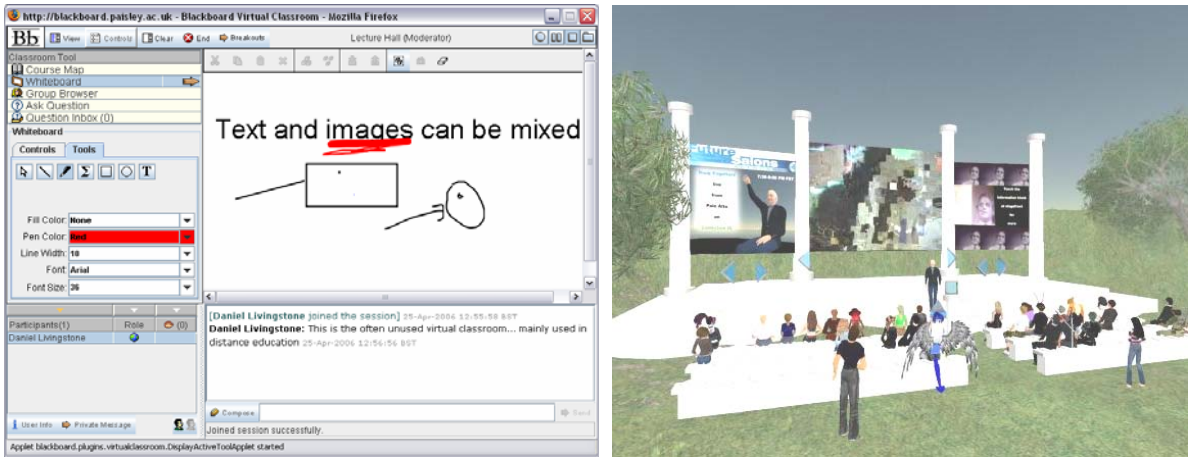


Figure 1. Interactive classroom settings in traditional LMS and Second Life.
Source: http://www.sluniverse.com/pics/pic.aspx?id=50270

Finally, amateur programmers create complex interactive applications in the proprietary Linden Scripting Language (LSL). They design objects that react intelligently to touch - making virtual "manipulatives" helpful for instruction (Resnick, 1998). For example, physics professor Anthony Crider at Elon University created a telescope trainer that teaches students the proper order for adjusting focus knobs on a real telescope (Crider, 2006). Objects respond in text chat to chatted commands allowing rudimentary teaching "agents" which answer questions and dispense domain content similarly to Harvard's River City MUVE project. (Dede, 2005). One object can even be programmed to move independently and control other items to create complex, multi-step building tools.

Objects can also send data to Web-based systems outside SL using the hyper-text transfer protocol (http). This data conduit is unique among all MUVE systems and opens immense opportunities for creating powerful connected learning applications.

While the features that already exist in LMS are not generally used to their fullest, they nicely fill in some of the current weaknesses of *SL* as a learning platform.

SL vs. LMS: Round 1

Many papers highlight benefits of learning within 3D worlds where students are embodied as avatars. For example, a review of two distance learning projects using Active Worlds is presented in Dickey (2005), concluding that the 3D immersive format has significant potential for "facilitating collaborations, community and experiential learning" and highlighting the situated embodied nature of the learning as a particular strength. A more speculative look at the future potential of 3D learning environments, albeit grounded in much prior practical experience, is presented in Dede (2004). Also see Antonacci & Modaress (2005).

As with the hypothetical example of Dede (2004), SL provides a sense of embodiment, yet one in which normal barriers between students and staff can be broken down as in Robbins (2006) concept of image slippage. Compared to other electronic tools for distance communication, there can be an improved sense of being 'there' in a classroom, rather than of being a disembodied observer, Figure 1.

Rich 3D demonstration models can be built in SL – leveraging the power of modern computers to allow students to *experience* phenomena of interest. The acknowledged power of multi-media to improve delivery of material over purely written means, (Laurillard, 1997), is worth exploiting – and SL makes this quite feasible, even for faculty with only modest scripting and modelling skills.

So, in terms of enhancing the experience of learning, it seems clear that SL should have some distinct advantages over traditional LMS. It also has some clear disadvantages.

SL vs. LMS: Round 2

If it is a weakness of LMS that they are often used only as document repositories, it is certainly the case that *MUVEs* including SL are very poor document repositories. The note cards used with SL are simple text documents which can support only very limited formatting. The documents which can be generated are essentially simple ASCII texts with embedded objects which require clicking on to view or open. Transferring documents between SL and desktop OS is also less straightforward than with LMS – generally requiring cut-and-paste.

SL developers have created PowerPoint-style presentations tools which require presenters to upload

each individual slide as a separate image – either to *Second Life* itself or to a web site such as *Flickr* (Metalab 2006).

Several other issues cause concern for the nascent community for educators. First, *SL* makes considerable hardware demands. The minimum technical requirements are beyond the capabilities of typical labs in most schools and colleges – particularly with regards to graphics cards. Some teachers must find secretly sympathetic technology administrators who accommodate their special needs (Delwiche 2003). This issue is exacerbated somewhat by a constant call for visual improvements from users with heightened expectations from the latest video game offerings. Linden Lab designers are tasked with serving an extremely heterogenous user base. Users range from game designers recreating traditional MMORPGs (Solvang 2006) to Barry Joseph's Global Kids (2006) youth program educating underserved communities.

Educators often raise the important topic of improving access for visually impaired students. Aside from the problems of navigating a 3D world, even the chat is inaccessible – the user-interface currently does not work with any screen-readers. For students with less severe visual-impairments, the ability to modify the user interface – to change colours and fonts to less stylish but more readable settings – would be a step in the right direction. Linden Lab promises to move toward a more flexible interface.

Disruptive players present another problem. For classes held in publicly accessible areas, these 'griefers' may interfere with classes and negatively impact the student experience such as paintballing the instructor (Kemp 2006). The virtual harm inflicted in many griefing incidents can cause very real distress (c.f. the well known incident reported in Dibbell, 1993).

Of these, only the issue of access for visually impaired students will concern users of LMS – and these students at least may rely on screen readers to some degree.

SL with LMS

Each platform offers complimantary affordances not available in the other. Connecting the two systems may allow instructional developers and teachers to explore exciting new opportunities for interaction on the Web and within the SL Multi-User Virtual Environment. It makes sense then to progress past the mindset of *SL* "vs." LMS, to the interconnection of the two - *SL* "with" LMS. We also want to avoid using *SL* as a weak rendition of LMS for document management or to continue using legacy Web learning systems by themselves with less interactivity and student engagement.

Survey Results

We recently completed a survey to better understand needs and desires for integrating both types of system for educators.

There are two distinct directions in which to progress this work. Moodle, or similar, can be modified to link or refer to SL. For example, using the Map API it might be possible to have links to SL locations, with maps, shown inside the LMS. LMS content generally allows HTML formatting, but not scripting, to be embedded in pages – thus a custom resource or similar would need to be developed.

Secondly, developers may put content, or links to LMS content, into *SL*.

We surveyed educators interested in using Second Life in their teaching to help determine whether these efforts would be worthwhile. To reach educators, a post was made to the Second Life Education mailing list and 27 educators responded. All respondents were able to exit the survey at any time or skip any question. A number of the questions were of general interest (showing, for example, that 80% of respondents had been active in SL for less than one year), while other questions were focused on questions relating to integrating SL and LMS. As it was possible to skip questions, for each of the findings we include details of the number of respondents that answered that particular question.

Asked which LMS they used, there was an equal split between Blackboard, WebCT, Moodle and 'Other', with 35% not using LMS at all (n=23).

Asked to compare aspects of SL and LMS environments, (n=16), 94% felt that SL was 'slightly better' or 'best' for synchronous chat, and 85% felt the same for live presentations or classes. Unsurprisingly, these opinions were reversed for features such as document storage, asynchronous discussion (e.g. forums) or grade-book support.

86% (n=22) thought integrating SL and LMS would be moderately, very or extremely useful. A final question asked what features of an integrated system would respondents find most useful, and allowed up to four choices to be selected (n=21). The most requested features, and number of times the feature was requested, were:

- Link to SL locations from inside LMS (e.g. SL Map API) (15)
- Broadcast LMS announcements in SL (13)

- Access assignment handouts from SL & LMS (13)
- Display text information from LMS in SL (13)
- Log of student time in SL sent to LMS (11)

Other requested features included linking live chat in SL and LMS, or allowing assignment submission in both, or accessing LMS forums from SL.

Sloodle

While the survey size was small, it was focussed very tightly on educators using – or planning to use – Second Life in their teaching. As such, we feel that the findings do illustrate genuine interest in SL/LMS integration, and provide motivation for designing and prototyping different integrated systems. The system we propose will integrate the Open Source Moodle LMS with SL, and which we call *Sloodle*.

Platform Layers

In thinking through the possible integration of these systems, it is helpful to consider them in the framework of "three tier" architecture (Wikipedia, 2006). Most modern Web-based teaching systems comprise three parts separated into the "layers" of data, logic and presentation.

The data layer includes passwords, pointers to assignment files, logs of interactions such as threaded messages and chat transcripts. It also includes guidelines for page designs and how static materials are arranged for viewing. LMS systems store this raw information in databases such as MySQL (Moodle) or Oracle (Blackboard Vista).

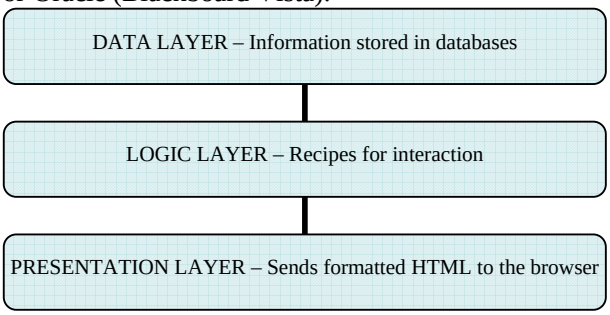


Figure 2. Typical three-tier architecture of an LMS

Logic is the second tier or layer in these systems. This layer implements interactive functions such as restricting access to materials, calculating grades, and multi-step operations such as quizzes and assignments. In the Moodle LMS system which we are currently working with, this layer is implemented using PHP. The final, presentation, layer delivers HTML code to the user combining images, static content and layout.

SL applications coded inside the environment may also be seen in this structure. Data is stored on notecards or chatted into the applications. For instance, museum owners set up "tour bot" agents that greet guests and take them on a pre-determined track with descriptions of the exhibits. The stopping points and text for the descriptions sit inside the "bots" as notcards. Logic is implemented using LSL, the presentation layer in 3D interactive objects.

Possibilities for interoperability

Now we take these three layers and see what areas lend themselves to interoperability. How will the two systems work together?

The logic layer for Moodle requires some minor adjustments to remove HTML formatting and to map the data onto the new interfaces offered in the SL environment. The SL logic layer mostly handles passthrough of data to the web-based database. Linden Lab limits access through this portal to a few times each minute so that real-time interaction is difficult. Thus, LSL scripting will be required to buffer data.

The presentation layer is the most interesting and holds the greatest potential for innovation. We think developers will be very active creating new ways to present previously web-delivered class information. Ubiquitous functions such as threaded messaging may be used in completely novel ways in this new setting where 3D metaphorical objects are generated automatically. Will artists create giant oak trees, each branch representing a thread of conversation? Or, as has often been the case, will fanciful interfaces be wittled down to bare-bones functionality, enabling students and their teachers to focus directly on the content being discussed?

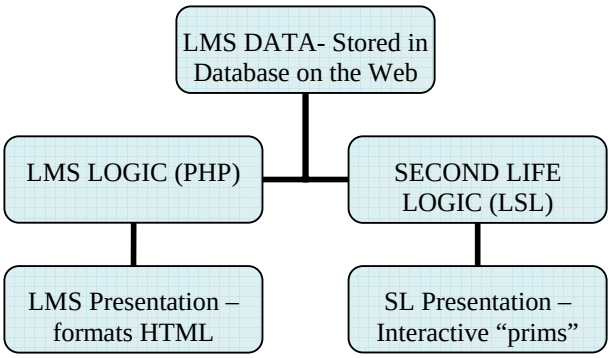


Figure 3. Three-tier architecture of a combined LMS-SL tool

Some features would only require changes to the LMS – such as adding resources which would allow the SL Map API (Second Life, 2006) to work inside Moodle. However, we would like to propose a set of tools to give access to Moodle resources from inside SL, and

to attempt to make effective and interesting use of the 3D space – otherwise why not simply open Moodle in a separate web-browser?

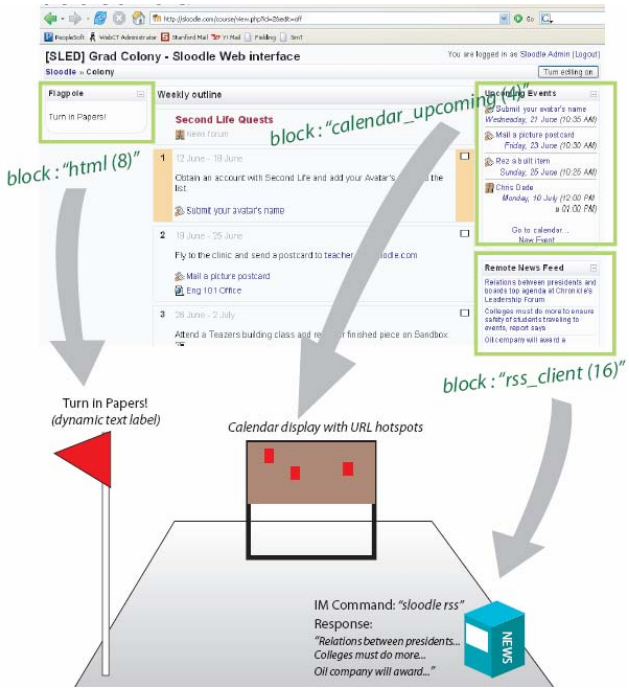


Figure 4. Sloodle will reflect the 2D page design in Moodle in a 3D 'office' space in SL

There are very many possible uses for this, but we propose a very simple example of this system at work. Our plan is that a standard-sized 512m^2 "office" in SL that reflects in 3D the Moodle page structure, Figure 4. This will be instantiated, or "rezzed", automatically based on blocks visible in the Moodle class. Each tool displayed in the Moodle class is re-created as interactive, metaphorical objects or "furnishings."

For example, notices in Moodle may be appear as flagpoles with text labels – providing clear visual cues to important new content. Calendar information may be rendered as a wall display, while real simple syndication "RSS" feeds appear in the form of radios or teletype machines. Interacting with any of these elements results in loading an appropriate URL or sending an IM text message to the user. Figure 5. shows three configurations of a Moodle class page along the top row and the corresponding SL office layouts below. The first column shows a calendar block on the left column and the flagpole on the opposite column announcing "Essays due now!" The reader board in SL shows the text included in the Moodle HTML block. The flagpole is down in the middle example, while the calendar and flagpole have shifted on the page and the RSS block is showing. The final column shows another flagpole announcement and the three blocks in their new positions.

Backend Functionality

The current prototype implementation uses "Sloodle distillers" loaded in PHP on the Moodle server. When the *Second Life* Sloodle objects are used, these use HTTP requests to PHP pages which then access the Moodle database. They output simplified, non HTML data that can be gathered by LSL scripts in-world.

It is hoped that as faculty re-arrange blocks in the Moodle shell, the office furnishings layout should change as well to mirror this. This repositioning might be either automatic or upon a "Sloodle reset" command chatted by the faculty member's avatar.

Conclusions

While previous work highlighted the distinct differences between SL and LMS, our subsequent investigations have identified a strong interest in

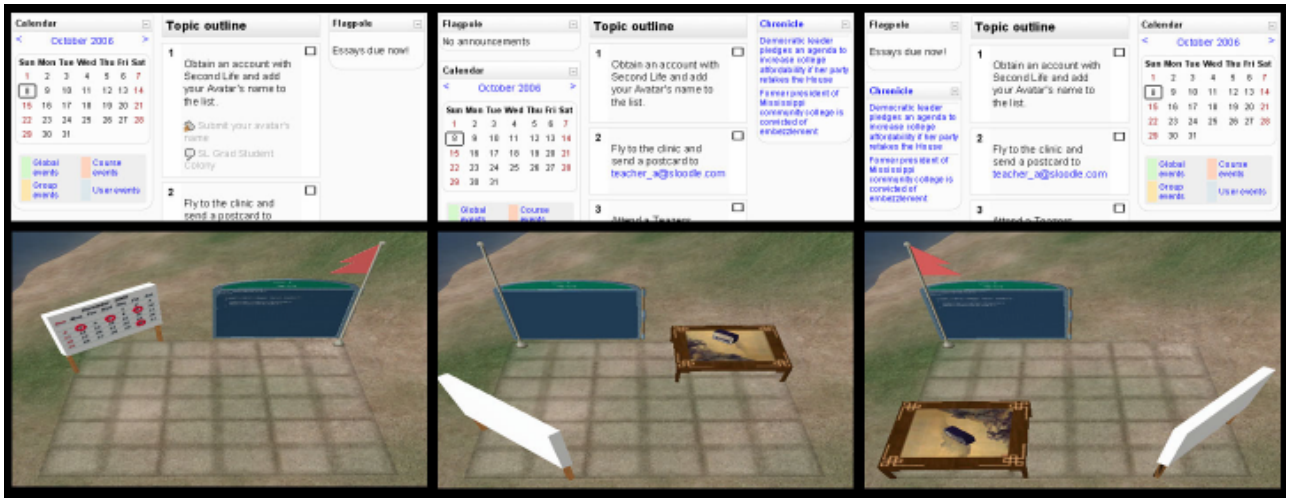


Figure 5. Moodle class page designs on the top row show calendar, flagpole (html) and RSS blocks. Corresponding layouts in SL show how 3D items reflect the Moodle design.

integrating these systems. We argue that any such integration should avoid merely presenting a weak LMS interface inside of SL, but should rather attempt to build something innovative that might lead to richer forms of interaction. Finally, we discussed how such integration may be achieved, and detailed our initial work in this area. While much remains to be done, we are confident that this will be a productive area of activity – and only time will tell what exciting shapes the flat worlds of LMS are transformed into when they become fully realised in three dimensions.

References

Antonacci, D., Modaress, N. (2005) Second Life: The Educational Possibilities of Massively Multiplayer Virtual Worlds (MMVW), EDUCAUSE Western Regional Conference, April 26, 2005, San Francisco, CA. http://www2.kumc.edu/netlearning/SLEDUCAUSESW2005/SLPresentationOutline.htm (accessed August 2006)

Crider, M. (2006). "Living and Learning in Second Life: A Firsthand Exploration and Tour of a User-Created Virtual World." Games, Learning, and Society Conference. Madison, WI, July 2006. http://homepage.mac.com/acrider/SL/SpaceportAlphaTalk-SV3.mov (October 2006)

Dede, C. (2004). "Enabling Distributed Learning Communities Via Emerging Technologies - Part One." T.H.E. Journal (September).

Dede, C., et al. (2005). "Fostering Motivation, Learning, and Transfer in Multi-User Virtual Environments." Paper presented at the American Education Research Association, Montreal. http://muve.gse.harvard.edu/muvees2003/documents/Dede_Games_Symposium_AERA_2005.pdf (last accessed October 2006)

Delwiche, A. (2003). MMORPG's in the College Classroom. *The State of Play: Law, Games and Virtual Worlds*. New York Law School, November 2003. http://www.nyls.edu/docs/delwiche.pdf (last accessed October 2006)

Dibbell, Julian. "A Rape in Cyberspace." The Village Voice 21 Dec 1993.

Dickey, M. D. (2005). "Three-dimensional virtual worlds and distance learning: two case studies of Active Worlds as a medium for distance education." British Journal of Educational Technology **36**(3): 439-451.

Democracy Design Workshop (2006) http://dotank.nyls.edu/DemocracyIsland.html (accessed October 2006)

Global Kids. (2006) http://www.globalkids.org/olp/ (accessed October 2006)

Kemp, J. (2006) "There: Fading platform offers good chat tools." [Weblog entry.] From where I hover…. http://www.simteach.com/blog/?p=16/ (accessed October 2006)

Livingstone, D. and J. Kemp (2006). "Massively multi-learner: recent advances in 3D social environments." Computing and Information Systems Journal, School of Computing, University of Paisley 10(2).

Laurillard, Diana (1997) Learning Formal Representations through Multimedia, in The Experience of Learning, Marton, Hounsell & Entwistle (eds), 2nd Edition, Scottish Academic Press

Metalab (2006), http://metalab.blogspot.com/2006/06/communal-whiteboard.html (accessed October 2006)

Resnick M. et al. (1998) Digital manipulatives: New toys to think with. Proceeding of CHI 1998. http://llk.media.mit.edu/papers/dig-manip (accessed October 2006)

Robbins, S. (2006) "Image Slippage: Navigating the Dichotomies of an Academic Identity in a Non-Academic Virtual World." *Education Workshop at the Second Life Community Convention.* San Francisco, August 2006.

Second Life (2006), http://secondlife.com/developers/mapapi/ (accessed August 2006)

SimTeach (2006a) http://www.simteach.com/wiki/ (accessed August 2006)

SimTeach (2006b) http://simteach.com/moodle (accessed August 2006)

Solvang, J. (2006) Dark Life in Second Life. Way Out There Radio. http://www.wayoutthere.net/GameReports/DarkLife.html/

Wikipedia (2006), http://en.wikipedia.org/wiki/Three-tier_%28computing%29 (accessed October 2006)

Authors Note

Daniel Livingstone teaches computer game development at the University of Paisley. Research interests cover ALife, Game AI and teaching and learning with game technology.

Jeremy Kemp has coordinated online communities since 1998 and taught Web-based distance learning courses at the university level, starting in 1999. As a research assistant for Stanford University's medical school in 2001, he created Flash simulations for radiation therapy further education. Contact: jkemp@simteach.com.

TEACHING FOR INNOVATION

TOPIC 4. THESIS PREPARATION AND GUIDANCE

Small Piddly Projects, and Big Time Undertakings

TP: The Roles and Phases of Mentorship

TP: Combining Undergraduate Research and Learning

Teaching Examples (Bricken):

HCI: Project Ideas and Refinement

HCI: Design a Software Toolkit

Ethics: Local Expert

SMALL, PIDDLY PROJECTS, AND BIG TERM UNDERTAKINGS

Excerpted from: K. Donaldson, "Outside the Classroom - Workload and Studying," in *The Engineering Student's Survival Guide,* Boston, MA, McGraw-Hill,1999, pp.77-79. © 1999 McGraw Hill Companies - Inc.

Engineering projects pop up in any course where there may be some engineering design going on (almost everywhere). Whether you choose door 1 (a heat exchanger!) or door 2 (you, too, could design your very own voltmeter!), recognize that projects always take more time than is budgeted. The key to painless projects is: Start early. Not a surprise, eh? We will subdivide projects into two categories: SPPs and BTUs. SPPs (small, piddly projects) and BTUs (big term undertakings) can be individual or group projects.

SMALL, PIDDLY PROJECTS (SPPs)

SPPs are those projects that professors decide are fun ways for you to apply what youve learned in their courses. You often have only two weeks (or Less) toward the end of the term to wrack your brain, then write your work up nicely, and maybe even present it to class.

1. First things first. Understand the project assignment. Many times the problem statement is much more confusing than the problem itself..

2. Research. SPPs usually do not require much outside researchunless paying attention in class isn't your forte. Are there any back-of-the-textbook computer programs you need to learn? Depending on the project, sometimes a quick trip to the library or a 15-minute Web search can help substantially. Once you figure out what you need to do, what equations youll need, what methods work the best, and what kind of results you should expect, you are set to...

3. Work it. Allow yourself lots of time (goes with starting early) for computer program, dont forget to allow time for debugging and sanity breaks.

4. Test it. Does everything work without glitches? Your work should be easy for the professor to follow when he or she grades it. Did you state your assumptions? Sometimes decisions that are clearly evident to us are not obvious to others tackling the same design problem. Look critically at your project for any potential holes that could cost you.

5. Fix things. Rework it Add. Subtract. When running short on time, fill in as much as possible and list what you might have done had you not run out of time.

6. Write it up. Isnt it exciting? You are almost finished..
7 Go the extra mile. It takes very little time to polish up a project. Throw all those sources into a bibliography. Draw a diagram that clarifies final concepts. Whip out a spreadsheet graph. Number your pages.

BIG TERM UNDERTAKINGS (BTUs)

And you thought BTUs were British thermal units! You just cant unknowingly and accidentally find yourself in a class with a term project (a.k.a. big undertaking). A BTU is not an SPP with more time to think. You should have been forewarned by upperclassmen and even by professors. Big term undertakings are often the finale-the end of a course series or maybe the end of your undergrad career (senior projects!). They tend to reflect real-life engineering and project management. Although BTUs are a lot of work, it is cool to see the engineering theory youve learned come together to produce something impressive. A good term project will make you an expert in your chosen topic.

1. First things first. Understand the project assignment and what is expected. Master the acronyms. Learn (or relearn) how to do journal searches at the library, operate machine-shop tools, use computer codes you have tried to forget, and so forth..

2. Research. It is this phase of the BTU that consumes the largest chunk of time; as much as half of the term can be spent getting up to speed on the assigned or chosen topic. Any good project that really puts you to work will cause anxiety and frustration early on. The whitecapped swell of frustration follows the realization-of-the-magnitude-of-what-you-have-gotten-yourself-into wave that washes over you as you weed through all the background information. First, start with the easy stuff. If you are doing a project on contaminant tracking in rivers, Begin with some water quality and contaminant readings from an environmental engineering text. After you get that nailed down, move on to the journal articles. Don't be afraid to ask questions of professors, graduate students, friends of distant relatives, and even people in industry.

3. Generate. Generate as many ideas, solutions, methods, possibilities and paths as possible. This is a great time for a mind map.

4 Work it. Working it means decision making by you and perhaps a computer. After the time and toil devoted to research, this step is almost disappointingly easy.

5. Double-check results. Does everything make sense? Check your assumptions. Double-check your constants and anything that involves a unit

conversion. The answers are often obviously good or bad for BTUs. If you find everything looks good, its time to feel relief. If things look monstrous, make an appointment with the professor there may still be time to recover. You are approaching the home stretch.

6. Tend to the small details. With big projects there are often many minor considerations that get tossed aside. Dot your is, cross your ts. Produce schematics of your solution and progress. Go back and fill in the gaps.

7. Write it up. After all the time you have devoted to this project, dont slack off now! Writing up the BTU will take more time, patience, and disk space tan expected. If possible, get a good nights sleep before the final edit. It is painful to see a typo in the freshly bound copy.

8. Go the extra mile. Polish it up. Scan in pertinent photographs. "Borrow" some slick graphics from the Web (its legal for educational purposes as long as you credit the source!). Generate some CAD drawings and renderings. Put a cool graphic on the cover page.

9. Take it to the printer for binding and extra copy. Don't do this last minute! ("The earliest we can have this done for you is tomorrow afternoon.") Pick a cool color for the cover (Kinkos has Rocket Red for the Aero/Astros). Make an extra copy. Finish out the semester with bang.

THE ROLES AND PHASES OF MENTORSHIP

Michael W. Galbraith,
Patricia Maslin-Ostrowski,

ROLES OF MENTORSHIP

Within the mentorship process, a mentor often assumes multiple roles to bring about the enhancement of the mentee's professional, personal, and psychological development. At different times, the mentor may be a role model, advocate, sponsor, adviser, guide, developer of skills and intellect, listener, host, coach, challenger, visionary, balancer, friend, sharer, facilitator, and resource provider. Along with these roles comes a responsibility to consider the psychological dimensions of the relationship, for example, accepting, confirming, counseling, and protecting. The role that best describes the mentor may be decided as a result of how well the mentor understands the total mentorship process. Clearly, the mentor role does not suit all people, including professors.

PHASES OF MENTORSHIP

There has been little investigation of mentoring phases or stages from a conceptual and theoretical perspective, except for the work of Kram (1985) and Cohen (1995a). Kram examined the phases of a mentor relationship from the perspective of psychological and organizational factors that influence career and psychological functions performed. She suggests that developmental relationships vary in length but generally proceed through four predictable, yet not entirely distinct, phases.

THE INITIAL PHASE is the period in which the relationship is conceived and becomes important to both mentor and mentee. This phase may last for a time span of six months to one year. From the undergraduate perspective, this would occur during the freshman year. Given the apparently overwhelming challenge of college to most freshmen on entrance, one can imagine the mentor on the team finding himself or herself in great demand. Yet, all students, undergraduate and graduate level, learn best in a supportive environment, and having a designated mentor on the team will give students much easier access to faculty. The mentor team member would be willing, able and desirous of this kind of interaction with students, instead of faculty whose academic preparation and research makes them offer "limited office hours."

THE SECOND PHASE, called the cultivation phase, lasts from two to five years. For the undergraduate, this then might take place during the sophmore and junior years, or even longer. During this phase, the positive expectations that emerged during the initiation phase are continually tested against reality. The mentor and mentee discover the real value of relating to each other and clarify the boundaries of their relationship.

PHASE THREE, separation, is marked by significant changes in the relationship and might happen during or soon after a student's senior year. It is a time when the mentee experiences new independence and autonomy, as well as turmoil, anxiety, and feelings of loss. The separation phase lasts from six months to two years. Mentors on teams that are teaching college seniors or students at the end of their graduate course work will represent a new resource to students feeling the anxiety of departure from the comfort of their college or university years and seeing the uncertainty of their postgraduate experience.

THE FINAL PHASE is redefinition. In this phase, the relationship takes on significantly different characteristics and becomes either a more pee-like friendship or one that is characterized by hostility and resentment. In general, during the redefinition phase, both the mentor and mentee recognize that a shift in developmental tasks has occurred and that the previous mentorship process is no longer needed or desired.

Getting out of sync with the developmental phases of the mentoring relationship could result in a less-than-positive experience for both mentor and mentee. Although everyone will not experience the phases at the same rate, it is essential that they go through all of them, and in sequence.

If one accepts the stage theory of mentoring, it is obvious that the time commitment required precludes this being accomplished in a single semester. Mentoring is not a short-term relationship. It does not fit the higher education model of taking a series of courses with different professors if the expectation is for all faculty to mentor all students. One course in one semester does not provide sufficient time to move through the total process.

It is, however, reasonable to expect that if the mentor team members are given the responsibility for teaching entry-level required courses, then they may begin to establish a relationship with future mentees early in the students' academic careers. This would be accomplished, in part, through active listening and questioning that

establishes a psychological climate of trust. This lays the foundation for a more engaging mentoring relationship. Without this kind of connection, the likelihood of a meaningful mentor-mentee experience is limited.

Although mentoring relationships evolve over an extended period of time, Advising can be effective in the short-term because the emphasis is more on information than on relationship and nurturing. On the other hand, if the team members chosen to be mentors are given the companion assignment of department advisers, they would have a better chance of getting to know students both in and out of the classroom. This would allow them to cultivate relationships further and continue building a foundation of trust. Advising may be transformed into mentoring. An additional benefit to this team approach is that students would get some of their needs met through the department mentor - for example, advising, career planning, and even some counseling needs - rather than having to seek out help from strangers located across the community.

CONCLUDING THOUGHTS

Good mentoring is a distinctive and powerful process that enhances intellectual, professional, and personal development through a special relationship characterized by highly emotional and often passionate interactions between the mentor and mentee. Although we can assume that all professors in higher education engage in some level of instructional activity, it cannot be concluded that all are actively involved in mentoring, nor should they be. The complete mentor role does not fit all individuals: some faculty are less inclined toward developing close relationships with students and with nurturing the students' development. Not all faculty are capable of or willing to take on this role and if required to do so would be inadequate or "incomplete" mentors. That is why the faculty team concepts has the promise of improving the quality of education. If only faculty who are well matched to this role become the team mentors, students will be better served.

Even if all professors are not mentors, understanding the role of the complete mentor can be a template for the good instructor. The essence of mentoring is grounded in the concept of one-on-one teaching. If one is engaged in mentoring, one is engaged in teaching. Thus, in addition to having the responsibility of mentoring students, the team mentor could also be asked to share his or her expertise regarding the mentor role with colleagues. The function of the effective mentor, which include building a relationship, providing information, being facilitative and challenging, serving as a role

model, and co-constructing a vision, are not far removed from what good teachers do. If one also examines the role of a skillful instructor, it will become clear that there is high correlation between the two roles (Brookfield, 1990, 1995; Daloz, 1986). Regardless of the academic discipline or subject, the instructional process can be enhanced by understanding and incorporating aspects of the complete mentor role.

Instructors as mentors, according to Daloz (1998), provide a balance of support and challenge such that our learners feel safe to move. From ancient times to contemporary life, mentors have challenged students to have a vision that places their journey in a larger context and invokes purpose in their lives. Mentoring is a special role that should only be assigned to professors who embrace it. Mentors support their students, challenge their students, and help their students construct a vision to further their educational journey. Complete mentors work in a truly responsive and interactive way with learners, which allows for a profound affirmation of both teaching and learning in the higher education environment. The faculty team model would permit the mentor-mentee relationship to flourish.

REFERENCES

Brookfield, S.D. (1990). The skillful teacher. San Francisco, Jossey-Bass

Cohen, N. H. (1995a). Mentoring adult learners: A guide for educators and trainers. Malabar, FL: Kriger.

Daloz, L.A. (1986) Effective teaching and mentoring. Ssan Francisco: Jossey-Bass

Daloz, L.A. (1998) Mentorship. In M.W. Gallbraith (Ed).), Adult learning methods (2nd ed. Malabar, FL: Krieger

Kram, K.E. (1985) Mentoring at work: Developmental relationships in organizational life. Glenview, IL. Scott, Foresman.

COMBINING UNDERGRADUATE RESEARCH AND LEARNING: A THREE-STEP APPROACH

Bunmi O. Olatunji and Donna M. Desforges, of Wisconsin-Stevens Point

Today's undergraduate is typically accustomed to traditional learning methods--taking information in through textbooks and lectures, memorizing that information, and reproducing it on an exam. Relying solely on such methods of teaching and learning may undermine the unique learning needs of students (Kaplan & Kies, 1995), and hence provide a less than optimal learning situation. In addition to the development of intellectual and critical thinking skills, student needs include learning about the process as well as the content in their disciplines. And underpinning all their needs, students have a basic need to feel and actually have ownership in their education. In light of that, we conceptualize the optimal teaching and learning experience as requiring three steps: stimulation, application, and integration. Further, we believe that faculty-student collaborative research and other scholarly activity offer an excellent means of incorporating these steps into students' learning experiences, thus more fully meeting their educational needs.

The Three-Step Approach

Stimulation

As it applies to learning, stimulation means a more active engagement with the material to be learned, as well as growth and development of student
interest in the material and in their abilities to work with it. Often times, students may experience participation in their classes as a daunting situation where the potential for ridicule or embarrassment reinforces well-rehearsed silence. To remedy this situation and really engage students, faculty must play the role of facilitator, developing a climate of trust in which students can openly risk examining their personal thoughts, confusions, and opinions (Barkham & Elender, 1995). Students require an environment in which they can go beyond merely memorizing facts in order to grow as intellectuals.

Several methods of stimulation help create such an environment. For example, asking specific controversial questions relevant to course material helps draw students out. Having students debate various positions on current topics or issues further stimulates students' critical thinking skills. The method for stimulating learning we advocate is student involvement in research and scholarly activity. For maximum effectiveness, this involvement should include every

aspect of the activity, from the initial conception of the research plan to the final research product. But more on this in a moment.

Application

Is the smartest person in the class the person who can remember the most information at the time of the exam? Or is it the person who can take that
information and correctly apply it to a novel situation? Studies have shown that when given two identical exams on different occasions, undergraduate students do significantly worse on the second exam (Harrison, 1995). Part of the problem may stem from the students' lack of broader application of the material they have studied. In other words, the exam was the only opportunity students had to apply the material. Long-term retention of information calls for a broader application or use of the information students study.

Integration

The final phase of learning--and of our model for improved teaching--comes when students are able to integrate material into a broader knowledge base. Fostering integrative learning requires that faculty encourage students to analyze and interpret class literature, as well as indicate the extent to which they agree or disagree with the author's perspective. Thus--harking back to the importance of establishing an environment of trust--integration not only requires that students include the rationale behind their perspective, but also that their rationale include an understanding of additional literature supporting their position.

While it requires rigor, integrative teaching reaps exponential benefits. Not only does integration serve as a culmination of earlier steps in learning, it also propels students toward higher levels of critical thinking. In fostering a keen sense of appreciation and understanding for explanatory information, integrative learning ultimately enhances students' ability to assimilate and utilize content (Olatunji, 1999).

The Model In Operation

Faculty-Student Collaborative Research Research involves the active pursuit of knowledge, and it is this process of pursuit that is often overlooked in undergraduate teaching in favor of covering content. But in general, undergraduates want to be more actively involved in the process of their disciplines (Long, 1994). Thus, one means of accomplishing the three steps to improving the student learning experience is through faculty-student collaborative research.

As we said earlier, to fulfill the stimulation step, students should be involved in each step of the process. One way that we have done this is to start each semester with a small set of readings with a common focus. Facilitated by a faculty member, student researchers work collaboratively to generate hypotheses and the means to test the hypotheses. Obviously, this process enhances students' sense of ownership in their education.

The application step involves applying the information students have learned through readings and discussions to the task of creating a novel way to test their hypotheses. It also involves the actual testing of one or more hypotheses through the methods students have generated. In this process, students learn to think critically about published works' hypotheses and the adequacy of tests, results, and conclusions. This critical awareness prepares them to apply that same process to the original work they are creating.

The integration step begins with the process of analyzing and interpreting the data collected from students' research. Students are encouraged to interpret and explain the rationale behind the acceptance or rejection of their hypotheses. After that, students have to integrate their findings into the appropriate knowledge base that already exists. Not surprisingly, this often generates more questions that students want to answer, which often prompts follow-up studies of some sort.

Conclusion

In practice it isn't as easy as "one, two, three," but our three-step model encourages faculty and students to take "one step beyond" the usual,
ordinary and expected right from the start. The payoffs from thinking beyond the current horizon of most undergraduate teaching are tremendous. We know that undergraduates have learning needs that go beyond taking in information and repeating it back at exam time, and we believe that faculty-student collaborative research goes a long way toward meeting those needs. Certainly, research within the proper atmosphere unquestionably promotes creativity, synthetic thinking, and the appreciation of knowledge (Seligman, 1999). Thus, even though an undergraduate research project may focus on a particular area, the processes and the educational benefits of doing the research extend well beyond that project or that subject area. Going one step beyond enhances all of the undergraduate's learning experience.

References

Barkham, J.; Elender, F. 1995. "Applying Person-Centered Principles to Teaching Large Classes." British Journal of Guidance and

Counseling, 23, 179-198.

Harrison, A. 1995. "Using Knowledge Decrement to Compare Medical Students' Long Term Retention of Self-Study Reading and Lecture Materials." Assessment & Evaluation in Higher Education, 20, 149-160.

Kaplan, E.J.; Kies, D.A. 1995. "Teaching Styles and Learning Styles: Which Came First?" Journal of Instructional Psychology, 22, 29-34.

Long, F. 1994. "Research as Living Knowledge." Studies in Higher Education, 19, 47-58.

Olatunji, B. 1999. "Undergraduate Research as an Invaluable Experience." APS Observer, 12, 24-27.

Seligman, M. 1999. "Teaching Positive Psychology." Eye on Psi Chi, 4, 16-17.

SOME PROJECT IDEAS

1. **Virtual University Course**

Develop a short course to be presented on the web. A clever twist would be to develop this course as a web-based remote course.

2. **Web-page Refinement**

Select a web-site from a moderately large company. Review and critique their design and performance. Then rebuild the entire site to correct the identified problems. A clever grounding for this work would be to send our results to the company.

3. **Internet Software Tool**

Select, design, and implement a software tool which facilitates some aspect of web-interaction. Ideas may include intelligent agents, search engines, web-page builders, site mapping and layout, download analysis, etc.

4. **Customized GUI Interface**

Evolve an existing interface toolkit with some customized refinements, such as a new type of widget, customization tools, dialog management, or interaction traking.

5. **Virtual Reality System**

Develop existing C code for parts of a VR system, including 3D interaction devices, smart terrain, multiple participants in a single environment, new virtual bodies, etc.

6. **Finite State Machine Emulators**

Software to emulate an interactive system, such as an ATM, a soda machine, a telephone answering system, airline ticket booking, etc.

7. **Java-based Gaming**

Implement a fun game for the internet. Could be a version of a standard adventure and fighting game, or a strategy game such as Diplomacy or Stratego, or a graphics/visual game such as Life or Centipede.

8. **Mathematics Visualization**

Develop a visual interface to some abstract mathematical structure such as an N-dimensional cube, 3D knots, or Fractals

9. **Spatial Arithmetic and Algebra**

Develop a graphic interface for 7th grade math, using manipulative structures rather than equations and symbols.

10. **Manipulable Logic**

Develop an interactive interface for Boundary Mathematics

11. **Innovation Prototyping**

Select a yet-to-be commercialized application, such as wearable computers or TV-wristwatchs, and develop a prototype functional design for the interface.

==========

PROJECT ORGANIZATION

Our class project will be **Knowledge-based Hyperlinks (KBHL)**. (The name is suggestive, not final.) The general idea is to develop a demonstration example (simulation of) traversing a network of hyperlinks based on content or semantic information, rather than on syntactic structures such as keywords.

The component tasks/roles for this project include:

0. Research: have others tried this approach? What did they learn? What topics partially address our project (eg: knowledge engineering, hypertext, web search, expert systems, interactive interface, software agents, etc.)?

1. Develop a sample database of content-carrying web-pages, with hyperlinks across various content components.

2. Develop a knowledge-based data-structure which attaches to each hyperlink. The knowledge-base will contain conventional expert system-like assertions of facts and relations.

3. Develop an inference strategy for traversing the knowledge of various hyperlinks. This may include pattern-matching, Baysian probabilities, various types of inference, and other semantic-like structures. See below for more ideas.

4. Develop an interaction plan which permits the user to understand and traverse knowledge-based links.

5. Develop an interface prototype for using KBHLs. The interface should require minimal learning, and have a “natural” feel.

6. Discuss and roughly design extensions to the KBHL, including user-extensibility, automated documentation, and software agency.

Discussion

The content web-pages require careful selection and design, to maximize the (apparent) utility of the KBHL tool. Research from Ontology Engineering (ie what the folks at Yahoo do to organize their weblinks) will guide this effort.

The available types of *intelligent traversal* require careful design, so that 1) the tool will be useful for finding information, and 2) the tool will be understandable to normal folks.

How semantics is captured and accessed is of critical importance. How do we know what the user is looking for? How will users be able to say what they are looking for? What types of intelligent traversal are useful?

A given links will usually contain may intelligent branches. How will the user know which to select? That is, the organization of information is not only, or even necessarily, logical.

What type(s) of organizational structure do we what our smart links to expose? Inferential techniques address implicit or embedded information. Other types of smart links expose different structures. For example,

a *refinement link* leads to more detail on a topic.
a *classification link* provides a property inheritance context.
a *chronological link* tells you what happened before or after.
a *spatial link* navigates through locations.
a *dependency link* identified prerequisites, requirements and causal structures.
a *structure link* decomposes an object into its component parts.
a *decision link* traces choices and their consequences.
an *analogy link* identifies thnig that are similar but not necessarily related.

Our problem maps onto a classic graph problem: what kind of nodes and vertices make sense? How many types of links can be used at one time? Should nodes or links provide consistency?

Observations

The simulation web-pages need to accurately reflect the data-structures underlying actual web-pages. We will need to figure out how additional information can be easily and portably attached to links.

Specialized types of inference are needed for different fields of knowledge. Only some kinds of knowledge are reducable to knowledge-based encoding.

Knowledge may not be about "content", it can also be about structure (the form of the link), about possibilities, about grouping, about proximity, etc.

We will need to show *critical functionality*. What does our tool do that other tools do not do? How is the advantage measured? Where are the strong. weak, and failure points?

There may be no solution for information overload. We can be overwhelmed by too many windows, by too many nodes and links, by too much scrolling. Perhaps links should filter and refine rather than enhance access.

Understandable structure may need to be designed and written into the website itself, rather than put into links.

Techniques for structuring and filtering:

Labeling: clear, concise labels and concepts
Chunking: relatively small, related hunks of information
Relevance: all information pertains to the content of the page or the goal of the user
Consistency: similar items are treated in similar ways

Hyperlinks may simply increase the desire for better content structure and more efficient linking models. That is, smart links may expose the greater weaknesses of hypertext systems.

Bottom line is that the information itself must have a structure for a smart link to expose.

PROJECT REFINEMENT

For our class project, we will be developing a single website which demonstrates ***knowledge-based hyperlinks*** *(perhaps just* ***SmartLinks****).* Active items (words, graphics, diagrams, sections, etc) will permit traversal of the website based on semantic rather than syntactic references.

Discussions and decisions:

1. Review of relevant class handouts, and research into possible approaches.

2. Identify the task that the potential user of the website will be trying to accomplish. Develop several scenarios which capture the semantic need and intent.

3. Identify the types of traversal available to the user. Rough out the engine functionality and the system architecture.

4. Select a content area for the site which facilitates task accomplishment. The types of smart links will depend directly on the content and functionality of the site.

5. Identify the requisite languages, skills and roles for the project. (content and site development, link definition, engine development, interaction design,...)

6. Discuss the issue of novice vs expert users.

7. Brainstorm possible models of interactivity and interface displays. How will the user:
   1) know what is possible? 2) know what to do? 3) communicate their needs?

8. Assignment of tasks to individuals.

Recall

- Our links will be more useful if they are filters rather than generators.

- We may use several types of traversal, but each type will have a separate underlying traversal graph. We will eliminate interaction between semantic components.

Individual assignment:

Construct a **graph** of a possible site, with nodes being content chunks and links being traversals. You will need to

1. make up some *rough* content chunks in a content area (the class should have decided the content area tonight),
2. imagine some tasks, queries and traversals,
3. identify the type of connection being traversed,
4. specify in detail some content containing the link and some content being traversed to, with emphasis on the semantic connection between the two, and
5. be prepared to show this graph to the class.

PROJECT REFINEMENT -- ONTOLOGY

The content of our project is **Childhood Ailment Diagnosis**. The rough architecture is to use **SmartLinks** to connect the HTML-page-structure to a semantic-network, traverse the semantic-network, and then return to the HTML location which correcsponds to the end of the semantic-network path.

So we'll be constructing two (or more) databases: the **structural-HTML-database** and the **semantic-net-database**. Then we'll be constructing a **linking database**, which contains the connectivity between syntactic and semantic elements. We will also need a **traversal engine** as the back-end, and a **query interface** as the front-end.

Issues:

1. Review class assignments; make a rough pass at the data structures (the HTML structure graph and the semantic network graph).

2. Refine content area.

3. Refine semantic nets, and identify what we can do semantically. Types of traversal. Identify the language of the semantic-net (ie what types of nodes and links)

4. Discuss the types of structural (HTML) forms. Grainsize of information units; grainsize of textual and paragraph HTML; types of knowledge units. Other types of syntactic organization. Other types of semantic queries.

5. Rough pass at the control structure architecture.

6. Discuss the implications of the semantic-net modleling approach, the notion of ontology and the mappings from semantic-nets to relational calculus.

7. Discuss the limitations and traps in the proposed control architecture.

8. Identify roles, esp. who will be writing what code. Identify the requisite languages, skills and roles for the project. (content and site development, link definition, engine development, interaction design,...)

9. Continue to develop usage scenerios.

10. Consider the interaction and user-interface issues.

11. Discuss the issue of novice vs expert users.

12. Discuss the issue of partial vs. complete knowledge. Hypothesis testing in diagnosis.

DESIGN A SOFTWARE TOOLKIT

The class is to specify the software tools and interface techniques for a Virtual Reality suite.

Do not use English for the specification. Use command words, or a formal language, or functions, or predicate calculus, or diagrams, or animations, or a programming language, or any form that you can be explicit about what a tool does or how it works, but do not use English, use a specification language.

Follow this ***organizational structure***:

Form eleven groups of two members. Each group will be responsible for a particular tool in the suite. No group may duplicate another group's functionality.

Include at least these six tools:

The Wand

The Virtual Body, sub-components:
- visual sensors
- audio sensors
- position sensors

The Physiological Model, sub-components:
- body position model
- voice recognition

The remaining five groups may specify a tool of their choice.

Each group must coordinate its specification language and protocol with other groups (tools) that use them. For example, the Wand may depend on the position of the virtual body.

Super-observers must report the progress of class activity.

This is a two hour exercise, I will collect a two page design specification from each tool/group and a two page design integration from the class as a whole.

Grades for the class will be based on functionality, integration, and clarity of specification.

Final Assignment

Computer Ethics is a field that covers many diverse and relevant topics. In order to anchor our skills, each student is to become a "local expert" in one particular branch of Computer Ethics.

Your assignment is to

1. Select a topic or focus area,

2. Locate and read selected articles, discussions, and case studies,

3. Formulate the ethical issues and analysis, policy needs, and moral grounds, and

4. Contribute your specialized knowledge to class discussions.

Your focus area could be any one of the following

- one wide area of Computer Ethics
  (regulation, privacy, security, property, professionalism)

- one specialized, fairly narrow topic
  (digital signatures, electronic voting, wire tapping, public encryption)

- one particular person or group
  (Deborah Johnson, Lawrence Lessig, CPSR, EFF)

- one particular dilemma or case study
  (software copying, gender bending, protection for minors, spamming)

You do *not* have to prepare a paper, talk, or presentation of your selected topic. You will be expected to contribute knowledge, opinion, analysis, current status, and other aspects of your topic during class discussion throughout the quarter.

Each class meeting will emphasize some particular aspect of Computer Ethics. Each student is expected to be able to discuss connections between their focus area and any other topic at any time during the quarter. When the class is focussed on a topic which is in your field of concentration, you will be expected to contribute more information and analysis than usual.

Students may bring in case studies, discussion topics, questions, analysis and/or opinions for class discussion at any time.

TEACHING FOR INNOVATION

TOPIC 5. CURRICULUM DESIGN

Teaching and Facilitating Learning Syllabus

TP: The Function of the Course Syllabus

TP: The Value of Writing a Course Portfolio

Syllabus Elements

Course Structuring

Cognitive Taxonomy

Affective Domain Taxonomy

Psychomotor Domain Taxonomy

TP: 101 Things You Can Do the First Three Weeks of Class

Teaching Examples (Bricken):

Situated Curriculum

Curriculum Exercises

Just What is VR Anyway?

Wonderful Computer Science Books

TEACHING AND FACILITATING LEARNING
Lake Washington Technical College
WINTER Quarter 2007

Text

Successful Beginnings for College Teaching, Angela Provitera McGlynn, Atwood Publishing. Amazon $23.75

Course Description

New instructors/learners will practice implementing a variety of instructional strategies and student assessments to meet course outcomes. Focus is on teaching strategies and methodology and ways in which instructors act as facilitators of learning in their classrooms. Instructors/learners will actively practice their teaching skills to begin to implement learner-centered instructional activities and lessons that they have devised. This course teaches to the global outcome of Communication.

Method of Instruction

This class uses a combination of concept lessons, demonstrations, cooperative, hands-on and independent practice using a textbook, handouts, resource books, technology, and the Internet. The Blackboard learning system will provide learners access to handouts, grades, and assignments 24 hours a day. Students need access to a computer or computers at the college to successfully complete discussion board assignments. Instructor will be available by appointment and during office hours.

Learning Outcomes

The instructor-learner will:

Design well-organized and learner-centered instructional activities that actively engage students and promote achievement of student-learning outcomes.

Analyze, identify, and select the essential information of a course syllabus--contract between instructor and student.

Deliver and/or facilitate instructional strategies (lecture/concept, demonstration, discussion, small group--cooperative) that provide students regular opportunities to actively engage with course content to achieve course objectives/outcomes.

Demonstrate the ability to convey ideas/feedback in a variety of formats to meet the needs of diverse students.

Outcome Assessments

1. Create/edit a course syllabus. The syllabus will consist of elements in the syllabus example and specifically to include elements in grading, attendance, classroom policies and procedures, and plagiarism.

2. Develop/revise a rubric or checklist tool to use in observing faculty.

3. Observe two faculty in the classroom and use the checklist tool to measure instructional effectiveness.

4. Facilitate a lesson incorporating a new teaching strategy which provides a safe classroom environment.

5. Develop and write one organized lesson plan using the 8 elements. This plan will provide an opportunity for student practice and instructor feedback.

6. Choose a Classroom Assessment Technique (CATS), sponge, or another teaching strategy and implement in the lab or classroom.

7. Facilitate a lesson integrating a strategy which focuses on engaging and soliciting feedback from every student.

8. Facilitate a lesson using a game, role play, and/or collaboration strategy.

9. Write a 5-paragraph Reflection Paper

10. Participate by providing feedback on current classroom experiences and sharing/conveying ideas during classroom and blackboard discussions.

Grading

Learners are evaluated on their ability to follow directions and work individually and/or cooperatively in a group to successfully complete all activities and assignments.

**A 10 percent penalty will result for any homework assignments turned in after due date and up until one week late. After one week from due date, maximum points to be earned will be 75%. No points available after 4 weeks from due date. Check calendar for due dates.

Complete 9 assignments (check outcomes assessment above for listing and calendar below for point distribution)

Blackboard Discussion Questions:

Complete sixteen (16) responses to questions A-H–one response to original question and one response to peer (25 pts per question)

Active participation (25 pts each class meeting)
Discussion during class and actively engaged in activities

COURSE POLICIES

Respect

Show courtesy by not talking with other classmates while instructor or another student is speaking. If you have a question or comment, please raise your hand, rather than starting a conversation with your peer. Show respect for other student's ideas and comments. Wait for your turn to speak.

Active Participation and Cooperation

Students are expected to participate verbally independently and cooperatively within groups. If a student disrupts the learning environment or does not participate cooperatively within a group, he/she will receive a reduction in participation points. If a student cannot attend class or is late to class, he/she will receive a reduction in participation points. Attendance during class time will increase success in this class.

Make-up policies

Students who miss class meetings are still responsible for lecture material covered, handouts, and announcements. It is a good idea to contact a class member for additional instructions and/or assignments given in class. Also check the Blackboard site, if available, for additional information and/or instructions. All assignments and handouts should be posted on this site. Students are encouraged to make an appointment with the instructor or meet during office hours for any further clarification.

Other/inclement weather

Check local radio stations in the event of weather or other possible closures. If the weather is such that it is dangerous for the student to come to school, the student will work at home, be counted present in class, and the classroom will be a lab environment until the weather improves. E-mail or voice mail instructor if weather is causing absence from class.

Class time is subject to change due to equipment/software problems and/or holidays, meetings, and other schedule changes.

Student Code of Conduct

Students are expected to follow the college student conduct code, WAC 495D-120, which prohibits cheating, plagiarism, theft, or hurtful behavior toward others and shall be grounds for discipline pursuant to college rules. See the Student Handbook for more details on the code. Plagiarism is defined as not doing your "own work." If you turn in assignments that are created by another student, you will receive a 0 for that assignment/test.

Cell Phones and Pagers

Turn cell phones or audible pagers to vibrate before the class session begins. If you must use this device in an emergency, quietly leave the classroom.

Computer Use

- Users are limited to applications listed on the menus available on screen.
- Users are not allowed to play games, use chat rooms, or use e-mail unless part of instruction.
- Users are not allowed to install programs, alter system configurations, defaults, system settings, system files, program files, data files, desktop configuration, or change colors.

Equal Opportunity Policy

Lake Washington Technical College is an equal opportunity college and is committed to principles of diversity. The college accepts students without regard to race, color, religion, national origin, gender, sexual orientation, age, marital status, disability, or status as a disabled veteran or Vietnam-era veteran. In keeping with this policy, slanderous, defamatory, or insulting remarks directed at any group of individuals shall not be tolerated in either classroom discussions or written assignments.

SUPPORT SERVICES FOR LEARNING

Computer Lab

The student Computer Lab, room T413, is open Monday-Friday 7:30a.m.-8:00p.m. There is a $25/quarter cost associated with the use of the lab, which can be paid at Registration by signing up for item number PCLB.

Peer Tutoring

Peer tutoring is available for students who are having difficulty in a class. If you would like to request a tutor, please contact the Academic Skills Center T217 to obtain and complete the appropriate paperwork.

Writing Center

Informational handouts, special grammar practice software, and writing tutorials are available in the Writing Center, room T217. There is no fee associated with the use of Writing Center computers or printers. Hours vary by quarter and are posted on the entrance to T217.

Campus Security

Your safety and security are taken seriously at the college; we have a very low incidence of crime on the campus. Although the college has no security force of its own, the campus is patrolled regularly by the Police Department and all incidents of confirmed or suspected crimes are reported.

Students requesting academic adjustments related to disability should contact Disability Support Services (DSS) in person, by phone or by email.

CLASS CONTENT

Syllabus–Classroom Management
Classroom Observation
Learning Styles
LWTC Students
Lesson Plans
Teaching Strategies
Disruptive Behaviors--Safe Environment
Retention
Reflection

CALENDAR

Directions for each of the following assignments will be discussed and handout provided.

DATE

Weekly Classroom Activities
Assignments, Readings, Discussion Questions Due Dates by midnight

Week 1
Pre-Test
Discuss elements of Syllabus–Assignment #1
Syllabus
Intro to Blackboard
Develop faculty observation checklist to assess instructional excellence.
Respond to Blackboard Discussion Question A during class time; Respond to Peer.

Week 2
Discuss observation rubric/checklist
Discuss online responses by students
Guest speaker--Scott
Read/Skim Chapter 2 in textbook in order to respond to discussion question B; respond to Peer

Week 3
Minimum 30-minute Faculty Observation
Discuss chapter 5; share an experience or favorite paragraph in chapter and why?
Read/Skim Chapter 3 in textbook in order to respond to discussion question C; Respond to Peer

Week 4
Lesson Plan intro
Edit rubric
CATS
#1 Assignment due–Syllabus (100 pts)
#2 Assignment due–Observation (50 pts)
Respond to Blackboard Discussion Question D; Respond to Peer

Week 5
No face-to-face class
Respond to Blackboard Discussion Question E; Respond to Peer

Week 6
CATS
Discuss chapter 4; share an experience or favorite paragraph in chapter and explain why? --Engaging Students
Intro to games
#3Assignment--Lesson Plan (100 pts)
Respond to Blackboard Discussion Question F; Respond to Peer

Week 7
Engaging Students
Games
Reflection
Read/Skim Chapter 6 in textbook in order to respond to discussion question G; Respond to Peer
#4 Assignment--Classroom Assessment Techniques (50 pts)

Week 8
Minimum 30-minute Faculty Observation
Respond to Blackboard Discussion Question H; Respond to Peer;
#5 Report on strategies to engage students (50 pts)

Week 9
Sharing face-to-face or online?
#6 Assignment--Classroom observation (50 pts)
#7 Assignment with edited observation checklist (100 pts)
#8 Assignment--Games/Role Playing (50 pts)
#9 Assignment–Reflection (50 pts)
Post Test

THE FUNCTION OF THE COURSE SYLLABUS

Syllabus Functions

Your syllabus can serve a wide variety of functions that will support and challenge students as they engage in their educational activities.

1) Establishes an Early Point of Contact and Connection Between Student and Instructor

Research has shown that students want more frequent interaction with faculty. You can begin to communicate your availability by including basic information such as your name, address, telephone numbers, e-mail address, office hours, how to arrange for a conference. [See Examples, Part II] You can also include a page soliciting biographical information (also address, phone #, e-mail, etc.) that will help you to learn students' names, their interests, and why they are in the course. To encourage interaction with other students in the course, you might use this information to develop a student roster (including name, address, phone #, e-mail, etc.) that is particularly useful for group work and work time out of class. You can include similar information about other important student contacts, such as TAs, technicians, main office staff, and librarians, when appropriate. This contact information will be useful in case plans change during the course of the term or semester.

2) Helps Set the Tone for Your Course

Your syllabus communicates much about your attitudes toward students and learning. The way in which you communicate your views helps students to understand whether your class will be conducted in a formal or informal manner. Communicating an openness to questions, concerns, and dialogue begins with the syllabus.

3) Describes Your Beliefs About Educational Purposes

You can explain whether your course has a product or a process orientation and how that determines your expectations of students. Explain how you have set your agenda for the course, how the course structure reinforces goals and objectives, how the activities and assignments will help them to meet both product and process goals. You may describe learning strategies and techniques you will use and your rationale for using them. You can make explicit how your criteria and standards for both their work process and products are aligned with course goals.

4) Acquaints Students with the Logistics of the Course

Courses vary in terms of the days classes meet, the instructors for each class, and the type of sessions which occur (i.e., guest lecturer, teamwork sessions, simulations, films, etc.). Your syllabus can detail this information so that students will know what to expect and can be prepared for each class meeting. Providing students with a course calendar helps them to plan their work. Noting holidays and any days on which class will be canceled or rescheduled allows students to plan ahead and prevent misunderstandings. It also shows that you respect the value of students' time. [See Examples, Part II]

5) Contains Collected Handouts Faculty often distribute handouts as they become appropriate to the topics covered. Often students put them into whatever notebook is at hand and then find it difficult to retrieve them. By planning your course, preparing the necessary handouts, and including them in your syllabus, you help students, among other things, to keep all course material together and accessible. These items, among other things, might include biographical information forms, detailed information on assignments, various evaluation forms, or diagrams and other visual representations.

6) Defines Student Responsibilities for Successful Course Work Your syllabus can help students to achieve some personal control over their learning, to plan their semester, and to manage their time effectively. If your students have a clear idea of what they are expected to accomplish, when, and even why, they will be more likely to finish assignments within a reasonable time and be appropriately prepared for classes and exams.

7) Describes Active Learning Students often conceive of learning as the acquisition of correct information, but they may not know what it means to take an active role in the process, beyond rote memorization and recall. You can include a description of your expectations for student initiative in your syllabus. If critical thinking, problem solving, and inquiry are part of your course, it is helpful to tell students that they will be asked to consider multiple viewpoints and conflicting values and to imagine, analyze, and evaluate alternate positions on issues or solutions to problems. It is also important to describe what students can expect from you in your role as teacher: content expert, formal authority, socializing agent, facilitator, role model, experienced learner, resource consultant, coach, counselor.

8) Helps Students to Assess Their Readiness for Your Course What are the prerequisites for your course? In addition to specific course prerequisites, students should be given some idea about what they should already know and what skills they should already have before taking your course so they can realistically asses their readiness. Your syllabus can provide information about the challenges students will face, the assumed skill level, the skills they will build upon, and the skills they will learn during your course. You may also include information about institutional or other sources for academic support.

Some faculty include self-assessment tools and learning contracts to assist students with this process.

9) Sets the Course in a Broader Context for Learning

Your syllabus can provide a perspective that allows students to see instructors in your discipline as active and experienced learners engaged in inquiry in their professional fields or disciplines. Many students are unaware that their instructors are involved in research and creative professional activity beyond the classroom, that they are not simply transmitters of knowledge and skills. You can encourage your students to approach the learning situation as apprentice learners in a community of scholars. You can help them to see you and other faculty as experienced active learners who can provide expert guidance about general and specialized knowledge of content and practice in your field. Your syllabus can provide information that shows students how your course fits within the discipline or profession, the general program of study, and their own educational plans. You can make students aware that every discipline or field has its unique way of knowing. You can encourage students to approach the field actively as ethnographic fieldworkers who want to understand the social and intellectual practices of the field. Assure them that you will guide them while they learn how to use the characteristic tools and modes of inquiry, patterns of explanation, discourse practices, and they types of artifacts that are valued and produced in their field.

10) Provides a Conceptual Framework Your syllabus can support major ideas, topics, and factual information. Include in it questions or issues for students to think about that range from major issues or key questions in the discipline to the meaning of a significant passage in a course reading (Bean, 1996). Such a framework will help students organize information and focus their learning.

11) Describes Available Learning Resources You can list campus resources such as libraries, reserve desks, reading rooms, laboratories, computer clusters, and studios that students may use (including their locations, availability, and policies) as well as any information concerning the location and use of aids such as tape recordings, copy services, CD ROMs or videos. You may also note the locations of specific books, videos, and sites on computer networks. [See Examples, Part II]

12) Communicates the Role of Technology in the Course Computers and computer networks have increased our ability to access information and communicate with each other. Computers are working tools that students use for their own learning: to enhance their thinking; plan and revise learning goals; monitor and reflect on their progress; set up and access their own personal knowledge files; share a common database; build their own database; use a spreadsheet; run statistical software; keep a journal; write, illustrate, and revise texts; and build up a portfolio. You can use computers as a resource tool to provide direct

instruction of new content, tutorials, and interactive simulations; to model extremely small or large phenomena (Brown, 1993; Davis, 1993a). E-mail is a practical way to interact with your students. Assignments, comments on their work, important class information, and questions to you and to other students, and extended classroom discussions are all possible uses and allow documents to be prepared, sent, received, and read by the recipient at convenient times. Institutions, individual faculty, and students are creating their own home pages on the World Wide Wed or using information servers to share course materials on-line, such as your learning-centered syllabus, reading lists, lecture outlines or notes, collaborative software, and other course information. When you use servers and the World Wide Web, you can control the information you want to access by navigating through the system to explore any topic of interest at your preferred pace and level of detail. Studies have shown that students derive much benefit from environments which encourage collaborative/cooperative learning. The Web and groupware (such as Lotus Notes) provide opportunities for asynchronous collaboration (participants can share work that may be done at different times and places). Networked writing environments encourage students to write more and to learn from each other. On-line discussion groups can lead to fuller participation in class discussions by students who may not participate in face-to-face classroom environments (Polyson, S., Saltzberg, S., & Goodwin-Jones, R., 1996).

13) Can Expand to Provide Difficult-to-Obtain Reading Materials

There are times when courses are developed before comprehensive literature is available on the topic. The syllabus can include copies of articles you want your students to read, as well as supplemental information not found in course texts. You can include materials that expand on, synthesize, and facilitate critical reflection on issues presented during formal instruction. You might include materials that fill in the gaps not covered by class presentations, or present questions raised by other points of view. When you use the syllabus in this way, be certain that you obtain necessary copyright clearances for reading selections.

14) Can Improve the Effectiveness of Student Note Taking

Good, carefully written notes are a significant resource for active learning. Active thinkers keep notebooks and journals of ideas from readings, lectures, presentations, and their own ruminations about topics. It is important to make every effort to help students improve the quality of this form of writing. As a model, you may want to include outlines that provide an orientation to topics for lectures and presentations, making it clear what you want students to remember, and providing room for their own interpretations and elaborations of the material. You can use notetaking pairs (Johnson, Johnson, & Smith, 1991) intermittently during or at the end of a lecture. (In this case, two students work together to review major concepts and pertinent information, to clarify

unresolved issues or concerns.) It is also helpful to include any detailed formulas and diagrams that students will be required to use. You may want to include study techniques that are specific to your c! ourse. In this way, the contents of the syllabus will help to organize and focus student notetaking and learning. [See Learning Tools, Part II]

15) Can Include Material that Supports Learning Outside the Classroom

Much learning takes place outside of the classroom. You can transform student study time outside of class by providing strategies in your syllabus that help students to interact more critically with the textbook, supplemental readings, or other work, so that they will be better prepared for class. For example, along with the readings you might give students a short (one page or less) writing assignment that asks them to support, reject, or modify the thesis or claims in the reading. You might include a guide for troubleshooting a story or a drawing. You can also provide self-check assignments that allow students to monitor their progress.

16) Can Serve as a Learning Contract

As an agreement or contract defining mutual obligations between instructor and students, your syllabus also speaks for the college and university. "You should realize that this fact gives you responsibilities but also gives you protection against complaints or challenges to your teaching. For example, the conditions, goals, and requirements you state enable (department chairs and academic administrators) to support your decisions on grades, teaching methods, readings, and topics of inquiry. That is only possible, of course, if you and the administration (and the students) have a record of what you promised and planned, and if your syllabus conforms broadly to program goals and policies" (SU Project Advance, 1995). You will need to be familiar with institutional policies regarding attendance, examinations, drop/adds, course withdrawals, learning disabilities, and academic integrity. Equipped with an understanding of the myriad ways a learning-centered syllabus can function, you can begin to use it in your course.

The Value of Writing a Course Portfolio

For most teachers, starting to explore students' learning can be a bit daunting. You ask yourself some tough questions: Are my students truly learning what I think I am teaching them? Am I meeting my course goals? Are my course goals right for this course? Is the work that students do having my impact on their learning? Do the materials I have chosen build connections and perspective?

Where do you look for the answers to these questions? You might turn on your computer, collect all of your course notes on your desk, and grab a stack of student papers that you have just finished grading. But you would probably find yourself wondering how to get started. Even though over the years you have given much thought to your course, this is probably the first time you have ever tried to create a written document that makes visible the intellectual effort you put into designing it and measuring its impact on student learning.

You are not along. A professor of art and art history found herself in a similar predicament:

I am a new teacher and an untenured faculty member. I teach intuitively. I go by how the class feels to me, and the overall atmosphere, and the general level of student response. I have a plan for each class day and I always vary it to respond to what arises in the studio. I used to feel strongly that the methods I used in a given situation were effective, but I never articulated why. I never voluntarily used the word "pedagogy" and was quite sure I would. I was insecure about the intellectual underpinnings of my teaching and fearful I wouldn't be able to justify how I teach if necessary.

After developing a course portfolio, she wrote, "I found to my enormous relief that many of the methods I had chosen intuitively are used by other teachers and that they even have a pedagogical basis, which I am beginning to be able to articulate."

The course portfolio provides a framework within which you think about your course design, ask yourself if your classroom practices are working, and assess the level and range of student learning that goes on in your classroom. Unlike a teaching portfolio, which might summarize all of the courses that you teach, a course portfolio is focused on a single course. More importantly, a course portfolio seeks to minimize the wheelbarrow effect of simply collecting all of your homework, handouts, and examinations into one unexamined pile. Creating a portfolio for a single course can often be more valuable than a broad teaching portfolio since it is a concise and reflective document that can be shared with peers for their review of what student learning looks like in your particular course. For example, if you were to write portfolios on different courses, the insights that you gained in your analysis of each course could form the basis of

the teaching statement that is the core of the more substantial teaching portfolio.

What constitutes a course portfolio is as individual as the instructor doing the teaching and the course being taught. Hutchings (1995) describes three common elements of a course portfolio: 1) explanation of the course design, 2) description of the enactment or implementation of the design, and 3) analysis of student learning resulting from the first two dimensions. Our model of a portfolio is similar and consists of the following essential parts:

* A reflective discussion of the content and goals of your course
* A description of your plans to accomplish key objectives in student learning
* Evidence, assessment, and evaluation of student achievement of these goals
* A reflective narrative on the relation among the above three elements

The raw material for the course portfolio is a set of three memos that you write about your course and that you then draw from to create a finished course portfolio that summarizes and analyzes student learning. The course portfolio emerges through the aggregation of the three memos about goals, methods, and learning. The faculty member's reflection on the relations among those elements is the connecting material that holds the portfolio together.

In this book we present models for two types of course portfolios: a benchmark course portfolio and an inquiry course portfolio. Each of these portfolio models offers a structure for exploring, reflecting on, and documenting a course. A benchmark portfolio presents a snapshot of your students' learning that occurs in one of your courses. This portfolio enables you to document your current teaching practices and to generate questions about your teaching that you would like to investigate further. An inquiry portfolio is useful for documenting improvement in teaching your course over time and for assessing the long-term impact of teaching changes, the success of teaching approaches, and the improvement in student learning. This inquiry process often moves teachers toward scholarship-of-teaching questions in their disciplines. In general, most instructors find it valuable to begin making their teaching visible through writing a benchmark portfolio. In subsequent offerings of the course, you might document the results of course changes with an inquiry portfolio.

You might be thinking, "Generate questions for further investigation? Document improvement over time? Looking at long-term impact of teaching changes? I don't want to become an educational researcher. I simply want to see if my students are learning what I think they are learning." This concern is common. But our model for course portfolios has been used by hundreds of teachers from numerous disciplines to provide a foundation on which to explore student learning. While these teachers had different teaching objectives and valued different forms of teaching, all of them found this process useful for thinking about their

students' learning in a structured and systematic way. For example, a professor of English observes:

Having a structure for reflecting on my course has been very useful for me. I have found that
ordinarily after I finish a class I might have some thoughts about it-what happened and what I
could do better in presenting the materials. Ideally after every semester I'd write these down,
though in reality only occasionally have I ever taken the extra effort. The course portfolio
framework has allowed me to think more systematically about my course and the activities that
were happening in the classroom. Having to write about it and then share my writing with peers
really forced me to look very closely at the things I was doing.

According to a professor of political science,

Writing a portfolio required me to be very conscious about how I was designing a syllabus, how I was evaluating students, and how I was approaching my teaching. It serves as a foundation on which my colleagues and I often start discussions about teaching and learning.

A professor of agronomy and horticulture emphasizes the variety of ways that a portfolio can be useful:

As I was describing the purpose and activities of the portfolio development profession to a colleague, I related that the process can serve many purposes, e.g., the creation of a course portfolio, documentation of teaching activities for promotion and tenure, a troubleshooting tool to assist in retooling an older or troubled course, but to me, it principally is a vehicle for an instructor to assess whether they are really teaching what they think they are teaching. I see it as more of a process than a product.

As these three teachers suggest, the process of creating a portfolio is often as valuable-or even more valuable-than the actual "product" generated in the end. While we agree that not all teachers need to be educational researchers, we do believe that if we want our students to be engaged in their learning, we ourselves need to be systematically and continually engaged in our teaching. Writing a course portfolio will help you become a better teacher, enhancing the classroom experience for current and future student learners not only in the course you are profiling but in all your courses.

Syllabus Elements

1. Course Information
   - Title of course
   - Course Number, course section, quarter, credit hours
   - Prerequisites (if any)
   - Time, location

2. Instructor Information
   - Instructor's name and title
   - Office location, office hours, office phone number, e-mail address

3. Assigned texts, required course/lab materials
   - Textbooks (titles, authors, editions)
   - Course/Lab/Shop materials--binders, spiral notebooks, dictionaries, calculator, flash drives, equipment, tools, access to computers, etc.
   - Readings or other resources such as videos, CDs, DVDs, etc. (titles, required or optional, where to locate the resources)
   - Electronic resources--web sites, listserv, newsgroup, Blackboard, etc.

4. Course Description
   - General description of the course from course outline and college catalog–from Dean.
   - Additional description of the course as instructor sees fit.

5. Statement of instructional methods
   - Instructional methods (lecture, group discussion, teamwork projects, use of online platform, etc.)

6. Course Specific Learning Outcomes
   - Course outcomes/objectives should match those on the course outline.
   - Global outcome requirements met by course
   - You may add course outcomes/objectives of your own in addition to those on the outline, but you may not omit any listed on outline.

7. Course Policies
   - Attendance/lateness policy–clearly define
   - Class participation policy –clearly define
   - Missed exams or assignments
   - Policies for dress during a lab, etc.
   - Equal Opportunity Policy
   - Computer, Cell phone, Pager usage
   - Respect
   - Student Code of Conduct
   - Weather

- Safety Policies/Procedures as pertains to lab–listed on separate handout
- A Statement Regarding Academic Honesty
  The following is an example of language you can use in your syllabi to address academic honesty.

  Plagiarism occurs when you knowingly submit someone else's ideas or words as your own. Plagiarism is an act of intentional deception that not only is dishonest, it robs you of the most important product of education---the actual learning. Should I suspect that you have plagiarized, I will talk with you one-on -one and ask you to prove that the work in question is your own. If you are found guilty of academic dishonesty, you will automatically fail that assignment. If you are caught plagiarizing again in the same quarter, you will fail this class. Cite policy in student handbook.

8. Assessment and Grading
   - Factors included in grade, how assessed and weighted
   - Grading scale –Clearly explain how percentage and/or points are to be calculated.
   - An example of how student can compute his/her grade after receiving each graded assignment.
   - Inform students when to expect turn around on graded assignments.

9. Course Schedule/Calendar (may be attached on separate sheet)
   - Schedule of daily topics to be covered
   - Due dates for all assignments -- readings, projects, etc.
   - Dates for exams, quizzes, papers, and other forms of assessment
   - Dates of required or recommended participation at special events.

10. Support Services
    - Library
    - Academic Skills Centering
    - Peer Tutoring
    - Computer Lab
    - Writing Center
    - Security
    - Bookstore

Things to consider for course construction

Ask yourself the following questions:

Schedule
-- Will it be self-paced, following a set calendar, etc.?

Calendar
-- Will you use a calendar with assignment and test due dates?

Sequencing
-- Determine the logical progression of your materials.
Will it be a linear or random order?
-- Do you want students to see all the materials in the entire course?
-- Does the content drive the organization?
-- Will you restrict them to a segment at a time?
-- Must a student master one segment before being allowed to move
to the next?
-- Must your material be taught in a step-by-step fashion?
-- Can students perform course tasks in any order?

Content Segments
-- How will you organize and split up your content?
-- Will you allow students to print content?
-- How will you present the course information?
-- What is the best way to group your information?
-- Will you use modules, weeks, lessons, units or something else?
-- Will you use an orientation, help section or general resource?

Pre-test or remediation
-- Will you pre-test students?
-- Will you offer remedial material?

Types of activities the student will be required to complete
-- Will you use written assignments, online self-tests, participation
in the discussion board, group work, etc.?

Procedure for submission of electronic assignments
-- What are the rules?
-- Can they submit multiple copies?
-- Can others view submissions?
-- Will you use drop boxes?
-- Must they use a specific format?
-- Will you specify a deadline?
-- How will you handle late submissions?
-- How will you handle technological problems?

Assignments and Class Size

- Considering your class size, how much time can you devote to reviewing assignments?
- Do you have TA support?
- Will you use peer review?
- Will you use groups?

Testing Procedures and Assessments

- What kinds of assessments will you use?
- Will they be online, proctored, etc.?
- How many assessments will you use?
- What must the student do or perform to provide sufficient evidence that they have indeed comprehended the materials?
- Will you require memorization, analysis or skill performance?
- Are the assessments low-stakes or high-stakes?
- Will you use self-assessment?
- How do you feel about cheating?
- Will your assessments be driven by your opinions on cheating? Eg, you may choose not to use multiple-choice exams, but have them write a paper.
- Will you weight your various assessments differently?

Course Communications

- What is your preferred mode of having students communicate with you? Will you use email, discussion board, etc. and how?
- What is your preferred mode of having students communicate with other students? Will they use instant messages, phone, course mail, email, discussion boards, etc. and how?

Syllabus

- Does your syllabus reflect your course structure, segments, projects and grades?

Orientation

- Have you considered an online orientation to both the online tools and your course?
- Is completion of the orientation required to access the course material?

Evaluation

- Will you provide an evaluation after each segment?
- Will you poll and survey student opinion?
- What do you want feedback on?

Bloom's Taxonomy of Cognitive Objectives

Bloom's taxonomy of cognitive objectives, originated by Benjamin Bloom and collaborators in the 1950's, describes several categories of cognitive learning.

Category	Description
Knowledge	Ability to recall previously learned material.
Comprehension	Ability to grasp meaning, explain, restate ideas.
Application	Ability to use learned material in new situations.
Analysis	Ability to separate material into component parts and show relationships between parts.
Synthesis	Ability to put together the separate ideas to form new whole, establish new relationships.
Evaluation	Ability to judge the worth of material against stated criteria

Many people also call the analysis, synthesis, and evaluations categories "problem solving."

Key Verbs
Using verbs is beneficial to writing effective learning objectives.

Behavioral Verbs Appropriate for Each Level of Blooms' Taxonomy

Knowledge

* Define	* Identify	* List	* Name
* Recall	* Recognize	* Record	* Relate
* Repeat	* Underline/Circle		

Comprehension

* Choose	* Cite examples	* Demonstrate use	* Describe
* Determine	* Differentiate	* Discriminate	* Discuss
* Explain	* Express	* Use own words	* Identify
* Interpret	* Locate	* Pick	* Report
* Restate	* Review	* Recognize	* Select
* Tell	* Translate	* Respond	* Practice
* Simulates			

Application

* Apply	* Demonstrate	* Dramatize	* Employ
* Generalize	* Illustrate	* Interpret	* Operate
* Operationalize	* Practice	* Relate	* Schedule
* Shop	* Use	* Utilize	* Initiate

Analysis

* Analyze	* Appraise	* Calculate	* Categorize
* Compare	* Conclude	* Contrast	* Correlate
* Criticize	* Deduce	* Debate	* Detect
* Determine	* Develop	* Diagram	* Distinguish
* Differentiate	* Draw conclusions	* Estimate	* Evaluate
* Examine	* Experiment	* Identify	* Infer
* Inspect	* Inventory	* Predict	* Question
* Relate	* Solve	* Test	* Diagnose

Synthesis

* Arrange	* Assemble	* Collect	* Compose
* Construct	* Create	* Design	* Develop
* Formulate	* Manage	* Modify	* Organize
* Plan	* Prepare	* Produce	* Propose
* Predict	* Reconstruct	* Set-up	* Synthesize
* Systematize	* Devise		

Evaluation

* Appraise	* Assess	* Choose	* Compare
* Critique	* Estimate	* Evaluate	* Judge
* Measure	* Rate	* Revise	* Score
* Select	* Validate	* Value	* Test

Affective Domain Taxonomy

Affective domains deal with changes in attitudes and changes in behaviors related to changes in attitudes. An example of a content areas with affective objectives would be diversity awareness and relating to peoples from different backgrounds.

Taxonomy levels indicate degree of commitment (affect)

TEMPLATE
Affective Domain Level
    Definition
        Example

1. Receiving
    Being aware of or attending to something in the environment
        Individual would read a book passage about software engineering.

2. Responding
    Showing some new behaviors as a result of experience
        Individual would answer questions about the book, read another book by the same author, read another book about software engineering

3. Valuing
    Showing some definite involvement or commitment
        The individual might voluntarily attend a software engineering lecture.

4. Organization
    Integrating a new value into one's general set of values, giving it some ranking among one's general priorities
        The individual might arrange a software engineering study group.

5. Characterization by Value
    Acting consistently with the new value
        The individual is firmly committed to the value, perhaps becoming a software engineer.

Krathwohl, D., Bloom, B., & Masia, B. (1956). Taxonomy of educational objectives. Handbook II: Affective domain. New York: David McKay.

Psychomotor Domain Taxonomy

Psychomotor objectives focus on physical and kinesthetic skills (including keyboarding, using technical instruments and other skills).

This domain is characterized by progressive levels of behaviors from observation to mastery of a physical skill.

TEMPLATE
Psychomotor Domain Level
    Definition
        Example

1. Observing
    Active mental attending of a physical event.
        The learner observes a more experienced person in his/her performance of the skill. Asked to observe sequences and relationships and to pay particular attention to the finished product. Direct observation may be supplemented by reading or watching a video. Thus, the learner may read about the topic and then watch a performance.

2. Imitating
    Attempted copying of a physical behavior.
        The learner begins to acquire the rudiments of the skill. The learner follows directions and sequences under close supervision. The total act is not important, nor is timing or coordination emphasized. The learner is conscious of deliberate effort to imitate the model.

3. Practicing
    Trying a specific physical activity over and over.
        The entire sequence is performed repeatedly. All aspects of the act are performed in sequence. Conscious effort fades as the performance becomes more or less habitual. Timing and coordination are emphasized. Here, the person has acquired the skill but is not an expert.

4. Adapting
    Fine tuning. Making minor adjustments in the physical activity in order to perfect it.
        Perfection of the skill. Minor adjustments are made that influence the total performance. Coaching often very valuable here. This is how a good player becomes a better player.

Key Verbs
    * bend * grasp * handle * operate * reach * relax * shorten
    * stretch * write * differentiate (by touch) * perform (skillfully)

101 THINGS YOU CAN DO THE FIRST THREE WEEKS OF CLASS

by Joyce Povlacs Lunde
The Teaching and Learning Center - University of Nebraska - Lincoln

Introduction

Beginnings are important. Whether it is a large introductory course for freshmen or an advanced course in the major field, it makes good sense to start the semester off well. Students will decide very early--some say the first day of class--whether they will like the course, its contents, the teacher, and their fellow students.

The following list is offered in the spirit of starting off right. It is a catalog of suggestions for college teachers who are looking for fresh ways of creating the best possible environment for learning. Not just the first day, but the first three weeks of a course are especially important, studies say, in retaining capable students. Even if the syllabus is printed and lecture notes are ready to go in August, most college teachers can usually make adjustments in teaching methods as the course unfolds and the characteristics of their students become known.

These suggestions have been gathered from UNL professors and from college teachers elsewhere. The rationale for these methods is based on the following needs: to help students make the transition from high school and summer activities to learning in college; to direct students' attention to the immediate situation forlearning--the hour in the classroom; to spark intellectual curiosity--to challenge students; to support beginners and neophytes in the process of learning in the discipline; to encourage the students' active involvement in learning; and to build a sense of community in the classroom.

Here, then, are some ideas for college teachers for use in their courses in the new academic year:

Helping Students Make Transitions

1. Hit the ground running on the first day of class with substantial content.

2. Take attendance: roll call, clipboard, sign in, seating chart.

3. Introduce teaching assistants by slide, short presentation, or self-introduction.

4. Hand out an informative, artistic, and user-friendly syllabus.

5. Give an assignment on the first day to be collected at the next meeting.

6. Start laboratory experiments and other exercises the first time lab meets.

7. Call attention (written and oral) to what makes good lab practice: completing work to be done, procedures, equipment, clean up, maintenance, safety, conservation of supplies, full use of lab time.

8. Give a learning style inventory to help students find out about themselves.

9. Direct students to the Academic Success Center for help on basic skills.

10. Tell students how much time they will need to study for this course.

11. Hand out supplemental study aids: library use, study tips,supplemental readings and exercises.

12. Explain how to study for the kind of tests you give.

13. Put in writing a limited number of ground rules regarding absence, late work, testing procedures, grading, and general decorum, and maintain these.

14. Announce office hours frequently and hold them without fail.

15. Show students how to handle learning in large classes and impersonal situations.

16. Give sample test questions.

17. Give sample test question answers.

18. Explain the difference between legitimate collaboration and academic dishonesty; be clear when collaboration is wanted and when it is forbidden.

19. Seek out a different student each day and get to know something about him or her.

20. Ask students to write about what important things are currently going on in their lives.

21. Find out about students' jobs; if they are working, how many hours a week, and what kind of jobs they hold.

Directing Students' Attention

22. Greet students at the door when they enter the classroom.

23. Start the class on time.

24. Make a grand stage entrance to hush a large class and gain attention.

25. Give a pre-test on the day's topic.

26. Start the lecture with a puzzle, question, paradox, picture, or cartoon on slide or transparency to focus on the day's topic.

27. Elicit student questions and concerns at the beginning of the class and list these on the chalkboard to be answered during the hour.

28. Have students write down what they think the important issues or key points of the day's lecture will be.

29. Ask the person who is reading the student newspaper what is in the news today.

Challenging Students

30. Have students write out their expectations for the course and their own goals for learning.

31. Use variety in methods of presentation every class meeting.

32. Stage a figurative "coffee break" about twenty minutes into the hour; tell an anecdote, invite students to put down pens and pencils, refer to a current event, shift media.

33. Incorporate community resources: plays, concerts, the State Fair, government agencies, businesses, the outdoors.

34. Show a film in a novel way: stop it for discussion, show a few frames only, anticipate ending, hand out a viewing or critique sheet, play and replay parts.

35. Share your philosophy of teaching with your students.

36. Form a student panel to present alternative views of the same concept.

37. Stage a change-your-mind debate, with students moving to different parts of the classroom to signal change in opinion during the discussion.

38. Conduct a "living" demographic survey by having students move to different parts of the classroom: size of high school, rural vs. urban, consumer preferences.

39. Tell about your current research interests and how you got there from your own beginnings in the discipline.

40. Conduct a roleplay to make a point or to lay out issues.

41. Let your students assume the role of professional in the discipline: philosopher, literary critic, biologist, agronomist, political scientist, engineer.

42. Conduct idea-generating brainstorming sessions to expand horizons.

43. Give students two passages of material containing alternative views to compare and contrast.

44. Distribute a list of the unsolved problems, dilemmas, or great questions in your discipline and invite students to claim one as their own to investigate.

45. Ask students what books they read over summer.

46. Ask students what is going on in the state legislature on this subject which may affect their future.

47. Let your students see the enthusiasm you have for, your subject and your love of learning.

48. Take students with you to hear guest speakers or special programs on campus.

49. Plan a "scholar-gypsy" lesson or unit which shows students the excitement of discovery in your discipline.

Providing Support

50. Collect students' current telephone numbers and addresses and let them know that you may need to reach them.

51. Check out absentees. Call or write a personal note.

52. Diagnose the students' pre-requisite learning by a questionnaire or pre-test and give them the feedback as soon as possible.

53. Hand out study questions or study guides.

54. Be redundant. Students should hear, read, or see key material at least three times.

55. Allow students to demonstrate progress in learning: summary quiz over the day's work, a written reaction to the day's material.

56. Use non-graded feedback to let students know how they are doing: post answers to ungraded quizzes and problem sets, exercises in class, oral feedback.

57. Reward behavior you want: praise, stars, honor roll, personal note.

58. Use a light touch: smile, tell a good joke, break test anxiety with a sympathetic comment.

59. Organize. Give visible structure by posting the day's "menu" on chalkboard or overhead.

60. Use multiple media: overhead, slides, film, videotape, audiotape, models, sample materials.

61. Use multiple examples, in multiple media, to illustrate key points and important concepts.

62. Make appointments with all students (individuallymn or in small groups).

63. Hand out wallet-sized telephone cards with all important telephone numbers listed: office, department, resource centers, teaching assistant, lab.

64. Print all important course dates on a card that can be handed out and taped to a mirror.

65. Eavesdrop on students before or after class and join their conversation about course topics.

66. Maintain an open lab gradebook, with grades kept current, during lab time so students can check their progress.

67. Check to see if any students are having problems with an academic or campus matter and direct those who are to appropriate offices or resources.

68. Tell students what they need to do to receive an "A" in you, course.

69. Stop the world to find out what your students are thinking, feeling, and doing in their everyday lives.

Encouraging Active Learning

70. Having students write something.

71. Have students keep three-week three-times-a-week journals in which they comment, ask questions, and answer questions about course topics.

72. Invite students to critique each other's essays or short answers on tests for readability or content.

73. Invite students to ask questions and wait for the response.

74. Probe students responses to questions and their comments.

75. Put students into pairs or "learning cells" to quiz each other over material for the day.

76. Give students an opportunity to voice opinions about the subject matter.

77. Have students apply subject matter to solve real problems.

78. Give students red, yellow, and green cards (made of posterboard) and periodically call for a vote on an issue by asking for a simultaneous show of cards.

79. Roam the aisles of a large classroom and carry on running conversations with students as they work on course problems (a portable microphone helps).

80. Ask a question directed to one student and wait for an answer.

81. Place a suggestion box in the rear of the room and encourage students to make written comments every time the class meets.

82. Do oral, show-of-hands, multiple choice tests for summary, review, and instant feedback.

83. Use task groups to accomplish specific objectives.

84. Grade quizzes and exercises in class as a learning tool.

85. Give students plenty of opportunity for practice before a major test.

86. Give a test early in the semester and return it graded in the next class meeting.

87. Have students write questions on index cards to be collected and answered the next class period.

88. Make collaborative assignments for several students to work on together.

89. Assign written paraphrases and summaries of difficult reading.

90. Give students a take-home problem relating to the day's lecture.

91. Encourage students to bring current news items to class which relate to the subject matter and post these on a bulletin board nearby.

Building Community

92. Learn names. Everyone makes an effort to learn at least a few names.

93. Set up a buddy system so students can contact each other about assignments and coursework.

94. Find out about your students via questions on an index card.

95. Take pictures of students (snapshots in small groups, mugshots) and post in classroom, office or lab.

96. Arrange helping trios of students to assist each other in learning and growing.

97. Form small groups for getting acquainted; mix and form new groups several times.

98. Assign a team project early in the semester and provide time to assemble the team.

99. Help students form study groups to operate outside the classroom.

100. Solicit suggestions from students for outside resources and guest speakers on course topics.

Feedback on Teaching

101. Gather student feedback in the first three weeks of the semester to improve teaching and learning.

SITUATED SYLLABUS

VR emphasizes the relationship between participant and environment. Theories which recognize that content and context are inextricably interrelated (intertwingled) are called *situated*. In AI, it's called situated automata; in industrial engineering, it's ecological psychology; in mathematics, it's general systems; in psychology, its Gestalt.

In traditional classes, what you are expected to learn is defined ahead of time. In this class, we will respond to events dynamically. This is called *situated learning*. The theory is that the dynamic context impacts what and how we learn. As a consequence, the syllabus will change dynamically over time, in response to class needs and evolving understandings.

In VR, you enter first into a void; you then load a constructed database that is the world. In physical reality, you are born into a world that is already full. The difference is that VR demands active participation, inside and out. VR emphasizes *constructivism*, the theory that mind and body coparticipate to construct our experience of reality. In education, constructivism says that students build their own understanding through experience. This means that students must actively engage in and work with the subject matter, and that will be the general rule for this class. It also means that each student will construct an essentially different understanding of VR. In VR software terms, we all live in divergent realities.

Finally, the essence of VR is direct interaction with information. Right now, universities use a *symbolic mediation* strategy almost exclusively. To learn something, we translate it into what it is not (symbols), study the symbols (words, formulae, programs), then reconstruct the something. In this class, we will attempt to engage in direct learning by constantly anchoring the symbols we use with the reality they refer to.

Virtual World Development is a graduate seminar. Class grading will be based on individual understanding, participation, and growth. You will be required to attend class and to develop or improve a skill related to the design and construction of virtual worlds. Each student will probably develop a different skill, at a different rate, with different criteria of success.

Students who are concerned about this approach to teaching should meet with the instructor. An option of individual performance contracts is available.

POSSIBLE CONTENT (no particular order, not necessarily complete):

VR software architectures and functionalities

VR varieties and taxonomies

Theory of Inclusion

Systems oriented programming
- parallelism
- modularity
- partitioning

Situated agents (entities)
- reactivity and responsiveness
- dramatic theory
- embedded narrative
- dispositions
- autonomy

Building virtual worlds
- software tools
  - CAD
  - dynamics
  - animation
  - scientific visualization
- techniques
  - enumeration
  - decomposition
  - CSG
  - boundary models
  - sweeps
- design
  - participation who/what/why
  - physiological constraints
  - tight coupling

Display
- rendering choices
- adaptive refinement
- viewpoint control, navigation

Abstraction
- varieties of space
- networks
- form abstraction
- application specific construction kits
- semantics

Computation
- pattern matching
- constraints and possibility spaces
- inference
- history and statistics
- resource management
- editors
- molecular programming

Inclusive tools
- cursors
- backdrops and foredrops
- virtual body
- wand
- inhabitation
- artificial life

Multiple participation
- inconsistency maintenance
- uniqueness
- negotiation

Experiential mathematics
- logic blocks
- boundary mathematics
- spatial algebra

Teleoperation and telepresence
- presence
- out-of-body experience
- physical and sensory extension

Physiological modeling
- sensory models
- physical constraints
- plasticity

Cognitive modeling
- information processing
- gestalt
- situated intelligence

PROJECTS

Each student is expected to contribute to our state of knowledge about VR. Projects document this contribution. Project work will be presented to the class, so that the class can review the work. This list is suggestive:

Design and/or build a VR tool
- wand
- physiological model
- virtual body
- virtual community
- editor for entities
- form abstraction
- logic blocks
- mapping tools
- projection tools
- navigation tools
- divergent worlds
- conversational programming
- music

Research and design a VR language
- behavioral modeling
- construction from inside VR
- algebraic specifications
- gesture languages
- virtual machines
- sound

Write and publish a VR article
- VEOS
- entities
- social implications

Develop VR operating system tools
- FERN
- Linda
- device drivers
- graphics drivers
- world building maintenance tools
- UM

Explore an important VR issue
- access
- cognitive plasticity
- ecstasy machines
- cultural bias
- ownership and legalities
- philosophy and metaphysics

Curriculum Exercises

1. Write down three questions that you would most like answered by this class.

2. Write down two things that almost everybody in the class will understand by June.

3. Write down the one thing that most concerns or worries you about taking this class.

==========

"HCI Overview"

Make a chart/list of the major areas of HCI. Include what you think the subject is about, and the areas you have had experience in.

"Curriculum Engineering"

Draft your ideas of the curriculum for this course. What topics will we study, what activities will we do, how shall we determine success? Include preferred and requested topics, and what to avoid. I will build a group map focusing on general content and specific interests.

Emphasis?	machines	computer science
	humans	psychology, physiology
	design	art
	human groups	sociology

JUST WHAT IS VIRTUAL REALITY ANYWAY?

What are the defining characteristics of virtual reality, of a virtual reality system? Suggest techniques for "measuring" each characteristic.

Develop a taxonomy (hierarchy, state space) of partial and complete VR systems.

computer-generated, television or live image?
inclusion or partial immersion or watching 3D?
wearing a computer?
how many dimensions? multisensory?
input and output bandwidth?
available system resources?
responsiveness and timeliness?
physical/virtual mixture?
degree of presence and physical remoteness?
occlusive, overlay or annotated?

Consider (some of) these issues:

bandwidth
sensory modality
degree of coupling and feedback
input and sensor types
output and display types
interactivity
realtime responsiveness
meaning
human physiology
presence
telepresence
dimensionality
realism
autonomy
formality
anthropomorphism

Classify (some of) the following systems in your taxonomy:

Dataglove
Heads-up display
Inclusive display
flight simulator
computer animation
stereo sound
Nintendo games
SIMNET
computer aided design (CAD)
Landsat database
remote controlled robot
voice recognition
holographs
email
command line computer interfaces
desktop metaphor (WIMP) computer interfaces
electron microscope
the physical world
television
telephone
automobiles
drawings
sculpture
thinking
meditation
dreaming
books and reading
photographs
movies
Disneyland
this assignment
[add your own]

Wonderful Computer Science Books

Every subfield of Computer Science has several journals, dozens of text books, and hundreds of technical books. The web makes this situation much worse by providing course notes and technical discussion from hundreds or thousands of practitioners. Below is the single best book or two (IMHO) in most of areas covered in this class.

Data Structures, Algorithms and Programming

H. Abelson and G.J. Sussman (1996)
Structure and Interpretation of Computer Programs, Second Edition
McGraw-Hill. ISBN 0-07-000484-6

Comprehensive Reference on Algorithms

T.H. Cormen, C.E. Leiserson, and R.L. Rivest (1990)
Introduction to Algorithms
MIT Press. ISBN 0-07-013143-0

Very High-level Programming

S. Wolfram (1996)
The Mathematica Book, Third Edition
Wolfram Media, Cambridge U. Press. ISBN 0-521-58889-8

Mathematical Models of Data Structures

Z. Manna and R. Waldinger (1985)
The Logical Basis for Computer Programming, Volume 1
Addison-Wesley. ISBN 0-201-18260-2

Old-style Procedural Algorithms

R. Sedgewick (1988)
Algorithms in C++, Third Edition
Addison-Wesley. ISBN 0-201-35088-2

Theory of Computation Complexity

J.E. Savage (1998)
Models of Computation
Addison-Wesley. ISBN 0-201-89539-0

Programming Theory

N.D. Jones (1997)
Computability and Complexity from a Programming Perspective
MIT Press. ISBN 0-262-10064-9

Philosophy of Computation

B.C. Smith (1996)
On the Origin of Objects
MIT Press. ISBN 0-262-69209-0

Programming Languages

B.J. MacLennan (1999)
Principles of Programming Languages, Third Edition
Oxford. ISBN 0-19-511306-3

M.L. Scott (2000)
Programming Language Pragmatics
Morgan-Kauffman. ISBN 1-55860-442-1

Computer Architecture

J.L. Hennessy and D.A. Patterson (1996)
Computer Architecture: A Quantitative Approach, Second Edition
Morgan-Kaufmann. ISBN 1-55860-329-8

Compilers

S.S. Muchnick (1997)
Advanced Compiler Design and Implementation
Morgan-Kaufmann. ISBN 1-55860-320-4

Discrete Mathematics

W.K. Grassmann and J. Tremblay (1996)
Logic and Discrete Mathematics: A Computer Science Perspective
Prentice-Hall. ISBN 0-13-501206-6

Understanding Computing in Simple Language

R. P. Feynman (A.J.G. Hey and R.W. Allen, eds) (1996)
Feynman Lectures on Computation
Addison-Wesley. ISBN 0-201-48991-0

Computer Science Culture

ACM Turing Award Lectures: The First Twenty Years (1987)
ACM Press, Addison-Wesley. ISBN 0-201-07794-9

Seminal Algorithms Text (recently updatedl)

D.E. Knuth (1997, 1997, 1998)
Fundamental Algorithms, Third Edition, volume 1 of *The Art of Computer Programming*
Seminumerical Algorithms, Third Edition, v. 2 of *The Art of Computer Programming*
Sorting and Searching, Third Edition, v. 3 of *The Art of Computer Programming*
Addison-Wesley, ISBNS 0-201-89683-4, 0-201-89684-2, 0-201-89685-0

Undergraduate Introduction to Data Structures and Algorithms

F.M. Carrano, P. Helman and R. Veroff (1998)
Data Abstraction and Problem Solving with C++
Addison-Wesley. ISBN 0-201-87402-4

Mark Allen Weiss (1998)
Data Structures and Problem Solving Using JAVA
Addison-Wesley ISBN 0-201-54991-3

Historical First Textbooks

Nicklaus Wirth (1976)
Algorithms + Data Structures = Programs
Prentice-Hall ISBN 0-13-022418-9

A.V. AHo, J.E. Hopcroft and J.D. Ullman (1983)
Data Structures and Algorithms
Addison-Wesley ISBN 0-201-00023-7

Computer Art

A.M. Spalter (1999)
The Computer in the Visual Arts
Addison-Wesley. ISBN 0-201-38600-3

TEACHING FOR INNOVATION

TOPIC 6. STRUCTURING CONTENT

Lesson Plan Outline

How to Write Clear Objectives

Matching Objectives to Learning Styles

Teaching Examples (Bricken):

Formal: Proof Techniques, An Extended Example

Programming: A Small Interpreted Language

Programming: Pseudocode Assignment Package

LESSON PLAN

1) Outcome–tell the students the skills and information they will learn by the end of the lesson. How does this learning fit into the big picture of this content?

2) Set the stage–draw relationship between new and previous learning. Statements starting with “remember when we talked about….” Draw analogies all students can relate to.

3) Purpose–why should the learner listen to you today?? How will this learning benefit him/her on the job?

4) Content (Concept/idea OR Process/demonstration)

INSTRUCTOR BEHAVIORS
Strategies used to teach lesson taking into account students’ learning styles

STUDENT BEHAVIORS
Students engage in activities, assignments, feedback, discussion, etc.

5) Feedback –-instructor preplanned questions, CATS, short assignments–every 15-20 minutes

6) Supplies/Equipment–

7) Practice–Planned assignments, activities, demonstrations to initially be monitored by instructor.

8) Evaluation–formal (quiz, test) informal (feedback)

Factors to consider in pedagogy design

* Motivation for the student: Why learn this? Where and when is this used? What are the payoffs for learning?

* Motivation for the faculty: What are the reasons for developing this course? How can you create the best learning experience for the students?

* Time: Preparing all materials ahead of time requires significant time commitment.

* Pedagogical considerations: What pedagogical models and learning therories will you be incorporating into the teaching of the material?

* Orientation: Help the student adjust to the environment or content being taught, i.e., preview, objectives, overviews, summaries, prerequisites, and schedule.

* Information: The content the student needs to master, i.e., facts and evidence, demonstrations and skill steps, definitions and examples, evidence and cases, control and explain events, recall data, perform tasks, identify concepts and infer outcomes.

* Application: How will the student demonstrate learning, i.e., practice, prompting, feedback, remediation?

* Evaluation: How will you assess what the student has learned, whether the content was relevant, or whether the instructional method was appropriate?

* Technical Competency: Are you comfortable with the technology used to deliver your course content? Will you need training or support?

How to Write Clear Objectives

Jones, 1997 – "Clear objectives can help the instructor design lessons that will be easier for the student to comprehend and the teacher to evaluate".

Lohr, no date – "A properly written objective tells you what specific knowledge, skill, or attitude is desired and what method of instruction and criteria for learner achievement are required."

Writing clear course objectives is important because:

* Objectives define what you will have the students do.
* Objectives provide a link between expectations, teaching and grading.

Questions you need to think about

* Who are your students? Freshman? Senior?
  A mix of different prior knowledge and experience?
* Is this course a general education course or required for the major?

The A.B.C.D. method

The ABCD method of writing objectives is an excellent starting point for writing objectives (Heinich, et al., 1996). In this system, "A" is for audience, "B" is for behavior, "C" for conditions and "D" for degree of mastery needed.

1. Audience – Who? Who are your learners?

2. Behavior – What? What do you expect them to be able to do? This should be an overt, observable behavior, even if the actual behavior is covert or mental in nature. If you can't see it, hear it, touch it, taste it, or smell it, you can't be sure your audience really learned it.

3. Condition – How? Under what circumstances or context will the learning occur? What will the student be given or already be expected to know to accomplish the learning?

4. Degree – How much? How much will be accomplished, how well will the behavior need to be performed, and to what level? Do you want total mastery (100%), do you want them to respond correctly 80% of the time, etc. A common (and totally non-scientific) setting is 80% of the time.

Examples of Well-Written Objectives

Audience (A), Behavior (B), Condition (C) and Degree of Mastery (D).

Cognitive Objectives (comprehension level)

(C) Given a paragraph in a newspaper article,
(A) the student
(B) will be able to accurately identify the grammatical subject of each sentence and explain his or her decision
(D) for all sentences given.

Cognitive Objective (application level)

(C) Given a foreign language sentence written in the past or present tense,
(A) the student
(B) will be able to rewrite the sentence in future tense
(C) with no grammatical errors

Cognitive Objective (problem solving/synthesis level)

(C) Given a set of current meteorological conditions taken from a weather station
(A) the student
(B) will write a weather forecast
(D) covering the next six hours.

Psychomotor

(C) Given an geometric object in Photoshop software,
(A) the student
(B) will be able to use the computer mouse and lasso tool to trace a usable outline
(D) which can be used to define the object for a montage.

Notes on Objective Writing

When reviewing example objectives above, you may notice a few things.

1. As you move up the "cognitive ladder," it can be increasingly difficult to precisely specify the degree of mastery required.

2. Affective objectives are difficult for many instructors to write and assess. They deal almost exclusively with internal feelings and conditions that can be difficult to observe externally.

3. It's important to choose the correct key verbs to express the desired behavior you want students to produce. See the pages on a page on cognitive objectives (Blooms' Taxonomy), affective objectives and psychomotor objectives to see examples of key words for each level.

Typical Problems Encountered When Writing Objectives

TEMPLATE
Problems
    Error Types
        Solutions

Too vast/complex
    The objective is too broad in scope or is actually more than one objective.
        Use the ABCD method to identify each desired behavior or skill in order to break objectives apart.

No behavior to evaluate
    No true overt, observable performance listed. Many objectives using verbs like "comprehend" or "understand" may not include behaviors to observe.
        Determine what actions a student should demonstrate in order for you to know of the material has been learned.

Only topics are listed
    Describes instruction, not conditions. That is, the instructor may list the topic but not how he or she expects the students to use the information.
        Determine how students should use the information presented. Should it be memorized? Used as background knowledge? Applied in a later project? What skills will students need?

Vague Assignment Outcomes
    The objective does not list the correct behavior, condition, and/or degree, or they are missing. Students may not sure of how to complete assignments because they are lacking specifics.
        Determine parameters for assignments and specify them for students.

Matching objectives to learning styles

Here are three common theoretical approaches to learning objectives.

* The behavioral approach to learning objectives is consistent with behavioral perspectives on learning and with teacher-centered teaching styles.

* The cognitive approach to learning objectives is consistent with cognitive perspectives on learning and an emphasis on teaching and learning knowledge and concepts.

* The constructivist approach to learning objectives is consisent with constructivism and situated learning frameworks and is more likely to accompany student-centered approaches to teaching.

Behavioral learning objectives

Behavioral objectives focus on identifying measurable, observable student behavior by specifying the following:

* conditions under which behavior will be performed
* the student behavior (using a verb to describe a measurable behavior)
* acceptable level or criteria for success

For example, after reading this page, you should be able to accurately name and define three different kinds of learning objectives, 100% of the time.

Cognitive learning objectives

Cognitive learning objectives are broader and less measurable. They may better reflect the goals some professors have for their course. Assessment is more challenging with this approach. Bloom's (1956, 2001) taxonomy of cognition is often used to help generate cognitive objectives:

* knowledge (remember and recall)
* comprehension (understand)
* application (use of concept in a new situation)
* analysis (break something down into its parts, interpret)
* synthesis (generate something new applying the ideas)
* evaluation (make judgments about value, appropriateness, other attributes)

For example, after reading this page, you should be able to identify the intended learning style of your course and write corresponding learning objectives. (application)

Constructivist learning objectives

The goals of a constructivist teacher are not to cover the curriculum, but instead to engage students as active learners constructing their own knowledge and beliefs within a content domain. Constructivist teachers start with very broad learning objectives and may even negotiate with the class to identify more specific learning goals. Constructivist teachers gather resources and set the stage to challenge learners to explore their existing beliefs, expose them to new ideas, and assign tasks which encourage learners to re-evaluate, re-define, and apply their emerging understandings. For example:

Our starting objectives are:

* consider your personal history of learning experiences
* explore and react to new theories and examples of teaching and learning
* reconsider your original beliefs
* form teams, plan, and develop a real world learning object for a client where the learning object is consistent with your new revised perspective on how technology can enhance learning
* justify your design choices based on the principles you have learned

Proof Techniques, an Extended Example

Here is an example from relational calculus to illustrate each of the four methods of proof (case-analysis or truth tables, natural deduction, resolution, and boundary logic) The example can be viewed as a knowledge-base query. A ***knowledge-base*** (KB) is a collection of ***facts*** (which contain no variables) and ***rules*** (which contain variables, and are usually stated in an if...then... format). Both the fact-base and the rule-base have been greatly simplified for this example. The deductive processes are essentially the same, regardless of the complexity of the knowledge-base.

A Relational Calculus (Database Query) Example

Facts:

```
F1.   (George  is-the-father-of  Harry)
F2.   (Rita    is-the-sister-of  Harry)
F3.   (Rita    is-never-married)
F4.   (Harry   is-a-male)
```

Rules:

```
R1.   If  (_1_  is-the-father-of  _3_)
          and (_2_  is-the-sister-of  _3_)
              then (_1_  is-the-father-of  _2_)

R2.   If  (_4_  is-the-father-of  _5_)
          and (_5_  is-a-male)
              then (_5_  is-the-son-of  _4_)

R3.   If  (_6_  is-the-son-of  _7_)
            then (_6_  has-same-last-name-as  _7_)

R4.   If  (_8_  is-the-father-of  _9_)
          and (_9_  is-never-married)
              then (_9_  has-same-last-name-as  _8_)

R5.   If  (_10_  has-same-last-name-as  _11_)
          and (_10_  has-same-last-name-as  _12_)
              then (_11_  has-same-last-name-as  _12_)

R6.   If  (_13_ has-same-last-name _14_)
            then  (_14_ has-same-last-name _13_)
```

Query:

```
Q.         (Harry has-same-last-name-as  Rita)
```

Abbreviations:

```
George                                =     g
Rita                                  =     r
Harry                                 =     h
(_1_  is-the-father-of  _2_)          =     1F2
(_1_  is-the-sister-of  _2_)          =     1T2
(_1_  is-the-son-of  _2_)             =     1S2
(_1_  has-same-last-name-as  _2_)     =     1L2
(_1_  is-a-male)                      =     1M
(_1_ is-never-married)                =     1N

Variables will be integers      =   {1, 2, 3, ...}
```

Abbreviated knowledge-base:

```
F1.   gFh
F2.   rTh
F3.   rN
F4.   hM

R1.   if   1F3   and    2T3   then   1F2
R2.   if   4F5   and    5M    then   5S4
R3.   if   6S7   then   6L7
R4.   if   8F9   and    9N    then   9L8
R5.   if  10L11  and   10L12  then  11L12       transitivity
R6.   if  13L14  then  14L13                    commutativity

Q0.   hLr
```

This particular example was designed with these objectives in mind:

1. Intuitive semantics, easy for a human to understand
2. Tractable size but enough to illustrate both natural and algorithmic processes
3. Simple but non-trivial proof in natural deduction
4. Easy forward-chaining proof in case-analysis (as a consequence of this, a complex backward chaining proof)
5. Surprising proof in algorithmic resolution
6. Illustrative proof of minimal boundary techniques, including more complex set techniques.
7. Difficulty general transitivity and commutativity rules
8. Tricky and subtle knowledge engineering issues.

Note: in pattern-matching systems, there is no substantive difference between algebraic functions (ie. functions which are not evaluated) and relations.

Natural Deduction

The natural deduction approach is to ***use reason*** to show that Harry and Rita have the same last name because George is their common father and Rita has never married. We show that George is the father of both Harry and Rita, then we show that George has the same last name as both Harry and Rita, then we conclude that Harry and Rita have the same last name. Although the logic is clear, the syntactic transformations to get the rules to confirm the logic require the additional skill of pattern-matching through unification.

```
Show  gFh         F1.   gFh          given
Show  gFr         F2.   rTh
                  R1.   gFh and rTh  therefore  gFr
Show  gLh         R3.   gFh and hM   therefore  hSg
                  R3.   hSg          therefore  hLg
                  R6.   hLg          therefore  gLh
Show  gLr         R4.   gFr and rN   therefore  rLg
                  R6.   rLg          therefore  gLr
Show  hLr         R5.   gLh and gLr  therefore  hLr
```

Case-analysis and Chaining

Truth tables list all possible facts. In a KB, rules can be seen as sets of facts that have yet to be enumerated. The identifying characteristic of case-analysis is that no variables are included in the final form of rules or queries. One approach is to substitute all possible variable bindings into the rules in all possible combinations. Since we have three people {George, Rita, Harry} in the KB, and all variables refer to these three people, each variable has 3 cases, and each rule would have 3^n cases, where n is the number of different variables in the rule. In the example, this would generate (3^3 + 3^2 + 3^2 + 3^2 + 3^3 + 3^2) = 90 rule cases for the six rules.

A more efficient procedure would be to use the known facts to constrain the generation of cases. We begin with the known facts and then use the rules to (indiscriminately) generate all the other possible facts that are consistent with both the initial facts and the rules. The forward generation of facts from initial conditions and rules is called ***forward-chaining***. We attempt to unify each fact with the premise of each rule; when unification is successful, the conclusion of the rule is asserted as a new fact.

The order of enumeration of facts (the ***enumeration strategy***) is a significant issue. The order in which facts are applied to rules determines which new facts get enumerated first. Since new facts themselves may trigger applications of rules, a choice can be made between ***depth-first*** enumeration (following new facts first) or ***breadth-first*** enumeration (following old facts first). Often a single fact may unify with one premise of a rule which requires two or more facts to fulfill its premise. In this case, a new, shorter rule is asserted.

Below we use the following strategy: first all new facts are generated, then they are in turn used to generate more facts. Duplications has been suppressed (this is called an ***occurs-check***). Using the current facts to generate more facts is called a ***set-of-support*** strategy, since the set of known facts support the conclusions.

```
F5.   F1+F2, R1:        gFr
F6.   F1+F4, R2:        hSg      no other rules unify, use new facts
F7.   F5+F3, R4:        rLh
F8.   F6,    R3:        hLg
F9.   F7,    R6:        hLr      QED
```

Note that this algorithmic proof is shorter than the natural deduction proof. It is still not optimal, since step F8 was unnecessary. Algorithmic proof is always committed to following a blind strategy, trading thought and efficiency for ease of implementation. There is a general computational heuristic here: almost always it is better to implement blind brute force rather than subtle computational intelligence. The corollary to this heuristic is that brute force only works with the appropriate data structure. It is almost always better to apply design intelligence to the representation of a problem than to the algorithm.

Another strategy is to use the query to generate all possible queries stemming backwards from the target query, until the existing facts terminate the search. Queries are matched with the conclusions of rules; the premises of these rules are then the new queries. This technique is called ***backward-chaining***. We first generate queries which can be answered by single facts, then queries which require more than one fact, finally trying to bind queries which contain variables to the initial fact base. The example follows:

```
Q0.                          hLr    ?
Q1.   Q0,  R3:               hSr    ?
Q2.   Q0,  R6:               rLh    ?
Q3.   Q2,  R3:               rSh    ?     No other simple queries
Q4.   Q0,  R4:     rFh  and  hN     ?
Q5.   Q0,  R5:    20Lh  and 20Lr    ?     Introduce new variable numbers
Q6.   Q1,  R2:     rFh  and  hM     ?     =>  rFh    using F4
Q7.   Q2,  R5:    21Lr  and 21Lh    ?     duplicate of Q5
Q8.   Q3,  R2:     hFr  and  rM     ?
Q9.   Q6,  R1:     rF22 and  hT22   ?     No more bindings or ground Qs
Q10.  Q5a, R3:               23Sh   ?     Begin using variable Qs
Q11.  Q5b, R3:               24Sr   ?
Q12.  Q10, R2:     hF25 and 25M     ?
Q13.  Q11, R2:     rF26 and 26M     ?
Q14.  Q5a, R6:               hL27   ?
Q15.  Q5b, R6:               rL28   ?
Q16.  Q14, R3:               hS29   ?
Q17.  Q15, R3:               rS30   ?
Q18.  Q16, R2:    31Fh  and  hM     ?     => 31Fh    using F4
Q19.  Q17, R2:    32Fr  and  rM     ?
Q20.  Q18, F1:               gFh    ?     bind 31 to g using fact F1
```

At this point we have back-chained to an initial fact, `gFh`. Reversing the logic, this fact combined with F4 and R2 (see line Q18) answer line Q16, binding variable 29 to `g`. This answers Q14, binding 27 to `g`. Q14 answers the first part of Q5 (that is Q5a), binding 20 to `g`, and leaving the query sequence below as Q5b. (While we are at an interrupt, note that if the below sequence fails, the queries would pick up where they left off, at Q12 where 25 would bind to `h` using F4. This would create the query `hFh` ? and so on)

```
Q21.  Q5b:                   gLr    ?
Q22.  Q21, R3:               gSr    ?
Q23.  Q21, R6:               rLg    ?
Q24.  Q23, R3:               rSg    ?     No other simple queries
Q25.  Q21, R4:     rLg  and  gN     ?
Q26.  Q21, R5:    33Lg  and 33Lr    ?
Q27.  Q22, R2:     rFg  and  gM     ?
Q28.  Q23, R4:     gFr  and  rN     ?     =>  gFr    using F3
Q29.  Q23, R5:    34Lr  and 34Lg    ?
Q30.  Q28, R1:     gF35 and  rT35   ?     No other grounded queries
Q31.  Q30, F1:     gFh  and  rTh    ?     Bind 35 to h using F1
Q32.  Q31, F2:               rTh    ?     =>  True   using F2
```

We have now reached a final conclusion, since all queries have been answered. Reconstructing the path in reverse order:

```
Q32:        rTh
Q31:        Q32 and gFh
Q30:        Q31
            R1              thus gFr
Q28:        Q30 and rN
            R4              thus rLg
Q23:        Q28
            R6              thus gLr
Q21:        Q23

Q20:        gFh
Q18:        Q20 and hM
            R2              thus hSg
Q16:        Q18
            R3              thus hLg
Q14:        Q16
            R6              thus gLh

Q5:         Q14 and Q21
            R5              thus hLr
Q0:         Q5              QED
```

Note that this proof is similar to the natural deduction, and not as direct as the forward-chaining proof. These differences are an artifact of the particular KB, and are not general. Some KBs are particularly efficient for forward-chaining and some are particularly efficient for backward-chaining. In general, which method is best depends on the specific query, on the particular KB, and on the way in which each rule is formulated (see the **Addendum**). Usually the methods need to be mixed. The resolution technique accomplishes this mixing.

Resolution

In resolution, the KB is converted into sets of clauses. A ***clause*** is a set of both positive or negative atoms joined by disjunction. A KB is a set of clauses. New clauses are added by matching and deleting positive and negative atoms which unify across two clauses. For instance, the logical form `(if A then B)` is converted into the equivalent form `((not A) or B)`, which then is turned into a set of atoms {~A, B}. Resolution looks like this:

{~C, A} resolve-with {~A, B} ==> add {~C, B}

This can be read for logic as `((C implies A) and (A implies B) therefore (C implies B))`. Since resolution is an algorithm, we proceed down the list of clauses in a linear fashion. The query is negated, and we hope to resolve it with an assertion of the positive fact to end the resolution with an empty clause. This looks like:

{A} resolve-with {~A} ==> { }

Several resolution strategies are possible, based on the structure of each clause. For instance facts (clauses with single positive atoms) could be resolved first. Or clauses with single atoms regardless of polarity could be resolved first. Another strategy might be to resolve all instance of a particular relation first. The strategy used below is to resolve all singular clauses first.

```
F1.   {gFh}
F2.   {rTh}
F3.   {rN}
F4.   {hM}
Q.    {~hLr}

R1.   {~1F3, ~2T3, 1F2}            if  1F3  and   2T3  then  1F2
R2.   {~4F5, ~5M, 5S4}             if  4F5  and   5M   then  5S4
R3.   {~6S7, 6L7}                  if  6S7            then  6L7
R4.   {~8F9,  ~9N, 9L8}            if  8F9  and   9N   then  9L8
R5.   {~10L11, ~10L12, 11L12}      if 10L11 and  10L12 then 11L12
R6.   {~13L14, 14L13}              if 13L14            then 14L13

C1.   {~20Th, gF20}          F1,  R1     rename varaibles
C2.   {~hM, hSg}             F1,  R2
C3.   {~hN, hLg}             F1,  R4
C4.   {~21Fh, 21Fr}          F2,  R1
C5.   {~22Fr, rL22}          F3,  R4
C6.   {~23Fh, hS23}          F4,  R2
C7.   {~hSr}                 Q,   R3
C8.   {~rFh, ~hN}            Q,   R4
C9.   {~24Lh, ~24Lr}         Q,   R5
C10.  {~rLh}                 Q,   R6
C11.  {gFr}                  F1,  C4
C12.  {hSg}                  F1,  C6
C13X  {gFr}                  F2,  C1     duplicate of C11
C14X  {hSg}                  F4,  C2     duplicate of C12

C15.  {~rFh, ~hM}            C7,  R2     begin using new facts
C16.  {~rFh}                 C7,  C6
C17.  {~rSh}                 C10, R3
C18.  {~hFr, ~rN}            C10, R4
C19X  {~25Lr, ~25Lh}         C10, R5     duplicate of C9
C20X  {~hLr}                 C10, R6     duplicate of query
C21.  {~hFr}                 C10, C5
C22.  {~26Tr, gF26}          C11, R1
C23.  {~rM, rSg}             C11, R2
C24.  {~rN, rLg}             C11, R4
C25.  {rLg}                  C11, C5
C26.  {hLg}                  C12, R3
C27X  {~hFr}                 F3,  C18    duplicate of C21
C28X  {rLg}                  F3,  C24    duplicate of C27
C29X  {~rFh}                 F4,  C15    duplicate of C16

C30.  {~rF27, ~hT27}         C16, R1     begin with new facts again
C31.  {~hFr, ~rM}            C17, R2
C32.  {~hF28, ~rT28}         C21, R1
C33.  {~hFh}                 C21, C4
C34.  {~rL29, gL29}          C25, R5
C35.  {~rL30, 30Lg}          C25, R5
C36.  {gLr}                  C25, R6
C37.  {~hL31, gL31}          C26, R5
C38.  {~hL32, 32Lg}          C26, R5
C39.  {gLh}                  C26, R6
C40X  {~hFh}                 F2,  C32    duplicate of C33
C41.  {gLg}                  C25, C34
C42X  {gLg}                  C25, C35    duplicate of C41
C43X  {gLg}                  C26, C37    duplicate of C41
```

```
C44X  {gLg}                    C26, C38    duplicate of C41
C45.  {~hFh, ~hTh}             C33, R1     begin with new facts again
C46.  {~gL33, rL33}            C36, R5
C47.  {~gL34, 34Lr}            C36, R5
C48X  {rLg}                    C36, R6     duplicate of C27
C49.  {~gLh}                   C36, C9     resolves with C39 to {}
C50.  {~gL35, hL35}            C39, R5
C51.  {~gL36, 36Lh}            C39, R5
C52X  {hLg}                    C39, R6     duplicate of C25
C53.  {~gLr}                   C39, C9     resolves with C36 to {}
C54.  {~gL37, gL37}            C41, R5
C55.  {}                       C54         QED.
```

The proof terminated with a clause which has a negative and a positive instance of the same atom. There are many observations to be made in this example. Let's begin by unwinding the logic of the proof. When the non-productive resolutions are pruned, the proof is quite straight forward and short.

```
C54:  {~gL37, gL37}
  R5:       {~10L11, ~10L12, 11L12}
  C41:      {gLg}
    C34:          {~rL29, gL29}
      C25:              {rLg}
                              proof below
      R5:               {~10L11, ~10L12, 11L12}
    C25:          {rLg}
      C5:               {~22Fr, rL22}
        F3:                 {rN}
        R4:                 {~8F9,  ~9N, 9L8}
      C11:              {gFr}
        F1:                 {gFh}
        C4:                 {~21Fh, 21Fr}
          F2:                     {rTh}
          R1:                     {~1F3, ~2T3, 1F2}
```

First, the resolution proof adopted a non-intuitive strategy, arguing from absurdity that a person cannot both have the same last name as someone (variable 37 in C54) and not have the same last name as that someone. This approach does not rely on any semantic knowledge about last names, obviously the computation does not understand naming conventions. The consequence is built into the transitivity rule (R5) itself.

Note the recursive use of C25. The established fact `rLg` (from C25) is used with R5 to construct the smaller rule `(if rL29 then gL29)`, if Rita has the same last name as someone, then so does George. It is then used again with that rule (C25 + C34) to show that the unknown person is George himself! Finally R5 is used again with the fact that George has his own last name to terminate the proof. Non-intuitive proofs and proof strategies are characteristic of algorithmic proof systems.

The proof would have been substantively different if R6, the commutative rule for last-names had not been included. In fact, it is not necessary for a proof. In this resolution proof, it is surprising that R2, R3, R6, and F4 were not used at all, even though from a natural deduction perspective they appear mandatory.

Note also the many convergent proofs toward the end. Had C54 not occurred, both C49 and C53 would have terminated the proof during the next cycle. Again, multiple paths with high redundancy are characteristic of algorithmic techniques.

Note also that the distinction between forward and backward chaining is largely lost, since matching positive facts and negative facts uses the same algorithm without distinction. The algorithmic proof followed all paths at the same time, taking small steps along each possible path without regard to conclusions or duplications.

Other control strategies for the resolution would have resulted in different proofs and even different proof strategies. It may have been more efficient, for example, to resolve the new facts with the shorter new rules first, before using the original rules, since the original rules R1 and R5 introduced excess variables.

In resolution, it is possible to resolve rules together, as well as just to follow facts. For example:

```
R2.   {~4F5, ~5M, 5S4}
R3.   {~6S7, 6L7}              ==>    {~20F21, ~21M, 21L20}
```

This generates a new rule, which is more direct for the purposes of the question that has been asked. When to do this becomes clear in the following boundary logic approach.

Boundary Logic

Again we transcribe the rules into a new, boundary, notation:

```
R1:   ( ((1F3) (2T3)) ) 1F2                 if  1F3  and  2T3  then  1F2
R2:   ( ((4F5) (5M)) ) 5S4                  if  4F5  and  5M   then  5S4
R3:   (6S7) 6L7                             if  6S7            then  6L7
R4:   ( ((8F9) (9N)) ) 9L8                  if  8F9  and  9N   then  9L8
R5:   ( ((10L11) (10L12)) ) 11L12           if 10L11 and 10L12 then 11L12
R6:   (13L14)  14L13                        if 13L14           then 14L13
```

In this notation, some redundant logical structure can be seen at the level of individual rules. We simplify the rules individually using Involution:

```
R1:   (1F3) (2T3) 1F2
R2:   (4F5) (5M) 5S4
R3:   (6S7) 6L7
R4:   (8F9) (9N) 9L8
R5:   (10L11) (10L12) 11L12
R6:   (13L14)  14L13
```

The boundary approach is based on reducing the entire collection of rules and facts as a whole. Rather than accumulate new facts, all the facts are combined into a single "conjunction of facts and rules implies conclusion" form. The general template is:

```
( ((fact1) ... (factn) (rule1)... (rulen)) ) query
```

which simplifies to

```
(fact1) ... (factn) (rule1)... (rulen)    query
```

For the example, the template is

```
(F1) (F2) (F3) (F4) (R1) (R2) (R3) (R4) (R5) (R6) Q
```

and the specific structure is

```
(gFh) (rTh) (rN) (hM)              facts
((1F3)(2T3) 1F2)                   R1
((4F5)(5M) 5S4)                    R2
((6S7) 6L7)                        R3
((8F9)(9N) 9L8)                    R4
((10L11)(10L12) 11L12)             R5
((13L14)  14L13)                   R6
hLr                                query
```

The boundary approach has taken yet another step away from intuition, now rules and facts are no longer distinguished. Like resolution, there is only one primary transformation, Pervasion. The idea is use the forms on the outside to extract their matching forms from the inside. Again, the matching technique is unification. Unlike resolution, the primary boundary transformation of Pervasion is augmented with two other transformations. Involution cleans up irrelevant logical distinctions, and Dominion tells the process when to stop.

Rule simplification and compilation

It is a good idea to simplify rules first, since they are abstractions applying to all facts, and are the source of complexity.

The first observation is that transitivity (R5) and commutativity (R6) apply all the time. They are not specific enough to help with deductions, but they do help to broaden the generality of facts. Use these rules only to generate new facts, not as part of a deduction.

Since the S relation shows up only once as a premise (in R3) and once as a conclusion (in R2), it can be compiled away. There is only one way to use (that is, to instantiate) the S relation, going from the premises of R2 to the conclusion of R3. In general we do not want to lose the ability to use either R2 or R3 by themselves (for instance in the case that the query is about an S relation), so we compile the S relation dynamically, in the presence of a known query.

Compile rules R2 and R3 into R23, using resolution `(A B) ((B) C) ==> (A C)`

```
S:  6 => 5,  7 => 4
((4F5)(5M) 5S4)  ((6S7) 6L7)  ==>   ((4F5)(5M) 5L4)
```

The new knowledge base:

```
(gFh) (rTh) (rN) (hM) hLr
((1F3)(2T3) 1F2) ((4F5)(5M) 5L4) ((8F9)(9N) 9L8)
((10L11)(10L12) 11L12) ((13L14) 14L13)
```

Should a rule have more than one premise, the ability to branch using the simple rule is lost in compiling. So, for instance, it is not possible to compile the `F` relation, even though it shows up only once as a conclusion (in R1).

Forced bindings

Now we make all forced bindings between the facts and the remaining rules. The Pervasion transformation says that a form on the outside of a boundary must match a form on the inside of a boundary, using unification as the matching technique. When a match is found, bind the variables and extract the inner form:

```
xAy  (1A2  1B3)  ==>  xAy  (xB3)
```

In the example KB, we have the fact `(rN)` on the outside which matches the inner form of R4 `((6F8) (9N) 9L8)`, binding the variable 9 to the atom `r` and erasing the `(9N)`:

```
(rN) ((8F9)(9N) 9L8)  =>  (rN) ((8Fr) rL8)
```

There is only one `(rN)` form on the outside and only one in all the insides, so there is only one possible extraction of an `N` relation. In this example, extracting `(rN)` leaves the knowledge base looking like:

```
N:  9 => r              (rN) ((8F9)(9N) 9L8)  =>  (rN) ((8Fr) rL8)
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr
((1F3)(2T3) 1F2) ((4F5)(5M) 5L4) ((8Fr) rL8)
((10L11)(10L12) 11L12) ((13L14) 14L13)
```

In general, there will be more than one fact matching each inner form, and more than one binding for each variable. This is what makes query management hard. The boundary approach lets us bind all possible variables, using sets of facts rather than individual facts. The set-based boundary approach would extract all matches, binding the variable to a *set* of matches.

We continue the forced (only one choice) bindings, using the strategy of binding the least number of variables first (ie using facts to their full extent). Portions of rules can be deleted when there is no possible way of using them again.

```
M:  5 => h              (hM) ((4F5)(5M) 5L4)  =>  (hM) ((4Fh) hL4)
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr
((1F3)(2T3) 1F2) ((4Fh) hL4) ((8Fr) rL8)
((10L11)(10L12) 11L12) ((13L14) 14L13)
```

There are several different strategies now available, the worst of which is to use either of the general transitivity or commutativity rules. `(gFh)` could extract `(4Fh)`, but this is premature since the rule portion could not be deleted in the presence of other `F` relations in the KB (in particular R1 may generate a need for the remaining portion of R23). Again seeking uniqueness and specificity, the best approach is to select the `T` relation which has only single occurrences on the outside and the inside of the KB.

```
T:  2 => r,  3 => h     (rTh) ((1F3)(2T3) 1F2)  =>  (rTh) ((1Fh) 1Fr)
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr
((1Fh) 1Fr) ((4Fh) hL4) ((8Fr) rL8)
((10L11)(10L12) 11L12) ((13L14) 14L13)
```

Now the `F` relation in both R1 and R23 should be extracted. Neither R1 nor R23 can be deleted. Along the way, a third possible `F` extract is generated and taken.

```
F:  1 => g              (gFh) ((1Fh) 1Fr)  =>  (gFh) (gFr)
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr
(gFr) ((4Fh) hL4) ((8Fr) rL8) ((1Fh) 1Fr)
((10L11)(10L12) 11L12) ((13L14) 14L13)

F:  4 => g              (gFh) ((4Fh) hL4)  =>  (gFh) (hLg)
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr
(gFr) (hLg) ((8Fr) rL8) ((4Fh) hL4) ((1Fh) 1Fr)
((10L11)(10L12) 11L12) ((13L14) 14L13)

F:  8 => g              (gFr) ((8Fr) rL8)  =>  (gFr) (rLg)
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr
(gFr) (hLg) (rLg) ((8Fr) rL8) ((4Fh) hL4) ((1Fh) 1Fr)
((10L11)(10L12) 11L12) ((13L14) 14L13)
```

There remains only one path for implication of F relations, that is the backward binding of `hLr` to the remains of R23. By taking this step, we are then free to erase all `F` rules.

```
F:  4 => r              hLr ((4Fh) hL4)  =>  hLr ((rFh))  =>  rFh
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr (gFr) (hLg) (rLg) rFh
((10L11)(10L12) 11L12) ((13L14) 14L13)
```

We have used the "query" `hLr` to generate another query `rFh`. There are now no more forced bindings, so we must use transitivity or commutativity of `L` to generate new facts. Finally we must use the branching rules, but with the comfort that every step thus far was without choice. Note that we can now focus on generating only new `L` facts.

The pair of facts `(hLg)(rLg)` provide a set match for either of the transitivity premises, but there is still a minimal approach to be taken. The commutativity rule has only one match, and further the `hLr` form only matches one form within the commutativity rule. We make that binding:

```
L:  14 => h, 13 => r    hLr ((13L14) 14L13)  =>  hLr ((rLh))  =>  rLh
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr (gFr) (hLg) (rLg) rFh
((10L11)(10L12) 11L12) ((13L14) 14L13) rLh
```

Above, we used the query `hLr` to extract a conclusion from R6. The resulting form which is not inside a boundary is also a query; that is, we would know `hLr` if we could show `rLh`. The critical point here is that we cannot collapse the commutativity rule out of the KB because there are facts present which could use it again. R6 can be used in either direction, forward or backward. The appropriate strategy now is to go ahead and make full use of R6 in the forward direction, with the set of bindings from `(hLg)` and `(rLg)`:

```
L:  13 => h, 14 => g    (hLg) ((13L14) 14L13)  => (hLg) (gLh)
L:  13 => r, 14 => g    (rLg) ((13L14) 14L13)  => (rLg) (gLr)
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr (gFr) (hLg) (rLg) rFh
((10L11)(10L12) 11L12) (gLh) (gLr) rLh  ((13L14) 14L13)
```

We now face many branches, but we have constrained them to only one rule, R5. There are two uses of transitivity in the backward direction, reasoning from queries, so we bind them both, without eliminating the rule.

```
L:  11 => h, 12 => r    hLr ((10L11)(10L12) 11L12) => hLr
((10Lh)(10Lr))
L:  11 => r, 12 => h    rLh ((10L11)(10L12) 11L12) => rLh
((10Lr)(10Lh))
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr (gFr) (hLg) (rLg) rFh
((10L11)(10L12) 11L12) (gLh) (gLr) rLh
((13L14) 14L13) ((10Lh)(10Lr)) ((10Lr)(10Lh))

L:  10 => g             (gLh) ((10Lh)(10Lr))  =>  (gLh) ((gLr))  =>  gLr
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr (gFr) (hLg) (rLg) rFh
((10L11)(10L12) 11L12) (gLh) (gLr) rLh
((13L14) 14L13) gLr ((10Lr)(10Lh))

L:                      gLr  (gLr)  =>  gLr  (   )  =>  (   )
```

New KB:

```
(gFh) (rTh) (rN) (hM) hLr (gFr) (hLg) (rLg) rFh
((10L11)(10L12) 11L12) (gLh) (   ) rLh
((13L14) 14L13) gLr ((10Lr)(10Lh))
```

The boundary deduction has concluded in its characteristic manner by asserting a `( )` into the KB. By the Dominion rule, this truth symbol erases all other forms in the problem space, leaving a mark of proof. Notice in the second to last step, the selection `(gLr)` was also available for binding. It would have been chosen next, should the current choice have failed. And it too would have terminated the proof process.

The signature characteristic of this boundary proof is its minimality. In contrast to resolution, very little search was conducted because the problem was structured as a global statement rather than as a collection of fragments. Thus the available strategies addressed the entire problem at all times. To reconstruct the logic of the boundary proof, we trace the binding processes of steps which are used to reach the conclusion (only the step which generated `rFh` was unnecessary):

```
((4F5)(5M) 5S4)  ((6S7) 6L7)  ==>   ((4F5)(5M) 5L4)          R2+R3      = R23

(rN) ((8F9)(9N) 9L8)          =>  ((8Fr) rL8)               F3+R4      = R4a

(hM) ((4F5)(5M) 5L4)          =>  ((4Fh) hL4)               F4+R23     = R23a

(rTh) ((1F3)(2T3) 1F2)        =>  ((1Fh) 1Fr)               F2+R1      = R1a

(gFh) ((1Fh) 1Fr)             =>  (gFr)                     F1+R1a     = (gFr)

(gFh) ((4Fh) hL4)             =>  (hLg)                     F1+R23a    = (hLg)

(gFr) ((8Fr) rL8)             =>  (rLg)                     (gFr)+R4a  = (rLg)

hLr ((13L14) 14L13)           =>  ((rLh)) => rLh            Q+R6       = rLh

(hLg) ((13L14) 14L13)         =>  (gLh)                     (hLg)+R5   = (gLh)
(rLg) ((13L14) 14L13)         =>  (gLr)                     (rLg)+R5   = (gLr)

hLr ((10L11)(10L12) 11L12)    =>  ((10Lh)(10Lr))            Q+R5       = R5a
rLh ((10L11)(10L12) 11L12)    =>  ((10Lr)(10Lh))            rLh+R5     = R5b

(gLh) ((10Lh)(10Lr))          =>  ((gLr)) => gLr            (gLh)+R5a  = gLr

gLr  (gLr)                                                  gLr+(gLr)  = QED
```

Addendum

The phraseology and structure of rules in a knowledge-base is extremely critical to the success of an inference engine. Examples:

1. To generate a sequence of numbers, it may be tempting to put in a integer generation rule such as

```
if (_1_ is-an-integer) then  ( (_1_ + 1) is-an-integer)
```

This rule could immediately generate an infinite string of integers, which would, of course, be expressed computationally as an over-flow crash.

2. Similar recursive overflows can occur with quite common rules such as transitivity and commutativity. Both of these rules occur in the example KB above (R5 and R6 for the has-same-last-name relation). Implicit in the actual transforms is an "occurs-check" which stops rules from being called when they generate items which duplicate already existing items.

3. Some rules can be expressed in different ways. The forms of these rules strongly effect both the sequence of fact generation, and the efficiency of the deductive process. For instance, transitivity is commonly expressed as

```
if (1R2 and 2R3) then 1R3
```

In the example KB, R5 expresses transitivity as

```
if (1R2 and 1R3) then 2R3
```

This design choice was made because of the natural semantics of the has-same-last-name relation. The design choice has a strong effect on the sequence of generated facts in each example.

4. Some rules implicitly incorporate other rules, in that the other rules are strictly redundant. If we let the variable 3 be equal to "1" in the above transitivity rule, (and we omit the trivial fact that a person has their own last name), we get the commutative R6.

```
if (1R2 and 1R1) then 2R1    ==>    if 1R2 then 2R1
```

Again, the choice of whether to include rules specifically, or let them be implicit in other rules has a strong and unpredictable effect on the performance of the engine.

5. Rules should take care to exclude unwanted cases, although it is often a difficult choice between simple rules with fast cycling time, or larger rules which take effort to compute. This issue also shows up in programming as choices about function decomposition, and in CPU design as RISC vs CISC architectures. In the example, we elected not to exclude the fact that a person has their own last name, but we could have expressed R1 as:

```
if (1F2 and 1≠2 and 2T3 and 2≠3) then 1F3
```

Alternatively, the deduction might have been able to make use of the same-last-name-as-yourself rule, and we may have wanted to include it as

```
      if ( 1 = 2 ) then 1L2
```

or as

```
      if  1L2  then 1L1
      if  1L2  then 2L2
```

These decisions are quite difficult to make, and depend on the expected types of queries, the structure and frequency of facts in the KB, and the other rules in the KB.

6. Rule ordering plays a critical role in algorithmic transformations. When a decision has to be made about what rule to bind next, it is often the case that a general strategy like set-of-support or simplest-first still results in several equally likely choices. Algorithms tend to take the next rule in sequence, but this may not be best or be most efficient. When having to answer a query like "Which people are a child of a President?" it is imperative that the search engine know something about the size of the domains. It is far better to approach this by looking for Presidents first, then looking for their children, than it is to look at all the children in the country and ask each if their parent is a President. You can play with this issue yourself by querying a web search engine for pages which have some common word, such as "set", and some rarer word such as "recursion". Do your results depend on the order of the query words?

7. Finally, the reason for this addendum is that I wasted many hours (and distributed faulty code to the class) with a poorly designed R1. This design flaw was subtle, since both the original and the final form of the rule were valid. R1 originally said "if person A is your father and you are the sister of person B, then person A is the father of person B". This makes sense, but logically it needs the support of another rule, "if you are the sister of person C, then person C is the brother-or-sister of you" in order to converge with the other rules. That is, there was no way for the inference engine to turn around the idea that you are a sister, making it a sibling relation. This inversion was necessary basically due to the structure of the fact-base, in that the constant "Harry" never found its way to a position where it could be matched. The solution in this case was to change the form of R1:

```
      if (1F2 and 2T3) then 1F3            NO
      if (1F3 and 2T3) then 1F2           YES
```

Note that this artifact is due to the very limited rule-base. A more acceptable and correct solution would be to include the entire spectrum of relationships:

```
      if (1 is-sister-of 2) and (2 is-male) then (2 is-brother-of 1)
      if (3 is-brother-of 4) and (4 is-female) then (4 is-sister-of 3)
      if (5 is-father-of 6) and (6 is-sister-of 7) then (5 is-father-of 7)
      if (8 is-father-of 9) and (9 is-brother-of 10) then (8 is-father-of 10)
```

A Small Interpreted Language

What would you need to build a small computing language based on mathematical principles? The language should be simple, Turing equivalent (i.e.: it can compute anything that any other language can compute) and relatively easy to use. Assume the computing hardware is constrained to vonNeumann processes, with memory, an ALU, and appropriate registers. We will also assume that we know about formal mathematical languages and the necessary mathematical pieces: representation, recognizer, constructor, accessor, invariants/facts, functions, and induction/recursion.

Base *Representation* of Atoms

First, the *alphabet* of a language is simply a collection of unique identifiers, called *atoms*. The essential memory management trick is to divide each memory cell into two parts, an address part (call it **First**) and a contents part (call it **Rest**). Addresses are also called *pointers*. We begin with an array of empty cells, each having some empty representation in both the **First** and the **Rest** parts. This is the *free list* of memory cells.

The ground: We need an atom which means nothing, the null atom. Call it **nil**.

The symbol table: This table consists of a collection of non-empty memory cells, one cell for each atom in the language. The **First** part of an atom cell contains nil. The actual literal representation of the atom is in the **Rest** of the cell. The symbol table is a dynamic array.

Constructor of Compound Expressions

We need to construct compound expressions. Consider an expression which uses two atoms, say FOO BAR. The symbol table contains each atom, so all we need is a way to connect them. This can be done simply by building another memory cell which contains the two addresses of FOO and BAR. We put all atom addresses in the **First** part of a cell (see cell 005 below) and connecting addresses in the **Rest** part. The instruction to build connecting cells is called **Cons**. The end of an expression has **nil** in the **Rest**.

If we build the expression (TRUE BAR TRUE FOO) in cell 007, memory would look like this:

Address	First	Rest	
000	nil	nil	symbol table
001	nil	FOO	
002	nil	BAR	
003	nil	BAZ	
004	nil	TRUE	end of symbol table
005	001	006	the expression (FOO BAZ)
006	003	000	end of expression
007	004	008	the expression (TRUE BAR TRUE FOO)
008	002	009	
009	004	010	
010	001	000	end of expression
011	empty	empty	begin free list
...			

To construct an expression, we **Cons** smaller pieces together. For instance:

```
Cons JOHN (TRUE BAR TRUE FOO) ==> (JOHN TRUE BAR TRUE FOO)
```

The operational memory changes are:

```
011         nil         JOHN        the atom JOHN
012         011         007         connect JOHN to (TRUE BAR TRUE FOO)
013         empty       empty
```

Consider **Cons**ing two compound expressions together:

```
Cons (FOO BAZ) (TRUE BAR TRUE FOO) ==> (FOO BAZ TRUE BAR TRUE FOO)
```

This operation is slightly more complex. For the entire expression to begin in cell 012, we need memory to end up as

```
011         003         007                 (BAZ TRUE BAR TRUE FOO)
012         001         011                 (FOO BAZ TRUE BAR TRUE FOO)
013         empty       empty
```

Several design decisions are involved with this result. Technically, we have used *structure sharing* for (TRUE BAR TRUE FOO) since both the original four atom expression and the final six atom expression use some of the same memory cells. However, the front of the expression, (FOO BAZ) is not engaged in structure sharing, and this may seem a little unsymmetrical. As it is, (TRUE BAR TRUE FOO) is confounded with **Rest** (**Rest** (FOO BAZ TRUE BAR TRUE FOO)).

An alternative which would allow us to continue to refer to the original would be to duplicate the four atom expression entirely in constructing the six atom expression.

Note also that the construction is slightly different, rather than adding a symbol cell, as in the case of JOHN, we have added a *cons cell*. To acknowledge these differences, we might consider **Cons** of two compound expressions to be a different operation. Call it **Append**. Now the first object in a **Cons** operation is restricted to be an atom. **Append** is used when the first object is compound. To keep the language simple, we would want to be able to build new operations out of the existing ones. For this, we use a recursive definition:

```
Append <obj1> <obj2> =def=
  If Isa-atom <obj1>
    then ERROR
    else if Is-empty <obj1>
      then <obj2>
      else Cons (First <obj1>)(Append (Rest <obj1>) <obj2>)
```

This recursive definition first does a *type-check* on <obj1>. It then tests the *base case*, that <obj1> is **nil. Append**ing nothing onto <obj2> results in <obj2>. Otherwise we proceed one piece at a time. The recursion bottoms-out when **Rest** <obj1> is **nil**. For this to be the case, <obj1> must have only one atom, as in (BAZ), which is **Cons**ed onto <obj2>. At that time, BAZ is the **First** of <obj1>. Just prior to this case, <obj2> is actually (BAZ TRUE BAR TRUE FOO), since we have **Cons**ed BAZ to (TRUE BAR TRUE FOO). <Obj1> is (FOO BAZ), and we are about to **Cons First** <obj1>, i.e. FOO, onto (BAZ TRUE BAR TRUE FOO).

This description has backed up from the end to the beginning. Tracing the events in memory:

```
Append nil (TRUE BAR TRUE FOO) ==> (TRUE BAR TRUE FOO)

011          000          007          Append nil
012          empty        empty        begin free list
```

By definition, cell 011 is the same as 007, so operationally this step is not necessary to take. We leave 011 free, treating **Append**ing **nil** as a *no-op*.

```
Cons BAZ (TRUE BAR TRUE FOO) ==> (BAZ TRUE BAR TRUE FOO)

011          003          007

Cons FOO (BAZ TRUE BAR TRUE FOO) ==> (FOO BAZ TRUE BAR TRUE FOO)

012          001          011
013          empty        empty
```

What we have done here is to specify exactly the sequence of operations on memory that result in the action of **Append**ing. And we have used the single construction tool of **Cons**.

This example illustrates the close connection between a software program, the attendant changes in memory, and the hardware architecture which unites both.

Recognizer of Atoms

The *recognizer* of each atom is a function which looks in the symbol table for the memory cell which contains that atom. For instance, the predicate **Isa-atom** is true if its argument can be found in the **Rest** portion of the symbol table. At this point, we have three separate memory areas (or uses): *free cells*, *atom cells*, and *cons cells*.

Isa-atom: Atom cells are recognized by having **nil** in the **First** part.
Is-empty: Empty expressions can be uniquely recognized because they have **nil** in the **Rest** part.
Equal: Tests if two atoms are the same atom.
Isa-expression: **Cons** cells are recognized as those cells having two addresses. An expression ends with **nil** in the **Rest** part.

The above are close to operational definitions. Here are some slightly more elaborated operational definitions. We will assume that each part of a memory cell (address, first, rest) has eight bits.

Is-empty <obj>:

Assign **nil** a special binary code, 00000000, and put it in address 00000000.
An object is empty, that is, it is equal to **nil**, if the **Rest** part is equal to the code of **nil**.
To distinguish **nil** from an empty cell on the free list, we could put a special code in free list cells, perhaps 11111111. A better approach is to use only seven bits of the address for address information, and use the eighth bit for marking if a cell is free. This is the basis for many *garbage collection* algorithms.

```
Is-empty <obj> =def= Equal (Rest <obj>) 00000000
```

Isa-atom <obj>:

Test the encoding of <obj> against all the encodings in the **Rest** part of memory which also have **nil** in the **First** part.

```
Isa-atom <obj> =def= for some memory cell
  (Equal (First <obj>) nil) and (Equal (Rest <obj>) <obj>)
```

Here is another *design choice*: is **nil** an atom or not? If it is not an atom, we will have to have special tests for atoms vs **nil**. For simplicity, let's say it is an atom:

```
(Isa-atom nil) is True
```

This design choice is our first *fact*, or invariant.

More generally:

```
Isa-atom (Is-empty <obj>) =def=
      True iff (Is-empty <obj>) is True
```

Recognizer of Expressions

We can use the instructions **First** and **Rest** to access and decompose all expressions. (**First** <obj>) looks at the first part of memory for the specific object, (**Rest** <obj>) looks at the rest part.

To recognize compound expressions, we test to see if each part of that expression is in the memory table, and the linking structure of the expression matches the rules for constructing that expression. Operationally:

```
Isa-expression <obj> =def=
  (Isa-expression (First <obj>)) and
  (Isa-expression (Rest <obj>))
```

Since we know that decomposing an expression will end in either atoms or nil, we will have to add those rules:

```
Isa-expression (Isa-atom <obj>) =def=
  True iff (Isa-atom <obj>) is True

(Isa-expression nil) is True.
```

This is another application of recursive decomposition. The rules specify the base cases, while the definition specifies the general recursive case. The two together specify a program.

The definition above is another example of *pseudo-code*, that is, machine specific instructions written in a mathematical style that is independent of the specifics of any programming language, yet specific enough to be implemented in any language. Of course, a *high level programming language* accepts something very close to pseudo-code specification as valid input. Another strong advantage of pseudo-code is that it can be proven to be correct using the Induction Principle.

The primary reasons that current programming languages appear to be very different than pseudo-code are

1. Many programming tasks lack a formal model (i.e. they are hacks).

2. Many programming languages lack mathematical structure (i.e. they are machine architecture specific.)

Accessors of Atoms and Expressions

First and **Rest** are the accessors. They let us take apart an expression. In this implementation, **First** and **Rest** have simple mappings onto the idealized physical structure of memory.

All objects except **nil** are constructed by **Cons**. Since **Cons** uses two objects as arguments, this means that all **First** and **Rest** parts are also objects. Eventually all objects end in **nil**, so **nil** is also an object, although a very special kind.

Cons is related to **First** and **Rest** by the following invariant, or rule:

```
<obj> = Cons (First <obj>) (Rest <obj>)
```

This says that all valid objects have been constructed by **Cons** to have a **First** part and a **Rest** part in memory. Alternatively, all objects in memory can be accessed through their **First** and **Rest** parts. The essential mathematical condition is that all valid objects are decomposable into unique subcomponents which bottom-out at the base cases. This is simply to say that all compound expressions are defined recursively.

Although recursive composition and decomposition are necessary to define data structures and algorithms, the more important aspect of recursive definition is to provide access to proof through the Induction Principle. Procedural languages do not provide this capability; they are thus immature. Declarative, functional, and mathematical programming languages all provide the capability of abstract proof (minimally in pseudo-code).

Note that recognizing, constructing, and accessing an expression involve almost the same steps. The difference is in the initial goal and the final result.

	GOAL	PROCESS	RESULT
Constructor:	build a pattern	rearrange memory	the pattern is in memory
Recognizer:	test a pattern	access memory	true if the pattern is accessible
Accessor:	get a pattern	access memory	return the pattern if found

SUMMARY of the ABSTRACT DATA STRUCTURE FUNCTIONS

First <obj> returns the expression indicated by the **First** of the <obj>

Rest <obj> returns the expression indicated by the **Rest** of the <obj>

Is-empty <obj> returns True if the cell containing <obj> has **nil** in **Rest**.

Isa-atom <obj> returns True if the <obj> is in the **Rest** part of a cell and **nil** is in the **First** part.

Isa-expression <obj> returns True if the <obj> has either **nil** or any address in the **First** part.

Equal <obj1> <obj2> In the case of atoms, returns True if both objects are in the **Rest** of the same symbol cell. In the case of compound expressions, returns True if following the addresses in the **First** leads to the same set of **Rest** symbol cells.

Cons <obj1> <obj2> builds an expression by adding <obj1> to the front of <obj2>

Invariants

The *equality invariant* (also called the Uniqueness Axiom) assures that each object is unambiguous. That is, objects are the same object when they are equal; equal objects are constructed and deconstructed in exactly the same way. This is a physical kind of equality, *structural equality*, in that the structure of memory is the same for two objects. It is not necessary that the same memory cells are used for both objects (structure-sharing), just that the contents of memory for both objects are the same. Recursively,

```
Equal <obj1> <obj2> =def=
  (Equal (First <obj1>) (First <obj2>)) and
  (Equal (Rest <obj1>) (Rest <obj2>))
```

We need to support this definition with base cases. For instance,

```
(Equal nil nil) is True
```

This is also an example of the *Induction Principle* at work in our implementation. To implement an equality test for expressions, the computation will test for identical structure over all memory cells of both objects. The Induction Principle is the only guarantee that this recursive process will end. The only end point is (**Equal nil nil**), all other cases are failures.

Note that equality for atoms is also covered in the above definition. What happens, though, when we have two atoms which have the same encodings, but each is in a different memory cell? This is an inconvenience for an implementation, since testing each object would require looking through the entire symbol table. A better approach is to insist that each atom is unique and occurs only once in the symbol table. This is why Equality and Uniqueness are the same ideas.

The uniqueness of atoms is implemented by having each new atom *register* itself in the symbol table. In the background, when an unrecognized, new atom is entered, the implementation verifies that it is new, and then puts it in the symbol table. To do this is to *intern* the atom. If the atom already exists, then the address of the cell which contains that object is associated with the new input.

A different kind of equality refers to equality under transformation. The actual expressions may be different, but transformation rules allow us to say that the meaning of the different expressions is the same. This is *semantic equality*, also called *algebraic equality* and *mathematical equality*. Only defined transformations are allowed; all transformations (with the exception of **Cons**) are required to keep meaning consistent. It takes a special *symbolic architecture* to implement mathematical equality, mainly because transformations refer to sets or classes of objects rather than to specific objects. In the above, we have designed a *literal architecture*, as yet it has no capacity for dealing with sets of objects.

Now on to the functional part of the language. We will elect to use *lambda calculus* as the mathematical model.

Functions* and *Recursions

A function is an expression with the function name first and then the arguments. (The order of operators and arguments is somewhat arbitrary, just so long as it is consistent and unambiguous.) For example:

```
+ 3 4
```

The Arithmetic Logic Unit (ALU) can process logical and arithmetic operators when applied to atoms. Internally, both arithmetic and logic are encoded by binary sequences, so it is the responsibility of the operator, or of a *type test*, to make sure that expressions meant to represent numbers are channeled to the arithmetic units and expressions intended to represent logic are channeled to the logic units.

One way to implement the difference between logic and arithmetic is to assign another single bit in the memory cell that records the type of object in that cell. Note that silicon gates process only logic. Thus arithmetic objects must be encoded into a logical form for processing. In computation, *logic is fundamental*, arithmetic is derivative.

All logic functions can be defined in terms of a single function, so we need only one primitive logic function. Let's use **IfThenElse** (**Nand** and **Nor** are alternatives).

```
IfThenElse <obj1> <obj2> <obj3>
```

IfThenElse will evaluate <obj1> and then either evaluate <obj2> (if <obj1> is True) or evaluate <obj3> (if <obj1> is False). Here we have another function which uses different types of objects (the first example was **Cons**). In particular, <obj1> must be a logical type, returning either True or False.

Function composition permits complex sequences of operations. A function expression can be put in any place that an atom can be put, since all functions will reduce to single atoms. To separate composed functions, we can use parentheses to contain each function expression. We will choose to evaluate all inner arguments first, then use these results to evaluate outer

functions. Lambda calculus permits another order of evaluation, outermost first. This choice is a design decision, and is based on mathematical characteristics of each form of evaluation.

For example, an innermost evaluation:

```
(* (+ 3 4) (+ 1 2)) or ((3 + 4) * (1 + 2))
```

means that expressions with atoms as arguments are evaluated, or *reduced*, first.

The memory for this object would look like this:

Address	First	Rest	
000	nil	nil	symbol table
001	nil	1	
002	nil	2	
003	nil	3	
004	nil	4	
005	nil	+	
006	nil	*	end of symbol table
007	006	008	expression ((3 + 4) * (1 + 2))
008	005	009	
009	003	010	
010	004	011	
011	005	012	
012	001	013	
013	002	000	end of expression
014	empty	empty	begin free list
...			

There are several things to note about the above memory configuration.

Operators and numbers are not distinguished in memory, they are distinguished by what happens when they are handed to the ALU.

Each operator has two arguments, but we have no way to have two references in one memory cell. The solution is to order the expression so that operators are followed by their arguments. When an operator is fetched for evaluation, the machine code recognizes that that operator requires two more fetches. Should a fetch return another operator, then the first operator waits until the second operator converts its two arguments into one result.

Fetches occur by following the addresses in sequence. This is efficient since the address register (the register which keeps track of what to fetch next) need only be decremented by one to find the next memory cell.

It is possible to turn all functions into one argument functions (the technique is called *currying*). This is effectively what has happened by storing the expression in the *operator first* form (also known as reverse Polish notation).

Finally, consider how close the syntax of many programming languages is to what actually happens at the *register transfer level* of the computer. We are still at the very early stages of development of computing languages, since the syntax reflects low level data shuffling rather than high level task requirements. Progress means moving our profession toward human capabilities, and moving away from low level machine details.

We need a way to define arbitrary functions and a way to bind the variables of functions to values for the ALU to process. For example

Square <obj> =def= (* <obj> <obj>)

so that **Square** 4 => (* 4 4) => 16

First consider variables, names which stand for any valid object. We have been using the names <obj1>, <obj2>, etc. as variables names. The angle brackets notate that the name in question is not the name of a single thing, but rather it is the name of a class, or *set*, of things, all of which are of a particular *type*.

Variables (or *parameters*, when the names are arguments of a function) are atoms also, so they are simply added to the symbol table. To assign a value to a variable symbol, we can put a reference to the location of the value we wish to associate with the variable in the **First** part of the memory cell for the variable. Thus variables are distinguished from objects representing a specific value because their **First** part is not **nil**. It is an error to access a *variable* which has **nil** as the **First** part. Objects which do have **nil** in the **First** part are called *ground objects*.

The function which assigns ground objects to variable objects is called **Let**.

We can use this same mechanism to store the definitions of functions. The memory cell which contains the name of the function in the **Rest** part can contain the address of the definition of the function in its **First** part. Consider the memory configuration for the above definition of Square:

Address	**First**	**Rest**	
000	nil	nil	symbol table
010	nil	OBJECT	
011	nil	*	end of symbol table
012	013	SQUARE	function definition
013	011	014	
014	010	015	
015	010	000	end of function definition

When the call **Square** 4 is added to memory we get:

016	nil	4	symbol table
017	012	018	function call (Square 4)
018	016	000	

To bind OBJECT to the value 4, we use the call **Let** object 4:

019	nil	LET	symbol table
020	019	021	function call (Let object 4)
021	010	022	
022	016	000	

Finally we need to get the processor to actually evaluate the function call. Let's call this **Eval**. We can actually make **Eval** the default. Whenever a new expression is added it can be automatically evaluated. This just shifts the issue to needing an instruction to stop evaluation. Let's call this evaluation stopper **Quote**.

What the above memory configuration contains is **Quote** (**Square** 4), which simply puts the *data structure* **Square** 4 into memory. If we write the *function* **Square** 4, then evaluation will happen automatically. This process consists of changing the value of object from **nil** to 4, and following the sequence until a single atom is returned. That is, the function **Let** says to the processor: go to the symbol which immediately follows **Let** and put the address of the second symbol which follows **Let** (i.e. 4) in its **First** part. This results in

```
010          016          OBJECT
```

Now the definition of **Square** will find the value of OBJECT and use it rather than using the symbolic variable "OBJECT". And, of course, symbolic variables are the only items in the symbol table which can contain something other than **nil**.

There is a slight problem here because the symbol "OBJECT" might be used in more than one function. This can be handled in one of two ways:

1) make sure all of the symbols are unique, or
2) divide the symbol table into subtables which associate and isolate each function with its own variables.

Finally, we simply use *recursion* directly as repeated actions of the same sort, since nothing in the above structuring stops this from working.

The *Function* Eval

In the above description, evaluation is an implicit action of the ALU. By claiming evaluation is automatic, we are committed to wiring the ALU in a specific way. However the above mechanism for handling memory can be made flexible by *defining* ***Eval*** *in the programming language itself*. This process is called *meta-circular evaluation,* cause it uses a language itself to define how that language should behave. All we have to do is to define the evaluation function by telling the system what to do when an expression is typed in. The function **Eval** takes two arguments, the expression to be evaluated and the *binding environment*, that is, an address of the memory array which contains all of the primitive functions and atoms (and any other symbols which we may have added) in the language. The binding environment contains the definitions of all user defined functions, and the values of each of the variables (function arguments).

Since the binding environment does not change in this example, (i.e. we have not designed the language to establish separate environments for each function call), we will treat the token **Eval** to mean "Eval-in-environment". The definition of **Eval** which follows uses only primitive functions introduced above. Some of the syntax has been changed to make it more readable.

This **Eval** function recognizes seven operators:

First	**Rest**	**Cons**	
IfThenElse	**Equal**	**Quote**	**Let**

In addition, **Eval** uses built-in tests to determine the types of objects, as operationalized above.

Is-empty	**Isa-atom**	**Isa-expression**

```
Eval exp  =def=

If Isa-atom exp
 Then                                                    ;process atom
   If ((Is-empty exp) or (Equal exp (Quote True)))
    Then                                                 ;return the SYMBOL
      exp
    Else                                                 ; or its VALUE
      Get-value-in-env exp
 Else                                                    ;process expression
   If Isa-atom (First exp)
    Then                                                 ;process Atom in First*
      Let token (First exp)                              ;naming the atom
        If Equal token (Quote Quote)
         Then                                            ;return what follows
           Second exp
         Else                                              ;other operators
           If Equal token (Quote IfThenElse)
            Then                                         ;process logic operator
              EvalLogic (Rest exp)
            Else                                           ;other operators
              If Equal token (Quote First)
               Then                                      ;First of Eval of Rest
                 First (Eval (Second exp))
               Else                                        ;other operators
                 If Equal token (Quote Rest)
                  Then                                   ;Rest of Eval of Rest
                    Rest (Eval (Second exp))
                  Else                                     ;other operators
                    If Equal token (Quote Isa-atom)
                     Then                                ;Isa-atom Eval of Rest
                       Isa-atom (Eval (Second exp))
                     Else                                  ;other operators
                       If Equal token (Quote Cons)
                        Then                             ;Cons Eval of Rest**
                          Cons (Eval (Second exp))
                               (Eval (Third exp))
                        Else                               ;other operators
                          If Equal token (Quote Equal)
                           Then                          ;Equal Eval of args
                             Equal (Eval (Second exp))
                                   (Eval (Third exp))
                           Else                          ;replace the token with
                             Eval (Cons                  ;its value
                                    (Get-value-in-env token) (Rest exp))
    Else                                                   ;compound First
      If Isa-expression (First exp)
       Then                                              ;process expression
         EvalExp exp
       Else ERROR
```

```
EvalLogic exp  =def=

If Equal (Eval (First exp)) (Quote True)                    ;if First is TRUE
 Then                                                       ;Eval second argument
   Eval (Second exp)
 Else                                                       ;Eval third argument
   Eval (Third exp))


EvalExp exp   =def=

If Is-empty exp                                             ;if at the end
 Then                                                       ;return ground
   nil
 Else                                                       ;Eval the parts
  Cons (Eval (First exp)) (Eval (Rest exp))                 ; and put them together
```

Notes:

* process Atom in First: Here we have defined a syntax for parsing. Every expression begins with an atom or is an atom. If an expression begins with an atom, that atom is taken by the processor to be an operator, and thus a processing instruction. The operator **Quote** is the no-op.

** Cons Eval of Rest: This is again a syntax constraint. Once we have removed the beginning operator of an expression, what follows is either an atom, or another expression which itself begins with an atom operator.

The syntax of the language is thus:

```
Expression ::= Atom | (Atom Expression*)
```

The Kleene star means that an operator atom can have any number of following arguments. Note that this BNF specification is one of a *regular language*.

Finally, note that a meta-circular language can evaluate its own definition of **Eval**, since the above definition is self-consistent.

The Punch Line

The above programming language actually exists, it is one of the very few oldest programming languages still in active use. Its name is *LISP*.

In 1955, John McCarthy followed a similar line of reasoning in developing LISP. Currently LISP stands uniquely among programming languages in that it is rigorous, efficient, largely machine independent, and permits simulation of all other programming language models (such as procedural, functional, object-oriented, logical, and mathematical). As well, when the function **Eval** is processing input, LISP is *interpreted*, responding dynamically to new inputs and definitions, and requiring no compilation or linking. It thus provides a powerful interactive programming environment which supports real-time debugging and symbolic proof of correctness.

Assignment 1

A short oral presentation in class.

Tell a detailed story about a programming experience you have had.

A good story revolves around a character and a life experience. It might have some tension, some humor, a critical event, and some learning.

A programming story should be about or include code, before and after the critical event.

You should include what you learned from the experience, and perhaps how you would like things to change.

Tell why you chose the particular story that you did choose. Why is it interesting or important to tell? Is there a moral?

Assignment II: Pseudocode Syntax

Hand in to instructor at beginning of class.

Pseudocode is a computing language which is designed to convey ideas, specifications, and algorithms. It eliminates as many implementation details as possible. In theory, a pseudocode program should be executable, given that the lower-level details are provided.

Design the syntax of a pseudocode programming language.

You will need to:

1. Select a small set of primitives for abstracting control, data, and names.
2. Decide upon a lexical form for your language primitives
3. Decide upon a syntax which structures how primitives are combined.
4. Use standard techniques to define acceptable lexical and syntactic forms.

Standard syntax specification techniques include task decomposition, regular languages, finite state acceptors, formal grammars, BNF and/or diagrammatic BNF.

Language primitives can be seen as addressing control, data, or naming. Control primitives are included in imperative languages to steer the course of program evaluation. Data primitives provide typing and abstraction. A language may provide a single data type, or several basic built-in types, or user-defined types. Naming primitives determine the binding of names to values, and the location of names and values in memory. Primitives that we have discussed in class include `loop`, `logic`, `let`, and `domain theories`. Others may include `sequence`, `order comparisons`, `subroutines`, `hierarchy`, and `i/o`.

Important:

1. The task is to think clearly and carefully about the meaning of the 19 *Principles of Programming Languages* (handed out in the first week). Check your design decisions for conformance to each of these principles.

2. You must *be explicit about the tasks* which your pseudocode is intended to address. Ask why each primitive and each structure is included in the design. What part of the task does each particular structure address?

3. Attempt to avoid structures which are intended to enhance implementation efficiency. Pseudocode is not intended to address implementation efficiency, rather it should maximize readability, comprehension, and absence of ambiguity.

4. The most difficult part of language design is minimizing interactions between primitives when they are combined.

Challenge: Implement a lexical scanner and syntactic parser for your language.

Assignment III: Pseudocode Emulation

Nothing to hand in.

Implement an emulator for your pseudocode formal syntax.

An *emulator* of a program is a different program, usually in a different language, that does the same thing as the target program. Emulators are often built for hardware: a software module performs the same functionality as the target hardware, but in software. Software emulation is usually much slower. Another example is programs which simulate, say, a Windows environment on a Mac OS; these programs emulate Windows on a Mac.

In Assignment II, you designed a language fragment and formalized it with BNF or another structuring tool. In this assignment, you will implement the syntax of your language. (You may elect to use a different fragment, or a completely different formal specification.)

The assignment is simple if there is a one-to-one correspondence between your specification and some existing language. For example, if you specify a WHILE construct, then the specification language can translate directly to WHILE is some existing language like C. What gets tricky is verifying that the correspondence holds for all cases and for all implementation strategies.

You can view the assignment as one of *metaprogramming*. You will be writing a program in say language A. This program takes another program in language B (the BNF spec for example) as input and translates it into a third program as output which is in language A but does the functionality of the input in language B.

A good emulator will include a lexical scanner and a syntactic parser to assure structural properties of the output.

Assignment IV: Pseudocode Semantics

A three-minute presentation to the class.

Describe the semantics of a small pseudocode fragment.

There is no generally agreed upon model of the semantics, or meaning, of computation. This assignment may require research and creativity.

1. Select a small pseudocode fragment.

2. Define the way it behaves at the register-transfer level. Ie: what interactions with memory occur; what parts are moved and to where; what processes change the configuration of memory. What are the exact changes? This is the *operational semantics*.

3. Describe the assurances that the fragment does what it is supposed to do. Develop a set of preconditions and postconditions. Attempt to use the postconditions to prove the preconditions, and thus to prove the correctness of the program fragment. This is the *axiomatic semantics*.

4. Think about other possible ways that you can clearly and unambiguously define or describe the intended functionality of your code fragment.

Assignment V: A New Method

One written page recounting your experiences.

Explore a new style of programming.

1. Select a programming language and metaphor that you have not previously used. Obtain a copy of the appropriate programming language (this can be time-consuming).

2. Select a small fragment of code that you have written (alternatively you can use your pseudocode fragment from previous assignments, or you can select some code from a published source).

3. Transcribe and implement your code fragment in the new language.

4. What did you learn? Prepare a one to two page report on your experiences. Concentrate on the differences between the languages you are using.

Language Sources:

JAVA (pure object oriented): http://jave.sun.com

Haskell (pure functional): http://haskell.org

Scheme (functional): http://www-swiss.ai.mit/ftpdir/scheme-7.4/

Forth (tiny, threaded) http://chemlab.pc.maricopa.edu/pocket.html

LISP: http://www.franz.com/downloads/

Prolog:
http://www.cs.cmu.edu/Groups/AI/html/faqs/lang/prolog/prg/part2/faq-doc-2.html

ATLAST (tiny, embedded) and DIESEL (tiny, string-based): http://fourmilab.ch/

Screamer (constraint-based extension of LISP, untested):
http://www.cis.upenn.edu/~screamer-tools/index.html

In general: http://dir.yahoo.com/Computers_and_Internet/Programming_Languages/

TEACHING FOR INNOVATION

TOPIC 7. PROJECTS AND ASSIGNMENTS

Teaching Examples (Bricken):

HCI: HCI Assignments

HCI: Interface Design Simulation

AI: LISP Program Modification Exercises

Management: Formal Model: Card Games

HCI: Complete Window System

VR: 3D Interactive Virtual Worlds

VR: Expandable Virtual Cube World

DS&A: Exam Package

HCI Assignments

In addition to readings in the text, (option 2) students are expected to complete two assignments, described below. We will work out the exact content for each student in class.

MID-TERM ASSIGNMENT — IN-DEPTH RESEARCH

Select a small area of HCI that you have interest in and explore it. Prepare a report back to the class about what you learned. Summarize the area in a discussion or oral report of 5-15 minutes.

You can use the web addresses provided earlier to locate writings on your topic of interest.

Content should be some small topic which you can find three or four articles/books on. (I will make suggestions to everyone who asks.) Best would be content that you encountered at work or in school. Another good idea is to select an issue which bothers you about some software of hardware system that you use regularly.

FINAL ASSIGNMENT — DESIGN CRITIQUE

Select one application or system interface that you are quite familiar with. Using the design principles discussed in class, in the lecture notes, and in the textbook readings, analyze and critique this interface. Consider:

the cognitive model
the visual layout
the dialogue management
the interactivity model
the information structure
the integration with i/o devices
the task appropriateness

Build a model of the information structure, following the content being presented and its organizational structure.

Build a model of the interactivity, following the flow of control and communication throughout a dialog transaction.

Then redesign the interface in any areas that you think could be improved. Be specific about what you think is wrong and why your redesign is an improvement.

Justify your critique and your redesign by citing references from the HCI literature.

Interface Design Simulation

Objectives:

Experience HCI design using a detailed task specification. Integrate suggestions for design in the text into a task-oriented design context. Provides a context to discuss design methodology and choices.

Task:

A programming team in your organization has developed a new deductive engine which allows application programmers to manipulate data for expert systems. Your job is to design a prototype interface for this engine.

The engine provides functions for a knowledge engineer to restructure, optimize, verify, and in general manipulate the components of a knowledge base of logical and arithmetic constraints. What is neat about this engine is that it maintains a graphic, network description of the logical transformation processes, and like a circuit, distributes logic over many network nodes.

However, different departments in your organization have different formats for their knowledge-bases, want to do different things to their data, and require different outputs and views of their data. Furthermore, some users want automated functionality and some want fine-grain interactivity with transformations.

Due to organizational preferences, the interface is to be constructed by three separate teams, one team for each of the following aspects:

Aspect 1: function calls to the interface	(programming, API)
Aspect 2: screen layout and interactivity	(interface, dialog)
Aspect 3: hardware i/o devices and functionalities	(architecture)

Fortunately, some members of each team can cross development boundaries and work with the other teams as advocates of their design process.

You are to add appropriate interface controls for things like opening and closing the system, trapping and notifying about input errors, and improper control configurations. Also, you should select appropriate names, labels, and displays for both naive and sophisticated users. You are not responsible for

explaining how the engine works,
the help system, or
the editors which allow databases to be constructed,

although you should include interface hooks to all three subsystems.

You are free to modify and enhance interface specifications to make the engine easy to use, so long as the requested functionality is available.

Below is a (loosely) structured listing of some of the requirements for your interface.

* Backend processor assignment: single, distributed

* Input language: logic, Prolog, Lisp

* Input form: files, keyboard

* Display: linear (textual) view in any input syntax,
  graphic (network) view,
  internal computational view if requested

* Simple transformations: absorb, clarify, extract, coalesce

* Compound transformations: subsume, cancel, collect

* High-level transformations:
  - optimize relative to specified time and complexity parameters
  - identify contradictions, verify consistency
  - delete irrelevant facts
  - cluster facts in groupings relative to a particular set of variables

* Network display controls:
  - select an active subnetwork to perform transformation on
  - rearrange network by hand
  - rearrange network using energy minimization algorithm
    - parameterized by spring coefficient, spring divisor,
      repulsion coefficient and relaxation stepsize
  - labels on or off

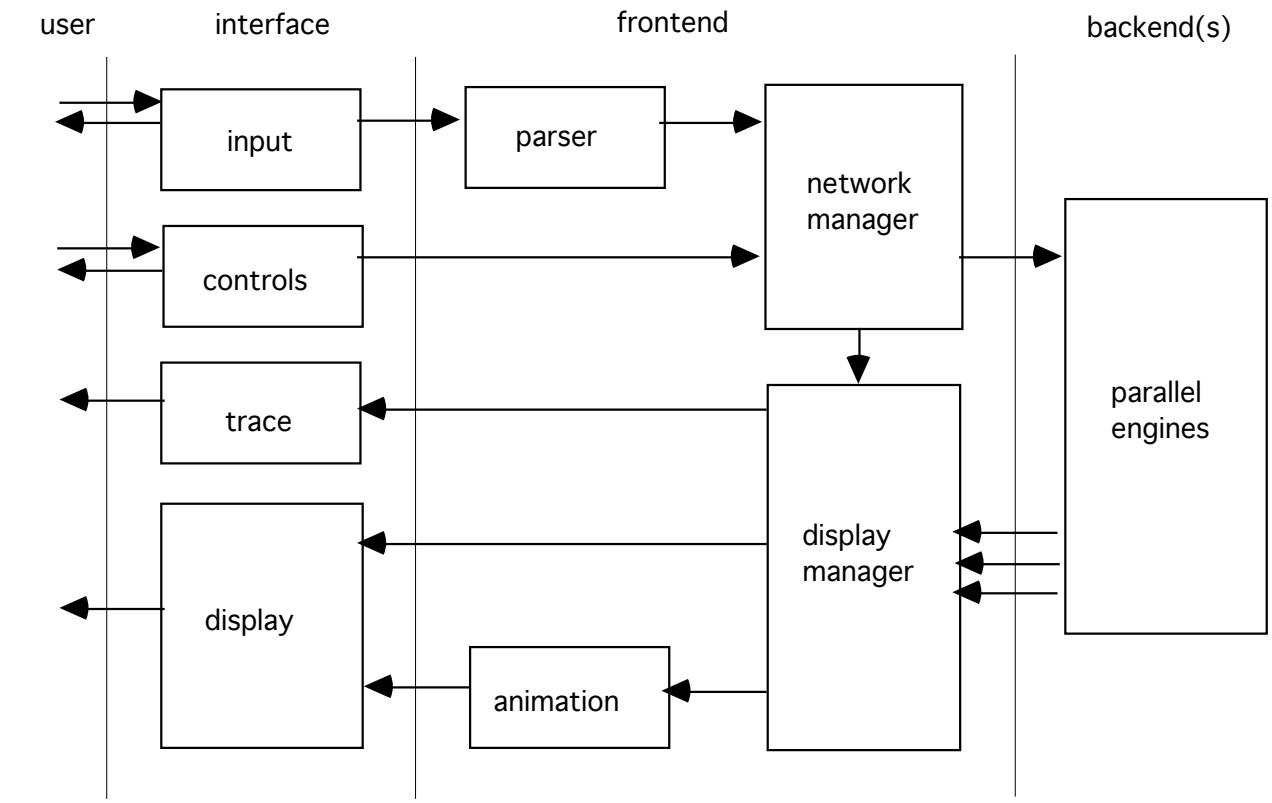
* Network animation:
  - show animation in forward or reverse order
  - stop and start animation freely
  - show active network components and their activity
  - specify rate of animation by
    - frames per second
    - transformations per second
    - specific transformations per second
    - activity indications per second

* Trace:
  - show engine activity by transformations performed and
    by animation instructions performed

* Users also want to be able to :
  - focus on any display with full screen, especially the network display
  - reset display at any point
  - load and display new logical databases
  - refresh display
  - select textual parts of a database for analysis

The engine transformations can be applied individually or in any grouping. The backend engine(s) are much faster than the display, so the display manager collects engine activity and structures a display which makes sense to a person. It is important that the users of the engine understand the logic of the transformations. The engine transformations use an internal coding that is not easy to understand.

Here is the functional architecture of the system:



Backend(s):

Computational machinery on which transformations are done; can be a parallel, distributed array of processors

Frontend:

Manages the interface and coordinates assignment of and communication with backend. Backend coordination can be organized by synchronous or asynchronous message-passing or by shared memory.

Interface:

The information and controls seen by the user.

LISP Program Modification Exercises

The following exercises are in approximate order of difficulty. Students wishing to achieve an "A" grade should attempt most of these exercises. These exercises constitute the bulk of homework and test assignments for the entire course. Most of these exercises ask you to extend the examples. As well as the exercises below, write your own examples. Explore!

All students: Load and run

1. Eliza
2. the recursive Unification algorithm
3. GPS for Blocks World
4. Parcil
5. Streams and filters
6. Logic Programming
7. Object-oriented
8. Parse

The Programs

1. **Eliza**, by Joseph Weizenbaum, transcribed by Peter Norvig.
A famous interactive dialog program, in the form of a Rogerian counselor. Contains a pattern-matcher. Consider how to extend the dialog manager.

2. **Unification and Search**, by Mark Kantrowitz, CMU.
Unification is an extended form of pattern-matching which is integral to inferences engines and query languages. Recursive and iterative versions are included for comparison. read the code, do not worry about the details. The search package is pedagogical, intended to show the similarities across all search algorithms. Although the code is complete, the knowledge representation is missing, so do not expect to run the entire code sample.

3. **GPS**, by Newell and Simon, CMU. Transcribed by Peter Norvig.
The first, famous General Problem Solver. Runs blocks world, so get familiar with the code. Don't forget to use debug mode.

4. **Parcil**, by Erann Gat, JPL.
A C syntax to LISP code recursive descent parser. Not industrial strength.

The next four programs focus on abstraction in coding. They are described in detail in another handout.

5. **Filters and Streams**, by Lugar and Stubblefield, from ideas by Curtis, Waters, and Steele.
Fundamentals of dataflow programming style.

6. **Logic Programming**, by Lugar and Stubblefield, with examples by Bricken.
Fundamentals of declarative logic programming style; an example of meta-linguistic abstraction.

7. **Object-oriented Programming**, by Lugar and Stubblefield.
Fundamentals of object-oriented programming style.

8. **Parse**, by students in Stanford LISP classes.
A recursive descent parser (similar to Parcil), for semi-natural English sentences.

Exercises

1. **Eliza**:

Extend the Eliza program to talk about your favorite subject.

2. **Unify**:

Describe the difference between
a. pattern matching for equality
b. pattern matching with variables
c. unification

3. **GPS**:

Extend the blocks world example to
a. use 4, 5, and 6 blocks
b. have limited table space (the trick is to name blocks which must stay on the table, and disallow moving to the table)
c. do the Towers of Hanoi puzzle
d. use other representations, such as an explicit Hand, an OnTable predicate, a ClearTop predicate.
e. Fix the Sussman anomaly.
f. Use boundary block representations (or others of your choice).

4. **Search**:

Write the missing generic-search functions for Optimal Paths.

5. **(harder) GPS**:

Modify GPS to use smarter search strategies, such as Hill-climbing, Best-first, or Branch-and-Bound.

6. **(harder) Search**:

Use the Kantrowitz generic-search functions to solve Blocks World by providing the knowledge representation as functions for initial-state, goal-p, and children. Evaluate which method is best, either by using the function TIME, or by counting the calls to TRACE.

Use generic-search or GPS to search the state space of some other problem domain, such as Missionaries-and-Cannibals, Sliding-Tiles, Tic-Tac-Toe, Cryptoarithmetic, or N-Queens.

7. **(harder) Parcil**:

Test Parcil on some real but restricted C code. Extend it to include some high-level C constructs.

8. **Streams**:

A. Write a stream/filter program to generate the first N prime numbers.

B. (harder) If you are fluent in another programming language, write the same function in that language and compare the ease of writing, the maintainability/extendibility and modularity, and the readability of each version. Write your other language program so that it handles an *arbitrary* N.

C. Change your program(s) to find the first N prime numbers with two occurrences of the digit 9.

D. Change your program(s) to find the first N prime Fibonacci numbers.

9. **Logic Programming**:

A. Substitute the normal FIRST/REST/CONS/EMPTYP functions for the (simulated) stream functions in the logic code. That is, expand the simulated stream abstraction inline to remove it. Does this improve readability for you?

B. Look at the problem of people liking themselves in the liking examples. Is there a better fix than the one proposed? Write some other liking rules which expose this problem.

C. Add some other facts (not rules) to the three animal databases to exercise the rules which are not used when asking questions about zeke, fred, and tony. What would you do if you wanted to know about types of animals (the animal taxonomy) but not about a particular named animal?

D. There is a major problem with the INFER function. What is it? Simplify the (first, smaller) addition database by removing commutative facts, and add the commutative rule:

```
(rule if ((var a) + (var b) = (var c) (var d))
      then
      ((var b) + (var a) = (var c) (var d)))
```

Figure out what is happening. (harder) Can you fix it?

E. Ask a question about ancestors in the relatives database and figure out what is happening. Ancestor is an example of a transitive relation. (This is related to question D.)

F. Write some other relative rules like GRANDMOTHER, GRANDPARENT, UNCLE.

G. Fix the sibling rule so that a person is not their own sibling.

H. Why is the fact

```
(1 + 0 = 0 1)
```

"out-of-order" in the first addition database? Organize the order of the facts and rules for maximum efficiency.

I. Write an addition rule which abstracts the carry operation out of the database. The trick is to use a carry-flag `((var x) = 1)` which unifies with the fact `(1 = 1)`.

J. (harder) Design an addition rulebase which handles addition of numbers of arbitrary magnitude.

K. (harder) Write a rulebase which solves Cryptoarithms (e.g. SEND+MORE=MONEY). The essence is to add rule(s) which enforce different numbers to be associated with each different letter. Then you might need rules to improve the efficiency of the solution process.

L. (longer but not harder) Write a rulebase for doing multiplication. For doing elementary algebra. For doing differential calculus.

10. **Object-oriented Programming**:

A. Get other object-oriented examples (from OO textbooks perhaps) and build them in the simple LISP system.

B. (harder) Figure out what you would need to do to include multiple inheritance.

11. **Language Parsing**:

A. Extend the grammar to include adverbs and pronouns. This will be (harder) if you have difficulty following the lambda forms in the code.

B. Incorporate PARSE-FIND into the code in place of FIND-* and FIND-?.

C. (harder) Write GENERIC-PARSE.

12. **Miscellaneous**:

A. Write at least five different versions of REVERSE. (harder) Write fifteen significantly different implementations.

B. (harder) Write a semantic net traversal algorithm. This is not hard if you use either the Logic Programming code or the Object-oriented code as a base.

C. Implement the simple robot-in-maze problem. Add the suggested extensions. When you get to adding the Wumpus, it gets (harder).

FORMAL MODELS

We'll learn five card games today. Identify the formal organization of each.

FORMAL ORGANIZATION

mathematics:	domain, operations, axioms
algebra:	pattern, match and substitute, equations
modeling:	state space, state transitions, decision models
human factors:	functional problem space, tasks, strategies
life:	events, property maps, behaviors

THE GAMES

1. ***Pick a Card:***
   Each player draws a card from a standard deck, without looking and without replacement. Everyone looks.

2. ***War:***
   Each player picks a card. Everyone looks. Highest card wins all the other cards.

3. ***Indian Poker:***
   Each player picks a card. Without looking, hold the card to your forehead so that all other players can see it. Simultaneously, every player either folds or bets. Highest betting card wins all bets.

4. ***Psychout:***
   Each player's hand consists of one suit. A different suit defines the point cards (A = 1, ..., K = 13). A point card is exposed. Each player selects a card from the hand, without replacement, as a bid for the point card. Highest bid card wins the point card. Repeat for 13 plays. Highest accumulation of point cards wins.

5. ***Elusis:***
   The game master writes down a secret pattern rule for a sequence of cards. In turn, each player freely selects a card from the deck, without replacement. The game master tells if the card fits the secret pattern. The player to name the secret pattern wins.

A Complete User Interface System

Primary Examples:

MacOS, Visual Basic, NeXTStep, Java, Common Lisp Interface Manager

- **windowing abstraction**
  containers, views

- **display components**
  button, checkbox, choice box, label, list, table,
  scrollbar, textarea, textfield, window, menu, dialog box

- **display tools**
  fonts and points
  color
  graphics system (drawing, clipping, 3D)
  image handling
  layout management

- **temporal data tools**
  time and synchronization model
  sound manager
  video manager
  animation manager

- **interactivity tools**
  event handling and management (mouse, keyboard, arbitraty input devices)
  streams and buffers
  scripting language

- **programming interface tools**
  object-oriented class, instance, and message system (initialize-, make-)
  load, compile, link, and evaluate
  language-specific text editor
  interface construction toolkit
  debugging and exception handling
  namespaces and packages
  foreign function interface

- **operating system tools**
  threads and multitasking
  concurrency, switching, scheduling, and synchronization
  memory management
  file system interface
  network interface and security
  low level: internal data structures, pointers, memory blocks, traps

Extended Examples (Java, CLIM)
using widget interactions as a simple example

Generic object operators/functions:

constructors: make-, initialize-, set-
assessors: get-
queries: ?-
functions: act-on-
relations: constrain-

Turnkey dialog boxes

throw-cancel and catch-cancel <aborts>
message-dialog
yes-or-no-dialog
get-string-from-user-dialog
select-item-from-list-dialog

Windows

nested-views, size, position, scroller, click-handler
title, font, color, active?, layer, zoom, grow, drag

Mac Common Lisp Menu Class structure

menu-element
- menubar (class, variable, function)
  - set-menubar
  - find-menu
  - <color-functions>
  - *default-menubar*
- menu
  - initialize-, set-
    - menu-title, menu-items, menu-colors
    - update-function
    - help-spec (balloon-help system)
  - install, deinstall, installed?
  - enable, disable, enabled?
  - font-style, <color-functions>
  - add-menu-item, remove-menu-item, get-menu-item, find-menu-item
  - menu-item
    - initialize-, set-, get-, query?-
      - owner, title
      - command-key, checked
      - action, action-function (call vs get)
      - colors, font-style
      - disabled?, update-function, help-spec
  - window-menu-item
    - close, save, save-as, save-copy-as, revert, hardcopy
    - cut, copy, paste, clear, select-all, undo, undo-more
    - load/evaluate-selection, load/evaluate-whole-buffer

Mac Common Lisp Dialog-items

initialize-, set-, get-, make-
    view-size, view-container, view-position, view-nickname, view-font
    dialog-item-text, dialog-item-handle, dialog-item-enabled?
    part-color-list, dialog-item-action, help-spec, window-pointer
install, activate, activate-event-handler, default

button-dialog-item
    press-button, default-button-dialog-item (make-, get-, set-, ?-)
static-text-dialog-item
editable-text-dialog-item
<key-stroke-handlers>
check-box-dialog-item (check-box-check, -uncheck, -checked?)
radio-button-dialog-item (radio-button-cluster, -push, -unpush, -pushed?)
table-dialog-item
    <table-constructors>, <cell-contents-handlers>, sequence-dialog-item
pop-up-menu (<handlers>)
scroll-bar (<handlers>)

Interface Toolkit

The toolkit provides a drag-and-drop interface for constructing display interfaces. After selecting and positioning the interface, the toolkit writes the appropriate source code for that interface. Toolkit components:

Menubar Editor
    Add Menu
        Add Menu Item
            Command key, Disabled, Check Mark
            Menu Item Action (provide function), Menu Item Colors
        Menu Colors
        Print Menu Source
    Rotate Menubars
    Add New Menubar
    Delete Menubar
    Menubar Colors
    Print Menubar Source
Use Dialogs (toggle with Design Dialog)
Design Dialogs
    Document
    Document with Grow
    Document with Zoom
    Tool (with title bar and close button)
    Single Edge Box
    Double Edge Box
    Shadow Edge Box
  Design Dialog Methods
    Include Close Box
    Color Window

Add Dialog Item
Static Text
Editable Text Field (Allow Returns, Allow Tabs, Draw Outline)
Button (Default Button)
Radio Button (Radio Button Pushed, Set Item Cluster)
Checkbox (Checkbox Checked)
Table (Set Cell Size, Horizontal Scroll Bar, Vertical Scroll Bar
Set Table Sequence, Set Wrap Length, Orientation)
Add Dialog Item Methods
Dialog Item Text
Enabled/Disabled
Set Item Action
Set Item Font
Set item Name
Set Color
Print Item Source
New Dialog
Add Horizontal Guide (for alignment during editing)
Add Vertical Guide
Edit Dialog
Print Dialog Source

Java Code for constructing some widgets

Named Button:

```
public void okButton() {
        Button b = new button("OK"); add(b); }
```

Unnamed button:

```
add(new Button("OK"))
```

Label:

```
add(new Label("Look at me"))
```

Checkbox:

```
add(new Checkbox("Check here if hungry"))
```

Checkbox Methods:
getLabel(), setLabel(String), getState(), setState(boolean)

Choice Menu:

```
{Choice myClassesMenu = new Choice;
  myClassesMenu.addItem("SE101");
  myClassesMenu.addItem("SE561");
  myClassesMenu.addItem("Special Project");
  add(myClassesMenu); }
```

Choice Menu Methods:
getItem(int), countItems(), getSelectedIndex(),
getSelectedItem, select(int), select(String)

PRODUCTION LISP CODE for a WINDOWING SYSTEM

Unedited, little documentation, good style.
This code is what you would have to write if you were developing an application windowing system without a toolkit or a class library.
Redundant code templates are omitted.

First the **class structure** for the windowing environment,
next the **menu system** with its corresponding **action functions**,
then the **control panel** with its corresponding action functions,
finally the **event handler** for text entry into the control window.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; PARENT-WINDOW

(defclass parent-window (window)
  ((children :accessor children :initarg :children :initform nil)
   (common-data :accessor common-data :initarg :common-data :initform nil)))

(defmethod initialize-instance ((self parent-window) &rest rest)
  (apply #'call-next-method self rest)
  (map-children self #'set-child-parent self))

(defmethod find-parent-child ((self parent-window) type)
  (car (member type (children self) :key #'type)))

(defmethod add-parent-children ((self parent-window) &rest children)
  (setf (children self) (append children (children self))))

(defmethod remove-parent-children ((self parent-window) &rest children)
  (setf (children self) (set-difference (children self) children)))

(defmethod parent-children ((self parent-window) &rest children)
  (apply #'add-parent-children self children)
  (mapcar #'(lambda (child) (set-child-parent child self)) children))

(defmethod map-children ((self parent-window) func &rest args)
  (mapcar #'(lambda (child) (apply func child args)) (children self)))

(defmethod open-child ((self parent-window) type &rest rest)
  (cond
   ((eq type 'entry) self)
   ((find-parent-child self type))
   ((eq type 'database)
    (apply #'make-instance 'database-window :parent self rest))
   (T
    (apply #'make-instance 'display-window
           :type type :parent self rest))))
```

```
(defmethod window-close ((self parent-window))
  (call-next-method self)
  (map-children self #'window-close))

(defmethod set-window-title ((self parent-window) new-title)
  (map-children self #'set-window-title new-title)
  (call-next-method self new-title))

;;;six window subclasses and methods omitted here

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; DISPLAY-WINDOW

(defclass display-window (child-window)
  ((display-view :accessor display-view :initform nil)
   (title :accessor title :initform "Display"))
  (:default-initargs
    :window-type :document-with-zoom
    :view-font '("Monaco" 9)
    :view-size #@(300 150) ))

(defmethod initialize-instance
      ((self display-window) &rest rest &key (type 'display-view))
  (declare (dynamic-extent rest))
  (apply #'call-next-method self :type type rest)
  (let ((view (make-instance type
                :view-container self
                :view-size (subtract-points (view-size self) #@(15 15))
                :view-position #@(0 0)
                :draw-scroller-outline nil)))
    (setf (display-view self) view)
    (setf (title self) (title view))
    (when (parent self)
      (set-window-title self (window-title (parent self))))
    (mapcar #'(lambda (x) (setf (scroll-bar-scroll-size x) 12))
            (view-scroll-bars view))
    (set-common-data view (common-data self))))

(defmethod set-view-size ((self display-window) h &optional v)
  (declare (ignore h v))
  (without-interrupts
   (call-next-method)
   (let* ((new-size (subtract-points (view-size self) #@(15 15))))
       (set-view-size (display-view self) new-size))))

(defmethod window-zoom-event-handler ((self display-window) message)
  (declare (ignore message))
  (without-interrupts
   (call-next-method)
   (let* ((new-size (subtract-points (view-size self) #@(15 15))))
       (set-view-size (display-view self) new-size))))

(defmethod clear ((self display-window))
  (call-next-method self))
```

```
(defmethod save-to-eval ((self display-window))
  `(make-instance 'display-window
     :type ',(type self)
     :window-title ,(window-title self)
     :view-position ,(view-position self)
     :view-size ,(view-size self)  ))

(defmethod window-close ((self display-window))
  (when (parent self)
    (remove-parent-children (parent self) self))
 (call-next-method self))

(defun make-trace-output-window (parent)
  (make-instance 'display-window
    :type 'trace
    :parent parent
    :close-box-p    nil
    :window-title   "Trace Output Window"  ))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; LOSP-MENU

(defvar *losp-menu* nil)
(defvar *db-edit-menu* nil)

(defun initialize-losp-menu ()
  (menu-install (setq *losp-menu* (make-losp-menu)))
  (menu-install (setq *db-edit-menu* (make-db-edit-menu))))

(defun make-losp-menu ()
  (MAKE-INSTANCE 'MENU
    :MENU-TITLE  "Losp"
    :MENU-ITEMS
    (LIST (MAKE-INSTANCE 'MENU-ITEM
            :MENU-ITEM-TITLE    "About..."
            :MENU-ITEM-ACTION   #'make-losp-ABOUT-WINDOW)
          ;(MAKE-INSTANCE 'MENU-ITEM
          ;  :MENU-ITEM-TITLE    "Load"
          ;  :MENU-ITEM-ACTION   #'menu-load-losp)
          (MAKE-INSTANCE 'MENU-ITEM
            :MENU-ITEM-TITLE    "Entry Window"
            :MENU-ITEM-ACTION   #'menu-make-entry-window)
          (MAKE-INSTANCE 'MENU-ITEM
            :MENU-ITEM-TITLE    "Control Panel"
            :MENU-ITEM-ACTION   #'make-losp-CONTROL-PANEL
            :COMMAND-KEY        #\=
            :MENU-ITEM-CHECKED  nil)
          (MAKE-INSTANCE 'MENU-ITEM
            :MENU-ITEM-TITLE    "Test Minimizer"
            :MENU-ITEM-ACTION   #'run-min-test)
          (MAKE-INSTANCE 'MENU-ITEM
            :MENU-ITEM-TITLE    "Quit Losp"
            :MENU-ITEM-ACTION   #'close-LOSP
            :command-key        #\Q)))        )
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; MENU ACTION FUNCTIONS
;;;
;;;   Menu items:             Activation function:
;;;     About...                 make-losp-about-window
;;;     Load                     menu-load-losp    [in initialize file]
;;;     Entry Window             make-entry-window
;;;     Control Panel            make-losp-control-panel
;;;     Test Minimizer           run-min-test
;;;     Quit Losp                close-losp

(defun menu-make-entry-window ()
  (setq *current-entry-window* (make-entry-window))
  ;(make-losp-control-panel))

(defun close-losp ()
  (if *current-entry-window* (window-close *current-entry-window*))
  (menu-deinstall *db-edit-menu*)
  (menu-deinstall *losp-menu*))

;;;several menu functions omitted here

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; ABOUT-LOSP

(defun make-losp-about-window ()
  (modal-dialog
   (MAKE-INSTANCE 'COLOR-DIALOG
     :WINDOW-TYPE      :DOUBLE-EDGE-BOX
     :WINDOW-TITLE     "about-losp"
     :VIEW-POSITION    #@(426 60)
     :VIEW-SIZE        #@(370 185)
     :CLOSE-BOX-P      NIL
     :VIEW-FONT        '("Chicago" 12 :SRCOR :PLAIN)
     :VIEW-SUBVIEWS
     (LIST (MAKE-DIALOG-ITEM
            'STATIC-TEXT-DIALOG-ITEM   #@(58 8)  #@(260 16)
            "Losp Boolean Minimization Engine 1.0"   'NIL)
           (MAKE-DIALOG-ITEM
            'STATIC-TEXT-DIALOG-ITEM   #@(146 31)  #@(73 16)
            "May 1995"   'NIL)
           (MAKE-DIALOG-ITEM
            'STATIC-TEXT-DIALOG-ITEM   #@(8 58)  #@(345 32)
            "Copyright  (C)   1995,    OZ...International, Ltd.  and Interval
Research Corporation, All Rights Reserved."   'NIL)
           (MAKE-DIALOG-ITEM
            'STATIC-TEXT-DIALOG-ITEM   #@(20 102)  #@(345 16)
            "Authored by William Bricken and Jeffrey James."   'NIL)
           (MAKE-DIALOG-ITEM
            'BUTTON-DIALOG-ITEM   #@(130 140)  #@(114 23)
            "OK"
            #'(lambda (item) (declare (ignore item))
               (return-from-modal-dialog t))
            :DEFAULT-BUTTON    T)))   ))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; CONTROL-PANEL

(defun make-losp-control-panel ()
  (setq *problem-number-comtab* (make-comtab))
  (comtab-set-key *problem-number-comtab*
                  '(#\Newline) 'accept-problem-number-text-entry)
  (setq *isolate-variable-comtab* (make-comtab))
  (comtab-set-key *isolate-variable-comtab*
                  '(#\Newline) 'accept-isolate-variable-text-entry)
  (setq *losp-control-panel*
        (MAKE-INSTANCE 'control-panel-window
          :WINDOW-TYPE         :TOOL
          :WINDOW-TITLE        (format nil "Losp Control Panel")
          :VIEW-POSITION       '(:TOP 208)
          :VIEW-SIZE           #@(230 254)
          :VIEW-FONT           '("Chicago" 12 :SRCOR :PLAIN)
          :parent              *current-entry-window*
          :VIEW-SUBVIEWS       (losp-control-panel-subviews)))
  (set-radio-buttons-when-opened)
  (set-logic-check-box-when-opened)
  (set-circuit-check-box-when-opened)
  (set-trace-check-box-when-opened)
  (set-database-check-box-when-opened))

(defun losp-control-panel-subviews ()
  (LIST (MAKE-DIALOG-ITEM
         'STATIC-TEXT-DIALOG-ITEM   #@(10 5)  #@(56 16)
         "Analysis"
         'NIL)
        (MAKE-DIALOG-ITEM
         'BUTTON-DIALOG-ITEM   #@(70 3)  #@(60 18)
         "Apply"
         #'(LAMBDA (ITEM) (apply-button-action item))
         :VIEW-FONT                  '("Courier" 12 :SRCOR :PLAIN)
         :view-nick-name             'apply-button
         :DEFAULT-BUTTON              NIL)
        (MAKE-DIALOG-ITEM
         'RADIO-BUTTON-DIALOG-ITEM   #@(10 28)  #@(110 16)
         "Transcribe"
         #'(LAMBDA (ITEM) (transcribe-radio-button-action item))
         :VIEW-FONT                   '("Geneva" 12 :SRCOR :PLAIN)
         :view-nick-name              'transcribe-radio-button
         :RADIO-BUTTON-PUSHED-P       nil)
        (MAKE-DIALOG-ITEM
         'EDITABLE-TEXT-DIALOG-ITEM   #@(160 222)  #@(52 15)
         ""
         #'(LAMBDA (ITEM) (case-variable-text-action item))
         :VIEW-FONT              '("Geneva" 12 :SRCOR :PLAIN)
         :view-nick-name         'isolate-variable-text-box
         :comtab                 *isolate-variable-comtab*
         :ALLOW-RETURNS          T)  ))

;;;18 other dialog-item specifications omitted here
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; ACTIONS
;;;
;; see process file for usage of the globals
;;   *valid-analysis-levels*  *current-analysis-level*
;;   *active-displays*  *most-recent-analysis-result*
;;   *current-entry-window*

(defun set-radio-buttons-when-opened ()
  (cond
   ((eq *current-analysis-level* '*TRANSCRIBE*)
    (radio-button-push
     (view-named 'transcribe-radio-button *losp-control-panel*)))
   ((eq *current-analysis-level* '*CLEAN*)
    (radio-button-push
     (view-named 'clean-radio-button *losp-control-panel*)))
   ((eq *current-analysis-level* '*SORT*)
    (radio-button-push
     (view-named 'sort-radio-button *losp-control-panel*)))
   ((eq *current-analysis-level* '*EXTRACT-LITERALS*)
    (radio-button-push
     (view-named 'extract-literals-radio-button *losp-control-panel*)))
   ((eq *current-analysis-level* '*CANCEL-BOUNDS*)
    (radio-button-push
     (view-named 'cancel-bounds-radio-button *losp-control-panel*)))
   ((eq *current-analysis-level* '*INSERT-BOUNDS*)
    (radio-button-push
     (view-named 'insert-bounds-radio-button *losp-control-panel*)))
   ((eq *current-analysis-level* '*MINIMIZE*)
    (radio-button-push
     (view-named 'minimize-radio-button *losp-control-panel*)))
   (T nil)))

(defun transcribe-radio-button-action (self)
  (setq *current-analysis-level* '*TRANSCRIBE*)
  self)

(defun clean-radio-button-action (self)
  (setq *current-analysis-level* '*CLEAN*)
  self)

(defun sort-radio-button-action (self)
  (setq *current-analysis-level* '*SORT*)
  self)

(defun database-display-box-action (self)
  (let ((win (when *current-entry-window*
               (find-parent-child *current-entry-window* 'database))))
    (if win
      (window-close win)
      (make-database-window *current-entry-window*)))
  self)

;;;20 other action specifications omitted here
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; ENTER LOSP-ENTRY-WINDOW
;;;
;; Controls the behavior of the 'return' key in the entry buffer.
;; If the cursor is not on the last line of the buffer, 'return' copies
;; the current line (without the prompt) to the end and moves the cursor
;; there too.  No error handling is provided here.

(defun accept-entry (entry-window &optional force)
  (let* ((bmark (fred-buffer entry-window))
         (end (buffer-line-end bmark))
         (eob (buffer-size bmark))
         (entry (string-left-trim *entry-prompt*
                  (buffer-substring bmark (buffer-line-start bmark) end)))
         (symbol-entry (string2symbol-boxed entry)))
    (cond                              ;; accept input
     ((or force (= end eob))
      (set-mark bmark eob)
      (ed-insert-char entry-window #\Newline)
      (if (null symbol-entry)
        (buffer-insert-at-end bmark "return")
        (let ((logic-type
               (intersection (flat symbol-entry) *valid-logic-functions*))
              (assertion-type (member *assertion-token* symbol-entry)))
          (cond
           (logic-type
            (buffer-insert-at-end bmark *multiple-form-message*))
           (assertion-type
            (buffer-insert-at-end bmark "Assert:  ")
            (buffer-insert-at-end bmark
              (assert-entry entry-window (remove-assert-mark symbol-entry))))
           (T (let ((result (process-entry entry-window symbol-entry)))
                (buffer-insert-at-end
                 bmark (prepare-text-out result)))))))
      (ed-insert-char entry-window #\Newline)
      (buffer-insert-at-end bmark *entry-prompt*))
     ;; otherwise copy entry to the end of the buffer
     (T (buffer-insert-at-end bmark entry)))))

(defun force-accept (entry-window) (accept-entry entry-window T))

(defun buffer-insert-at-end (bmark string)
  (buffer-insert bmark string (buffer-size bmark))
  (set-mark bmark (buffer-size bmark)))

(defun prepare-text-out (form)
  (cond
   ((null form)  "")
   ((marked form) "()")
   (T (let ((string-form (symbol2string form)))
      (remove #\) (remove #\( string-form :count 1)
              :count 1 :from-end t)))))

;;;many other handlers and functions omitted here
```

3D Interactive Virtual Worlds

Below is a list of several on-line virtual worlds. Your assignment:

- Select one or several virtual worlds to visit.
- Go inside a world (be careful, most are free, but some require a fee to join). You will probably have to download some specialized interactivity software (free).
- Build an avatar.
- Explore the environment.
- Make a list of the ethical issues that occur to you.

* http://habbo.com
  Very easy to explore.

* http://www.gcoj.com/english/index.html
* http://www.activeworlds.com
* http://www.everquest.com
* http://www.uo.com
* http://www.furcadia.com
  Adventure worlds

* http://www.ntts.com/ispace.html
* http://www.worlds.com
  Commercial worlds

* http://www.sics.se/dive
  One of my UW students built this world.

* http://ultravixen.com
  CAUTION: pornographic adventure world

Want to read about it?
* http://imaginaryrealities.imaginary.com/article_index.html
* http://www.rider.edu/users/suler/psycyber/psycyber.html

THE STRUCTURE OF A CUBE

The *key idea* is that the structure (geometry) of an object is an intrinsic property. Structure should make no reference to external relations.

Note that translation, rotation, scale, and orientation are Relations between an object and an external coordinate system, and are thus not part of a cube's geometry.

Fortunately, there are established conceptual tools (Cartesian geometry, unit vectors) for describing "cube space".

EMBED THE CUBE IN A SPACE

Assume unit vectors i, j, and k. Associate each with an orthogonal side of the Cube.

Given rules for ijk: i*j = i*k = j*k = 0

Assume a local origin (0i 0j 0k).

```
i = (1i 0j 0k)
j = (0i 1j 0k)
k = (0i 0j 1k)
```

DIFFERENTIATE PARTS

Cubes have 27 parts: 8 vertices, 12 edges, 6 faces, 1 volume.

Notation: (ai bj ck) for all parts.

Let {a, b, c} take on three possible states: {0, _, 1},

where _ is any value 0 =< _ =< 1

Let d = {0, 1} (Knonecker delta, either 0 or 1)

```
Vertices:  {di dj dk}
Edges:     {di dj _k} or {di _j dk} or {_i dj dk}
Faces:     {di _j _k} or {_i dj _k} or {_i _j dk}
Solid:     {_i _j _k}
```

More notation:

Let i, j, and k be symmetrically equivalent, and thus unlabeled.

Vertices:	{d d d}	(all three states are Kronecker)
Edges:	{d d _}	(one state is not Kronecker)
Faces:	{d _ _}	(only one state is Kronecker)
Solid:	{_ _ _}	(no state is Kronecker)

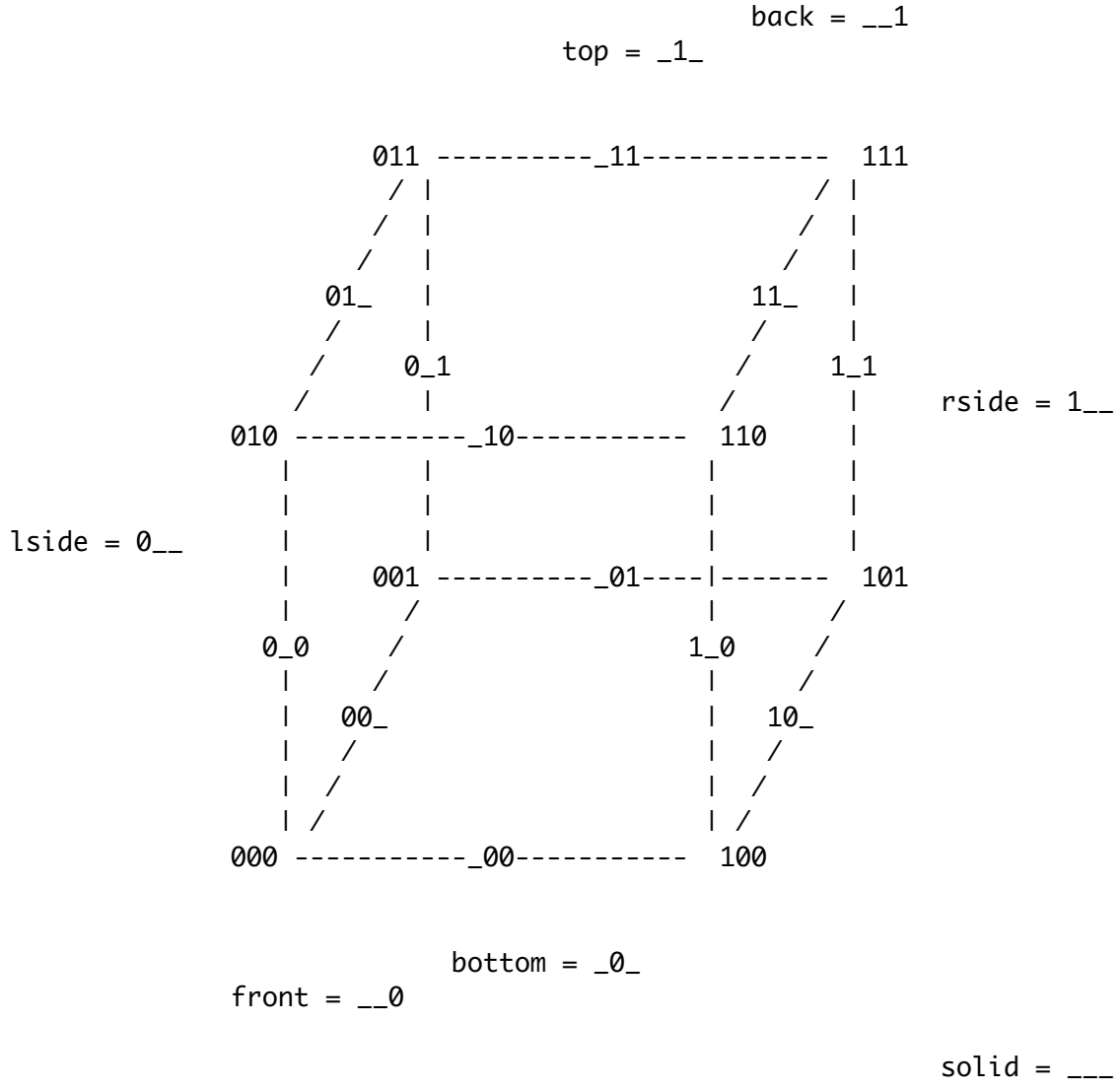
Let u stand for any of i, j, or k.

PROPERTIES

```
Parallel(e1_ e2_) =    e1{d d _} = e2{d d _}            _ in same location
Parallel(f1_ f2_) =    f1{d _ _} = f2{d _ _}            _ _ in same location


Perpendicular(e1_ e2_) =    not(Parallel(e1 e2))
Perpendicular(f1_ f2_) =    not(Parallel(f1 f2))


On(v_ e_) =    v{du} = e{du}                            values of d equal
On(v_ f_) =    v{du} = f{du}                            value of d equal
On(e_ f_) =    e{du} = f{du}                            value of d equal


Meets(e1_ e2_) =     e1{du} = e2{du}                    some d equal
Meets(f1_ f2_) =     not(Parallel(f1 f2))


Distance(v1_ v2_) =    number of different {du}
Distance(e1_ e2_) =    number of different {du}
```

PICTORIALLY

```
                    011  _11  111
                    0_1  __1  1_1
                    001  _01  101

                 01_  _1_  11_
                 0__  ___  1__
                 00_  _0_  10_

           010  _10  110
           0_0  __0  1_0
           000  _00  100
```



MULTIPLICATION TABLES

To determine vertex of intersection of two edges (or edge of intersection of two faces, or more generally, lower dimensional element defined by two other elements), down-multiply representation:

*	0	_	1
0	0	0	_
_	0	_	1
1	_	1	1

To determine edge formed by two vertices (or general up element), up-multiply representations:

%	0	_	1
0	0	_	_
_	_	_	_
1	_	_	1

Note than non-intersecting vertices identify faces (or solids)

NOTES ON REPRESENTATION

By multiplying i, j, or k by a scalar, the cube generalizes to an arbitrary block.

ijk provides lots of established mathematical support.

{0 _ 1} provides unification of different parts of a cube and visual imagery.

Binary Kronecker delta provides easy implementation, but could be renamed (0 = low, 1 = high, _ = any) for understanding.

Properties are trivial calculations.

Generality of notation is difficult to express algebraically. In general, the more abstract, the more powerful and the harder to express.

ALGEBRAIC SPECIFICATION EXAMPLES, CUBE

```
GenericCube = {

      ;the structure of the SPACE embodying cubeness

      USE[ UnitVector ] = { i j k }
      V = { 0 - 1 }
      D = { 0 1 }
      < vi_V, vj_V, vk_V > = [vi, vj, vk] * [i, j, k]T
      < di_D, dj_D, dk_D > = [di, dj, dk] * [i, j, k]T

      origin = < 0, 0, 0 >
      center = < .5, .5, .5 >

      ;the PARTS of a cube, the DOMAIN of elementary elements

      PART = { < vi_, vj_, vk_> }
      VIRTEX = { < di_, dj_, dk_ > }
      EDGE = { <di_, dj_, ->  <di_, -, dk_>  <-, dj_, dk_> }
      FACE = { <di_, -, ->  <-, dj_, ->  <-, -, dk_> }
      SELF = { <-, -, -> }

      ;the operator which yields properties of the cube

      (p1_PART ^* p2_PART) = < ^*[p1.i p2.i] ^*[p1.j p2.j] ^*[p1.k p2.k] >

      ^*[ 0 0 ] = 0
      ^*[ 0 - ] = 0
      ^*[ 0 1 ] = -
      ^*[ 1 - ] = 1
      ^*[ 1 1 ] = 1
      ^*[ - - ] = -

      ;properties

      parallel[ p1_PART p2_PART ] = {

            p1_EDGE ^* p2_EDGE = _FACE
            p1_EDGE ^* p2_EDGE = _SOLID
            p1_EDGE ^* p2_FACE = _EDGE
            p1_EDGE ^* p2_FACE = _FACE
            p1_FACE ^* p2_FACE = _SOLID
            }
```

```
      perpendicular[ p1_PART p2_PART] = {

            p1_EDGE ^* p2_EDGE = _VIRTEX
            p1_EDGE ^* p2_FACE = _VIRTEX
            p1_FACE ^* p2_FACE = _EDGE
            }


      skew[ p1_PART p2_PART ] =

            p1_EDGE ^* p2_EDGE = _EDGE


      on[ p1_PART, p2_PART ] = {

            p1_VIRTEX ^* p2_VIRTEX = _VIRTEX
            p1_VIRTEX ^* p2_EDGE = _VIRTEX
            p1_VIRTEX ^* p2_FACE = _VIRTEX
            }


      connectedby[ p1_PART, p2_PART ] = {

            p1_VIRTEX ^* p2_VIRTEX = _EDGE
            p1_VIRTEX ^* p2_VIRTEX = _FACE
            p1_VIRTEX ^* p2_VIRTEX = _SOLID
            p1_VIRTEX ^* p2_EDGE = _EDGE
            p1_VIRTEX ^* p2_EDGE = _FACE
            p1_VIRTEX ^* p2_FACE = _FACE
              }

GenericBlock = {

      USE[ GenericCube ] = { cube }
      BLOCK = { [a1_ARITH, a2_ARITH, a3_ARITH] * [cube.i, cube.j, cube.k]T }
      IsCube[ b_BLOCK ] = ( b.a1 = b.a2 = b.a3 )
      }
```

```
CubesInaCube = {

      USE[ GenericBlock ] = { world b1 ... }
      worldscale = [1000, 1000, 1000]
      bigworld = worldscale * world
      location[ b_ ] = < bi, bj, bk >
      InWorld[b_] =
        (( <0,0,0> <= location[ b ] >= worldscale * <1,1,1> ) = true )
      location[b1] = <0,0,0>
      location[b2] = <1,0,0>
      }


StackOfBlocks = {

      USE[ GenericBlock ] = { world b1 ... }
      STACK = { [[ b1_ ... ]] }
      CONFIGURATION = { [[ b1___ ]]__ }

      emptytable = [[ ]]
      [[ ]] [[ ]] = [[ ]]
      location[ emptytable ] = < _, 0, _ >

      PutBlockOnStack[ b1_, s_CONFIGURATION ] =
        ( [[ b1 ]] [[ s ]] = [[ b1, s ]] )

      TakeBlockOffStack[ b1_, s_CONFIGURATION ] =
        ( [[ b1, s ]] = [[ b1 ]] [[ s ]] )

      On[ b1_, b2_ ] =
        ( [[ ___, b1, b2, ___ ]] = true )

      Above[ b1_, b2_ ] =
        ( [[ ___, b1, ___, b2, ___ ]] = true )

      OnTable[ b_ ] = ( [[ ___, b ]] = true )

      OnTopOfStack[ b_ ] = ( [[ b, ___ ]] = true )
```

EXPANDABLE VIRTUAL CUBE WORLD
DESIGN/DEVELOPMENT/SPECIFICATION ASSIGNMENT

Using the VR specification language, do as many of the following tasks as you can. Those working in groups should attempt more.

1. Specify the geometry of a cube.

2. Specify some properties of a cube. Choose properties that permit some specific cube functionality.

3. Specify some transformations on a cube.

4. Specify an environmental cube and a contained object cube.

5. Specify some form of interaction with a cube, using a defined device such as the glove, the wand, or the spaceball.

6. Add some more cubes and specify some ways in which they relate.

7. Specify some multisensory viewpoints on a cube.

8. Specify a disposition of a cube. Choose behaviors that permit some specific cube goals.

Combine the above specifications to build a world:

9. Block and Wand: A wand (or a spaceball, or ...) is used to manipulate a block.

10. Blocks World: pick up blocks and build structures with them.

11. Logic blocks: block structures map onto propositional calculus and prove theorems.

12. Block structure builder: name a particular configuration of blocks, the existing configuration will rearrange itself to form the target configuration.

13. Block structure builder + restructuring baby: Blocks will take steps to rearrange into a particular configuration while a baby dynamically changes the existing configuration.

14. Block obstacles: Move a virtual body through a maze of blocks.

15. Topple blocks: Remove blocks from a block structure until it falls down.

16. Architectural blocks: Configurations of blocks represent architectural spaces. Write design constraints for a building or a community.

17. Creative blocks: make up your own block world interactions.

Assignment I

Mapping Your Knowledge of Data Structures and Algorithms

One or two pages to be handed in to the instructor at classtime.
Time allocation: two hours thinking, two hours writing

Construct an outline of what you know about data structures and algorithms.

A **data abstraction** is a way to organize computational information and consequently computer memory. It usually consists of a storage representation and a set of operations to construct, access, modify, delete, deconstruct, test for membership, and/or display stored instances.

An **algorithm** is the structured computational process which converts data into a solution to a particular set of problems. Useful algorithms usually apply to large classes of problems.

These concepts can be understood at **different levels of the design hierarchy**. Data structures and algorithms can be seen

- as ways of implementing low level computational processes,
- as ways of structuring an implementation in a programming language,
- as ways of organizing pseudo-code in preparation for implementation,
- as ways of constructing the mathematical model of a problem, and
- as ways of thinking about and coming to understand a problem space.

How to outline your knowledge:

There are many different ways to collect and organize what you know about a particular topic. The most important thing to recognize is that each person is unique and has a unique understanding of the world. Therefore you should use an outlining technique which feels most comfortable to you. Some choices include

- list major topics and minor topics, similar to the chapter organization of a textbook
- collect words which you can define, and how they relate to each other, similar to a semantic network of object nodes and relational links
- form clusters of similar ideas
- rank the topics covered in the textbook in order of your confidence of understanding
- write down all the things that you have heard about but do not understand
- copy someone else's organization of the topic, marking what you understand and don't
- draw a picture of what you see when you visualize the topic

Remember: Outlines are short. This assignment is not asking you to demonstrate what you know, only to indicate strong and weak areas of knowledge. You do not need to include any form of justification, rationale, or documentation.

Final suggestion: It is often very useful to indicate the degree of confidence you may have for your understanding of a particular topic, as well as the degree of understanding itself. Be sure to test your understanding by asking things like:

- what is the definition? how is this used?
- have I ever actually used this in an implementation? was it successful?
- do I know when not to use this? do I know how to select between alternatives?

Assignment 2: Data Structure Hierarchy

You will not be turning in this assignment.
Time allocation (max): thinking, 2 hour; mapping, 3 hours

Build a type hierarchy for the common data structures.

Many data structures depend on other data structures for their definition. For example, rational numbers depend upon integers, since rational numbers are composed of two integers.

Here is *a listing of most common data types*. Can you construct an inheritance hierarchy which defines their dependencies? You can claim that some entries are not data structures. It will also help if you divide the task into subgroups such as elementary data structures, efficient storage structures, mathematical structures, exotic structures, implementation hacks, etc.

bit
byte
bit stream
character
string
stream

truth value
boolean (proposition)
propositional sentence
function
relation
equality relation
partial ordering relation
total ordering relation

set
bag (multiset)
infinite set
ordering (ranking)
non-negative integer
integer
rational number
real number (specific precision)
complex number
random number

object
class
pattern
persistent object

array
sequence
list
linked list
doubly linked list
association list (bucket)
circular list

stack
queue
priority queue
vector
matrix
table

dictionary

tree
binary search tree
balanced binary tree
red-black tree
B-tree
splay tree
binomial heap
fibonacci heap
graph
directed graph
directed acyclic graph
inheritance graph

tuple

hash table

font
screen image
point
2D graphic
3D graphic
sound bite
video image
video stream
color

hyperlink
URL
socket
button
checkbox
panel
window
scrollbar
menu

equation
procedure
buffer
error (exception)
file
directory
continuation
namespace
package
script
pointer

Challenge problem: include *all* of the data structures above in the inheritance hierarchy, noting the arbitrary design decisions (ie some forms support a choice of subcomponents).

Assignment 3: ADS for SUBSTRINGS

You will not be turning in this assignment.
Time allocation (max): thinking, 1 hour; pseudocode, 3 hours

Design an abstract data structure for the data type SUBSTRING.

1. Select a set of *tokens* to represent constants and variables in the domain.

2. Identify the *component parts* (the subtypes) of the data structure, and the functions used to recognize those parts (eg empty-substring, character, substring).

3. Identify the *decomposition axiom* which specifies how to construct and take apart SUBSTRING objects. This axiom will include the definition of *accessor functions* which access parts of a SUBSTRING.

4. Identify a *constructor function* (again defined in the decomposition axiom) which permits building compound substrings out of simple substrings. Consider the difference between a proper substring (A proper subcomponent is a component which is always smaller than its container.)

5. Identify some rules, or *invariants*, which hold between component parts of the data structure. These define the "methods" of the object type.

6. Name some *facts* which are true of this data structure. These define the type hierarchy and the other constraints on the object.

7. Identify the *induction principle* for SUBSTRINGS.

8. Using the induction principle, write pseudocode for some simple recursive functions/methods which implement the invariants of the structure.

9. Finally, suggest some computational data structures which would be appropriate for implementing the SUBSTRING ADS.

Here are some axioms you may need:

The definition of a substring:

```
x sub y  =def=  z1*x*z2 = y
```

The empty string is a substring of every string

```
E sub y
```

No string is a substring of the empty string

```
not(y sub E)
```

Prefixing a character to a string maintains the substring relation

```
if (x sub y) then (x sub u·y)
```

The following three properties of the substring relation establish that *substring is an ordering relation.*

transitivity	if `s1` is a substring of `s2`, and `s2` is a substring of `s3`, then `s1` is a substring of `s3`
antisymmetry	if two strings are substrings of each other, they are equal
reflexivity	a string is a substring of itself

Prove or define the above relations. Then prove:

- A string is a substring of itself when a character is prefixed.
- A string is a substring of the empty string when it is the empty string.
- Substring implies all the characters in the substring are in the string.
- The length of a substring is equal to or less than the length of the string.

Extend the results:

The definition of a **proper** substring:

```
x proper-sub y  =def=  not(z1=E and z2=E) and z1*x*z2 = y
```

Prove the properties of *proper* substrings (transitivity, irreflexivity, asymmetry)

Assignment 4: ADS for a Square

Turn in the assignment at the beginning of class.
Time allocation (max): thinking, 1 hour; design, 3 hours; implementation, 5 hours

Implement an abstract data structure for the object "square".

CONCEPTUALIZATION

Define the object and the invariant relations between its parts.
Select a mathematical formalism which suits your conceptualization.
Separate internal and external transformations.
    An external transformation refers to an external coordinate system or origin.

MODELING

Identify:
    Domain
        the component parts of the object
        the appropriate recognizers, accessors, and constructors
    Properties
        the uniqueness relation which defines equality tests
        the containment relationships between the components
        the relevant relations between components at the same level
    Functions
        all relevant functions between components
        some interesting functions between the object
            and an external coordinate system

IMPLEMENTATION

Select:
    a data structure for the object and its components
    data structure transformations between components
    implementation language or algorithm strategy for functions
Write:
    a make function which builds accessors automatically
    a get function for locating each part of the object
    methods/functions and predicates defined above
    use the Induction Principle to write the recognizer `isa-square`

EFFICIENCY

Translate your implementation into bit manipulations.

CHALLENGE ASSIGNMENT

Implement the object "cube",

or more generally still, the object “N-dimensional cube”.

Final Assignment: Control Structures

A three to five minute presentation to the class on your approach.
Time allocation (max): thinking, 8 hours; designing, 8 hours; implementing 20 hours
HAND IN YOUR WORK (notes, code, comments, results)

Select from and complete as many of the exercises as you can within the time allocation.

I. Sorting

Comprehension exercise: Implement several *sorting algorithms* (possibly by copying algorithms from the book). Design and implement a *record generator* which generates random collections of record indices (ie numbers or key words). Design and implement a simple *test statistics package* which counts the number of sorting steps for each sorting algorithm. Answer these questions:

1. Make a graph of sorting steps (record swaps or relocations) vs size of input (number of records) for each sorting algorithm. Do your algorithms perform as the book predicts?

2. Characterize the essential difference between each algorithm. Why do some algorithms perform better or worse than others?

3. Design several different input orderings to sharpen the difference in performance between your algorithms. That is, build input sets which are almost completely ordered, almost completely out-of-order, in some random ordering, each value duplicated once, all the same value, etc. Vary the size the input sets and their ordering characteristics, and test the efficiency of your algorithms.

4. For all experiments, abstract the performance in terms of asymptotic notation.

5. Characterize the stability of each algorithm. That is, differentiate between sorting which may move elements with the same key, and those that will not move them.

6. ***Hybrid algorithm* (challenge):** mix what appear to you to be the best parts of your algorithms, to build a hybrid sorting algorithm with better characteristics than the ones you started with.

II. Algorithm Variety

1. Factorial: Implement many substantively different versions of the factorial algorithm (factorial computes the product of 1..n integers). Compare each for efficiency and for ease of writing and maintenance.

If you look on the web, there are collections of dozens of ways to implement factorial (that is, this problem can be addressed with research as well as with creativity). Some methods require an appropriate engine, which you may or may not have. For example, an object-oriented implementation requires an object-oriented language. Don't implement engines, do use different languages when appropriate.

Can you find the fastest possible algorithm?

2. Fibonacci: Implement many substantively different versions of the Fibonacci algorithm Fibonacci computes the sum of the previous two Fibonacci numbers:

```
Fib[0] = 0
Fib[1] = 1
Fib[n] =  Fib[n-1] + Fib[n-2]
```

Compare for efficiency and ease of use. Identify three classes of algorithm (highly inefficient, efficient, and highly efficient) with an implementation of each.

***3. Generalization* (challenge)**: Design and implement an abstract procedure which computes any simple recursive function, given the base and the recursion relation.

III. Tree Searching

Assume that you have a tree data structure and you have to search it for a leaf with a particular value.

1. Searching: Design and implement several search algorithms for visiting the leaves of the tree. Standard techniques include depth-first, breadth-first, best-first, hill-climbing, iterative deepening, and iterative broadening. Which are most efficient and why?

2. Structured search: In what ways can specialized tree data structures (such as balanced search trees) help to improve the efficiency and ease of programming search algorithms? Analyze the tradeoff between complex data structures for search and complex search algorithms.

3. Generic search: Design and implement a single algorithm which takes the type of search as a parameter. Other generalization parameters which may be of interest include

- the goal predicate which tests when the leaf being sought after is found
- a function which returns the children of an interior node in the tree. (This is used for determining the cost of various search strategies.)
- a priority function which determines which node to search next.

***4. Smart search* (challenge):** Design and implement an algorithm which dynamically determines which kind of search is best for the particular context, given the depth, branchiness, and other characteristics of the tree at the node currently being explored.

Final Assignment (option): ADS for Control Structures

Design an abstract data structure for program control structures.

Caution: This exercise is too difficult to assign a legitimate due date or time allocation, because of its exploratory nature. No one has been able to do the task of control structure abstraction well. Thus, this exercise requires a research mentality. Think about how to do it, what the task requires, the tools you have available, and where the hard and easy parts are. Sketch a partial solution and do the parts that you believe are possible. Almost all time should be spent puzzling about what the exercise means. Do not expect to complete any subparts of this assignment.

Definition: Program control structures are those components of a program which determine the sequence and style of program execution. The *semantics*, or meaning, of a program is defined by its behavior, which is guided by control structures. The major program-level control structures are:

- logic (Boolean primitives such as AND, OR, NOT, EQUALS, IF-THEN-ELSE, CASE)
- assignment (assigning names to return values)
- loops (structures which repeat instructions a specific number of times) including iteration, recursion, and {DO, FOR, WHILE, UNTIL}
- sequencing (calls that are executed sequentially as they are written in the program)
- function invocation (calls to specific functions)
- mapping (applying a function to a collection of items)
- catch and throw (jumps from one part of the program to another, usually in exceptional circumstances)

Level of effort: First try to design and implement an ADS which generalizes one of the above control structures. The above structures are elementary, similar to the elementary data structures of bits, arrays and pointers. The idea is to build higher-level control structures from these primitives. The following discussion may help you to choose a difficulty level.

Logic is expressed by propositional calculus or by Boolean algebra. An ADS for logic would identify possible Boolean structures (the propositions in propositional logic can be functions that return a specific value). Most languages extend the semantics of logic so that any return is viewed as True. But how would you specify a logic structure abstractly?

Assignment looks simple, but has been one of the most difficult concepts to understand. Naming is inherently tricky, and naming to a memory location is trickier still. Scoping rules make the duration of assignment tricky also. Several languages do not use assignment, it is not an essential concept. Good programming isolates all assignments. ASSIGNMENT is still very much in use, but it is beginning to look like a bad idea. It is being replaced by LET and by parameter passing in FUNCTION INVOCATION.

Loops are a dominant tool for repetitive actions. They come in many varieties. It is easy translate between FOR, DO, WHILE and UNTIL. Converting between iteration and recursion is significantly harder, but not tricky. Iteration repeats over a data structure (the loop index); recursion repeats over a function invocation. Newer repetitive techniques use an *iterator*

data/control structure which generates items as needed. *Streams* and *mappings* can also drive a repetitive call without introducing looping.

Sequencing is fairly easy if it is not mixed with other control structures. Assignment and goto make sequencing particularly difficult.

Function invocation is captured by the rules of lambda calculus, and is not difficult. Since lambda calculus provides two evaluation regimes, there are significant design decisions about what to evaluate when. This has driven the debate between *eager and lazy evaluation*.

Mapping is easy and is widely under utilized in languages. It is very similar to implicit looping.

Catch and throw are dynamic exits into non-local environments. They are principled and easy to model since they essentially throw away intermediate results. But since they cross algorithmic boundaries, they have very limited use (ie for exception handling and occasionally when a loop must be interrupted). GOTO (jumps out of a program without encapsulation) is a control structure which is antiquated and no longer in use.

Algorithms: When the simple control structures above are combined to make a compound program, that program is an algorithm. Your ADS should be able to handle algorithms. The essential utility of the control structure ADS is to convert algorithms which do the same task into different programming approaches or metaphors. This is called a *meta-protocol*. We all know that we can choose between different ways of implementing a specification, iteration vs recursion for example. Your ADS should be able to isolate the choice of a particular control structure from the meaning of the specification.

Examine several SORT algorithms. How do they differ in control structure? How can the concept of sorting be separated from the implementation of sorting? How can the concept of sorting be separated for the various sorting algorithms?

Examine several tree traversal algorithms. How do they differ? How can they be abstracted into one algorithm which steers the search based on a parameter?

What other compound control structures do you commonly use? How would you design a program which could switch between members of a family of algorithms (e.g. the SORT family) depending on the incoming data structure, expectations about the average data structure being processed, and the available computing resources?

Discussion: This assignment is intended to let you think about a hard problem, to use the ADS template on difficult structures in unique ways. Be sure to follow the ADS guidelines. Be sure that you understand the limitations of your language of choice; often language design will preclude your access to manipulating control structures (this is because it is easier to implement languages when control is fixed by a design choice). For example, C++ uses assignment, sequence (or *compound*), logic, and for-loops as elementary control structures.

Stick to a subset of your programming language which avoids the difficult constructs. (Then learn to program like this all the time, never using the problematic ideas of assignment, goto, and possibly loops and variables.) Assume a **pure language** which does not include very tricky concepts like memory allocation and deallocation, destructive operations on memory, dynamic scoping, global variables, and non-local jumps.

TEACHING FOR INNOVATION

TOPIC 8. SMALL GROUP ACTIVITIES

Managing Learner-Instructor Interaction and Feedback

TP: Group Presentations

TP: Establishing Ground Rules for Groups

TP: Integrating Team Exercises with Other Course Work

TP: Peer Instruction

TP: Difference Between Cooperative and Collaborative Learning

Teaching Examples (Bricken):

Management: simulation game

Management: archeologist, telephone drawing, consensus

Managing Learner-instructor Interaction and Feedback

Garrison, 1990 – "It has been found that students who interacted regularly with their instructor and with other students were more motivated and had better learning experiences."

Oliver & McLoughlin, 1997 – "Communicative interactions can be used to engage learners, to cause them to reflect on and to articulate ideas. Interactions encourage and facilitate cognition and play an important part in promoting learners' intellectual operations and thinking processes."

Rationale

* Students should not have to wait until after failing a midterm exam to find that they aren't learning what the instructor expects them to learn.
* Instructors do need to be available and therefore need to have a sensible plan for interacting with students and providing feedback.
* "few chances to interact with the instructor limits students' ability to clarify and negotiate instructional goals, explore alternative methods, or construct meaning within in a social context based on personal knowledge" (Garrison, 1993).

Types of Instructor-Student Interaction

The type of interaction on the Web site refers to learner-instructor interaction. The goal of the section is to make instructors consider how they can manage and regulate interaction with students so that interaction is not excessively time-consuming.

Learner-instructor Interaction in a course should:

* Stimulate and maintain the learner's interest
* Motivate the learner to learn
* Provide counsel, support and encouragement to each learner
* Provide timely feedback to learners to make sure they are making progress

Due to the busy schedule and multiple responsibilities of instructors in higher education, they cannot be available at all times to students. They may not have enough time to look at, use, grade, and give feedback for each activity. Therefore, there much be a plan for learner-instructor interaction.

General Large Class Tips

To encourage more interaction even in larger classes, try the following

* Walk around the lecture before class begins. You can even help handing out notes.
* Note if you will be staying a few minutes after class to answer questions.
* Address questions to specific groups of students (e.g. freshmen, people living off-campus).
* Provide an inbox (a real box or a virtual discussion area online) for student questions outside of class. You can also allow for anonymous submissions.
* Appropriately praise questions students may ask (e.g. "Good follow-up" or "Yes, that's a typo...good catch").
* Create a seating chart to learn student names.
* Upload lecture notes into ANGEL or other course space. This ensures that students have all data points, graphs, quotations or citations mentioned in the class.
* Avoid reading from a script (unless you are pre-recording audio for an online presentation). Many instructors use bullet points as mental ticklers of what they want to say in full.

Ideas for managing student feedback on assignments

It's important for students to receive feedback on how much course content they have understood, yet grading a large number of assignments can be daunting. Here are some tips to manage the load:

Fact Check Exercises

Students typically need to master basic facts before thay can move to more advanced analytic topics

* Low-stakes, self-scoring quizzes in ANGEL.
* Short in-class polls or quizzes.
* Require quizzes to check if students have read material

Discussion Board Assignments

Student posts should be monitored to ensure assignments are progressing as expected.

* Use a simple check/check-minus grading system for most posts (like in a class discussion)
* Summarize your comments instead of replying to each student

Weekly Assignments

For many courses, students need feedback before a midterm, and few students do practice assignments unless they are graded.

* Provide an answer key for simple problems, and grade only harder problems.
* Require answers that require research, yet are easy for instructors to scan (e.g. a specific number, derivation or fact).
* Incorporate easier exercises into a class session to assess comprehension and provide variety in the class session.

Student-Student Interactions

Another channel of communication is student-to-student. Encouraging students to interact with each other can make classroom atmosphere friendlier and allow students to explain concepts to each other (sometimes a student will be able to put a different twist to a concept that is still accurate).

* Break students into impromptu groups or pairs to solve more complex problems in class.
* Break up large classes into smaller online discussion sections so there's a cohort who knows each other.
* Create a policy to allow students to communicate with each other to get help on assignments even if they are required to turn in their own formulations.
* Use intro-surveys to determine if students are "experts" in related areas. For example a linguistics class may want to know what dialects or languages students natively speak. A geology class may want to know what geological regions students grew up in.

More Student Involvement

Student Mentors: A higher level student may be able to tutor

* Students may find another student less intimidating
* A student mentor may have an alternate explanation that is more relatable.
* Use undergraduates who have already taken the course.
* Encourage more experienced students in class to answer basic questions.

Learning Teams: Students are assigned to small groups to work on problems together.

* Students are in a smaller cohort and may be more motivated to attend class
* Students work together to interpret content

* Assign permanent teams early in the semester.
* Make sure team members have a chance to get to know each other before high-stakes team work begins.
* Make sure grade includes individual performance portion as well as team participation.
* Clarify expectations including how conflicts can be addressed.

Student Discussion Leaders

* Student discussion leaders must read material more in-depth to prepare questions.
* Students are exposed to multiple perspectives in analysis.
* Student leaders may ask questions in an original way.
* Assign Leadership tasks on a rotating basis
* Require students to respond to discussion leader questions.
* Encourage additional questions from non-leaders.

Replace Lecture with Student Presentations

* Students can practice their presentation and management skills.
* Students learn to analyze content readings instead of just the pre-digested lecture.
* Instructor can monitor content presentation, but not have to create a full set of lecture notes.
* Students present main points in the lessons or readings.
* Assign topics to students on a rotating basis.
* Provide guidelines on how you want presentations to be structured.
* Be available to answer questions from student presenters.

Peer Reviews: Students critique project work of others

* Students get feedback from multiple perspectives
* Students learn how to adjust projects for multiple audiences.
* Create a peer evaluation rubric for students to use.
* Make quality of critiques part of the critiquer's grade.
* Allow students to respond constructively to critiques.

GROUP PRESENTATIONS

In the group presentation or lecture method, the instructor tells, shows, demonstrates, dramatizes, or otherwise disseminates subject content to a group of learners. This pattern can be utilized in a classroom, an auditorium, or a variety of locations through the use of radio, amplified telephone, closed-circuit television transmission, interactive distance television, or satellite communication (teleconferencing).

While lecturing, the teacher may include media materials, such as transparencies, recordings, slides, video recordings, or multimedia presentations, either singly or in multi-image combination. These activities illustrate the one-way transmission of information from instructor to learners, often for a set period of time (generally a 40- to 50-minute class period). In small classes there may be some degree of two- way communication between teacher and learners, but most frequently, learners are passively listening and watching.

Strengths

The benefits of choosing a group presentation method to accomplish certain learning objectives include the following:

-A lecture format is familiar and conventionally acceptable to both instructor and learners. This method is the most common form of instructional delivery.

-Lectures can often be fairly quickly designed since the instructor is familiar wit the material and will make the actual presentation. The designer often works with the subject-matter expert to provide the instructor with a list of objectives and a topic outline with the unwritten agreement that the instructor will follow the outline. The assumption is that the instructor can make the necessary strategy decisions. This strength is a particular advantage when instruction is needed to address a critical, short-term need.

-A lecture places the instructor in direct control of the class and in a visible authority position. For some instructors and in many teaching contexts, these factors are advantageous for achieving the objectives.

-Large numbers of learners can be served at one time with a lecture. The group is limited only by the size of the room; thus, lectures can be highly economical.

-As instructional needs change, a presentation can be easily modified by deleting content or adding new content just before or even during the delivery. Also, the presentation can be easily adapted for a specific group of learners (e.g., made longer or shorter, more or less difficult).

-Lectures are a feasible method of communicating when the information requires frequent changes and updates or when the information is relevant for only a short time period, such as the implementation of a new travel policy.

-A good lecture can be motivating and interesting for students.

Limitations

The group presentation method of instruction suffers from the following limitations:

-Learning is typically very passive, involving listening, watching, and taking notes, with little or no opportunity for exchanging ideas with the instructor.

-To maintain learners' attention during a presentation, the lecturer needs to be interesting, enthusiastic, and challenging.

-When an instructor lectures, demonstrates, shows a video, or otherwise presents subject content to a class of learners, the assumption is made that all learners are acquiring the same understanding, with the same level of comprehension, at the same time. They are forced to learn at a
pace set by the teacher. Thus, lectures are not adaptive to individual differences.

-If questioning is permitted, instruction stops and all learners must wait until the question is answered before the presentation can proceed.

-In a large lecture class, it is difficult for the instructor to receive individual feedback from learners pertaining to misunderstandings and difficulties encountered during the presentation. Thus, some learners may leave the class with incorrect learning.

-A presentation may be inappropriate for teaching psychomotor and affective objectives, as these objectives typically require some form of practice or active learning environment.

-A large-group presentation may vary from presentation to presentation. Thus, the consistency of information and topics covered may not be the same for any two groups. This problem is particularly relevant when the training needs to be consistent, such as when teaching policies or
procedures.

-Students who have difficult with auditory learning will be at a disadvantage throughout the presentation.

Applications

There are specific situations and times at which a presentation to a group of learners is most valuable:

-As an introduction, overview, or orientation to a new topic

-To create interest for a subject or topic

-To present basic or essential information as common background before learners engage in small-group or individual activities

-To introduce recent developments in a field, especially when preparation is limited

-To provide such resources as a one-time guest speaker, a video, or other visual presentation that can most conveniently and efficiently be shown to the whole group at one time

-To provide opportunities for learners to make their own presentations as reports to the class

-As a review or summary when the study of the topic or unit is concluded

-To teach a large group of learners in a highly economical manner

Guidelines for Effective Lecturing

Keep in mind that learning is enhanced when learners are actively involved. Therefore, it is important to develop a plan for including learner participation activities when lecturing. Also, to facilitate learners' understanding of the material, lectures should be clear and well organized. We recommend the following components:

-Active interaction with the instructor. Prepare questions to use at various points during the verbal presentation; encourage or direct learners to answer and enter into discussion with the instructor. Decide on places to stop a presentation (often at the conclusion of a section or
the end of information presented on a concept), and ask questions to measure understanding and encourage discussion.

-Now taking. Encourage note taking by learners so that they will actively work with the material. Notes taken in the students' own words are useful in producing meaningful learning rather than rote memorization.

-Handouts. Consider preparing structured notes on topics requiring the leaner to (1) fill in an outline of content (e.g., structured notes), (2) complete diagrams that accompany visuals used in the presentation, (3) write replies to questions, (4) solve problems, and (5) make applications of content and concepts as the presentation proceeds. Learners can also complete self-check exercises or quizzes of the content presented. The key is to stimulate active processing of the information. For this reason, detailed notes are generally not recommended, since they eliminate the need for the students to generate their own. Other forms of handouts include slides from a multimedia presentation such as PowerPoint that allows you to print three slides per page with room for notes.

-Other mental activity. Encourage thinking by helping learners verbalize answers mentally to rhetorical or direct questions that you or another learner pose. You can also ask learners to formulate their own questions relating to the materials for use in follow-up, small-group sessions.

-Terminology. Use clear terminology and meaningful examples to illustrate concepts.

-Organization. Organize the lecture by constructing an outline. Bring the outline (or note cards) to the presentation and talk “from” it rather than reading it verbatim (a guaranteed painful experience for listeners). Unless you are very accomplished as a lecturer and highly familiar with the presentation, do not try to speak extemporaneously; a frequent result is a disorganized and rambling presentation.

-Enthusiasm. Show enthusiasm and interest in your subject.

-Format. A standard model (adapted from Slavin, 1994) is as follows:
1. Orient the students to the topics (an outline, story, or overview).
2. Review prerequisites.
3. Present the material in a clear, organized way.
4. Ask questions.
5. Provide independent practice.
6. Review and preview.

References

Slavin, R.E. (1994). Educational psychology (4th ed.). Needham Heights, MA: Allyn & Bacon.

ESTABLISHING GROUND RULES FOR GROUPS

Ground rules can be very useful indeed in group work contexts. The following suggestions include some of the issues and starting points from which groups can be encouraged to agree their own set of ground rules.

1 Create ownership of the ground rules. The various ground rules agendas suggested below should only be regarded as starting points for each group to adopt or adapt and prioritize. It is important that groups feel able to include ground rules which are appropriate for the particular people making up the group.

2 Foster a culture of honesty. Successful group work relies on truthfulness. Suggest that it is as dishonest for group members to 'put up with' something they don't agree about, or can't live with, as it is to speak untruthfully. However, it is worth reminding learners about the need to temper honesty with tact.

3 Remind group members that they don't have to like people to work with them. In group work, as in professional life, people work with the team they are in, and matters of personal conflict need to be managed so they don't get in the way of the progress of the group as a whole.

4 Affirm collective responsibility. Once issues have been aired, and group decisions have been made as fully as possible, they convention of collective responsibility needs to be applied for successful group processes. This leads towards everyone living with group decisions and refraining from articulating their own personal reservations outside the group.

5 Highlight the importance of developing and practising listening skills. Every voice deserves to be heard, even if people don't initially agree with the point of view being expressed.

6 Spotlight the need for full participation. Group work relies on multiple perspectives. Encourage group members not to hold back from putting forward their view. Group members also need to be encouraged to value the opinion of others as well as their own.

7 Everyone needs to take a fair share of the group work. This does not mean that everyone has to do the same thing. It is best when the members of the group have agreed how the tasks will be allocated amongst themselves. Group members also need to be prepared to contribute by building on the ideas of others and validating each other's experiences.

8 Working to strengths can benefit groups. The work of a group can be achieved efficiently when tasks are allocated according to the experience and expertise of each member of the group.

9 Group should not always work to strengths, however! Activities in groups can be developmental in purpose, so task allocation may be an ideal opportunity to allow group members to build on areas of weakness or inexperience.

10 Help group members to see the importance of keeping good records. There needs to be an output to look back upon. This can take the form of planning notes, minutes or other kinds of evidence of the progress of the work of the group. Rotate the responsibility for summing up the position of the group regarding the tasks in hand and recording this.

11 Group deadlines are sacrosanct. The principle, 'You can let yourself down, but it's not OK to let the group down' underpins successful group work.

12 Cultivate philanthropy. Group work sometimes requires people to make personal needs and wishes subordinate to the goal of the group. This is all the more valuable when other group members recognize that this is happening.

13 Help people to value creativity and off-the-wall ideas. Don't allow these to be quelled out of a desire to keep the group on task, and strike a fair balance between progress and creativity.

14 Enable systematic working patterns. Establishing a regular programme of meetings, task report backs and task allocation is likely to lead to effective and productive group performance.

15 Cultivate the idea of group rules as a continuing agenda. It can be productive to review and renegotiate the ground rules from time to time, creating new ones as solutions to unanticipated problems that might have arisen. It is important, however, not to forget or abandon those ground rules that proved useful in practice, but which were not consciously applied.

INTEGRATING TEAM EXERCISES WITH OTHER COURSE WORK

Ruth Federman Stein
Sandra Hurd
pp. 13-16

For the most part, college-level instruction is not now organized around the principles of cooperative learning. Assignments, textbooks, the examination system, and even the physical arrangements of many large classrooms reflect a more individualistic conception of learning. Under these conditions, how are principles of cooperative learning to be introduced without the appearance of inconsistency?

Instructors who initiate team projects often point out that team activities increase learning. They note that teamwork is widespread in industry and other organizations. Justification along these lines, however, may fail to motivate students because they say little about how teams actually achieve the benefits that are claimed on their behalf, and how a team project complements the content and organization of the specific course in which it is being introduced. This section suggests some ways to supplement the conventional justification for them.

The suggestions are arranged under two headings: rationales for the use of teams in a course or discipline, and the integration of team exercises with other course content. You will note, however, that these categories may overlap in practice.

RATIONALES FOR TEAMS

The following rationales address team exercises as a form of cooperative learning and are thus potentially applicable to a wide range of activities

Constructivist rationale. Most psychological theory portrays learning as a process of construction (Fosnot, 1996). Students can make sense of a concept only if they build it into the structure of their own prior experience. It is very difficult to create such a structure by oneself, especially in an unfamiliar subject area. Discussion in small groups of peers makes this undertaking much easier.

Linguistic perspective on learning. Scholars of professional language and rhetoric, such as Charles Bazerman (1998, 1991) and James Boyd White (1995), note that when students encounter a discipline or a professional filed, they are being exposed to a specialized language. In learning concepts and terms, they are learning to engage in a particular form of discussion. Their grasp of a topic is usually evaluated on the basis of their ability to understand questions about it and to write cogent answers. Students are much more likely to develop this linguistic proficiency if they have both informal and formal opportunities to speak, rather than being restricted to listening and reading.

Tacit dimension of professional and disciplinary knowledge. As Donald Schon has pointed out (19983, 1987), there are many forms of learning that cannot be characterized in terms of propositional knowledge, and thus are not reducible to statements in a textbook or lecture. Practical skills, intuitive judgement, and social context cannot generally be taught by exposition. Some sort of collaborative activity is required. Thus, for example, in a team exercise in a marketing course, students would get a chance to act out the role of a marketing specialist and discover some of the practical exigencies and constraints of the practice of marketing. This background understanding of the social context of marketing would provide a framework within which students may subsequently organize more detailed information of pricing strategy, promotional techniques, and problems of distribution.

Habits and attitudes needed for academic achievements. As Kenneth Bruffee (1999) has pointed out, higher education can be thought of as a form of acculturation. According to this model, becoming successful as a student is a cultural acquisition. Academic competence is not just mastering course content: It also involves the formation of attitudes about schoolwork and the acquisition of habits of regular class attendance, consistent and thorough preparation, and disciplined management of time. Interaction with peers in a classroom can help students learn habits and attitudes needed for academic success more easily. This interaction can be especially helpful for students who come to the United States from other cultures.

Strategies for Integrating Team Exercises

Team exercises provide instructors with feedback mechanisms of unparalleled sensitivity. If teams had no other benefits, they would

be justifiable solely on the grounds that they provide detailed information about the success of instruction and bring to light areas of misunderstanding. The following strategies are designed both to take advantage of that feedback and to emphasize its importance to students.

Anticipatory strategies. Formal instruction can be designed to anticipate team exercises. For example, a lecture might introduce a problem or a question and review some of the information that could be brought to bear on it. The question or problem could then be posed to teams, who would review their notes and come up with an answer or solution. Alternatively, a lecture could introduce a series of related concepts, and specialized terms and teams convened to explain them and provide illustrations.

Involvement and attention. It is essential that the instructor not be aloof from team exercises. Circulating among the groups, listening, asking questions, and evaluating students' understanding both of concepts and tasks will all help to provide a clearer sense of the students' progress and will also steer them back to the task at hand if they should be inclined to stray from it. The instructor's active attention will emphasize to the students the importance of the team exercise and its connection to other parts of the course.

Short-term adaptation. Information gleaned from the teams can be incorporated into formal lessons. At the start of the next lecture, briefly summarize progress observed in teams, correct specific misconceptions, or highlight unresolved questions that have been raised in the teams.

Longer-term follow-up activities. Subsequent lectures, discussions, and
assignments can be designed to build on the team activities. Teams can report their conclusions in general discussion, a question related to the team activities. Teams can report their conclusions in general discussion, a question related to the team activity could be included on
the exam, readings related to questions raised by the teams could be assigned.

PEER INSTRUCTION

Introduction to Physics at Harvard University
Professor: Eric Mazur

In 1989, I read an article in the American Journal of Physics that contained a test to assess understanding of Newtonian mechanics. I gave the test to my students at Harvard and was shocked by the results - the students had merely memorized equations and problem-solving procedures and were unable to answer the basic questions, indicating a substantial lack of understanding of the material. I began to rethink how I was teaching and realized that students were deriving little benefit from my lectures even though they generally gave me high marks as a lecturer. So I decided to stop preaching and instead of teaching by telling, I switched to teaching by questioning using a teaching technique I have named "peer instruction."

My students now read the material before class. To get them to do the reading, I begin each class with a short reading quiz. The lecture periods are then broken down into a series of digestible snippets of 10 to 15 minutes. Rather than regurgitating the text, I concentrate on the basic concepts and every 10 or 15 minutes I project a "Concept Test" on the screen. These short conceptual questions generally require qualitative rather than quantitative answers. The students get one minute to think and choose an answer. They are also expected to record their confidence in their answer. After they record their answers, I ask their students to turn to their neighbors and to convince them of their logic. Chaos erupts as students engage in lively and usually uninhibited discussion of the question. I run up and down the aisles to participate in some of the discussions - to find out how students explain the correct answer in their own words and to find out what mistakes they make.

After one or two minutes, I call time and ask students to record a revised answer and a revised confidence level. A show of hands then quickly reveals the percentage of correct answers. After the discussion, the number of correct answers and the confidence level typically rise dramatically. If I am not satisfied, I repeat the cycle with another question on the same subject. When the results indicate a mastery of the concept, I move on to the next subject.

I have been lecturing like this now for more than four years. During this time the students have taught me how best to teach them. As for the students, nothing clarifies their ideas as much as explaining them to others. As one student said in a recent interview,: "There is this ah-hah! Kind of feeling. It's not that someone just told me; I actually figured it out. And because I can figure it out now, that means I can figure it out on the exam. And I can figure it out for the rest of my life."

YES VIRGINIA THERE IS A BIG DIFFERENCE BETWEEN COOPERATIVE AND COLLABORATIVE LEARNING PARADIGMS

By Dr. Theodore Panitz
Cape Cod Community College

Author's note:

The following serves as an introduction for a longer more detailed comparison of the student centered learning paradigms, cooperative and collaborative learning. Over the years I have received many questions about the differences between these two paradigms. I have scoured the literature and extracted viewpoints from many of the key people who use and research these teaching/learning paradigms. In addition I have tried to present my interpretation based upon my own experiences in the classroom.

Collaborative learning will be defined by comparing it's characteristics to those of cooperative learning paradigms. Each paradigm represents one end of a spectrum of teaching-learning which ranges from being highly structured by the teacher (cooperative) to one which places the responsibility for learning primarily with the student (collaborative).

The underlying premise for both collaborative and cooperative learning is founded in constructivist epistemology. Knowledge is discovered by students and transformed into concepts students can relate to. It is then reconstructed and expanded through new learning experiences. Learning consists of active participation by the student versus passive acceptance of information presented by an expert lecturer. Learning comes about through transactions among students and between faculty and students, in a social setting, as they construct a knowledge base.

Ken Bruffee (1995 "Sharing our toys- Cooperative learning versus collaborative learning". Change, Jan/Feb, 1995 pp12-18) identifies two causes for the differences between the two approaches. He states: "First, collaborative and cooperative learning were developed originally for educating people of different ages, experience and levels of mastery of the craft of interdependence. Second, when using one method or the other method, teachers tend to make different assumptions about the nature and authority of knowledge. The age or education levels as a distinction have become blurred over time as practitioners at all levels mix the two approaches. However, what determines which approach is used does depend upon the sophistication level of the students involved, with

collaborative requiring more
advanced student preparation working in groups." (p12)

Brufee sees education as a reacculturation process through constructive conversation. Students learn about the culture of the society they wish to join by developing the appropriate vocabulary of that society and by exploring that society's culture and norms (i.e. that of mathematician, historian, journalist, etc.). He identifies two types of knowledge as a basis for choosing an
approach. Foundational knowledge is the basic knowledge represented by socially justified beliefs we all agree on. Correct spelling and grammar, mathematics procedures, history facts, a knowledge of the contents of the constitution, etc., would represent types of foundational knowledge. these are best learned using cooperative learning structures in the early grades

Nonfoundational knowledge is derived through reasoning and questioning versus rote memory. The other way in which nonfoundational education differs from foundational is that it encourages students not to take their teacher's authority for granted. Students should doubt answers and methods for arriving at answers provided by their professors, and perhaps more importantly they need to be helped to come to terms with their doubts by participating actively in the learning and inquiry process. Out of this process knew knowledge is often created, something not likely to occur when dealing with the facts and information associated with foundational knowledge. Collaborative learning shifts the responsibility for learning away from the teacher as expert to the student, and perhaps teacher, as learner.

Brufee sees the two approaches as linear with collaborative learning being designed to pick up where cooperative learning leaves off. In effect, students learn basic information and processes for interacting socially in the primary grades and then extend their critical thinking and reasoning skills and
understanding of social interactions as they become more involved and take control of the learning process through collaborative activities. This transition may be viewed as a continuum from a closely controlled, teacher-centered system to a student-centered system where the teacher and students share authority and control of learning.

The following definitions for collaboration and cooperation form the basis for their teaching paradigms.

Collaboration is a philosophy of interaction where individuals are responsible for their actions, including learning and respect the abilities and contributions of their peers. Collaborative learning is a

personal philosophy, not just a classroom technique. In all situations where people come together in groups, it suggests a way of dealing with people which respects and highlights individual group members' abilities and contributions. There is a sharing of authority and acceptance of responsibility among group members for the groups actions. The underlying premise of collaborative learning is based upon consensus building through cooperation by group members. (T. Panitz , (1997), "Collaborative Versus Cooperative Learning: Comparing the Two Definitions Helps Understand the nature of Interactive learning" Cooperative Learning and College Teaching, V8, No. 2, Winter 1997, Panitz, T., and Panitz, P., (1998) "Encouraging the Use of Collaborative Learning in Higher Education." In J.J. Forest (ed.) Issues Facing International Education, June, 1998, NY, NY: Garland Publishing

Cooperation is a structure of interaction designed to facilitate the accomplishment of a specific end product or goal through people working together in groups. Cooperative learning is defined by a set of processes which help people interact together in order to accomplish a specific goal or develop an end product which is usually content specific. It is more directive than a collaborative system of governance and closely controlled by the teacher. While there are many mechanisms for group analysis and introspection the fundamental approach is teacher centered whereas collaborative learning is student centered. (Panitz 1997, 1998) Spencer Kagan (1989, Educational Leadership (Dec/Jan 1989/1990)) defines cooperative learning: "The structural approach to cooperative learning is based on the creation, analysis and systematic application of structures, or content-free ways of organizing social interaction in the classroom. Structures usually involve a series of steps, with proscribed behavior at each step. An important cornerstone of the approach is the distinction between "structures" and "activities". To illustrate, teachers can design many excellent cooperative activities, such as making a team mural or a quilt. Such activities almost always have a specific content-bound objective and thus cannot be used to deliver a range of academic content. Structures may be used repeatedly with almost any subject matter, at a wide range of grade levels and at various points in a lesson plan."

Johnson, Johnson, and Smith (1998, Johnson, D.W., Johnson, R.T., Smith, K.A., Change, July/August) clarify theories which govern cooperative learning strategies. "Social interdependence theory assumes that cooperative efforts are based on intrinsic motivation generated by interpersonal factors and a joint aspiration to achieve a significant goal. Behavioral learning theory assumes that cooperative efforts are powered by extrinsic motivation to
achieve rewards. Social interdependence theory focuses on relational

concepts dealing with what happens among individuals, whereas the cognitive-development perspective focuses on what happens within a single person (p29).

Many of the elements of cooperative learning may be used in collaborative situations. For example students work in pairs together in a Think-Pair-Share procedure, where students consider a question individually, discuss their ideas with another student to form a consensus answer, and then share their results with the entire class. In the Jig Saw method (Aronson, E., Blaney, N., Stephan, C., Sikes, J., Snapp, M. (1978) The Jigsaw Classroom, Beverly Hills, CA: Sage Publication), students become "experts" on a concept and are responsible for teaching it to the other group members. Slavin (1978, "Student Teams Achievement Divisions", Journal of Research and Development in Education, 12 (June), pp39-49) developed Student Teams-Achievement-Divisions where the teacher presents a lesson, then the students meet in teams of four or five members to complete a set of worksheets on the lesson. Each student then takes a quiz on the material. Bonus points are given to the team if any member's score improves according to a preset criteria. The highest scoring teams are recognized in a weekly class newsletter.

GAME THEORY

The class will form into four teams of two, with two superobservers. The game will consist of alternating periods of discussion, voting, and resolution.

During discussion, teams can decide on a voting strategy among themselves, and they can negotiate with other teams for coordinated voting. The only constraints on negotiation is that resource exchanges must be recorded with a superobserver.

During voting, each team will cast one of their three possible voting options.

During resolution, the four votes will be combined to determine a ***group outcome*** from the outcome table.

The superobservers will collect information on the processes and strategies of each team and keep records.

Each team is different. The initial assets (expressed in units), and the voting choices of each team are below.

Teams	**A**	**B**	**C**	**D**
Initial assets:	2	5	10	20
Voting options:	{V, 0, 1}	{V, 0, 2}	{0, 1, 2}	{0, 2, S}

V	can be interpreted as Veto
{0, 1, 2}	can be interpreted as a strength of monetary support
S	can be interpreted as Strong monetary support

Under these interpretations,

Team A	poor
Team B	workers
Team C	professionals
Team D	wealthy

The game is to increase the wealth of each team.

The outcome is each game round is expressed by a decision table:

TEAM	A	B	C	D
OUTCOMES				
VVS	x2	x1.5	x.7	x.2
VV	x1	x1	x.6	x.5
VS	if V then x2 else x1		x.9	x.7
V0123	if V then -5 else +5		+ 5	0
V456	if V then +10 else +2		- 5	+ 5
012	0	+ 1	+ 5	+10
34	+4	+ 6	+10	+ 8
567	+6	+10	+12	+10
S012	+5	+ 6	+ 7	+ 5
S345	x2	x3	x4	x5

Some outcomes are triggered by several different voting results. For example, a vote sum of 4, 5, or 6 all trigger the **V456** outcome row, even if one team voted V.

"+" means add the specified amount to the team assets

"-" means subtract the specified amount to the team assets

"x" means multiply the current assets of the team by the specified factor.
note that multiplication by less than 1 is a loss of assets.

The *expected gain* for each team for each round is 5.

THE THIRSTY ARCHEOLOGIST

An archeologist was digging in a Paleozoic mudflat. She came across an imprint of a raindrop that fell 400 million years ago. The sun was hot, and she took a drink from her canteen. How many molecules of the original Paleozoic raindrop did she drink?

You will have fifteen minutes to generate an answer. Write down all assumptions, choices, and decisions you make. No justifications are needed.

THE TELEPHONE DRAWING

Describe the figure below so that a person on the other end of a phone line can draw it.

Record your decisions about accuracy.

CONSENSUS

Each class member has contributed $5 to form a relatively large cash reserve. Your task is to *give* the resource to one and only one member of the class.

The purpose of this exercise is to observe the process of consensual decision making.

Rules

1. The decision of who gets the resource must be a *consensus*. Class members must unanimously agree on a single recipient.

2. No explicit or implicit agreements to divide the resource (now or later) are permitted. The gift is to have no strings attached.

3. The decision must be made before class ends today.

4. Violation of the above rules will result in a class failure for this exercise.

TEACHING FOR INNOVATION

TOPIC 9. CHALLENGING THE STUDENTS

Contests Motivate Top Students in Large Classes

Teaching Examples (Bricken):

Foundations: Chapter 0 and responses

DS&A: Versions of Factorial

Management: Measurement

Management: Critical Incidents

Formal: Formal Cube, Algebraic Specification

AI: Streams with Delayed Evaluation

AI: Knowledge Engineering

Ethics: Six Dilemmas

Applications to Teaching Mathematics (Bricken):

Foundations: Timeline

Foundations: Functions

Formal: Combinatorial Circuit Minimization

Spatial Math

CONTESTS MOTIVATE TOP STUDENTS IN LARGE CLASSES

Eric Roberts tackles neglected issue of motivating elite students through recognition, camaraderie, teaching opportunities.

Roberts provides ample opportunities for his students to earn extra-credit through contests that usually entice about 10 percent of a given class to enter. Work is judged for categories including "best algorithm" and "best aesthetics." Past winners have programmed adventure games and created computer animations.

Eric Roberts is the principal architect of what was for many years the largest course at Stanford-Computer Science 106A, an introductory programming class with an enrollment that waxes and wanes with the NASDAQ. In a fat year, 1,000 students may enroll, with more than 400 students in a single class. How professors can encourage top scholars in large classes was the topic of Roberts' "Award-Winning Teachers on Teaching" talk Nov. 18 in Building 460.

"How do you manage in a large course not completely snowing the people on the low end of the scale or boring silly the people on the high end of the scale?" asked Roberts, the John A. and Cynthia Fry Gunn University Fellow in Undergraduate Education.

Only 6 percent of Roberts' introductory students end up majoring in computer science. Many major in other engineering disciplines, and almost 20 percent major in social sciences or humanities. "It's a wide spectrum, and I wanted to encourage those people to be in that class," Roberts said. "It was one of the signature aspects of the Stanford curriculum that we tried to keep an introductory computer science sequence that really did have broad appeal."

Success in such classes means supporting students every step of the way so they can do well, Roberts said. It also means setting a high bar. "One of the difficulties in a large class is if you decide that you're going to curve it rigidly, you've forced it into a mode where many people are going to be unhappy," he said. Roberts favors a grading system that rewards those who meet clearly delineated objectives-no matter how many students meet those objectives. "If everybody does enough work to get an A, then everybody will in fact get an A."

That's a big incentive to do well. Further, superlative work in Roberts' classes has earned A+ and even A++ marks. (The latter designates work that "exceeds all expectations," according to a jury of section leaders, teaching assistants and the professor.)

Top students also have the opportunity to gain teaching experience after the class ends. Owing to economic necessity and a dearth of graduate students

willing to assist teaching introductory computer courses, the teaching assistants (TAs) in such courses at most universities are undergraduates.

"We decided at Stanford to make a virtue of necessity and really train those students to be wonderful as teachers," Roberts said. Undergraduate TAs also provide "stepping stone role models" that help increase the number of underrepresented minorities in computer science.

Programming is one of the most varied intellectual activities in terms of productivity and ability, Roberts said. "The difference between this person who's sort of good and that person who's really great is extraordinary."

A 1968 study of working programmers showed 20 to 1 variations in productivity-how much code a person could generate-among individuals with the same levels of education and experience. The best programmers also tended to be the fastest and to have the fewest bugs. "You see the same [enormous variability] in classrooms," Roberts said.

Move over, Ed McMahonâ¤©

Roberts said he can't gear the class to top students without risking losing those on the bottom. Instead, he leverages the features of large classes to encourage excellence. And that means-drumroll, please-lots of contests that provide extra credit. Students can enter three contests per quarter in his CS 106 A and B classes. Only large classes have enough students to make contests feasible, as only about 10 percent of the class usually enters. A large class may have 40 to 60 entrants. Winners rise from the obscurity of a large class.

Judged by an army of section leaders, programs can win for such categories as "best algorithm" or "best aesthetics." Some contests have explored the limited world of Karel the robot, who can turn left but not right, forcing students to program three left turns to make a right when Karel runs a maze. One winning program calculated the weakest and strongest countries in a game of Risk to determine who could attack whom. Another winner created an animation of IBM's Deep Blue supercomputer beating chess champion Garry Kasparov in 1997. Yet another winner created an adventure game using text-based commands to have players find magic wands and potions.

Prizes provide recognition (certificates for first-place, runner-up and honorable mention contestants, and classroom presentations of winning entries), camaraderie (a group dinner for winners at Roberts' house the following quarter), perks (offers of letters of recommendation from Roberts) and glory (an automatic 100 percent score on the final exam for the first-prize winner of any contest).

The contests have opened up new worlds for some students. When a former English major in Roberts' Computer Science 105 course won a contest, she became so fired up that she got an undergraduate degree in symbolic systems and a master's degree in computer science. She went on to become employee number nine at Google.

Roberts' teaching awards include the Bing Fellow Award for Excellence in Teaching (1993), the Perin Award for Undergraduate Engineering Education (1995), the Lloyd W. Dinkelspiel Award (1998), the John A. and Cynthia Fry Gunn University Fellowship in Undergraduate Education (2002), the Association for Computing Machinery's Special Interest Group on Computer Science Education Award (2003) and the Laurance and Naomi Carpenter Hoagland Prize (2004).

The Center for Teaching and Learning sponsors the "Award-Winning Teachers on Teaching" lecture series. Deborah Gordon, a professor of biological sciences, will deliver the next talk Jan. 20 at noon in the Hartley Conference Center of the Mitchell Earth Sciences Building.

Chapter 0 Responses

Almost all mathematical textbooks begin their story in the middle, as if you already knew the assumptions, critical choices, and relevance of each topic. For example, your text begins to teach about proof theory on page 1.

The following questions come prior to studying the content of mathematics. Answer each question in *one sentence or less*. These questions identify what you believe about mathematics, therefore there are no right or wrong answers, just answers that are less or more thoughtful.

HAND IN YOUR ANSWERS AT THE BEGINNING OF CLASS.

Science

1. Which of the following *physically exist*?	*EXIST*	*NOT*
a. electrons	15	2
b. the temperature of the center of the Sun	6	11
c. the cosmological big bang	3	14
d. the state of the internet	3	14

2. Is *length* an objective concept;
   that is, is the length of an object independent of a particular observer?

   OBJECTIVE: 8 *NOT: 9*

3. Can you know something about the world without changing the world?

   YES: 7 *NO: 10*

Mathematics

4. Are mathematical ideas invented, discovered, or something else?

   INVENTED: 2 *DISCOVERED: 8* *OTHER: 7*

5. What is mathematical Truth?

Universally true, follows the rules, anything is possible, logically/deductively true, consistency of results, laws that can't be disproved, reality in numbers, true or false, validity

6. In what sense does π exist? Where might it exist?
   Is π a constant? Is it eternal or might it change over time?

EXISTS	*yes: 9*	*no: 1*	*as a concept: 3*
CONSTANT	*yes: 14*	*no: 0*	
ETERNAL	*yes: 10*	*no: 2*	

7. Are there any mathematical concepts that have only one property (i.e., are there pure concepts independent of other concepts)?

YES: 9 (zero, void, axioms) NO: 8

8. When is the following equation True?: 7 + 8 = 3

watch-time, mod12, based on false premise, mod10, = means >, change the definition of the symbols
NEVER: 4

9. Is every number either even or odd? How do you know?

YES: 11 not the other, every integer, by generalization, theory of divisors, years of learning, whole numbers, by induction
NO: 5 zero, infinity-1, Pi, fractions, irrationals,

10. Prove the *Pigeonhole Principle*, that N+1 pigeons cannot fit into N holes without sharing.

By contradiction: assume N+1 pigeons can fit into N holes. There are N holes, therefore N pigeons, (one-to-one correspondence). Thus N+1 = N contradiction. This is an example of something that cannot be proved by an algorithm.

11. Imagine two points as close together as possible. Is there another point in between them? Are there an infinite number of points in between them?

YES, one and infinite: 13 NO: 4 (finite geometry)

12. Is the following statement True or False or something else?:
Somewhere in the decimal expansion of π, there are exactly 34 sevens in a row.

TRUE: 3 non-repeating
FALSE: 9 non-terminating algorithm, non-repeating
OTHER: 5 can't determine

13. What kinds of mathematics can be beautiful?

subjective choice, any kind, all, elegant and abstract, following rules, Mandelbrot set, language of science, gives explanations, analytical, fractal, integral calculus, simple clear and correct.

Computation

14. What is the shortest program which will produce a *random number*?

NONE: 7
EGS: sampling noise, pop a number 1, pick a number, non-trivial algorithm, rand() %n

15. Is a *bit* the simplest computational object?

YES: 15 NO: 2

16. Binary computation uses two states, 0 and 1. Is *unary* computation possible?

YES: 11 (stroke arithmetic, change sensitive systems) NO: 6

17. Name three mathematical concepts which cannot be computed.
Pi , infinity, i, point, circle, irrational numbers, void, chaos theory, empty, set, space, division by zero, black holes, real numbers, trisecting an angle, doubling a cube, squaring a circle, 1/3, set of natural numbers.

Attitude

18. How many of the above questions have you thought about before today?

0: 8 1-2: 4 3: 3 >3: 2

19. Look at the Chapter 0 quotes which follow. Make three lists of names:
people that you agree with,
people that you disagree with, and
people that you do not understand well enough to have an opinion about.

	AGREE	*DISAGREE*	*HUH?*
William of Occam	*7*	*2*	*5*
John Locke	*12*	*2*	*0*
G. Spencer-Brown	*8*	*8*	*0*
A. Einstein	*12*	*0*	*4*
V.I. Arnold	*4*	*5*	*5*
A. Eddington	*2*	*7*	*7*
S.G. Shankar	*4*	*3*	*8*
H. Weyl	*13*	*2*	*1*
G. Rota	*9*	*4*	*3*
I. Lakatos	*9*	*3*	*3*
R. Feynman	*8*	*3*	*3*
D. Knuth	*8*	*8*	*1*
T. Norretrandres	*10*	*1*	*4*

20. *Essay:* Write less than one page in response to this famous question:

Why is mathematics so unreasonably effective in describing and predicting reality?

Math concepts come from reality.

From years of discovery and quantification, well defined rules.
Objective and precise, can prove anything that is true, builds upon itself.
Simple, clear and correct. reasonable, applicable to real-life.
Build upon known principles, symbols and abstraction, what leads to wisdom?
The way to understand the Universe, sea of interrelated details, successful prediction, laws.
Makes a lot of assumptions and not precise, rules of the way reality works.
Humans avoid analysis and reasoning, reality comes through visionary senses.
An invention of human minds, doesn't apply to feelings, may not apply outside our experience.
Determines patterns, simplicity of concepts, describes what we perceive.
Math truncates information, simulates world, based on probability.
Explains natural mechanisms, prediction helps survival, identifies structural stabilities.
Ineffective, misses non-linear relations and rapid changes and irregular shapes.
Encompasses objective information, misses intangibles and complexities, simplified view.
Removes emotion, rigid rule sets, scientific beauty (simplicity, harmony, brilliance).

Chapter 0 Quotes

Attitude

"Multiplicity ought not be posited without necessity."
-- William of Occam (1340)

"The acts of the mind, wherein it exerts its power over simple ideas, are chiefly these three:
1) Combining several simple ideas into one compound one, and thus all complex ideas are made.
2) The second is bringing two ideas, whether simple or complex, together, and setting them by one another so as to take a view of them at once, without uniting them into one, by which it gets all its ideas of relations.
3) The third is separating them from all other ideas that accompany them in their real existence; this is called abstraction, and thus all its general ideas are made."
-- John Locke (1690)

"The language of all art forms, such as cookery, drawing, programming, research, mathematics, and music, is a set of instructions which, if followed, will lead the reader to the same ecstacies as those experienced by the original artist."
-- G. Spencer-Brown

Science

"As far as the propositions of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality."
-- A. Einstein

"The axiomatization and algebraization of mathematics, after more than fifty years, has led to the illegibility of such a large number of mathematical texts that the threat of complete loss of contact with physics and the natural sciences has been realized."
-- V. I. Arnold

"I believe there are exactly
15,747,724,136,275,002,577,605,653,961,181,555,468,044,717,914,527,116,709,366,231,425,076,185,631,031,296 protons in the universe, and the same number of electrons."
-- Sir Arthur Eddington

Mathematics

"Mathematics has always skirted dangerously close to the shores of metaphysics."
-- S. G. Shanker

"We now come to the decisive step of mathematical abstraction: we forget about what the symbols stand for. ... There are many operations which [the mathematician] may carry out with these symbols, without ever having to look at the things they stand for."
-- H. Weyl

"Of all escapes from reality, mathematics is the most successful ever. It is a fantasy that becomes all the more addictive because it works back to improve the same reality we are trying to evade. All other escapes -- sex, drugs, hobbies, whatever -- are ephemeral by comparison."
-- G.C. Rota

"Mathematics, this product of human activity, 'alienates itself' from the human activity which has been producing it. It becomes a living, growing organism, that *acquires a certain autonomy* from the activity which has produced it; it develops its own autonomous laws of growth, its own dialect."
-- I. Lakatos

Computation

"Computer science also differs from physics in that it is not actually a science. It does not study natural objects. Neither is it, as you might think, mathematics."
-- R. Feynman

"There is no need for infinities; there are quite enough finite numbers to serve any purpose."
-- D. Knuth

"Calculation is a method of getting rid of information in which you are not interested. You throw away what is not relevant."
-- T. Norretranders

Versions of Factorial

Focal concepts:

Each of these encodings of the *factorial function* is functionally equivalent. How they achieve the functionality differs.

Almost all are legitimate Mathematica code. Since the core process in Mma is the same for each encoding, we have a demonstration that all are *statically equivalent*. Dynamically, ie how the code runs, all are different.

The *style of encoding* should match as closely as possible the form of the natural problem. Second, the style should match the coder's natural way of thinking about the problem.

Types of *dynamic differences* include:

- ***Syntactic sugar***: the same dynamic behavior (ie the same language). Macros expand the sugared notation *at read-time* into standard notation. Eg:

```
(a + b)  ==>  +[a,b]

declare a=5;  (a + b)
```

- ***Functional syntactic sugar***: shorter and specialized versions of functions. The compiler usually standardizes these variants. Eg, all of the various loop constructs are the same.

```
for i=1 to n do Process[i]

i:=0; (do Process[i]; i:=i+1 until i=n)

dotimes[n, Process[#]]

StreamProcess[IntegerStream[1, n]]
```

- ***Functional model difference***: different processes for achieving the same functional objective. Most of these compile into different machine instructions, but a good optimizing compiler might standardize some of them. Eg: iteration vs recursion vs mapping

```
do[i from 1 to n, acc from nil, Process[i, acc]]

(if i=n, acc, Process[i-1, F[acc, i]])

(if i=n, 0, F[i, Process[i-1]])

map[Process, {1,i,n}]
```

- ***Operational difference***: different engines achieve the same objective but use different operational characteristics. Eg:

```
F[1]=1; F[n]= G[n, F[n-1]]

(if test[n] then (res:=F[i], ++i) else res)

(send F, n)
```

- ***Mathematical difference***: different mathematical computations achieve the same objective but use different models. Eg:

```
F[n] = G[n]             eg Fac[n]=Gamma[n+1]

Decode[Process[Encode[F,n]]]

When (F[Guess[n1] - F[Guess[n2]] = <small>), F[n1]
```

- ***Level of Implementation difference***: different processes occur at different levels of abstraction. Eg:

```
2 + 5 = 7

010 + 101 = 111

r1=Load[i0]; r2=Fetch[j0]; r3=Add[r1,r2]; Store[r3]

b0 = xor[i0,j0]; b[1] = xor[i1,j1]
```

VERSIONS

```
1.  proceduralFactorial[n] :=
      if ( Integer[n] and Positive[n] )
            then
                  Block[ {iterator = n,
                          result = 1  },
                    While[ iterator != 1,
                        result := result * iterator;
                        iterator := iterator - 1 ];
                    return result]
            else  Error


2.  sugaredProceduralFactorial[n] :=
      Block[  {result = 1},
          Do[  result = result * i, {i, 1, n} ];
          result]
```

```
3.  loopFactorial[n] :=
      { For[ i=1 to n, i++, result := i*result ];
        result }


4.  guardedFactorial[n, result] :=
      Precondition:     Integer[n] and Positive[n]        /also end condition
      Invariant:        factorial[n] = n * factorial[n - 1]
      Body:             guardedFactorial[ (n - 1), (n * result) ]
      PostCondition:    result = Integer[result] and Positive[result]
                                 and (result >= n)


5.  assignmentFactorial[n] :=
      { product := 1;
        counter := 1;
        return assignmentFactorialCall[n, product, counter] }


6.  assignmentFactorialCall[n, product, counter] :=
      if[ (counter > n)
            then
                  return product
            else
                  { product := (counter * product);       /error if these are
                    counter := (counter + 1);             /in reverse order
                    return assignmentFactorialCall[n, product, counter] } ]


7.  recursiveFactorial[n] :=
      if[  n == 1, 1,  n*recursiveFactorial[n - 1] ]


8.  rulebasedFactorial[1] = 1;
    rulebasedFactorial[n] := n * rulebasedFactorial[n - 1]


9.  accumulatingFactorial[n, result] :=
      if[ (n = 0)
            then
                  return result
            else
                  return accumulatingFactorial[ (n - 1), (n * result) ]


10. upwardAccumulatingFactorial[product counter max] :=
      if[ (counter > max)
            then
                  return product
            else
                  return upwardAccumulatingFactorial[ (counter * product)
                                                      (counter + 1)
                                                      max ] ]
```

```
11. mathematicalFactorial[n] =
      Apply[ Times, Range[n] ]


12. generatorFactorial[n]
      Times[ i, Generator[i, 1, n] ]


13. combinatorFactorial :=
      Y f< n< COND (=0 n) 1 (* n (f (-1 n))) >>


14. sugaredCombinatorFactorial =
      S (CP COND =0 1) (S * (B FAC -1)))


15. integralFactorial[n] =  Gamma[ n + 1 ]  :=
      integral[ 0 to Infinity, (t^n * e^(1 - n)), dt ]


16. streamOfFactorials  =
    streamAttach[ 1 streamTimes[streamOfFactorials streamOfPositiveIntegers]
]
streamOfPositiveIntegers =
    streamAttach[ 1 streamBuild[ Add1 CurrentStreamValue ] ]


17. JamesCalculusFactorial[n] =
      Decode[Standardize[Do[Stack[Encode[i], acc] {i,1,n}]]]

      Stack[jf, acc] =
            Subst[jf UnitToken acc]
```

From Abelson and Sussman, *Structure and Interpretation of Computer Programs*

```
18. abstractMachineFactorial =  <p385>

19. registerMachineFactorial =  <p511>

20. compiledFactorial = <p596-7>
```

MEASUREMENT

Table Exercise

Measure the height of a table. Observe the techniques and the sources of variation.

Richardson Exercise

Measure the perimeter of your hand using three indivisible units of measurement (say 6 inches, 1 inch, and 1/10 inch). Observe and explain the relationship between the result of the measurement and the arbitrary choice of measurement unit.

Measurement Types Exercise

Consider the definitions of these seven types of measure. Each adds a new constraint to the previous type.

Indicative:	existence
Nominal:	set membership
Ordinal:	ordering relation
Interval:	composition relation maps onto addition
Ratio:	meaningful zero
Real:	continuity
Imaginary:	complex structure of a number

Which types of measurement can be used on common things like: people in a room, light in a room, hairs on your head, hunger, thoughts, ...

CRITICAL INCIDENTS

As you read the text (Chs 1-5), write down the location of sections which make you think. These critical incidents may be confusions, insights, daydreams, strong connections, disagreements, surprises, etc.

ALGEBRAIC SPECIFICATION EXAMPLES, CUBE

```
GenericCube = {

      ;the structure of the SPACE embodying cubeness

      USE[ UnitVector ] = { i j k }
      V = { 0 - 1 }
      D = { 0 1 }
      < vi_V, vj_V, vk_V > = [vi, vj, vk] * [i, j, k]T
      < di_D, dj_D, dk_D > = [di, dj, dk] * [i, j, k]T

      origin = < 0, 0, 0 >
      center = < .5, .5, .5 >

      ;the PARTS of a cube, the DOMAIN of elementary elements

      PART = { < vi_, vj_, vk_> }
      VIRTEX = { < di_, dj_, dk_ > }
      EDGE = { <di_, dj_, ->  <di_, -, dk_>  <-, dj_, dk_> }
      FACE = { <di_, -, ->  <-, dj_, ->  <-, -, dk_> }
      SELF = { <-, -, -> }

      ;the operator which yields properties of the cube

      (p1_PART ^* p2_PART) = < ^*[p1.i p2.i] ^*[p1.j p2.j] ^*[p1.k p2.k] >

      ^*[ 0 0 ] = 0
      ^*[ 0 - ] = 0
      ^*[ 0 1 ] = -
      ^*[ 1 - ] = 1
      ^*[ 1 1 ] = 1
      ^*[ - - ] = -

      ;properties

      parallel[ p1_PART p2_PART ] = {

            p1_EDGE ^* p2_EDGE = _FACE
            p1_EDGE ^* p2_EDGE = _SOLID
            p1_EDGE ^* p2_FACE = _EDGE
            p1_EDGE ^* p2_FACE = _FACE
            p1_FACE ^* p2_FACE = _SOLID
            }
```

```
      perpendicular[ p1_PART p2_PART] = {

            p1_EDGE ^* p2_EDGE = _VIRTEX
            p1_EDGE ^* p2_FACE = _VIRTEX
            p1_FACE ^* p2_FACE = _EDGE
            }


      skew[ p1_PART p2_PART ] =

            p1_EDGE ^* p2_EDGE = _EDGE


      on[ p1_PART, p2_PART ] = {

            p1_VIRTEX ^* p2_VIRTEX = _VIRTEX
            p1_VIRTEX ^* p2_EDGE = _VIRTEX
            p1_VIRTEX ^* p2_FACE = _VIRTEX
            }


      connectedby[ p1_PART, p2_PART ] = {

            p1_VIRTEX ^* p2_VIRTEX = _EDGE
            p1_VIRTEX ^* p2_VIRTEX = _FACE
            p1_VIRTEX ^* p2_VIRTEX = _SOLID
            p1_VIRTEX ^* p2_EDGE = _EDGE
            p1_VIRTEX ^* p2_EDGE = _FACE
            p1_VIRTEX ^* p2_FACE = _FACE
              }

GenericBlock = {

      USE[ GenericCube ] = { cube }
      BLOCK = { [a1_ARITH, a2_ARITH, a3_ARITH] * [cube.i, cube.j, cube.k]T }
      IsCube[ b_BLOCK ] = ( b.a1 = b.a2 = b.a3 )
      }
```

```
CubesInaCube = {

      USE[ GenericBlock ] = { world b1 ... }
      worldscale = [1000, 1000, 1000]
      bigworld = worldscale * world
      location[ b_ ] = < bi, bj, bk >
      InWorld[b_] =
        (( <0,0,0> <= location[ b ] >= worldscale * <1,1,1> ) = true )
      location[b1] = <0,0,0>
      location[b2] = <1,0,0>
      }


StackOfBlocks = {

      USE[ GenericBlock ] = { world b1 ... }
      STACK = { [[ b1_ ... ]] }
      CONFIGURATION = { [[ b1___ ]]__ }

      emptytable = [[ ]]
      [[ ]] [[ ]] = [[ ]]
      location[ emptytable ] = < _, 0, _ >

      PutBlockOnStack[ b1_, s_CONFIGURATION ] =
        ( [[ b1 ]] [[ s ]] = [[ b1, s ]] )

      TakeBlockOffStack[ b1_, s_CONFIGURATION ] =
        ( [[ b1, s ]] = [[ b1 ]] [[ s ]] )

      On[ b1_, b2_ ] =
        ( [[ ___, b1, b2, ___ ]] = true )

      Above[ b1_, b2_ ] =
        ( [[ ___, b1, ___, b2, ___ ]] = true )

      OnTable[ b_ ] = ( [[ ___, b ]] = true )

      OnTopOfStack[ b_ ] = ( [[ b, ___ ]] = true )
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; STREAMS with DELAYED EVALUATION
;;;
;;;  Ideas from Pavel Curtis, Richard Waters, Guy Steele and
;;;    the Common LISP community.  Most of the code by
;;;    George Lugar and William Stubblefield.
;;;
;;;  DELAY and FORCE permit fine grain program control of when
;;;  a function or subprogram is executed.  Instead of flow of
;;;  control being managed by glue logic, loops, and function
;;;  nesting, evaluation is either delayed or called when needed
;;;  by the dynamic context of the program.
;;;
;;;  Conceptually, a STREAM is a dynamically available data
;;;  structure, implemented by a GENERATOR function:
;;;
;;;  {first-data-item <generator-function-for-next-data-item>}
;;;
;;;  When the first item is used by a process, the generator is
;;;  called to construct the next data-item.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defmacro delay (exp) `(function (lambda () ,exp)))
(defun force (function-closure) (funcall function-closure))

;;;add a new first element to a stream
(defmacro cons-stream (new stream)
  `(cons ,new (delay ,stream)))

;;;get the first element
(defun first-stream (stream)
  (car stream))

;;;the rest of the stream is a generator function
(defun rest-stream (stream)
  (force (rest stream)))

;;;test for empty stream
(defun empty-streamp (stream)
  (null stream))

;;;make an empty stream
(defun make-empty-stream ()
  nil)

;;;append two streams
(defun combine-streams (s1 s2)
  (append s1 s2))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;GENERIC-FILTER for LISTS

(defun filter (lst test)
  (cond ((null lst) nil)
        ((funcall test (first lst))
         (cons (first lst) (filter (rest lst) test)))
        (T (filter (rest lst) test))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;GENERIC-FILTER for STREAMS

(defun filter-stream (stream test)
  (cond ((empty-streamp stream) (make-empty-stream))
        ((funcall test (first-stream stream))
         (cons-stream (first-stream stream)
                      (filter-stream (rest-stream stream) test)))
        (T (filter-stream (rest-stream stream) test))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;FIBONACCI STREAM FILTER
;;;   consider generating the first N odd Fibonacci numbers
;;;   using the minimum of computational effort
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;the standard mathematical definition of Fibonacci numbers
;;;  fib[1] = 0
;;;  fib[2] = 1
;;;  fib[n] = fib[n-1] + fib[n-2]
(defun fib (n)
  (cond ((= n 1) 0)
        ((= n 2) 1)
        (T (+ (fib (- n 1)) (fib (- n 2))))))

(defun collect-fibs (n)
  (let ((result nil))
    (dotimes (i n (reverse result))  ;iterator i starts at 0, so we
must
      (push (fib (+ i 1)) result)))) ;add 1 to start fib counter at 1


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;an efficient doloop implementation which collects prior results
(defun memo-fibs (n)
  (let ((memo '(1 0)))
    (dotimes (i (- n 2))
      (push (+ (first memo) (second memo)) memo))
    (reverse memo)))

;;;a memoized doloop with an arbitrary filter
(defun filter-fibs (n filter-test-fn termination-test-fn)
  (do ((acc '(1 0))
       (test-acc nil))
      ((funcall termination-test-fn test-acc n)
       (reverse test-acc))
    (let ((new (+ (first acc) (second acc))))
      (push new acc)
      (when (funcall filter-test-fn new)
        (push new test-acc)))))
```

```
#|
;;; code fitted with lots of debug options,
;;;   to illustrate inline debug tools
(defun filter-fibs (n filter-test-fn termination-test-fn)
  (do ((acc '(1 0))
       (test-acc nil))
      ((funcall termination-test-fn test-acc n)
;(print (list 'VALUES-PRIOR-TO-DO-EXIT acc test-acc n))
;(break "At exit: count ~A fibs ~A filtered-fibs ~A" acc test-acc n)
       (reverse test-acc))
    (let ((new (+ (first acc) (second acc))))
;(break "new fib value: ~A"  new)
      (push new acc)
;(print (list 'new-stack acc))
      (when (funcall filter-test-fn new)
        (push new test-acc)))))
|#

(defun have-sufficient (lst n)
  (= (length lst) n))

(defun three-fives (n)
  (at-least-digits n 3 5))

;;; given a number N, find at least THIS-MANY of a given digit DIGIT
(defun at-least-digits (n this-many digit)
  (let ((digit-string (format nil "~D" n)) ;convert symbol to string
        (test-char (digit-char digit)))    ;convert symbol to character
    (count-digits-test digit-string this-many test-char)))

(defun count-digits-test (string num char)
  (>= (count char string) num))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;an implementation based on streams
;;;  cleaner, more maintainable, and as efficient as FILTER-FIBS
;;;  better than FILTER-FIBS when the termination test is complex
(defun fibonacci-stream ()
  (fib-stream-aux 0 1))

(defun fib-stream-aux (fib1 fib2)
  (cons-stream (+ fib1 fib2)
               (fib-stream-aux fib2 (+ fib1 fib2))))

(defun accumulate (n stream)
  (cond ((zerop n) nil)
        (T (cons (first-stream stream)
                 (accumulate (- n 1) (rest-stream stream))))))

(defun filter-stream-fibs (n test-fn)
  (accumulate n (filter-stream (fibonacci-stream) test-fn)))


;;;most abstract
(defun accumulate-filtered-stream (number-items generator-fn filter-fn)
  (accumulate number-items
    (filter-stream (funcall generator-fn)  filter-fn)))
```

```
#|
;;;filter examples
(filter '(1 3 -9 5 -2 -7 6) #'plusp)
;;==>  (1 3 5 6)
(filter '(1 2 3 4 5 6 7 8 9) #'evenp)
;;==>  (2 4 6 8)
(filter '(1 a b 3 c 4 7 d) #'numberp)
;;==>  (1 3 4 7)

;;;try these
(filter (collect-fibs 20) #'oddp)    ;yields only 13 odd fibs, so guess
(filter (collect-fibs 30) #'oddp)    ;to get 20 odd fibs
(time (filter (collect-fibs 30) #'oddp))
;;;don't try (filter (collect-fibs <some-guess>) #'three-fives)

;;;try these
(filter-fibs 20 #'oddp #'have-sufficient)
(filter-fibs 8 #'three-fives #'have-sufficient)
(time (dotimes (i 10) (filter-fibs 8 #'three-fives #'have-sufficient)))

;;;try these
(filter-stream-fibs 20 #'oddp)
(filter-stream-fibs 8 #'three-fives)
(time (dotimes (i 10) (filter-stream-fibs 8 #'three-fives)))

;;;try this
(accumulate-filtered-stream 8 #'fibonacci-stream #'three-fives)
|#
```

Technical Difficulties in Modeling and Knowledge Representation

(using blocks world in LISP as an example)

1. What is important to describe?
   Build little theories of little worlds.

```
(Block A)      (OnTable A)      (Hand Empty)
```

2. How should descriptions be partitioned?
   Functions or Relations, special or general objects?

```
(OnTable A)      (On A Table)      (not (OnTable Table))
```

3. How do we talk about groups and classes of objects?
   Quantification and abstraction

```
(All (x) (Block x))
```

4. How do we address things with no names?
   Functions as indirect, compound names.

```
(House-of John)
```

5. How do we handle things with more than one name?
   unique name hypothesis, unification

```
(Uncle John) = (Brother (Father John)) = Bob
```

6. How do we make general rules which define the structure of relations?
   quantification

```
(All (x) (iff (Uncle x) (Brother (Father x))))
```

7. How are typing and filters on domains represented?
   predicates in conjunction

```
(All (x) (and (Person x) (Father x y)))
```

8. How do we join more than one fact?
   conjunction

```
(and (F x) (G x))
```

9. How do we compute with logic?
   inference as natural deduction and as resolution

```
(if (and (P x) (if (P x) (Q x))) (Q x))
```

10. How do we compute with quantifiers and classes of objects?
    implicit universal quantification, Skolemization

```
(Exist (x) (P x)) ==> (P (Sk-1 x))
```

11. What is the difference between a fact and a query?
    query combination rules

    A. conjunction with negated query

```
(and (P x) (not (Q ?)))
```

    B. Skolemization of query variables

```
(Q ?) ==> (Q Sk-1)
```

    C. Facts imply Query

```
(if (P x) (Q ?))
```

    D. The answer predicate

```
(if (P x) (Answer x))
```

12. What kinds of rules do we need for query answering?

    A. definitions

```
(iff (P x) (Q (R x)))
```

    B. mathematical structures (symmetry, transitivity, etc)

```
(if (and (if (P x) (Q x)) (if (Q x) (R x))) (R x))
```

    C. permissible state transformations

```
(Pick-up x) = (Assert (not (onTable x)))
```

13. How can we control the inference/search procedure?

    A. Pre and Post conditions

    B. Compound queries

    C. Searching databases of rules and facts

14. How do we steer the resolution process?

    A. set of support

    B. ordered resoultion

    C. static vs dynamic approaches (compiled vs run-time)

    D. lookahead, cheapest first, dependency directed search

15. How do we express meta-level reasoning (rules about rules)
    measure the savings vs brute force

16. What is the appropriate reasoning strategy?

    look-up tables
    natural deduction
    resolution
    forward or backward chaining
    simulation
    boundary logic

17. What do we do about contradictions and inconsistencies?

    A. Forbid them

    B. Default reasoning

    C. Exception handlers

    D. Three-valued logic (True, False, Inconsistent)

    E. Multi-valued logic (eg True, False, Contradictory, Meaningless)

    F. Contradiction maintenance

Inappropriate Computation Dilemma

A senior professional software engineer in a large firm is asked by management to make a decision about using a battery of computerized selection tests for hiring new employees. The firm has the opportunity to purchase specialized software which tests the competence of potential employees and makes predictions about their success as an employee of the firm.

The engineer's analysis generates the following assertions:

1. No computer program can possibly do the job of selecting new employees.

2. However, the accuracy of selecting good employees is the same for either human or computer selection; both do a terrible job.

3. The computer uses demographic information (such as owning a home, living less than five miles from work, membership in professional organizations, type of car driven) which increases accuracy of selection but is known to be bias and is technically illegal to use.

4. Removing the demographic information degrades performance of computer selection to almost random. If such information were strictly forbidden during interviews by a person, the selection accuracy of humans would also degrade to almost random.

5. Management is looking for a way to reduce selection costs. Computerized selection accomplishes this objective, while human selection does not.

6. If computerized selection is used, the personnel department can be reduced by 70%, saving the firm $80 million per year. Computerized selection costs about $20 million per year, human selection costs about $100 million per year.

Management gave the senior engineer the responsibility to make the implementation choice. There are three alternatives, all lead to the same personnel decisions and to the same overt corporate behavior.

Choice 1: Keep the personnel department and the selection processes the same.
Total cost: $100 million

Choice 2: Institute computerized selection, even though it is both bias and inefficient.
Total cost: $20 million

Choice 3: Select new employees randomly by computer.
Total cost: $1 million

What choice should he make? Is there a professional ethical issue here?

What if, instead of hiring for a job, the computerized questions were to determine whether or not a person should be given a home improvement loan (or issued a credit card)? Would your decisions be the same? Would the ethical issues be the same?

Massive Impact Dilemma

In the Walt Disney movie *Flubber*, a professor invents an anti-gravity gel. He put a little flubber in his car, and immediately had a flying car. The following dilemma is about computational flubber.

A Computer Science professor has been working for many years on a new conceptualization of what computation is. He has published very little, because he has not yet come to a clear vision. One night, the professor has a dream in which he sees a new way to build computers. All of the work over the last many years condenses, and in the next two weeks, he designs and specifies a new type of computer, call it MetaShift computation.

The professor talks to several colleagues about the new idea, all under non-disclosure, since he suspects the idea may have commercial value. The net result after six months of close and secret collaboration, using only personal resources, is that the professor's team has built a new computer chip with some unusual advantages:

1. MetaShift computation is fully backward compatible. All programs which would run on vonNeumann architectures run on MetaShift. However, MetaShift has its own unique operating system.

2. MetaShift is fully reconfigurable. One MetaShift chip can be converted into *any* functionality within microseconds. With a MetaShift chip, specialized hardware (modems, decryption, cell phones, parallel processors, video acceleration, etc) is unnecessary. Instead the MetaShift is rapidly reconfigured into the desired functionality in real-time.

3. MetaShift works for all types of hardware architecture: CPUs, DSPs, floating-point coprocessors, real-time systems, cellular phones, automobile fuel-injectors, etc.

4. New programs written for MetaShift can be developed in a special language which is very easy to write, and provides automated debugging and program verification. Development time and cost for new programs is about 20% of that for other techniques.

5. The cost of manufacturing MetaShift is half of conventional fabrication costs.

6. The performance of a MetaShift chip is at least 5 times better than any other existing technology.

7. Because of the secrecy and uniqueness of the technology, it would be impossible for any potential competitors to market a similar product. That is, no competition to MetaShift would be possible.

When the MetaShift team did an in-depth market analysis, they discovered that within five years, MetaShift Corporation would probably take 50% market share from each of the companies listed below. Next to the company name is its market capitalization (market cap) in billions of dollars. (*Market cap* is the value of a company, the number of outstanding shares times the value of each share.) Further, when MetaShift went on the market, the market caps of these companies were likely to reduce rapidly to about 40% of current values.

COMPANY	***Market-cap ($billions)***
Microsoft	325
Cisco	270
Intel	249
IBM	196
SUN	98
TI	76
HP	72
Dell	68
Applied Materials	41
Compaq	40
Micron	27
Applied Micro	22
Xilinx	18
Altera	12
AMD	8
Apple	7
Total Impact:	$1.5 trillion

MetaShift itself would very likely reach a valuation of several trillion dollars within ten years. When MetaShift went to market, *most of the leading hardware and software operating system companies in the world will be put out of business within a few years*. As well, any company using computer technology would have to switch to MetaShift products within a few years in order to remain competitive. (This pressure is analogous to the impact of the Internet; the market dominance is analogous to any monopoly.)

The market analysis team then approached both the government and the financial sector with the commercial potential of MetaShift, disguised as a hypothetical story about innovation. After extensive consultation with the world's leading economic and regulatory authorities, MetaShift knew that if it went to market with its product, the following result was most likely:

> *Wide-spread economic chaos* would follow. The technology sectors of the world stock market would crash, losing $2 trillion in assets, and over a million people would loose their jobs. Next, several countries which were highly dependent on technology (US, Ireland, India, Germany, Japan) would enter a *moderate economic depression*.

At the same time, MetaShift would succeed spectacularly, making its first few thousand employees into billionaires.

QUESTIONS

Should MetaShift go to market with its product?
Are there ethical components to this decision?
Are there ethically appropriate compromise positions?
If MetaShift goes to market, should the government immediately intervene?

Capitalist Dilemma

[This is a *computer* ethics question since most of the computer and dotcom industry reaches evaluation through techniques somewhat similar to the extreme case which follows.]

In the United States, there are two institutions which can legally mint (that is, create) money. One is the Federal Treasury; the other is any corporation.

When a start-up is formed, it is given the right to issue stock. That stock (which can be created with almost any cash value) can be sold on the open market. Should the national stock exchanges grant the new corporation an Initial Public Offering (IPO), the stock becomes very easy to sell since it can be traded through the stock exchange to anyone willing to pay for it.

Naturally, these processes are regulated, but the basic idea is that corporations can essentially print their own money. This is one of the two major mechanisms through which the supply of money in our economy grows (the other way is for the Federal Reserve to increase the national debt by printing more money). Contrary to popular belief, incorporation is a “free lunch”.

The British Vancouver stock exchange is not as tightly regulated as the US exchanges. It is known as a “highly speculative” exchange. This means that there is no strict requirement to have any actual value for a corporation to reach IPO.

Here is a strategy for making money:

1. File for incorporation (cost is around $3000) as BogusCorp.

2. Issue junk stock with no value. Claim that the *potential* of BogusCorp gives it a value of, say, $1 per share. Issue 20 million shares.

3. Take the company to IPO on a speculative exchange. In reality, you usually take the company to a venture capitalist (VC) who provides the illusion of value by infusing the new company with temporary funds. The VC then takes the company to IPO for 80% of the profits.

4. There are always gambling and speculative investors. Assume that a few buy shares in BogusCorp, and $1M is raised.

5. Buy a company with actual value using the $1M as a deposit.

6. Now claim that BogusCorp has increased in value, the stock has actual value, and the company is succeeding. Tell your investors that their investment has increased in value, and watch the stock prices rise.

7. Cash out your own stock (which you give to yourself at incorporation for, say, 1 cent per share) as quickly as possible.

Question

Is there an ethical problem with the way our economy works?

Cyber-addiction Dilemma

From Communications of the ACM, 3/98, p11:

"Almost a fifth of college students spend more than 20 hours a week on the Internet...this amount of time qualifies as addition....a New York University study (that) correlates high student Internet use with doubled rates of academic dismissals. As a way of dealing with this problem, schools in Michigan, Maryland, Texas, and Washington have imposed limits on student Internet use. Dominant areas of user involvement: email, Web surfing, MUD interactive role-playing, and home page production."

ibid. p.128 (by Peter Neumann):

"...activities that can lend themselves to addictive or compulsive behavior include...even programming itself -- which seems to inspire compulsive behavior in certain individuals....computers intensify and depersonalize whatever activity is being done, enabling it to be done remotely, more expeditiously, less expensively, and perhaps without identification, accountability, or answerability.

The effects of compulsive computer-related behavior can involve many risks to individuals and to society, including diminution of social and intellectual skills, loss of motivation for more constructive activities, loss of jobs and livelihood, and so on. A reasonable sense of physical reality can be lost through immersion in cyberspace. Similarly, a sense of time reality can be lost through computer use that is totally encompassing and uninterrupted by external events."

Biological systems are incomprehensibly complex. Computational systems are incomprehensibly simple. Since the world we live in is beyond our comprehension, we construct projections (virtual worlds with detail removed) to support the illusion that we understand and are in control. The manufactured flat surfaces which surround us everywhere are an example of the removal of natural complexity to enhance our illusion of tractability.

People fall into cyberspace because it is unnaturally simple and therefore supports the illusion of competence. Of course, cyberspace is not simple, it too is an artifact of biological activity. It is the illusion of potential simplicity which makes computational systems attractive.

Questions

Why have you chosen a profession which requires you to stare at a computer screen all day?

Was your mother correct when she asked you not to sit too close to the television screen?

Is the modern mind committed /addicted to representations of reality (reading-writing-arithmetic, books, films, computers, etc.) rather than to reality itself?

How do you think physical reality will respond to the competition of virtual reality for the attention of humanity?

Logic Dilemma

As Computer Scientists, we may want to know: What is computation? Here are several possibly disturbing ideas about computing:

1. Formal logic defines the control structure of programs and the silicon/physical basis of computation.

2. Logic is underneath most of our culture's conceptual structures (at least in academia).

3. Logic is the simplest and most useful formal system, with the hardest computational problem (is `P=NP`?).

4. Logic has been in our language since the beginning of language.

AND

5. People do not use logic well, and have a long history of not understanding it.

6. Logic is inconsistent when self-reference is incorporated into the domain model. No program can refer to itself safely.

7. In computation, we convert the basic concept of integers into logical structures.

8. Deduction, and computation in general, consists of following meaningless tables and rules while transforming a string of characters from one form to another.

9. Natural deduction is too difficult to use for most logic problems. Machine-based resolution is too difficult to understand.

10. The if-then construct of logic is based on a confusing table mapping: if the antecedent is false, then any consequent is true.

IN FACT, computation as logic seems to be antagonistic to all the grounds of philosophy:

- *aesthetics*: Yuk, computation is difficult and confusing and meaningless
- *ethics*: Everything we program is simply timed logic, so a program's impact on culture is solely in terms of manipulating very limited digital logic forms.
- *epistemology*: How can we know anything when our basic tool is hard to understand and inconsistent?
- *metaphysics*: What is reality in an information society, where the virtual is defined by computation?
- *logic*: Ah! Logic itself is one of the five fundamental areas of *philosophy*.

Form into study groups of three members, to answer these questions:

1. Is there a problem in the above ideas?
2. If so, what is it and what can we do about it?

Triviality of Computation Dilemma

Quotes from Gian-Carlo Rota, **Indiscrete Thoughts**:

"The philosophy of mathematics carries out its work by focusing on the correlation between mathematical things and mathematicians." Robert Sokolowski, p.xiii

That is, between the object-concept of mathematical items (which may or may not exist in a Platonic world independent of our minds) and the process-concept of mathematical minds.

"Of all escapes from reality, mathematics is the most successful ever. It is a fantasy that becomes all the more addictive because it works back to improve the same reality we are trying to evade. All other escapes -- sex, drugs, hobbies, whatever -- are ephemeral by comparison." p.70

"Not only is every mathematical problem solved, but eventually every mathematical problem is proved trivial. The quest for ultimate triviality is characteristic of the mathematical enterprise." p.93

Computer Science deals with a trivial subset of mathematical triviality by excluding the sacred concept of Infinity and the mysterious concept of Void, and even by minimizing *intractable* (i.e. non-polynomial, search-based, mathematically interesting) complexity. Computer Science (at least Artificial Intelligence and Cognitive Science) pretends that the mind is like a computer, so that the issues of complexity of mind and of humanity can be conveniently ignored or forgotten.

Computer Science engages in an extreme of abstraction neurosis, let's say **abstraction psychosis**, by constructing the narrowest of worlds (binary bit-streams which interact only over timed Boolean networks), and then by suggesting that this extreme reduction is somehow whole, and somehow reflects physical reality. In fact, computation addresses only **trivial trivialities**.

Questions

How can humanity become so enamored with a technology that it forgets the reality within which it is embedded?

Why are we so ready and able to limit our experiences to a small screen of phosphors and a tableaux of a few dozen labeled keys?

How can our minds so easily confuse a pixel array with fully visceral experience? Confuse an email exchange with fully interactive human dialog? Confuse digital information processing with bodily experience?

What is the ethical dimension to these questions?

The Age of Mathematical Concepts and Symbols

Our clarity of understanding of mathematical concepts corresponds to the time evolution of these concepts. That is, *older is simpler*. As well, the sequence of math concepts taught in schools pretty much follows the historical evolution of mathematical ideas. Here is a rough road map of the time evolution of various mathematical concepts. Asterisks, *, mark content covered in class.

8000 BC*		one-to-one correspondence
4000 BC*		counting
1000 BC	.	zero (as dot)
400 BC*		zero as blank space
300 BC*	0	zero
300 BC*		syllogistic logic
1050	__	horizontal fraction bar
1417	+	plus
1425	%	percent
1432*		mathematician
1484		exponent
1484		billion, trillion,...
1530	0.0	decimal fractions
1544		division
1549		parallel
1551		irrational numbers
1551		theorem
1556*	()	parentheses
1557*	=	equals
1570		equation
1570		prime number
1575	x	variables as letters
1583	sin	sine function
1618	*	times (X in 1618, * in 1659)
1624	log	logarithm function
1631	>	greater/less than
1634		angle
1637		imaginary, real (Descartes)
1647	π	pi
1655	A,B,C	lettering for triangles
1655	∞	infinity
1672		“math” (Newton)
1674	cos	cosine function
1675	d/dx	derivative, integral
1690	e	base of natural logs

1718		probability
1734	f(x)	function symbol
1763		natural number
1770	∂	partial derivative
1777	i	imaginary unit
1786	lim	limit
1808	!	factorial
1816	ax = bx+c	linear equation
1827		long division
1839		“Fermat’s last theorem”
1840		pencil
1841	| |	absolute value
1843	[]	matrices
1848		factor
1851*		set
1882*		isomorphism
1883		eigenvalue
1887		tensor
1888*	U	union, intersection
1891		histogram
1892		standard deviation
1902*	e	identity element
1910*	~, V	not, or, and symbols
1921*		truth table
1931		spinor
1935*		homomorphism
1938		googol, googolplex
1940*	Ø	null set
1940*		onto
1975		fractal
1975		chaos
1989*		boundary mathematics

Functions

Ordered Pairs

We have seen elements, `a`, and sets of elements `{a,b}`. Adding an ordering relation creates a lattice of ordered functions. Each function is specified by a collection of ordered pairs, `(a,b)`.

Example:

> The logical function `(if a then b)` is defined by a collection of three ordered pairs of the form `(a,b)`, where the values of `a,b` are in the set `{0,1}`:
>
> ```
> if a then b =def= {(0,0),(0,1),(1,1)}
> ```
>
> The sixteen different ways of collecting the four possible ordered pairs, `N` at a time, `N=0..4`, define the sixteen different Boolean functions of two variables.

Functions and Relations

> *relation:* `xRy isTrue` *function:* `f(x)=y isTrue`
>
> The set of all first values of a set of ordered pairs is called the ***Domain***.
>
> The set of all second values of a set of ordered pairs is called the ***Range***.

A ***relation*** is a collection of ordered pairs over two sets, the domain set and the range set.

A ***function*** is a relation `(x,f(x))`, such that

1. Every member of the domain is associated with a member of the range, and
2. No element in the domain is associated with more than one element in the range.

Perspectives on Functions

1. *Formal constraints on a relation*

   existence: `all x inDomain . exists y inRange`

   uniqueness: `all pairs (x,f(x)) . if x1=x2 then f(x1)=f(x2)`

2. *Graph*

   Domain on x-axis, Range on y-axis
   uniqueness permits the graph to cross any vertical line (i.e. x-value) *only once*.

3. *Lookup table*

```
x      f(x)
1      1
2      4
3      9
```

4. *Static relation between variables*

```
x = y + 5
```

"=" is an equivalence relation

5. *Dynamic relation between variables*

```
f(x) = y
```

`x` is the independent variable (controlled measurement)
`y` is the dependent variable (observed measurement)

6. *Pure operation*

```
(lambda (#)  #^2 + # + 1)
```

`#` is the formal parameter of the function which binds to any value

7. *Sequence of combinators*

```
fac = (Y) lf.ln.(((0)n)1) ((*)n) (f) (-1) n
```

A tree of substitution instructions

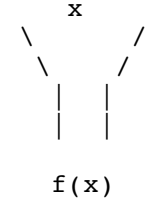
8. *Rule of correspondence/algorithm*

take a number	`x`
double it	`2*x`
add 3	`2*x + 3`

9. *Set transformation*

```
Domain          Range
  a    ----------->   b
  b    ----------->   c
  c    ----------->   d
  d    ----------->   d
```

10. *Input-output machine*



11. *Way of finding and assigning names to unnamed objects*

    `2^100` is the short name of a large number

12. *Digraph*

```
(1) ---> (3) ---> (5)
```

Types of Functions

Surjective, Onto, Epic	all `y` inRange, exists `x` inDomain . `f(x) = y`
Injective, 1-to-1, Monic	`if f(x1) = f(x2) then x1 = x2`
Bijective	1-to-1 and Onto

Bijective functions have an **inverse**, since every element in both the Domain and the Range are in correspondence:

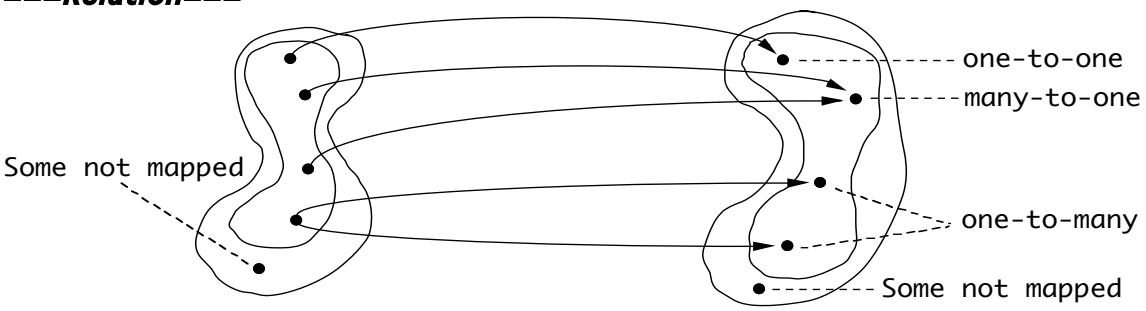
two-way existence	`all x inD, exists y inR . f(x) = y` `all y inR, exists x inD . f(x) = y`
two-way uniqueness	`all (x,f(x)) . x1 = x2 iff f(x1) = f(x2)`
inverse:	Exists `f-inverse` iff `f` is onto and one-to-one

Special Functions

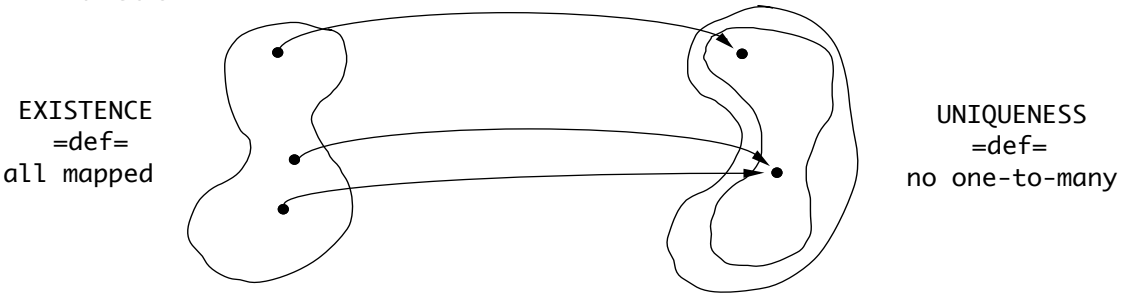
Identity	`f(x) = x`
Characteristic	`f(x) = 1 if x inA` `= 0 if x not inA`
Permutations	`(1,2,3) <--> (3,1,2) <--> (2,3,1)`
Sequences	`1 .. n <--> 1/1 .. 1/n`

Mappings

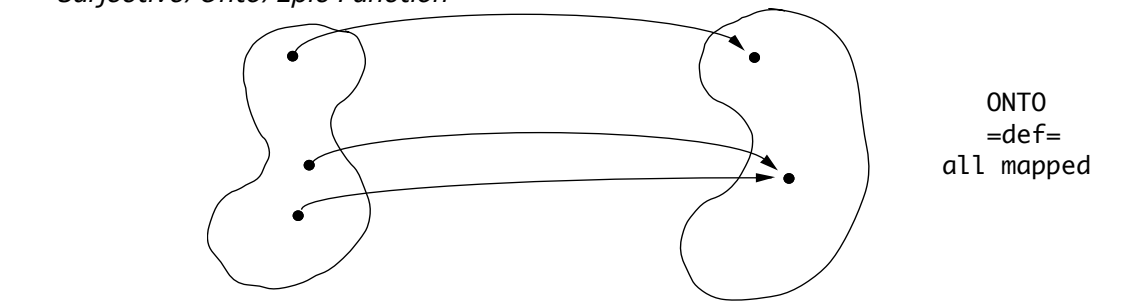
===Relation===



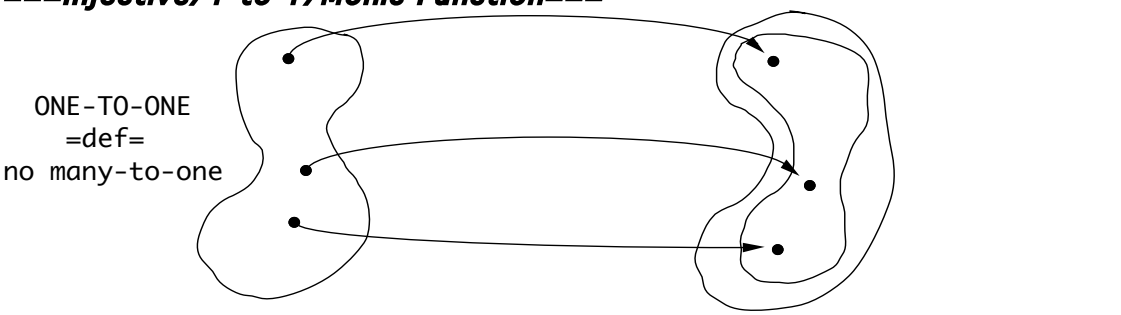
===Function===



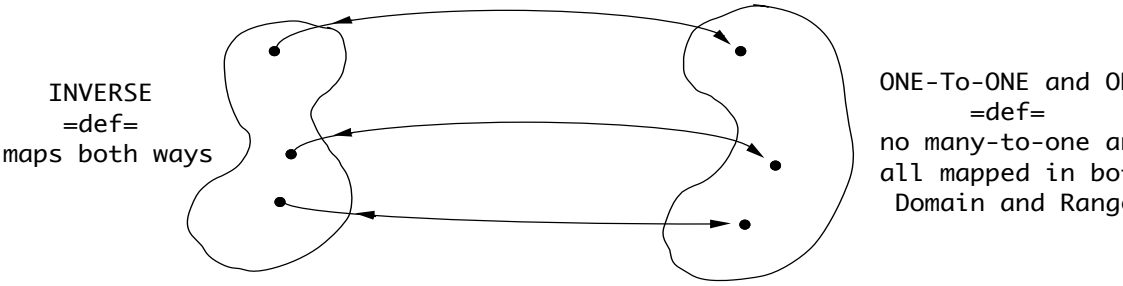
===Surjective/Onto/Epic Function===



===Injective/1-to-1/Monic Function===



===Bijective/1-to-1 and Onto Function===



Function Composition

`(f o g)` = All pairs `(x,z)` Exists `y` such that `(x,y)` in `g` and `(y,z)` in `f`
Note that the Range of g is a subset of the Domain of f

`(f o g)(x) = f( g(x) )`

Associative: `(f o g) o h = f o (g o h)`

Not commutative: `f o g =/= g o f`

Maintains the type of the function:

if `f` and `g` are functions, then `(f o g)` is a function
if `f` and `g` are onto, then `(f o g)` is onto
if `f` and `g` are one-to-one, then `(f o g)` is one-to-one

Composition of a function with its inverse:

`f o f-inverse = identity I` on Range of `f`
`f-inverse o f = identity I` on Domain of `f`

Inverse of a composition: `(f o g)-inverse = g-inverse o f-inverse`

Binary Functions

Binary functions are a mapping of ordered pairs onto elements: `((a,b) c)`
e.g.: `a + b = c  + = {((a,b),c) such that (a,b) in S X S and c inS}`

The domain consists of ordered pairs rather than single elements.

If `a`,`b`, and `c` are in the Domain,
then the Domain is closed with regard to the function:

`All x1,x2 inD such that f(x1,x2) inD`

Combinational Circuit Minimization

Section 4-11 of Floyd's *Digital Fundamentals* (handout) introduces a small practical example of the use of Boolean algebra in digital circuit design.

The segments of a familiar seven-segment display (labelled **a** through **g**) are activated to read-out as integers (0-9) by a network of logic gates actualized in silicon. The integers to be displayed are input into the logic circuit encoded as four binary bits (0000 to 1001, with 1010 through 1111 not used), in a code named BCD for binary-coded-decimal.

Thus the *BCD-to-7segment*. parsing problem is to convert four binary input signals into seven binary output signals in a given configuration. The parser/decoder itself is built onto a silicon chip.

There are many different configurations of logic gates which achieve bcd-to-7segment decoding. And there are many different criteria that a circuit manufacturer might wish to optimize when designing the decoder logic circuit. To get the best price at a particular silicon foundry, or for a particular substrate material, the logic network might need fewer wires but more gates, or it may need to consume very little power, or perhaps it might have fit a regular array of particular types of gates.

The parse logic can be represented as a set of seven equations in Boolean algebra, with four inputs (the four BCD bits) and seven outputs (the on-off bits for each segment). Fortunately, fundamental features of the mathematical representation map well onto important design features of the silicon circuit.

For example, here is one particular solution in which each parenthesis boundary is a logical NOR gate:

```
a = (( D B ((C)(A)) (C A) ))
b = (( D (C) (B A) ((B)(A)) ))
c = (( D C (B) A ))
d = (( (C A) (C (B)) ((B) A) ((C) B (A)) ))
e = (A ((C) B))
f = (( D (B A) ((C) A) ((C) B) ))
g = (( D ((B) A) ((C) B) (C (B)) ))
```

Inputs: The number of occurrences of input labels (A-D) is the fanout of the input, which relates to wiring and to power consumption. It is customary to use literals (either positive or negative occurrences of an input, as in A or (A)) as circuit inputs, since both signal and negated signal are usually available. This version has a low number of input references, 42, but 24 is a minimum.

Chip Area: The number of parentheses represents the number of NOR gates, which maps well to the surface area of silicon that the circuit will take. Again in counting gates, the (A) form is an input literal, and does not count in the gate count. This version uses 28 gates.

Wiring: Wiring is becoming the dominant design issue for sub-micron silicon layout technology. The number of wires in a circuit is indicated by the number of subterms of each above expression, viewing the parentheses as a representation of a tree structure. The above solution has few wires, 70 (equal to the number of gates plus the number of inputs), but it is easy to reduce this number.

Timing: Perhaps the most important design criteria for a combinational circuit is its *critical path*, the longest path from any input to an output. This determines the delay time of the circuit and thus the rate of the driving clock. Critical path is modeled by the deepest nesting of parentheses, since each parenthesis is a gate. This solution is well balanced for timing, several equations have the maximum depth of 3 gates.

Noise: The noise in a circuit refers to the

Power Consumption: When a signal passes through a logic gate

Finally, we must recognize that silicon layout introduces new geometrical issues which require the simple Boolean equation model to be extended. The primary example is *structure sharing*, when the output of a subtree is used more than once. This converts the Boolean tree model into a Boolean graph model. In the above example, the subcircuit "((C) B)" occurs in equations e, f, and g. These can be implemented as one circuit with three outputs, resulting in a savings of two gates and four literals. (The example is not well suited for structure sharing.) To indicate structure sharing, construct an new variable name for the shared structure:

```
n = ((C) B)
e = ((A n))
f = (( D (B A) ((C) A) n ))
g = (( D ((B) A ) n (C (B)) ))
```

EXERCISE

Find the (close to) minimal configurations of the BCD-to-7segment decoder for

number of simple NOR logic gates
number of wires between gates
length of critical path
number of literals

Early in the assignment, read the section from DeMicheli's *Synthesis and Optimization of Digital Circuits*. Try your understanding of optimization techniques on example 8.2.2.

SPATIAL REPRESENTATION of ELEMENTARY ALGEBRA

William Bricken
January 1992

Abstract

Our understanding of a concept is tightly connected to the way we represent that concept. Traditionally, mathematics is presented textually. As a consequence novice errors, in elementary algebra for example, are due as much to misunderstandings of the nature of tokens as they are to miscomprehensions of the mathematical ideas represented by the tokens. This paper outlines a *spatial algebra* by mapping the structure of commutative groups onto the structure of space. We interact with spatial representations through natural behavior in an inclusive environment. When the environment enforces the transformational invariants of algebra, the spatial representation affords experiential learning. *Experiential algebra* permits algebraic proof through direct manipulation and can be readily implemented in virtual reality. The techniques used to create spatial algebra lay a foundation for the exploration of experiential learning of mathematics in virtual environments.

1. Introduction

How we think about mathematical concepts is often constrained by our representation of those concepts. Syntax and semantics (representation and concept) are tightly connected. The addition operation, for example, is conceptualized as binary when written in linear text:

```
x + y
```

To add three numbers, we must use two addition operators:

```
x + y + z
```

Column addition, however, reconceptualizes the addition operation to be variary (one operator can be applied to an *arbitrary* number of arguments):

```
  w
  x
  y
+ z
----
```

Naturally, the addition algorithms and techniques taught to students differ for the different representations.

The traditional representation of binary addition is one-dimensional. There are two locations for arguments, one on either side of the textual operator. Column addition increases the dimension of representation to the plane; digits of individual numbers are expressed horizontally, different numbers are expressed vertically. From a spatial perspective, the number of arguments that can be added in one operation depends upon the dimension of the representation.

In general, how we represent numbers is a matter of convenience. For learning mathematics (and for doing mathematics) it is often more convenient to call upon visual interaction and natural behavior than it is to conduct symbolic substitutions devoid of meaning. *Spatial algebra* uses the three dimensions of natural space to express algebraic concepts. A higher dimension of representation greatly simplifies the visualization and the application of algebraic axioms. Algebraic transformation and the process of proof are achieved through direct manipulation of the three-dimensional representation of the algebra problem.

The difficulties children have when they begin to learn algebra are well documented [9] [7] [17] [8] [4]. Spatial algebra addresses common errors made by novice algebra students by permitting experiential interaction with abstract representations. Spatial representations enhance understanding [11]. Concrete manipulation is known to be an effective teaching technique [15] [1] [14].

Virtual reality is a computer generated, multi-dimensional, inclusive environment which can be accepted by a participant as cognitively valid [6]. VR teaching systems overcome the inconvenience of an insufficiently abstract physical reality by combining mathematical abstraction with the intuition of natural behavior. The programmability of VR allows a curriculum designer to embed pedagogical strategies into the behavior of virtual objects which represent mathematical structures [2]. Using a VR presentation system, the axioms of algebra can be, so to speak, built into the behavior of the world.

The visual programming community has developed taxonomies of visual approaches [13]. The experiential approach to mathematical formalism presented in this paper is sufficiently unique not to fit into existing taxonomies of visual languages. The approach of mapping formal operations onto the topological structure of space itself is not diagrammatic, iconic, or form-based. Most fundamentally, experiential mathematics imparts semantics onto the void (empty space). Actively using the void is both simple and conceptually treacherous [3]. The spatial techniques in this paper are general and have been applied to several formal systems, including elementary logic and integer arithmetic [3] [5].

2. Spatial Algebra

The components of space which can be used for the representation of mathematical concepts include:

-- empty space (the void),
-- partitions between spaces (boundaries, objects),
-- labeled objects which share a space, and
-- labeled objects which share a boundary (touch one another).

This is sufficient structure for the expression of elementary algebra. One possible map from algebraic tokens to algebraic spaces is:

Constants:			
	{ 1,2,3,...}	-->	{ labeled-blocks }
Variables:			
	{ x,y,z,...}	-->	{ labeled-blocks }
Operators:			
	{ + }	-->	{ sharing-space }
	{ * }	-->	{ sharing boundaries }
Relations:			
	{ = }	-->	{ partitions of space }

Examples of a spatial representation of the above map follow. The appendix to this paper contains a list of principles for designing spatial representations.

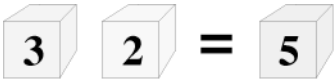
Constant as labeled block:

3

Variable as labeled block:

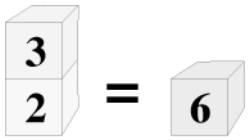
x

Space sharing as *addition*:



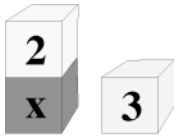
3 + 2 = 5

Touching as *multiplication*:



3 * 2 = 6

A simple algebraic term:



2x + 3

The gravitational orientation of the typography (top to bottom of page) in the above examples is not an aspect of spatial algebra, although gravitational metaphors are useful for the representation of sequential concepts such as non-commutativity. As well, the sequencing implied by stacked blocks is an artifact of typography; stacks only represent groups of objects touching in space.

3. Group Structure of Spatial Forms

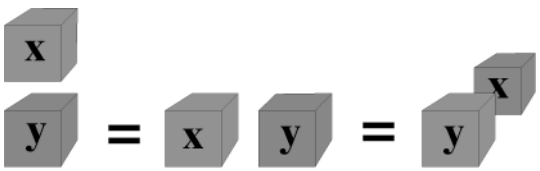
Generally, spatial representation can be mapped onto group theory. A commutative group is a mathematical structure consisting of a set and an operator on elements of that set, with the following properties:

-- The set is closed under the operation.
-- The operation is associative and commutative.
-- There is an identity element.
-- Every element has an inverse.

The integer addition and multiplication operators taught in elementary school belong to the commutative group.

3.1 Commutativity

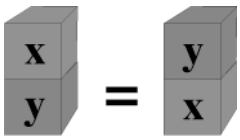
Spatial representation permits the implicit embedding of commutativity in space. The *commutativity of addition* is represented by the absence of linear ordering of blocks in space (visualize the blocks in this example as floating in space rather than in a particular linear order):



x + y = y + x

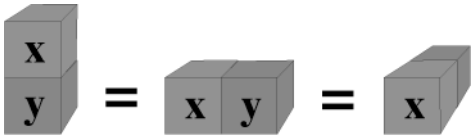
We intuitively recognize objects contained in a three-dimensional space as ordered solely by our personal perspectives. In contrast, typographical objects are necessarily ordered in sequence by the one-dimensional nature of text and by the two-dimensional nature of the page.

Commutativity of multiplication can be seen as the absence of ordering in touching blocks:



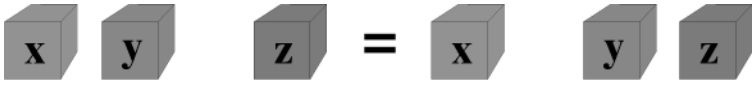
x * y = y * x

Again, in space there is no preferential ordering to touching objects:



3.2 Associativity

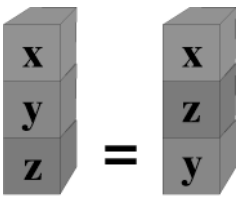
Associativity of addition is the absence of an explicit grouping concept in space:



(x + y) + z = x + (y + z)

The apparent visual grouping expressed by differences in metric distance between blocks can be assigned a semantics of associativity (for example, add closest objects first), or it can be ignored, permitting the operation assigned to space to address multiple arguments in parallel. From an intuitive perspective, operations embedded in space apply to any number of objects in that space. Whatever grouping we use is a matter a choice and convenience. Parallel computers provide techniques for addressing all objects at the same time.

Associativity of multiplication is the absence of an explicit grouping concept in piles:

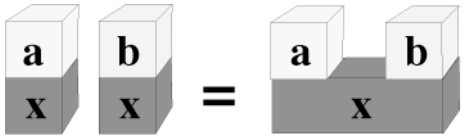


(x * y) * z = (x * z) * y

The apparent visual ordering of piles can be overcome by assuming that all objects in a pile touch one another directly. Rather than displaying stacked objects, VR might present objects in piles as completely interpenetrating. Every object in this non-physical representation is in contact with every other object, forming a Cartesian product of touching objects.

3.3 Distribution

Precedence operations associated with the distributive rule are the most common algebraic error for first year students [12] [4]. The representation of distribution in spatial algebra is particularly compelling. Generally, the distributive law permits combining blocks with identical labels into a single block with that label. Conversely (read right to left), *distribution* permits splitting a single block that touches separate piles into separate but identical blocks touching each pile:



$ax + bx = (a + b)x$

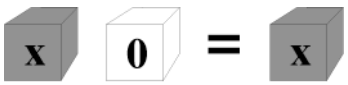
Blocks with identical labels are both singular and arbitrarily subdividable in space. This ability to arbitrarily divide and combine blocks with a common name is the same as the ability to arbitrarily create duplicate labels in a textual representation. Changing the size and the number of occurrences of a labeled block is easy in a virtual environment.

Any potential ambiguity between distributive idempotency and the use of space as the addition operator is avoided by the effect of *context* on interpretation. Idempotency requires the context of touching blocks (multiplication). Addition requires the context of non-touching piles.

3.4 Identities

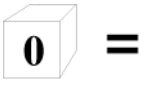
Zero is the identity element for addition. The identity in the spatial metaphor is the void; identities are equivalent to empty space.

The *additive identity*:

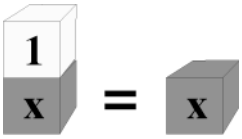


$x + 0 = x$

That is, zero disappears in space:

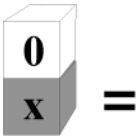


The *multiplicative identity*:



1 * x = x

The One block disappears only in the context of an existing pile. A zero in a pile makes the entire pile disappear:



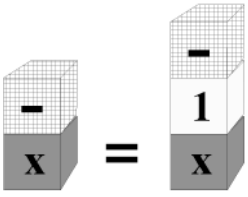
0 * x = 0

3.5 Additive Inverse

The inverse of a positive number is a negative number. *Negative numbers are the most difficult aspect of arithmetic for elementary students.* One way to directly represent inversion is to create an *inverter block*. Another way is to create an inversion space; for example using "under-the-table" for inverses. Inverses can be represented in many ways: as inverters, as colors, as orientations, as different spaces, as binary switches, as dividing planes, as inside-out objects.

In this version of spatial algebra, piles are inverted by the inclusion of a special inverter block:

Since a negative number can be seen as being multiplied by -1, the inverter block is expressed as touching (multiplying) the pile which is inverted:

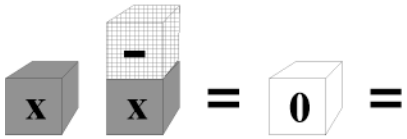


-x = (-1) * x

The inverter block expresses subtraction as the addition of inverses,

x - x is written as x + (-x)

The *additive inverse*:



x + (-x) = 0

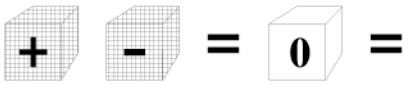
3.6 Calculus of Signs

The use of the inverter block for negative numbers introduces a calculus of signs into the algebra of integers. A sign calculus requires the explicit introduction of the positive block:

The positive block is the inverse of the inverter block. It introduces the concept of polarity and the act of cancellation. Numbers without signs are usually assumed to be positive. Making signs explicit removes this assumption.

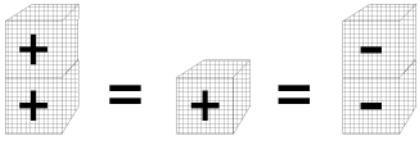
The following rules of sign calculus assume each sign has a unit value associated with it.

Additive cancellation in space:

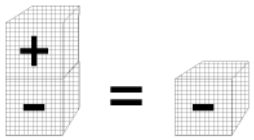


Cardinality in space:

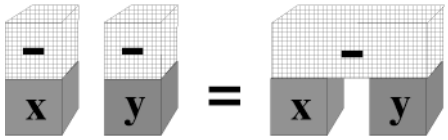
Multiplicative cancellation in piles:



Multiplicative dominance in piles:



The following example illustrates an inverter sign distributed across all objects in a space:



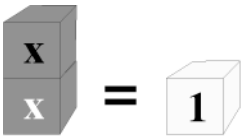
(-x) - y = -(x + y)

3.7 Multiplicative Inverse

Finally, division is the multiplicative inverse. Again, there are many possible ways to represent an inverse in a spatial representation. Since the traditional notation for fractions is primarily two-dimensional, it already has many spatial aspects. The division line that separates numerator from denominator could be carried over to the spatial representation as a plane dividing a pile into two parts. Here however, the multiplicative inverse is represented by inverse shading of the block label:

1/x

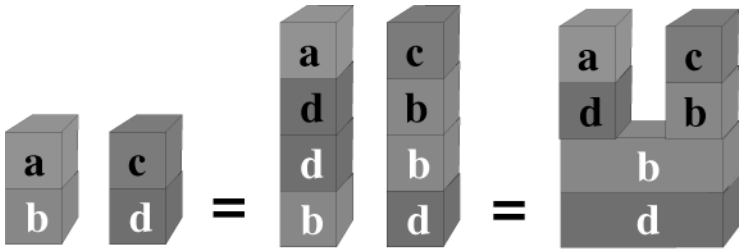
The *multiplicative inverse*:



x * 1/x = 1

One weakness with the choice to represent a reciprocal as differently shaded labels is that composition of reciprocals -- for example 1/(1/x) -- is not visually defined. Choice of representation necessarily effects pedagogy. It is an empirical question as to which representations facilitate learning algebraic concepts efficiently.

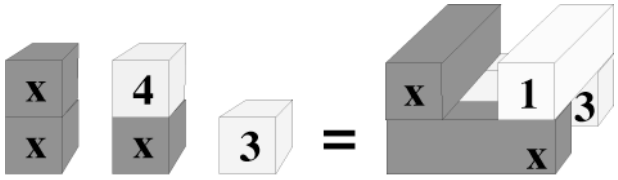
Fractions are the second most difficult area for students of arithmetic. A typical problem using fractions requires the application of the distributive rule:



a/b + c/d = (ad + bc)/bd

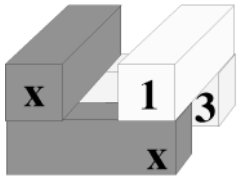
4. Factoring

Factoring polynomial expressions is equivalent to multiple applications of distribution. For instance:



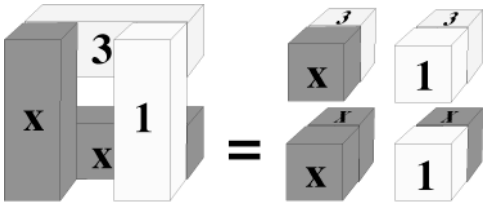
x^2 + 4x + 3 = (x + 1)*(x + 3)

One advantage of the spatial representation on the right-hand-side of this equation is that both the factored and the polynomial forms are visible concurrently. Looking from the side, we see two completely touching spaces which represent the factored form:



(x + 1) * (x + 3)

Looking down from the top, we see four piles which represent the polynomial form:



x^2 + 1*x + 3*x + 1*3

Here, the factored form is converted to the polynomial by slicing each addition space through the middle.

5. Caveats

Experiential mathematics is quite new as a formalism. The idea of mapping semantics onto the void first appeared in a mathematical text that is widely acknowledged as impenetrable [16]. Spatial algebra is an interpretation of the abstract mathematics developed by Spencer-Brown in *Laws of Form* and by Louis Kauffman at the University of Chicago [10].

The representational details of the spatial algebra presented here are, like any choice of syntax, somewhat arbitrary. The text lists many options, for example, for the representation of inverses. This representational freedom can be constrained by empirical studies intended to determine which particular representations are effective for task performance. There is no reason to believe that effective representations are generic. More probably, different individuals will prefer and understand different representations in the context of different tasks. One strength of VR is that it is completely customizable to individual participants. Still, the research to determine which representations are effective has yet to be conducted. In fact, demonstrating that spatial algebra actually improves performance in high school algebra remains as future research.

Significant components of a complete spatial mathematics have not been included in this paper. In particular, a compelling representation for exponentiation is missing. Spatial arithmetic has been assumed. A technical refinement is needed for the calculus of signs, to either remove cardinality completely from signs, or embed it deeper, expressing "-" as "-1". We also need to consider representation of functions such as the logarithm and sine. Spatial solutions to these shortfalls exist, but a completely integrated spatial mathematics is not yet formulated.

The weakest aspect of the proposed spatial algebra is the representation of three or more multiplied objects, x*y*z for example. This form can be represented by either completely interpenetrating blocks or by "blocks" with complex shapes that twist around to touch all other blocks. This problem gets particularly difficult for multiplying several factored expressions, for example: (x + 1)*(x + 2)*(x + 3)

In general, the cubic blocks presented in this paper are misleading, since they imply a Cartesian coordinate system. In fact, the spatial representation proposed here has no associated metric (or rather, the metric is irrelevant to the mathematical formalism). The treatment of space might be improved by explicitly including a representation for the table which blocks can be imagined to rest upon.

We also have little experience with animation of and interaction with spatial forms in VR. This paper presents the design phase of a wider study into the utility of VR for mathematics education[18].

6. Conclusion

Spatial representation provides a map to a wide range of new visual languages. The examples in this paper are expressed in a language of labeled blocks. The spatial rules, however, map just as easily onto people in a room, toys in a box, salmon in streams, and bricks in a wall.

The techniques of spatial algebra and the display capabilities of virtual environments have coevolved. Spatial algebra is proposed as an experimental approach for exploring the representation-dependent aspects of novice algebra errors. Virtual reality display systems are proposed as a straightforward way to present spatial algebra as an experiential mathematical system. During the next phase of this work, we will explore the pedagogical characteristics of spatial representations in virtual reality.

7. Appendix: Principles of Spatial Mathematics

I use the term *boundary mathematics* to describe the collection of rules and tools used to generate representations of spatial algebra. Boundary mathematics is general in that its principles can be applied to many mathematical domains. This paper, for instance, has implicitly assumed a model of spatial integers.

The roots of boundary mathematics can be found in G. Spencer-Brown's mathematical text *Laws of Form*. Boundary mathematics is quite unique, since it incorporates both the participant and the void into its formal structure. This makes formal theorems sound somewhat like pop psychology.

General Principles

1. Mathematics is the experience of abstraction.

2. Experience is not a recording. Representation is not reality.

3. The void cannot be represented.

4. Space requires participation. To participate is to partition space, to construct a boundary.

5. Boundaries both separate and connect.

6. Boundaries identify an intentional construction.

7. Representation and meaning are different sides of the same boundary.

8. Our body is our interface.

Mathematical Principles

9. Operators, invariants, and identities can be embedded in space.

10. Multiplicity is generated by observation.

11. Commutativity is embedded in space, ordering is embedded in time. All virtual entities are asynchronous parallel processes.

12. Associativity is the choice of the participant. All entities are autonomous.

13. Entities are both singular and plural in form, depending upon the construction of the participant. Entities with the same name are the same entity.

14. That which is common to every entity in a space is common to the space itself, forming the ground of the space.

15. Touching spaces are in pervasive contact (Cartesian product).

16. Crossing a boundary inverts a space. Inversion unites partitioned spaces.

17. Normalized spaces are those equivalent to the void. They can support arbitrary grounds.

8. References

[1] Berman, B., & Friederwitzer, F. (1989) *Algebra can be elementary ... When it's concrete* Arithmetic Teacher, 36 (8), 21-24.

[2] Bricken, M. (1991) *Virtual reality learning environments: potentials and challenges* Computer Graphics Magazine, ACM, 7/91.

[3] Bricken, W. (1986) *A simple space* Proceedings of the Sign and Space Conference, University of California at Santa Cruz. Also as HITL Technical Report R-86-3 , University of Washington, 1989.

[4] Bricken, W. (1987) *Analyzing errors in elementary mathematics* Doctoral dissertation, School of Education, Stanford University.

[5] Bricken, W. (1989) *An introduction to boundary logic with the Losp deductive engine* Future Computing Systems 2-4.

[6] Bricken, W. (1992) *Virtual reality: directions of growth* Proceedings, Imagina'92, Centre National de la Cinematographie, Monte-Carlo.

[7] Gerace, W.J., & Mestre, J.P. (1982) *The learning of algebra by 9th. graders: Research findings relevant to teacher training & classroom practice* Final Report, National Institutes of Health, Washington DC, (Contract # 400-81-0027).

[8] Greeno, J. (1985) *Investigations of a cognitive skill* Technical Report, Pittsburgh: University of Pittsburgh Learning and Development Center.

[9] Kaput, J.J. (1978) *Mathematics and learning: roots of epistemological status* In J. Lochhead & J. Clements (Eds.), Cognitive process instruction Philadelphia, PA: Franklin Institute Press.

[10] Kauffman, L. (1980) *Form dynamics* Journal of Social and Biological Structures **3**, 171-206.

[11] Larkin, J.H., & Simon, H.A. (1987) *Why a diagram is (sometimes) worth ten thousand words* Cognitive Science, 11, 65-99.

[12] National Assessment of Educational Progress (1981) *Results from the second mathematics assessment* National Council of Teachers of Mathematics, Reston, Va.

[13] Shu, N. C. (1988) *Visual programming* Van Nostrand Reinhold, New York.

[14] Shumway, R.J. (1989) *Solving equations today* School Science and Mathematics, 89, 208-219.

[15] Sowell, E.J. (1989) *Effects of manipulative material in mathematics instruction* Journal of Research in Mathematics Education, 20, 498-505.

[16] Spencer-Brown, G. (1972) *Laws of form* Julian Press, New York.

[17] Thwaites, G.N. (1982) *Why do children find algebra difficult?* Mathematics in school, 11(4), 16-19.

[18] Winn, W. & Bricken, W. (1992) *Designing virtual worlds for use in mathematics education* Proceedings of AERA, 1992.

TEACHING FOR INNOVATION

TOPIC 1. TEACHING PRACTICES

Best Practices in Teaching and Learning

TP: Good Teaching: The Top Ten Requirements

TP: Seven Principles of Good Practice in Undergraduate Education

Teaching Styles

TP: Just-in-Time Teaching

TP: Problem-based Learning

TP: Learning by Doing

Instructor Control vs. Learner Control

TP: Silence and Structure in the Classroom

TP: Minimizing the Distances Between Teacher and Student

TEACHING FOR INNOVATION

TOPIC 2. LEARNING STYLES

TP: Major Learning Theories of the Twentieth Century

TP: The Nature of Learning

TP: How Students Learn, How Teachers Teach, and What Goes Wrong

How People Learn

Learning Styles

Types of Learners

Meyers-Briggs Type Indicator

Multiple Intelligences

Talkers and Listeners

Teaching Examples (Bricken):

Management: Classification

TEACHING FOR INNOVATION

TOPIC 3. TEACHING STYLES AND METHODS

TP: New Technologies in Teaching and Learning: Evolution of Lectures

TP: Powerpoint Debate

Teaching Large Classes: Strategies for Improving Student Learning

Activity Breaks: A Push for Participation

TP: Problem Solving Through Design

TP: Asking the Right Questions in Class

TP: Keeping Discussion Going Though Questioning, Listening, Responding

TP: Tactics for Effective Questioning

TEACHING FOR INNOVATION

TOPIC 4. THESIS PREPARATION AND GUIDANCE

Small Piddly Projects, and Big Time Undertakings

TP: The Roles and Phases of Mentorship

TP: Combining Undergraduate Research and Learning

Teaching Examples (Bricken):

HCI: Project Ideas and Refinement

HCI: Design a Software Toolkit

Ethics: Local Expert

TEACHING FOR INNOVATION

TOPIC 5. CURRICULUM DESIGN

Teaching and Facilitating Learning Syllabus

TP: The Function of the Course Syllabus

TP: The Value of Writing a Course Portfolio

Syllabus Elements

Course Structuring

Cognitive Taxonomy

Affective Domain Taxonomy

Psychomotor Domain Taxonomy

TP: 101 Things You Can Do the First Three Weeks of Class

Teaching Examples (Bricken):

Situated Curriculum

Curriculum Exercises

Just What is VR Anyway?

Wonderful Computer Science Books

TEACHING FOR INNOVATION

TOPIC 6. STRUCTURING CONTENT

Lesson Plan Outline

How to Write Clear Objectives

Matching Objectives to Learning Styles

Teaching Examples (Bricken):

Formal: Proof Techniques, An Extended Example

Programming: A Small Interpreted Language

Programming: Pseudocode Assignment Package

TEACHING FOR INNOVATION

TOPIC 7. PROJECTS AND ASSIGNMENTS

Teaching Examples (Bricken):

HCI: HCI Assignments

HCI: Interface Design Simulation

AI: LISP Program Modification Exercises

Management: Formal Model: Card Games

HCI: Complete Window System

VR: 3D Interactive Virtual Worlds

VR: Expandable Virtual Cube World

DS&A: Exam Package

TEACHING FOR INNOVATION

TOPIC 8. SMALL GROUP ACTIVITIES

Managing Learner-Instructor Interaction and Feedback

TP: Group Presentations

TP: Establishing Ground Rules for Groups

TP: Integrating Team Exercises with Other Course Work

TP: Peer Instruction

TP: Difference Between Cooperative and Collaborative Learning

Teaching Examples (Bricken):

Management: simulation game

Management: archeologist, telephone drawing, consensus

TEACHING FOR INNOVATION

TOPIC 9. CHALLENGING THE STUDENTS

Contests Motivate Top Students in Large Classes

Teaching Examples (Bricken):

Foundations: Chapter 0 and responses

DS&A: Versions of Factorial

Management: Measurement

Management: Critical Incidents

Formal: Formal Cube, Algebraic Specification

AI: Streams with Delayed Evaluation

AI: Knowledge Engineering

Ethics: Six Dilemmas

Applications to Teaching Mathematics (Bricken):

Foundations: Timeline

Foundations: Functions

Formal: Combinatorial Circuit Minimization

Spatial Math

TEACHING FOR INNOVATION

TOPIC 10. TESTING AND EVALUATION

Nine Principles of Good Practice for Assessing Students

Assessment and Outcomes

Evaluating a Course

Teaching Examples (Bricken):

Foundations: Map of the territory

Ethics: Questions and text answers, course summary, content evaluation

Management: What you have learned

9 Principles of Good Practice for Assessing Student Learning

Alexander W. Astin; Trudy W. Banta; K. Patricia Cross; Elaine El-Khawas; Peter T. Ewell; Pat Hutchings; Theodore J. Marchese; Kay M. McClenney; Marcia Mentkowski; Margaret A. Miller; E. Thomas Moran; Barbara D. Wright

1. The assessment of student learning begins with educational values.

2. Assessment is most effective when it reflects an understanding of learning as multidimensional, integrated, and revealed in performance over time.

3. Assessment works best when the programs it seeks to improve have clear, explicitly stated purposes.

4. Assessment requires attention to outcomes but also and equally to the experiences that lead to those outcomes.

5. Assessment works best when it is ongoing not episodic.

6. Assessment fosters wider improvement when representatives from across the educational community are involved.

7. Assessment makes a difference when it begins with issues of use and illuminates questions that people really care about.

8. Assessment is most likely to lead to improvement when it is part of a larger set of conditions that promote change.

9. Through assessment, educators meet responsibilities to students and to the public.

Assessment and Outcomes

Assessment is an iterative feedback process for continual program improvement:

1) Define intended program learning objectives, specifically, what do we want our graduates to know and actually to be able to do?

2) Define measurable outcomes that will serve as evidence of how well each objective has been met, and then actually to measure them. Because this step requires explicit articulation of program success criteria, it often has the added benefit of clarifying faulty assumptions.

3) Compare actual observed outcomes to intended program objectives: how well did we meet our objectives in general, and our student learning objectives in particular?

4) Based on how well or how poorly achieved outcomes compare to intended outcomes, elements of the program (including assessment elements) are redesigned as appropriate, and a new assessment cycle begins.

The fundamental role of assessment is to provide a complementary methodology for monitoring, confirming, and improving student learning.

Good assessment practice is based on several assumptions:

- -- Good assessment practice assesses what is most important.
- -- Anything that can be taught or learned can be assessed.
- -- Assessment should be applied at course, program and institutional levels.
- -- Every program and every course should be organized around clearly articulated learning goals and objectives, explicit assessment methods, and measurable outcomes.
- -- An assessment process should be logistically feasible and practically manageable to insure that it is regular and ongoing.

In the traditional "teacher-centered" model, the focus has been on inputs: the credentials of faculty, the topics to be presented, the sequencing of presentations, and so forth.

In the "student-centered," or "learner-centered" model, the focus is on outputs: what knowledge have students actually acquired, and what abilities have they actually developed?

- -- The primary measure of program success is what graduates actually know and are able to do.
- -- Student-centered programs are competency-based.
- -- Learner-centered education is dedicated to continual improvement through ongoing assessment of student learning.

Evaluating a Course

Ferraro Smith – "The main goal of evaluation is to improve learning. In classroom settings, evaluation centers on the quality of instruction, the environment and the course materials, as well as knowledge and test scores".

1. Instructors should evaluate the quality of their teaching to improve it.
2. Instructors should know if the students are learning what they should learn.
3. Instructors should know if the course fits into the goals of the department.
4. Good course evaluation is important in promotion and tenure processes.

Don't wait until the end of the course to evaluate teaching.

Formative Evaluation: Usually conducted during the process of instruction to evaluate student learning in order to improve teaching.

Summative Evaluation: Conducted at the end of the instruction to determine the effectiveness of the teaching/learning process.

Feedback from different sources:
- -- Classroom observation
- -- Students assessment
- -- Teaching staff
- -- Other "stakeholders" (employers, former students, external examiners)

1. Mid-semester evaluation:
- -- Is the course moving too slowly? Or too quickly?
- -- Does the absence of questions indicate comprehension? Or confusion?
- -- How do students feel about being in the class?
- -- Are they more interested in the subject matter or less than at the start?
- -- What's the most important thing they have learned thus far?

2. Develop a teaching portfolio to collect multiple sources about teaching effectiveness:
- -- A teaching portfolio is a "factual description of a professor's major strengths and teaching achievements. It describes documents and materials which collectively suggest the scope and quality of a professor's teaching performance" (Seldin, 1997, p. 3).
- -- Collect samples of teaching materials such as syllabi, assignments, etc.
- -- Collect samples of students learning such as papers, projects, etc.

3. Review and reflect on teaching process:
- -- Reflections on the choice of class activities
- -- Reflections on specific measurements
- -- Reflections on the ways to provide feedback, etc.

Final Project

HAND IN AT THE BEGINNING OF CLASS.

Make of a map of the territory of discrete mathematics.

- Include each of the topics we have covered in class, and each topic in the text.

- Include the defining characteristics of each separate topic: the domain, the axioms, the essential idea.

- Pay particular attention to the relationships between topics.

- Order your map so that it clearly displays which topics are subsumed by (or are subsets of, or "inherit the characteristics of") other topics.

- Distinguish between old and new topics, between topics that are well understood by the mathematical community and those that are still evolving rapidly.

- Next, assign to each topic and to each collection of defining characteristics, three rating values:

    1. *your understanding* of the topic or characteristic
       (0= no understanding at all, 10 = total and complete understanding)

    2. *your confidence* in the above understanding rating
       (0 = no confidence, 10 = complete confidence)

    3. *the importance* of the topic to you
       (0 = completely irrelevant, 10 = completely relevant and important)

- You may want to extend your map with topics that you consider to be part of discrete math, but were not covered in class.

- *ONE PAGE* only please.

Initial Perspectives, Johnson's Responses (from the text)

Answer these questions briefly:

1. *What is ethics?*

   The evaluation of arguments, reasons, and theories that justify morality.

   Exploring what humans beings ought to do.

2. *What is ethical Computer Science?*

   Professional ethics is strongly differentiated,
   the role gives the role-holder special powers and responsibilities.

   Computer professionals owe a responsibility to their field and to society.

   Following the code of ethics for the profession.

3. *What are some major issues in Computer Ethics?*

   professional ethics
   privacy
   free speech
   property rights
   accountability
   social values

4. *What is unique about Computer Ethics as opposed to regular ethical behavior?*

   new entities: programs, software, microchips, web sites, video games
   massive scale, power and pervasiveness
   new kinds of knowledge
   new levels of complexity and unreliability
   new instrumentation for new types of human action
   virtuality

Summary of Computer Ethics

Make a list of items which you consider central for each of the topics below.
My answers plus class readings:

- ***techniques of ethical analysis***

  case studies and dilemmas
  utilitarianism
  idealism
  relativism
  human rights and duties
  fairness and justice
  virtue
  ethical egoism
  communal good
  psychometrics, sociometrics
  argumentation
  information mapping
  professionalism
  deep feeling
  informed discussion
  codes of ethics
    Ten Commandments for Computer Ethics
    ACM Code of Ethics and Professional Conduct
    Software Engineering Code of Ethics: Approved!
  spiritual belief

- ***central issues for computer ethics***

  privacy
    Camp, Web Security and Privacy
  ownership and property rights
    Spinello, An Ethical Evaluation of Web Site Linking
  accountability and agency
    Ulrich, IT Ethics
  trespass (hacking) and computer crime
    Tavani, Defining the Boundaries of Computer Crime
  regulation and control
    UTICA Case Study
  accuracy and digital trust
    Smith, Limits of Correctness in Computers
  instrumental, digital reasoning
    Weizenbaum, Computerized Gods
    Weizenbaum, Against the Imperialism of Instrumental Reason
  disembodiment (avatars, cyborgs, simulacrum)
    3D Interactive Virtual Worlds

the nature of software (open-source, patentable, etc)
Walker, Programs Are Programs
Lessig, The Laws of Cyberspace
Raymond, The Cathedral and the Bazaar
access and cyberliteracy
security and encryption
freedom of speech and censorship
the differences between reality and virtuality, social values

- ***ways that computer ethics impacts society and culture***

living in virtuality
Berry, Why I Am Not Going to Buy a Computer
Virtual Reality, As Unreal as It Gets
trust in digital machines
Inappropriate Computation Dilemma
ecommerce and the new economy
Capitalist Dilemma
transformation of the work environment
Kreie & Cronan, Making Ethical Decisions
addiction
Cyber-addition Dilemma
technological utopianism
Joy, Why the Future Doesn't Need Us
corporate giants
Massive Impact Dilemma
hacktivism
Manion & Goodrum, Terrorism or Civil Disobedience:
Toward a Hacktivist Ethic
ownership of virtual properties
surveillance and loss of privacy and autonomy
electronically mediated communication
digitization of cognition
administrative cluelessness
new paradigm for science
Triviality Dilemma

- ***how you anchor you own ethical perspective?***

deep feeling
ethical egoism

Lecture and Handout Evaluation

Do not put your name on this. Check one rating box for each topic or handout.

	RATING			
	didn't read don't recall	***good***	***neutral***	***not good***

Textbook

Johnson, Computer Ethics, Third Edition.

Class materials

Initial Perspectives

Ethical Issues and Case Studies

Case Study Practice; UTICA Case Study

Resources

Final Assignment (local expert)

Inappropriate Computation Dilemma

Capitalist Dilemma

Cyber-addiction Dilemma

Triviality Dilemma

Massive Impact Dilemma

3D Interactive Virtual Worlds

Virtual Reality, As Unreal as It Gets

Summary of Computer Ethics

RATING

didn't read
don't recall ***good*** ***neutral*** ***not good***

Class Content

1) Th 1/4
Intro to Philosophy and Ethics
2) Tu 1/9
Conceptual frameworks
3) Th 1/11
Case-study methodology
4) Tu 1/16
Case-study practice
5) Th 1/18
Psychometric methodology
6) Tu 1/23
Professional ethics
7) Th 1/25
Uniqueness of software
8) Th 1/30
Netiquette
9) Th 2/1
Security, encryption
10) Tu 2/6
Freedom of speech
11) Th 2/8
Regulation, content control
12) Tu 2/13
Privacy, data mining
13) Th 2/15
Privacy
14) Tu 2/20
Agency, accountability
15) Tu 2/22
Hacking and crime
16) Tu 2/27
Ownership and property
17) Th 3/1
Intellectual property rights
18) Tu 3/6
Social impact

RATING

didn't read
don't recall good neutral not good

Articles

Ten Commandments for Computer Ethics

ACM Code of Ethics and Professional Conduct

Software Engineering Code of Ethics: Approved!

Camp, Web Security and Privacy

Spinello, An Ethical Evaluation of Web Site Linking

Ulrich, IT Ethics

Tavani, Defining the Boundaries of Computer Crime

Smith, Limits of Correctness in Computers

Weizenbaum, Computerized Gods

Weizenbaum,
Against the Imperialism of Instrumental Reason

Walker, Programs Are Programs

Lessig, The Laws of Cyberspace

Raymond, The Cathedral and the Bazaar

Berry, Why I Am Not Going to Buy a Computer

Kreie & Cronan, Making Ethical Decisions

Joy, Why the Future Doesn't Need Us

Manion & Goodrum, Terrorism or Civil Disobedience:
Toward a Hacktivist Ethic

FINAL EXAMINATION

Next week, you will be asked to complete a final exam. This sheet contains the final examination question, the rules of the game, and hints about how to do your best on the exam.

Final Exam Question

Record what you have learned in this class.

Final Exam Rules

Answer the question in real-time, during the exam time slot. However, you may do any amount of preparation and bring any resources to the examination room.

Materials prepared at home can be attached to the real-time exam so long as your exam discusses them.

Hints for Doing Well

Honesty counts double.

Use any media, but writing is the obvious default.

Be abstract. Don't recount every activity and every thought. Rather, summarize and condense.

Be brief. This is not a justification of your learning, it is a recording.

Charts, graphs, decision networks, and mathematical models are highly encouraged. These techniques summarize information well.

Be self-referential. It's a big win if you can illustrate what you have learned in the way you talk about what you have learned.

Avoid fantasy. Don't talk about what you wish you have learned.

Use succinct nuggets, piercing clarity, meaningful indicators.

Illustrate ideas with brief personal stories.

The idea is for you to generate a real-time reflection of what you have learned. When you know something, it is easy to record.