RULEBUILDER ENTITIES

William Bricken
June 1994

RuleBuilder is a software toolkit for designing educational experiences in virtual environments.

Entities are the primary organizational structure for the RuleBuilder software toolkit. Like the match-and-substitute computational approach of the silicon layer, entities provide a uniform, singular metaphor and design philosophy for the organization of software resources and displays.

An entity is a collection of resources that can accomplish a specific task. The operating system itself is the prototype entity. That is to say, the functional core of an operating system provides management of data, processes, and communications; every entity in the instructional design has these same capabilities.

In virtual environments, entities can have 3D display properties. Each entity in a virtual world can also have behavioral properties, so that they interact dynamically with the environment. In screen-based multimedia, entities are windows which exhibit display capabilities such as video and sound. Multimedia entities can also exhibit intelligent behavior, such as knowing how to resize in any display environment, or even filtering display content which is irrelevant to the task being learned by a participant.

Internal Processes

The internal process of an entity consists of a sense-process-act loop. This loop is called a behavioral cycle.

The analogy to human behavior is not strong. An entity senses by gathering relevant data from the database. Relevance is the key. Each entity has a "perceptual" filter on the database which limits the amount and the type of data that the entity must process on each behavioral cycle. Filters are patterns which must be matched for a data fragment to be relevant to the entity. For example, an entity configured with a sound play-back capability can sense sound bits stored in the database. If it lacks a sound capability, it will ignore all data labeled as sound. Since filters are expressed in terms of matching, they are implemented by concurrent match-and-substitute techniques. That is, all information relevant to a particular filter is matched in parallel, in a few cycles of the CPU clock.

The processes internal to an entity are controlled by two separate processing loops. The React loop reads sensed data and immediately responds by posting modified data to the common database. This cycle handles all real-time interactions and all reactions which do not require additional computation or local storage. The Persist loop runs on resources local to the specific entity, and is not responsive in real-time. Persist loop computations typically require local memory, function evaluation, and inference over local data. Persist functions can copy data for the shared database and perform local computations in order to gather information, but there is no time constraint on returning results. By decoupling local computation from environmental reactivity, entities can respond in a timely manner while complex responses can still be evaluated whenever computational resources permit.

Dynamics

Entity dynamics is achieved by associating behavior functions with sensory input and with internal processes. The RuleBuilder toolkit provides an extensive array of designed behavioral functions. The pedagogical style for a particular lesson, for example, could be selected from the following

    ignore errors
    store errors without notice
    notify errors immediately
    notify repeating errors
    notify errors and display correction.

Each style would be automatically applied across the lesson, while the style of notification could be different for each different trainee.

The RuleBuilder interface tools provide templates for designing and writing the logical structures that trigger behaviors. The interface can be as simple as "Copy my actions", which would trigger a rule that recorded the actions of the user for later playback and display.

Editing dynamic characteristics of entities is accomplished by the rule based interaction tool for creating entity behaviors, RuleBuilder. Behaviors can be algorithmic, reactive, responsive, inferential, coordinated or autonomous.

Entities exhibit persistent behavior by running algorithms that do not interact with the environment. This is the general case with most programmed simulations.

The RuleBuilder toolkit permits associating any entity attribute with an arbitrary looping function, causing that attribute to vary over time. Smooth functions create continuous behavior; chaotic functions create complex behavior. Branching behavior is provided by logical rules which use conditional functions.

Reactive entities have rules that trigger when specific events are posted to the environmental database. The trigger is expressed as a perceptual capability of the entity. Entities which can perceive and react to a hand gesture, for example, can perceive (can read the database of) the participant's hand, and in particular, changes in the relative configuration of the fingers. That entity will also have a rule which causes it to post an action to the database. The action might be to approach open hands.

RuleBuilder, the dynamics editor, provides simple tools for asserting perceptual limitations and preferences for constructed entities, for asserting reactive rules in if-then-else format, and for prioritizing the distribution of computational resources available to the entity for behavioral activities.

Responsive entities also react to their environment, but they have memory resources to store previous experiences. These entities can exhibit complex delayed responses and critical incidence behavior. A screw being inserted into the wrong hole by a novice carburetor repair trainee may wait until the third repetition of the error before responding with a correction. RuleBuilder provides simple accumulation commands for specifying memory dependent behavior.

Inferential entities have a small inference engine which they can apply to their accumulated database of experiences and internal rules. Although the search is time consuming, inferential entities can find interesting, useful deductions which can generate unique behavior. Inference can also be used for goal directed activity such as planning and information gathering. RuleBuilder makes coupling inference with data straightforward by providing mathematical relational structures, such as automated rules of symmetry, transitivity, and distribution.

Coordinated entities share a common time stamp and synchronized clocks. Synchronization is a simple command for RuleBuilder. Coordination rules not only provide flocking behavior in entities, they can be applied to multiple participants, assuring, for instance, that classes of trainees all complete similar training experiences. The concept of coordination can be generalized to the existence of a measure relation between arbitrary attributes of multiple entities. RuleBuilder, for example, permits the linking of "all Blue entities" so that their behaviors might exhibit a group constraint.

Autonomous entities provide exploratory tools for emergent behaviors. RuleBuilder provides the capability of observing multitudes of similar entities as their behavior unfolds over time. The designer, rather than having to specify all behaviors concretely, can just watch for behaviors of interest, then specify the particular entity for further cloning and observation. These *genetic techniques* are computationally expensive, and we do not know the size and variation of populations that can be supported in real-time.

The most complex entity is one that is inhabited by a human participant. In *inhabited entities*, dynamic behavior is slaved to physical transducers attached to the participant. These signals are standardized to the participant's physiological body model. The physiological model is then mapped onto a virtual body, which can be an arbitrary representation. A participant teleoperating a claw, for example, may prefer his virtual body also to be represented by a claw, even though the physiological model is of a hand.