NOTES ON SELLING FRACTURTLES

William Bricken
August 1988

Management:

Having a programmer work on it in his "off hours" isn't the best way to get a product out the door. But in order to get people interested in turning it into a product, someone has to answer the questions that determine the market potential; at least to some degree. That means, simply, that I (or you or whoever) have to be able to answer questions like, "Who is going to use this?" and "what's does it do for them?" and the all-powerful (and sometimes indefinable) "what is the market for this thing?".

To give you a recent example of a "good new idea", consider a digital painting program. We convinced marketing that the "desktop video" market was something that existed (by way of references to other, somewhat similar products), that there was a unique opportunity (because the products didn't exist on PC's yet), and that we had just the right people to do the work (because they had done one for the Atari and Amiga already). The "research phase" of the project lasted about 2 weeks, until we could determine that the hardware (IBM VGA and clones) was capable of supporting animation; if it didn't we wouldn't be able to do anything.

So these are the kind of issues that people are interested in finding answers to before assigning resources to a new product.

If Fracturtles has potential, then it needs to be explained to people what that potential is, and it needs a design that exploits that potential. Or we can make it into something separate that explores the benefits of procedural object definition vs. static description as a sort of educational tool.

It seems to me that what you have right now is a general purpose system for drawing recursive pictures. The question is, is THAT something that should be turned into a product, or are there some things we should/can wrap around it to make something else? Either way, we can make something, but we need a rough feel for who needs a tool for making recursive pictures, how many of them are there, etc.

William:

Your message makes sense, and I deeply appreciate the time you take to lay out strategy.

Here's a synopsis about market potential:

Who will use it: anyone who uses a drawing tool.

What does it provide: easy, unique drawing capabilities, including natural modeling of plants, clouds, mountains. Tools for easy specification of complex drawings, including fractals, graftals, iterated spirals, repetitive mechanical parts. Tools for easy duplication of objects, for abstract spatial layouts, for graphical programming including conditional figures. Fast and efficient algorithms for graphics.

What is the market: folks generating drawings that would be enhanced by natural figures, abstract figures, symmetrical figures, mechanical figures. All educational levels, as a compelling toy and as a tool for teaching graphic manipulation. Particularly empowering for young kids.

Why it is a good idea: CAD is moving solidly into modeling, now that drawing is substantially covered. This is a formal modeling tool of extreme generality.

The compelling test: Let's take the prototype into a classroom, let folks play with it, and ask if they would buy it. Multiply the percent yes by the number of kids in the country to estimate the market. Or just do it in house.

Fracturtles is drawing theory. It's a general purpose system for drawing recursive pictures. Have you ever said "recursive pictures" before? Isn't that alone -- *a new class of drawing* -- sufficient for interest? But Fracturtles is much more. I focused on *multiplying drawings* but also showed *adding drawings*. Its a *drawing algebra*, a system that lets the user enter sketches, while at the same time letting the implementation apply powerful abstract rules and transformations.

I am fairly familiar with capabilities of (low-end) drawing tools, and with their market. This offers unique functionality. We do not know *how* people will use it, we do know that however they choose to use it, it will not break. There is no way to gather market statistics on new, unique capabilities, cause people don't know to ask for what they haven't yet imagined. I can prove the value given resources. For instance I have expertise in experimental design, and given the support to conduct market surveys, I can do so (my first degree was in market research, under the banner of Social Psychology).

So exactly who do I need to convince? What more do I need to add to create a leap of faith? And where in the above synopsis is there a weak point?

William:

I'm motivated to offer one more comment.

You characterized the fracturtle work as "intrinsic coordinates". I said it was coordinate free. After thinking about it last night, I believe that your conception of geometry is limited to what Alvy Ray Smith calls the "cubist approach".

The string representation of a figure is *structural* only, and has no reference or requirement of coordinates. Computation over strings is not even metric. The key insight is that coordinates are introduced *in the process of interpreting* strings for graphic display. "F" can be interpreted as forward one unit, or it can be interpreted as one structural element without reference to a metric. The main point is that the implementation does not introduce an interpretation; its basic operations are concatenate and substitute strings. To the implementation, the form is abstract and structural. When we want to see an interpretation of the form, we add coordinates.

So the main contribution of fracturtles is a non-metric algebra of form that has a consistent interpretation for display. No cubist drawing package offers form abstraction.

Yes, existing drawing packages can duplicate this functionality, but the critical difference is that to do so, the user must provide the model in his head. Fracturtles provides algorithms that off-load user supplied semantics. The whole issue is a contrast between mathematical modeling and user provided models. Each has specific advantages, and a responsible package should provide both. But I cannot agree that only one is sufficient. Think about 3D display compared to 3D modeling (with mathematical orientability). The user can see 3D in the first, but the second is required for manipulation.

Mathematical abstraction is a necessary and wonderful idea. It is, after all, the heart of programming, the core of modeling (and therefore understanding), and the source of power.