3D Behavioral Model Design for Simulation and Software Engineering*

Paul A. Fishwick

University of Florida

November 24, 1999

Abstract

Modeling is used to build structures that serve as surrogates for other objects. As children, we learn to model at a very young age. An object such as a small toy train teaches us about the structure and behavior of an actual train. VRML is a file standard for representing the structure of objects such as trains, while the behavior would be represented in a computer language such as ECMAScript or Java. VRML is an abbreviation for Virtual Reality Modeling Language [2], which represents the standard 3D language for the web. Our work is to extend the power of VRML so that it is used not only for defining shape models, but also for creating structures for behavior. "Behavior shapes" are built using metaphors mapped onto well-known dynamic model templates such as finite state machines, functional block models and Petri nets. The low level functionality of the design still requires a traditional programming language, but this level is hidden underneath a *modeling* level that is visualized by the user. We have constructed a methodology called *rube* which provides guidelines on building behavioral structures in VRML. The result of our endeavors has yielded a set of VRML Prototypes that serve as dynamic model templates. We demonstrate several examples of behaviors using primitive shape and architectural metaphors.

1 INTRODUCTION

One physical object captures some information about another object. If we think about our plastic toys, metal trains and even our sophisticated scale-based engineering models, we see a common thread: to build one object that says something about another—usually larger and more expensive—object. Let's call these objects the *source object* and the *target object*. Similar object definitions can be found in the literature of metaphors [6] and semiotics [10]. The source object *models* the target, and so, modeling represents a relation between objects. Often, the source object is termed *the model* of the target. We have been discussing scale models identified by the source and target having roughly proportional geometries. Scale-based models often suffer from the problem where changing the scale of a thing affects more than just the geometry. It also affects the fundamental laws applied at each scale. For example, the hydrodynamics of the scaled ocean model may be different than for the real ocean. Nevertheless, we can attempt to adjust for the scaling problems and proceed to understand the larger universe through a smaller, more manipulable, version.

Our task is to construct a software architecture that encourages 3D construction of objects and their models. Our focal point is the *behavioral model* where one defines the behavior or dynamics of an object using another set of objects. This sort of modeling may be used in representing the models of large-scale systems and software in the case where models have been used to specify the requirements and program design [1, 11]. We call this modeling architecture *rube*. An example use of *rube* is defined in the Sec. 2, and the methodology of *rube* in Sec. 3. Facets of the VRML implementation are defined in Secs. 4, 5 and 6. We close the paper with philosophical issues on the art of modeling in Sec. 7 and the summary in Sec. 8.

2 NEWELL'S TEAPOT

In the early days of computer graphics (c. 1974-75), Martin Newell rendered a unique set of Bézier surface spline patches for an ordinary teapot, which currently resides in the Computer Museum in Boston. The teapot was modeled by Jim Blinn and then rendered by Martin Newell and Ed Catmull at the University of Utah in 1974. While at this late date, the teapot may seem quaint, it has been used over the years as an icon of sorts, and more importantly as a benchmark for all variety of new techniques in rendering and modeling in computer graphics. The Teapot was recently an official emblem of the 25th anniversary

*Department of Computer & Information Science and Engineering, P.O. Box 116120, Gainesville, FL 32611, Email: fishwick@cise.ufl.edu

Figure 1: Office scene with Newell Teapot, dynamic model and props.

of the ACM Special Interest Interest Group on Computer Graphics (SIGGRAPH).

Since 1989 at the University of Florida, we have constructed a number of modeling and simulation packages documented in [4, 3] In late 1998, we started designing *rube*, named in dedication to Rube Goldberg [8], who produced many fanciful cartoon machines, all of which can be considered *models* of behavior. One of our goals for *rube* was to recognize that the Teapot could be used to generate another potential benchmark—one that captured the entire teapot, its contents and its models. The default teapot has no behavior and has no contents; it is an elegant piece of geometry but it requires more if we are to construct a fully *digital teapot* that captures a more complete set of knowledge. In its current state, the teapot is analogous to a building façade on a Hollywood film studio backlot; it has the shape but the whole entity is missing. In VRML, using the methodology previously defined, we built TeaWorld in Fig. 1. We have added extra props so that the teapot can be visualized, along with its behavioral model, in a reasonable contextual setting. The world is rendered in Fig. 1 using a web browser. *World* is the top-most root of the scene graph. It contains a *Clock*, *Boiling_System*, and other objects such as the desk, chairs, floor and walls. The key fields in Fig. 2 are VRML nodes of the relevant field so that the *contains* field refers to multiple nodes for its value. This is accomplished using the VRML *MFNode* type. The hierarchical VRML scene graph for Fig. 1 is illustrated in Fig. 2. The scene contains walls, a desk, chair and a floor for context. On the desk to the left is the teapot which is filled with water. The knob controlling whether the teapot heating element (not modeled) is on or off is located in front of the teapot. To the right of the teapot, there is a pipeline with three machines, each of which appears in Fig. 1 as a semi-transparent cube.

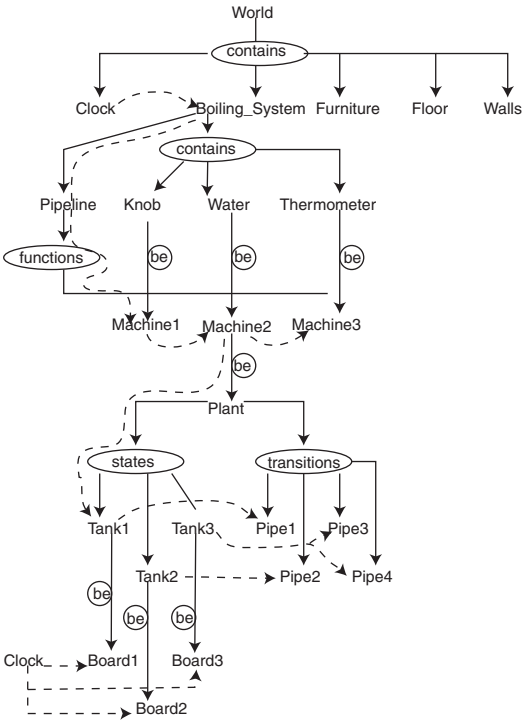

Figure 2: VRML Scene Graph for the Teapot and its models.

Each of these machines reflects the functional behavior of its encapsulating object: *Machine1* for *Knob*, *Machine2* for *Water* and *Machine3* for *Thermometer*. The *Thermometer* is a digital one that is positioned in *Machine3*, and is initialized to an arbitrary ambient temperature of 0° C. Inside *Machine2*, we find a more detailed description of the behavior of the water as it changes its temperature as a result of the knob turning. The plant inside *Machine2* consists of *Tank1*, *Tank2*, *Tank3*, and four pipes that move information from one tank to the next. Inside each tank, we find a blackboard on which is drawn a differential equation that defines the change in water temperature for that particular state. The following modeling relationships are used: *Pipeline* is a Functional Block Model (FBM), with three functions (i.e., machines); *Machine* is a function (i.e., semi-transparent cube) within an FBM; *Plant* is a Finite State Machine (FSM) inside of Machine 2; *Tank* is a state within a FSM, and represented by a red sphere; *Pipe* is a transition within a FSM, and represented by a green pipe with an attached cone denoting direction of control flow; and *Board* is a differential equation, represented as white text. The following metaphors are defined in this example. The three cubes represent a sequence of machines that create a pipeline. One could have easily chosen a factory floor sequence of numerically controlled machines from the web and then used this

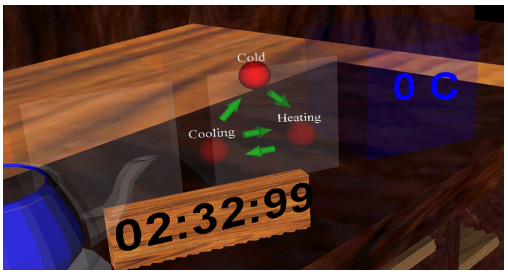

Figure 3: Pipeline closeup.

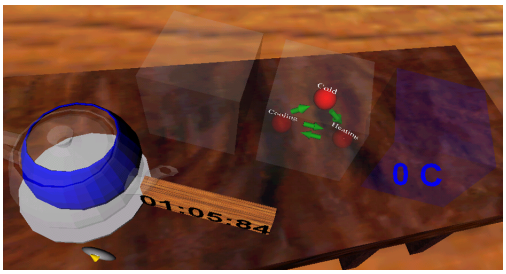

Figure 4: Cold state.

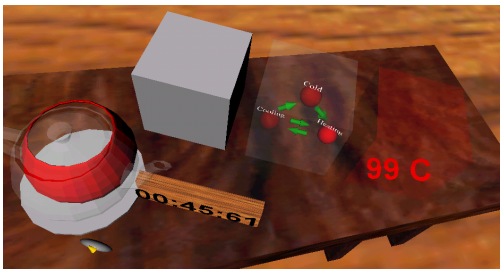

Figure 5: Heating state.

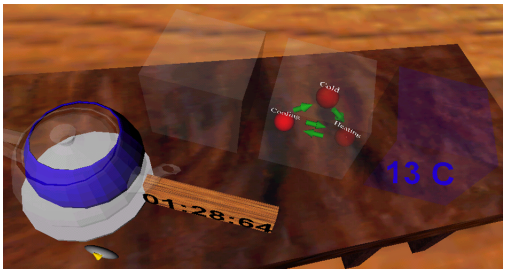

Figure 6: Cooling state.

in TeaWorld to capture the information flow. Inside the second machine, we find a plant, not unlike a petroleum refinery with tanks and pipes.

The *Pipeline* and its components represent physical objects that can be acquired from the web. For our example, we show simple objects but they have been given meaningful real-world application-oriented names to enforce the view that one object models another and that we can use the web for searching and using objects for radically different purposes than their proposed original function. The overriding concern with this exercise is to permit the modeler the freedom to choose *any* object to model *any* behavior. The challenge is to choose a set of objects that provide metaphors that are meaningful to the modeler. In many cases, it is essential that more than one individual understand the metaphorical mappings and so consensus must be reached during the process. Such consensus occurs routinely in science and in modeling when new modeling paradigms evolve. The purpose of *rube* is not to dictate one model type over another, but to allow the modelers freedom in creating their own model types. In this sense, *rube* can be considered a meta-level modeling methodology.

The simulation of the VRML scene shown in Fig. 2 proceeds using the dashed line thread that begins with the *Clock*. The clock has an internal time sensor that controls the VRML time. The thread corresponds closely with the routing structure built for this model. It starts at *Clock* and proceeds downward through all behavioral models. Within each behavioral model, routes exist to match the topology of the model. Therefore, *Machine1* sends information to *Machine2*, which accesses a lower level of abstraction and sends its output to *Machine3*, completing the semantics for the FBM. The FSM level contains routes from each state to its outgoing transitions.

Fig. 3 shows a closeup view of the pipeline, that represents the dynamics of the water, beginning with the effect of turning of the knob and ending with the thermometer that reads the water temperature.

Figs. 4 through 6 show the pipeline during simulation when the knob is turned ON and OFF at random times by the user. The default state is the cold state. When the knob is turned to the OFF position, the system moves into the heating state. When the knob is turned again back to the OFF position, the system moves into the cooling state and will stay there until the water reaches the ambient temperature at which time the system (through an internal state transition) returns to the cold state. Temperature change is indicated by the color of *Water* and *Machine3*, in addition to the reading on the *Thermometer* inside of *Machine3*. The material properties of *Machine1* change depending on the state of the knob. When turned to the OFF position, *Machine1* is semi-transparent. When turned on, it turns opaque. Inside *Machine2*, the current state of the water is reflected by the level of intensity of each *Plant*. The current state has an increased intensity, resulting in a bright red sphere.

The dynamics of temperature is indicated at two

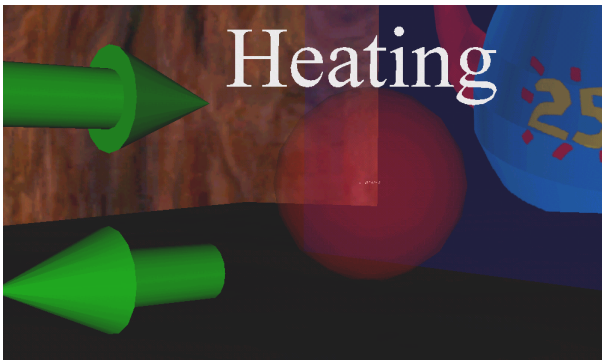

Figure 7: Outside the Heating phase.

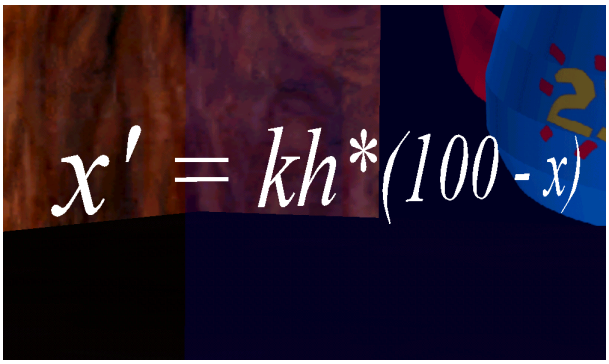

Figure 8: Inside the Heating phase.

levels. At the highest level of the plant, we have a three state FSM. Within each state, we have a differential equation. The equation is based on Newton's Law of Cooling and results in a first order exponential decay and rise that responds to the control input from the knob. The visual display of temperature change confirms this underlying dynamics since the user finds the temperature changing ever more slowly when heating to 100°C or cooling back to the ambient temperature. Fig. 7 displays a closeup of the heating phase from the outside, and Fig. 8 is a view from inside the red sphere modeling the phase.

3 *rube* Methodology

The procedure for creating models is defined as follows:

1. The user begins with an object that is to be modeled. This object can be primitive or complex—such as a scene–with many sub-objects. In the case of the teapot, we identify the teapot, water, knob, heating element as well as other aspects of the environment: room, walls, desk and floor. Not all of these objects require embedded behavioral models; some objects are props or exist for contextual reasons.

2. The *scene* and object interactions are sketched in a story board fashion, as if creating a movie or animation. A scene is where all objects, including those modeling others, are defined within a VRML file. The *rube* model browser is made available so that users can "fly though" an object to view its models without necessarily cluttering the scene with all objects. However, having some subset of the total set of models surfaced within a scene is also convenient for aesthetic reasons. The modeler may choose to build several scenes with models surfaced, or choose to view objects only through the model browser that hides all models as fields of VRML object nodes. In Fig. 1, the object *Pipeline* models the heating and cooling of the water, which is inside the teapot to the left of *Pipeline*. We could also have chosen to place a GUI behavioral model handle for the teapot *inside* the teapot itself or within close proximity of the teapot.

3. The shape and structure of all objects are modeled in any modeling package that has an export facility to VRML. Most packages, such as Kinetix 3DStudioMax and Autodesk AutoCAD have this capability. Moreover, packages such as CosmoWorlds and VRCreator can be used to directly create and debug VRML content. We used CosmoWorlds for the walls, floor and *Pipeline*. Other objects, such as the teapot, desk and chair were imported from the web.

4. VRML PROTO (i.e., prototype) nodes are created for each object, model and components thereof. This step allows one to create *semantic attachments* so that we can define one object to be a behavioral model of another (using a *behavior* field) or to say that the water is contained within the teapot. Without prototypes, the VRML file structure lacks semantic relations and one relies on simple grouping nodes, which are not sufficient for clearly defining how objects relate to one another. PROTOs are created for all physical objects, whether or not the objects are role-playing as a behavior model or as a behavior model component. This is discussed in more depth in Sec. 4.

5. Models are created. While multiple types of models exist, we have focused on dynamic models of components, and the expression of these components in 3D. Even textually-based models that must be visualized as mathematical expressions can be expressed using the VRML text

node. Models are objects in the scene that are no different structurally from pieces of visible objects being modeled—they have shape and structure. The only difference is that when an object is "modeling" another, one interprets the object's structure in a particular way, using a dynamic model template for guidance.

6. Several dynamic model templates exist. For Newell's Teapot (in Sec. 2), we used three: FBM, FSM, and EQN. These acronyms are defined as follows: FSM = Finite State Machine; FBM = Functional Block Model; EQN = Equation Set. Equations can be algebraic, ordinary differential, or partial differential. The FBM serves to capture the control flow from the activity of the knob to the temperature change of the water, and on to the thermometer. The FSM inside *Machine2* of *Pipeline* models the water temperature changes.

7. The creative modeling act is to choose a dynamic model template for object behavior, and then to pick objects that will convey the meaning of the template within the scenario. This part is a highly artistic enterprise since literally any object can be used. It is not the policy of *rube* to recommend or insist upon one metaphor. In practice, different groups will evolve and certain metaphors may compete in a process akin to natural selection. Our *Pipeline* could easily have been more artistically modeled so that it appeared more as a pipeline, and so the *Plant* looked more like an industrial plant. We were caught between trying to employ metaphor to its fullest extent and wanting those familiar with traditional 2D behavior models to follow the *rube* methodology.

8. There are three distinct types of roles played by modelers in *rube*. At the lowest level, there is the person creating the *model templates* (FSM,FBM,EQN,PETRI-NET). Each dynamic model template reflects an underlying system-theoretic model [5]. At the mid-level, the person uses an existing model template to create a *metaphor*. An industrial plant is an example of a manufacturing metaphor. At the highest level, a person is given a set of metaphors and can choose objects from the web to create a model. These levels allow modelers to work at the levels where they are comfortable. Reusability is created since one focuses on the level of interest.

9. The simulation proceeds by the modeler creating threads of control that pass events from one VRML node to another. This can be done in one of two ways: 1) using VRML Routes, or 2) using exposed fields that are accessed from other nodes. Method 1 is familiar to VRML authors and also has the advantage that routes that extend from one model component to an adjacent component (i.e., from one state to another or from one function to another) have a topological counterpart to the way we visualize information and control flow. The route defines the topology and data flow semantics for the simulation. Method 2 is similar to what we find in traditional object-oriented programming languages where information from one object is made available to another through an assignment statement that references outside objects and classes. Such an assignment is termed "message passing." In method 1, a thread that begins at the root node proceeds downward through each object that is role-playing the behavior of another. The routing thread activates Script nodes that are embedded in the structures that act as models or model components for the behaviors. All objects acting as behavioral model components are connected to a VRML clock (i.e., TimeSensor) so that multimodeling is made possible by allowing model components to gain control of the simulation and proceed in executing lower level model semantics.

10. Pre- and Post-processing is performed on the VRML file to check it for proper syntax and to aid the modeler. Pre-processing tools include wrappers (that create a single VRML file from several), decimators (that reduce the polygon count in a VRML file), and VRML parsers. The model browser mentioned earlier is a post-production tool, allowing the user to browse all physical objects to locate objects that model them. In the near future, we will extend the parser used by the browser to help semi-automate the building of script nodes. The browser and underlying VRML parser is based in Java (using JavaCUP) and therefore can be activated through the web browser.

rube treats all models in the same way. For a clarification of this remark, consider the traditional use of the word "Modeling" as used in everyday terms. A model is something that contains attributes of a target object, which it is modeling. Whereas, equation and 2D graph-based models could be viewed as being fundamentally different from a commonsense model, *rube* views them in exactly the same context: everything is an object with physical extent and modeling is a relation among objects. This unification is theoretically pleasing since it unifies what it means

to "model" regardless of model type. We are able to unify the commonsense view of modeling (i.e., scale or clay models) with more abstract modeling techniques used on the computer.

4 VRML IMPLEMENTATION OF THE TEAPOT

Newell's Teapot has the VRML scene graph structure as shown in Fig. 2, but there are also key prototype definitions used for objects and the models. The structure of the PROTOs are as follows. First, we have the PROTO for the dynamic model type FSM (Finite State Machine):

```
EXTERNPROTO FSM [
  eventIn      SFFloat   set_clock
  field        SFVec3f   position
  exposedField SFBool    input
  eventIn      SFString  set_state
  field        SFNode    start_state
  field        MFNode    sounds
  field        MFNode    states
  field        MFNode    transitions
  field        SFBool    passive
  field        MFNode    active
] "fsm.wrl#FSM"
```

The FSM is composed of `states` and `transitions`, with each state having a sound. There is a `start_state` and a `position` for placing the FSM in the scene. `set_clock` is the clock input and allows the FSM to take its `input` to drive the state transition changes.

Each FSM may be modeled at one of two levels: `active` and `passive`. The use of the 3 tanks and 4 pipes is passive since there is no motion–only a change in intensity of each tank when a state is enabled. An active mode implies the existence of two extra nodes, a `mover` and a `path`, both of which define the geometry associated with an object moving along a path. For example, if `active` has a field value of `[avatar24 spline3]` then node `avatar24` would make a motion along a physical path defined by node `spline3` to denote a change in state.

```
EXTERNPROTO FSM_STATE [
  eventIn      SFFloat  set_clock
  exposedField SFBool   enabled
  field        MFNode   audio
  exposedField MFNode   geometry
  field        MFNode   behavior
] "fsm.wrl#FSM_STATE"
```

Each state and transition has a geometry model and a behavior mode. The geometry model is that which defines how the object is to be rendered. The behavior model defines how the state is to be executed. `behavior` may terminate in a VRML Script node, but may also be further defined by another 3D structure to any abstraction level. Using `geometry`, we may allow any 3D scene or object to reflect the notion of state.

```
EXTERNPROTO FSM_TRANSITION [
  eventIn      SFFloat  set_clock
  exposedField SFBool   enabled
  eventOut     SFString state_changed
  field        SFNode   from
  field        SFNode   to
  field        SFNode   fsm
  field        SFNode   object
  exposedField MFNode   geometry
  field        MFString behavior
] "fsm.wrl#FSM_TRANSITION"
```

The transition nodes are similar in that one may assign both `geometry` and `behavior` to them. Each transition has `from` and `to` states. The transition also carries the behavior "logic" that determines whether a `state_change` occurs.

The metaphor elements are mapped directly to model templates, each of which is defined by a PROTO node:

- Industrial Plant metaphor → Finite state machine → FSM PROTO
- Tank (in Plant) metaphor → State → FSM-STATE PROTO
- Pipe (in Plant) metaphor → Transition → FSM-TRANSITION PROTO

5 PROGRAMMING USING VRML

Object-Oriented Software engineering has long advocated the use of modeling in defining the creation of software. A recent example is the significant interest in the Unified Modeling Language (UML) [12, 9]. The embedded nature of software encourages the modeling approach since the design reflects the distributed nature of the hardware components. Software engineers evolve into system modelers. Using VRML, we created a small operating system kernel that involves metaphors for tasks (using avatars), routes through the system (using colored paths) and resources (using office desks with attendants). The overall operating system is shown as an office building in Fig. 9

and inside the building there is a floor designated as the kernel (Fig. 10). Tasks begin in a waiting

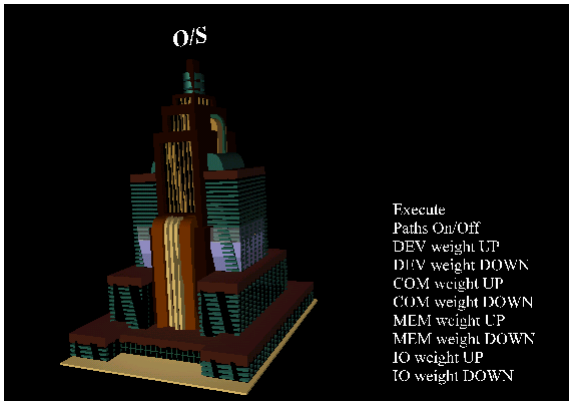

Figure 9: The operating system structure.

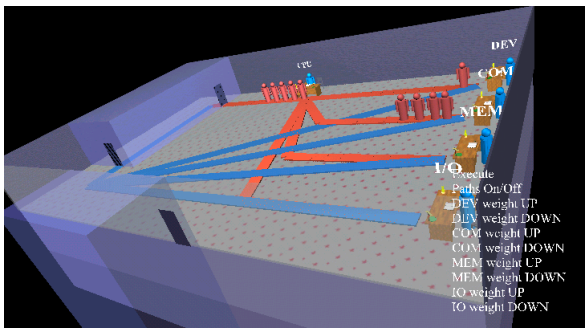

Figure 10: Inside the O/S kernel.

area on the left side of Fig. 10 and proceed to the CPU desk. Tasks continue to move toward the key resources (DEVice, COMmunications,MEMory,I/O). The paths to the resources are in orange and the return paths back to the holding area are blue. When a resource is requested, this begins an audio track specifying which resource is being used.

6 VRML ISSUES

We found VRML to be fairly robust for the task set side within *rube* but there are a number of issues that need to be resolved in the future. The most serious issues, having an effect on *rube*, are delineated:

1. VRML needs a strong object-oriented (OO) structure with classes, inheritance, object instances and a capability for referring to a current object with a *this* field. We found it difficult to create a regional scoping of public variables, for example, in allowing a component to gain access to its parents fields. Instead, one has to expose a field to every other node. One side-effect of a solid OO architecture would be that there would be a distinct difference between defining a Node and creating one. The existing `DEF` both defines and creates.

2. Exposed fields should operate exactly as if one were to specify an `eventIn`, `field` and `eventOut`. Several VRML texts suggest an equivalence, but in practice they are quite different. Currently, if one creates an `exposedField`, then one cannot define an `eventIn` for setting values in a script node using a function of the `eventIn` name. There are many instances where it would be useful to use both methods of access to an exposedField: 1) directly with `node.set_field`, or 2) indirectly with a route using `set_field` as an `eventIn` within a script node. Routes are useful in surfacing connections between one node and another, but prudent use of `exposedField`s (if they were more completely implemented) simplifies a spaghetti-like network of routes.

3. Both forward and backward references to nodes should be possible. Currently, one cannot specify `USE somenode` unless it has already been defined with `DEF`. This may require a multi-pass option on parsing the VRML scene graph, which would slow the parsing but give the VRML author the freedom to choose forward referencing where they may wish to implement it.

4. Scripting capabilities need to be expanded to support native code, and need to be consistent among browsers in supporting Java in the Script node and full implementations of Javascript. Instead of being implementor options, they should be required. Native code is essential if VRML is to both *compete with*, *expand upon* and *reuse* other 3D software.

5. We found PROTO and EXTERNPROTO support to be variable among VRML software developers. Since these are among the most important node types in VRML, their implementations should be ubiquitous in all VRML modelers and software support.

7 ART OF MODELING

Given the Newell Teapot scene, there are some key issues which we should ask ourselves:

- *Is it "just" a visualization?* The work in *rube* provides visualization, but models such as

Newell's Teapot demonstrate active modeling environments whose existence serves an engineering purpose and not only a post-project visualization purpose for outside visitors. This sort of modeling environment is needed from the very start of a mission—as an integral piece of the puzzle known as model design. There is little question that this sort of production is useful for teaching purposes, but we also view this as a precursor to next generation software engineering. The power of VRML in this regard is that it can be used to reinvent software engineering through the surfacing of 3D models. It is one thing to think of this as a *visualization* of an Operating System kernel, but it is quite another to call it the Operating System *itself*. We need to bridge this gap if we are to progress beyond textual, linear programming styles.

- *Is it economical?* Is this a lot of work just to create an FSM? All 3D objects are reused and so can be easily grabbed from the web. The concept of reuse is paramount to the *rube* approach where the metaphor can be freely chosen and implemented. Without the web, *rube* would not be possible. 3D object placement can be just as economical as 2D object placement, but object repositories are required.

- *What is the advantage?* If we consider psychological factors, the 3D metaphor has significant advantages. First, 3D spatially-specific areas serve to improve our memory of the models (i.e., mnemonics). Second, graphical user interfaces (GUIs) have shown that a human's interaction with the computer is dramatically improved when the right metaphors are made available. *rube* provides the environment for building metaphors. One should always be wary of mixed metaphors. We leave the ultimate decision to the user group as to which metaphors are effective. A Darwinian-style of evolution will likely determine which metaphors are useful and which are not. Aesthetics plays an important role here as well. If a modeler uses aesthetically appealing models and metaphors, the modeler will enjoy the work. It is a misconception to imagine that only the general populous will benefit from fully interactive 3D models. The engineers and scientist need this sort of immersion as well so that they can understand better what they are doing, and so that collaboration is made possible.

- *Is this art or science?* The role of the Fine Arts in science needs strengthening. With fully immersive models, we find that we are in need of workers with hybrid engineering/art backgrounds. It is no longer sufficient to always think "in the abstract" about modeling. Effective modeling requires meaningful human interaction with 3D objects. So far, the thin veneer of a scale model has made its way into our engineering practices, but when the skin is peeled back, we find highly abstract code and text. If the internals are to be made comprehensible (by anyone, most importantly the engineer), they must be surfaced into 3D using the powerful capabilities of metaphors [7, 6]. This doesn't mean that we will not have a low level code-base. Two-dimensional metaphors and code constructs can be mixed within the 3D worlds, just as we find them in our everyday environments with the embedding of signs. At the University of Florida, we have started a *Digital Arts and Sciences* Program with the aim to produce engineers with a more integrated background. This background will help to produce new workers with creative modeling backgrounds.

- *What role does aesthetics play in modeling?* It is sometimes difficult to differentiate models used for the creation of pieces of art from those used with scientific purposes in mind. Models used for science are predicated on the notion that the modeling relation is unambiguously specified and made openly available to other scientists. Modeling communities generally form and evolve while stressing their metaphors. In a very general sense, natural languages have a similar evolution. The purpose of art, on the other hand, is to permit some ambiguity with the hopes of causing the viewer or listener to reflect upon the modeled world. Some of the components in worlds such as Fig. 1 could be considered non-essential modeling elements that serve to confuse the scientist. However, these elements may contribute to a more pleasing immersive environment. Should they be removed or should we add additional elements to please the eye of the beholder? In *rube*, we have the freedom to go in both directions, and it isn't clear which inclusions or eliminations are appropriate since it is entirely up to the modeler or a larger modeling community. One can build an entirely two dimensional world on a blackboard using box and text objects, although this would not be in the spirit of creating immersive worlds that allow perusal of objects and their models.

  It may be that a select number of modelers may find the TeaWorld room exciting and pleasing, and so is this pleasure counterproductive to the scientist or should the scientist be concerned only

with the bare essentials necessary for unambiguous representation and communication? Visual models do not represent *syntactic sugar* (a term common in the Computer Science community). Instead, these models and their metaphors are essential for human understanding and comprehension. If this comprehension is complemented with a feeling of excitement about modeling, this can only be for the better. Taken to the extreme, a purely artistic piece may be one that is so couched in metaphor that the roles played by objects isn't clear. We can, therefore, imagine a kind of continuum from a completely unambiguous representation and one where the roles are not published. Between these two extremes, there is a lot of breathing space. Science can be seen as a form of consensual art where everyone tells each other what one object *means*. Agreement ensues within a community and then there is a mass convergence towards one metaphor in favor of another.

8 SUMMARY

Effort to unify behavioral and software engineering modeling methodologies are useful, but we should also have a way to express models more creatively and completely. Model communities will naturally evolve around 2D and 3D metaphors yet to be determined. *rube* has a strong tie to the World Wide Web (WWW). The web has introduced a remarkable transformation in every area of business, industry, science and engineering. It offers a way of sharing and presenting multimedia information to a worldwide set of interactive participants. Therefore any technology tied to the web's development is likely to change modeling and simulation. The tremendous interest in Java for doing simulation has taken a firm hold within the simulation field. Apart from being a good programming language, its future is intrinsically bound to the coding and interaction within a browser. VRML, and its X3D successor, represent the future of 3D immersive environments on the web. We feel that by building a modeling environment in VRML and by couching this environment within standard VRML content, that we will create a *Trojan Horse* for simulation modeling that allows modelers to create, share and reuse VRML files.

Our modeling approach takes a substantial departure from existing approaches in that the modeling environment and the material object environment are merged seamlessly into a single environment. There isn't a difference between a circle and a house, or a sphere and a teapot. Furthermore, objects can take on any role, liberating the modeler to choose whatever metaphor that can be agreed upon by a certain community. There is no single syntax or structure for modeling. Modeling is both an art and a science; the realization that all objects can play roles takes us back to childhood. We are building *rube* in the hope that by making all objects virtual that we can return to free-form modeling of every kind. Modeling in 3D can be cumbersome and can take considerable patience due to the inherent user-interface problems when working in 3D using a 2D screen interface. A short term solution to this problem is to develop a model package that is geared specifically to using one or more metaphors, making the insertion of, say, the petroleum refinery a drag and drop operation. Currently, a general purpose modeling package must be used to carefully position all objects in their respective locations. A longer term solution can be found in the community of virtual interfaces. A good immersive interface will make 3D object positioning and connections a much easier task than it is today.

ACKNOWLEDGMENTS

We would like to thank the students on the *rube* Project: Robert Cubert, Andrew Reddish, John Hopkins and Linda Dance. Also, we thank the following agencies that have contributed towards our study of modeling and simulation: (1) Jet Propulsion Laboratory under contract 961427 *An Assessment and Design Recommendation for Object-Oriented Physical System Modeling at JPL* (John Peterson, Stephen Wall and Bill McLaughlin); (2) Rome Laboratory, Griffiss Air Force Base under contract F30602-98-C-0269 *A Web-Based Model Repository for Reusing and Sharing Physical Object Components* (Al Sisti and Steve Farr); and (3) Department of the Interior under grant 14-45-0009-1544-154 *Modeling Approaches & Empirical Studies Supporting ATLSS for the Everglades* (Don DeAngelis and Ronnie Best). We are grateful for their continued financial support.

References

[1] Grady Booch. *Object Oriented Design*. Benjamin Cummings, 1991.

[2] Rikk Carey and Gavin Bell. *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley, 1997.

[3] Robert M. Cubert and Paul A. Fishwick. MOOSE: An Object-Oriented Multimodeling and Simulation Application Framework. *Simulation*, 70(6):379–395, 1998.

[4] Paul A. Fishwick. Simpack: Getting Started with Simulation Programming in C and C++. In *1992 Winter Simulation Conference*, pages 154–162, Arlington, VA, 1992.

[5] Paul A. Fishwick. *Simulation Model Design and Execution: Building Digital Worlds*. Prentice Hall, 1995.

[6] George Lakoff. *Women, Fire and Dangerous Things: what categories reveal about the mind*. University of Chicago Press, 1987.

[7] George Lakoff and Mark Johnson. *Metaphors We Live By*. University of Chicago Press, 1980.

[8] Peter C. Marzio. *Rube Goldberg, His Life and Work*. Harper and Row, New York, 1973.

[9] Pierre-Alain Muller. *Instant UML*. Wrox Press, Ltd., Olton, Birmingham, England, 1997.

[10] Winfried Noth. *Handbook of Semiotics*. Indiana University Press, 1990.

[11] James Rumbaugh, Michael Blaha, William Premerlani, Eddy Frederick, and William Lorenson. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[12] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Reading, MA, 1999.

AUTHOR BIOGRAPHY

PAUL FISHWICK is Professor of Computer and Information Science and Engineering at the University of Florida. He received the PhD in Computer and Information Science from the University of Pennsylvania in 1986. His research interests are in computer simulation, modeling, and animation, and he is a Fellow of the Society for Computer Simulation (SCS). Dr. Fishwick will serve as General Chair for WSC00 in Orlando, Florida. He has authored one textbook, co-edited three books and published over 100 technical papers.