

Virtual Reality Modeling Language Templates for Dynamic Model Construction

T. Kim^a and P. Fishwick^b, U. of Florida

Computer and Information Science and Engineering Department
University of Florida, Gainesville, FL 32611

ABSTRACT

The use of our *rube* Methodology permits the design of computing models that are framed using user-specified aesthetics. For example, we can design a functional block model using 3D blocks and pipes, or alternatively, using rooms and portals. Our recent approach has been to construct dynamic models using the Virtual Reality Modeling Language (VRML) by taking familiar 2D icons, and subsequently mapping these to 3D primitives. We discuss two specific transformations from 2D to 3D within the context of finite state automata and functional block models. These examples are simple and serve as tutorial examples of how more complex transforms can be accomplished.

Keywords: *rube* Methodology, VRML template, 3D, dynamic modeling, aesthetic programming

1. INTRODUCTION

The rapid growth in technologies enables us to incorporate our aesthetic sensibilities within our model constructions. Our goal is to encourage modelers to maximize their user-oriented aesthetics in designing dynamic models. We show examples of dynamic models created by implementing the *rube* Methodology and by using VRML, which is currently our primary language for building the models.

Since 1989 at the University of Florida, Fishwick and his students have constructed a number of modeling and simulation packages such as a set of C programs called SimPack [1] and OOPM (Object Oriented Physical Modeler) [2]. In late 1998, the *rube* Methodology was formulated, and which focuses on the use of 3D web-based virtual world construction models [3]. One goal is to construct a software architecture that encourages 3D construction of objects and their models focusing on the behavioral model where one defines the behavior or dynamics of an object using another set of objects [4].

With our *rube* Methodology, user can design dynamic computing models that incorporate user-specified aesthetics. For instance, a functional block model could be designed by using 3D blocks and pipes, or alternatively, using rooms and portals. In our recent approach, the Virtual Reality Modeling Language (VRML) [5] has been used to construct dynamic models by mapping familiar 2D icons to the corresponding 3D primitives. VRML, the standard 3D language for the web, serves as a tool for the *rube* Methodology to create geometries and dynamic behaviors of objects in a model. Several dynamic model templates have been constructed in order to support the *rube* methodology [6]. These templates have been provided through the internet as the building blocks for creating dynamic models which incorporate personal aesthetics of individual modelers.

In this paper, we discuss two specific transformations from 2D to 3D within the context of finite state automata (FSA) and functional block models (FBM), which serve as dynamic model templates. These simple examples of model templates serve as tutorial guidelines on building behavioral structures in VRML and on how more complex dynamic model buildings can be accomplished. These dynamic modeling templates were provided on the internet as the building blocks to create metaphors which then can be used to create more complicated dynamic models. Some of the related topic is addressed in section 2. In section 3, the two transformations and the example use of these model templates are discussed, along with several related issues. Conclusions and future work are discussed in section 4.

^a E-mail: twkim@cise.ufl.edu

^b E-mail: fishwick@cise.ufl.edu

2. BACKGROUND

Dynamic Model Types

Before we discuss specific modeling methods and start modeling dynamic behaviors of an object, let us overview a unifying formalism that serves to represent a wide variety of system models. A deterministic system $\langle T, U, Y, Q, \mathbf{W}, \mathbf{d}, \mathbf{l} \rangle$ within classical systems theory is defined as follows [7]:

- T is the time set. For continuous systems, $T = \mathbb{R}$ (reals), and for discrete time systems, $T = \mathbb{Z}$ (integers).
- U is the input set containing the possible values of the input to the system .
- Y is the output set.
- Q is the state set.
- \mathbf{W} is the set of admissible (or acceptable) input functions. This contains a set of input functions to use during system operation. Often, due to physical limitations, \mathbf{W} is a subset of the set of all possible input functions ($T \rightarrow U$).
- \mathbf{d} is the transition function. It is defined as $\mathbf{d} : Q \times T \times T \times \mathbf{W} \rightarrow Q$.
- \mathbf{l} is the output function, $\mathbf{l} : Q \rightarrow Y$.

A pair consists of a time and a state (t, s) , where $s \in Q$ is called an *event*. Events are points in event space just as states are points in state space. Event space is defined as $T \times Q$. State and Event are critical aspects of any system and by focusing on one or the other, we form two different sorts of models: *declarative* models that focus on the concept of state, and *functional* models that focus on the concept of event. In declarative modeling, as shown in Figure 1, in 2D diagram, we build models that focus on state representations and state-to-state transitions. In functional modeling as shown in Figure 2, we focus on the system as a coupled network of functions each of which takes inputs and produces outputs.

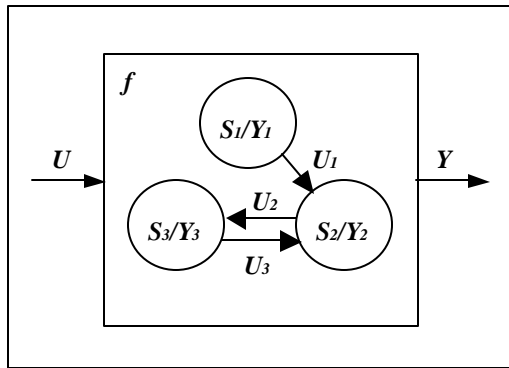


Figure 1: Declarative Model

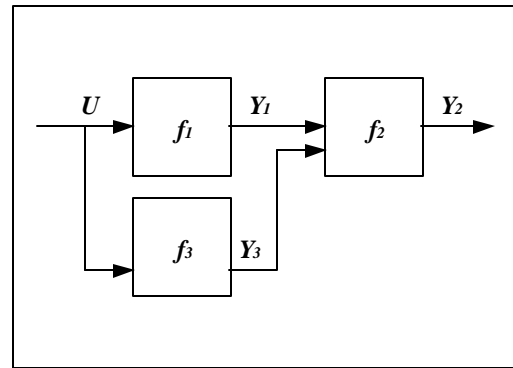


Figure 2: Functional Model

The declarative modeling approach suggests that we look at the world through a sequence of changes in state and it is very good for modeling problem domains where the problem decomposes into discrete temporal phases. Finite State Automata (FSA) permit us to model systems with a minimum of components. An FSA is a system with states and transitions. A state, depicted as a circle, represents the current condition of a system for some length of time and transitions, depicted as arrows, enable the system to move from one state to another during the simulation. The basic definition of system $\langle T, U, Y, Q, \mathbf{W}, \mathbf{d}, \mathbf{l} \rangle$ provides the semantics for declarative models as described above for the deterministic system.

Functional models map very easily into the OO paradigm since a function is represented as a method located within an object and it is often used to model continuous systems and discrete event systems composed of coupled functional blocks. These functional models are also useful when the problem is given in terms of distinct physical objects which are connected in a directed order or the problem involves a material flow throughout the system. Function-based models contain a transfer function which relates input to output. The functions, along with inputs and outputs, are often depicted in a “block” form, especially when a block is the iconic representation of physical device being modeled.

VRML

VRML, which is an abbreviation for Virtual Reality Modeling Language, represents the standard 3D language for the web. To be precise, VRML is not a modeling language but an effective 3D file interchange format, a 3D analog to HTML. 3D objects and worlds are described in VRML files using a hierarchical scene graph, which is composed of entities called nodes. Nodes can contain other nodes and this scene graph structure makes it easy to create complex, hierarchical systems from subparts. VRML also defines an event or message-passing mechanism by which nodes in the scene graph can communicate with each other and Script nodes can be inserted between event generators and event receivers. Script nodes allow the world creator to define arbitrary behaviors, defined in any supported scripting languages, such as JavaScript [5].

Even though VRML is not a full-fledged programming language, some of its features make it a useful tool for modeling dynamic systems in 3D environment. These features include the ability to create user prototype nodes that can be reused in various models and the ability to incorporate scripts inside these custom nodes, which enable objects to perform various functional behaviors. This addition of behavior information to 3D objects in VRML files and the fact that created worlds using VRML are freely accessible over the internet make VRML a robust tool for our *rube* Methodology.

A *template* is defined by a collection of prototypes, along with an example, to enable efficient instantiation of specific dynamic models. For example, the FSM template includes the following key VRML PROTO structures: FSM, FSM-STATE, and FSM-TRANSITION.

***rube* Methodology**

The procedure for creating models in *rube* Methodology is defined using the following steps [4]:

1. *Choose an object that is to be modeled.* This object can be primitive or complex – such as a scene with many sub-objects.
2. *Sketch the scene and object interactions* in a storyboard fashion, as if creating a movie or animation. A scene is where all objects, including those modeling others, are defined within a VRML file.
3. *Model the shape and structure of all objects* in any modeling package that has an export facility to VRML. Packages such as CosmoWorlds and VRCreator can be used to directly create and debug VRML content.
4. *Create VRML PROTO (i.e., prototype) nodes* 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.
5. *Create models.* While multiple types of models exist, focus on dynamic models of components, and the expression of these components in 3D.
6. *Choose appropriate dynamic model templates*, such as FSM (Finite State Machine), FBM (Functional Block Model) or EQN (Equation set), 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.
7. *Choose the type of role needed.* 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 [7]. At the mid-level, the person uses an existing model template to create a 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.
8. *Proceed the simulation by creating threads of control* that pass events from one VRML node to another. This can be done by using VRML Routes or using exposed fields that are accessed from other nodes. The route defines the topology and data flow semantics for the simulation. The route thread activates Script nodes that are embedded in the structures that act as models or model components for the behaviors.
9. *Perform Pre- and Post-processing 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 parser.

To support our *rube* Methodology from step 3 to step 7, some of the primitive dynamic modeling templates and examples were created and provided as a repository of dynamic modeling building blocks on the internet. The tutorial web page [8] provides several dynamic modeling templates which include FSM, FBM, SD (System Dynamics), PN (Petri Net) and the mover and path PROTOs. Example world for each template and source code are provided with a brief description. We focus on FSM and FBM modeling templates presented in the tutorial web page that serve as building blocks in creating dynamic models.

3. 3D DYNAMIC MODELING USING VRML TEMPLATES

As shown in Figures 1 and 2, declarative and functional dynamic models can be represented in 2D graphics. By implementing the *rube* Methodology, we could map these 2D models into 3D model templates created in VRML. We will start from constructing the FSM model template and move on to the FBM. Figure 3 shows a sample 2D FSM that has three states and three transitions. The 2D FSM can be transformed into a 3D FSM as shown in Figure 4 by following the steps 1 through 5 in *rube* Methodology.

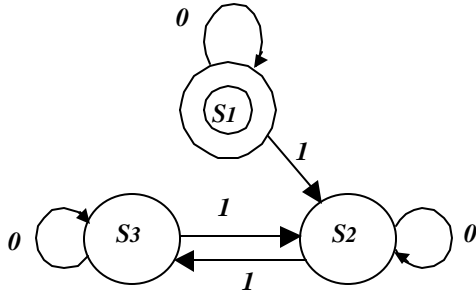


Figure 3: 2D FSM Model

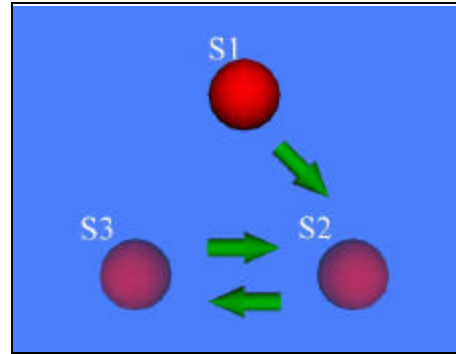


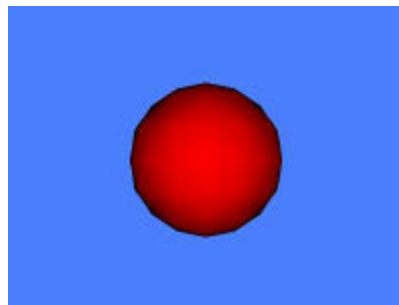
Figure 4: 3D FSM Model

To create 3D dynamic template for FSM which has three states and three transitions (*step 1*), a scene and object interactions were sketched on a paper by representing the circles and arrows in 2D FSM with spheres and 3D arrows (*step 2*). The idea was to represent the behavior of the system or the state transitions through the intensity changes in color of each sphere. Initially, state S1 is highlighted and when an event occurs which is denoted by 1, S1 is dimmed and S2 is highlighted. In this case, mouse click on any objects is considered as an event. State of the system changes back and forth between S2 and S3 as event occurs. The implemented logic is as follows: (S1,0)->S1; (S1,1)->S2; (S2,0)->S2; (S2,1)->S3; (S3,0)->S3; (S3,1)->S2. CosmoWorlds was used to model the shape and structure of all objects (*step 3*). First, a simple world with a red sphere was created in VRML as shown in Figures 5(a) and (b), and then a world with three spheres with label and three arrows was created without using any PROTO nodes for the tutorial purposes. PROTO node structures for the objects such as sphere and arrow were created (*step 4*) as shown in Figures 5(c). To focus on the implementation of the behaviors of the model, other PROTOs such as FSM, FSM_STATE, and FSM_TRANSITION which define the behaviors of each component have been created (*step 5*) as shown in Figures 6(a), (b), and (c). Following the steps leads us to the FSM dynamic template in VRML, which displays the dynamic behaviors of the FSM model by interacting with the user.

As described earlier, a VRML world is made up of nodes, which contain fields. Nodes can contain other nodes as fields to create a hierarchical structure. In a simple world shown in Figure 5(a), **Transform** contains **Shape**, which contains two fields called **appearance** and **geometry**. **Appearance**, which specifies the visual attributes, contains **Material** where each attribute fields are defined such as **diffuseColor** and **transparency**. A VRML world viewed in Cortona VRML Client 3.0 is shown in Figure 5(b). PROTO node defines structure for its interface, i.e. how the prototype communicates with the rest of the world and what parameters may be set for each instance of the prototype as shown in Figure 5(c).

```
# VRML V2.0 utf8
# World with a Red Sphere
Transform {
  children
  Shape {
    appearance Appearance {
      material Material {
        diffuseColor 1 0 0
        transparency 0}
    }
    geometry Sphere {}
  } #Shape
} #Transform
```

(a) A Sphere



(b) A VRML world with a Sphere

```
EXTERNPROTO SPHERE [
  exposedField SFFloat transparency
  field SFVec3f text_position
  field MFString name
  field SFVec3f position
] "protos2.wrl#SPHERE"

EXTERNPROTO ARROW [
  field SFVec3f local_frame
  field SFVec3f translation
  field SFRotation rotation
  field SFVec3f cylinder_translation
  field SFVec3f cone_translation
  field SFRotation cone_rotation
] "protos2.wrl#ARROW"
```

(c) Sphere and Arrow

Figure 5: Static VRML world and PROTO structures for FSM Model

VRML PROTO nodes can be used as a simple method for defining a library of reusable objects that can be reused in a world as shown in Figure 5(c) or as a method that defines a new node type in terms of other predefined nodes so that it allows geometry and behavior to be nicely packaged and give semantic relations to the VRML file structures. The structure of the PROTOs for the dynamic model type FSM is shown in Figures 6(a), (b), and (c).

<pre> EXTERNPROTO FSM [eventIn SFFloat set_clock field SFVec3f position exposedField SFBool input eventIn SFString set_state field SFString start_state field MFNode states field MFNode transitions] "protos3.wrl#FSM" </pre>	<pre> EXTERNPROTO FSM_STATE [eventIn SFFloat set_clock exposedField SFBool enabled field SFNode node field SFNode nodeG field MFNode audio exposedField MFNode geometry field MFString behavior] "protos3.wrl#FSM_STATE" </pre>	<pre> 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] "protos3.wrl#FSM_TRANSITION" </pre>
(a) FSM	(b) FSM_STATE	(c) FSM_TRANSITION

Figure 6: Dynamic VRML PROTO structures for FSM Model

start_state and **position** in the Figure 6(a) are used for placing the **FSM** in the scene and **set_clock** is used to drive the state transition changes. This **FSM** is modeled in passive mode since there is no motion but only a change in intensity of sphere when a state is enabled. An active mode modeling will be explained in the FBM model example. Each **FSM_STATE** and **FSM_TRANSITION** has **geometry** and **behavior** as shown in Figures 6(b) and (c). **behavior** defines how the state is to be executed. **geometry** defines how the object is to be rendered and it is **geometry** that allows user to implement any 3D object to represent the state. Thus, the user has freedom to use any geometry that is pleasing and aesthetic to reflect the notion of state. Each transition has **from** and **to** states and it also carries the behavior logic that decides whether **state_change** occurs.

The FSM dynamic model template created in VRML was presented to a group of students on a web page as a part of the courses taught at the University of Florida and let the students to create their own dynamic models using the template. The purpose of the assignment was to let the students learn the dynamic behaviors of model types via creating them with the templates provided on the internet. Each students applied their own aesthetic objects which was pleasing to them and developed various FSM models. Two of the example worlds created by the students are shown in Figures 7(a) and (b). The FSM model shown in Figure 7(a) was created by Donahue where each tank represents a state, and each set of pipes from one tank to another represents a possible transition from one state to another. Clicking the left mouse button over the model signifies an input. The water emptying from one tank to another illustrates which transition is activated by the input. Another FSM model shown in Figure 7(b) was created by Kohareswaran, a student in simulation class, with the following remarks:

"I started with the model in Dr. Fishwick's website. I first went through the code and understood the whole code and planned out what to do next. Then I thought of a metaphor or the FSM. I came up with the model of gazebos, with a person walking through them. The gazebo represents the states and the path between them the transition, while the direction of movement of the lady represents the direction of the state transition. The presence of the lady in the gazebo and the glowing of the light in the gazebo indicates that the state is active."

By implementing the creative ideas of the students to the same FSM dynamic template, intriguing 3D FSM dynamic models were created that are more pleasing and familiar to our sensory systems than those 2D diagrams.

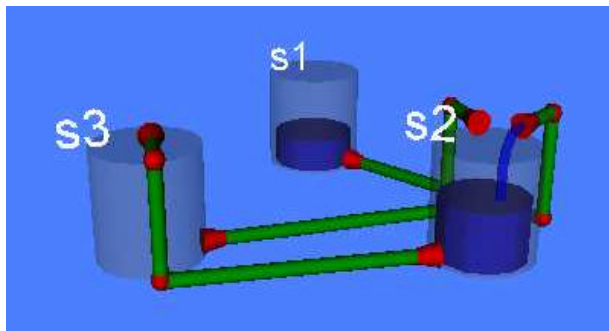


Figure 7(a): FSM 1 (Ryan Donahue)

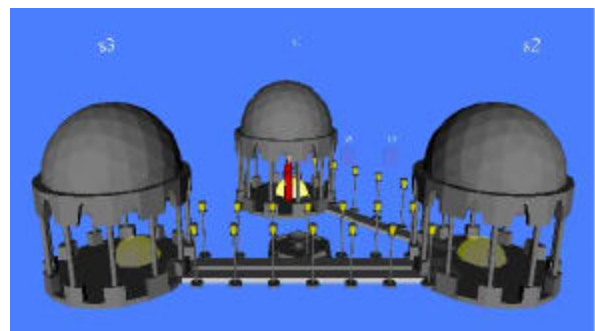


Figure 7(b): FSM 2 (Naganandhini Kohareswaran)

Now, let us turn our attention to the FBM dynamic models. Figure 8 shows a sample of 2D FBM Model that has three functions with one input and one output for each function. The 2D FBM Model can also be transformed into a 3D FBM dynamic Model as shown in Figure 9 by following the steps 1 through 5 in *rube* Methodology as we did for creating the FSM dynamic model template. Since FBM models contain a transfer function, which relates input to output, we need to implement this function within our FBM model template along with 3D representation of each function.

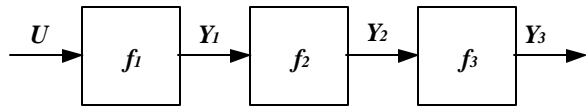


Figure 8: 2D FBM Model

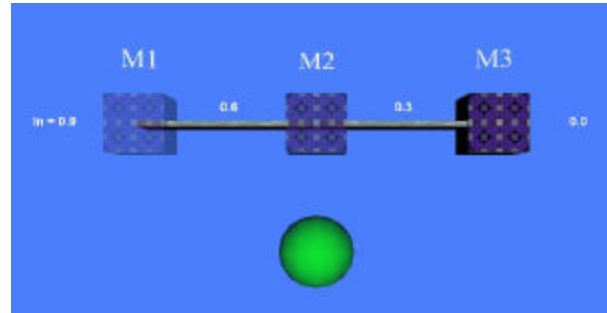
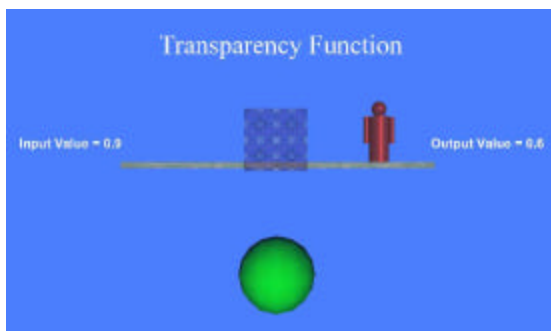


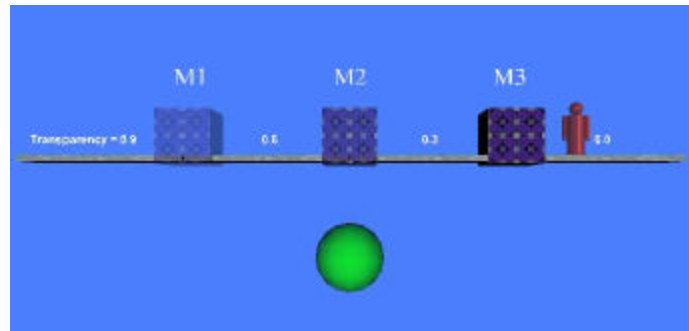
Figure 9: 3D FBM Model

In Figure 9, the transparency value of an object, which we used as an input data, is fed into the first function. Here, a function is represented as a box. Initial value 0.9 was put into the first function, which outputs the 0.6 as a result. The transparency value decreases by 0.3 as each function is applied to it. Initially, the transparency of each box is set to the input transparency value and the change of the value is displayed through the box as the transparency value is being processed gradually. The result value is then fed into the next function and so on. The transparency value is represented as the following function: $f(0) = 0.9$; $f(t+\Delta t) = f(t) - 0.3$. For the 3D FBM dynamic model template, we started out with a single function that has one input and one output as shown in Figure 10(a) and used it as a building block to expand the model. Also, this model template was created in an active mode which implies the existence of two extra nodes, a *mover* and a *path* that define the geometry associated with an object moving along a path as shown in Figure 10(b).

The idea for the FBM model template was to visualize the behavior of the function and the transition of the input and output data from one function to the other through the changes of transparency value of the box and the animation of a mover.



(a) FBM with one function



(b) FBM with mover and path

Figure 10: 3D FBM dynamic model templates

To create the 3D FBM dynamic model template which has three functions (*step 1*), a scene with three boxes and behavior of each boxes along with a mover were sketched on a paper (*step 2*). CosmoWorlds was used to create the boxes and traces in the world (*step 3*). To visualize the behavior of the functions, PROTO node structures for **FBM**, **FBM_TRACE**, and **FBM_FUNCTION** were created as shown in Figures 11(a) and (b). To visualize the transition of the input and output data from one function to the other, **MFACTORY** PROTO created by Cubert was reused to create and animate a mover to move along the paths between each function denoted as M1, M2, and M3 (*step 4*). The **MFACTORY** PROTO node structure is shown in Figure 11(c). With proper definitions of ROUTE, which passes a message from one node to the other when an event occurs, the FBM dynamic template in VRML is created (*step 5*).

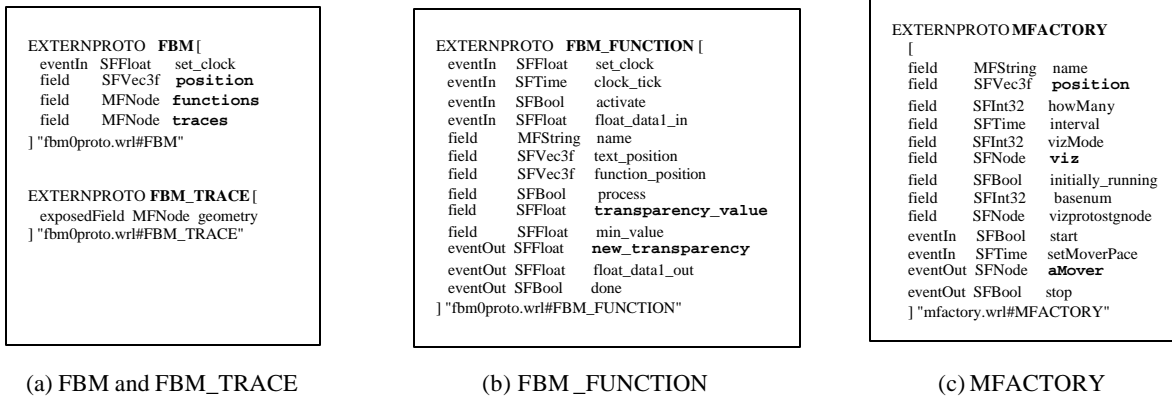


Figure 11: Dynamic VRML PROTOstructures for FBM Model template

position in Figure 11(a) is used for placing the **FBM** in the scene. Unlike the **FSM** model, this **FBM** model is created in active mode so that the transition of the data has been represented along with the behavior of the function. As we can see in Figure 11(a), **FBM_FUNCTION** PROTO and **FBM_TRACE** PROTO are incorporated in **FBM** PROTO as **functions** and **traces** to represent geometry and behavior of **FBM**. **transparency_value** and **new_transparency** in Figure 11(b) are used to set the initial transparency value of the box and gradually display the changes of transparency as it gets processed. With **MFACTORY** PROTO as shown in Figure 11(c), one can create a mover, denoted as **aMover**, with any user defined shape or geometry, denoted as **viz**. A mover can be placed at **position** in the world and can be animated along the path defined by the user.

Two specific examples of FSM and FBM transformations from 2D to 3D have been discussed by implementing the *rube* Methodology and VRML. These examples show how to build behavioral structures in VRML with user aesthetics. More complex 3D dynamic model structures can be constructed by re-using these model templates as building blocks.

4. CONCLUSIONS

Our work is to extend the power of VRML so that it could be used not only for defining shape models, but also for creating structures for behavior. Several VRML dynamic modeling prototypes have been constructed by using the *rube* Methodology, which provides guidelines on building behavioral structures in VRML. These 3D dynamic model examples are simple and serve as tutorial examples of how more complex transforms can be accomplished. A set of VRML Prototypes that serve as dynamic model templates and building blocks have been provided on the internet as a repository for those modelers who wish to create and model dynamic behaviors of real world systems in a virtual world.

The fact that the VRML is designed to fit into the existing infrastructure of the internet and the addition of behavior information to 3D objects in VRML files opens up vast new possibilities for creating realistic dynamic models and share the dynamic model templates between modelers all over the world. Also, more pleasing and intuitive behavior shapes could be built using metaphors mapped onto well-known dynamic model templates such as finite state machines and functional block models. Result works of the students implementing the *rube* Methodology, which is in its infancy, shows the possibility of these new students to have aesthetic sensibilities to take advantage of the rapid technologies that supports the 3D graphics and open up a new Renaissance era in modeling.

The process of creating aesthetic dynamic models by students implementing the dynamic model templates invokes curiosity of how humans understand and utilize a new technology and concepts such as metaphors in modeling dynamic behaviors. Would there be a better way to help people to understand the process and encourage them to maximize their aesthetic abilities while enjoying and appreciate the exciting activities of modeling? For future work, we plan on extending these templates into UML with the goal of providing personalized interface options for UML models in 3D. While the underlying graph grammar will match the UML specification, the user will have options to choose alternate objects, replacing the 2D icons inherent in UML diagrams.

ACKNOWLEDGMENTS

Thanks to Robert Cubert for his work on the MFACTORY and PATH PROTOs. We would like to thank the following funding sources that have contributed towards our study of modeling and simulation: (1) 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 (2) 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. P. A. Fishwick, Simpack: Getting Started with Simulation Programming in C and C++, In 1992 Winter Simulation Conference, pages 154-162, Arlington, VA, December 1992.
2. R. M. Cubert and P. A. Fishwick, OOPM: An Object-Oriented Multimodeling and Simulation Application Framework, *Simulation*, 70(6):379-395, June 1998.
3. P. A. Fishwick, "Aesthetic Programming", in Leonardo, MIT Press, accepted, December 2000.
4. P. A. Fishwick, "3D Behavioral Model Design for Simulation and Software Engineering", 2000 Web3D/VRML Conference, Monterey, CA, pages 7-16, February 2000.
5. R. Carey and G. Bell, *The Annotated VRML 2.0 Reference Manual*, Addison-Wesley, 1997.
6. J. Hopkins and P. A. Fishwick, "A Three-Dimensional Human Agent Metaphor for Modeling and Simulation", in IEEE Trans. On Syst., Man and Cybern., in press, April 2000.
7. P. A. Fishwick, *Simulation Model Design and Execution: Building Digital Worlds*, Englewood Cliffs, NJ: Prentice-Hall, 1995.
8. P. A. Fishwick, *rube* tutorial web page, <http://www.cise.ufl.edu/~fishwick/rube/tutorial/>
9. P. C. Davis, P. A. Fishwick, C. M. Overstreet, C. D. Pegden, "Model Composability as a Research Investment: Responses to the Featured Paper", Proceedings of the 2000 Winter Simulation Conference, pages 1585-1591, December 2000.
10. James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Reading, MA, 1999.
11. Martin Fowler with Kendall Scott, *UML Distilled*, Addison-Wesley, May 1997