1 Introduction

The business of worldmaking is largely a symbolic affair. Especially in the case of computer-based realities, worlds exist only as descriptions. World-makers are confined to the expressive limits of their symbol systems. The range of possible worlds is limited to the available ways for defining them.

This paper describes an experimental 'construction set' for building multi-media learning environments. The system was designed to support MIT faculty in developing worlds for students – fictional places for the experience of remote or intangible phenomena, for traversing conventional structures of knowledge, and for practicing the skills of knowledge fabrication.

The goal of the project was to cut the time and skill needed to construct such learning environments. The basic approach was to diversify the tools of symbolic representation provided to the worldmaker. Rather than providing just one system, the aim was to offer several complementary approaches. While language is useful for many descriptions of the world, drawing is often more effective; in the same way, procedural computer code is often not the most efficient way to express the structures of information and change in a would-be computer world.

Our design strategy was first to choose a set of applications representative of the faculty projects, and then to abstract the common forms and structures from these. Next the problem was to devise efficient means of representation for these forms, and finally to design an appropriate data model and tool set to manipulate them. The resulting prototype combines four distinct representational approaches: directed graphs, multi-dimensional spatial frameworks, declarative constraints, and a procedural language. The objective of the system is to provide an environment where these four approaches are integrated so that an application developer can use all of them, defining each part of an application with the most efficient representational scheme.

The remainder of this section outlines the delivery platform and the initial set of target applications that shaped the Athena Muse system. The four methods of representation are described in the following sections, with examples drawn from the target application set. The paper concludes with a discussion of editing issues and a review of the current status of the project.

1.1 Project Athena's Visual Computing Group

Muse was initiated in Project Athena's Visual Computing Group, where the primary focus has been on exploring the uses of still and full motion imagery in educational software.

As an experimental delivery platform, the group modified a standard Athena networked workstation to support full color graphics as well as full motion, digitized video. The 'Visual Workstation' uses either a DEC MicroVaxII or IBM RT computer as a base with a Parallax Graphics board added as the display subsystem. NTSC video images are presented on the display through the X Windows System, where they can be managed as any other X window (acknowledgements to Levine, Goodwin, Levy).

1.2 Application-Driven Design

(brief description of the projects)

2 Directed Graphs

In educational computing, the distinction between 'electronic documents' and 'application software' is becoming blurred. Electronic documents are no longer static constructions sharing the limitations of their paper-based counterparts. Especially with a multi-media workstation, electronic 'documents' become richly choreographed presentations, often with complex user interaction. In this context, dynamic data processing and display becomes part of the expressive medium, and a document must be viewed more as a set of dynamic resources than a display-only 'electronic book'.

As a result of this shift, the authoring environment for a multi-media workstation must combine facilities both for preparation and manipulation of documents in a conventional sense along with facilities for creating interactive programs to manipulate and display data. A suitable starting point is to provide tools for building and manipulating directed graphs. As a system of representation it serves a useful role in both cases as seen in the following examples. The directed graph is a conventional notational system that has held an important place in math and science for centuries.

Hypermedia In terms of managing documents, the directed graph plays a fundamental role in the 'hypertext' and 'hypermedia' paradigm, where the aim is to manage extensive cross-references among a set of documents.

Here each document becomes a node in the graph, with the cross-references signified by the arcs or edges connecting them. Research at Brown University's IRIS project (Yankelovich, et. al., 1985, Garrett, et, al., 1986) as well as other hypermedia systems (see Conklin, 1987), suggest the utility of the approach in designing educational materials.

The Neuroanatomy project at Athena has something of this form, with the basic problem of cross-referencing 1400 descriptive text documents against an archive of still and motion images of the brain in cross section. In this case, the cross-references connect the text descriptions to regions of images where the anatomical structures appear.

State Transition Networks With regard to interactive programs, the directed graph is the essential form of a state transition network, which represents a fairly common class of application software in educational computing. In the state transition model, the nodes of the directed graph represent finite states of a system, and the arcs indicate branching pathways from one state to the next.

The Athena Language Learning projects use two different forms of this model. The first is a branching movie, a simulated interaction with a cast of fictional characters. Each scene occupies a node in the network. The order of scenes is based on input from the viewer, which is generally gathered at the end of each scene. The entire system can be represented as a tree structure, where viewer choices determine which pathway through the tree is followed, and ultimately which conclusion is reached.

The second example is a fixed route surrogate travel system. In the French foreign language project, "Direction Paris", students are asked to visit and explore apartments in Paris as part of the interactive story. The apartments themselves are represented as sequences of photographs that step through the various rooms. In this case, each sequence of photographs becomes a node in the network. The branch points are intersections in the apartment where the student can choose to turn left or right, continue straight ahead, or turn around. The form is slightly different from the previous example in that the graph does not follow a tree structure.

2.1 Limitations

While making extensive use of the directed graph model, the foreign language projects posed several challenges that indicate some of the limitations of this approach as a representational scheme.

The first was a need to provide dynamic subtitling for the full motion video. The subtitles were to be computer-generated texts synchronized phrase-for-phrase with the native speakers in the video.

The second requirement was that the student should be able to stop the video at any moment and back up an arbitrary amount to have the presentation repeated. The problem, of course, is that with dynamic subtitles this means rewinding the text to the proper phrase along with the video.

In addition, the teachers wanted students to be able to select any word or phrase in the subtitle stream and have it looked up in a glossary, and also to provide intermittant "cultural flags" that would pop up on the screen when the speaker used an idiomatic expression or made reference to something of particular cultural interest.

These results could be achieved with a directed graph model, but in a relatively awkward way. The whole configuration is more easily represented as a timeline with onset and offset times for the auxiliary annotations. The timeline imples a spatial system fundamentally different from the spaceless web of nodes that make a directed graph. The approach taken in Muse, described in the following section, is an extension to the directed graph in which each node contains a space of arbitrary dimensionality.

3 Multi-Dimensional Information Spaces

3.1 First Dimension

The easiest way to couple subtitles with video is to create linkages between frame numbers on the videodisc and text fragments in the subtitle stream. With this arrangement, the video stream becomes the representative of 'time' in the display, and the subtitles look directly to the video for timing information.

Instead of using frame numbers so directly, we chose to create a layer of abstraction between the two channels. We created an independent dimension of time and attached both the video and text to this abstraction separately. This way, none of the actual components of the display are directly tied to one another. This makes it easy to add and remove channels of information without disrupting the whole presentation.

The abstract dimension can be controlled by the system clock, driving time forward and determining when the text and video appear and disappear. An arbitrary number of channels can be attached to this dimension of time, so it can handle multiple streams of subtitles, cultural flags, and so on, in a rather complex 'display choreography' (see Fig. 1).

This general approach of building the display around an abstract dimension is useful in any situation where there is a need to link data resources to a stream of motion video. Beyond the foreign language applications, several of the other projects have requirements of this kind.

Gloriana Davenport's *New Orleans* project at the Film/Video group of the Media Lab is a good example, where the action centers on a large database of full motion documentary footage stored on videodisc. These materials were produced in New Orleans in 19**. They document the planning process for the New Orleans World's Fair, and include meetings of the principal designers, developers, and policy makers involved with that project.

The materials are being used by students of social policy and urban planning to explore the decision making processes that went on, and the roles of the various participants. They are also being used by students of documentary film-making who can selectively draw on the materials to create miniature documentaries representing different interests and perspectives. In both cases it is important for the users to be able to attach annotations and data references of various kinds to specific points in the video. This can be implemented using the same strategy as the subtitles.

3.2 Second Dimension

It's a small step from one to two dimensions, where x and y dimensions define a Cartesian space. Here the dimensions function again as an abstract framework used to organize and manipulate the contents of a spatial display.

Almost all of the video-based projects have some requirement for this format, but two in particular are driving the design. The first is Steve Wertheim's *Neuroanatomy* project [Wertheim, 19..]. This project is based on a large database of still and motion images pertaining to brain structure and dissection. This imagery is to be linked to an extensive collection of supporting textual and graphic materials.

Part of the problem is simply manipulating images on the screen – zooming and panning, for example. By using the dimension structure to organize the image display, these effects can be achieved by manipulating the dimensions. In this respect, the dimensions provide a coherent system for handling different kinds of display information.

The Muse prototype allows the x and y dimensions to be defined as virtual coordinates, so the display window becomes a viewport into a larger space. Zooming and panning over an image becomes a matter of moving the viewport within this virtual space and changing the scope of its display. These parameters are controlled directly through the x and y dimensions.

3.3 n-th Dimension

Muse does not limit the number of dimensions that can be created. This allows for spaces of arbitrary size and dimensionality. Dimensions need not represent real time or physical space; they can be used to represent any changing value in a dynamic system.

Dynamic systems are often modeled as a 'state space' and a 'dynamic', the dynamic being a function or process that governs how the 'current position' of the system changes and moves through its state space. The state space is a set of coordinates or dimensions that define the range of movement along each degree of freedom in the system.

Dynamic systems of this kind can be used to implement some forms of simulation. Simulation is one of the preferred pedagogic approaches used by the MIT faculty. Given the combination of multi-dimensional spaces and graphs, Muse offers some interesting possiblities for implementing educational simulations.

The reason is that the Muse dimensions are the basic mechanism not only for organizing the simulation state space but the hypermedia cross-referencing as well. The problem is not simply to build a simulation, but to *integrate* simulation with other pedagogic approaches such as tutorial and reference materials. The Muse dimensions allow a teacher to make cross references directly into the state space of a dynamic system. This provides a generic method for a.) querying the state of the simulation; b.) changing the state of the simulation; and c.) setting 'trigger' conditions within the simulation that can cause messages to be sent to other nodes in the directed graph.

The classic difficulty with educational simulation is that one generally does not know what the student is doing – whether important conditions have been tried, whether the student really sees what the simulation is intended to show, and so forth.

The Muse system gives relatively easy access to the workings of the simulation. The state can be monitored simply by checking the current position of the different dimensions. A separate document can be set up to watch what happens. Trigger conditions can be defined directly within the simulation so that if the current position moves within a specified range the simulation 'document' will react in some way, such as sending a message to a companion document. Furthermore, since the document dimensions are relatively accessible within the directed graph, one document can be set up to take control of another by setting it's position. This can be used, for instance, to drive a simulation through a specific illustration or example.

The Navigation project is an example of a time-driven simulation implemented with the Athena Muse. This material simulates a boat moving through approximately 2 square miles of coastal waters in Maine, using a library of approximately 10,000 images to represent the environment.

The whole simulation is implemented as one interactive document. Two dimensions are used to represent the boat's x and y position within the environment. Two more are used to represent the boat's heading and speed (used to compute the boat's position). A separate dimension is used to keep track of the user's viewing angle, since it is possible to look off in any direction regardless of the boat's heading. Finally, two dimensions are used to manage a simulated compass that can be positioned anywhere on the image to take bearings.

The effect is that the user can steer through the space, using buttons and slider bars to control the 'boat', and use the mouse to take compass bearing directly from the images. The simulation becomes active whenever the document is opened, and the dimensions make it possible to watch what happens from outside or to step in and take control.

In order to implement such a simulation, a fairly high level of orchestration needs to be set up among the various components of the display. The directed graph and spatial framework provide a foundation for organizing and controlling the individual elements of the display, but some facility is also needed to link the behavior of these elements together. In many cases, the desired relationships can be defined as simple functions and so implemented as declarative constraints. The next section describes the use of constraints in Muse as a third tool available for building applications.

4 Declarative Constraints

Constraints are functional bindings within a system that are enforced among different components and automatically propagate change among them (Borning). In Muse such bindings can be applied both to the dimensions and to various attributes of the display elements.

An example is the relationship between a slide bar and a dimension in Muse. It is often important to have a visible representation of the dimension, either to control it or to see what state it is in. A scroll bar or slide bar can be used for both operations. A two-way constraint needs to be established between the scroll bar and the dimension, so that mouse click or drag events on the scroll bar will be propagated to the dimension and similarly any change in the dimension which may occur from some other input or timing event will be propagated to the scrollbar so that it can readjust its display. In Muse, this relationship is effected explicitly by declaring a 'represents' relation in the scroll bar definition.

A second case is that of dynamic graphic objects which modify their display attributes based on constraint bindings to dimensions or other data resources in the system. A numeric representation of a dimension, similar in purpose to the scrollbar example above, can be made by creating a graphical text object whose text value is defined as a function of the dimension position. Whenever the dimension changes, the text object generates a new text string to print on the display.

A simulated hand-bearing compass was created for the Navigation project by constructing a similar text object that moved to the location of a mouse click and printed a value between 0 and 359. This interaction was effected indirectly through the manipulation of dimensions.

A single point can be used to plot a function by constraining its x position to a dimension and then constraining its y value to be a function of x. Iterating through the dimension space then causes it to plot the function.

5 Procedural Descriptions

(Russ goes here)

5.1 Interpreted Objects

5.2 Access to System Resources

(both Data and Process)

6 An Integrated Environment

The primary objectives of the Athena Muse system are to reduce the time and skill needed to produce educational materials. These goals depend on three things: a.) defining proper representational systems so that materials can be produced efficiently, b.) a coherent data model that is commensurate with these representational schemes, and c.) editing tools that support appropriate functions in a way that is conceptually clear to the user.

Work on the Muse prototype has focused on integrating multiple representational schemes and developing a coherent data model. The next phase of the project takes the issues of editing as its principal aim.

6.1 A Coherent Data Model

By way of summary, the Muse data model seeks to provide integrated support for four distinct representational systems. The underlying foundation is a directed graph model that affords support for both 'hypermedia' cross-referencing and state transition models.

The nodes of the graph have an internal structure that allows for n-dimensional state spaces to be constructed. The spatial model provides a mechanism for organizing information spatially or temporally, and more generally affords a way to implement dynamic system simulations.

(similar paragraphs on directed graphs and objects, accompanied by a little picture)

6.2 Issues in Editing

(some little opener...)

In general, it should be noted that the approach taken in Muse has been to permit as much of an application as possible to be defined as declarative data specifications as opposed to procedural specification. In many cases, educational materials can be generated with no procedural coding on the part of the developer. The advantages are twofold: first, that data specification requires less time and skill than procedural coding, and second, that it lends itself more readily to 'direct manipulation' style of interface.

Integrating four principal schemes of representation implies some editing facilities for each. For the most part, each editor is working on a different aspect of a single data model. (sentence or so on the breakdown of the editors – primitive data, compound, etc.)

Editing directed graphs and procedural code are relatively familiar. The focus of this section will be on specific issues in editing dimensional spaces and constraints.

Timeline Editor The first issue with repect to a dimensional information space is that of binding and manipulating data within the spatial frame. The basic tool we have experimented with is a 'timeline editor'. The technique is not intrinsically related to problems with time as a dimension, but the notion of a timeline with a stream of events starting and ending captures the basic idea.

Normally, a dimensioned space is used to establish relationships among a number of separate elements or sets of display data. With the timeline editor, the dimension is represented as a line. Each element or set of display data is assigned to a separate track, or channel, displayed below the line (see Figure..). Range markers within the data channels can be manipulated directly with the mouse to create, delete, or modify onset and offset times.

¡timeline figure about here¿

This technique can be used for any dimension. One serious limitation, of course, is that this method does not show relationships involving more than one dimension. (miscellaneous remarks...)

Cutting and Pasting Space A second issue with editing dimensioned spaces has to do with editing the spaces themselves as opposed to the display data they may contain. As with the timeline editor, the initial experiments with Muse have focused on 1-dimensional spaces.

One example is a video editor that manipulates annotated video; that is, an editor that cuts and pastes video along with any subtitles, scripts, or other annotations that may be linked to it. In effect, this means cutting a slice of time and collecting all of the display elements bound to it.

(one or two more paragraphs)

Spreadsheet Patch Panel

6.3 Summary

(Future development) (Two heads are better than one)