

Virtual Worlds for Education 2006

Lee H. Tichenor
Computer Science Dept. Western Illinois University
Macomb, IL 61455

1. Abstract. *Creation of a K-12 Virtual World educational tool set while providing support for our MS graduate students has been a goal of this lab for a number of years. We have tested different programming and 3-D environments with single semester team projects in real time animated graphics systems to find an environment appropriate to these ends. VRML and java based systems seem to be the best for allowing our students to learn the development environment while leaving enough time in one semester to create meaningful content.*

2. Keywords. Virtual Worlds, Distance Education Tools

3. Introduction. Resources for primary and secondary education are dwindling, and the disparity between have and have-not school districts is increasing. Be they inner city schools, challenged to attract specialized educators, or the more rural districts that lack the financial resources or student populations to attract specialized teachers, many schools are unable to offer opportunities to their students that might significantly enhance their background and their abilities to succeed at the next level[1].

Distance learning has been proposed as an option to level the playing field, and has, in a number of studies [2] been found to be as effective as other techniques. Virtual Worlds are a specialized area within this arena. As outlined in a number of publications [3-5] they would provide the student and teacher a virtual meeting space to accommodate the interactions

required to complete a course. Given content relevant to a class, (lecture notes, video, sound, a relevant URL, etc. and a meeting time) most interactions experienced in the classroom would be possible in a virtual environment.

Given all of the possibilities, one might ask why are not more virtual options available to the K-12 students in this country?

The answer is both simple and problematic. First, there is no tool for implementing Virtual Worlds for primary and secondary educators that will work within a very real set of constraints. These constraints arise from two areas of concern. The first is a very significant lack of financial resources, and the second, a fundamental lack of technical expertise of our K-12 educators. My experience with many teachers within this population makes very plain the level to which these tools must be built if they are to be used by classroom teachers to develop and provide virtual learning opportunities. As the developer and technical assistance contact for a pre-school student test report and analyses tool, I have discussed system issues with a number of teachers. I, have also been involved with a summer outreach program with K-12 science teachers providing stipends and an extensive software library to get them involved with computer simulation in their classes. The results of these interactions lead me to believe a general tool for development of content by K-12 teachers will need be TRIVIAL to learn and use, with a single GUI, and FREE.

What is needed is a tool that can be accessed by a classroom teacher and is very easy to start, has a small learning curve, is consistent, provides the depth needed to develop meaningful applications, and allows the typical educational tasks to be automated and implemented without a significant effort. My contact with these teachers has allowed me to create a list of system and environmental issues to be avoided. These are items that have caused problems and considerable frustration among the educators I have worked with and are referenced in the table below.

- | |
|--|
| <ol style="list-style-type: none"> 1. reference to file or storage paths 2. reference to file types 3. variation within the interface 4. deviance from standard I/O 5. technical jargon |
|--|

Table 2 System Issues to Avoid

While avoiding the items mentioned above the system must also include certain tools and attributes to enable K-12 teachers to build content and link their creation process to their classroom experience. We feel the items in table 2, are required for a successful usable system.

Finally, the tool set must allow for publication of the content to appropriate platforms and allow for appropriate student to student and student to teacher real time interactions, in addition to stand alone use.

There is currently no tool set that is usable by our target population for content development that comes close to the requirements cited above. Native VRML and EAI is not an option. Any environment that looks like programming is doomed to failure in this domain. Environments like Adobe's Atmosphere, although well intentioned, have evolved into extremely inbred niche systems with no ability to communicate transparently with the rest of the world.

Outsourcing the integration of content into

- | |
|---|
| <ol style="list-style-type: none"> 1. import (trivially) all common text types 2. import all common graphics standards (2-d & 3d) 3. interface that relates to the concept of a class (grade book - notes, etc) 4. provide tools <ol style="list-style-type: none"> i. Text reading ii. PDF iii. Virtual 3d <ol style="list-style-type: none"> a. Poser b. Viewpoint iv. Tutorial v. Self Quiz vi. Scored Quizzes vii. System logs viii. Grade books 5. Consistent GUI <ol style="list-style-type: none"> i. Inserts structure & tools seamlessly ii. Inserts all components exactly the same way iii. TRIVIAL access to structure options & attributes <ol style="list-style-type: none"> a Font Color Style etc b. Text or graphical content 6. A 3-D World development tool <ol style="list-style-type: none"> i. Builds 3-d space with basic primitives ii. Imports standard 3-d mesh structures iii. Simple animation tools iv. Provides object manipulation and view options 7. Student GUI <ol style="list-style-type: none"> i. Stand alone multi platform ii. Supports VRML - VW's iii. Supports messaging iv. Provides access to "all sorts of" teaching tools |
|---|

Table 1 System Requirements

virtual environments may seem like an appropriate alternative. However, options like Active Worlds are costly and may not allow the flexibility needed to precisely mold content to specific cases. Whether these environments

provide all of the support needed to maintain the content for classes (in addition to other support) is not relevant, as they are cost prohibitive. For example, the basic pricing for an Active Worlds educational classroom is \$650 with an \$400 annual fee [6]. Class content still must be compiled into a coherent world. This cost in money, time, and expertise is beyond that available to many, if not most, school districts.

Our lab at Western Illinois University is looking to develop a tool set for such an environment to be built with the lower level tools currently available. Our ideal would be an environment where our graduate students could thrive, gain meaningful experience, and provide significant input toward the development of our system. WIU has some very bright, imaginative, and energetic graduate students, many of whom would love the chance to work on the development of such a system. As Masters Degree students, however, they are here only for a short time, often with only a single semester available. The hope, then, is to find a reasonably stable VW development environment in which our typical MS level student could become comfortable in a few weeks. This would allow a significant portion of a semester, which is about all the time many of these students have to work on external projects, for content and system tool development. Rooms like the “Font Museum” a portion of which is shown in Figure 1, are doable in a few short weeks.

4. Course Projects. During the last few years we have investigated various environments through the development of Virtual World prototypes as semester length class projects. This was done in a graphics programming course offered to masters level students who have completed a first general survey course in computer graphics. Students were expected to have a working knowledge of the various rendering techniques, projection and transformation options, and some ability to produce useable models from our 2-D and 3-D graphics development software. In each of the classes we started a completely new project, and hoped to produce a functioning prototype by

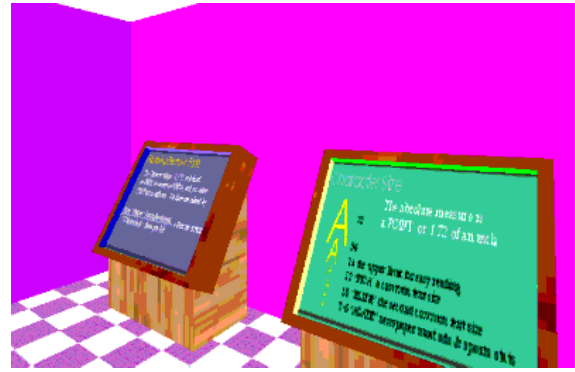


Figure 1 VRML Educational Content

semester’s end. Each of the past five years a development environment using tools available over the WWW, appropriate compilers, and 3-D content creation software was selected and project goals determined for the semester. Three to four member student team self select for the group projects.

The development environments have been Adobe Atmosphere, VRML and Java based Deep Matrix, JAVA with Swing and VRML or C++ with OpenGL. The schedule for each of the semesters opens with six to eight weeks used to present the development environments and design goals, and leaves the remaining time for creation and presentation of the projects. The JAVA systems, with VRML taking care of many of our virtual world navigation requirements, were to be completed with an embedded VRML multi-user panel on a GUI providing world user information and chat options. The C++ environments where reading model data, navigation, and terrain following needed to be coded, were designed to implement a single window open to a 3-D terrain with embedded animated and non-animated objects, including (just for fun) a particle system.

Adobe Atmosphere, in it’s Beta release in 2002, looked like a promising starting point for an educational VW tool set. It was composed of a building tool, both stand alone and plugin viewers, good avatar support, and chat capabilities. It provided a reasonable interface for the placement of the basic world components such as walls, floors, and stairs, Figure 2. Additionally, there was an extremely active and

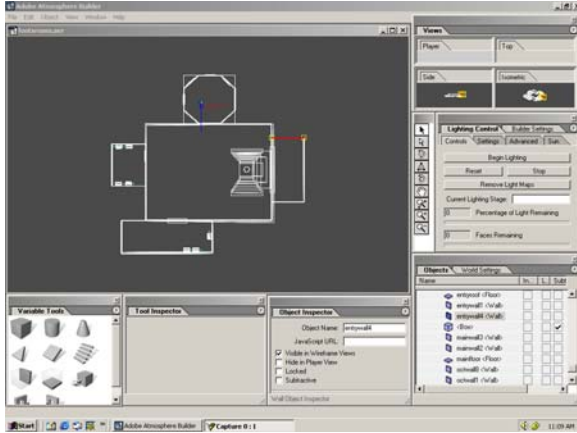


Figure 2 Adobe Atmosphere Builder

enthusiastic user community growing up with the software. Remarkable environments were being made. Beautiful worlds were created with the texture mapping tools provided, even though, only limited graphic file access was available in the early releases. Beyond these basic building techniques, however, things were very difficult. Animation and object manipulation was only provided through Viewpoint Technologies' very complex scene development. This turned out to be too complicated for most of our MS students to implement in any meaningful way in a single semester. Also, there was limited or no capability to link to tools or content outside the Atmosphere domain. The environment in its configuration of 2002 had promise but was not nearly easy enough to use or flexible enough to be a meaningful tool in education [7]. The product was released in 2004 and removed from the market by Adobe later that same year.

The focus of the next class was VRML and its External Authoring Interface, EAI, linked through a web browser and a JAVA client-server system. This type of environment, as VRML is exported by most modeling software, allows a physical model to be created with little difficulty. A JAVA client server system can be built with 50 lines of code. The first eight weeks of the semester covered VRML in great detail. In the last eight weeks of our semester we were able to write scripts to VRML's EAI to accomplish some trivial tasks such as getting an avatar into a scene, chat, and activating a

number of embedded VRML animations. Behind the scenes coding like user databases, multiple user objects, chat modes were not completed during the one semester course. They were planned and probably would not have proved too difficult. An HTML interface provided a GUI to the basic functionality.

The downside to this semester's work was the EAI and its interaction with the remainder of the system. Our class had considerable difficulty providing stable systems that would work in the different labs we have in our building. The combinations of browser type and release number (Netscape 4.7 vs other Netscape releases for example) and the plugin (Cosmo Player, Cortona, or Blaxxun's) required to read the VRML, and Java run time environments required seemingly endless "tweaking".

A third semester investigated Deep Matrix by Geometrek [8] that provided functionality similar to where our previous semester's project was headed. However, strategies like embedding text within HTML files to provide data for the java portion of the system, were

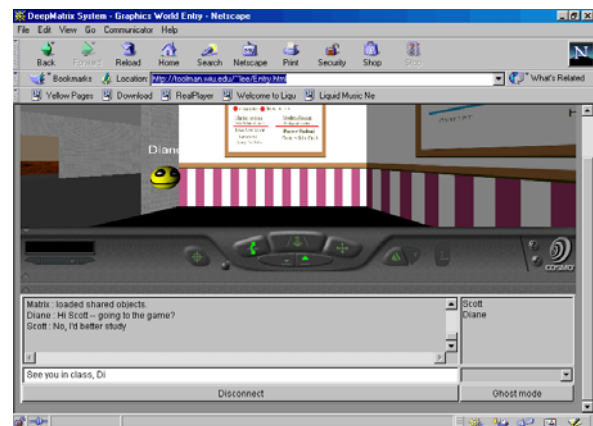


Figure 3 VRML and Avatar

among the methods created to allow one component of the system to communicate with another, and they proved difficult to design and implement. Additionally, any data manipulation functionality required special coding in C++ or JAVA within the model files. Although we were able to get many of these techniques to work, we feel less complex methods are desirable. Also, the semester's

project using this code proved, again, to be too system dependant. We could get projects to work in one lab but not another, in one browser but not another, with one plugin but not another, and so on.

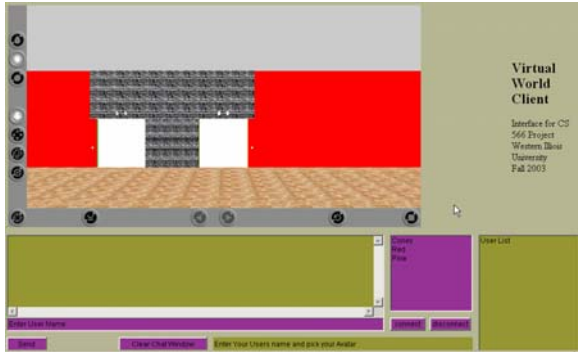


Figure 4 Java Browser & Swing GUI

These experiences have led us to the opinion that any software of this type will need a dedicated interface not linked to a standard browser family. The current implementation of VRML requires a plugin in a browser to view a document. That in itself is problematic. Installation of one of the three most common plugins is not trivial to the common user. Compound that with the possibility of multiple plugins and various JAVA installations and problems will and do occur. For example, the last week of class during the semester these projects were completed a faculty member installed some software on our classroom machine that adjusted the java paths. As a result none of the VR programs would work without resetting some system variables. This is the type of occurrence that can not happen with an educational software package.

Our last experiment with VRML based environments, Fall 04, involved the relatively new X3D/XML standard. Our interface was designed using JAVA Swing classes, components of the EAI, and borrowed from some of the J3D browser class ideas. By semesters' end all of the groups had been able to build functioning prototypes supporting limited connectivity. User lists were maintained, as was chat. The 3-D worlds were viewable, but we were not able to implement remote user

interactions within them. We simply ran out of time.

This past semester we used C++ and OpenGL to build our environments. We included wavefront, .OBJ, files for our static objects and quake MD2 models for animated objects. Each group was also to include a particle system of their own design in their projects. Our thought for testing this environment was that the C++ executable code would be cleaner and more easily transportable. It was hoped that the back end of the system, user lists, system data, etc. would be created in much less time and be more understandable.

We unfortunately never got that far. The complexity associated with simply generating a 3-D window that allows the user to terrain follow took too much time. Terrain was created and stored in .RAW files. Many of the students used Terragen or a similar program to create their base environment. A simple AVATAR / Camera construct was implemented using the glFrustum and gluLookAt commands. A "future location" vector is projected forward and its Y value adjusted to the terrain elevation. This works quite well if the terrain elevation differences are small relative to the forward stepping rate.

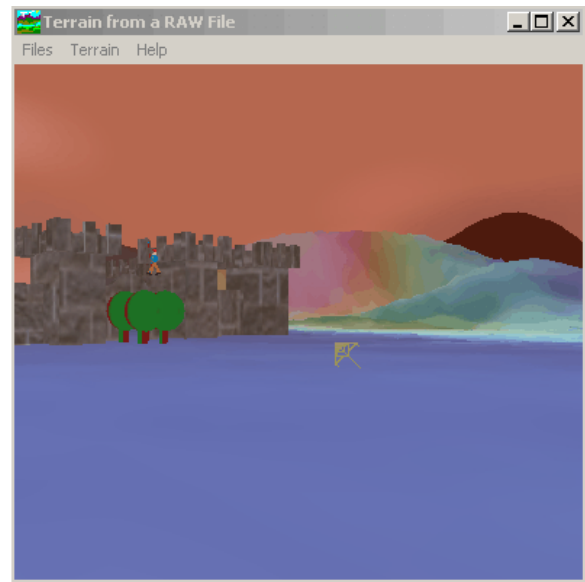


Figure 5 A C++ OpenGL VW Window

Tasks of this type are very simple to demonstrate, the students understand them easily, and really enjoy getting them coded. However, the complexity associated with loading and displaying virtual components of different data types, the OpenGL matrix stack and associated transformations required to display and animate a world took weeks to learn.

We spent nearly four weeks coding the VW functionality into our environments that comes built into a basic VRML file. Additionally, the groups that built their windows in native code and did not use the GLUT libraries were hard pressed to complete the basic world window within the semester time frame.

5. Conclusions. Of the environments tested we will focus in the future on the new X3d tools and associated Java constructs. It seems to offer considerable flexibility and enough built in functionality that we can create reasonably complex environments in a single semester. Current validated systems support most all of VRML2, and its basis in XML should allow system interaction not available with the old HTML/EAI systems. Additionally, the ability through XML to create and seamlessly use tools to manipulate data ranging from class rosters and grade books, to lesson plans, to 2d & 3d content, to sound to etc., etc. provides opportunities to develop the VW teaching environment described above.

6. Literature Cited

[1] 2004. Getting Highly Qualified Teachers Where They Are Needed Most, Rural Policy Matters - September 2004.

[2] Jane M. Carey. 2004. Effective Student Outcomes: A Comparison of Online and Face-to-Face Delivery Modes. DEOSNEWS Volume 13 - Issue 4

[3] Charles E. Hughes, J. Michael Moshell, (1997). Shared Virtual Worlds for Education: The ExploreNet Experiment, ACM Multimedia 5(2), pp. 145-154.

[4] J.M. Moshell and C.E.Hughes (2002).

Virtual Environments As a Tool for Academic Learning, Handbook of Virtual Environments, K.M. Stanley Ed., LEA Pub pp 893-911.

[5] Cobb, S. Neale, H. Crosier, J. and Wilson, J.R. (2002). Development and Evaluation of Virtual Environments for Education, Handbook of Virtual Environments, K.M. Stanley Ed., LEA Pub pp 911-936.

[6] Active Worlds Educational Universe Pricing Information (2006) at http://www.activeworlds.com/edu/awedu_pricing.asp

[7] Tichenor, L.H (2002). Current State of Adobe Atmosphere as a VW Tool for Educators, abst. Transactions of the Illinois State Academy of Science, 95, p67.

[8] Reitmayer, G. Carroll, S Reitemeyer, A. Wagner, M.G. (1998). DeepMatrix - An Open Technology Based Virtual Environment System, White Paper @ <http://www.geometrek.com/developers/whitepapers.html>