Building a VR Narrative

Josephine Anstey, Dave Pape, Dan Sandin
Electronic Visualization Laboratory
University of Illinois at Chicago

ABSTRACT

In this paper we discuss issues involved in creating art and cultural heritage projects in Virtual Reality with particular reference to one interactive narrative, “The Thing Growing”. In the first section we will briefly discuss the potential of VR as a medium for the production of art and the interpretation of culture. In the second section we describe “The Thing Growing” project. In the third section we discuss building an interactive narrative in VR using XP, an authoring system we designed to simplify the process of producing projects in VR. In the fourth section we will discuss some issues involved in presenting art and cultural heritage projects in VR.

Keywords: narrative, interactive art, authoring

1. VR POTENTIAL FOR ART AND CULTURE

The combination of interaction, immersion, and the digital computer make VR a unique medium for cultural productions, providing new opportunities, but also new challenges. VR offers artists a brave new world for the imagination; a tantalizing mix of absolute freedom and niggling limitations. On the one hand you have a limitless 3D canvas in which to create a visual effect. On the other the computer can only redraw a very limited number of polygons and textures at the 20 or more frames-per-second needed to effectively preserve the virtual illusion. On the one hand the viewer can wander at will through your world, on the other the artist can no longer easily control visual surprises. Visual immersion is only one aspect of this new medium. Equally important is the aspect of the user's interaction with the virtual environment. Since, as humans, we live our whole lives in interactive environments, we assume that it will be easy to create interactive, virtual environments. But it proves to be difficult to build intuitive and rich interaction in VR.

At its best VR gives artists the opportunity to realize visions that no previous medium could express. Examples of compelling and thought-provoking VR art experiences are “Osmose” by Char Davies and “World Skin” by Maurice Benayoun. “Osmose” is a non-linear HMD experience, involving 3D sound and real-time motion tracking based on breathing and balance which translates into navigation. It is “a place for facilitating awareness of one's own self as embodied consciousness in enveloping space.” [1] World Skin is a CAVE® installation in which users are plunged into a war environment. 2D cut out images of war are placed in the 3D world and this is combined with very realistic, spatialized sound effects. Three of the users are given cameras which are tracked. As they take pictures the image caught by the camera frustum at that instant is removed from the world – leaving blank, cut-out whiteness.[2] Museums worldwide are also becoming increasingly aware of the possibility of using virtual environments in the interpretation and demonstration of cultural heritage. At the Foundation for the Hellenic World, Maria Roussou and her team are currently building a reconstruction of the ancient city of Miletus in VR. [3]

Creating tools and techniques that will facilitate the construction of VR narratives is important for both artists and cultural workers. VR offers a new way of experiencing fiction. Someone reading a book or viewing a film or video may identify with the protagonist but in VR the relationship is more direct; the user *is* the protagonist. Culturally based VR will be able to move beyond fly-throughs to deploy virtual characters in reconstructions of events that the user will be able to alter and manipulate. In VR, it is possible to construct archeological and anthropological theories as immersive, surround sound, 3D, interactive simulations.

2. DESCRIPTION OF THE THING GROWING

The Thing Growing is an example of an artwork that could only be fully realized in VR. The project started life as a short story idea; an exploration of a relationship that was cloying and claustrophobic but emotionally hard to escape. We realized that transposing the idea to a CAVE environment would make it possible to create the tensions and emotions of such a relationship for a user directly. Our challenge, therefore, was to create an interactive narrative in VR, in which the user was

established at the center of the story and would become emotionally involved with a believable computer-controlled character.

2.1 The storyline

The narrative in The Thing Growing is built in a classical bridge structure with three acts. In each act the user is involved in interactive activity. The narrative as a whole is moved on either as a result of the user's choices, or by time. In act one the user enters the virtual environment and navigates across a plain. There is a small shed in the distance – a voice-over prompts the user to approach it. Inside she finds a box. A key appears. If the user clicks on it, the box bursts open and the "Thing" leaps out. It dances around and shouts, "I'm free! You freed me!" Then it bows before the user saying, "I love you!" Thus, the two main protagonists, the user and the Thing, are introduced and the Thing declares its interest in the user. This act also familiarizes the user with the input hardware (the wand joystick and buttons).

The second act consists of the Thing trying to teach the user a dance – whatever the user wants to do. It begins to reveal that its "love" is of the dominating, manipulative kind. Here the interaction involves the user's whole body. The Thing demonstrates a dance step and cajoles, whines, begs or demands that the user to copy it by moving her own, tracked, physical body. The Thing becomes a constant nagging presence, always too close, always telling the user what to do. Finally the Thing flies into a temper and runs off. The user is relieved, but once the Thing has gone, rocks on the plain come alive and herd and stalk her. One rears up, making slobbering, sounds and traps the user. Within seconds the Thing arrives. If the user dances nicely for it, it will release her. Once freed from the rock, the Thing suggests that it will copy the dance steps of the user.

Act Three interrupts the dancing. Lightning flashes. The plain disappears and the Thing and the user are plunged into a new environment. Here the Thing's four cousins, horrified by the "unnatural" relationship they perceive between the user (a meat-object) and the Thing, imprison the user and beat the Thing. They then exit muttering dark threats. The Thing produces a gun, and it becomes the user's job to blast them out of prison and then to fight the cousins. Finally all the cousins are killed or have escaped. The Thing and the user are alone again. But now the user has a gun. The entire piece is designed for this moment. The Thing suddenly realizes that the user could turn the gun on it. The question for the user is should she kill the Thing or not? There are two endings, one for each alternative. However, neither allows the user to ultimately escape the trap of this clinging relationship.

Figure 1. The Thing and the user

Figure 2. The angry cousins

2.2 The experience

The project was initially designed to run in a CAVE virtual reality system or an ImmersaDesk™. These systems provide a stereoscopic, viewer-centered perspective, visual display, and audio from loudspeakers. The default input devices are a wand, with 3 buttons and a small joystick, and a 6 degree-of-freedom electromagnetic tracker, which tracks the user's head and the

wand. For The Thing Growing, additional tracking sensors are attached to the user's other hand and body. These inputs define the forms of interaction available to a user in a CAVE; an application can react to the position of the user or the wand, or to button presses or joystick use. Only one person at a time, the tracked viewer, will be able to interact directly with the application, although others may be watching with them in the CAVE.

Our choice of dancing for the central interaction in the second act of the story was driven by the available hardware. We wanted a fully two-way interaction between the user and the Thing, with the computer character responding to the user and vice-versa. Without attempting to add more technology, such as voice recognition, or an unnatural, indirect interface, such as 3D menus, the main feedback that our program could expect to get from the user was the position of the user's head and hands. Thus, we had the user 'communicate' with the Thing by moving her body.

Figure 3. A tracked user dancing

3. VR PRODUCTION

3.1 Using a CAVE for art

Our discussion of VR production will be primarily confined to issues involving the CAVE and ImmersaDesk, although the XP authoring tool we describe and the CAVE library can be configured to run on any VR systems. The CAVE was originally designed for computational scientists and engineers; it addressed a number of issues with head-mounted displays which made virtual reality difficult to use in their work.[4] It provides a high-resolution display, comparable to that of desktop workstations; it offers a very wide field of view; it is relatively unencumbering, requiring only lightweight LCD shutter glasses; and it can accommodate multiple viewers, something that is very important to researchers who expect to be able to work in groups. These features are significant for artistic and cultural heritage applications of VR as well. Artists and cultural workers want the highest resolution for their work. Museums that want to invest in VR, benefit from a system which can accommodate groups.

Despite the attractiveness of VR and the CAVE for creating new art works, there are a number of problems which must be faced when using the current technology. One fundamental problem is the high expense and therefore relative inaccessibility of CAVEs, an issue we will address in section four. A second, equally difficult problem, is that many artists who are interested in creating virtual reality pieces are daunted by the complex computer coding that is required. Typically, artists work with engineers, but this scenario does not give an artist the opportunity to experiment and work in a hands on fashion with the application; also, an engineer's time can be expensive! The Thing Growing was built using XP, a framework for

virtual environment applications that takes care of many of the basic tasks of creating VR worlds and is easily extendible.[5] We specifically designed XP to give artists an entry point into creating virtual environments.

3.2 XP

The XP system grew from software developed for the "Multi-MegaBook in the CAVE" [6], and "Mitologies" [7] application. These two applications are both large-scale environments. They involve hundreds of megabytes of models, texture-maps, and audio clips; the environments cover a large virtual space, and include multiple scenes. Typically, CAVE applications such as these would be developed in C or C++ using tools such as Silicon Graphics' IRIS Performer.[8] Such an approach, however, would have kept the artists themselves from being able to do much direct work beyond creating the raw materials (models and sounds). We believe that, as interaction is a key element of VR art, it is important for artists to be able to define that interaction; that is, to program it. But many artists need a simpler programming system, rather than the full blown, yet arcane, power of something like C++ and Performer. Furthermore, in many cases the sort of programming (interaction definition) that needs to be done is fairly simple – things such as "let the user pick these objects up" or "when the user enters this room, play a fanfare and start this object's animation". On the other hand, many artists would still like to be able to include some more advanced behaviors or effects which an experienced programmer could build for them. Systems such as Alice [9], which is based on the Python scripting language, have been successful at allowing novice developers to create virtual environments. We wanted to create a system that would not only be useful to novices, but also allow them to employ the efforts of more experienced VR programmers. Our XP framework includes many of the features common to virtual art environments, such as navigating through the environment, picking up and dropping objects, transitioning between scenes when the user moves through a portal, and "clicking on" objects to trigger special reactions. It also makes it possible for programmers to add new tools needed for specific applications, and allows artists to create the final environment by assembling the appropriate pieces at a script-like level.

XP is based on Performer, the CAVE libraries, a sound library, and C++. The system is divided into two major aspects – the text file(s) defining an application as a collection of nodes and their connections via events and messages, and the lower-level C++ classes which implement the nodes. With this division, it is possible to split the work of world-creation between experienced programmers and non-programmers. The programmers create any new node classes which are needed for application-specific behaviors, such as graphical effects or elements of a character's intelligence. The other team members build the full application by plugging together object and behavior nodes in the text files. In practice, there is likely to be an overlap between these groups; because XP handles many of the trickier details of Performer and the CAVE libraries, artists have been able to start doing some of the C++ programming of new nodes using a prototype class template. The XP framework also makes it easier to re-use code between applications, because the code is all in modular nodes with standardized interfaces.

Application authors create a virtual environment in XP by editing a scene file. The file is a high-level description of the environment's scene graph. It lays out the structure of the scene graph, listing the nodes and their hierarchical relationships, assigns attributes to the nodes, and defines interactions among the nodes. User interactions, and behaviors involving multiple nodes, are defined by events and messages. In XP, an event is considered to be something that happens in a node. As in many systems, this can often be a change in state of some of the node's data; however, in general, an event can be anything that a particular node class is interested in detecting and signaling others about, such as the user entering a region, or a sound finishing its playback. Messages can then be sent from the node to other nodes in response to the event. For instance, a trigger node could detect when a wand button-press event has occurred, and send the message "toggle" to a light, to turn the light on or off in response to the user's action. The detection of events is done by the C++ coded node class implementation; the connection between events and the messages they trigger is defined in the scene file. Thus, specific nodes are used to detect specific events, but they may be re-used arbitrarily when the artist is building a new environment. Figure 4 shows an example fragment of the scene graph from The Thing Growing; in scene 2, after the user enters the shed and approaches the box, the key appears; when the user clicks on the key with the wand, it tells the box to burst and release the Thing.

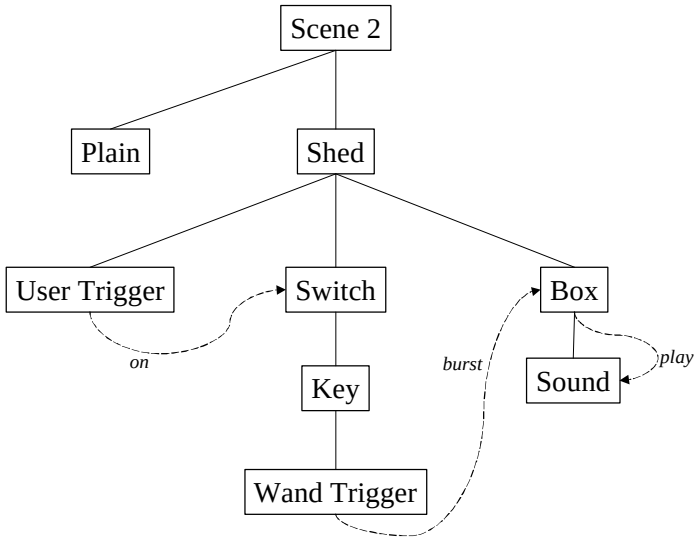

Figure 4. Fragment of a scene graph

One feature of the event/message definition that has proven very useful is delays. For any event/message combination in the scene file, the message can be given a delay; the message will then be sent the given number of seconds after the event occurs, rather than instantaneously. In many applications, especially a narrative, the author will often have a set of actions, among multiple objects, which should occur in a particular, scripted order. For instance, when the user clicks on the key, it animates, then the box bursts open and a sound plays, rocks fly out and land at various places on the plain, and finally the Thing emerges and introduces itself to the user. This is implemented by a single trigger that detects when the user clicks the key; a short "script" of messages then queues up all the succeeding actions with their pre-planned timing.

The standard classes that are part of the basic system include *transformation*, *switch*, *object*, *light*, *sound*, and a set of *trigger* classes which respond to user actions. *transformation* nodes are used to translate, rotate, and scale the parts of the world which are under them in the scene graph. By default, they are static, but subclasses are often defined to create dynamic transformations, such as playing back key-framed animations. A *switch* node is used to turn parts of the virtual world on or off at run-time, such as in a transition between scenes. *object* nodes encapsulate 3D object models, which can be in any modeling format supported by Performer. Performer provides database-loaders for a number of common formats, and new custom-built ones can be easily added; because many of the artists we work with use the Softimage animation package, we created a loader for Softimage's hrc format, which was not part of the standard Performer loaders. *object* nodes have a number of options, which include being grabbable (i.e. a user can pick up and drop the object), being used for collision detection, or being used for terrain-following. They can also be marked as un-drawn, in the case of objects which are solely intended to control the user's movement. *sound* nodes contain audio clips which can be played in response to messages; being a part of the scene graph, they have a 3 dimensional position, and their amplitude can be varied based on the user's distance from the sound source. Many systems which implement 3D sounds, such as VRML [10], attempt to model these sounds realistically. That is, they define sounds as point sources, whose amplitude decays in a spherical or ellipsoid pattern around the point. In our case, rather than focusing on strict realism, we have added features that are useful to artists in creating their environments – we extended the model of sounds to allow them to occupy volumes (spheres or boxes). Within a sound's volume, its amplitude is constant; outside the volume, the amplitude decays normally. This makes it simpler to create such things as a sound that is emitted uniformly by a large object, or a background sound that fills an entire room. The *trigger* classes detect user actions, and are used for much of the basic interaction in environments. They detect events such as the user entering or exiting a region, the wand entering or exiting a region, a button being pressed while the wand is within a region, or the user pointing at something.

Many nodes also have a debugging state, which is used during development and testing. When debugging is enabled, additional elements are drawn, showing normally invisible aspects of the scene. For example, triggers will draw their bounding volumes, so that the developer can check their size and placement in the scene; their state changes are indicated by

changing colors. Events and significant messages are printed to the terminal, so that the flow of the application can be monitored. Figure 5 shows a view of the Multi-MegaBook environment in debugging mode.

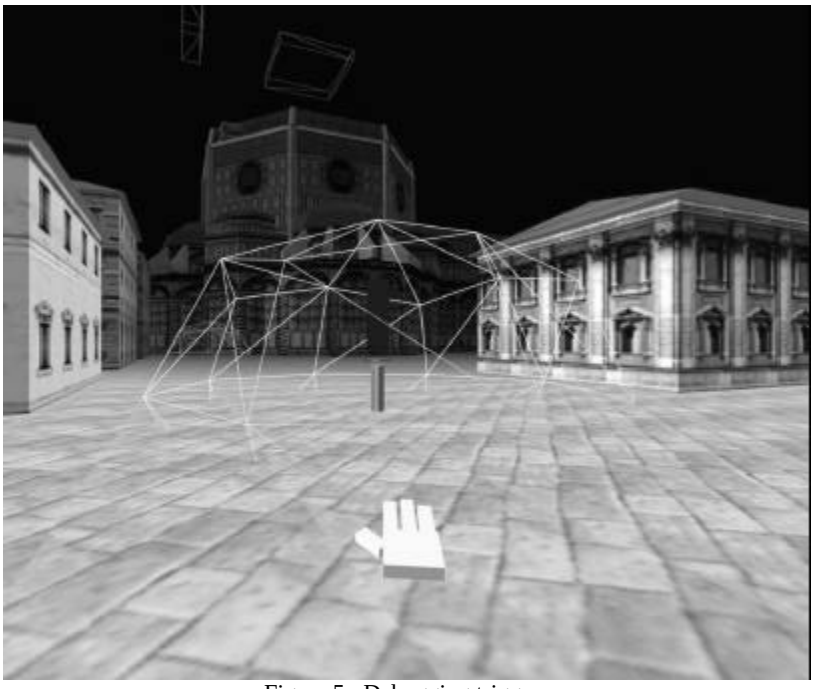

Figure 5. Debugging triggers

In addition to the scene graph defining the virtual world, other major elements of the XP framework include the navigator and world nodes, which are created automatically for any environment. The navigator is used to move the CAVE through the virtual world. It provides tools for both user-controlled and application-controlled navigation. Typically, the user travels through the world under her own volition, via the wand – she points the wand in the direction to move, and uses the joystick to control the speed of movement, or to turn left or right. In many cases, however, an application needs to take control of the user's movement. Pre-defined features for this include teleporting to a specific location, following a spline path, or attaching the CAVE to an object in the scene. The navigator node also provides optional collision-detection, to prevent users from passing through walls, and terrain-following, to keep users walking on the ground of the environment. Further application-specific features can be added by sub-classing the standard navigator node class.

The world node serves as the root of the scene graph, and provides an interface to some global attributes. It can be used to change the background sky color, enable or disable fog, and vary the clipping plane distances. It also encapsulates all of the scene text file parsing code, and controls the scene graph traversals.

3.3 Extending XP for The Thing Growing

The XP message system was invaluable for constructing the narrative backbone of the Thing Growing. The narrative structure was created with the text files and scripts. Timed sequences were intercut with the interactive episodes. The narrative flow as a whole was structured using triggers based on time, proximity, or the completion of specific events. The text file serves as production manager for the story, which can therefore be easily edited and changed.

Much of the extension of XP for this application lay in creating classes with autonomous behavior for moving objects in the VR environment. The rocks that chase the user are one example. They have to have enough intelligence to know where the user is, they have to avoid each other, and they have to sneak up on the user and try to trap her. We extended the basic XP transform class and made a rock-object class which kept a list of all other rock-objects and programmed them with a set of rules on how to move until one grabs the user. When that happens, all the other rocks scatter and a message is sent to the navigator to disable navigation. The user is trapped with a rock slobbering on her.

More specifically, we built XP classes for the creation of the virtual character, the Thing. Essentially, programming for the Thing had two major categories – body and brain. Visually, the Thing is very simply a collection of detached translucent triangles, one for the head, two for arms, one for a body and four or five for a tail. It is animated using motion capture. The life-like movement that results causes the user to sketch in the lines of a dragon-like creature. The Thing is a speaking creature. Based on the initial story board, we recorded many short phrases for its voice. The animation was done in time to these phrases. The motion-captured movements describe the motion of the Thing's limbs as it stands in one place. In addition we needed to create movement for its body as a whole – depending on whether we wanted the Thing to move relative to the user or move autonomously about the virtual environment.

This process built up a library of "actions" for the Thing. Each action consists of three parts; the phrase, the motion-captured movement and the body-as-a-whole movement. The brain's job is to select an appropriate action according to the point in the narrative, the user's actions, and the Thing's own emotional state. Then it passes the information on to the Thing's body parts, to its voice, and to its global body, so that they all execute this action. As the program runs, the body interpolates between the end of one action and the beginning of the next, so that the switch between actions is fluid. The actions, and parts of actions, can be added in the text file so it is very simple to add, remove or alter actions and essentially to edit the Thing's behavior.

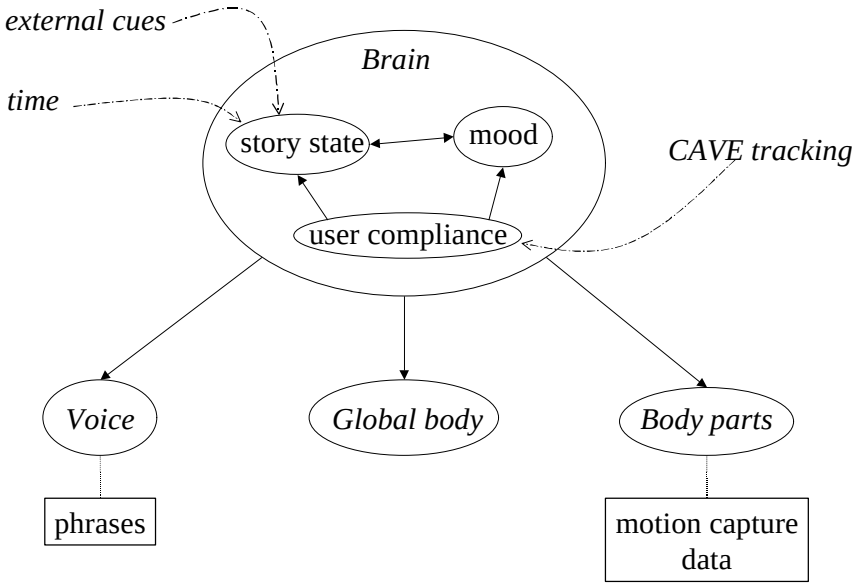

Figure 6. The Thing's brain

In order to be able to respond quickly to changing situations, the brain has several basic strategies. Certain state changes will send it a message to interrupt its current action – the specific state change will also send an additional message to indicate which kind of action should now be picked. Otherwise the brain will complete its action, go through a series of checks on the state of the world and the user, and if none of these trigger alternative actions follow an internal set of rules for selecting the next action.

The narrative becomes a very useful tool for constraining the kind of action the brain can pick, thus simplifying the rule structure. For example, when the Thing is attempting to teach the user to dance, it has a basic routine to follow. It demonstrates each part of the dance, then observes or joins the user as she copies the movement. Information on whether the user is dancing correctly is fed back to its decision-making process. It may repeat a part of the dance that the user is doing incorrectly. It may admonish, encourage or praise the user according to the user's behavior and its own mood. Then it will teach another step. This routine is interrupted if the user tries to run away and behavior is triggered to make the Thing run after the user and plead with or scold her to continue the dance.

4. SHOWING VIRTUAL ART

Creating an art or cultural heritage project for a CAVE is challenging enough, but showing the project brings up another set of difficulties. Most of these problems stem from the fact that the CAVE is still basically a research device, and not a more mainstream medium such as video or even the World Wide Web. CAVEs are very expensive. In addition to the cost of the basic structure, the video projectors, and the high-end graphics workstation needed to drive a multi-screen stereoscopic display, there is the architectural cost. A standard 10x10 foot CAVE with projections on 3 walls and the floor requires a 20x30 foot space with a 14 foot high ceiling; this space must also be light-controlled because of the relative dimness of CRT projectors. Just allocating such an area is a difficult task for many institutions. As a result, access to CAVEs is very limited, both for developers and for audiences. Smaller scale devices like the ImmersaDesk and Responsive Workbench specifically address the problems of cost and space, although they sacrifice the fully immersive nature of the CAVE to do so.[11][12] As high-tech museums begin to acquire CAVEs and similar devices, they will become more accessible; currently, the Ars Electronic Center in Linz, Austria, the NTT InterCommunication Center in Tokyo, and the Foundation for the Hellenic World in Athens are the only institutions providing public access to CAVEs.

Other problems facing artists and museums wishing to use CAVEs include fragility, obsolescence, and tracker limitations. The shutter glasses and projection screens are fairly fragile and expensive; glasses can cost a few hundred dollars apiece, and are always at risk of being dropped by visitors; a minor accident in a CAVE can ruin a screen and require its complete replacement. A CAVE is usually driven by a top-of-the-line graphics computer; any such machine is usually obsolete within a couple years, and so will either need regular, large purchases to update equipment, or require content developers to restrict themselves to the capabilities of older systems. Although the CAVE is described as being unencumbering in comparison to a typical HMD, trackers must still be attached to the glasses and wand, and these trackers have wires that can easily get tangled and in the way of users. Current electromagnetic trackers are also of limited quality – the tracking range is typically restricted to a few feet, beyond which errors can become large; and they suffer from noticeable latencies, which can detract from the quality of a virtual experience.

The quality of experience in the CAVE is also affected by time pressures. Museums for art and culture that use VR display devices share logistical problems with centers such as DisneyQuest, which has both CAVE-like and HMD VR experiences. In both cases the popularity and expense of VR means that there is pressure to put many people through the experience quickly and expeditiously. In the case of DisneyQuest this results in stimulating four to five minute experiences in which the interaction is more shadow than substance. The museum spaces typically allow a longer time, but the general public does not normally have experience with high-end VR, and often must be guided through the experience. Audience members can also vary widely; the hardware is often not built with small children or handicapped users in mind. Currently VR audiences very often get a very satisfactory visual and immersive experience, but the interactive potential of VR is sacrificed – either by the creators of the virtual world themselves or by the exigencies of showtime. Finally, since VR experiences are still relatively new, most users are novices and the sheer pyrotechnics of the immersive technology – the "wow" factor – can for good or ill completely overwhelm the specificity of an art piece.

Figure 7. Typical CAVE crowd

Figure 8. Smaller users

The XP system addresses some fundamental logistical showtime issues. Typically audience members experience the virtual world in groups. Applications which have a specific narrative or flow need to be re-initialized whenever a new user, or group of users, enters the environment. Because of the scale of the applications we have been working on, we cannot simply exit the program and reload everything from scratch – long delays are not acceptable at showtime. Therefore, all nodes include a method to instantly reset them to their initial state. Also, inexperienced users can sometimes lose their way in an application, or can find unanticipated holes in the environment; to rescue them in these cases, the navigation code allows the user to immediately pop back to a fixed starting location, or to temporarily disable collision detection to get out of a "trap". In other cases, we can anticipate difficulties users will have, and design the application to help them. For example, precise navigation can be very hard for people who have never used a wand before; many novice users would have spent a long time just trying to get through the small door of the shed at the start of The Thing Growing if they were on their own. But instead, we detect when the user has gotten close to the door, and (very briefly) automatically guide them through it on a fixed path. With occasional assists such as this, users can focus more on the intended experience, instead of being frustrated by the technology.

The Thing Growing depends for its meaning on the interaction, the growing relationship, between the user and the Thing. Therefore, once she has been given a very basic introduction to the hardware, the user is on her own. This means that the story elements – narration and character dialogue – have had to be designed to guide users through any possible confusion created by the novel technology. Our experiences showing the project have convinced us that it is possible to hand the control of the virtual environment over to a novice user and to make the interaction central to the experience.

5. CONCLUSION

We have successfully shown The Thing Growing in the CAVE, on an ImmersaDesk, on a Barco Baron and using a Panoram display – in the last two cases we added tracking systems to the basic display system. We have been gratified by the reaction of audiences. Feedback from users leads us to believe that they do become involved with the Thing and feel immersed in the narrative as a protagonist. However, some participants have indicated that they feel pressure from the audience that typically surrounds the tracked user in a show setting. We would therefore prefer it if the experience could be one on one.

We have started to experiment with the XP system in Linux on commodity PC systems using the Ascension Technologies SpacePad tracking system. Although a museum or gallery space may be unable to afford a CAVE, they might be able to put on a VR show with a one wall projection screen plus low cost tracking and PCs. If we substitute an HMD for the projection screen – even the creation of home VR systems is a possibility. The economics of this scenario make it much more possible for a one on one VR experience to become a reality. However, VR environments and displays will have to be much more robust, intelligent and able to explain themselves before they can be left to fend for themselves in a gallery. We are therefore also interested in developing stand alone VR systems with these characteristics.

Cheaper systems are not the answer for every artist working in VR since the quality of the image will suffer. However, despite the expense, an increasing numbers of art organizations are interested in using VR, especially for the display of cultural heritage. We believe that authoring systems like XP system will play an important part both in the creation of innovative art projects and in moving culturally-based VR experiences beyond fly-throughs of reconstructed ruins. Our experience indicates ways to increase interaction, use narrative as an extra immersive component, and deploy virtual characters to enrich these experiences.

ACKNOWLEDGMENTS

The virtual reality research, collaborations, and outreach programs at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago are made possible by major funding from the National Science Foundation (NSF), awards EIA-9802090, EIA-9871058, ANI-9712283, ANI-9730202, and ACI-9418068, as well as NSF Partnerships for Advanced Computational Infrastructure (PACI) cooperative agreement ACI-9619019 to the National Computational Science Alliance. EVL also receives major funding from the US Department of Energy (DOE), awards 99ER25388 and 99ER25405, as well as support from the DOE's Accelerated Strategic Computing Initiative (ASCI) Data and Visualization Corridor program. In addition, EVL receives funding from Pacific Interface on behalf of NTT Optical Network Systems Laboratory in Japan. CAVE and ImmersaDesk are trademarks of the Board of Trustees of the University of Illinois.

REFERENCES

1. Char Davies, John Harrison, “Osmose: Towards Broadening the Aesthetics of Virtual Reality”, *Computer Graphics* 30 (4), pp. 25-28, ACM SIGGRAPH, November 1996.
2. Maurice Benayoun, “WorldSkin: A photo-safari in the land of war”, http://www.benayoun.com/Worskieng.htm.
3. Maria Roussou, “Incorporating Immersive Projection based VR in Public Spaces”, *Proceedings of the 3rd Immersive Projection Technology Workshop (IPTW ’99)*, Stuttgart, Germany, pp. 33-39, May 1999.
4. C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. “Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE”. *Proceedings of SIGGRAPH '93 Computer Graphics Conference*, pp. 135-142, ACM SIGGRAPH, August 1993.
5. Dave Pape, Tomoko Imai, Josephine Anstey, Maria Roussou, Tom DeFanti, “XP: An Authoring System for Immersive Art Exhibitions”, *Proceedings of VSMM ’98*, Gifu, Japan, November 1998.
6. F. Fischnaller and Y. Singh. “Multi-MegaBook”. *Catalog of Ars Electronica Festival ’97*, Linz, Austria, September 1997.
7. M. Roussos and H. Bizri. “Mitologies: Medieval Labyrinth Narratives in Virtual Reality”. *Proceedings of 1st International Conference on Virtual Worlds*, Paris, France, July 1998.
8. John Rohlf, Jim Helman. “IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics”. *Proceedings of SIGGRAPH ’94 Computer Graphics Conference*, pp. 381-395, ACM SIGGRAPH, August 1994.
9. UVa User Interface Group. “Alice: Rapid Prototyping for Virtual Reality”. *IEEE Computer Graphics and Applications*, pp. 8-11, May 1995.
10. The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997.
11. M. Czernuszenko, D. Pape, D. J. Sandin, T. A. DeFanti, G. L. Dawe, M. Brown. “The ImmersaDesk and Infinity Wall Projection-Based Virtual Reality Displays”. *Computer Graphics*, Vol. 31 No. 2, pp. 46-49, May 1997.
12. Wolfgang Krueger, Bernd Froehlich, “The Responsive Workbench”, *Computer Graphics and Applications*, Vol. 14, No. 3, pp. 12-15, May 1994.