Tommi Ilmonen
Helsinki University of Technology
Telecommunications Software and Multimedia Laboratory
Tommi.Ilmonen@hut.fi

IMMERSIVE 3D USER INTERFACE FOR COMPUTER ANIMATION CONTROL

Abstract

This paper describes a virtual reality -based method to control 3D animation. We chose a topic that is particularly difficult to control with traditional desktop applications – a particle system. Particle systems have the potential to create high-quality computer graphics, but they need lots of control data if one wants to create control their dynamics in detail. We have developed a new – more natural – way to control a particle system by working directly in 3D virtual environment. By working in an interactive 3D virtual environment animators can spend more time on adjusting and tuning the animation and less time on entering control data and learning how the system works.

keywords: *Particle systems, Immersive user interface*

1 INTRODUCTION

This paper proposes a new way to create 3D animations. The topic of this research is animation creation with direct manipulation user interface. As a case study we have built an application that gives the animator control of a particle system. The aim of this study is to find better ways to create 3D animations.

Our approach is to use virtual reality (VR) system as an interface to control the behaviour of a particle system. With this application the user controls the animations directly in life-size 3D environment and gets instant visual feedback (see figure 1). In our application – AnimaLand – the user can control the forces that affect a particle system and also add new particle creators and killers to the system. We use several interface devices (motion tracker, data glove, wireless mouse) simultaneously to gather the necessary input data. The input data is then mapped to animation controls in real-time.

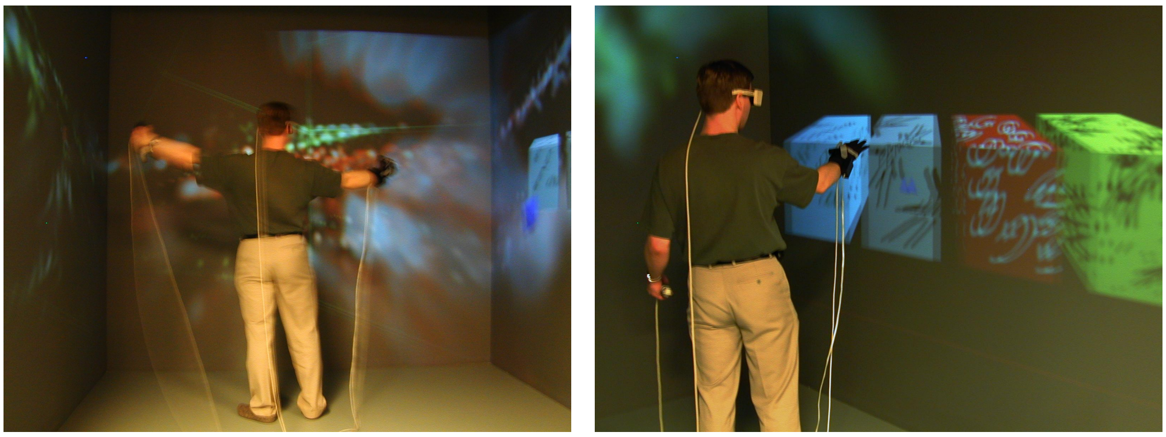
(a) The user is working on the animation. (b) The user is selecting a tool.

Fig. 1 . Different ways to interact with the AnimaLand system.

3D content creation is very slow and expensive with current technology. One reason reason for the high price is in the huge amount of hand-work that must be done to create the models and animation paths.

In computer graphics and animation one of the primary problems is that the user must supply plenty of data to the system. When dealing with 3D graphics the problems are intensified. Traditional desktop computing environments can provide the users with 3D graphics, but fail to offer 3D input devices. Unfortunately artists are mostly forced to use normal keyboards and mice to create 3D content. These input tools work fairly well for 2D data or text strings, but they are not optimal for operations in 3D environment. To specify a location in 3D – the most primitive operation in 3D computer graphics – one must typically operate multiple windows of an application and even use keyboard shortcuts. By operating directly in a virtual 3D environment we can avoid the bottle-neck caused by such 2D-bound user interface. The application does not need to convert 2D controls to 3D actions and the user does not need to learn how to enter 3D data with 2D user interface. It should be noted that a 3D user interface does not solve all problems and it even creates new challenges. Our approach is to use the 3D user interface for the work it best suited for and use it as one of the many tools in animation production.

The application area for this work is in animation creation for 3D productions.

2 BACKGROUND

The most usual way to control animation is the use of key-frames. This approach has been used successfully over the past decades. The problem with key-framing is the slowness of the work – it takes a lot of effort to specify the optimal key-frames.

To counter these problems researchers have tried to create better ways to control animation. An example is Laszlo's work on creating an interface to animate limbed bodies[1]. Their approach was to give the user direct control over some of the joint control parameters. This is an state-of-the-art system that demonstrates well the problems that animators face. With their system users can control animation with 10-40 slow-down factor.

Another trend in computer animation is the creation of tools that create better animations with less control data. Approaches like this try get away with fewer keyframes or get rid of keyframes altogether. For example Popović has published a system that combines physical simulation with animation constraints given by the user[2]. Methods like this try to decrease the need for human intervention in the animation process. In contrast our system makes it easy to apply lots of human intervention in the animations.

As far as we know our project is the first one to target animation control with virtual reality user interface. A closely related topic is the creation of 3D models with a VR system. This has been done by previously by other researchers. Keefe has reported a system for painting in 3D[3]. Hartman has also presented an application to create 3D models[4] with similar motion tracking approach. These two projects were aimed at creating models in immersive 3D environment and they do not address the problem of controlling an animation.

Our application – AnimaLand – is a tool to control the operation of a particle system in real-time. Particle systems are usually controlled by moving the forces that act in the scene and by moving the particle sources and sinks. This way the animator can control a group of particles without needing to control each particle individually. Particle systems have been around for decades and there are even "build your own particle system" -articles in journals and magazines. While coding a particle system is straightforward business the user interfaces to control particle systems are often difficult to use. The traditional method to set the parameters of the forces is via text files or direct coding. Modern computer graphics applications can also control particle systems with textures (for example Blender Creator)While these improvements make the particle systems easier to control it they do not remove the problem that a particle system requires a lot of control data if it is to be controlled precisely.

As far as animation of a particle system is concerned a close analogy can be found from Wejchert's work[5]. Their system was designed to enable the user to control particle behaviour by controlling the physically modeled forces in the scene. By controlling just a few forces in the scene the user could indirectly control the paths of several particle-like objects. This is in fact close to what the user does with our application. One can consider our application to be a continuation for Wejchert's work with the following differences: The user 1) works in an immersive VR environment, 2) controls the system via a direct-manipulation interface 3)can create new visible objects (particles) to the scene, 4) has real-time view of what is happening in the system, 5) can perform multiple passes to create very complex systems and 6) gets audio feedback from the system.

3 THE APPLICATION

AnimaLand is a virtual reality application that takes the animator to the virtual animation world. The animator has a collection of tools that are used to create particles and control their behaviour. These tools are applied to the act that is being animated in real-time.

The applications takes its input from user's gestures – hand and finger motion. The hand motion is measured with magnetic motion tracker and the finger movements are measured with data glove. To make the environment truly immersive the user's head is tracked and the images on the wall are rendered to match the head motion. By using stereoscopic rendering and stereo glasses we get even better immersion. A photograph of a real situation is in figure 1.

The artist selects tools and controls their parameters in the 3D environment with his own motion. The created animation is played in real-time and new controls take effect instantly. More complex animations are created by playing the act many times and adding new animations as needed.

This method to create animations is simple to use: the user does not need to learn a great number of special keystrokes or application-specific parameter mappings.

3.1 Multipass Animation and Action Lists

The animations are created as acts – short animation sequences with duration ranging from seconds to minutes. An act might need a lot of work – several moving forces and particle sources. A user cannot create all of them in one pass. To overcome this problem the animations are constructed by running multiple passes over the act (analogous to multitrack recording in music production). The user can work on the same act many times, adding new animation control at each pass.

Each pass over the act generates a new *action list*. Action list is a collection of commands that are executed in sequential order. The application interprets the user input and stores the animation commands to such lists. In each animation passage the user creates a new list of actions that are applied to the system. These lists can contain any commands to the particle system.

3.2 Interaction Modes

There are two basic operating modes in the AnimaLand. The first is tool selection mode. In this mode we place virtual toolboxes to scene and tools are selected by dipping the hand to the box (see figure 1(b)). The user can also disable or enable action lists. Thus if one pass generated a poor action list the list can be ignored later on. We use audio feedback to given the user verification that a tool was successfully selected or that action list was disabled/enabled.

The other mode is the animation creation mode. In this mode the user is using the selected tool. Each tool has its own set of parameters that are controlled in real-time. Usually the right hand indicates the location of the tool and more parameters are given with the glove and the location of the left hand.

At all modes the system displays a few extra objects in the scene. The locations of the hands are renderer with small pyramids. In theory the virtual and real world should be perfectly superpositioned but unfortunately this not the case due to tracking errors. The pyramids show where in the virtual world the user's hands are. Minimal hand markers are also useful in situations where the hand is masked by other object. If we only relied on the real hands of the user then one gets disorienting problems when one's hand should be inside or behind an object, but it is still visible in front of the object. The forces are represented by simple line graphics to indicate how the forces are moving. This is necessary especially when there are several forces in the scene and one would like to know what the forces are and where exactly are they (example in figure 2).

3.3 Tools

The animation tools in AnimaLand fall into two categories. The first is the collection of forces. Force tools (or objects) push the particles to some direction and thus cause the animation to happen (for example wind, hair-dryer, explosion, gravity). Particle tools in turn create and delete particles from the act (particle sources, and particle killers).

Available tools:

- Wind – Global force that pushes all particles towards given direction at constant speed
- Blow – A force object that blows particles away (like hair-dryer)
- Explosion – A force object that repels particles away, only lives for a short period of time

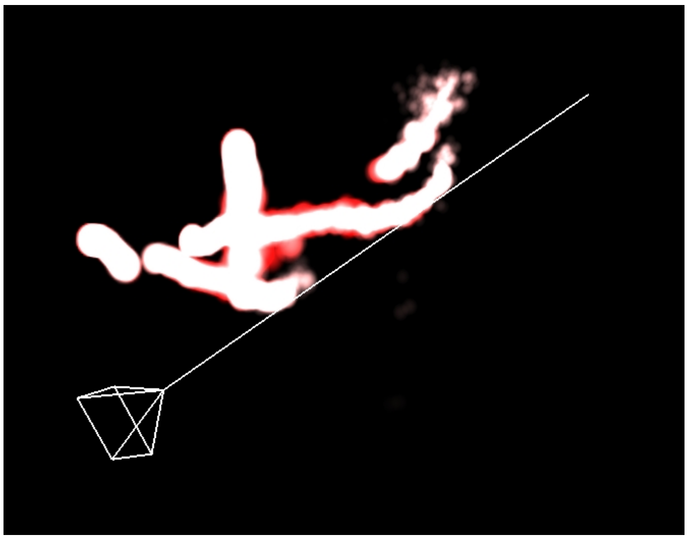

Fig. 2 . A screen shot with a blow-object in action.

- Sink – Collect particles to one point
- Vortex – A swirl around given axis with given speed
- Sphere source – Creates stationary particles within a sphere
- Disk source – Creates stationary particles to a disk
- Spherical emitter – Emits particles from one point to all directions
- Directed emitter – Emits particles from one point to a given direction

Each tool comes with a set of parameters that it can accept. We have tried to create easy mappings from user motion to the parameters. These parameters and their control mappings are hardcoded and there is currently no easy way to change them.

3.4 Audio Feedback

Often application needs to give user some information that is not visible in the scene. For example when changing the tool it is nice to know that the selection was successful. In these situations we use audio feedback to help the user. One could deliver the same information visually, but that would lead to visual clutter. To maximize the visibility of the animation we must minimize the number of visual components that are not actual part of the scene.

The audio feedback comes from audio files. Each tool has its own special sound that is played as the tool is chosen.

3.5 The Animation Pipeline

Creating the control data is the first step in creating the animation. There are other steps that need to be taken as well. The control data is stored to the sidk in form as a series of action lists. Then one runs a utility application that recreates and converts the action lists to a particle system and to an animation file (in Renderman RIB-format). In this phase one can change the parameters if necessary. For example we often multiply the number of particles in the system beyond what one could run in the real-time environment. The final animations are rendered with a Renderman-compatible rendering engine. We have used the Blue Moon Rendering Tools (BMRT) by Larry Gritz[6]. Figure 3 shows frames from the final animations that have been created.

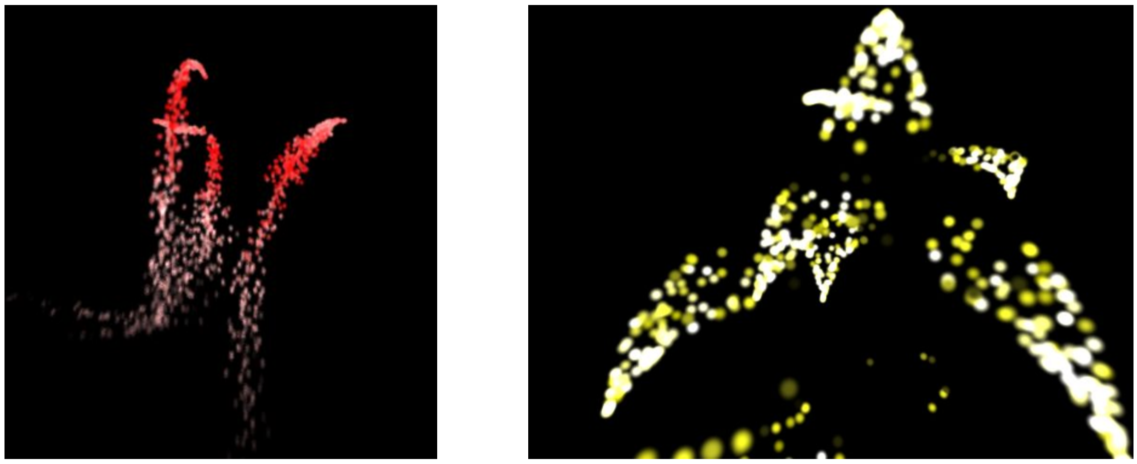

Fig. 3. Examples of the final renderings. Each particle source has been guided manually.

The final off-line rendering enables us to get high-quality animations of the particle system. We can also use much more particles in that case (there is a strict limit on how complex systems one can run in real-time).

4 BENEFITS

We have only lately built the system. Thus we do not have comprehensive understanding of it's strengths and weaknesses. There are two kinds of benefits that we predict: 1) Improved usability (easier to learn, faster to use, more fun). 2) Improved final animations.

We have not yet conducted serious user tests to formally verify that the system is superior to desktop applications. With our limited experience we can say that most people like to create animations with this method. It also seems that the tools are easy to learn – for most people it takes less than one minute to learn how a particular tool works.

In future we also hope to build high-quality animations with the system. Our artists have indicated that they believe the final outcome will be different from what one would get with desktop animation system. This is another question that we would like to verify and also see whether the increased amount of human input is visible in the final animations.

At any case we can safely say that the system is fast. If one wants to create an act with ten moving particle sources and five moving forces, each with a few parameters controlled in real-time, the animation work takes exactly fifteen times the length of the act. We do not believe this would be possible with any desktop system. In practice one does not create quality animations that quickly with AnimaLand either. The animations must be planned before starting the work and the post-processing takes work and time.

It is possible to use AnimaLand in conjunction with other animation tools that work with the supported graphics back-ends (currently OpenGL and Renderman) and new graphics back ends can be easily written. Thus there is no need to turn AnimaLand into full animation package, but it can be used as an instrument to create animations that would be difficult to create with other methods.

5 PROBLEMS

An inevitable problem for the animator is that the complexity of the act dictates how fast the particle system can be calculated and rendered in real-time. If one creates an act that is complex enough (lots of forces and particles) then eventually the frame rate will drop below comfortable operation (VR systems must have a reasonable frame rate or the users will experience disorientation). There is little that an application programmer can do about this – besides optimize the software and try to simplify the real-time animations as much as possible (and buy more hardware). This problem does not directly affect the final animations that are calculated off-line, but it does reduce work ergonomics which in turn may have adverse effects on the quality of the animations.

Another problem is that there is no easy way to combine 2D and 3D user interfaces. There are operations that are easy to do with normal desktop system – for example select file names. For these operations the virtual reality environment is difficult to use. A potential solution would be a system that could run both 2D and 3D user interfaces at the same time.

With the current very high prices of virtual reality installations AnimaLand cannot become cost-effective. This problem is reduced by the fact that animator do not need to own the virtual environment, typically they would rent a relevant installation and use it to create the particle animations. Also the price of such installations is dropping continuously. Thus they may become much more affordable in the future.

6 TECHNICAL DETAILS

AnimaLand has been created to run in our virtual room EVE. EVE is a typical CAVE-like virtual room with four active walls. The graphics are generated with an SGI Onyx2 graphics workstation.

Our software has been written with the C++ programming language. All real-time rendering has is done with the OpenGL. We use the VR Juggler[7] toolkit to manage the rendering process (wall projections, threading) in the virtual room. To handle the input devices we use our own input device toolkit – FLexible User Input Design (FLUID)[8].

The audio feedback was done by using a modular signal processing application Mustajuuri[9] as a sound server. The server runs on a dedicated Linux-PC and produces sound for 15 loudspeakers that are in the EVE. AnimaLand sends the necessary control data to the server via normal socket connection. The server then synthesizes the sound, places it in the 3D scene and compensates for the damping of the screens[10].

7 CONCLUSIONS AND FUTURE GUIDELINES

We have presented a method to control 3D animation with a virtual reality interface. Our approach combines the strengths of direct manipulation 3D user interface and traditional particle systems. We have used particle system as a case for animation. We believe this system generalizes to other animation forms as well. In particular the animation forms where users must specify trajectory data are suitable for direct manipulation. The difficult cases can probably be found when dealing with complex hierarchical models with several movement constraints (for example human motion).

While AnimaLand can be used to create interesting particle animations it is still research software. If it was to become a tool for real-world animation projects it would need more features. Such features would be selectable animation speed, easier customization and better integration to other animation tools.

We could develop the software further to be a tool for real animation projects. Although there are many possible routes for technical refinement we feel that the human factors involved in audio-visual 3D interfaces are a more general and interesting topic. For such work AnimaLand is a good test-bench. In future we plan to evaluate the immersive direct manipulation interface in more detail. This calls for thorough examination on how users relate to the system and how the interaction works. On the technical front an interesting case would be the combination of physical modeling and direct manipulation.

We do not expect that AnimaLand could solve all the problems in 3D animation. It is more appropriately seen as a new tool that can be used with other tools – each tool doing the job it is best for.

REFERENCES

[1] Laszlo, Joseph , van de Panne, Michiel , and Fiume, Eugene . Interactive control for physically-based animation. In Proceedings of the Conference on Computer Graphics, pages 201–208, Colorado Springs, Colorado, United States, 2000. ACM.

[2] Popović, Jova , Seitz, Steven , Erdmann, Michael , Popović, Zoran , and Witkin, Andrew . Interactive Manipulation of Rigid Body Simulations. In Proceedings of the conference on Computer graphics, pages 209–217, Colorado Springs, Colorado, United States, 2000. ACM.

[3] Keefe, Daniel , Feliz, Daniel , Moscovich, Tomer , Laidlaw, David , and LaViola, Joseph . CavePainting: a fully immersive 3D artistic medium and interactive experience. In Proceedings on 2001 Symposium on Interactive 3D graphics, pages 85–93. ACM Press New York, NY, USA, 2001.

[4] Hartman, Chris and Brody, Bill . Navigating Space by Drawing. In Visual Data Exploration and Analysis VIII, Proceedings of SPIE, pages 251–258, 2001.

[5] Wejchert, Jakub and Haumann, David . Animation aerodynamics. ACM SIGGRAPH Computer Graphics, 25:19–22, July 1991.

[6] Gritz, Larry and Hahn, James K. . BMRT: A Global Illumination Implementation of the RenderMan Standard. Journal of Graphics Tools, 1(3):29–47, 1996.

[7] Bierbaum, Allen , Just, Christopher , Hartling, Patrick , Meinert, Kevin , Baker, Albert , and Cruz-Neira, Carolina . VR Juggler: A Virtual Platform for Virtual Reality Application Development. In The Proceedings of IEEE VR Conference 2001, 2001.

[8] Ilmonen, Tommi and Kontkanen, Janne . Software Architecture for Multimodal User Input – FLUID. In The Proceedings of the 7th ERCIM Workshop, October 2002 (to be published).

[9] Ilmonen, Tommi . Mustajuuri - An Application and Toolkit for Interactive Audio processing. In Proceedings of the 7th International Conference on Auditory Displays, pages 284–285, 2001.

[10] Hiipakka, Jarmo , Ilmonen, Tommi , Lokki, Tapio , Gröhn, Matti , and Savioja, Lauri . Implementation issues of 3D audio in a virtual room”. In Proc. 13th Symposium of IS&T/SPIE, Electronic Imaging 2001, volume 4297B (The Engineering Reality of Virtual Reality), pages 486–495, 2001.