USING VIRTUAL REALITY TECHNIQUES IN THE ANIMATION PROCESS

Daniel Thalmann
Computer Graphics Lab
Swiss Federal Institute of Technology
CH-1015 Lausanne
Switzerland

Abstract

This paper tries to find the various functions involved in an animation system and how virtual reality techniques and multimedia input could play a role. A classification of VR-based methods is proposed: real-time rotoscopy methods, real-time direct metaphors and real-time recognition-based metaphors. Several examples are presented: 3D shape creation, camera motion, body motion control, hand animation, facial animation. The hardware and software architecture of our animation system is also described.

1. Introduction

The traditional main difficulty in the process of 3D animation is the lack of 3D interaction. Visual feedback, in a typical computer graphics application that requires items to be positioned or moved in 3-D space, usually consists of a few orthogonal and perspective projection views of the same object in a multiple window format. This layout may be welcomed in a CAD system where, in particular, an engineer might want to create fairly smooth and regular shapes and then acquire some quantitative information about his design. But in 3-D applications like 3D animation where highly irregular shapes are created and altered in a purely visual and esthetic fashion, like in sculpting or keyframe positioning, this window layout creates a virtually unsolvable puzzle for the brain and makes it very difficult (if not impossible) for the user of such interfaces to fully understand his work and to decide where further alterations should be made. Moreover good feedback of the motion is almost impossible making the evaluation of the motion quality very difficult.

For a long time, we could observe virtual worlds only through the window of the workstation's screen with a very limited interaction possibility. Today, new technologies may immerse us in these computer-generated worlds or at least communicate with them using specific devices. In particular, with the existence of graphics workstations able to display complex scenes containing several thousands of polygons at interactive speed, and with the advent of such new interactive devices as the SpaceBall, EyePhone, and DataGlove, it is possible to create applications based on a full 3-D interaction metaphor in which the specifications of deformations or motion are given in real-time. This new concepts drastically change the way of designing animation sequences.

In this paper, we call **VR-based animation** techniques all techniques based on this new way of specifying animation. We also call **VR devices** all interactive devices allowing to communicate with virtual worlds. They include classic devices like head-mounted display systems, DataGloves as well as all 3D mice or SpaceBalls. We also consider as VR devices MIDI keyboards, force-feedback devices and multimedia capabilities like real-time video input devices and even audio input devices. In the next Section, we present a summary of these various VR devices. More details may be found in (Balaguer and Mangili 1991; Brooks 1986; Fisher et al. 1986).

2. The Animation Process

Three-dimensional animation scenes usually contain static objects grouped into a decor and animated objects that change over time according to motion laws. Moreover, scenes are viewed using virtual cameras and they may be lit by synthetic light sources. These cameras and lights may evolve over time as though manipulated by cameramen. In order to create all the entities and motions, coordinate and synchronize them, known collectively as choreography, it is necessary to know the appearance of the scene at this time and then Computer Graphics techniques allow us to build and display the scene according to viewing and lighting parameters. The problems to solve are how to express time dependence in the scene, and how to make it evolve over time. Scenes involving synthetic actors imply more complex problems to manage. Human-like synthetic actors have a very irregular shape hard to built especially for well-known personalities. Once the initial human shape has been created, this shape should change during the animation. This is a very complex problem to ensure the continuity and the realism of the deformed surfaces. The human animation is very complex and should be split into body motion control and facial animation. Basically a synthetic actor is structured as an articulated body defined by a skeleton. Skeleton animation consists in animating joint angles. There are two main ways to do that: parametric keyframe animation and physics-based animation. An ultimate objective therefore is to model human facial anatomy exactly including its movements to satisfy both structural and functional aspects of simulation.

During the creating process, the animator should enter a lot of data into the computer. The input data may be of various nature:

- **geometric:** 3D positions, 3D orientations, trajectories, shapes, deformations
- **kinematics:** velocities, accelerations, gestures
- **dynamics:** forces and torques in physics-based animation
- lights and colors
- sounds
- commands

The following table shows VR-devices with corresponding input data:

VR-device	input data	application
DataGlove	positions, orientations, trajectories, gestures, commands,	hand animation
DataSuit	Body positions, gestures	body animation
6D mouse	positions, orientations	shape creation, keyframe
SpaceBall	positions, orientations, forces	camera motion,
MIDI keyboard	multi-dimensional data	facial animation
Stereo display	3D perception	camera motion, positioning
Head-mounted display (EyePhone)	camera positions and trajectories	camera motion
Force transducers	forces, torques	physics-based animation
Real-time video input	shapes	facial animation
Real-time audio input	sounds, speech	facial animation (speech)

3. A Classification of VR-based Methods for Animation

3.1 Real-time rotoscopy methods

Traditional **rotoscopy** in animation consists of recording the motion by a specific device for each frame and using this information to generate the image by computer. For example, a human walking motion may be recorded and then applied to a computer-generated 3D character. This off-line approach will provide a very good motion, because it comes directly from reality. However, it does not bring any new concept to animation methodology, and for any new motion, it is necessary to record the reality again.

We call a **real-time rotoscopy method** a method consisting of recording input data from a VR device in real-time allowing to apply at the same time the same data to a graphics object on the screen. For example, when the animator opens the fingers 3 centimeters, the hand on the screen do exactly the same.

3.2 Real-time direct metaphors

We call a **real-time direct metaphor** a method consisting of recording input data from a VR device in real-time allowing to produce effects of different nature but corresponding to the input data. There is no analysis of the meaning of the input data. For example, when the animator presses the fourteenth key on a MIDI synthesizer, the synthetic actor's face on the screen opens his mouth depending on the pressure on the key.

An example of traditional metaphor is the puppet control. A puppet may be defined as a doll with jointed limbs moved by wires or strings. Similarly glove-puppets are dolls of which the body can be put on the hand like a glove, the arms and head being moved by the fingers of the operator. In both cases, human fingers are used to drive the motion of the puppet. This is a metaphor, as the

A strange situation that we have experimented consists in driving a virtual hand using the DataGlove. The virtual hand moves the strings of a puppet. When we consider the motion of the virtual hand, it is a typical real-time rotoscopy method, but the animation of the puppet from the DataGlove is a typical real-time direct metaphor.

The relationship between the VR device and the animated motion is not as straightforward as one might think. Usually, some sort of mathematical function or "filter" has to be placed between the raw 3-D input device data and the resulting motion parameters.

3.3 Real-time recognition-based metaphors

We call a **real-time recognition-based metaphor** a method consisting of recording input data from a VR device in real-time. The input data are analyzed. Based on the meaning of the input data, a corresponding directive is executed. For example, when the animator opens the fingers 3 centimeters, the synthetic actor's face on the screen opens his mouth 3 centimeters. The system has recognized the gesture and interpreted the meaning.

3.4. The Ball and mouse metaphor

In essence, motion parallax consists of the human brain’s ability to render a three-dimensional mental picture of an object simply from the way it moves in relation to the eye. Rotations offer the best results because key positions located on the surface move in a larger variety of directions. Furthermore, in a perspective projection, depth perception is further accentuated by the speed in which features flow in the field of view _ points located closer to the eyes move faster than the ones situated in back. In a 3-D application, if motion parallax is to be used effectively, this implies the need for uninterrupted display of object movements and thus the requirement for hardware capable of very high frame rates. To acquire this depth perception and mobility in a 3-D application, we make use of a SpaceBall.

When used in conjunction with a common 2-D mouse such that the SpaceBall is held in one hand and the mouse in the other, full three-dimensional user interaction is achieved (LeBlanc et al.,. The SpaceBall device is used to move around the object being manipulated in order to examine it from various points of view, while the mouse carries out the picking and transformation work onto a magnifying image in order to see every small detail in real time (e.g. vertex creation, primitive selection, surface deformations, cloth panel position, muscle action). In this way, the user not only sees the object from every angle but he can also apply and correct transformations from every angle interactively. In order to improve our approach using stereo display, we also use “StereoView”.

4. Virtual Tools and Virtual Studio

As shown in, Fig. 1, our example is based on scene composed of three elements: a virtual actor, a light and a camera. The example has been created using Virtual Studio, an animation environment allowing the manipulation and the animation of all aspects of synthetic worlds entirely in three dimension. *Virtual Studio*'s desktop configuration uses a *Spaceball* and a mouse as input devices, and *LCD* shutter glasses for binocular perception of the synthetic world. The *Spaceball* is used for the continuous specification of spatial transformations, while the mouse is used as a picking device. Although this configuration does not fully provide the illusion of immersion, we find it more effective for our applications than currently available virtual reality equipment.A tool and an animation controller have been associated to each element of the scene. The virtual actor is walking along a circular path, the light moves vertically in direction of the head of the actor and the camera is fixed. At the end of the actor motion, the camera moves towards the actor.

In order to guarantee that the light is always in the direction of the actor head, the animator introduces a constraint between the tool associated to the light and the actor head,

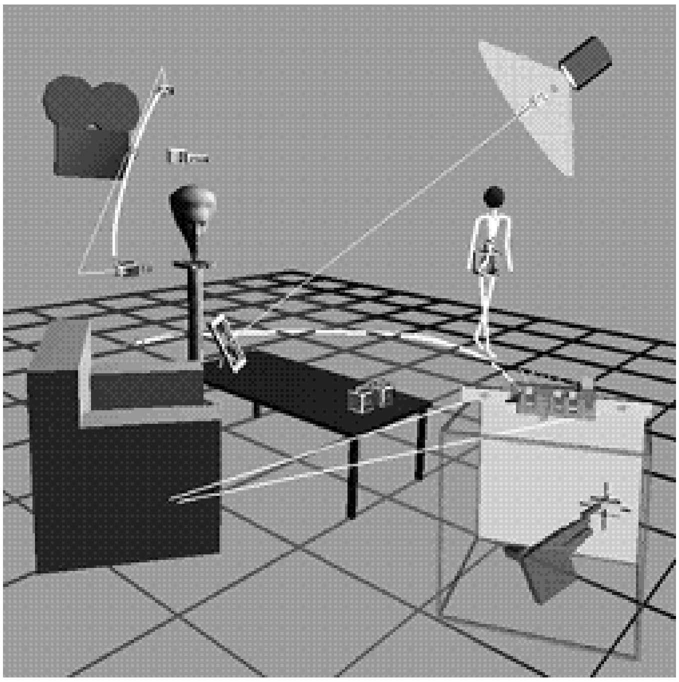

Fig.1. An animation scene with a virtual human in a virtual environment

In order to specify the actor animation, the animator uses a tool that encapsulates a motor of kinematics simulation (Boulic et al. 1991) [i]. This tool adds the walking behavior to the articulated structure of the virtual actor. When the animator moves the tool, the walking velocity is calculated from the velocity of displacement of the tool. The animation corresponding to the movement of the legs and the arms is applied to the articulated body by the simulation motor. The couple tool-model defines a functional unit (Zeltzer 1985) which the user may guide and animate by specifying goals.

The animator specifies that the actor motion and the light motion must have the same duration by connecting the ports of initial synchronization and the ports of final synchronization of the both tracks: the actor track and the light track. The animator also connects the orientation port of the camera tool to the actor's head in order that the camera always looks at the virtual actor. He also connects the port of initial synchronization of the camear to the port of final synchronization of the actor track; this means that the camera aniamtion should start when the motion of the actor is finished.

Virtual Studio has been implemented using *Virtuality Builder II* (*VB2*), a system for developing VR-based applications developed at the Swiss Federal Institute of Technology. A *VB2* application is composed of a group of continuously running processes that asynchronously produce and consume IPC messages to perform their tasks. A central application process manages the virtual world's model and simulates its evolution in response to events coming from the processes that encapsulate the input devices. The system's state and behavior are uniformly represented as a network of interrelated objects. Dynamic components are modeled by *active variables*, while multi-way relations are modeled by *hierarchical constraints*. *Daemons* are used to sequence between system states in reaction to variables' changes. Daemons register themselves with a set of active variables and are activated each time their value changes. The action taken by a daemon can be a procedure of any complexity that may create new objects, perform input/output operations, change active variables' values, manipulate the constraint graph, or activate and deactivate other daemons. Examples of *VB2*'s daemons are inverse kinematics simulation for articulated chains and scene rendering triggers.

Multiple devices are used to interact with the synthetic world through the use of various interaction paradigms, including immersive environments with visual and audio feedback. Interaction techniques range from *direct manipulation*, to *gestural input* and *three-dimensional virtual tools*. Adaptive pattern recognition is used to increase input device expressiveness by enhancing sensor data with classification information. Virtual tools, which are encapsulations of visual appearance and behavior, present a selective view of manipulated models information and offer an interaction metaphor to control it. Since virtual tools are first class objects, they can be assembled into more complex tools, much in the same way that simple tools are built on top of a modeling hierarchy. Figure 2 show three examples of virtual tools: a tool used to edit the texture mapping function of a model by controlling the parallel projection of an image on the surface of the manipulated model, a light tool and a material editing tool.

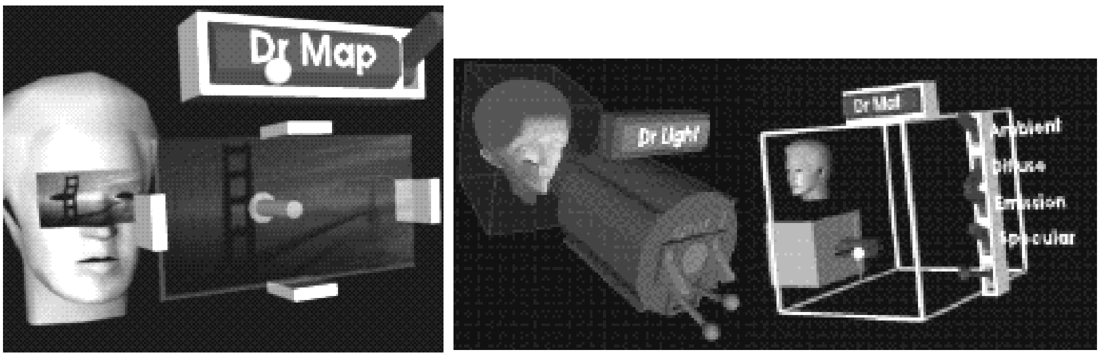

Fig.2. Virtual tools for texture mapping, lighting and material editing

References

Balaguer F, Mangili A (1991) Virtual Environments in: N.Magnenat-Thalmann and D.Thalmann (Eds) New Trends in Animation and Visualization, John Wiley and Sons, pp.91-106.

Boulic R, Magnenat Thalmann N, Thalmann D (1991) A Global Human Walking Model with Real Time Kinematic Personification, The Visual Computer, Vol.6 (6),1991

Brooks F.P. Jr (1986) Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings Proceedings 1986 Workshop on Interactive 3-D Graphics ACM, pp.9-22

Fisher S.S., McGreevy M., Humphries J., Robinett W.,(1986), "Virtual Environment Display System", Proceeding 1986 Workshop on Interactive 3-D Graphics, ACM, pp 77-87

Gobbetti E, Balaguer JF, Thalmann D (1993) VB2: An Architecture for Interaction in Synthetic Worlds, Proc. ACM UIST '93, Atlanta, 1993, pp.167-178.

LeBlanc A, Kalra P, Magnenat-Thalmann N, Thalmann D (1991) Sculpting With the "Ball & Mouse" Metaphor - Proc. Graphics Interface '91, Calgary, Canada

Zeltzer D (1985) Towards an Integrated View of 3D Computer Animation, The Visual Computer, Vol. 1, No4, pp.249-259