VHuP: a Tool to Visualize Virtual Humans

Diogo Strube de Lima, Henry Braun, Soraia Raupp Musse

Computing Science Department, Postgraduate course in Computer Science
PUCRS, Av. Ipiranga, 6681, building 32, 90619-900 Porto Alegre, RS, Brazil
{diogo.lima, henry.braun}@cpph.com.br soraia.musse@pucrs.br

Abstract

This paper presents an application which aims to easily provide visualization and post processing of virtual human simulations. The interactive interface reproduces, in a real-time frame rate (30 frames per second, FPS), a previously calculated scenario of characters simulation, including multiples characters and their trajectories, on different environments. Features to control the light sources, shadows and weather effects provide a wide range of possibilities for visualizing and video recording the simulation data. The application also facilitates the generation of analyses data, generating information for a future application, as for instance ground truth.

1. Introduction

Crowd simulation is an area of computer graphics and artificial intelligence concerned with the process of simulating a large number of models. Nowadays, virtual human crowd's simulation is an important topic of research since it is crucial for some applications as for instance, security, education and entertainment. Such simulations, as the number of 3D objects involved can be huge, are not trivial to render at high resolution on standard computers and graphic cards [1].

Current methods to simulate virtual humans and crowds are often mediated by local rules [2], forces [3], flows [4] or hybrid approaches [5]. Usually, these methods consider the characters as simplified geometries such as points, spheres or cubes. This abstraction is important to facilitate the usage of complex mathematical models, however it can create visualization artifacts when those simplified geometries are replaced by high polygon characters, that perform a variety of animations.

Current approaches [1, 5, 6] for crowd simulations are designed for a specific use, and normally the simulations prototype also includes the visualization features, limiting the integration of different algorithms. In our lab (Virtual Humans Laboratory, at PPGCC, PUCRS), we used an approach which separates the visualization from simulation aspects. In this case, the advantage was the possibility of integrating several simulators of virtual humans, by mixing their output in a same visualized scenario. Our main idea is to create an independent application for the real-time visualization of virtual human simulations. That application was named Virtual Human Player (VHuP).

The paper is organized as follows: related works are resumed on the next section, followed by section 3, which has an overview of the VHuP architecture and its key components. Section 4 presents some results obtained developing and testing the application. On section 5 the ideas for future work and new components are described.

2. Related Work

Several researches on virtual human visualization have been provided in last years. Hamill and O'Sullivan [1] resumes a number of methods available for culling such as portal, occlusion horizons, occlusion shadow volumes and the simple bounding box tests.

The vast type of environment, such indoors and outdoors, used on the crowd simulation makes common the use of multiple techniques combined. The Virtual Dublin [1] uses culling and collision techniques combined with the trade of model details for texture details, reducing the building's polygons, but increasing the memory need.

The approach of high-level wayfinding using agent's communication and roles, from Pelechano and Badler [5], proposes an architecture that combines and integrates MACES and PMFserv frameworks increasing the crowd behavior accuracy.

The framework created by Pettre, Ciechomski, Maïm, Yersin, Laumond and Thalmann [6] achieved the real-time visualization of crowd simulations combining levels of detail (LOD) techniques and billboards, dramatically reducing the cost of animations updates and deformations on the models.

Another work from Thalmann and Pablo [7] presents a rendering engine to visualize a crowd of virtual humans in real-time. The virtual humans are sorted by their distance to the observer into render fidelities groups. The engine combines impostors and different rendering acceleration techniques, such as caching schemas, levels of detail, shader and state sorting.

3. VHuP Architecture

VHuP performs the real-time visualization of virtual humans and crowd, using a group of open sources platforms and toolkits. This group is coordinated by a core, which has the main functionalities of the application. All the services are built over this core, providing flexibility and an easy update capability.

The features are implemented maintaining the primary goal of the application: the ability to visualize a variety of crowd and virtual human simulations algorithms used on our research lab. The idea is making the visualization process simple, reducing the time of developing specific interfaces and avoiding any rework. The **Figure 1** is an overview of the VHuP architecture and is followed by an explanation of each component.

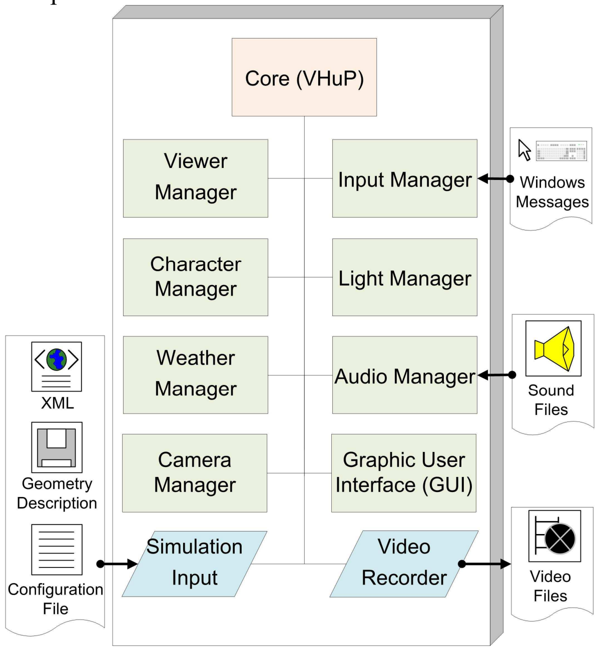

Figure 1. VHuP architecture overview.

The **Core (VHuP)** is the main component of the application, responsible for executing and managing the other components. The core coordinates the application logic: updates, draws and events. This is performed using the Open Scene Graph (OSG) [8] graphic toolkit as rendering engine. It also contains several important functionalities and interfaces for communication between the components and OSG.

Our application uses a composite view model, in other words, the application is allowed to place up to five views in the scene, including a mini map view and a record view. Those views, controlled by the **View Manager**, provide a greater perspective of the simulation allowing the possibility to focus on areas of interest. This component handle the window resize events, cameras and every scene data attached to each view.

The characters are structured in three parts: model, bones and animations. The **Character Manager** organizes this structure in memory and is in charge of all actions related to characters along the simulation. The component uses the 3D Character Animation Library (Cal3D) [9] through the Cal3D adapter for OSG (osgCal2) [10] to perform animations. It also provides hardware skinning, avoid streaming dynamic data to GPU, and animation blending, smoothly playing animations.

The **Light Manager** allows determining how the environment's illumination should be. It is possible to add and modify lights sources customizing the scene and increasing the simulation realism.

The shadows created by the light sources are also controlled by this component. These shadows can be modified, changing the shadow appearance, type, quality or even disabling it to obtain the desired tradeoff between performance and graphic quality.

In order to increase the graphic quality and realism of the visualization, there is also the **Weather Manager**. This component is capable of creating weather effects such as fog, rain, snow and clear weather.

The **Camera Manager** component allows navigating in the scenario easily. It uses the OSG camera architecture making it possible to set camera manipulator to each point of view in the application. Our default camera manipulator is similar to strategy games cameras.

The Microsoft Windows-based application uses messages in order to handle events. Those messages, sent by other applications, keyboard or mouse, are handled by the **Input Manager**. The features of this component are the input buffer, computer pens and parallel communications. The input buffer supports multiple pressed keys. Computer pens can be used and the pressured sensitivity is also controlled. The parallel communication makes possible an interaction with other applications such as eyes and gesture recognition.

Audio Manager performs the simulation audio and uses the Ambiera IrrKlang Audio Library [11] to provide support to different audio file formats such as MP3, OGG and WAV. The additional files are previously loaded before the simulation rendering avoiding possible slowdowns.

VHuP has a friendly **Graphic User Interface (GUI)** granting access to the application functionalities easily and in a fast way. The **GUI** architecture is component based and was built over the OSG event handlers classes. This approach allows compatibility with OSG applications and new components are added with no effort. Each component of the **GUI** have a set of events related to it, the actions to each event are easily customized. The mouse click is an event example and coloring a button when it's clicked is a simple action. The architecture divides the GUI components on three layers: window, forms and objects. Buttons, images and textboxes are examples of objects. A group of objects are organized in forms, and the GUI scope is the window.

The **Video Recorder** uses a separated view to record on a set of image files. Those images are recorded on a chosen frame rate and with a custom graphic quality, relative to the recording view. The files are compressed and filtered using any video software, as the Virtual Dub, resulting on a video file.

This **Simulation Input** receives all the simulation data files and is responsible to read and parse the data. These files are the xml defining the simulation, the application configuration xml and the meshes. Those meshes are supported on different formats, such as OSG, 3DS and OBJ.

4. Results

The results were obtained executing VHuP on a AMD 64 x2 Dual Core 4200+, 2GB RAM DDR, equipped with a GeForce 8800GTS OC. The frame-rate difference (measured in FPS) between the higher and lower graphic quality is easily visible when a simulation with a small number of characters is rendered.

When the number of agents increases, the computational time for high quality rendering is decreased. This shows that using finer techniques, shadows and weather effects are not the bottleneck of the application. **Figure 2** shows a graphic where FPS obtained as a function of changing rendered number of characters is presented.

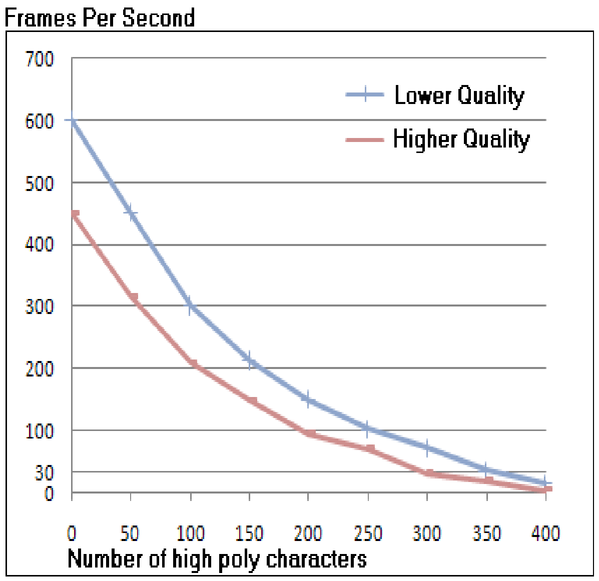

Figure 2. FPS related to the number of characters.

The application's bottleneck is visible when rendering more than 300 characters. The virtual humans animation's update is bounded to the computer processing unit (CPU), requiring a large amount of process. This issue makes the higher quality, illustrated at **Figure 3**, with the best tradeoff between performance and graphic quality.

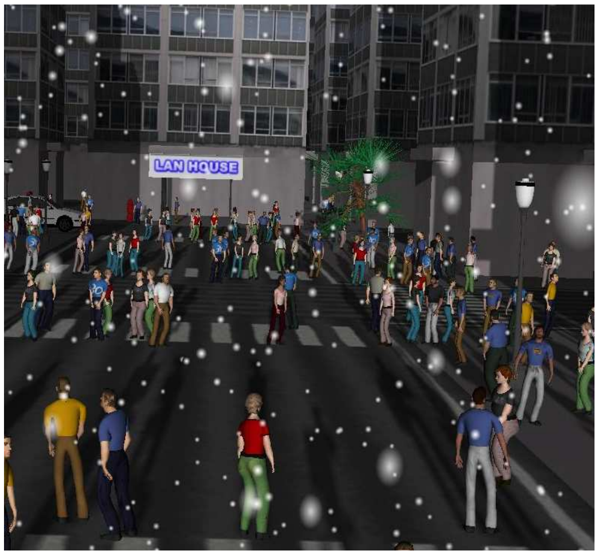

Figure 3. VhuP at high quality.

5. Future Work

The obtained results were acquired with the first VHuP version. On the second version new techniques will be studied and implemented to improve the graphic quality and to increase the number of characters rendered in real-time.

The graphic quality could be improved using shaders to better render lights and different materials, so rendering realistic water and fire effects becomes possible. Different techniques for creating shadows and weather effects could be performed. Higher polygon characters could also be used, adding LOD and culling techniques [1].

Those techniques, combined with impostors and rendering acceleration techniques [7], could also increase the number of character rendered in real-time. At last, creating a database for characters animations and models, could increase both the graphic quality and the performance of the application.

6. Acknowledgements

We thank all the support from Marcelo Paravisi and Rafael Rodrigues during the development and test of the VHuP. Besides being authors of the VHSimul project, they created unique simulations and gave important ideas for the application. This work was developed in collaboration with HP Brazil R&D.

7. References

[1] J. Hamill, C. O'Sullivan, J "Virtual Dublin – A Framework for Real-Time Urban Simulation", WSCG, 2003.

[2] S. R. Musse, and D. Thalmann, "Hierarchical model for real time simulation of virtual human crowds", *IEEE Transaction on Visualization and Computer Graphics*, 2001, pp. 152-164.

[3] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic", *Nature*, 2000, pp. 487-490.

[4] A. Treuille, A. Lewis, and Z. Popovic, "Model reduction for real-time fluids", SIGGRAPH, 2006.

[5] N. Pelechano, and N. Badler, "Modeling Crowd and Trained Leader Behavior during Building Evacuation", *IEEE Computer Graphics and Applications*, Volume 26, 2006, pp. 80-86.

[6] J. Pettre, P. H. Ciechomski, J. Maïm, B. Yersin, J. P. Laumond, and D. Thalmann, "Real-time navigating crowds: scalable simulation and rendering", *Computer Animation and Virtual Worlds*, Volume 17, 2006, pp. 445–455.

[7] P. S. H. Ciechomski, and D. Thalmann, "Rendering Massive Real-Time Crowds", *Thèse n° 3534 - École Polytechnique Fédérale de Lausanne*, 2006.

[8] Open Scene Graph (OSG), *open source high performance 3D graphics toolkit*, http://www.openscenegraph.org/, accessed July 20, 2008.

[9] 3D Character Animation Library (Cal3D), *skeletal based character animation library*, https://gna.org/projects/cal3d/, accessed July 20, 2008.

[10] Cal3D adapter for OpenSceneGraph (osgCal2), *adapter of cal3d for use inside OSG*, http://osgcal.sourceforge.net/, accessed July 20, 2008.

[11] Ambiera IrrKlang Audio Library, *high level 2D and 3D cross platform sound engine and audio library*, http://www.ambiera.com/irrklang/, accessed July 20, 2008.