PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
DEPARTAMENT OF INFORMATICS
GRADUATE PROGRAMME IN COMPUTER SCIENCE

CROWDVIS: A FRAMEWORK FOR REAL TIME CROWD VISUALIZATION

HENRY BRAUN

Dissertation presented as partial requirement for obtaining the master's degree in Computer Science from Pontifical Catholic University of Rio Grande do Sul.

Adviser: Prof. Dr. Soraia Raupp Musse

Porto Alegre, Brazil
2012

This Page Intentionally Left Blank.

This Page Intentionally Left Blank.

“There must be a beginning of any great matter, but the continuing unto the end until it be thoroughly finished yields the true glory.”

Sir Francis Drake

“We cannot change the cards we are dealt, just how we play the hand. If I don’t seem as depressed or morose as I should be, I’m sorry to disappoint you.”

Randy Pausch

ACKNOWLEDGMENTS

During this two year experience I had the opportunity to meet incredible people and to be in extraordinary places. I want to specially thank my adviser Soraia Raupp Musse, who trusted me in all the moments and helped me push my limits even further. Professors Marcio Pinho, Marcelo Cohen and Christian Lykawka, thank you for helping me cross the bridge to get into the CG world. Professor Donald Marinelli, thank you for everything you have done for me and the wonderful time at the Dream Fulfillment Factory.

My VHLab workmates, thank you all for listening to my metal musics without going nuts! Vinicius Cassol, thank you for your friendship and all the help! Rafael Rodrigues, Rafael Hocevar, Fernando Marson, Humberto Souto, Adriana Braun, Juliano Moreira, Cristina Scheibler, Luiz Cunha, Anderson Silva, Rossana Queiroz, Leandro Dihl and Julio Jacques, I am pretty sure that you guys are going to be successful at any place and at any time, keep up the good work.

My VRoom360 team, you guys know how much awesome you all are. Tim Forsythe, thank you for being an incredible leader. Anmol Nagpal and Dev Ghai, thank you for sharing your amazing culture. Ping Li Andy, thank you for being supportive during all the moments. Samantha Collier, thank you for not punching me in the face. Terry, thank you for helping me out and showing me Pittsburgh! Pan, Mrs. Baby and Mr.Pepper, thank you for making my days happier by your presence.

Swordtales team, thank you for the opportunity to work with such good talents. Luiz Alvarez, Conrado Testa and Alessandro Martinello don't give up on games.

Manuel Lopez, Breogan Amoedo and Marc Estruch, thank you for providing me shelter when I had nowhere to go and mostly for trusting me your friendship.

Priscila Wagner and Diogo Ribeiro, thank you for helping me out during bad times.

Diogo Lima and Marcelo Paravisi, thank you my two brothers. For all the coding, listening, explaining, gaming, drinking, supporting and for yelling at me when needed. Eduardo Muccillo, thank you for all the chatting and good beers!

Juliana Pacheco, I love you. Thank you for everything, mainly for showing me how simple things can be, and being there for me everyday.

My father Harold Braun, mother Norma Braun and family, thank you for being just the way you are, for providing everything I needed in life.

This work was made in collaboration with *Hewlett-Packard Brazil R&D*.

CROWDVIS: A FRAMEWORK FOR REAL TIME CROWD VISUALIZATION

ABSTRACT

Crowd visualizations are present mostly in digital games and computer animated movies, but they are also observed in simulations and virtual reality applications. In crowd simulations we should represent the behavior of agents given different scenarios, and also, such simulations can be provided by different software and tools. This document describes a framework for real time crowd visualization, which no programming knowledge and modeling skills are required from the users. Our main goal is to be able to visualize previously created crowd simulations in real time, combining rendering techniques and providing easy support for managing the scene and the virtual humans.

Keywords: Crowd Visualization; Crowd Simulation; Virtual Humans; Reconstruction; Rendering; Level-Of-Detail; Special Effects.

CROWDVIS: UM FRAMEWORK PARA VISUALIZAÇÃO DE MULTIDÕES EM TEMPO REAL

RESUMO

Visualizações de multidões estão presentes principalmente em jogos digitais e filmes de animação computadorizada. Essas visualizações também são observadas em simulações e aplicações de realidade virtual. Em modelos para simulações de multidões é preciso representar o comportamento dos agentes de acordo com os diferentes cenários, além disto, tais simulações podem ser originadas de diferentes *software* ou ferramentas. Esta dissertação apresenta um *framework* para visualização de multidões em tempo real, a qual não requer conhecimento de programação e modelagem. O principal objetivo é apresentar visualizações de simulações previamente criadas, combinando o uso de técnicas de *rendering* em tempo real, além de proporcionar ferramentas para gerenciar a cena e os humanos virtuais.

Palavras-Chave: Visualização de Multidões; Simulação de Multidões; Humanos Virtuais; Reconstrução; *Rendering* em Tempo Real; Níveis De Detalhe; Efeitos Especiais.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2D	Two-Dimensional
3D	Three-Dimensional
AI	Artificial Intelligence
BVH	Biovision Hierarchy
CG	Computer Graphics
CPU	Central Processing Unit
CVE	Collaborative Virtual Environment
DOF	Degrees-Of-Freedom
FOV	Field-Of-View
FPS	Frames Per Second
GPU	Graphics Processing Unit
HDD	Hard Disk Drive
HDR	High Dynamic Range
IBR	Image-Based Rendering
IK	Inverse Kinematics
LOD	Level-Of-Detail
VH	Virtual Human
VR	Virtual Reality
VRML	Virtual Reality Modeling Language

SUMMARY

1. INTRODUCTION

Impressive game engines have become accessible to general users in the past few years. Engines such as *Unity 3D* [1] and *Unreal Engine* [2], among others, provide tools and mechanisms for displaying Three-Dimensional (3D) graphics and creating games. However, these and other engines are rarely focused on real time crowd visualization, besides that, such engines, require a certain set of programming skills and also artists to coherently create the assets and the Virtual Humans (VHs) for visualizing crowds.

Crowd simulation is an area of Computer Graphics (CG) and, sometimes, Artificial Intelligence (AI) concerned with the process of simulating a large number of virtual agents. Nowadays, crowd simulation is an important topic of research since it is crucial for some applications as, for instance, in the fields of 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 [20].

Methods to simulate VHs and crowds are often mediated by local rules [32], forces [22], flows [49] or hybrid approaches [35]. Usually, these methods consider the agents as simplified geometries, such as points, spheres and cubes. These abstractions are important to facilitate the usage of complex mathematical models. However, when the simulations should be visualized, some artifacts can be noticed, such as geometry intersections or issues regarding the VHs motion.

Examples of pioneer methods are Musse [32] and Helbing [22], which provide different models for crowd simulation, that could be specific employed for different scenarios (e.g. a social event or an evacuation situation). Normally, the simulation prototypes include their own visualization module, which can limit the integration of different simulation models in their visualization.

Isolating the simulation from the visualization may facilitate the creation process for the users, as they can now be only concerned with the simulation model or with the visualization. Besides that, a dedicated visualization module could integrate several simulation models in order to create an unique scene, e.g. a specific model for small groups can simulate people inside buildings, while another model can simulate the crowd outside. Dedicated visualization modules can also improve realism (e.g. by the usage of embodied agents and better rendering), granting superior understanding of the simulation.

Improved visualization of crowds can be used for other purposes, such as entertainment, marketing, education or safety training systems. These applications usually require a certain set of features, such as load simulation files, provide support to embodied VHs and also

[1] `http://www.unity3d.com`
[2] `http://www.unreal.com`

represent motion in a coherent way accordingly the simulation. The usage of virtual humans is essential, since the agents are important for several scenarios, as in games or Virtual Reality (VR) applications, especially in virtual scenes that aims to realistically recreate the real world [47].

In this work we present *CrowdVis*, a framework to be handled either by naive or experienced users. The main goal is to provide tools and enable the users to visualize, in 3D environments, previously created crowd simulations. *CrowdVis* can reproduce the simulations created by any source, by simply matching their output in our simulation input format. This approach can be used in order to create an unique visualization by combining different crowd simulation techniques. Moreover, *CrowdVis* assures that the simulation and the visualization is conservative, i.e. the visualization represents exactly what is going on the simulation.

Specifically, *CrowdVis* can create embodied agents, with animations as well, and propose visual behaviors for agents interactions. This is motivated due to the fact that the game industry is always engaged in unraveling new gameplay experiences in order to provide a deeper game immersion. One of these new experiences is by allowing the player to use her/his own appearance-like avatar while playing, presented in Electronic Arts Sports[3] games (e.g. Fifa Soccer 2012[4]).

1.1 Objectives

We propose a model for virtual humans, which includes the detection of possible interactions, techniques for a realistic motion and visually accepted reconstruction of VHs based on a single Two-Dimensional (2D) image. The techniques presented in our model could be employed for different virtual worlds, such as games and movies. Besides this, *CrowdVis* prototype can generate visualizations that could be employed for different purposes. Some examples are listed bellow:

- a video for evacuation procedures illustrating the emergency exits;
- a video for entertainment purposes such as games or events; and
- a virtual walkthrough intended to show off buildings and houses, among others.

The main goal of *CrowdVis* is to provide a dedicated framework for real time visualization of crowd simulations. The main idea is to provide total control for the user, requiring no programming experience, capable to visualize previously created simulations using custom VHs and/or the ones created by the reconstruction model. Moreover, *CrowdVis* also

[3] `http://www.ea.com/games`
[4] `http://www.ea.com/soccer/fifa`

presents tools for animation, special effects, environment management, behavior control, VHs reconstruction and illumination. Table 1.1 organizes in a list requeriments we believe it is essential for a crowd visualization framework to attend.

Table 1.1 – A list of requeriments we believe it is essential for a crowd visualization framework to attend.

Name	Description
Virtual Humans Support	To be fully capable of using customs VHs, a set of default ones or provide an easy way to create VHs on the fly.
Environment Support	To be fully capable of loading and manipulating the scene assets and its illumination.
Crowd Simulation Support	To manipulate the simulation playback, visualize paths and play simultanslesly simulations at the same visualization.
Realism Features	To provide animation blendings, environmental virtual humans, advanced lightning, detect and be able to display interactions between VHs.
Rendering Optimizations	To be able to reproduce large scale simulations, providing optimizations techniques.
Audio & Visual Effects	To be able to script and visualize audio and visual effects such as fire, rain and explosions.

1.2 Organization

This document is organized as follows: related work are divided into three sections and are described in the next chapter. Chapter 3 contains a detailed description of *CrowdVis* model. Section 3.1 explains the processing work for detecting possible interactions among the VHs, and also includes how the VHs are manipulated. Section 3.2 explains our techniques for reconstructing a virtual human from a single picture. Chapter 4 explains our prototype and its features, including special effects and rendering optimizations.

Results are shown in Chapter 5, where we compare *CrowdVis* with another engine, show some of our reconstructed virtual humans (see Section 5.1) and also illustrate test cases using different crowd simulators (see Section 5.2). Chapter 6 explains important topics for improving *CrowdVis* as well final considerations are discussed.

2. RELATED WORK

CrowdVis relies on several techniques and different approaches such as crowd behavior analysis, VHs reconstruction, mesh deformation, texturing, rendering optimizations, among others. In this chapter we present some related work to facilitate the understanding of our model. Next sections include work related to individual behavior of crowds, reconstruction of VHs (based on mesh deformation) and real time crowd visualization.

2.1 Individual Behavior in Groups or Crowds

Several games and crowd simulation models use interactions among VHs in order to create dynamic and realistic scenes for different purposes. Algorithms to perform those interactions have been greatly improved by researchers over the last years. It is known that several solutions were already created in order to provide interactions, but solutions aiming to provide visual representation of interaction behaviors are not common.

Becheiraz and Thalmann [4] propose a nonverbal communication and interpersonal relationship model between virtual humans, consisting in reactions and social interactions in the presence of other characters. In this work, besides interpersonal relationships, they also analyze the virtual human's walking, such as a sad walk or happy walk, their postures also have influence on the behavior of others, increasing the believability of the virtual humans in social interactions. Figure 2.1 illustrates Thalmann and Beicheiraz nonverbal communication model generated between two agents.

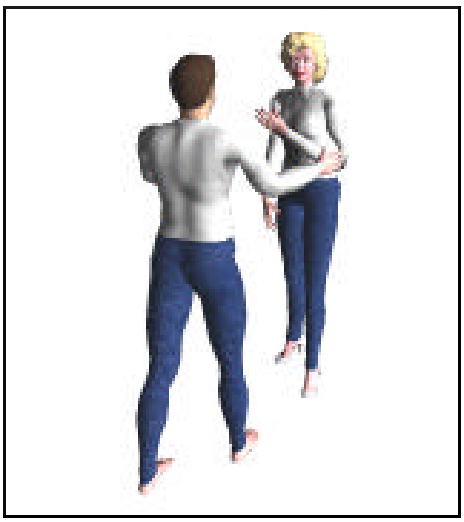


Figure 2.1 – Illustration of Thalmann's and Beicheiraz [4] nonverbal communication model generated between two agents.

Neff and Fiume propose a work to provide body postures motion combining feedback based balance control with a hybrid Inverse Kinematics (IK) system that utilizes optimization and analytic IK components [33]. Their approach allows an animator to use a single scalar to vary a character's pose within a specified shape class, providing an intuitive parameterization of a posture. Later, same authors describe an improvement in their model [34] by providing a high-level constructions for expressive animation, which impose different modalities of different animated interpretations of the same script

Another work in the area is from Diane Chi *et al.* [11]. According to the authors, only seeing at the psychological notion of gesture is insufficient to capture movement qualities needed by animated characters. For this reason they presented a 3D character animation system called EMOTE. This system applies Effort and Shape qualities to independently define underlying movements, and thereby generates more natural synthetic gestures for the characters.

Pelechano and Badler [35], propose an architecture that combines and integrates MACES[1] framework in order to increase the crowd behavior accuracy. MACES works without a central controller, where each agent contains a behavior using personality for real psychological factor. The behaviors developed by them relies on communication and local perceptions.

These few examples of related work describe sophisticated methods, using physics or artificial intelligence techniques, in order to provide body postures and animation accordingly to the simulated scenario. The body animations are performed during the visualization and can affect other variables into the simulation, as well other agents.

2.2 Virtual Humans Reconstruction

One of the pioneers work about reconstructing a VH based on image data is described by Hilton *et al.* [23]. The main idea of this work is to capture human models to populate virtual worlds. In order to do that, Hilton *et al.* used four human silhouettes (front, back and both sides) against four silhouettes obtained by a *Virtual Reality Modeling Language 2* (VRML) generic model. By comparing these silhouettes in a 2D universe, it is revealed the pixels displacement in relation with each other. After that a pixel displacement factor is mapped to the 3D generic model, shaping it in a custom and unique form. At last, the generic model is colorized, using an approach of cylindrical texture mapping [40].

Tavares [44] developed a model to generate several virtual humans in a fast and simple way. Using a minimum set of information called *somatotypes*, the authors are able to create different virtual humans. *Somatotype* is a list of parameters defining the individuals strength, slimness and fatness, that describes the shape of human bodies. In order to

[1] Multi-Agent Communication for Evacuation Simulation is a system without a centralized controller.

create individuals based on *somatotype* values, they use a generic VH model divided in a few mesh groups. Then each mesh group is deformed according to the results of the dominant *somatotype* influence. This approach allows the generation of several different characters by just changing the *somatotype* parameters. Figure 2.2 illustrates three different virtual humans created by Tavares model [44] using different *somatotype* values.

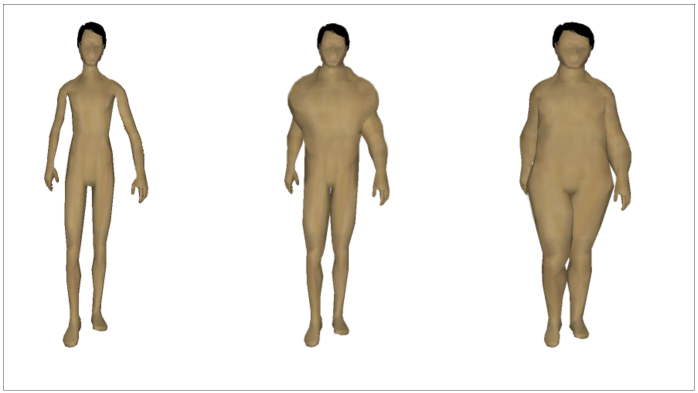


Figure 2.2 – Illustration of three different virtual humans created by Tavares model [44] using different *somatotype* values.

Guan *et al.* manage to retrieve a human shape and its pose from a single image [18]. Their approach involves the usage of a shape database (SCAPE [1]) containing several different human models, one image as input and manual handwork intervention. When the picture has been loaded, it is asked for the user to click on the main human body joints. According to Taylor [45] knowing the position of the joints in a 2D image make possible to find out the depth structure based on a bone length calculation as long the image provides an orthographic perspective. Retrieving the human body silhouette based on segmentation algorithms, Guan uses the shading of the picture to correct possible perspective ambiguities[2] resulted by Taylor approach.

Vieira *et al.* [10] developed a technique for corporal manipulation of virtual characters. This technique is based on the deformation of anthropometric measures using influence zones and deformation restrictions. This generic approach is achieved using spherical influence zones, in which the vertexes near to the center of the sphere are applied with a larger displacement factor than the ones near to the border. In this work, it is the sphere that is manipulated by scale and translation operations. This process creates a "*force on the vertexes*", defining a new appearance on the virtual human. To achieve better results, this work uses anthropometric landmarks for defining manipulation rules, which characterizes the influence zones.

A recent work on body reshape estimation has been done by Hasler *et al.* [21]. This

[2]Different 3D poses can look exactly the same when defined in 2D images.

work proposes a multlinear model of human pose and body shape. The model is generated combining factorized measures with an ICP[3] base registration method. It is important to highlight that their estimation is made from a single image, on the other hand, if several images are employed the results can be improved.

Representing the appearance and motion of realistic virtual humans is a hard task. That is due to the fact that the evaluation is made by us, human beings, who are experts in viewing human beings. Independent of the techniques for virtual humans creation or reconstruction, these techniques can be employed for different purposes, such as games, movies or *crowd visualizations*.

2.3 Crowd Visualization

Several researches in virtual humans and crowd visualizations have been provided in the last two decades. These researches explore techniques for handling large sets of polygonal models efficiently, specially when dealing with entities like crowds, which usually needs more computational power than what is available on current hardware.

Amaury Aubel *et al.* [2], propose a hardware independent technique which the main goal is to improve the frame rate of the visualization of virtual humans. Their approach acts on the geometry and rendering information and combine with the usage of impostors.

An important work by Tecchia *et al.* [46] describes a set of techniques for rendering crowds in real time. Their approach is not only concerned with the human behaviors and collision detection in crowds, but also in optimization techniques for fast rendering, which include shadowing and quality shading. Using what they called "*Pre Computed Impostors*" as a Image-Based Rendering (IBR) technique and the availability of large texture memory buffers, they maximized rendering speed. Figure 2.3 illustrates Tecchia's *et al.* model for rendering real time crowds with shadows.

Hamill and O'Sullivan [20] resume a number of methods available for culling such as portal, occlusion horizons, occlusion shadow volumes and the simple bounding box tests. The vast types of environments, such as indoors and outdoors, used in the crowd simulation makes common the use of multiple techniques combined. The Virtual Dublin [20] make use of culling and collision techniques combined with the trade of model details for texture details, reducing the building's polygons, but increasing the memory required.

Another topic of research is concerned to improve better depth impression on impostors. An interesting billboard technique was developed by Decoret [15] in 2003, which he called *Billboard Cloud*. His technique relies on combining a set of billboards in different layers to make it almost indistinguishable when compared to 3D polygonal mesh.

The framework created by Pettre *et al.* [36] achieve the real time visualization of crowd

[3] Iterative Closest Point is a method to align free-form shapes, often used to reconstruct 3D surfaces.

Figure 2.3 – Illustration of Tecchia *et al.* [46] model for rendering real time crowds with shadows.

simulations by combining Levels-of-Detail (LOD) techniques. Their LOD techniques were combined with billboards, which dramatically reduced the cost of animations updates and deformations on the models.

De Heras Ciechomski [13] present 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 fidelity groups. Their engine combines impostors and different rendering acceleration techniques, such as caching schema, Level-of-Detail, shader and state sorting.

Kavan *et al.* [28] propose a method focused on rendering animated characters for large crowds in real time. Their method reduces the amount of memory necessary for rendering crowd impostors. Also, they create the image, used for rendering a virtual human, as a 2D polygonal texture providing more visual fidelity and delivering a variety of animations with almost no memory overhead.

2.4 CrowdVis and the State-of-the-Art

CrowdVis is a framework focused on real time crowd visualization. We were inspired by several works in order to achieve our goals, which include the reconstruction of VHs, behavioral analysis, animation aspects and rendering optimizations. The reconstruction of VHs is not only important for crowd visualization, but also for Collaborative Virtual Environments (CVEs) and VR applications. In some cases, there are interfaces for VHs customization, but the techniques presented in *CrowdVis* replace such interfaces for a semi-automatic approach, where the user choose a picture for creating an avatar. The employed techniques, from 3D posing [45] to mesh deformation [8] can also be used to avoid the work of generating several different characters for computer animated movies or games.

Once the scene is populated with virtual humans, we can work to increase the

visualization believability, i.e how coherent the visualization is if compared with the reality. One way is through a *behavior analysis* of the virtual humans presented in the simulation. In Samovar and Porter's words [39], "*Language gives people a means of interacting with other members of their culture and a means of thinking. Language thus serves both as a mechanism for communication and as a guide to social reality*". In theaters there is a French expression called *Mise-en-Scène* and literally means "putting on stage". It represents the actors movements, positioning and background interactions during a theater play. Transposing this expression to virtual environments, such as those found on simulations and computer games, it symbolizes arranged interactions between background virtual humans.

As explained in the related work chapter, the interactions among VHs can represent verbal or nonverbal communication, such as virtual humans checking their wristwatches, a simple "hello" gesture or even a conversation between two or more characters. In *CrowdVis*, the interactions are performed in "second plane", without changing the simulation data. The main difference in comparison with our approach is that we provide a model to generate visual behaviors, which can be displayed as body animation or other any visual representation. But mainly, in our approach, the interaction behavior is generated without changing the simulation, treated as a post-processing analysis, since it is used for already simulated situations in our visualization. The main goal is to improve the realism and believability of the visualization (maintaining the simulation intact) and the techniques employed can also be used for other applications, such as games and any other suitable real time scenario.

CrowdVis is also concerned with audio-visual inputs and feedback. It is important for a visualization framework being able to represent events that enriches both simulation and visualization. For example, if we are visualizing a simulation concerned in analyzing the evacuation of a metro station after an explosion, it would be nice to represent the calm moment and suddenly the bomb explosion with the sound of it, along with coherent motion of the virtual humans. These and other event situations [43] are common in crowd simulations.

Rendering real time crowds can be a challenge given the amount of geometry necessary for displaying. *CrowdVis* rendering techniques are based on Maciel [31], Tecchia [46] and Pettre [36] works. We combine Level-Of-Detail techniques, including Progressive Mesh [24] for geometry simplification, and animated impostors. The main difference in our approach is being able to adjust the LOD distances according to the simulation and also the usage of a large set of textures. Moreover, we are able to provide realistic impostors because the newer hardware increased their video memory capacity enabling great amount of textures being handle, either by the Central Processing Unit (CPU) or the Graphics Processing Unit (GPU).

3. THE *CROWDVIS* MODEL

The main focus of this work is to provide a model for crowd visualization, that should act after the performed crowd simulation. Visualizing such simulations require a set of data, e.g. virtual humans, terrains, virtual environments among others. Indeed, our model should deal with the following aspects:

- creation of virtual humans;
- manipulation of scenarios and assets;
- manipulation of the simulation playback;
- control of virtual humans behaviors; and
- management of rendering features, special effects and scene illumination, presented in Chapter 4.

An overview of *CrowdVis* is illustrated in Figure 3.1, where it is possible to observe the three main modules. The first module consists of a *behavior analysis*, responsible for parsing the simulation data, analyzing movement and animation patterns (human motions), detecting possible interactions between agents and manipulating *screenplay* animations (further explained in Section 3.1.3). The second module is concerned with *reconstruction of virtual humans*, where *CrowdVis* uses algorithms such as 3D pose identification, human segmentation and silhouette processing, combined with template posing, mesh deformation and texturing techniques. At last, *audio-visual* features, exploring how to integrate *special effects* with the simulation in order to provide realism and believability among setting up the scene and rendering optimizations. This last part is presented in Chapter 4.

3.1 Behavior Analysis Module

The inputs of this module are two files: the *simulation XML* (see Listing 3.1) and the *screenplay XML* (see Listing 3.3 in Section 3.1.3). The first input describes the motion of the virtual humans and the second input contains specific animations that should be played for each VH at a specific simulation frame. The motion of the agents is described by a set of 3D points over a time sequence. In our model, all the simulations must follow this input format (exemplified in Listing 3.1), where each frame contains the positions for all the agents.

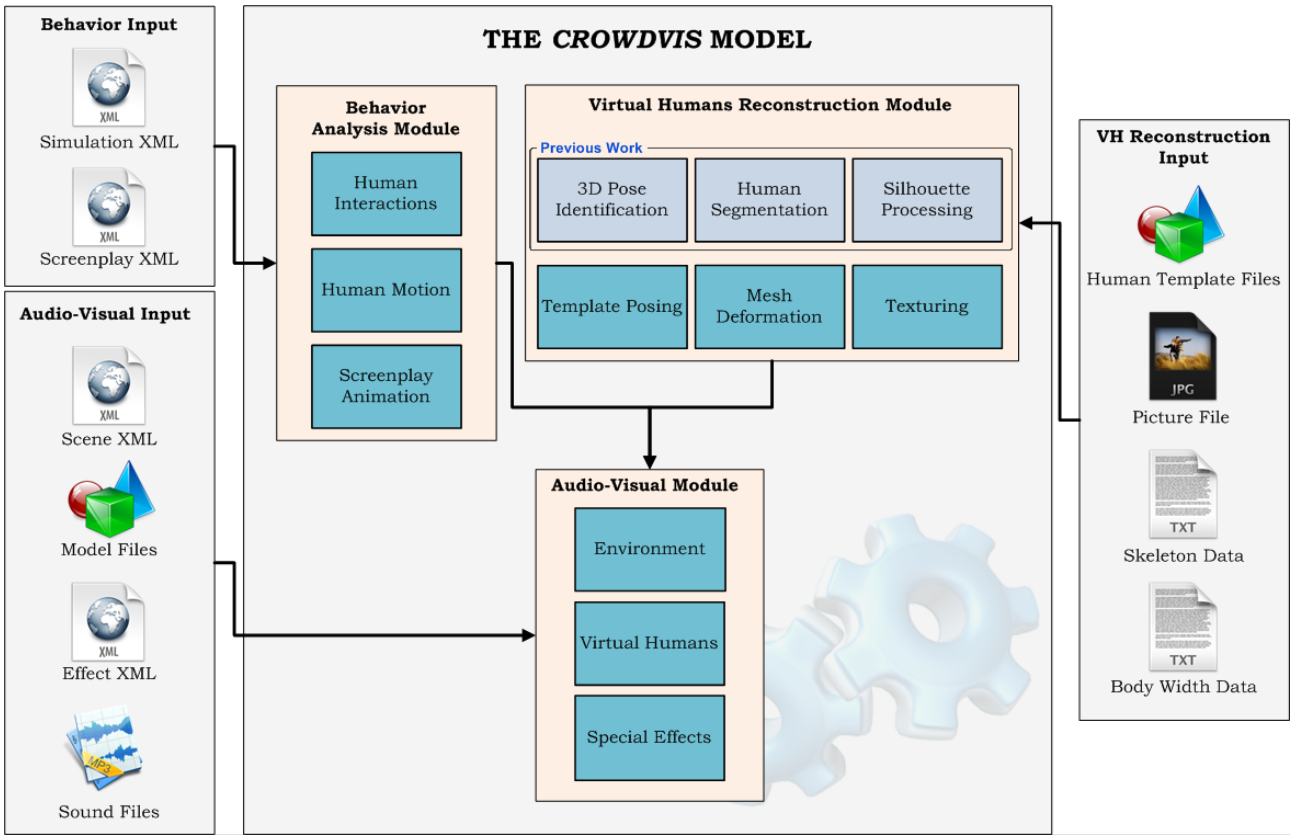


Figure 3.1 – An overview of *CrowdVis* model. It is possible to observe the three main modules (*Behavior Analysis*, *Virtual Human Reconstruction*, *Audio-Visual*) and their respective inputs.

```
1  <SIMULATION>
2          <AGENTS>
3                  <AGENT id="0"></AGENT>
4                  <AGENT id="1"></AGENT>
5          </AGENTS>
6          <FRAMES>
7                  <FRAME>                                                    //Frame [0]
8                          <AGENTS>
9                                  <AGENT id="0">
10                                         <POSITION>3.05 12.21 0.00</POSITION>          //Position of Agent [0] at Frame [0]
11                                 </AGENT>
12                                 <AGENT id="1">
13                                         <POSITION>37.41 12.21 0.00</POSITION>         //Position of Agent [1] at Frame [0]
14                                 </AGENT>
15                         </AGENTS>
16                 </FRAME>
17                 <FRAME>                                                    //Frame [1]
18                         <AGENTS>
19                                 <AGENT id="0">
20                                         <POSITION>3.05 12.21 0.00</POSITION>          //Position of Agent [0] at Frame [1]
21                                 </AGENT>
22                                 <AGENT id="1">
23                                         <POSITION>37.41 12.21 0.00</POSITION>         //Position of Agent [1] at Frame [1]
24                                 </AGENT>
25                         </AGENTS>
26                 </FRAME>
27         </FRAMES>
28 </SIMULATION>
```

Listing 3.1 – Simulation XML input data example for two distinct agents at two frames.

Considering the input files information, the *Behavioral Analysis* module has the goal to read the simulation data (set of 3D points over the time) and, after an analysis process, improve the results of crowd visualization. We detect and store possible *human interactions* among the VHs to be further visualized. These interactions are detected according to the agents distances and orientations, as detailed in next section. Another way to improve the visualization results is by providing a realistic and believable motion of the VHs, for this, we use time coherence and an analysis of the agent's velocity. At last, this module is also responsible for coordinating predefined animations that the user may want to visualize during the playback, which we called *screenplay* animations. Next section details some of such aspects.

3.1.1 Human Interactions

While studying and observing groups of humans, the anthropologist Edward Hall [19] proposed the *proxemics* concept, which describes that the distance between people, when interacting with each other, varies according to their levels of intimacy. Hall divided the distances into four possible intervals: *intimate* $[0.00, 0.45]m$, *personal* $(0.45, 1.20]m$, *social* $(1.20, 3.60]m$ and *public* $(3.60, 7.60]m$, as illustrated in Figure 3.2. Knowing these distances between pairs of agents, we can play different types of animations in order to provide more believability to the whole scene, e.g. a gesture of hello or even to make a couple holding hands according to the intimacy distance.

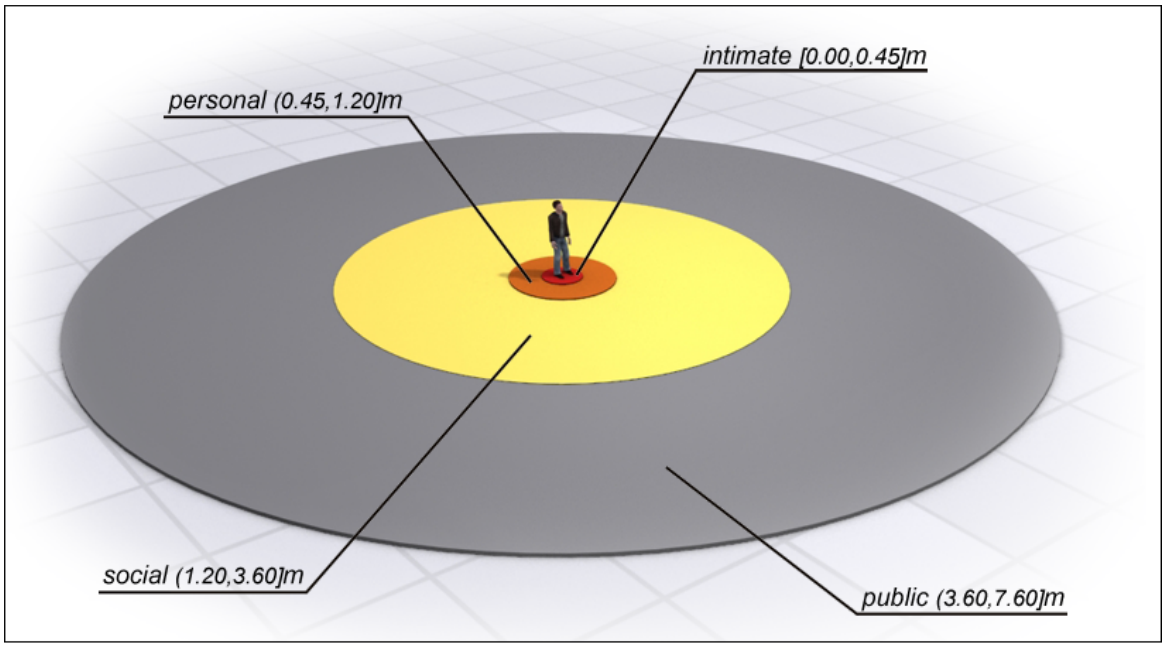


Figure 3.2 – Hall's proxemics distances illustration of the four possible intervals: *intimate* $[0.00, 0.45]m$, *personal* $(0.45, 1.20]m$, *social* $(1.20, 3.60]m$ and *public* $(3.60, 7.60]m$.

Since our goal is to provide "*visual behaviors*" for interacting agents, we need to find out the possible relationship between them in order to set a proper animation. Once we are not able to determine the affinity or even estimate the relatedness degree between the agents, we opted for using only Hall's *social* and *public proxemics*. For these, we detect two common situations to perform real time interactions: *i)* agents walking together; *ii)* agents passing by each other in opposite directions. In order to detect these interactions we use the *proxemics*, agent's *Field-Of-View* (FOV), the duration of the interaction and the agent's speed. Figure 3.3 illustrates the two types of *humans interactions* detected in our model: agents walking together (top) and agents passing by each other in opposite directions (bottom).

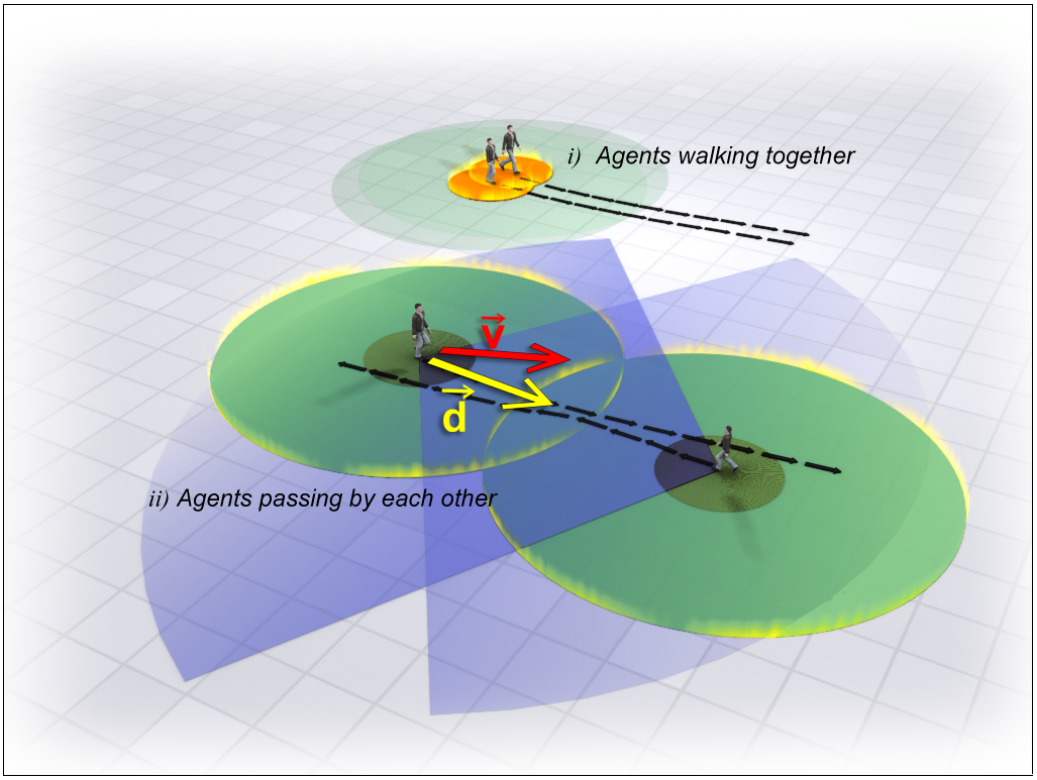


Figure 3.3 – Two types of VHs interactions detected in our model: the agents walking together (top) and agents passing by each other in opposite directions (bottom).

In order to detect if agents are passing by each other in opposite directions, we firstly verify the *public proxemics* according Hall's *public* interval $(3.60, 7.60]m$ (as illustrated in Figure 3.2). In our model, we opted for using a threshold of six meters from each one of the agents. In case a pair of agents are located in each other *proxemics*, we verify their orientations and FOV to determine if they are going in opposite directions and if they are able to see each other, as illustrated in Equation 3.1:

$$\gamma = \arccos(\vec{v} \cdot \vec{d}), \tag{3.1}$$

where $\vec{v}$ is the agent's orientation and $\vec{d}$ is the vector between the two agents. If the vision angles calculated for both agents are inside our threshold[1], we consider that the agents are seeing each other. The final step is responsible for checking how long this process lasts, in other words, when the agents do not see each other anymore. The process of verifying the event's duration can also remove some visualization artifacts, i.e. when the interaction between the agents lasts shorter than the animation itself. Such situations can get worser when we have a higher density of agents interacting with each other.

The approach to detect if agents are walking together is a little bit different. In this case, instead of checking the agent's FOV, we analyze the average velocity of each one of them. Since they are going into the same direction, they do not "see" each other using our FOV method. However, by analyzing the average velocity, we can tell if the agents are really walking together (side-by-side) or just passing by each other in the same direction. As the approach previously explained, we also verify the *proxemics*, but for this particular case we opted for Hall's *social distance* interval ($(1.20, 3.60]m$), using two meters as threshold, and the difference of the average velocity must be inferior than $0.1m/s$, value empirically defined based in our tests.

In both approaches we check if the interaction events last longer than six seconds, we decided this time interval based on the current animations length we provide. This is the minimal time, for a pair of agents, to have any sort of interaction. Figure 3.4 illustrates an agent passing by other while moving in the same direction, this case is ignored using the average speed metric and/or the interaction event duration metric.

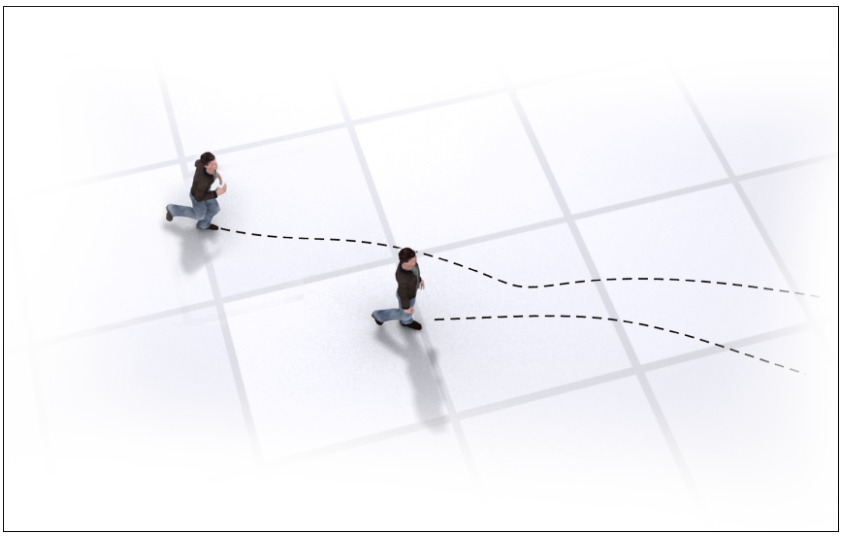


Figure 3.4 – Illustration of an ignored case using our metrics, where an agent pass by another while moving in the same direction.

[1] We adopted the common value of 120^o for representing the human field-of-view [3].

It is important to highlight that the *human interactions* do not change any aspect of the simulation. The animations performed by *CrowdVis* in the agents only provide visual behaviors aiming to represent common real-life situations. All the animations are performed only in the upper bones of the virtual humans. This approach allows the animation blending with *regular* animations, such as *walk* and *idle*.

3.1.2 Human Motion

The animations of embodied agents usually can be seen in three ways: predefined, real time computed animations or a combination of both. Predefined animations can be created by artists using animation capable software and also by motion capture[2]. Real time computed animations commonly are performed by using skeletal structures[3]. In such process, the bones are manipulated according to mathematical models creating the animation. The combination of both techniques is what is often seen in novel 3D games. In this case, the VHs can interact with the scene while blending their predefined animation with a real time IK[4] approach, mostly used for feet landing on top of the terrain or for grasping objects.

CrowdVis approach for embodied agents animation relies on combining a set of predefined animations. Besides that, our model is responsible for playing and blending the animations coherently. We divided into four possible animations, which we called *regular* animations: *idle*, *slow walk*, *normal walk* and *jog*. Our approach is based on analyzing the agent's displacement in order to provide a visually accepted realistic motion for the VHs, which can be seen during the visualization. Figure 3.5 illustrates the *regular* animations of the three movement cases: *slow walk*, *normal walk* and *jog*, using a male virtual human.

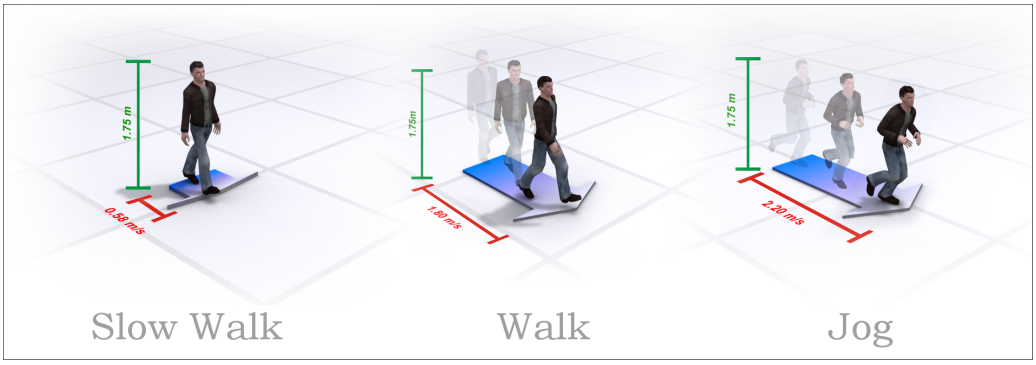


Figure 3.5 – Illustration of the *regular* animations of the three movement cases: *slow walk*, *normal walk* and *jog*, using a male virtual human.

[2]Process of recording the movement of real persons.
[3]Structure often used to animate VHs, allowing to manipulate the geometry using the skeleton bones.
[4]Equations to determine the joint parameters providing a desired position at the end-effector.

In order to provide a visually accepted realistic motion, firstly we need three average velocities for the VHs that match their *slow walking*, *normal walking* and *jogging* animations. The average velocities of a VH relies on a set of factors, such as the height of the human, the animation itself (e.g. *jogging* or *walking*), age, weight, among others. For determining for each movement case the expected average velocity, we simply obtained the distance traveled[5] after one second for that VH. These velocities information are stored in the virtual humans's configuration file, as described in Listing 3.2.

```
1  scale=0.022                          //Virtual human scale
2
3  slowwalkspeed=0.58                   //Slow walk animation speed ratio in one second
4  walkspeed=1.80                       //Normal walk animation speed ratio in one second
5  jogspeed=1.97                        //Jog animation speed ratio in one second
6
7  skeleton=man5-skeleton.CSF           //Cal3d skeleton file name
8
9  animationname=idle                   //Idle animation name to be used inside CrowdVis
10 animation=man5-idle.CAF              //Idle animation file name
11
12 animationname=walk                   //Walk animation name to be used inside CrowdVis
13 animation=man5-walk.CAF              //Walk animation file name
14 animationname=walkslow               //Slow Walk animation name to be used inside CrowdVis
15 animation=man5-walkslow.CAF          //Slow Walk animation file name
16 animationname=run                    //Jog animation name to be used inside CrowdVis
17 animation=man5-run.CAF               //Jog animation file name
18
19 animationname=humaninteractionA      //Default passing by human interaction animation name to be used by CrowdVis
20 animation=man5-wave.CAF              //Human interaction A animation file name
21 animationname=humaninteractionB      //Default walking together human interaction animation name to be used by CrowdVis
22 animation=man5-gesturing.CAF         //Human interaction B animation file name
23
24 mesh=man5-mesh.cmf                   //Virtual human mesh file name
25
26 material=man5-material.CRF           //Virtual human material file name
```

Listing 3.2 – Virtual human's configuration file example illustrating the average velocities previously calculated.

Once we have the average velocities for each movement case, we discover the distance traveled by the agent at each second of the simulation. By analyzing if the agent's displacement is smaller than $0.1m$, the model is responsible for blending the current animation to one of the available *idle* animations. On the other hand, if it is greater or equal than $0.1m$, then one of the three movement animations could be played (*slow walk*, *normal walk* or *jog*). Fruin [16] described intervals for average adult walking velocities. These intervals are used to decide which animation shall be played, as listed bellow:

- *idle* animation $[0.0, 0.1)m/s$;
- *slow walk* animation $[0.1, 0.6)m/s$;
- *normal walk* animation $[0.6, 1.4)m/s$; and
- *jog* animation $[1.4, +\infty)m/s$.

[5]Distance obtained while the VH is moving in a straight line.

By comparing the calculated distance in the simulation with the average velocity stored in the VH's configuration file, we can change the movement animations speed in order to be played slower or faster, making the VH motion more believable and synchronized, preventing that the VHs "slide" on the floor, artifact usually noticed when the translation is faster than the VH walking animation. In order to obtain, for each agent i, the animation speed factor $\mathbf{s_i}$, we measure the distance traveled dt_i after one second of simulation, and then divide it by the current average velocity av_i previously estimated for that movement case (information stored in the VH's configuration file), as illustrated in Equation 3.2:

$$\mathbf{s_i} = \frac{dt_i}{av_i}. \tag{3.2}$$

In *CrowdVis*, the only information required in the *simulation XML* are the positions of the agents recorded during the simulation, separated by frames. The small amount of information required facilitates the output creation for the crowd simulators. On the other hand, we are not able to take advantage of other information that could be provided by the simulator, such as the agent's orientation. The next step in this module is to use the simulation information regarding the agent's position for calculating where the agent should be facing (oriented) at the end of each frame. Figure 3.6 illustrates the virtual humans's orientation path for one second of simulation.

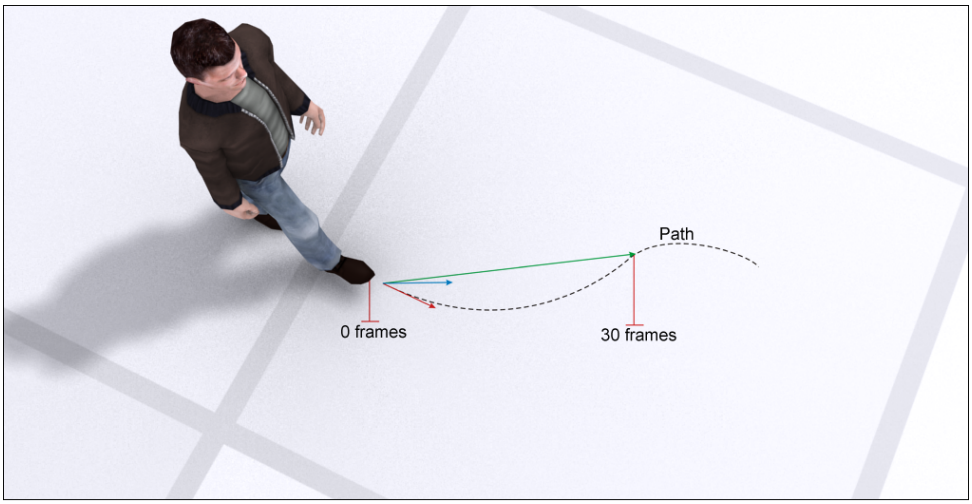


Figure 3.6 – Example of a VH's orientation path for one second of simulation.

For orienting our embodied agents during the visualization, one can calculate a directional vector between the positions regarding the current frame and the next one. This approach creates several visualization artifacts, given that usually the models create the agents motion as points and not bipeds. As previously explained, a set of 3D positions represent exactly one second of simulation (usually one second is represented by 30 3D positions, but it may vary according the input simulation). For this reason, the slower

an agent translates, more positions are plotted for that same agent in the same distance interval. Figure 3.7 illustrates the 3D points provided by a simulation model for a straight path, where the black dots represent the 3D positions and the red lines an illustration of the straight path, desired in the simulation.

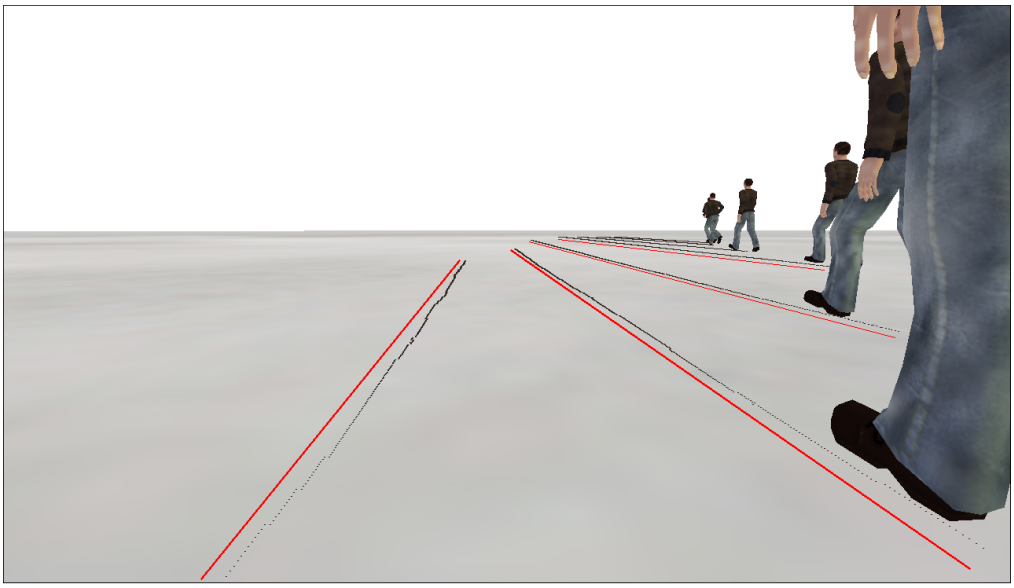


Figure 3.7 – Illustration of the 3D points provided by a simulation model for a straight path, where the black dots represent the 3D positions and the red lines an illustration of the straight path, desired in the simulation.

In order to achieve a better orientation for the agents, we use time coherence in our calculations. As the velocity calculation analyzes the next second of the simulation, we also calculate the resultant vector that indicates the destination that each agent desires to face after one second. We use this data to calculate the orientation $\vec{O}(t)$ for the next frame, where t means the time in frames, $p(t)$ is the current position and $p(t+1)$ is the next position. In our prototype we use the value 30 for N, executing at a rate of 30 frames per second. It is important to mention that, during the sum, the distances between this and the next step must be smaller than 2.5 meters, otherwise it is not taken into account. This approach avoids incoherent behaviors given sudden changes of direction of an agent, as illustrated by Equation 3.3:

$$\vec{O}(t) = \begin{cases} t = 0, & 0 \\ t > 0, & \frac{(\vec{O}(t-1)) + (\sum_{t}^{t+N} p(t+1) - p(t))}{2}. \end{cases} \tag{3.3}$$

3.1.3 Screenplay Animation

Even if *CrowdVis* is in charge of determining a set of *regular* animations to be played automatically, the user must be able to play their own specific animations, which we called *screenplay* animations. These animations can be played at any time during the simulation, for example: *Agent [39] Play [Jump] at Frame [348]*. To possibilitate users to do this, a XML script (see Listing 3.3) can be loaded containing the following information: *agent id*, *name of the animation* and a *start frame*. Figure 3.8 illustrates the virtual human playing a sit down animation at a specific simulation frame.

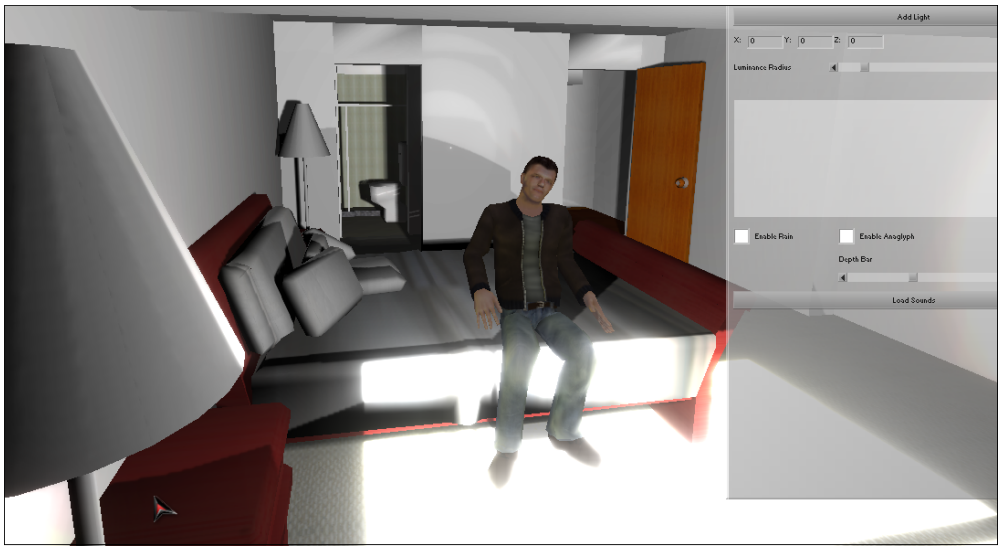


Figure 3.8 – Illustration of a *screenplay* animation provided by the user, where a virtual human starts playing in loop a custom animation at a specific simulation frame.

When a *screenplay* animation is loaded, it is played repeatedly in loop. In case the user does not desire to be in control of the agent's animation anymore, simply need to provide the *animation name* as *clear* in the desired *frame* in order to turn it off (also illustrated in Listing 3.3).

```
1  <SCENE>
2      <ACTONS>
3          <ACTON>
4              <AID>0</AID>              //Agent's ID
5              <NAM>sitdown</NAM>        //Animation name to be played
6              <FRA>100</FRA>            //Start frame for the animation to be played
7          </ACTON>
8          <ACTON>
9              <AID>0</AID>              //Agent's ID (the same as before)
10             <NAM>clear</NAM>          //Clear word for releasing animation control
11             <FRA>200</FRA>            //Frame it should be released back to CrowdVis
12         </ACTON>
13     </ACTONS>
14 </SCENE>
```

Listing 3.3 – Screenplay XML input data example where we can observe the word *clear* for releasing the animation control back to the *CrowdVis*.

CrowdVis must not interfere in *screenplay* animations, in other words, *regular* animations and *human interactions* animations are only calculated in case there are no *screenplay* animations being executed for the current agent. The following pseudo-algorithm illustrated in Listing 3.4 indicate the priorities between the possible type of animations to be played by the VHs.

```
1 IF (screenplay)
2         play screenplay animation;
3 ELSE IF (NOT screenplay AND interaction)
4         play human interaction animation blended with identified regular animation;
5 ELSE
6         play identified regular animation.
```

Listing 3.4 – Pseudo-algorithm indicating the priorities between the possible type of animations to be played by the VHs.

A particular case in our approach can occur when the user loads a *screenplay* animation for a virtual human which is currently involved in an *human interaction*. When this happens, as explained before, our model prioritizes the *screenplay* animation over the *human interaction* animations. In this particular scenario, one agent can interact alone. It is important to reiterate that all the processed and stored interaction behaviors can be turned off at any moment during the visualization, using *CrowdVis* prototype's interface. Appendix A contains a flowchart for more technical detailed information about the *Behavior Analysis* module.

3.2 Virtual Humans Reconstruction Module

To create a virtual human, a set of basic steps must be satisfied such as: modeling, rigging[6], texturing and animating. These steps can take a large amount of time from the artist, situation worsened when the artist needs to create several different virtual humans. For this particular reason we provide in *CrowdVis* a set of VHs that we created and also a module for semi-automatic reconstruction of VHs [8].

The reconstruction approach is semi-automatic, since the user must provide a few clicks locating the joints of the human structure. There are six steps[7] in order to achieve our reconstruction goals: *3D Pose Identification*, *Human Segmentation*, *Silhouette Processing*, *Template Posing*, *Mesh Deformation* and *Texturing*.

Step 3.2.1 is responsible for generating a bone structure based on Taylor work [45], including biomechanical restrictions, and recording the result data in a XML file. This technique works with an user input (2D clicks at the joints of the human of the picture). Step 3.2.2 consists in image processing. The goal at this point is to segment and extract the human in the picture. Step 3.2.3 finds out the body parts width and record it on the

[6] Usually rigging is the name given to the process of preparing the mesh and bones for animating.

[7] Steps 3.2.1, 3.2.2 and 3.2.3 represent models developed by other people in research group and are briefly explained.

same XML file generated by the first module. Steps 3.2.4, 3.2.5 and 3.2.6 are responsible for manipulating all obtained data and generate a fully animated and textured virtual human, using algorithms for reconstructing (mesh deformation) and texturing.

3.2.1 3D Pose Identification

In this first step, the user must click to indicate where in the picture is the human body major joints (see Figure 3.9 left). This data is handled by Taylor algorithm [45], which analyzes every joint location based on human body skeletal dimensions and then determines the estimated depth of each bone. In other words, Taylor's algorithm retrieves the 3D world coordinates of each joint.

This approach generates uncomfortable or even impossible poses for human beings, such as broken arms or twisted legs, due to the ambiguity resulted when applying Taylor's technique. In order to discard these incoherent poses, an approach was proposed in order to check the possible angles between the joints based on the human's biomechanical knowledge [5]. The body biomechanical constraints consider the relationship of bones linked through joints and also the joints Degrees-Of-Freedom (DOF). The results of this step are stored in a data file (*Skeleton Data*) containing each joint position representing the posture according to the picture used as input. Further details of this work can be found at [8].

3.2.2 Human Segmentation

The automatic segmentation of human subjects is still a challenge, mainly due to the influence of numerous real-world factors such as shading, image noise, occlusions, background clutter and the inherent loss of depth information when a scene is captured into a 2D image [25].

The bone structure data, previously clicked by the user, provides information of each body part location, allowing the color model to stipulate a color region for each one of those. The regions are based on human anthropometry [48] creating a structure we called *Bounding Bones*[8], that covers approximately the human body area (see Figure 3.9 center). This region is used for learning and searching the dominant colors that represents the human in the picture. Figure 3.9 shows the initial pipeline of the virtual humans reconstruction where is illustrated the 2D bone clicking process (left), the bounding bones (center) which are the regions used by our color model algorithm and the binary image representing the extracted human segmentation (right).

At the end of this step, for each body part, regions of the picture are selected to be used as texture (see Step 3.2.6). In some pictures the face employed in the template mesh may be not accurate, due to the head orientation (works only with frontal poses). If the

[8]Our adopted concept of bounding box representing each bone.

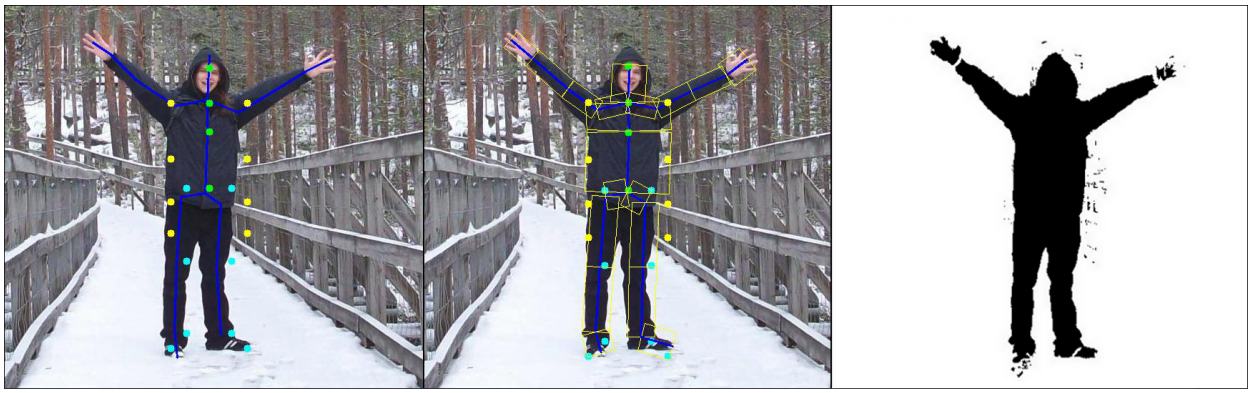


Figure 3.9 – The initial pipeline of VHs reconstruction, where is illustrated the 2D bone clicking process (left), the bounding bones (center), which are the regions used by our color model algorithm, and the binary image representing the extracted human segmentation (right).

result is not visually acceptable, the user can try another picture or manually provide a face texture. Figure 3.10 illustrates the human and the automatic texture chunks obtained for future texturing.

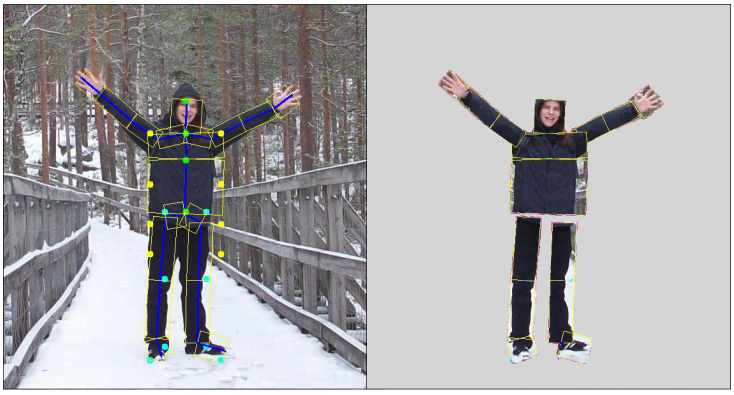


Figure 3.10 – Illustration of the the automatic texture chunks obtained for future texturing.

3.2.3 Silhouette Processing

Given the binary silhouette of the person and the 2D clicked joints data, we estimate the width of each body part. The main idea is to compute the length of line segments connecting a bone and the boundary of the silhouette, robustly estimating the corresponding width.

We retrieve the bones by connecting the joints. For each bone is traced a perpendicular set of line segments. Each line segment ends following two rules: *i)* the next pixel is of white color (end of the human body part). *ii)* the line segment has 35% of the bone length (value estimated given human limbs average width [48]). After collecting all data, the median

for each body part is recorded in a data file (*Body Width*). Figure 3.11 shows the binary silhouette (left) and the estimated widths for each body part (right), where the green line is the median width.

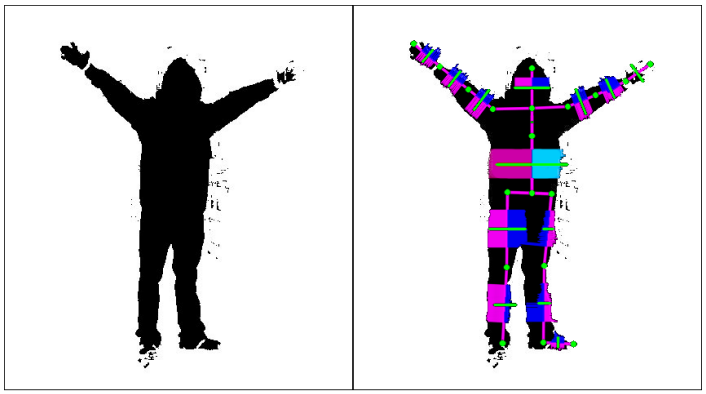


Figure 3.11 – Binary silhouette (left) and the body parts width estimation for automatic reconstruction (right), where the green line is the median width.

3.2.4 Template Posing

This and the next steps are responsible for employing the previously obtained data in order to make the human template mesh to look like the human in the picture. *CrowdVis* prototype contains only a few buttons and panels in order to enable the user to load a *JPG picture* file. All the other data is automatically loaded. Figure 3.12 illustrates *CrowdVis* VH reconstruction interface.

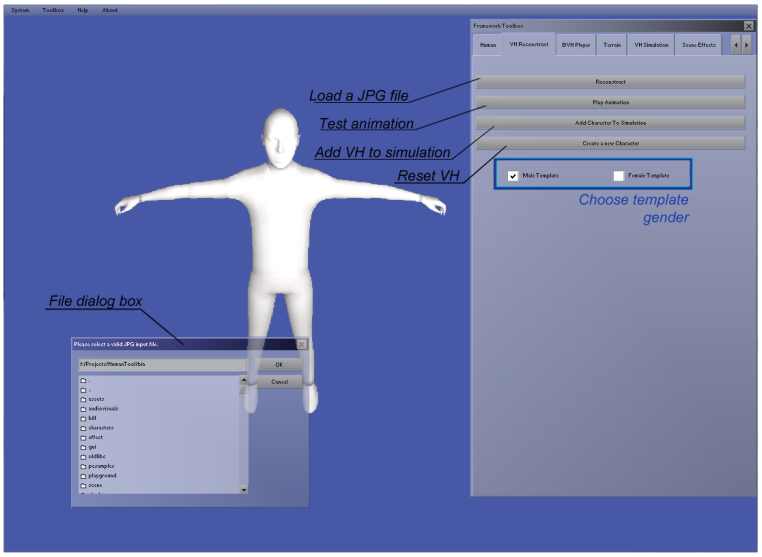


Figure 3.12 – Illustration of *CrowdVis* VH reconstruction interface.

In this model, we are not able to decide the gender of the human by just processing the picture, so the user needs to previously decide if the template model will be male of female, each one containing 4825 and 4872 vertexes, respectively. Figure 3.13 illustrates the male (left) and female (right) template models in their initial state (T-like posture, without any mesh deformation and texture information).

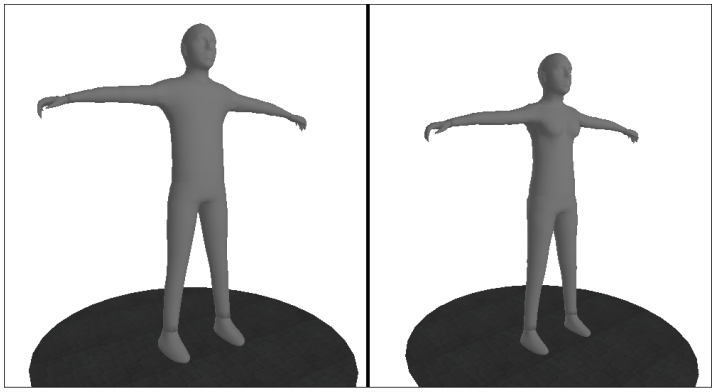


Figure 3.13 – Illustration of the male (left) and female (right) template models in their initial state (T-like posture, without any mesh deformation and texture information).

The VH template initially adopts a standard T-like posture, as illustrated in Figure 3.13. We represent each bone i of the virtual human vh by a vector $b_{i_{vh}}$ given by the difference of its final joint position and its initial joint position, observing the skeleton hierarchy. Its correspondent bone from the real human of the picture p (recorded in the *Skeleton Data* file explained in Step 3.2.1) is represented by the vector b_{i_p}. We can find out the angle Θ_i between these vectors by computing the arccosine of their dot product. After that, it is possible to determine the quaternion q_i in order to rotate $b_{i_{vh}}$ to achieve the desired orientation (i.e. the orientation of b_{i_p}, where $\vec{n}_i$ is computed by the cross product between $b_{i_{vh}}$ and b_{i_p} as shown in Equation 3.4:

$$q_i = (cos(\frac{1}{2}\Theta_i),\ sin(\frac{1}{2}\Theta_i)\vec{n}_i). \tag{3.4}$$

Our approach can also generate angle ambiguity for rotations, this is fixed by comparing the joints 3D position with the ones provided in the *Skeleton Data* file. Each quaternion, describing the rotation of every joint of the VH template, is multiplied by the correspondent quaternion stored in the skeleton. In other words, we can determine the quaternion q_i in order to rotate $b_{i_{vh}}$ to the desired orientation (i.e. the orientation of b_{i_p}), according to Equation 3.5:

$$b'_{i_{vh}} = q_i\ b_{i_{vh}}\ q_i^{-1}. \tag{3.5}$$

3.2.5 Mesh Deformation

After rotating the bones of the VH template to match the 3D pose identified in the picture we deform the mesh of the VH template using the *Body Width* data obtained in Step 3.2.3. Given the width of each body part w_i and the length of the bone l_i recorded in the XML files (described in steps 3.2.3 and 3.2.1) from the real human in the picture, we compute for each body part i in the picture p, the ratio $r_{i_p} = \frac{w_{i_p}}{l_{i_p}}$. The correspondent ratio $r_{i_{vh}} = \frac{w_{i_{vh}}}{l_{i_{vh}}}$ is computed for the virtual human vh in its original form. Using these ratios, we compute a vertex displacement factor f_i according to the Equation 3.6:

$$f_i = r_{i_p} - r_{i_{vh}}. \tag{3.6}$$

Then, for each vertex of the virtual human, we apply such displacement factor in the direction of the normal vector $\vec{n}_v$. Equation 3.7 calculates the vertex new position $\mathbf{v}' = (x'_v, y'_v, z'_v)$ related to its original position $\mathbf{v} = (x_v, y_v, z_v)$:

$$\mathbf{v}' = \mathbf{v} + f\ |l_{i_{vh}}|\ \vec{n}_v. \tag{3.7}$$

In some cases, given a problem during the segmentation process of the picture and/or the silhouette processing, the width factor of some computed body parts can be too big or too small than expected, as illustrated in the right thigh of Figure 3.11 (right). In order to avoid incoherent deformations there are two rules in this step: *i)* case the *calf* factor is greater than the *thigh* factor, then the *thigh* uses the *calf* factor. *ii)* case the *forearm* factor is greater than the *arm* factor, then the *arm* uses the *forearm* factor.

Since the virtual human is divided in a set of body parts, after displacing all the vertex to their correspondent deformed positions, the VH's mesh needs to be "stitched" at each body part intersection in order to prevent visualization artifacts, as illustrated in the examples (a), (b) and (c) in Figure 3.14. For stitching up the vertexes we created a based on the original template mesh associating the boundary vertexes closer than one centimeter, value empirically obtained given the template mesh geometry topology.

(a) (b) (c)

Figure 3.14 – (a), (b) and (c) are illustrations of reconstructed virtual humans without being stitched after the vertex displacement process.

3.2.6 Texturing

Texturing is definitely an important task to be done, not only in order to increase the similarity between the virtual human and the real human in the picture, but also for the VH to be used in a realistic crowd visualization. Figure 3.15 illustrates the template model without (left) and with (right) texture, where the color information increase the realism of the VH.

Figure 3.15 – Illustration of the template model without (left) and with (right) texture, where the color information increases the realism of the VH.

The challenge in texturing the reconstructed virtual humans is due to the fact that a single 2D picture does not provide the information of occluded body parts. In other words, this lack of information does not allow to fully paint a 3D model with the correct color information (such as the VH template back part). The approach used in this work aims to ensure that the frontal part of the VH is similar to the human in the picture, since we have the VH in the same posture than the picture. The idea is to simply remap the texture coordinates for the head, chest and belly body parts, creating a planar technique to plot the previously obtained textures. Figure 3.16 illustrates the difference between default UV mapping (left) and the *planar mapping* (right) techniques.

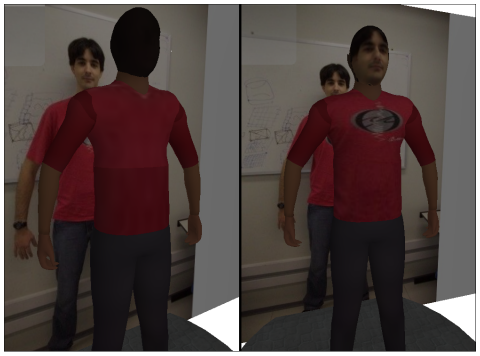


Figure 3.16 – Illustration comparing the default UV mapping (left) and the *planar mapping* (right) techniques.

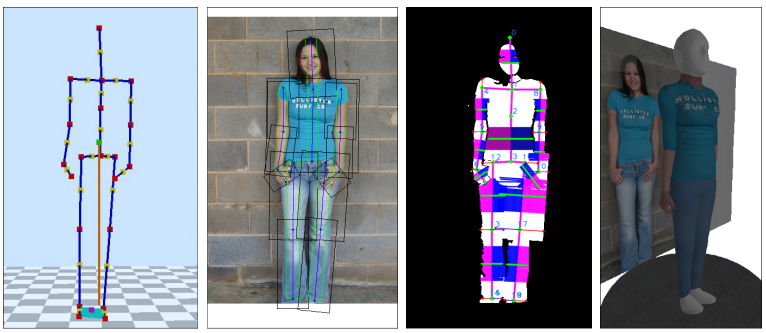
In order to execute what we called *planar mapping* technique for the body parts described above, we first search for the six extreme vertexes (left/right most, top/bottom most and near/far most). After this process, we can retrieve the frontal visible part (group of vertexes) for each one of the three body parts. For each group of vertexes we remap each vertex's texture coordinate and calculate the vertex new texture coordinate for X axis $\mathbf{vtx}'$ and Y axis $\mathbf{vty}'$, where r_x and l_x represents the right most and left most extreme vertex of the X axis, respectively, as shown in Equations 3.8 (X axis) and 3.9 (Y axis):

$$\mathbf{vtx}' = \frac{\mathbf{x} - r_x}{\|r_x - l_x\|}, \tag{3.8}$$

$$\mathbf{vty}' = 1 - \left(\frac{\mathbf{y} - t_y}{\|t_y - b_y\|} \right). \tag{3.9}$$

The *CrowdVis* prototype (presented in next chapter) is able to export the reconstructed characters to OBJ[9] format files. Once the VHs are exported, artists are able to fix texture artifacts created during the reconstruction, adjust the mesh topology or even use the VHs for other purposes, such as in a game. The drawback is the loss of the default animations presented in the reconstructed VHs, since this information is not supported by OBJ files.

Although the reconstruction may not be highly accurate, the whole creation process is fast, and it provides good visual results that maintain coherence with the original picture. The *CrowdVis* VHs are all *Cal3D*[10] format, which means that users can load any character of their own, created with any tool, if they are in the proper format. User's customized characters must follow the format of the agent configuration file (see Listing 3.2). Figure 3.17 illustrates the pipeline of VHs reconstruction.



(a) (b) (c) (d)

Figure 3.17 – Illustration of the VH reconstruction pipeline, where (a) is the estimated 3D pose detected, (b) is the segmentation and texture chunks attainment process, (c) illustrates the silhouette processing and (d) is the result of mesh deformation and texturing techniques.

[9] File format developed by Wavefront Technologies containing geometry information.
[10] http://gna.org/projects/cal3d/

4. THE *CROWDVIS* PROTOTYPE

As mentioned before, *CrowdVis* uses an approach which detaches the visualization from simulation aspects. In this case, the advantage is the possibility of using different crowd simulators and only be concerned with the visualization itself. This chapter aims to present the *CrowdVis* prototype's audio and visual features, which include rendering optimizations techniques and the available special effects, that can be used for enriching the visualization.

4.1 Audio-Visual Module

Once we have the simulation data and the virtual humans, we can work to provide a good visualization in real time frame rates[1], that also includes setting up the *environment*, configuring the *virtual humans* visualization along with LOD techniques, lighting control and also a set of *special effects* in order to include sound, shadows, stereoscopic and particle rendering.

4.1.1 Environment

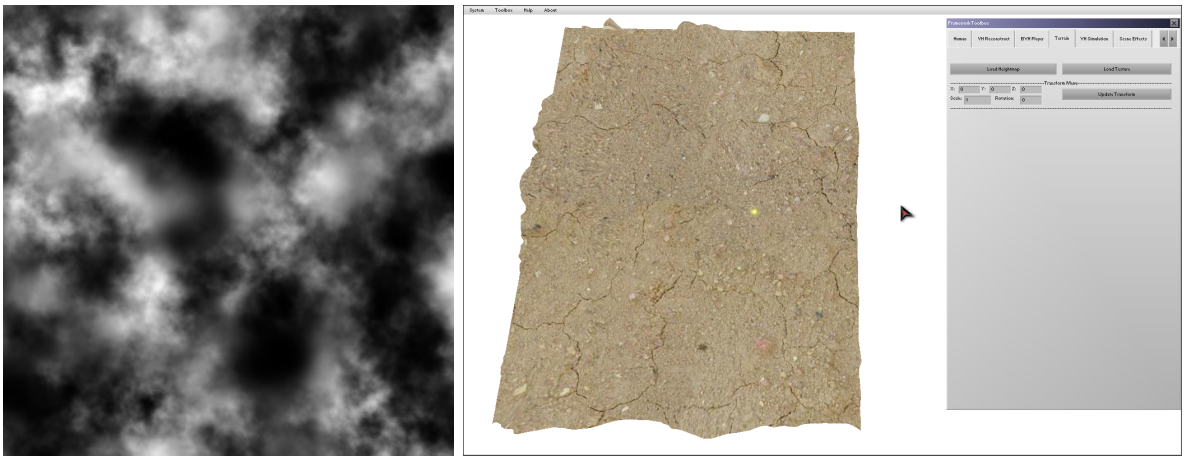
When simulating crowds for evacuations, emergency situations or any other purposes, it is important to be able to visualize the script which the simulation is running on. For example, either simulating the end of a soccer match or people running to get aboard lifeboats, visualizing the halls, doors and obstacles helps comprehension of the situated environment being simulated. For this reason, the framework provides tools for loading and manipulating 3D assets. Since *CrowdVis* uses *Irrlicht Engine*[2] for the fixed pipeline rendering, a set of assets formats can be used, which include: *B3D* (.b3d), *Microsoft DirectX* (.x), *Quake* (.md2, .md3 and .bsp), *3D Studio* (.3ds), *Alias Wavefront Maya* (.obj), among others.

Some of the crowd simulations also aim to evaluate the motion of the virtual humans in realistic environments. To represent these scenarios, such as terrains, can be a difficult and complex task, since natural scenarios are composed by forms with a lot of details that should be reproduced realistically in the virtual world [9]. The 3D terrains can be represented by a data structure called *heightmaps*, which stores the height values of the terrain using a matrix of points [30].

The *heightmaps* matrices can be stored as textures, in image files such as *BMP* or *PNG*, as illustrated in Figure 4.1 (a), where the brighter parts represent the higher parts of the terrain. We enable the users to visualize terrain-driven simulations, for this, the users can

[1] We consider to be real time when the frame rate executes at 30 frames per second or more.
[2] http://irrlicht.sourceforge.net/

load a *heightmap* image (smaller than 256x256 due to the implementation optimizations), and the users can also customize a few terrain properties: terrain's texture, size, orientation and position (see Figure 4.1 (b)).



(a) (b)

Figure 4.1 – Illustration of a *heightmap* image (a), where the brighter parts represent the higher parts of the terrain. The users can also customize a few terrain properties: terrain's texture, size, orientation and position (b).

In *CrowdVis*, any loaded asset to be used as part of the scene is marked as *static* in the GPU caching, which means that the vertexes and indexes rarely will be changed, for this reason, dynamic meshes are not supported. On the other hand, this approach enables a faster rendering time for the visualization. There are two possible ways of loading the scenario: *i)* manually choosing the mesh files (one at a time) and positioning them in the scene. *ii)* Using a script (*scene XML* file) which contains the name of the mesh file, position in the virtual world, its orientation (0^o to 360^o) and its scale (three values, one for each dimension), as described in Listing 4.1.

```
1  <SCENE>
2          <OBJCTS>
3                  <OBJCT>
4                          <NAM>tree.3ds</NAM>                    //Name of the object to be loaded
5                          <POS>5 0 0</POS>                       //Position of the object (X,Y,Z)
6                          <ROT>0 90 0</ROT>                      //Rotation of the object (X,Y,Z)
7                          <SCA>0.8 1 0.8</SCA>                   //Scale of the object (X,Y,Z)
8                  </OBJCT>
9                  <OBJCT>
10                         <NAM>bench.obj</NAM>
11                         <POS>7 0 -2</POS>
12                         <ROT>0 0 0</ROT>
13                         <SCA>1.5 1 1</SCA>
14                 </OBJCT>
15         </OBJCTS>
16 </SCENE>
```

Listing 4.1 – Scene XML input data example for two objects.

Besides being able of positioning, rotating and scaling the scene assets, the user can also change a few object's properties: *wireframe rendering*, *transparent material*, enable

or disable *cullface* and enable or disable if object receives *lighting*. Figure 4.2 illustrates the manipulation of scene assets and the interface, indicating the tools presented for manipulating objects and light source positions, orientations, among others.

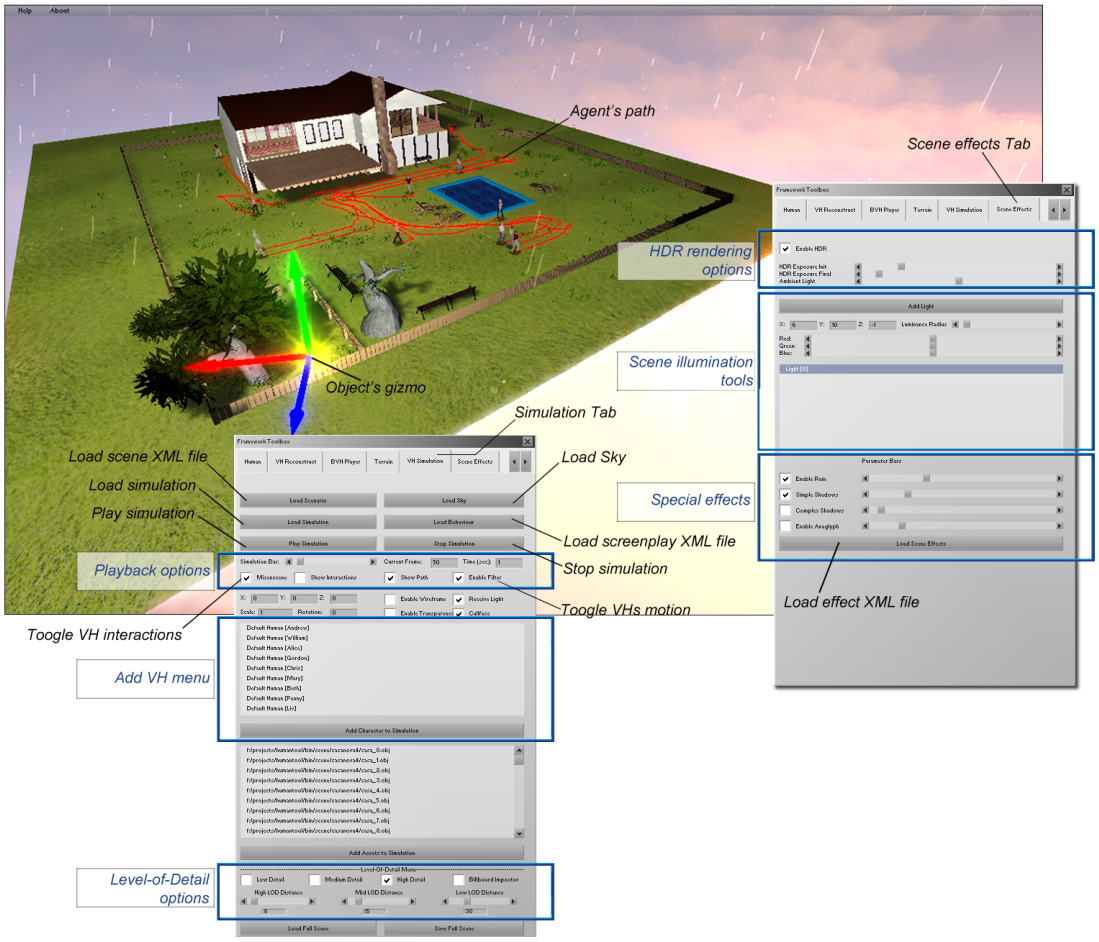


Figure 4.2 – Illustration of the manipulation of scene assets and the interface, indicating the tools presented for manipulating objects and light source positions, orientations, among others.

It is also important to navigate into the scenario easily. We provide three types of camera control:

- *First Person Camera* - control the camera using the Arrows/WASD keys and mouse;
- *3rd Person Camera* - by clicking over a virtual human the camera follows the agent enabling the scroll wheel for zooming and right and middle mouse buttons for height adjust; and
- *Script Camera* - flies through a set of 3D points in the scene using our prototype's interface.

4.1.2 Virtual Humans

The virtual humans are for sure the most important part of our visualization, without them we would have only points or cylinders moving around the scene, what is usually how the humans are represented in crowd simulators. As explained in Section 3.1, our virtual humans are capable of self adjusting their animation speed, detecting possible *human interactions* among them and playing *screenplay* animations provided by the user. In order to do that and achieve real time frame rates we worked on a few optimization techniques, such as *Level-Of-Detail*, *animated impostors* and *normal maps*.

Levels-Of-Detail [12] can be explained as different representations of the same object that can be used in a way to improve performance. The common approach in games and other real time applications is to use LOD bounded to the geometry of the object, where these objects possess several layers of mesh simplifications, the simpler is the geometry the lesser is the memory consumption and higher is the computation speed of vertexes transformations in the GPU.

Progressive Mesh is a mesh simplification technique developed by Hughes Hoppe [24] in 1996. The author presents a mesh simplification procedure which the main idea is to preserve the geometry of the original mesh and its overall appearance making the usage of material identifiers, color values, normals, and texture coordinates. Using this technique, besides simplifying our VHs meshes, we can also maintain their appearance making barely unnoticeable the geometry popping[3] when navigating with the camera through the scene. Figure 4.3 illustrates the three Levels-Of-Detail, low quality (left) medium quality (center) and high quality (right), with the respective numbers of triangles and polygons.

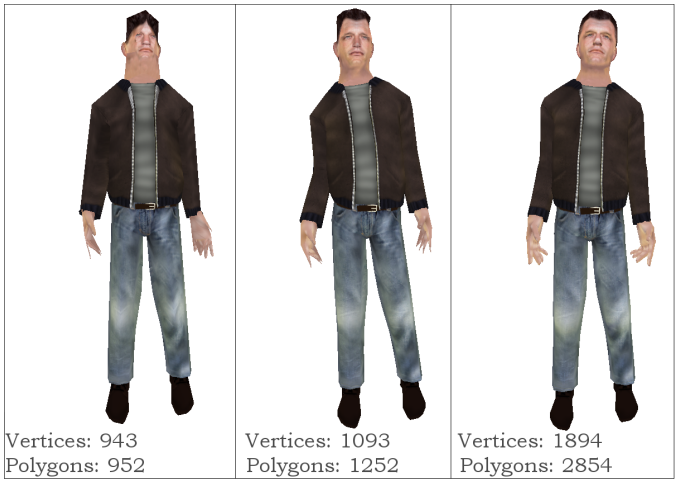


Figure 4.3 – Illustration of the three Levels-Of-Detail, *low* quality (left) *medium* quality (center) and *high* quality (right), with the respective number of triangle and polygons.

[3] In computer graphics the abrupt geometry changes for different Levels-Of-Details are known as popping.

Rendering geometries with low amount of polygons is definitely faster for the GPU, but the visual quality of these meshes comes up worse than the expected. In 1978, James Blinn [6] created the concept of *bump mapping*. In a few words, the idea is to replace the vertexes normals by a *heightmap* texture in order to calculate the lighting for the 3D object. Further, this technique was improved using normal map [29] textures, where each pixel (using RGB channels) represents a normal vector for lighting calculations. Both approaches are very handy for visualizing simple geometry meshes with better details. Figure 4.4 illustrates one of our virtual humans using normal map technique (left) and using regular lighting technique (right).

Figure 4.4 – Illustration of a virtual human using normal map technique (left) and using regular lighting technique (right).

There are other techniques to reduce the amount of polygons presented in the embodied agents than just simplifying the 3D geometry [26]. The common approach is to create a single image of the agent, drastically reducing the geometry necessary for representing the virtual human, on the other hand, a lot of information is lost, such as lighting, animation and depth information. Image-Based Rendering has received a large set of research results [46], in *CrowdVis* we use these techniques for rendering animated impostors, which is how the VHs are represented when they are far away from the camera.

For incorporating animations in impostors, two approaches are commonly used: *i)* previously store the animation frames as textures and play it during simulation. *ii)* render the textures dynamically in real time [41] for each character. Both approaches consume memory and GPU, besides that, rendering textures in real time for a large number or agents can be computational expensive, leading to the necessity of other rendering techniques. In *CrowdVis* we opted for the first approach, using a previously created set of textures for the impostors.

Usually in crowd impostors techniques, there is a trade-off between memory and visual quality. The amount of necessary video memory can be significantly larger if the agents are represented by more images (e.g. the usage of an extra set of images for animation).

CrowdVis combines an animated impostors technique with the agent's geometry LOD, which means that distant agents in our visualization are replaced by view-dependent impostors, same approach presented by Maciel [31]. Each impostor contains a set of 3072 textures, each one with 128x128 resolution. These textures are previously processed and are stored in the Hard Disk Drive (HDD). Figure 4.5 illustrates part of our set of impostors texture for a single agent.

Figure 4.5 – Illustration of four textures (front, right, back, left) of the 3072 impostor texture set. All the textures are previously generated and stored in the hard disk drive, each one with the resolution of 128x128 pixels.

According to the camera point-of-view, the direction that the agent is facing (agent's orientation) and the current animation frame, it is possible to determine which one of the 3072 textures should be used as impostor for that frame. This algorithm is only executed for those agents which are at a specific distance from the camera, saving CPU resources. Equations 4.1 and 4.2 present the approach for determining the best texture (according to the agent's orientation in relation to the camera point-of-view) using the spherical coordinate system for 64 angle intervals, where az is the azimuth, lat is the latitude, θ is the angle computed in the horizontal plane, φ is the angle computed in the vertical plane, K is the constant that represents the number of slices (*CrowdVis* uses 4 as K value) and r is the squared distance between the agent and the camera. The computed values are within the following domain: $0 \leq r < \infty$, $0 \leq \theta < 2\pi$ and $0 \leq \varphi < \pi$. Follow the equations:

$$az = \left\lfloor \frac{4\theta}{\pi} \right\rfloor, \tag{4.1}$$

$$lat = \begin{cases} \theta < \pi, & \left\lfloor \frac{K\varphi}{\pi} \right\rfloor \\ \theta \geq \pi, & \left\lfloor \frac{K(\varphi+\pi)}{\pi} \right\rfloor. \end{cases} \tag{4.2}$$

Once we obtain the computed angles we can now change the animated impostor main texture to the best suitable one, as illustrated in Figure 4.6. *CrowVis* provides two animations for the impostors: *idle* and *walk*. Each animation has a set of 24 textures and are represented

by 64 different camera angles around the virtual human (eight azimuth slices and another eight slices for latitude). We opted for using two animations due to the memory required for loading all the texture set. Besides that, the user can also choose the frame frequency of the impostor's animation ($1Hz$ to $24Hz$), according to the user's computational power required for rendering. Table 4.1 presents the amount of video memory used and Frames Per Second (FPS) when loading a scene with nine different VHs instances, using only animated impostors at a $24Hz$ animation frequency.

Table 4.1 – Comparison between the number of agents illustrating the amount of video memory used and the frame rate (in seconds) for nine different VHs instances, using only animated impostors at a $24Hz$ animation frequency.

Number of Agents	Video Memory (MB)	FPS
100	1.269	195
210	1.279	140
4200	1.625	16

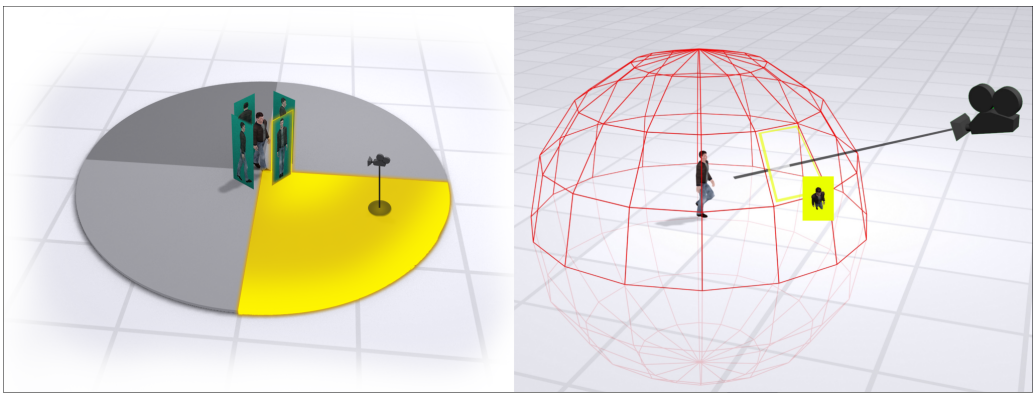


Figure 4.6 – Illustration of *CrowVis* approach for determining the best suitable animated impostor texture according to the agent's orientation and the camera point-of-view. On the right we can see the sphere illustrating the 64 slices for the different available perspectives.

Table 4.2 presents the distances, the number of polygons and the LOD being utilized in our views for only one of our virtual humans in the scene, the distances were decided based on quantitative and qualitative tests. These tests considered the FPS in a crowded environment and the geometry popping artifact for the agents. The LOD distances can be adjusted by the user at any moment of the visualization.

Once a VH is loaded into *CrowdVis*, we create a static structure for storing them. This approach is faster than loading dynamically the geometry for each animation frame. On the other hand, it complicates the usage of a continuous LOD. Figure 4.7 illustrates the exact same scene and camera view for 100 agents without (left) and with (right) LODs.

Table 4.2 – Comparison for evaluating the FPS using the distances, the number of polygons and the LOD utilized for one virtual human in the scene. These LOD distances can be adjusted using *CrowdVis* prototype's interface according to the user need.

Camera Distance	Level-of-Detail	Number of Polygons	Frames Per Second
[0:4] meters	High Detail	2856	722
[5:9] meters	Medium Detail	1254	782
[10:25] meters	Low Detail	954	798
[25:above] meters	Impostor	2	900

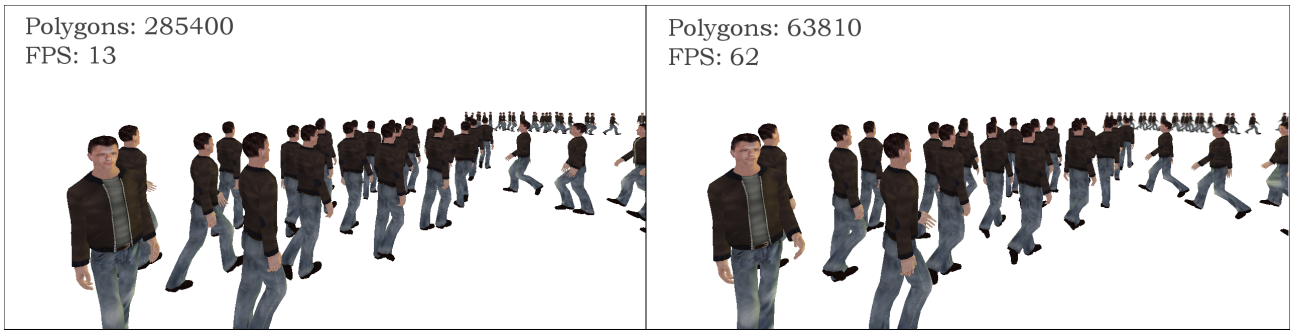


Figure 4.7 – Illustration of the exact same scene and camera view for 100 agents in two different renderings: without LOD technique (left) and with LODs technique and impostors (right).

For large scale simulations the user can represent the agents as animated impostors only. This approach is faster with good visual results, but the drawback are the loss of multiple animations and the *human interaction* events. Figure 4.8 illustrates *CrowdVis* prototype rendering 5000 animated impostors for a large scale simulation.

Figure 4.8 – Illustration of 5000 animated impostors for large scale simulations. The drawback of this approach is the loss of multiple animations and the *human interaction* events.

We also provide the possibility of loading *environmental agents*. These virtual humans are not part of the simulation and can be loaded only with the purpose of enriching the scene visualization. For loading them the user just fill a set of parameters located in *CrowdVis* prototype's interface corresponding to the agent's position, scale, rotation and animation (played in loop during the entire visualization).

4.1.3 Special Effects

This part is responsible for several tasks in order to improve the framework visual quality and to enrich the visualization. At each new day the hardware industry enables better graphics, and the latest engines provide several techniques to create such graphics, usually these techniques can execute during the frame loop rendering or even as post process rendering shader. In order to achieve good graphical results and enable the user some features for visualization, we included a few set of techniques related to real time rendering, usually employed in games.

The usage of shadows helps to improve the overall realism of the scene and also provides visual cues that assist the perception that anchors the agents to the ground [46]. In order to provide more realism during visualizations, *CrowdVis* proposes two shadowing techniques for the VHs.

The user may choose one of them or neither (removing the shadows optimize the visualization performance). The simpler and cheaper computational technique is largely used in old video game titles, it consists in loading a plane with a "circular" texture beneath the objects. Figure 4.9 illustrates two types of shadowing technique present in *Tomb Raider*[4] game series: the simple circular shadow plane (left) and a stencil buffer technique (right). These shadow textures have 128x128 of pixel resolution, besides that, each texture is adapted to the scale of the VH.

Figure 4.9 – Illustration of two types of shadowing technique present in *Tomb Raider*[4] game series: the simple circular shadow plane (left) and a stencil buffer technique (right).

[4]http://www.tombraider.com - Tomb Raider series is published by Eidos Interactive.

The other shadowing technique present in *CrowdVis* improves the visual results considerably, but it is more computational expensive. This shadowing approach is based on shadow maps technique. Shadow maps concept was created in 1978 by Lance Williams [51] and the main idea is to use a light source projection and a target, in order to create the shadows from that perspective. We offer three possibilities: *i)* attach the light source to the camera, *ii)* attach the light source to one of the scene lights. *iii)* attach the light source to one of the VHs, above the head. Figure 4.10 illustrates both shadowing techniques present in *CrowdVis* for visual comparison.

Figure 4.10 – Illustration of both shadows techniques present in *CrowdVis*: simple shadowing (left) and complex shadowing (right).

CrowdVis complex shadowing technique uses a 1024x1024 texture, and it is implemented in the GPU for real time results. Even though, it severely decreases performance, so we recommend to only use this approach when visualizing a small set of VHs. Table 4.3 shows the number of agents and FPS when shadow technique is enabled and disabled. When the agents are rendered as animated impostors they can only use the simple shadow technique due to the lack of geometry for calculating the shadows.

Table 4.3 – Comparison between the impact on performance when using the shadows techniques, according the number of agents.

Number of Agents	FPS No Shadow	FPS Simple Shadow	FPS Complex Shadow
50	102	97	87
100	57	55	45
150	41	37	30
200	29	27	22

Another method for obtaining better visual results was first introduced by Gregory Ward [50], nowadays the technique is commonly known as *High Dynamic Range* (HDR) rendering, it consists of creating a greater range of luminance between the brightest and darkest areas for each frame. *CrowdVis* HDR rendering is a post process technique[5]. As the shadowing technique, the user can also enable or disable the HDR rendering, case the visualization is overflowed due to a large crowd or polygonal count. Table 4.4 compares the HDR rendering against the fixed pipeline rendering according to the number of agents and polygons in the scene.

Table 4.4 – Comparison of the impact on performance when using the HDR technique according to the number of agents and the number of polygons in the scene.

Number of Agents	Number of Polygons	FPS HDR Disabled	FPS HDR Enabled
50	15614	102	96
100	32734	57	55
150	48724	41	38
200	70868	29	28

Figure 4.11 illustrates the usage of the HDR rendering in a scene. On the left is the scene with the default fixed pipeline rendering. On the right is the same scene with the HDR rendering.

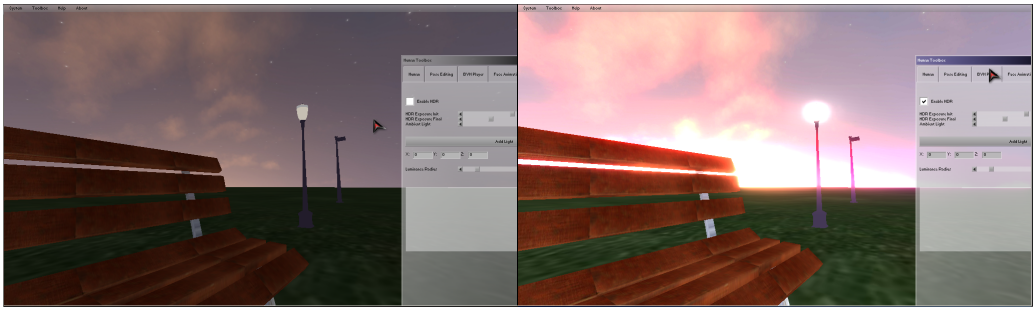


Figure 4.11 – Illustration of a scene with the default rendering (left) and the same scene with HDR rendering (right), both using the same dusky sky.

Crowd simulations results can be used for analyzing human motion in a scene with obstacles for different purposes. Specifically, it can be used in order to analyze the human behaviors during stress situations (e.g. evacuations), climatic conditions or any other events [43]. For example, a evacuation scenario after an explosion or fire situation could represent

[5]At the end of each frame, before sending out the final image to the screen, it is deviated to a shader for processing.

the stress of the agents where their motion should be different from non stress situations. A climatic example can be seen when comparing the crowd motion between sunny and rainy days, where crowds could move towards shelters to avoid getting wet. Having these in mind we decided to enable the user to create such situations by incorporating into the framework particle systems.

Particles in computer graphics was introduced by Reeves [37] in 1983, where the main idea was to create fuzzy objects that hardly can be represented using 3D meshes. Particle systems are largely used in movies and games for representing nature elements, such as fire, rain, clouds and water.

For providing nice particles, we use *SPARK Particle Engine*[6], responsible for handling the particle systems and particle's emitters[7]. Figure 4.12 illustrates two examples of particle systems present in *CrowdVis*: rain (left) and fire (right).

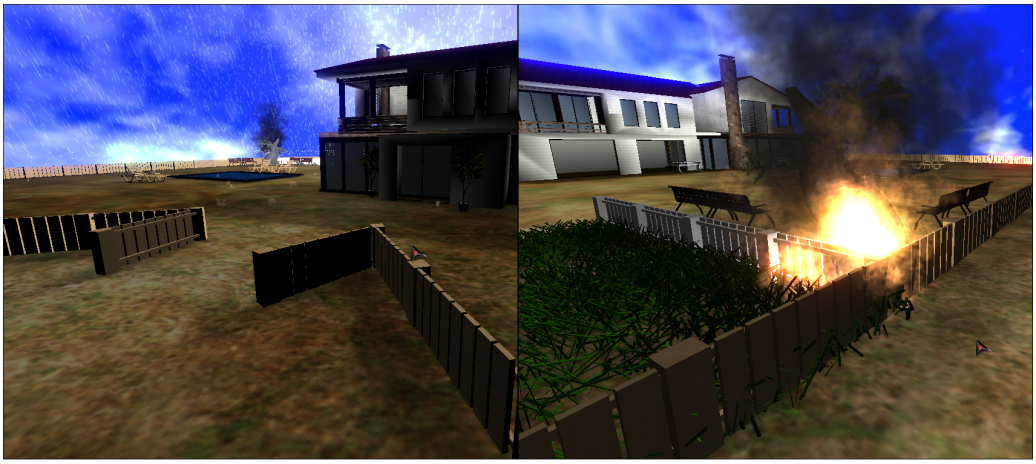


Figure 4.12 – Illustration of two examples of particle systems present in *CrowdVis*: rain (left) and fire (right).

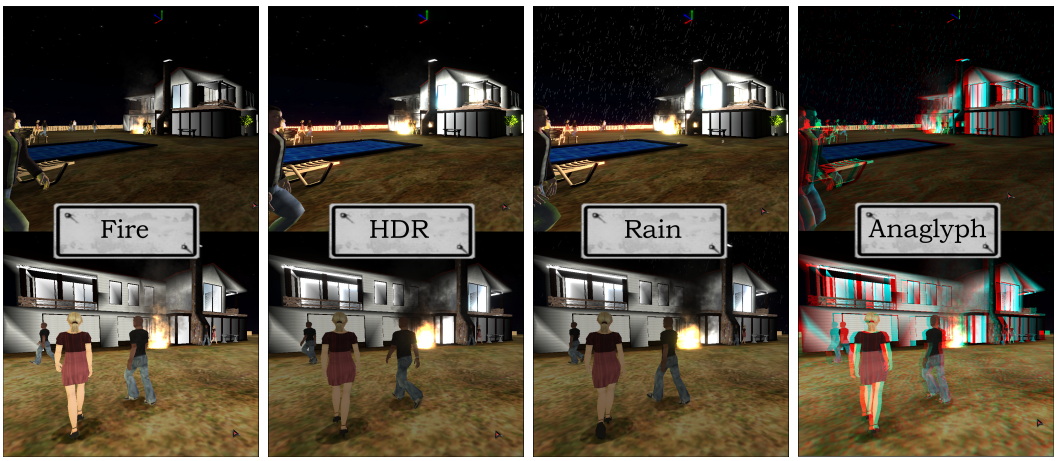
CrowdVis also provides Music and sound effects during visualizations, increasing the immersion feeling of users. According Schell [42], "*the audio feedback is much more visceral than visual feedback, and more easily simulates touch*". *IrrKlang* [8] is an audio library that is attached in our visualizations allowing the user to play sounds and musics using different audio file formats which include: *Free Lossless Audio Codec* (.flac), *MPEG-1 Audio Layer 3* (.mp3), *Ogg Vorbis* (.ogg) and RIFF WAVE (.wav). It is important being able to decide which sound and when to be played during the visualization, in order to do that, we use an *effect XML* file containing not only the sound files path, but also the visual effects as rain, explosion and fire. This file contains which frames the effect should start and end, as well other parameters as seen in Listing 4.2.

[6]http://spark.developpez.com/
[7]The emitters are regions in space where the particle are created inside.
[8]http://www.ambiera.com/irrklang/

```
<SCENE>
        <AUDIOS>
                <AUDIO>                         //Sound Effect
                        <FIL>rain.mp3</FIL>     //Sound file name (must be in the same folder)
                        <FRA>160</FRA>          //Frame to start playing
                        <LOP>true</LOP>         //If should be played in loop
                </AUDIO>
        </AUDIOS>
        <VISUAL>
                <VISUA>                         //Visual Effect
                        <TYP>rain</FIL>         //Name of the effect (only three available: rain, explosion, fire)
                        <ONN>100</ONN>          //Frame to start
                        <OFF>-1</OFF>           //Frame to end (case negative it is never ended)
                        <POS> 0 0 0 </POS>      //Position of the particle emitter (X,Y,Z)
                        <SCA> 1 1 1 </SCA>      //Emitter scale (X,Y,Z)
                </VISUA>
                <VISUA>
                        <TYP>fire</FIL>
                        <ONN>250</ONN>
                        <OFF>500</OFF>
                        <POS> -5 0 0 </POS>
                        <SCA> 1 1 1 </SCA>
                </VISUA>
        </VISUAL>
</SCENE>
```

Listing 4.2 – Effect XML Input data example.

CrowdVis also provides tools enabling the user to manipulate the scene light sources, sky and objects. An amount of eight light sources is restricted due to implementation limitations, but for each light the user can easily configure its position, intensity and color. For the sky the user can load an image that will be placed as a sky dome[9]. We also created an anaglyph rendering for 3D stereoscopic effect that can be enabled and disabled at any time. Figure 4.13 illustrates some effects presented in this section, where (a) we can see the fire effect, (b) the HDR rendering, (c) particle system created for rain and (d) the anaglyph technique for stereoscopic rendering.



(a) (b) (c) (d)

Figure 4.13 – Illustration of some effects presented in this section, where (a) we can see the fire effect, (b) the HDR rendering, (c) particle system created for rain and (d) the anaglyph technique for stereoscopic rendering.

[9]Unreachable dome shape geometry used for rendering background scenes, such as mountains and skies.

5. RESULTS

This chapter presents the results obtained with *CrowdVis*. The experiments were performed using an Intel Xeon W369 equipped with NVidia Quadro 4000. The creation of the virtual humans are made using Autodesk 3D Studio MAX 2010[1]. We divided the results into sections for better organization. Section 5.1 shows the results obtained from our module regarding the reconstruction of virtual humans. In Section 5.2 we compare the techniques from *CrowdVis* with *Unity 3D*[2] engine, and also illustrate some scenarios and the usage of two different crowd simulators (explained in Section 5.2.1).

5.1 Virtual Humans Reconstruction

Figure 5.1 illustrates the reconstruction pipeline proposed in this work. In this case, it is possible to observe problems that happen in segmentation. The segmentation result impacts in errors in the computed width of legs, which can be solved based on the simple rules specified in Section 3.2.5.

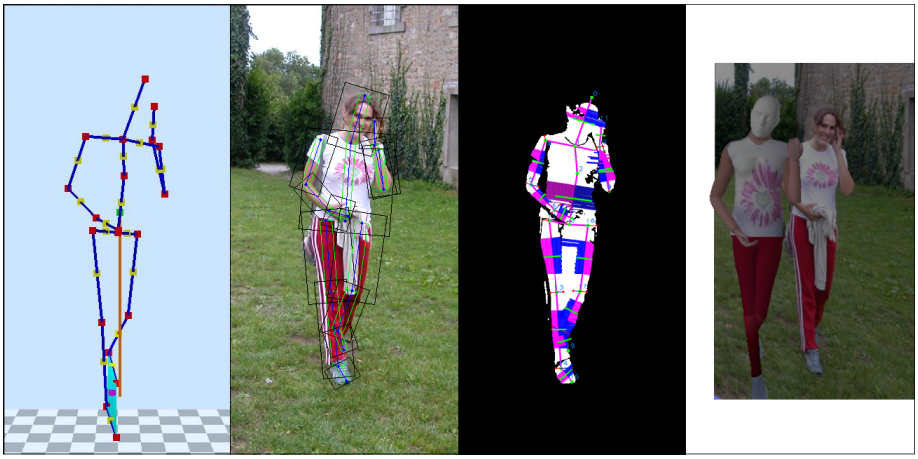


Figure 5.1 – Another result of our model, illustrating four steps.

Figures 5.2 and 5.3 show visually adequate results, however it is possible to observe that the hands are not touching the body or in the pockets, as they are in the pictures. The reconstruction of the virtual human is not performed in real time, and it is impacted negatively as the number of vertexes in the models increases. In average, the whole process to reconstruct a VH based on techniques described in Section 3.2 takes around a couple minutes.

[1] http://usa.autodesk.com/
[2] http://unity3d.com/

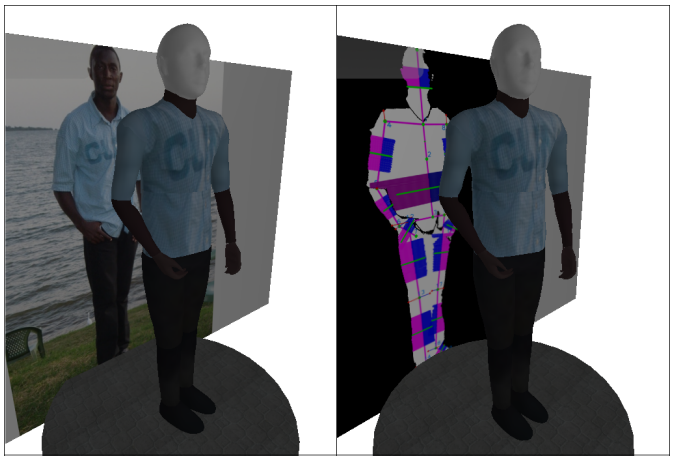


Figure 5.2 – Illustration of the original image and the generated VH (left) and the processed silhouette and the VH (right).

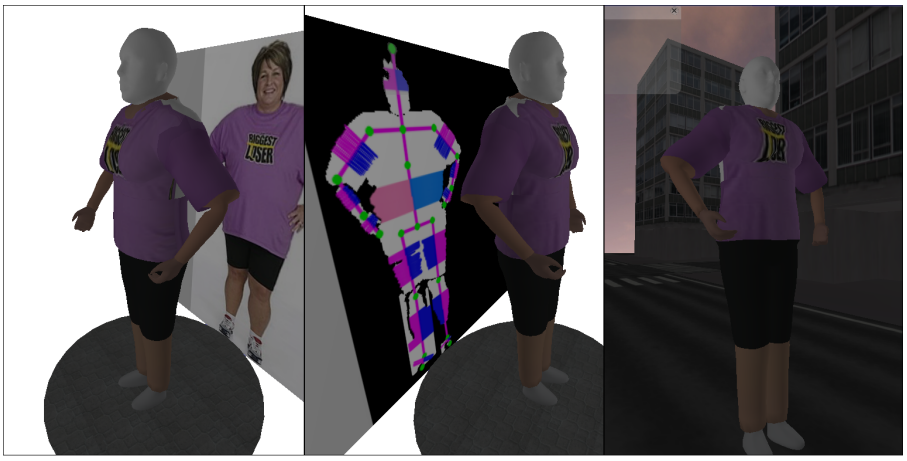


Figure 5.3 – Illustration of the original image and the generated VH (left), another point of view and the processed silhouette (center) and the VH in a virtual word (right).

Figure 5.4 shows a reconstructed virtual human (left) and the usage with other virtual humans during a visualization (right). The reconstructed virtual humans are fully capable of animations, the whole visualization process took less than three minutes to be set up.

5.2 Crowd Visualization

Figure 5.5 illustrates a single character performing different actions in different rooms from a house. On the left shows the virtual human executing a sleeping animation, while on the right, a grasping animation using the data provided by the *screenplay* XML file combined with the positions provided by the *simulation* XML file.

Figures 5.6 and 5.7 illustrate the results of the *Behavior Analysis* module, explained in Section 3.1. In both Figures we can observe the difference when the *Human Interactions* are

Figure 5.4 – Illustration of a reconstructed virtual human (left) and the usage with other virtual humans during a visualization (right).

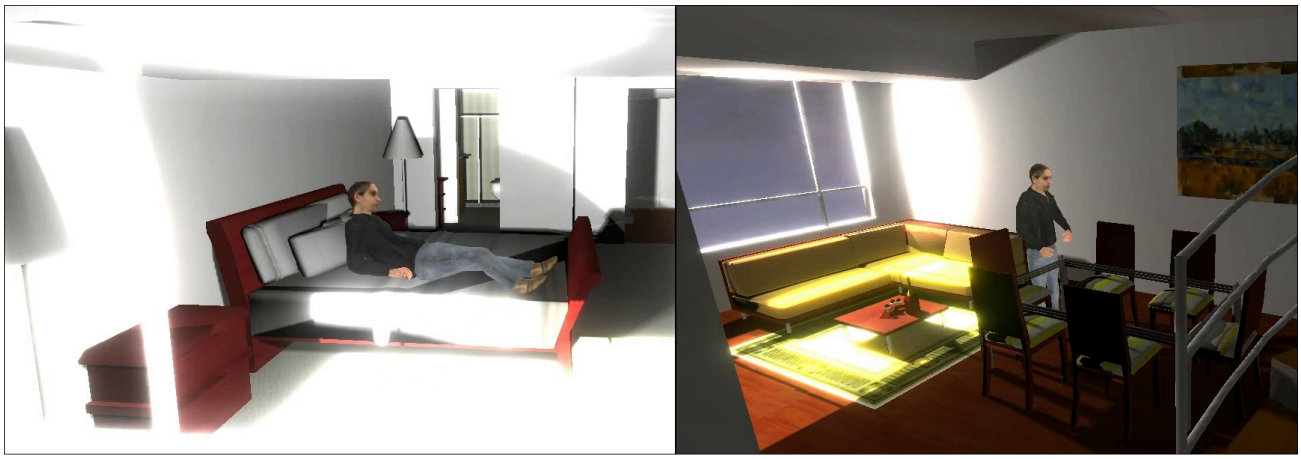


Figure 5.5 – Illustration of the virtual human executing a sleeping animation (left) and a grasping animation (right) using the data provided by the *screenplay* XML file combined with the positions provided by the *simulation* XML file.

disabled (left) and enabled (right). An interesting aspect is regarding the animations played by the VHs. The VHs used in the visualization, including the environmental agents, can play any animation accordingly to the user's desire.

Unity 3D can be considered a powerful real time graphic engine attached to an editor fully capable for game development content[3]. In order to compare *CrowdVis* with *Unity 3D* we had to detach our virtual humans from *Cal3D* and use Autodesk's FBX file format (only when rendering inside *Unity 3D*). Even though, *Unity 3D* has better frame rates results when comparing a large number of agents, as showed in Table 5.2. In our comparison we forced the usage of mesh geometry information when comparing up to 1000 agents. It is

[3] http://www.moddb.com/engines/unity/reviews/

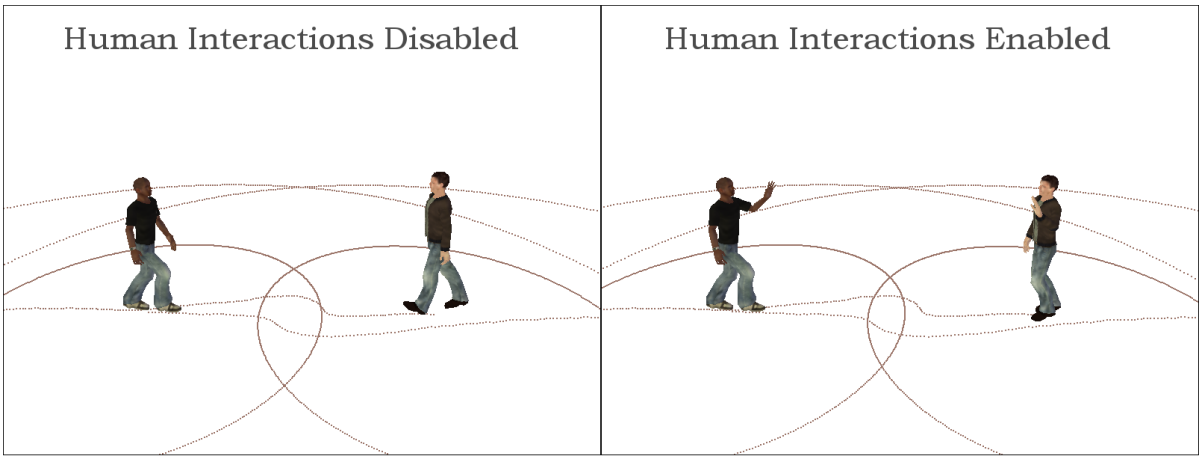


Figure 5.6 – Illustration of a *human interaction* detected by *CrowdVis Behavior Analysis* module when a pair of agents is passing by each other in opposite directions.

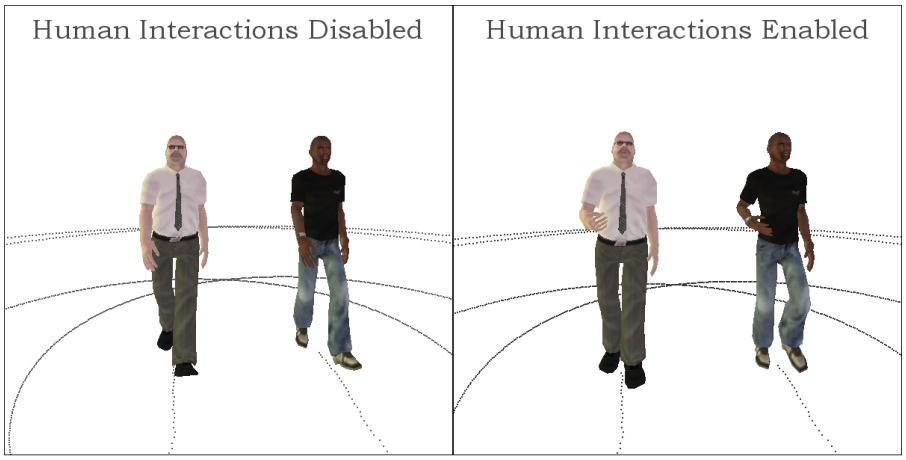


Figure 5.7 – Illustration of another *human interaction* detected by *CrowdVis Behavior Analysis* module when a pair of agents is moving together in the same direction.

also possible to observe that, for smalls and very large crowds, *CrowdVis* had better results when using our LOD techniques combined with animated impostors.

Table 5.1 – Comparison of the performance between *CrowdVis* and *Unity 3D* engine according to the number of agents and the number of polygons in the camera frustum.

Number of Agents	FPS CrowdVis (*Cal3D*)	FPS *Unity 3D* (FBX)
50	131	67
250	29	55
500	17	27
1000	10	13
4200	8	–

In addition to performance (see Table 5.2), we can try to evaluate the quality of the rendering. Figure 5.8 illustrates two different renderings, both containing 1000 agents inside the camera's frustum. On the left is illustrated the scene rendered in *CrowdVis*, where VHs are rendered using LOD techniques (according to the camera distance as explained in Section 4.1). On the right it is the same scene programmed in *Unity 3D*, using their default rendering technique.

Figure 5.8 – Illustration for evaluating two different renderings, both containing 1000 agents inside the camera's frustum. On the left is illustrated the scene rendered in *CrowdVis*, where VHs are rendered using LOD techniques. On the right is the same scene programmed in *Unity 3D*, using their default rendering technique.

Table 5.2 – *CrowdVis* frame rate according the number of agents.

Number of Agents	Frames Per Second
100	650
1000	150
10000	80
50000	30
100000	20
500000	8

5.2.1 Crowd Simulators

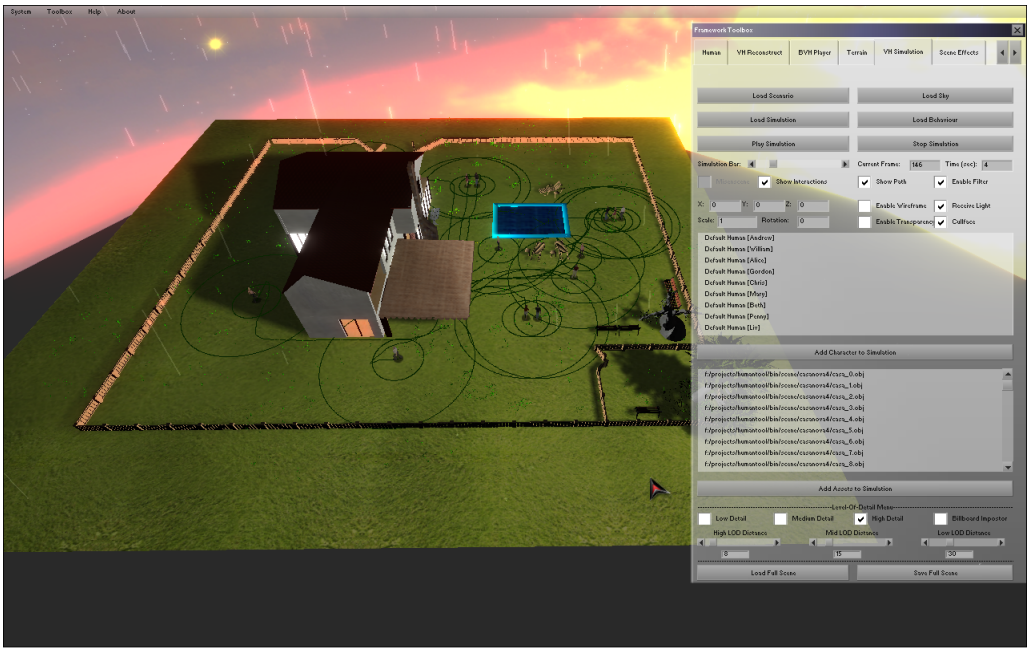
In order to validate *CrowdVis*, we produced crowd visualizations taking into account two different models of crowd simulations: *BioCrowds* and *CrowdSim*. Following is a short explanation of these models and illustrations of their visualizations.

Bicho proposes *BioCrowds* model [14] for crowd simulations. The author considers Runion's leaf venation patterns approach [38] in order to simulate the motion of the agents

through space. In Runion's model, a set of *auxins* are randomly[4] placed in a plane, which represents the leaf. The leaf's veins, when growing, compete for space by appropriating the available *auxins* (hormones). The same concept is adopted for the agents, where each agent can be in hold of a set of *auxins* that are within a radius, these auxins cannot be collected by other agents until the one in hold release them. This creates a collision avoidance model itself and after incorporating a small set rules, the result is an agent's motion very similar to human crowds. Figure 5.9 illustrates the visualization of two simulation frames created using *BioCrowds* model for for a group of agents.

CrowdSim model was created in 2011 by Marcelo Paravisi and Rafael Rodrigues, their approach for crowd simulation performs collision avoidance by dividing the world into cells. For the agents that are inside these cells, it is created collision buffers which are verified during the next simulation second. The collision avoidance is only tested using these buffers, and after one second their are cleared and buffered again, with the new set of agents currently inside each cell. Figure 5.10 illustrates the visualization of two simulation frames created using *CrowdSim* model for 6000 agents.

[4] Auxins are placed using the dart throwing technique.



(a)

(b)

Figure 5.9 – Illustration of the visualization of two simulation frames created using *BioCrowds* model for a group of agents.

(a)

(b)

Figure 5.10 – Illustration of the visualization of two simulation frames created using *CrowdSim* model for 6000 agents.

6. FINAL CONSIDERATIONS

We presented *CrowdVis*, a model for real time crowd visualization. With our approach, the user is able to visualize previously generated crowd simulations without any programming intervention. The main contributions of this work are to provide automatic detection of possible humans interactions, realistic motion of the virtual humans and also automatic optimizations in rendering techniques. Besides that, the possibility of fast VH creation without requiring training databases, but only deforming templates.

The most part of models for crowd simulations consider the agents as simple points or spheres. Such factors are concerned with agents behaviors in order to provide interactions between two agents during their motion, and also the coherent agents motion when the simulations are visualized with virtual humans (biped agents). Also, several special effects are presented in *CrowdVis* considering audio and video possibilities.

Currently *CrowdVis* model is only able to reconstruct average women and men. We intend in a future work to include more anthropometric data for children, old people and athletics, providing a larger set of VHs to be used during the visualizations.

The usage of Biovision Hierarchy (BVH) animations incorporated with the virtual humans could improve the *CrowdVis* animation system. The BVH pattern provides a large number of animations, mostly collected by motion capture. These animations can be very useful for the users don't be dependent of animators and also be able to use a different set of *regular* animations than the one currently present. In order to fully provide BVH support to virtual humans we would need a *bone map* from the user associating each bone with the bone name existing in the BVH, and then convert the rotations to match with the BVH patterns. On the other hand, in case the users prefer to use only the default or the reconstructed VHs makes it unnecessary the *bone mapping* process.

CrowdVis should be able to reproduce specific motion behaviors for the VHs during visualizations. These behaviors should include automatic detection of obstacles and objects, making the VHs avoid and interact with them according to the context of the scene. For example, the VHs could automatically twist their shoulders in order to pass through a tight place, providing better interaction with the environment. We also address the possibility for the VHs to interact with smart objects [27, 17]. The goal of these interactions is to increase the overall realism for the visualizations. One example could be the usage of smart object for store fronts, where the VHs nearby it could stop and admire the products for sale. Another example could make the VHs interact with vending machines, ATM, among others. The major challenge in this approach is making the VHs motion realistic without compromising the simulation.

Moreover, the set of special effects could be increased. For that, *CrowdVis* could provide

the tools for enabling users to create their own custom particle effects. Another important step could be made towards rendering optimizations. *CrowdVis* bottleneck is revealed when rendering a large scale of VHs, since the animation system is currently attached to the CPU. The usage of hardware skinning, threading and the improvement of *Cal3D* library would severely increase the frame rates, bearing a higher quantity of embodied VHs during the visualization.

During the master program experience, the author of this dissertation was involved in several researches in computer graphics, resulting in five published papers (see Appendix B). The two major researches were regarding the reconstruction of VHs and how to provide fun interactivity in surveys. The research in reconstructing VHs was the main focus during first year and it was funded by *Hewlett-Packard Brazil R&D*. The main goal was to reconstruct a VH according to the human in the picture and be able to edit the pose in the image. The research regarding fun and interactive surveys was funded by *General Motors R&D*. This project provided the opportunity of working with cutting edge technology and visiting the Carnegie Mellon University facilities for six month period (July to December, 2011).

Also, during the last year, besides improving the VHs reconstruction model, we studied a set of techniques regarding real time rendering and we developed models for detecting human interactions and improve VHs motion. Combining these, we were able to create *CrowdVis*, a framework capable of providing good visualizations for different crowd simulation models and also useful for generating visual results in our latest researches.

REFERENCES

[1] ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. Scape: shape completion and animation of people. *ACM Trans. Graph 24* (2005), 408–416.

[2] AUBEL, A., BOULIC, R., AND THALMANN, D. Animated impostors for real-time display of numerous virtual humans. In *Virtual Worlds* (1998), J.-C. Heudin, Ed., vol. 1434 of *Lecture Notes in Computer Science*, Springer, pp. 14–28.

[3] BARLOW, H. B., AND MOLLON, J. D. *The Senses. Cambridge Texts in Physiological Sciences*. Cambridge University Texts, Cambridge, United Kingdom, 1982.

[4] BECHEIRAZ, P., AND THALMANN, D. A model of nonverbal communication and interpersonal relationship between virtual actors. In *Proceedings of the Computer Animation* (Washington, DC, USA, 1996), CA '96, IEEE Computer Society, pp. 58–67.

[5] BENNO M. NIGG, W. H. *Biomechanics of the musculo-skeletal system*. John Wiley & Sons, 2003.

[6] BLINN, J. F. Simulation of wrinkled surfaces. *SIGGRAPH Comput. Graph. 12*, 3 (Aug. 1978), 286–292.

[7] BRAUN, H., CASSOL, V. J., HOCEVAR, R., MARSON, F. P., AND MUSSE, S. R. Crowdvis: A framework for real time crowd visualization. In *Proceedings of the 28th ACM Symposium On Applied Computing - SAC 2013* (Coimbra, Portugal, 2013), ACM, p. to appear.

[8] BRAUN, H., JUNIOR, H., JACQUES JUNIOR, J. C. S., DIHL, L. L., BRAUN, A., MUSSE, S. R., JUNG, C. R., THIELO, M. R., AND KESHET, R. Making them alive. In *Proceedings of the 2011 Brazilian Symposium on Games and Digital Entertainment* (Washington, DC, USA, 2011), SBGAMES '11, IEEE Computer Society, pp. 165–170.

[9] CASSOL, V. J., MARSON, F. P., VENDRAMINI, M., PARAVISI, M., BICHO, A. L., JUNG, C. R., AND MUSSE, S. R. Simulation of autonomous agents using terrain reasoning. In *Proc. the Twelfth IASTED International Conference on Computer Graphics and Imaging (CGIM 2011)* (Innsbruck, Austria, 2011), IASTED/ACTA Press.

[10] CAVALCANTE VIEIRA, R., AUGUSTO VIDA, C., AND CAVALCANTE NETO, J. Manipulação corporal de personagens virtuais por deformações de medidas antropométricas. *Symposium on Virtual and Augmented Reality*, 12 (2010), 102–111.

[11] CHI, D., COSTA, M., ZHAO, L., AND BADLER, N. The emote model for effort and shape. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 173–182.

[12] CLARK, J. H. Hierarchical geometric models for visible surface algorithms. *Commun. ACM 19*, 10 (Oct. 1976), 547–554.

[13] DE HERAS CIECHOMSKI, P. S. *Rendering massive real-time crowds*. PhD thesis, IC, Lausanne, 2006.

[14] DE LIMA BICHO, A. *Da modelagem de plantas à dinâmica de multidões: um modelo de animação comportamental bio-inspirado*. PhD thesis, Universidade Estadual de Campinas - UNICAMP, Campinas, 2009.

[15] DÉCORET, X., DURAND, F., SILLION, F. X., AND DORSEY, J. Billboard clouds for extreme model simplification. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 689–696.

[16] FRUIN, J. *Pedestrian Planning and Design*. Metropolitan Association of Urban Designers and Environmental Planners, 1971.

[17] GONÇALVES, L. M. G., KALLMANN, M., AND THALMANN, D. Defining behaviors for autonomous agents based on local perception and smart objects. *Computers & Graphics 26*, 6 (2002), 887–897.

[18] GUAN, P., WEISS, A., BALAN, A., AND BLACK, M. J. Estimating human shape and pose from a single image,. In *Int. Conf. on Computer Vision, ICCV* (February 2009), pp. 1381–1388.

[19] HALL, E. T. *The hidden dimension / Edward T. Hall*, [1st ed.] ed. Doubleday, Garden City, N.Y. :, 1966.

[20] HAMILL, J., AND O'SULLIVAN, C. Virtual dublin - a framework for real-time urban simulation. In *Journal of WSCG* (2003), pp. 221–225.

[21] HASLER, N., THORMÄHLEN, T., AND SEIDEL, H.-P. Multilinear pose and body shape estimation of dressed subjects from image sets. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)* (San Francisco, USA, June 2010), IEEE, pp. 1823–1830.

[22] HELBING, D., FARKAS, I., AND VICSEK, T. Simulating dynamical features of escape panic. *Nature 407* (2000), 487–490.

[23] HILTON, A., BERESFORD, D., GENTILS, T., SMITH, R., AND SUN, W. Virtual people: Capturing human models to populate virtual worlds. *Computer Animation 0* (1999), 174.

[24] HOPPE, H. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 99–108.

[25] JACQUES, J., DIHL, L., JUNG, C., THIELO, M., KESHET, R., AND MUSSE, S. Human upper body identification from images. In *ICIP* (Hong Kong, China, 2010), IEEE.

[26] JESCHKE, S., WIMMER, M., AND PURGATHOFER, W. Image-based representations for accelerated rendering of complex scenes. In *EUROGRAPHICS 2005 State of the Art Reports* (aug 2005), Y. Chrysanthou and M. Magnor, Eds., EUROGRAPHICS, The Eurographics Association and The Image Synthesis Group, pp. 1–20.

[27] KALLMANN, M., AND THALMANN, D. Direct 3d interaction with smart objects. In *VRST* (1999), pp. 124–130.

[28] KAVAN, L., DOBBYN, S., COLLINS, S., ŽÁRA, J., AND O'SULLIVAN, C. Polypostors: 2d polygonal impostors for 3d crowds. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2008), I3D '08, ACM, pp. 149–155.

[29] KRISHNAMURTHY, V., AND LEVOY, M. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 313–324.

[30] LUNA, F. *Introduction to 3D Game Programming with DirectX 9.0c: A Shader Approach*. Wordware Game and Graphics Library. Wordware Pub., 2006.

[31] MACIEL, P. W. C., AND SHIRLEY, P. Visual navigation of large environments using textured clusters. In *Proceedings of the 1995 symposium on Interactive 3D graphics* (New York, NY, USA, 1995), I3D '95, ACM, pp. 95–ff.

[32] MUSSE, S. R., AND THALMANN, D. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics 7* (April 2001), 152–164.

[33] NEFF, M., AND FIUME, E. Methods for exploring expressive stance. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), SCA '04, Eurographics Association, pp. 49–58.

[34] NEFF, M., AND FIUME, E. Aer: aesthetic exploration and refinement for expressive character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 161–170.

[35] PELECHANO, N., AND BADLER, N. I. Modeling crowd and trained leader behavior during building evacuation. *IEEE Comput. Graph. Appl. 26* (November 2006), 80–86.

[36] PETTRÉ, J., CIECHOMSKI, P. D. H., MAÏM, J., YERSIN, B., LAUMOND, J.-P., AND THALMANN, D. Real-time navigating crowds: scalable simulation and rendering: Research articles. *Comput. Animat. Virtual Worlds 17* (July 2006), 445–455.

[37] REEVES, W. T. Particle systems a technique for modeling a class of fuzzy objects. *ACM Trans. Graph. 2*, 2 (Apr. 1983), 91–108.

[38] RUNIONS, A., FUHRER, M., LANE, B., FEDERL, P., ROLLAND-LAGAN, A.-G., AND PRUSINKIEWICZ, P. Modeling and visualization of leaf venation patterns. *ACM Trans. Graph. 24*, 3 (July 2005), 702–711.

[39] SAMOVAR, L. S., AND PORTER, R. E. *Intercultural Communication: A Reader*, 6 ed. Wadsworth Publishing Co Inc, Belmont (CA):Wadsworth, 1982.

[40] SANNIER, G., AND THALMANN, N. A user-friendly texture-fitting methodology for virtual humans. 167–176.

[41] SCHAUFLER, G. *Per-Object Image Warping with Layered Impostors*, vol. 1. Springer-Verlag Wien New York, 1998, pp. 145–156.

[42] SCHELL, J. *The Art of Game Design: A book of lenses*. Morgan Kaufmann, 2008.

[43] SUN, L., AND QIN, W. Simulation of crowd behaviors based on event reaction. In *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on* (june 2011), vol. 2, pp. 163 –167.

[44] TAVARES DA SILVA, A. Geração automatica de população de personagens virtuais. Master's thesis, Unisinos, São Leopoldo, 2005.

[45] TAYLOR, C. J. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Comput. Vis. Image Underst. 80*, 3 (2000), 349–363.

[46] TECCHIA, F., LOSCOS, C., AND CHRYSANTHOU, Y. Visualizing crowds in real-time. *Computer Graphics Forum 21*, 4 (2002), 753–765.

[47] THALMANN, D. Frontiers of human-centred computing, online communities and virtual environments. Springer-Verlag, London, UK, UK, 2001, ch. The role of virtual humans in virtual environment technology and interfaces, pp. 27–38.

[48] TILLEY, A. R. *The measure of man and woman - Human factors in design*. John Wiley & Sons, inc, 2002.

[49] TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. Model reduction for real-time fluids. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 826–834.

[50] WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph. 22*, 4 (June 1988), 85–92.

[51] WILLIAMS, L. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), SIGGRAPH '78, ACM, pp. 270–274.

Appendix A. Behavior Analysis Module

1. begin of the process, the selected frame and selected VH are initialized with null;
2. get the next frame information from the memory data;
3. condition to check if there are more frames to update;
4. reset the VH counter to null;
5. reset the frame counter to null;
6. get the next VH's information from the memory data;
7. condition to check if there are more VHs to update;
8. verify if current VH is enabled to play *regular animations*;
9. play and update the *regular animations* for current VH and frame;
10. checks if current VH have for the current frame a *screenplay animation* to be played;
11. check if the *humans interactions* animations are enabled, these animations can be enabled or disabled through the prototype interface;
12. for all the computed *humans interactions* animations, check if there is any animation to end or start at current frame;
13. blend the current *regular animation* with the *humans interactions* animation to be played;
14. check if *regular animations* should be enabled for the current VH;
15. enables *regular animations* for the current VH;
16. disable *regular animations* for the current VH;
17. play the *screenplay animation* for that VH.

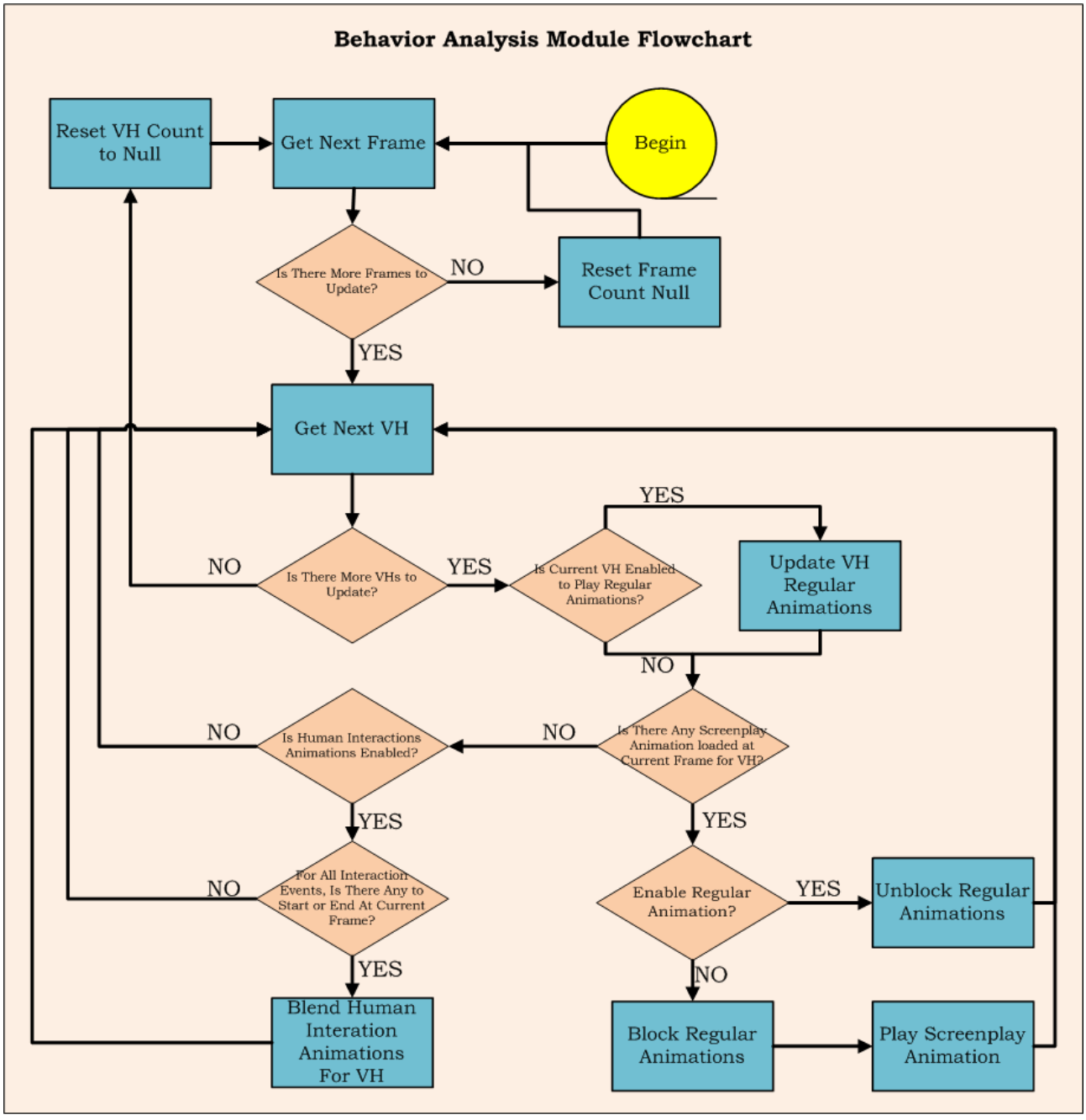


Figure A.1 – Behavior Analysis module flowchart illustrating the decision flow and priorities between the animation types present in *CrowdVis*.

Appendix B. Publications

BRAUN, H., HOCEVAR, R., QUEIROZ, R. B., COHEN, M., MOREIRA, J. L., JACQUES JÚNIOR, J. C., BRAUN, A., MUSSE, S. R., AND SAMADANI, R. *VHCVE: A collaborative virtual environment including facial animation and computer vision*. In Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment (Washington, DC, USA, 2009), SBGAMES '09, IEEE Computer Society, pp. 207-213.

DE LIMA, D. S., BRAUN, H., AND MUSSE, S. R. *A model for real time ocean breaking waves animation*. In Proceedings of the 2010 Brazilian Symposium on Games and Digital Entertainment (Washington, DC, USA, 2010), SBGAMES '10, IEEE Computer Society, pp. 19-24.

COUTO, J. M. C., MARCELINO, T., BRAUN, H., DE LIMA, D. S., AND MUSSE, S. R. *Generation of cartoon 2d cracks based on leaf venation patterns*. In Proceedings of the 2010 Brazilian Symposium on Games and Digital Entertainment (Washington, DC, USA, 2010), SBGAMES '10, IEEE Computer Society, pp. 165-170.

BRAUN, H., JUNIOR, H., JACQUES JUNIOR, J. C. S., DIHL, L. L., BRAUN, A., MUSSE, S. R., JUNG, C. R., THIELO, M. R., AND KESHET, R. *Making them alive*. In Proceedings of the 2011 Brazilian Symposium on Games and Digital Entertainment (Washington, DC, USA, 2011), SBGAMES '11, IEEE Computer Society.

HOCEVAR, R., MARSON, F., BRAUN, H., CASSOL, V., BIDARRA, R., AND MUSSE, S. R. *From their environment to their behavior: A procedural approach to model groups of virtual agents*. In IVA '12: Proceedings of the 12th International Conference on Intelligent Virtual Agents (2012).

BRAUN, H., CASSOL, V. J., HOCEVAR, R., MARSON, F. P., AND MUSSE, S. R. *Crowdvis: A framework for real time crowd visualization*. In Proceedings of the 28th ACM Symposium On Applied Computing - SAC 2013 (Coimbra, Portugal, 2013), ACM, p. to appear.