CrowdVis: A Framework For Real Time Crowd Visualization

ABSTRACT

Crowd visualization is present mostly in digital games and computer animated movies, but 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 softwares and tools. This paper presents a framework for real time crowd visualization, that 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 Simulation and Visualization; Rendering.

1. INTRODUCTION

Game engines, as *Unity 3D*[1] and *Unreal Engine*[2], are easily accessible to general users, providing tools and mechanisms for displaying 3D graphics and creating games. These and other engines rarely provide focus on real time crowd visualization, besides that such engines require a certain set of programming skills and also artists for creating coherently the scene assets and the Virtual Humans (VHs).

In order to provide a good visualization of crowds, usually a certain set of features must be presented. Firstly, it needs to be capable to work with simulations coming from different models or tools. Secondly, when visualizing crowds, we should consider factors as VHs motion, animation, knowledge about others agents in the simulation and also the interactions among them. Another important subject that must be considered is that the simulation and the visualization need to be conservative, i.e. the visualization needs to represent exactly what is going on the simulation.

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

SAC'13 March 18-22, 2013, Coimbra, Portugal.

In this work we present *CrowdVis*, a framework to be handled either by naive or experienced users. The goal is to provide tools and enable the users to visualize, in 3D environments, previously created crowd simulations. The main contribution of this paper is a framework able to easily visualize simulations of virtual humans generated in any crowd simulator, since the outputmust include only agents positions as a function of time. Specifically, *CrowdVis* can create embodied agents, with animations as well, and proposing visual behaviors for agents interactions. Moreover, several rendering features are included.

The paper is organized as follows: next, we present the related work, while in Section 3 we present a way to produce crowd simulations. Our framework is detailed in Section 4. Results are presented and discussed in Section 5 and after, some final considerations are discussed.

2. RELATED WORK

Different approaches can be observed in the literature regarding crowd visualization. Related work can consider a set of factors, as rendering, mesh manipulation, texturing, behavior analysis, VH's reconstruction and so on.

Games and crowd simulation models can generate interactions among virtual humans in order to create dynamic and realistic scenes. Algorithms to perform those interactions have been improved by researchers over the last few years. It is known that several solutions were already created for interaction at specific artificial intelligence's models or virtual environments, but solutions aiming to provide visual representation of interaction behaviors are not common. Thalmann and Becheiraz [2] worked in a nonverbal communication and interpersonal relationship model between VHs, consisting in reactions and social interactions in the presence of other agents. Besides interpersonal relationships, they also analyze the VHs walking, such as a sad walk or happy walk, these postures also have influence on the behavior of others, increasing the believability of the VHs in social interactions.

According to [3], only seeing at the psychological notion of gesture is insufficient to capture movement qualities need 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. The main difference in comparison with our approach is that we provide a module to generate visual behaviors, which can be displayed as body animation or any other visual representation. But mainly, in

our approach, the interaction behavior is generated without changing the simulation, treated as a posprocessing analysis, since it is used for already simulated situations. The main goal is to improve the realism and believability of the visualization and the techniques employed can also be used for other applications such as games or any other real time simulations.

Tecchia *et al.* [15] 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 optimizations techniques for fast rendering, which includes shadows and quality shading. Using what they called "*Pre Computed Impostors*" as an Image-Based Rendering (IBR) technique and the availability of large texture memory buffers, they maximized rendering speed.

Hamill and O'Sullivan [9] resume a number of methods available for culling such as portal, occlusion horizons, occlusion shadow volumes and the simple bounding box tests. The vast type of environment, such indoors and outdoors, used in the crowd simulation makes common the use of multiple techniques combined. The Virtual Dublin [9] uses culling and collision techniques combined with the trade of model details for texture details, reducing the building's polygons, but increasing the memory need.

The framework created by Pettre *et. al.* [14] achieved the real time visualization of crowd simulations combining Levels-of-Detail (LOD) techniques. Their LOD techniques were combined with billboards, which dramatically reduced the cost of animations updates and deformations in the models. Another work from [4] presents a rendering engine to visualize a crowd of VHs in real time. The VHs are sorted by their distance to the observer into render fidelities groups. The engine combines impostors and different rendering acceleration techniques, such as caching schema, Level-of-Detail, shader and state sorting.

Kavan *et al.* [12], propose an approach focusing on rendering animated characters for large crowds in real time. They propose a method that reduces the amount of memory necessary for rendering crowd impostors. Their method creates 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.

3. CROWD SIMULATION

This section presents a way to simulate crowds and generate data to be used as input in our framework. The model considered in this step, aiming to provide crowd simulation data, was proposed by [6] and was inspired in a biological algorithm, based on competition for space in a coherent growth of veins and branches. Since this original model presents free-of-collision motion and group formation technique, we use such method to provide collision avoidance in our method as well. This method reproduces groups and agents motion and their interactions into the environment. As in the original model, the environment is represented by a set of markers (dots) used to create a space discretization. Together with the markers, we create a grid of nodes in the space where the motion is allowed. The nodes will be used as reference to the A* path planning algorithm [10]. While agents move from their initial location to their target by using the nodes in the environment, at each time step, we verify the presence of other agents, obstacles and other groups, in order to avoid collision.

According to [11] we can represent the environment in a semantic way and provide motion of groups of agents across the environment. To this, we generate the main input parameters for the simulation:

1. Total time of simulation;
2. The number of agents. In this case, the simulation process is responsible for creating and killing the agents (e.g. at the beginning and ending of a party) in order to attain the expected density of agents at a specific period of the simulation (e.g. low, medium or high).
3. Groups distribution presented in a certain population, i.e. how many agents are not grouped (individuals) or grouped in groups of two or three agents; and iv) the distribution of interest, entry and spawn locations (spaces or objects in the semantic environment) where agents should be created or go to.

It is also possible to simulate events, i.e. situations that modify the state of the environment, which reflect directly in the behavior of the agents in the affected area. When the event occurs, the agents immediately start to move to the defined safe points. We define as safe points regions that are comfortable/safe for the agents to be in. For example, if an event of rain starts, the agents should go to a protected place where they feel comfortable, like under an awning (a safe point).

At the end of this process, we are able to provide the *simulation XML file* with information meaning the agents trajectory across the time. The trajectory is composed by the agents position for each frame of simulation. The *simulation XML* is used together other inputs information in *CrowdVis*. It is important to observe that this is just a simple way to provide *CrowdVis* inputs. Different models can be used in order to provide the specified inputs.

4. CROWDVIS: A CROWD VISUALIZATION FRAMEWORK

The focus of our work is to provide a framework for crowd visualization that enables any user to load and play previously created simulations without any programming, modeling and animating knowledge. Visualizing such simulations requires a set of data, i.e. virtual humans, terrains, virtual environments. Our framework is responsible for providing a set of functionalities which are:

- loading and reconstructing VHs;
- loading and manipulating scenarios and assets;
- loading and manipulating the simulation playback;
- controlling of VHs behaviors; and
- managing rendering features, special effects and scene illumination.

An overview of *CrowdVis* is illustrated in Figure 1, where it is possible to observe the three main modules: *Reconstruction of Virtual Humans*, *Behavior Analysis* and *Audio-Visual Module* .

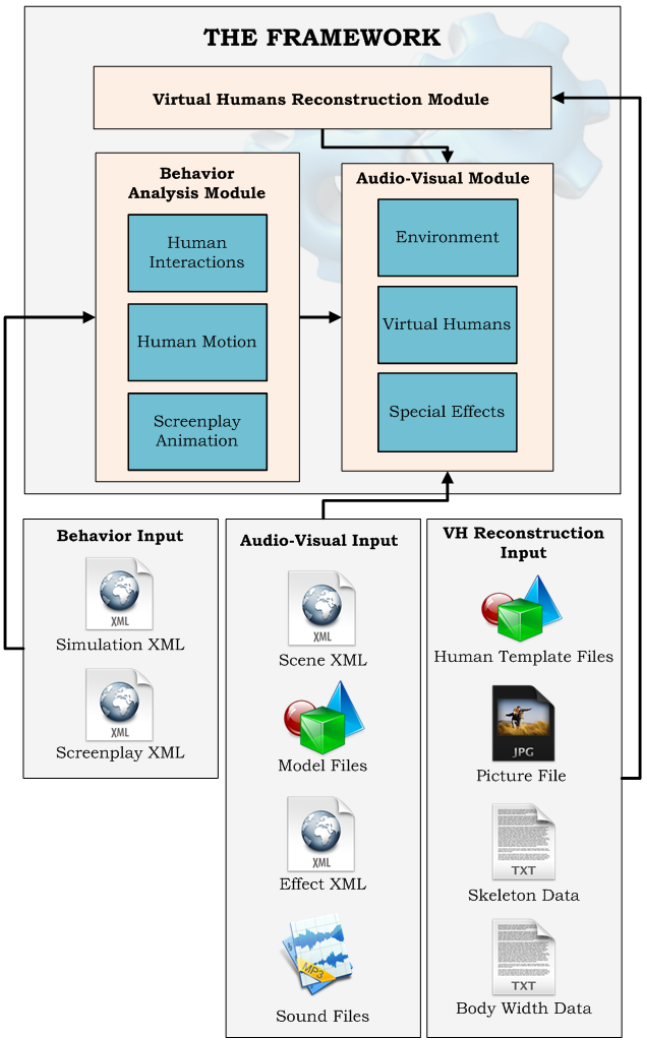

Figure 1: Overview of *CrowdVis*. It is possible to observe the three main modules (Virtual Human Reconstruction, Behavior Analysis and Audio-Visual) and their respective inputs.

4.1 Virtual Humans Reconstruction

Our model is able to reconstruct virtual humans from a single image. VHs semi-automatic reconstruction process can provide a fast way to create VHs models, with few or any designer intervention. According to this, in our model, we are able to populate virtual worlds using a set of different reconstructed VHs aiming to improve the agents variability during the crowd visualization. We adopted the work from [1], that pipeline phases are illustrated in Figure 2, where the user can quickly create different VHs from different single images. Such pipeline, considering the input image, is composed by four steps: *3D Pose Identification*, *Human Segmentation*, *Silhouette Processing* and *VH Reconstruction*.

4.2 Behavioral Analysis Module

The *behavioral analysis module* was developed with main goal to read the simulation data (set of 3D points over the time) and after an analysis process, to improve the results of

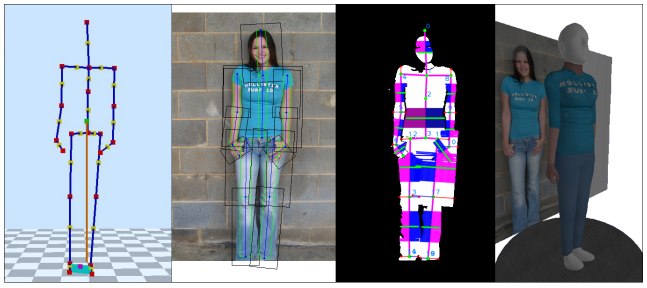

Figure 2: The VHs reconstruction pipeline is composed by four steps: *3D Pose Identification*, *Human Segmentation*, *Silhouette Processing* and *VH Reconstruction*.

crowd visualization, detecting possible *Human Interactions* among the VHs. These interactions are detected according to the virtual humans distances, orientations over time intervals, as detailed next. The inputs of this module are basically two files: the *Simulation XML* and the *Screenplay XML*. The first input describes the motion of the virtual humans and the second contains specific animations that should be played for each virtual human at a specific simulation frame.

Studying and observing groups of humans, the anthropologist Edward Hall [8] proposed the *proxemics* concept, which describes that the distance between people reveals their relationship. Hall divided the distances into four possible cases: *public* (7.6 m.), *social* (3.6 m.), *personal* (1.2m) and *intimate* (0.46 m.). Knowing these distances between pairs of agents, we can play different types of animations in order to provide more believability to the whole scene, i.e. a gesture of hello or even to make a couple give hands to each other according their intimacy distance.

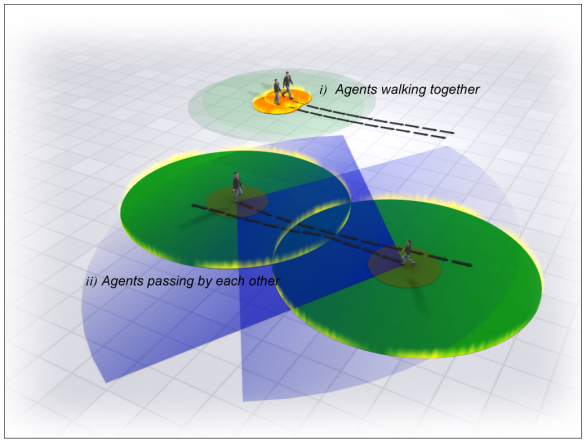

Figure 3: Two types of *Humans Interactions* detected by *CrowdVis*: the agents walking together (top) and agents passing by each other in opposite directions (bottom).

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 Hall distances, agent's *Field-Of-View* (FOV), interaction duration and speed difference. Figure 3 illustrates two types of *Humans Interactions* detected by the Framework: 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 compute distances between agents and verify if they are inside a circular region, as Figure 3, according Hall's *public distance* (six meters). In the case a pair of agents are inside each other circular region, we verify their orientations and FOV to know if they are going in opposite directions and are able to see each other. If the vision angle calculated for both agents is inside a specific threshold[3], we consider that the agents are seeing each other. The final step checks how long this process takes, 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 last lesser 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 it 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 to the same direction, they may be not "see" each other using our FOV method. By analyzing the average velocity, we can find out if the agents are really walking together or just passing by each other in the same direction. As the approach previously explained, we check inside a circular region, but for this particular case we opted for Hall's *social distance* (two meters) and the difference of the average velocity must be inferior than $0.1m/s$, value empirically defined in our tests.

In both approaches we check if the interaction events last longer than *six* seconds, we justified this interval empirically as the minimal time for a pair of agents have any sort of interaction between them. Figure 4 illustrates an agent passing by other while moving in the same direction, this case is ignored by *CrowdVis* using the average speed and interaction event duration metrics.

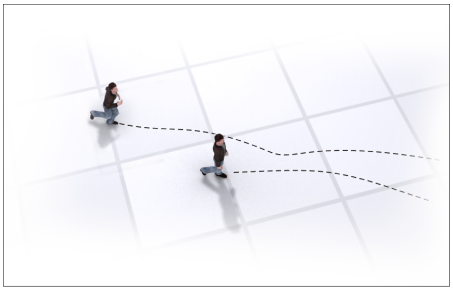

Figure 4: Illustration of a case where agents do not interact with each other.

It is important to highlight that the *Human Interactions* do not change any aspect of the simulation. The different visual representations performed by *CrowdVis* only increase the details of the scene and represent common real-life situations. All the animations are performed only in the upper

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

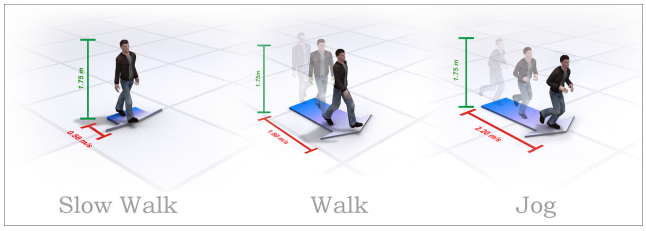

Figure 5: Illustration of animation used for each case.

bones of the virtual humans, this approach allows the animation blending with the *Regular Animations*, such as *walk* and *idle animation*.

Our Framework is responsible for determining which animation must be played for each agent, given the agent's displacement within the next second of simulation. If the displacement is smaller than $0.1m$, *CrowdVis* is responsible for blending the current animation to one of the available *idle animations*, on the other hand if it is greater than (or equal) $0.1m$, then there are three animations that can be played: *slow walk*, *normal walk* or *jog animation* (Fig. 5). Fruin [7] described intervals for average adult walking velocities. These intervals are used to decide which animation shall be played, as listed bellow:

- *idle animation* [$0.0m/s$ and $0.09m/s$];
- *slow walk animation* [$0.1m/s$ and $0.5m/s$];
- *walk animation* [$0.6m/s$ and $1.3m/s$]; and
- *jog animation* [$1.4m/s$ and above].

The next step is to use the simulation information regarding the agent's position for calculating where the agent should face (be oriented) at the end of each frame. While *CrowdVis* aims to deal with very few and simple information coming from crowd simulators (only the positions as a function of time), information not included. For instance, animations and orientations are not included in the *Simulation XML file*. Moreover, the fact that agents are represented by points can contribute to existing visualization artifacts during the simulation. For instance, if we decide to calculate a directional vector between the current frame and the next one, in order to define orientations, the result is not smooth. To achieve a better orientation for the agents, we use time coherence. As the velocity computation analyzes the next second of the simulation, we calculate the resultant vector that indicates the destination that each agent desires to face after one second. We use this data to smooth the orientation calculation for each frame.

Even if *CrowdVis* is in charge of determining a set of *Regular Animations* to be played automatically, the user must be able to play specific animations, which we called *Screenplay Animations*, at any time during the simulation, for example: *Agent [39] Play [Jump] at Frame [348]*. To allow users to perform this, a XML script can be loaded containing only a few set of required data: *agent id*, *name of the animation* and *start frame*. When a *Screenplay Animation* is loaded, it is played repeatedly in loop, so in case the user do not desire to be in control of the agent's animation anymore, he/she

just need to provide the *animation name* as *clear* with the desired *frame* to be turned off.

The Framework must not interfere in this type of animations, in other words, the *Regular Animations* and the *Human Interactions* animations are only calculated if there are not *Screenplay Animations* being played for the current agent.

A particular case can happen when the user loads a *Screenplay Animation* for a virtual human which is evolved in a *Human Interaction*. When it happens, the Framework prioritizes the *Screenplay Animation* over the *Human Interaction* animation. It is important to reiterate that all the processed and stored interaction behaviors can be turned off at any moment during the visualization using the prototype Interface.

We also provide the possibility of loading *"Environmental Agents"*, these virtual humans are not part of the simulation and can be loaded only for enriching the scene visualization. For loading them the user just fill a set of parameters located in our prototype interface corresponding to the agent's position, scale, rotation and animation. (Figure 14 illustrates environmental agents).

4.3 Audio-Visual Module

This module is responsible for several tasks in order to improve the visual quality of *CrowdVIs* results. In order to achieve good graphical results, we included a few set of techniques related to real time rendering, usually employed in crowds visualization and 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 [15]. To provide more realism we included two types of shadowing technique for the agents. The user may choose one of them or neither, removing all the shadows calculations and optimizing 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. The other shadowing technique is much more expensive, but the visual results improves considerably, we implemented our second shadowing technique based on Shadow Map.

Another method for obtaining greater visual results was first introduced by Gregory Ward [16], nowadays the technique is commonly known as *High Dynamic Range* (HDR) rendering, it consists of creating a greater range of luminance between the lightest and darkest areas for each frame. In *CrowdVis*, we use the post process technique implemented in the GPU, enabling it to be executed at real time frame rates.

Figure 6 illustrates the particle systems implemented in our framework for rain (left) and fire (right).

Music and sound effects during visualizations can increase the immersion feeling of users. It is important to decide which sound to play and when, so, we propose an *Effect XML* file containing not only the sound file paths to be played, but also the visual effects as rain and fire.

CrowdVis also provides the tools enabling the user to have control of the scene lights, sky and objects. An amount of 8 lights is restricted due 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

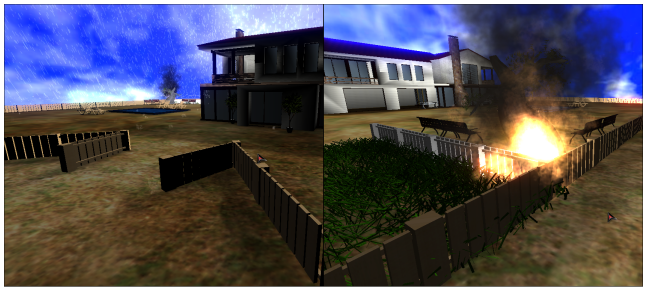

Figure 6: Illustration of the particle systems implemented in *CrowdVis* for rain (left) and fire (right).

be placed as a sky dome[4]. We also created an anaglyph rendering for 3D stereoscopic effect that can be enabled and disabled.

As explained before (see Section 4.2), the 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 to work in a few optimization techniques, such as *Level-Of-Detail* [5], *Impostors* and *Normal Maps*. Figure 7 illustrate one of our virtual humans and the three Levels-Of-Detail, low quality (left) medium quality (center) and high quality (right), with the respective number of triangle and polygons.

Figure 7: Illustration of one virtual human and the three Levels-Of-Detail, low quality (left) medium quality (center) and high quality (right), with the respective number of triangle and polygons.

In *CrowdVis*, we combined the *billboard impostors* [15] technique with the agent's geometry LOD, which means that distant agents in our visualization are replaced by view-dependent impostors, same approach proposed by Maciel [13]. Each impostor agent contains four textures, each one with 128x128 resolution. These images are previously processed and are stored in the Hard Drive (HD). Figure 8 illustrates our set of impostors texture for a single agent.

[4]Unreachable geometry in form of a dome used for rendering mountains, skies and any other background scene.

Figure 8: Illustration of the four impostors textures stored in the HD. Each texture contains the resolution of 128x128 pixels.

5. RESULTS

In this section we present some results obtained with *CrowdVis*. The Framework was implemented in C++ using: i) *Irrlicht Engine* [5] and *OpenGL* [6] for fixed pipeline renderization; ii) *CAL3D* [7] for character animation; iii) *IrrKlang* [8], an audio library allowing the user to play sounds and musics using different audio file formats and iv) *Spark Particle Engine* [9] for visual effects.

Follow images illustrate some examples of the features provided with *CrowdVis* that can be easily applied in real time when visualizing a crowd. Figure 9 illustrates the results of the *Behavior Analysis module*. We can observe the difference when the *Human Interactions* are disabled (left) and enabled (right).

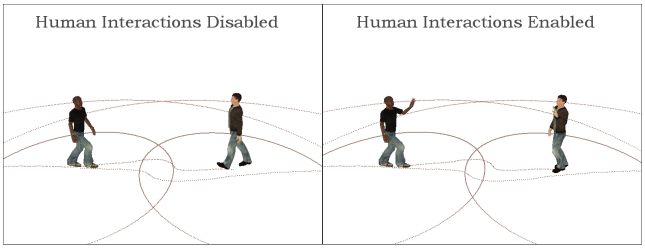

Figure 9: Illustration of a Human Interaction detected by our Behavior Analysis module when a pair of agents is passing by each other in opposite directions. On the left it is disabled and on the right it is enabled.

Considering scene visualization improvements, we work with shadows, illumination and other effects including rain and fire for particular situations. The shadowing techniques implemented are illustrated in Figure 11, and Figure 10 illustrates the usage of the HDR effect in different scenes (on the left the scene with the HDR rendering and on the right the scene with the fixed pipeline rendering).

Figure 12 illustrates some of special effects that can be applied when using our *CrowdVis*.

An important feature considered in our Framework is the use of LOD to improve the visualization performance. Table 1 presents the distances, the number of polygons and the

[5] `http://irrlicht.sourceforge.net/`
[6] `http://www.opengl.org/`
[7] `http://home.gna.org/cal3d/`
[8] `http://www.ambiera.com/irrklang/`
[9] `http://spark.developpez.com/`

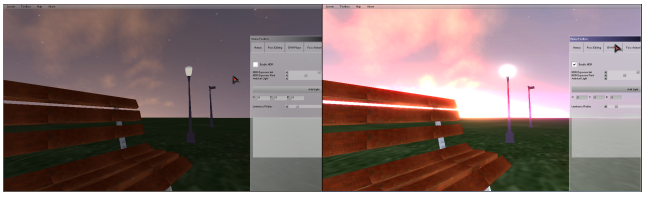

Figure 10: Scene illustrating the HDR effect (right) and the default rendering (left) with a dusky sky background.

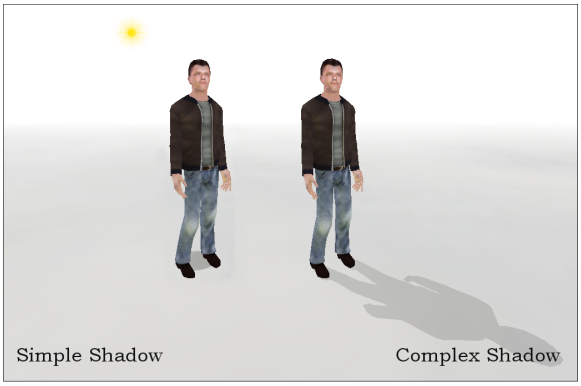

Figure 11: Illustration of both shadows techniques: simple shadowing (left) and complex shadowing (right).

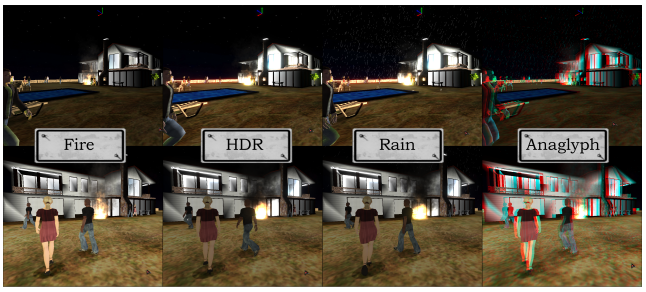

Figure 12: Illustration of some of the effects provided by our Framework.

LOD being utilized in our views for only one of the 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. Anyway, the user is able to adjust the distances as desired, using *CrowdVis* interface.

Once a virtual human 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. Figure 13 illustrates the exact same scene and camera view for 100 agents in two different rendering techniques without (left) and with (right) LODs.

Here we present some results when visualizing a complete scene. In the crowd simulator, we define a semantic environment that was populated with virtual agents. Such agents

Camera Distance	LOD	Polygons	FPS
[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

Table 1: Comparison between the distances, the number of polygons and the LOD being utilized in our views for only one of our virtual humans in the scene.

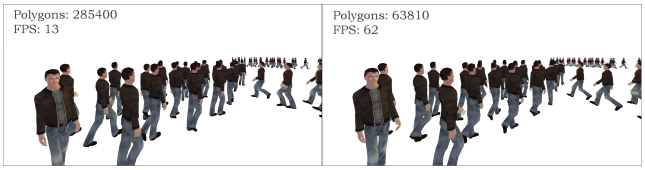

Figure 13: Illustration of the exact same scene and camera view for 100 agents in two different renderings without any LOD technique (left) and with LODs technique and impostors (right).

Figure 14: Two views of a scene in *CrowdVis*.

are considered as simple points in the simulation module and their 3D coordinates are the input to *CrowdVis*. Figure 14 illustrates results of our framework[10].

6. FINAL CONSIDERATIONS

This paper presented *CrowdVis*, a framework for real time crowd visualization. With our approach, the user is able to visualize previously generated crowd simulations without any programming intervention.

The most part of models for crowd simulations consider the agents as simple points or spheres. Considering this aspect, *CrowdVis* provides some features to be used to provide good quality of the visualization results. 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.

Results can show the facilities of our Framework and the applicability for crowd visualization. Moreover, acceptable visual results are achieved in real time.

[10]A video with *CrowdVis* results is available for download in `http://www.mediafire.com/?fnbf2vzg5yi1awt`

7. REFERENCES

[1] Omitted for blinding review.

[2] P. Becheiraz and D. Thalmann. A model of nonverbal communication and interpersonal relationship between virtual actors. In *Proceedings of the Computer Animation*, CA '96, pages 58–, Washington, DC, USA, 1996. IEEE Computer Society.

[3] D. Chi, M. Costa, L. Zhao, and N. Badler. The emote model for effort and shape. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 173–182, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[4] P. S. H. Ciechomski and D. Thalmann. Rendering massive real-time crowds. *ï£¡cole Polytechnique Fï£¡dï£¡rale de Lausanne*, 2006.

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

[6] A. de Lima Bicho. *Da modelagem de plantas ï£¡ dinï£¡mica de multidï£¡es: um modelo de animaï£¡ï£¡o comportamental bio-inspirado*. PhD thesis, Universidade Estadual de Campinas - UNICAMP, Campinas, 2009.

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

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

[9] J. Hamill and C. O'Sullivan. Virtual dublin - a framework for real-time urban simulation. In *Journal of WSCG*, pages 221–225, 2003.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *SIGART Bull.*, pages 28–29, 1972.

[11] R. Hocevar, F. Marson, V. Cassol, H. Braun, R. Bidarra, and S. R. Musse. From their environment to their behavior: A procedural approach to model groups of virtual agents. In Y. Nakano, M. Neff, A. Paiva, and M. Walker, editors, *Intelligent Virtual Agents*, volume 7502 of *Lecture Notes in Computer Science*, pages 370–376. Springer, 2012.

[12] L. Kavan, S. Dobbyn, S. Collins, J. Žára, and C. O'Sullivan. Polypostors: 2d polygonal impostors for 3d crowds. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, I3D '08, pages 149–155, New York, NY, USA, 2008. ACM.

[13] P. W. C. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, I3D '95, pages 95–ff., New York, NY, USA, 1995. ACM.

[14] J. Pettré, P. d. H. Ciechomski, J. Maïm, B. Yersin, J.-P. Laumond, and D. Thalmann. Real-time navigating crowds: scalable simulation and rendering: Research articles. *Comput. Animat. Virtual Worlds*, 17:445–455, July 2006.

[15] F. Tecchia, C. Loscos, and Y. Chrysanthou. Visualizing crowds in real-time. *Computer Graphics*

Forum, 21(4):753–765, 2002.

[16] G. J. Ward, F. M. Rubinstein, and R. D. Clear. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.*, 22(4):85–92, June 1988.