CEIG'09, San Sebastián, Sept. 9-11 (2009)
C. Andújar and J. Lluch (Editors)

A Framework for Rendering, Simulation and Animation of Crowds

N. Pelechano, B. Spanlang, A. Beacco
Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract

Real-time crowd simulation for virtual environment applications requires not only navigation and locomotion in large environments while avoiding obstacles and agents, but also rendering high quality 3D fully articulated figures to enhance realism. In this paper, we present a framework for real-time simulation of crowds. The framework is composed of a Hardware Accelerated Character Animation Library (HALCA), a crowd simulation system that can handle large crowds with high densities (HiDAC), and an Animation Planning Mediator (APM) that bridges the gap between the global position of the agents given by HiDAC and the correct skeletal state so that each agent is rendered with natural locomotion in real-time.

The main goal of this framework is to allow high quality visualization and animation of several hundred realistic looking characters (about 5000 polygons each) navigating virtual environments on a single display PC, a HMD (Head Mounted Display), or a CAVE system. Results of several applications on a number of platforms are presented.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Animation; I.6.8 [Simulation and Modeling]: Types of Simulation – Animation.

1. Introduction

Natural looking crowd simulation is necessary for applications such as video games and training simulators. Natural looking animations become even more important in immersive applications where a real participant can interact in real-time with characters in a virtual environment. If our goal is to be able to interact with relatively large number of agents, it is necessary that those agents perform natural path planning and animations, making sure that there is consistency between what we would expect in the virtual world from our experiences in the real world.

In order to achieve visually appealing results it is necessary to integrate motion planners with motion synthesizers. Most crowd simulation systems focus on the path planning and local motion details of the agents while using a character animation library to render 3D fully articulated characters playing a small set of animation clips. The main problem that can be easily observed is that the characters have a very limited number of movements available and that constraints such as ground contact points are not respected.

Generating empathetic characters with realistic movements and responses to other virtual agents and obstacles in the environment can be very tedious work, since they usually require large number of animation clips to be able to exhibit a large variety of movements. Being able to generate synthesized motions from small sets of animation clips is thus of extreme relevance especially if we want to exhibit variety while following the requirements given by a crowd simulation module dealing with path planning and local motion.

In this paper we present a framework for real-time crowd simulation which integrates a crowd simulation module that determines the position, velocity and orientation of each agent at every frame with a motion synthesizer that will guarantee smooth and natural looking animation for every agent.

In order to integrate both modules we developed an Animation Planning Mediator which with the information

DOI: 10.2312/LocalChapterEvents/CEIG/CEIG09/001-010

Figure 1. *Crowd Simulation, Animation and Rendering.*

provided by the crowd simulation module selects the best parameters to feed the motion synthesizer at the same time that it feeds back to the crowd simulation module with the required updates to guarantee consistency.

Figure 1 shows an example of the type of crowd simulation achieved with the framework presented in this paper.

2. Related Work

Probably the most well known approach for crowd simulation is based on Reynolds' work where agent behavior is simulated through a local rules model [Rey87], Brogan and Hodgins [BH97] used particle systems and dynamics for modeling the motion of groups with significant physics in combination with a rule-based model.

Social forces have also been widely employed for crowd simulation [HFV00]. Other approaches are based on flows [Che04]. Hybrid approaches have been presented combining forces with rules based on psychological and geometriacal features of the environment to achieve a wider range of heterogeneous behaviors [PAB07]. Shao and Terzopoulos introduced cognitive models to simulate pedestrians navigating in a train station [ST05].

To achieve a real-time simulation of very large crowds, Treuille et al. [TCP06] used a dynamic potential field to integrate global navigation with moving obstacles and people. This approach is not agent based, which means that we cannot have individual goals for each pedestrian; instead, goals are common to all the crowd members.

Lerner et al. [LCL07] introduced a novel approach for rule-based simulation based on examples, in which tracking data from real crowds is used to create a database of examples that is subsequently used to drive the simulated agents' behavior.

Hierarchical schemes have been proposed to address scalability. In particular, Musse and Thalmann [MT01] provide crowds with different levels of autonomy for hierarchical crowd behaviors.

To navigate a complex environment, some semantically meaningful geometric representation of the environment is essential. Among the most popular techniques for crowd navigation are cell and portal graphs (CPGs) [LCC06][PB06], potential fields [Che04], and roadmaps [SGA*07].

Lau and Kuffner [LK06] introduced precomputed search trees for planning interactive goal-driven animation. The method consisted of a finite state machine composed of a small set of animation clips (jog forward, left, right, stop, crawl, and jump). Their algorithm creates a tree of all the possible animations run from a starting point and selects the path within the tree that takes them closer to the goal in the environment.

Through interpolation and concatenation of motion clips, new natural looking animations can be created [WP95]. Kovar et. al. introduced motion graphs to create new animations by following a path within the graph of motions clips where transitions between similar frames in those clips were available [KGP02]. Zhao and Safanova extended motion graphs to improve both connectivity and smooth transitions [ZS07]. Such techniques can achieve natural animations, but they require a large database of motion capture data to allow for interactive changes of walking speed and orientation. Furthermore, these algorithms require searches in complex graphs and thus are not appropriate for interactive control of crowds. In our framework we only need a small set of animations clips to synthesize new motions.

Since rendering large numbers of agents can become a bottleneck as the numbers of polygons increase, a solution can be to apply Level Of Detail for the characters depending on their distance to the camera [UHT04]. Another option to improve the performance of rendering is to use impostors. Aubel et al [ABT00] described their impostors as "a simple textured plane that rotates to continuously face the viewer" a snapshot of the virtual human is textured onto the plane. The snapshots are produced at run-time.

Pre-generated impostors with improved shading have also been used [TLC02]. Impostors can achieve rendering of large crowds (10th of thousands), but their main disadvantge appears when the viewer gets too close to an agent, since then the character will appear pixelized.

Dobbyn et. al. [DHC*05] introduced the first hybrid system that was presented by using impostors on top of a full, geometry-based human animation system, and switching between the two representations with minimal popping artifacts. In order to reduce the memory requirements of impostors, while keeping a high level rendering efficiency, 2D polygonal impostors have been used [KDC*08].

De Heras Ciechomski et al. [HUCT04] avoid computing the deformation of a character's mesh by storing precomputed deformed meshes for each key-frame of animation, and then sorting these meshes to take cache coherency into account.

Owing to the programmability and performance of recent graphics hardware [Ros06] different Skinning techniques have been introduced recently [KCZO07]. In Skinning the segments of a character's skeleton are associated with the mesh that describes the character by vertex weights. Skinning methods blend between skeletal segment transformations in order to create smooth mesh deformations in joint areas, for example between the upper and the lower arm of an avatar. These techniques are used by our animation library described in section 3.3 of this paper.

It is also important to note that variety in the animation clips has an impact on how we perceive crowd variety, as shown in recent work by McDonnell et al. [MLD*08].

With the framework presented in this paper we are able to control and visualise avatars on a high and on a low level. The high level control deals with navigation of agents in informed cell and portal graphs [PB06] where local motion is carried out through psychological, physiological and geometrical rules combined with physical forces [PAB07]. In order to animate the skeletal segments of the individuals in the crowd we are using HALCA [Spa09] that employs motion blending techniques to achieve a large variety of animations from a small set of recorded motions. HALCA performs the skeletal skin and clothing deformations of the avatars using GLSL shaders on the GPU in addition to texture and bump mapping techniques to visualise fairly realistic looking avatars in a very efficient way.

3. Crowds Framework

The framework presented in this paper performs a feedback loop where the Animation Planning Mediator acts as the communication channel between the crowd simulation module and the animation module. The outline of this framework is shown in Figure 2.

For each frame, the crowd simulation module calculates the position p, velocity **v**, and desired orientation of the torso θ. This information is then passed to the Animation Planning Mediator in order to select the parameters to be sent to the animation module. These parameters are the appropriate animation clip, the level of blending required and a description of skeletal segment adaptation as described in section 3.2.

The Animation Planning Mediator may need to slightly modify any of the parameters given by the crowd simulation module in order to guarantee that the animations rendered are smooth and continuous, since breaking those restrictions will result in unnatural crowd behavior.

It is thus essential that the crowd simulation model employed works in continuous space and allows for updates of the position, velocity or orientation of each agent at any given time in order to guarantee consistency with the requirements dictated by the animation module.

Our animation module, HALCA, contains a motion synthesizer which can provide a large variety of continuous motion from a small set of animations by allowing direct manipulation of the skeleton configuration during the animations.

It is very important that the information coming from the crowd simulation module is well processed and that noise is properly filtered out, otherwise it could result in unnatural character animation. At the same time, it is important that the feedback the crowd simulation module receives still allows for enough flexibility in the type of continuous movement available, otherwise we could have the animation module limiting freedom of movement and also affecting reaction times of the agents.

As in other crowd simulation architectures, characters can be created either by a modeling tool such as 3D studio max, Maya or Blender, or by employing 3D whole body scanners. For body scans during a pre-processing phase holes from the mesh are removed, the mesh has to be smoothed and skeletal bone weights have to be associated.

To create the library of animations, we decided to use hand created animations instead of motion captured data.

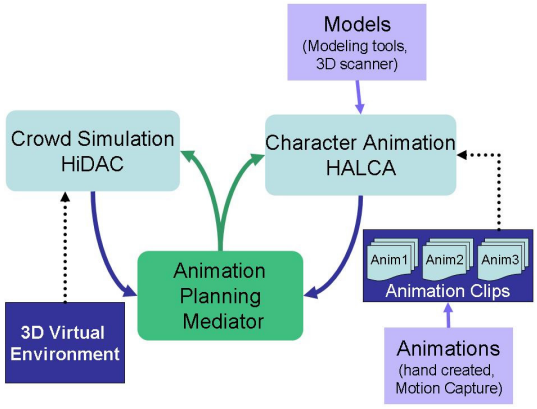

Figure 2: *Framework*

3.1 HiDAC – Crowd Simulation Module

For this work the crowd simulation model HiDAC has been used, although we could have used other real-time models such as social forces, rule-based or cellular automata. The key limitation imposed by our framework to guarantee smooth and natural looking animation is that the crowd simulation model utilized works in continuous space.

HiDAC provides as an output the position, velocity and orientation for each agent at each frame. The position indicates where the character should be rendered in the virtual environment, the velocity vector indicates the direction of movement and the orientation indicates the desired torso orientation of the fully articulated 3D figure for rendering purposes.

The torso orientation is given by the velocity vector but after applying a filter so that it will not be unnaturally affected by abrupt changes. This is done as we want the position of the character to be able to react quickly to changes in the environment such as moving agents and obstacles, but we need the torso to exhibit only smooth changes, as without this the result will be unnatural animations where the rendered characters appear to twist constantly. Through filtering we can simulate, for example, an agent that moves with a slight zig-zag effect, while the torso of the rendered character faces a constant forward direction

3.1.1 HiDAC model description

HiDAC addresses the problem of simulating high-density crowds of autonomous agents moving in a natural manner in dynamically changing virtual environments. To address the problem of realistically simulating local motion under different situations and agent personalities, it uses psychological, physiological and geometrical rules combined with physical forces. Since applying the same rules to all agents leads to homogeneous behavior, agents are given different psychological (e.g., impatience, panic, personality attributes) and physiological (e.g., locomotion, energy level) traits that trigger heterogeneous behaviors. Each agent is also endowed with perception and reacts to static and dynamic objects and other agents within the nearby space.

Agent behaviors are computed at two levels:

• High-level behavior: navigation, learning, communication between agents, and decision-making. [PB06]

• Low-level motion: perception and a set of reactive behaviors for collision avoidance, detection and response in order to move within a bounded space [PAB07].

Local agent motion is based on a combination of geometrical information and psychological rules with a forces model to enable a wide variety of behaviors resembling those of real people. HiDAC uses psychological attributes (panic, impatience) and geometrical rules (distance, areas of influence, relative angles) to eliminate unrealistic artifacts and to allow new behaviors:

- Preventing agents from appearing to vibrate
- Creating natural bi-directional flow rates
- Queuing and other organized behavior
- Pushing through a crowd
- Agents falling and becoming obstacles
- Propagating panic
- Exhibiting impatience
- Reacting in real time to changes in the environment

HiDAC will provide natural trajectories for each agent within a dynamic virtual environment even under high densities. The output of the system will be a position, velocity and torso orientation for each agent.

3.2 Animation Planning Mediator

To achieve realistic animation for large crowds from a small set of animation clips, we need to synthesize new motions.

The Animation Planning Mediator (APM) is used to find the best animation available while satisfying a set of constraints. On the one hand it needs to guarantee that the next animation frame will reflect as closely as possible the parameters given by the crowd simulation module (p, $\mathbf{v}$, θ), and on the other hand it needs to guarantee smooth and continuous animations. Therefore, the selection of the next best frame needs to take into account the current skeletal state, the available transitions in the graph of animations, the maximum rotation physically possible for the upper body of a human that will look natural, and if there are any contact points to respect between the limbs of the skeleton and the environment (for example, contact between a foot and the floor). Once the APM determines the best next frame and passes this information on to HALCA for animation and rendering, it will also have to provide feedback to the crowd simulation module in the cases where the parameters sent needed to be slightly modified to guarantee natural looking animations with the resources available (i.e. with the set of animation clips and transitions).

The APM will pre-process the set of animation clips available in order to extract relevant information that is necessary to classify the animations and store it so that it can be accessed in real time during the simulation. For each animation clip available, the APM will calculate its velocity $\mathbf{v}$ in m/s by computing the total distance traveled by the character through the animation clip divided by the total time of the animation clip, as well as the angle α between the orientation of the torso and the velocity vector $\mathbf{v}$.

Figure 3 shows the communication between HALCA and the APM during this pre-processing. In order to calculate the required information, the APM needs to obtain one after another the consecutive skeletal states for each animation. HALCA offers skeletal configurations through a

morphing value m in the interval [0,1], where 0 corresponds to the first key frame, and 1 corresponds to the last key frame. The APM thus sends the animation identifier, A_i, and the morphing value, m, and will receive back the corresponding skeletal state, sk(m).

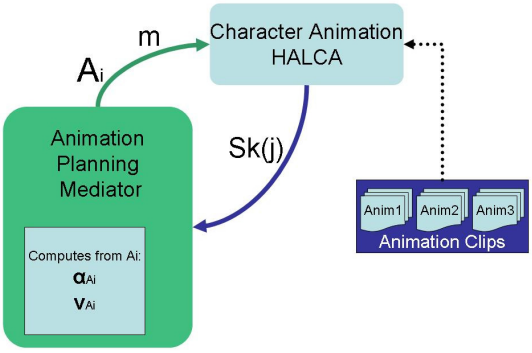

Figure 3: *Pre-processing information regarding the animations.*

3.2.1 Animation Clip Selection

Instead of achieving different walking speeds by having tens of different walking animations, the APM employs one animation clip that can be played at different speeds by modifying the number of frames per animation clip used in real time. When using Cal3D for animation we can have an animation of, for example, 60 frames per second which by varying the differential of time within frames rendered allows for a large variety of speeds for each given animation clip.

For this paper we use only four walking forward animations (very slow, slow, normal and fast, each with corresponding variations in step length) to allow an agent to move in a range of speeds from 0m/s to 1.57m/s. Each animation clip covers a subset of speeds going from the minimum speed when dt is very small, to the original speed of the animation clip when dt is 1. For a dt of value 0 the animation will appear to stop, since it will return the same frame twice.

An animation clip of walking forward can also be used to have the agent turn, as we can slightly rotate the figure within given contact constraints. This works very well for high walking velocities, but for slower animations it is worth having several turning animation clips and blending between them.

If we consider α to be the angle between the direction of movement **v** and the torso orientation θ we can classify animations based on α. For example an animation of walking sideways will have α = 90 degrees. In addition to 0 and 90 degrees, we use animations of α = 12 and α = 55 degrees in our examples.

Our ten animation library with four walking forward animations, two side-step animations and four walking on an angle animations is thus defined in Table 1. Each of these animations could also be run with a negative dt in order to obtain the opposite animation (i.e. walking forwards with a negative dt will result in walking backwards) This doubles the number of animation clips available in our library.

Anim. Id	v (m/s)	α	Anim descrip
0	0.3444	0	Walk Forwards Very Slow
1	0.5899	0	Walk Forwards Slow
2	1.0668	0	Walk Forwards Normal
3	1.5752	0	Walk Forwards Fast
4	0.3444	90	Walk Sideways Normal
5	0.0960	90	Walk Sideways Slow
6	0.8895	-12	Walk with small angle Left
7	0.8895	12	Walk with small angle Right
8	0.0960	-55	Walk with big angle Left
9	0.0960	55	Walk with big angle Right

Table 1. *Animation Clip Library*

During run time, the Animation Planning Mediator will select the best animation based on the current velocity, **v**, of the agent, and the angle, α, between the velocity and the torso orientation, θ, provided by the crowd simulation module.

3.2.2 Animation Speed

As explained in the previous section, each animation is defined by a set of key frames. When playing an animation clip at its original frame speed, that is when dt is 1, we can calculate the real velocity of the agent in m/s by computing the displacement of the root during the animation divided by the total time of the animation clip.

The calculated velocity determines the real velocity of the animation when we run it in the global coordinate system of the virtual environment with dt=1. We can easily run the same animation at lower velocities by interpolating intermediate frames.

The Animation Planning Mediator can thus calculate the speed at which the next frame should be calculated so that it best matches the velocity of the agent given by the crowd simulation system.

If the crowd simulation module gives the same global position for an agent during two consecutive frames, then the APM can determine not to calculate a new frame by passing dt=0 to the animation module.

3.2.3 Closing the loop

With the information calculated by the APM, we need on the one hand to pass it to HALCA to carry out the animation and rendering of the agents, and on the other hand to feed back information to HiDAC in the case that any of the parameters provided (p, **v**, θ) needed to be replaced by the slightly adjusted version (p', **v'**, θ') to guarantee continuous and smooth animations (see Figure 4). This step is important to guarantee consistency between what we are observing in the rendered crowd and all the computations that HiDAC carries out regarding collision detection, avoidance and path planning.

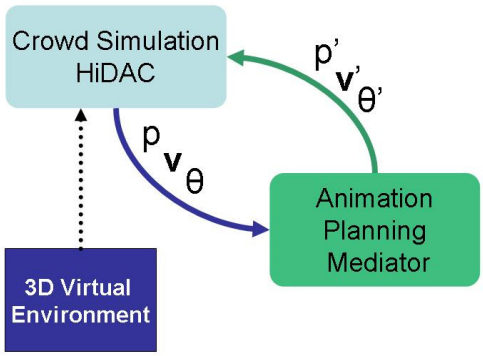

Figure 4: *Communication between HiDAC and the APM*

HALCA will proceed to render an agent in the position calculated by the displacement between the previous frame and the new frame given by the differential of time, dt, and the animation identifier, A_i. If A_i was different from the previous one, then HALCA will blend between the two animations with the blending value, b, to achieve a natural looking next frame that maintains continuity and smoothness in the locomotion. The blending value, b, is given by the APM to determine how the previous animations clip will fade out and the new one will fade in. The APM also needs to send to HALCA the correct orientation (θ) and velocity (**v**) of the character in the virtual environment, which was provided by HiDAC. If necessary, the APM can determine which joints along the upper body of the character should be re-oriented in order to satisfy the given constraints. Therefore the APM will also feed HALCA with a vector of rotations, **sk=**(α_1, α_2, α_3, **…**, α_n) that are required to adapt some skeletal segments such as the spine and head rotations after the blending of animation clips was applied [PSB09]. HALCA will then calculate the resulting skeletal configuration for the given parameters. Figure 5 shows the exchange of parameters between HALCA and the APM.

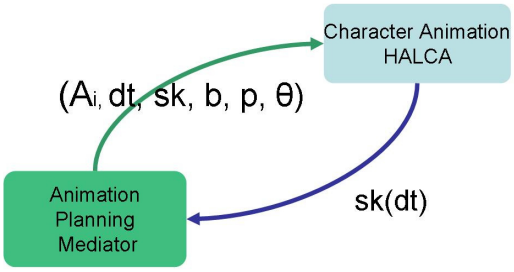

Figure 5: *Parameters from APM to HALCA in order to carry out animation (Ai, dt, sk) and rendering (p, θ)*

3.3 HALCA – Character Animation Module

For the animation and visualisation of avatars we are using the Hardware Accelerated Library for Character Animation (HALCA) [Spa09]. HALCA uses the Cal3D XML file format [Cal3D06] to describe skeleton weighted meshes, animations, and materials. The core of HALCA consists of a motion mixer and an avatar visualisation engine.

The main goal of this hardware accelerated library is to allow the user to animate and visualise several hundred realistic looking characters on single display PCs, in HMD (head mounted displays) and CAVE like systems. HALCA is closely related to Cal3D, extends Cal3D's animation and visualisation functionalities and allows the user to script it from VR applications.

3.3.1 Visualisation Engine

HALCA can be run in different rendering modes. In its simplest mode it uses basic OpenGL and can therefore run on any graphics card that supports OpenGL.

If HALCA is run in shader mode then it can either load and activate GLSL shader programs [Ros06] from a text file or it assumes that a shader program of the OpenGL context was activated by the hosting application.

In addition to the usual shading in HALCA, shaders are used to perform the deformation of an avatar's skin according to the skeleton state of the avatar. Owing to the highly parallel nature of this problem, current graphics hardware can carry out the required computations much more efficiently than the CPU. In addition, much less data is transferred between the CPU and the GPU. This is a very important feature when it comes to visualising large crowds of realistic looking avatars. This is even more important if the display system consists of multiple projectors that are driven by a networked render cluster [MTB07].

During the initialisation when avatars are loaded, the mesh information along with morph target information is loaded into OpenGL Vertex Buffer Objects (VBO)s on the GPU. HALCA has functionality to reuse vertex and image map data as much as possible. For example if two avatars

share the same mesh but have different textures, then the same VBO is used for both characters with the corresponding texture maps. This functionality is very useful in crowd simulation in order to keep the amount of data required low but to allow for variety in the appearance of avatars.

For animation in shader mode HALCA only needs to transfer the skeletal joint transformations to the GPU either as transformation matrices or as dual quaternions [KCZO07].

Two basic GLSL shader programs that can handle linear blend skinning (LBS) or dual quaternion skinning (DQS) to deform the mesh of a character according to the pose of the character's skeleton are provided. These basic shaders can easily be extended to create effects that are required in an application, such as illumination effects or more elaborate skin rendering techniques. For example, complex global illumination effects such as those created in a Virtual Light Field (VLF) were used in combination with HALCA [MYK*08].

In HALCA multiple image map files can be defined in the material files of a character in order to simulate per pixel diffuse, opacity, gloss or bump properties of an avatar's skin and clothing. For example, bump maps are used to visualise wrinkles on a character's skin. Such maps can also be used to interactively deform the character's skin by using them as displacement maps in the vertex shader by GPUs that support VTF (Vertex Texture Fetch). These techniques can be used to increase the realism of the individual avatars in the crowd.

3.3.2 Animation Mixer

HALCA gives access to properties such as the duration, frame rate, and the skeletal states of an animation to the hosting application. Such information can be useful to compute, for example, the actual walking speed of a character when animated as described in section 3.2.2.

For avatar animation HALCA extends Cal3D's abstract mixer class and adds the following functionalities:

- play and blend animations of different avatars at different speeds.
- play and blend temporal parts of an animation.
- play or morph an animation only on specified body parts.
- go through an animation not by time but by an external value. We call such animations morph animations not to be confused with morph targets.
- directly access and manipulate joint rotations and translations on top of blend and morph animations.
- efficiently access and manipulate the whole skeletal state (one function call is required to access the skeleton or to set it to a particular pose by passing the skeletal state vector of an avatar to or from HALCA). This is useful for the integration with a real time whole body motion capture system or other systems that require changes to the skeletal state of the whole body. Efficient access to the skeletal state is useful, for example, for a physics engine to check and respond to collisions between avatars or limbs of an avatar.

Blending of animations, temporal or between multiple animations is achieved by using spherical linear interpolation (Slerps).

Blending motions does not always create natural looking movements. In order to be blendable, animations have to be pre-processed to be time aligned by a Dynamic Time Warping (DTW) method.

Owing to the simple access to skeletal state of the character, several Inverse Kinematics algorithms have been created for HALCA in the S3D scripting language and in C++ [Spa09][MYK*08].

3.3.3 Animation Blending

In our system, animation clips are represented by frames of character poses. Each frame is described by a state vector that describes all joint rotations of the character's skeleton as quaternions and the character's root translation by a 3D vector. We use two types of blendings as illustrated in Figure 6. The first type of blending occurs between frames of the same animation clip. In order to smoothly play back an animation clip, a frame is created at each time t by computing the spherical linear interpolation of the two quaternions that represent the same skeletal joint rotation in the two nearest key frames. This type of blending allows us to play back animation clips at any required speed.

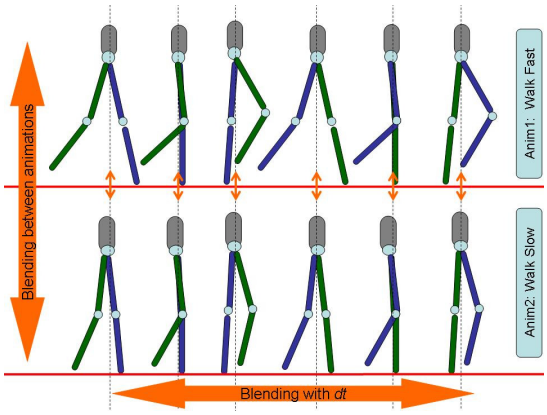

Figure 6: *Animation blending.*

The second type of blending is between frames of different animation clips, for example blending between an animation clip of a character walking slow and a character walking fast. Again, spherical linear interpolation of the joint rotations and linear interpolation for the root translation is used.

Our animation system allows us to specify the weight and delay required to blend in and out animation clips. The weight and delay are used for the interpolation calculation.

Not all animation clips are suitable for blending. For example, if we blend between a walking slow and a walking fast animation we have to make sure that the same foot is on the ground at the same time in both animations clips. This restriction has to be taken into account when manually creating key frame based animations. If motion capture data is used then the animation clips have to be preprocessed by a dynamic time warping technique such as the one presented by Kovar and Gleicher [KGP02] to ensure that the clips can be blended.

4. Results and Future Work

We have presented a framework that integrates a crowd simulation controller with a motion synthesizer. Our technique allows us to perform crowd simulation in continuous space, while generating the appropriate animations at every frame to guarantee quality of locomotion. This is of extreme importance for virtual reality applications, since no matter how good the crowd simulation method is, the realism experience is degraded if inadequate animations are played.

The framework presented has been tested with two virtual reality applications. We are currently using the system to simulate virtual humans on a 64 bit platform with visualization software based on OpenGL. This first application is for an interactive virtual game for educational purposes. We are also using the framework on a 32 bit platform with the system XVR (eXtreme Virtual Reality) [XVR08]. XVR also works with OpenGL and allows for fast development of interactive virtual environments. Currently we can simulate, animate and render up to 200 agents at 25 frames per second on a typical laptop. Our system renders the full geometry of each agent, which consists of 5000 polygons on average.

Our framework has been integrated with XVR [CTB*05][XVR08] and therefore the avatars can be displayed in a web browser on a PC, and on immersive virtual reality display devices, such as a CAVE or a head mounted display (HMD) without any additional programming work.

Currently we can simulate on a laptop with GPU in real time approximately 200 fully articulated figures with approximately 5000 polygons each, as shown in Figure 1 at over 30 frames per second.

Our framework shows visually convincing results in terms of the animations achieved by synthesizing new motions from a small set of animation clips, while conforming to the constraints imposed by the crowd simulation model.

One of the main limitations of our system is that it requires hand-created animations. It would be interesting to use motion capture data, and we are planning to do so in the future, but it will require us to implement motion graphs, or create blendable animations, that is, animations need to be pre-processed to be time aligned.

At present we are using a set of animations that correspond to walking with different velocities in combination with the velocity vector and the torso orientation. In the future we would like to include more animations such as running, jumping, or performing idle behaviors when the agents are not advancing. At present when an agent performs small steps and then the crowd simulation module sends the same position for several frames, the Animator Planning Mediator sends the value of dt=0 to HALCA which implies that the same frame will be rendered. This gives natural results when the crowd controller repeats the same position for a small period of time, but in cases where we have big lines of people waiting, it will look more natural if the agents will slightly change the skeletal state (move body weight from one foot to the other, scratch their heads, look at their watch, etc).

The main issue when it comes to including idle behaviors in our current framework is that the Animation Planning Mediator would need to know the future positions that the crowd simulation module will send for a specific number of frames. This is not currently possible, since the system works interactively in real-time, so the Animation Planning Mediator can only decide based on the current and previous frames without including any future information.

Finally it would also be interesting to run perceptual studies to evaluate the naturalness of crowd appearance, behavior and locomotion.

5. Acknowledgments

This research has been funded by the Spanish Government grant TIN2007-67982-C02-01 and the Integrated Project PRESENCCIA (6th Framework FET, Contract Number 27731)

References

[ABT00] AUBEL A., BOULIC R., THALMANN D.: Realtime display of virtual humans: Levels of detail and impostors. *IEEE Transactions on Circuits and Systems for Video Technology 10, 2* (2000), 207–217.

[BH97] BROGAN D., HODGINS, J.: Group Behaviors for Systems with Significant Dynamics. *Autonomous Robots, 4.* (1997), 137-153.

[CTB*05] CARROZZINO M., TECCHIA F., BACINELLI S., CAPPELLETTI C., BERGAMASCO M.: Lowering the development time of multimodal interactive application: the real-life experience of the xvr project. *Proc. of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*,

(2005) 270–273.

[Che04] CHENNEY S.: Flow Tiles. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, (Grenoble, France, 2004), 233-242.

[DHC*05] DOBBYN, S., HAMILL, J., O'CONOR, K. O'SULLIVAN, C.: Geopostors: A Real-Time Geometry/Impostor Crowd Rendering System, *ACM Transactions on Graphics, 24, 3* (2005), 933 - 933.

[Hei06] HEIDELBERGER B.: Character Animation Library Cal3D. (2006).

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating Dynamical Features of Escape Panic. *Nature 407* (2000), 487–490.

[HUCT04] DE HERAS CIECHOMSKI P., ULICNY B., CETRE R., THALMANN D.: A case study of a virtual audience in a reconstruction of an ancient roman odeon in aphrodisias. *VAST '04: The 5th International Symposium on Virtual Reality, Archaeology and Cultural Heirtage* (2004), 9–17.

[KCZO07] KAVAN L., COLLINS L., ZARA J., O'SULLIVAN C.: Skinning with Dual Quaternions. *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. (2007), 39-46

[KDC*08] KAVAN L., DOBBYN S., COLLINS S., ZARA J., O'SULLIVAN C.: Polypostors: 2D polygonal impostors for 3D crowds. *Symposium on Interactive 3D Graphics and Games*, (2008), 149-155

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion Graphs. *ACM Transactions on Graphics 21, 3,* (2002), 473-482.

[LCC06] LERNER, A., CHRYSANTHOU, Y. COHEN-OR, D.: Efficient Cells-and-Portals Partitioning. *Computer Animation & Virtual Worlds. Wiley, 17 (1).* (2006) 21-40.

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI, D.: Crowds by Example, *Computer Graphics Forum*, (2007), 655-664.

[LK06] LAU M., KUFFNER J.: Precomputed Search Tree: Planning for Interactive Goal-Driven Animation, *ACM SIGGRAPH / Eurographics Symp. on Computer Animation (SCA).* (2006)

[MLD*08] MCDONNELL, R., LARKIN, M., DOBBYN, S., COLLINS, S. AND O'SULLIVAN, C.: Clone Attack! Perception of Crowd Variety. *ACM Transactions on Graphics (SIGGRAPH)* 2008, 27 (3).

[MTB07] MARINO G., TECCHIA F., BERGAMASCO M.: Cluster-based rendering of complex Virtual Environments. The 4th International INTUITION Conference on Virtual Reality and Virtual Environments. (2007)

[MT01] MUSSE S.R., THALMANN, D.: Hierarchical Model for Real Time Simulation of Virtual Human Crowds. *IEEE Transaction on Visualization and Computer Graphics 7(2).* (2001) 152-164.

[MYK*08] MORTENSEN J., YU I., KHANNA P., TECCHIA F., SPANLANG B., MARINO G, SLATER M.: Real-Time Global Illumination for VR Applications. *IEEE Computer Graphics and Applications*. (2008), 56-64

[Spa09] SPANLANG B.: HALCA: a library for Presence Research. Technical Report, EventLab. Universitat de Barcelona, (2009)

[PAB07] PELECHANO N., ALLBECK, J., BADLER N. I.: Controlling Individual Agents in High-Density Crowd Simulation *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA),* (August 2007), 99–108.

[PB06] PELECHANO N., BADLER N. I.: Modeling Crowd and Trained Leader Behavior during Building Evacuation. *IEEE Computer Graphics and Applications, 26 (6).* (2006) 80-86.

[PSB09] PELECHANO N., SPANLANG, B., BEACCO. A.: Efficient Foot-Sliding Avoidance for Crowd Simulation. Submitted for publication, (2009)

[Rey87] REYNOLDS, C. Flocks, Herds, and Schools: A Distributed Behavior Model. *Proceedings of ACM SIGGRAPH*, (1987).

[Ran06] ROST. R. OpenGL(R) *Shading Language (2nd Edition) (OpenGL)* Addison-Wesley Professional, (2006)

[SGA*07] SUD, A., GAYLE, R., ANDERSEN, E., GUY, S., LIN, M. MANOCHA, D.: Real-time Navigation of Independent Agents Using Adaptive Roadmaps. *ACM Symposium on Virtual Reality Software and Technology*, (2007), 99-106.

[ST05] SHAO W., TERZOPOULOS D.: Autonomous Pedestrians. *ACM SIGGRAPH/ Eurographics Symposium on Computer Animation,* (2005), 19-28.

[TLC02] TECCHIA F., LOSCOS C., CHRYSANTHOU Y.: Image-based crowd rendering. *IEEE Computer Graphics and Applications 22, 2* (2002), 36–43.

[TCP06] TREUILLE A., COOPER S., POPOVIC Z.: Continuum Crowds. *ACM Transactions on Graphics (SIGGRAPH),* (2006), 1160-1168.

[UHT04] ULICNY B., DE HERAS CIECHOMSKI P., THALMANN D.: Crowdbrush: Interactive authoring of real-time crowd scenes. *SCA*

'04: Proceedings of the 2004 ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation, (2004), 243–252.

[WP95] WITKIN A., POPOVIĆ Z.: Motion Warping, *SIGGRAPH'95*. (1995), 105-108.

[XVR08] XVR (eXtreme Virtual Reality). http://www.vrmedia.it/.

[ZS07] ZHAO L., SAFANOVA A.: Achieving Good Connectivity in Motion Graphs, *ACM SIGGRAPH / Eurographics Proc. Symp. on Computer Animation (SCA).* (2007).