Geopostors: A Real-Time Geometry / Impostor Crowd Rendering System

Simon Dobbyn* Trinity College Dublin
John Hamill† Trinity College Dublin
Keith O'Conor‡ Trinity College Dublin
Carol O'Sullivan§ Trinity College Dublin

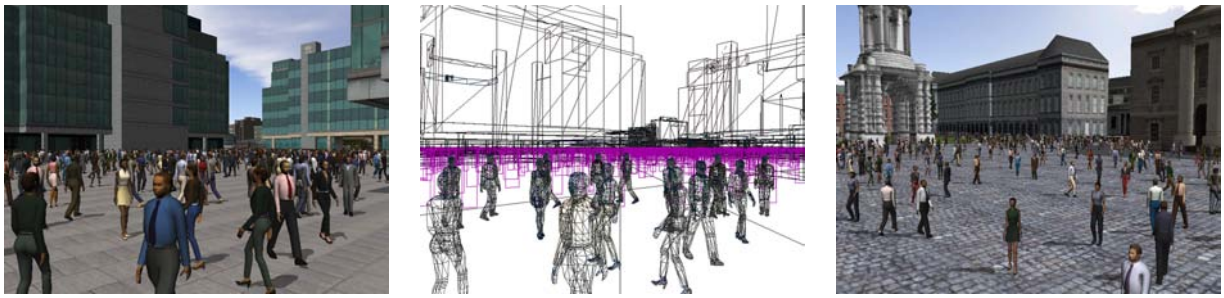

Figure 1: Real-Time Virtual Crowds in an Urban Environment

Abstract

The simulation of large crowds of humans is important in many fields of computer graphics, including real-time applications such as games, as they can breathe life into otherwise static scenes and enhance believability. We present a novel hybrid rendering system for crowds that solves the classic problem of degraded quality of image-based representations at close distances by building an impostor rendering system on top of a full, geometry-based, human animation system. This enables almost imperceptible switching between the two representations based on a "pixel to texel" ratio, with minimal popping artefacts. Seamless interchanges are further facilitated by exploiting programmable graphics hardware to efficiently enhance the realism and variety of the dynamically-lit impostors, thereby also improving on existing impostor techniques. To test our system, our virtual crowds are embedded in an urban simulation system (as shown in Figure 1). The results demonstrate a system capable of rendering large realistic crowds with the visual realism of a high-resolution geometry rendering system, but at a fraction of the rendering cost.

CR Categories: I.3.7 [Computer Graphics]: Animation— Colour, Shading, Shadowing and Texture — Virtual Reality

Keywords: human modeling and animation, simplification/level of detail, image-based rendering

1 Introduction

Although many new games are released each year, it is very unusual to find large-scale crowds populating the environments depicted. Until recently, real-time applications either resorted to fixed view methods, or could only provide for a small number of detailed virtual humans at any time. Fixed view crowds, such as those employed as spectators in several Electronic Arts sport games, are realistic from a distance. This method works well for this genre of game, however its application is limited as a result of the crowds cyclical behaviour and the fact that they become unbelievable once viewed from other angles. The more detailed geometric humans as seen in the GTA series from Rockstar can only be displayed in small numbers due to processing and display limitations. Typically fewer than 50 humans are observed at any one time.

So why is there such a lack of large virtual crowds in real-time applications? Such applications need to deal with having limited resources available each frame. With many hundreds or thousands of potential virtual humans in a crowd, traditional techniques rapidly become overwhelmed and are not able to sustain an interactive frame-rate. Therefore, simpler approaches to the rendering, animation and behaviour control of the crowds are needed. Additionally, these new approaches must provide for variety, as environments inhabited by carbon-copy clones can be disconcerting and unrealistic.

We present a system that enables the rendering of large crowds of virtual humans at interactive frame rates. Our crowd system provides for a hybrid combination of image-based (i.e. impostor) and detailed geometric rendering techniques for virtual humans. By switching between the two representations, based on a "pixel to texel" ratio, our system allows visual quality and performance to be balanced. We improve on existing impostor rendering techniques and present a programmable hardware based method for adjusting the lighting and colouring of the virtual human's skin and clothes. The system is implemented in an urban simulation, and takes advantage of the occluding nature of buildings to allow greater variety amongst the crowd with reduced memory overhead.

The paper is organised as follows: in Section 2 we introduce related work with regards to rendering large-scale crowds and how to introduce variety within the crowd. Next, we describe the modelling of a virtual human's geometric and impostor representations, followed by our new technique of switching between the two representations based on a pixel to texel ratio. We then describe how these representations are rendered in real-time, and how we introduce variety in how the models look. In Section 5, we discuss the optimisation techniques we used for implementing the crowd in a urban environ-

*e-mail:Simon.Dobbyn@cs.tcd.ie
†e-mail:John.Hamill@cs.tcd.ie
‡e-mail:Keith.OConor@cs.tcd.ie
§e-mail:Carol.OSullivan@cs.tcd.ie

ment. Finally, we tested our crowd system both alone and within a large-scale city environment and the results are presented in Section 6.

2 Background

2.1 Visualisation of Virtual Crowds

The visualisation of large-scale animated crowds is an area of research that has been receiving an increased amount of interest in recent years. Even though there has been extensive research conducted on human modelling and animation, the majority of it has been concerned with realistic approximation of limited numbers of humans, often using complex and expensive geometric representations. With regards to large-scale crowds, these approaches are too computationally expensive and different approaches are needed in order to achieve interactive frame rates [Musse and Thalmann 2001], [Tecchia et al. 2002a]. Impostors and low-resolution geometric models can provide the means whereby large crowds can be animated and rendered successfully in real-time.

In the work of Tecchia et al. [2000], pre-generated impostors are used in place of a virtual human's geometric representation. This involves the offline rendering of a set of images of the human model from different viewpoints. At run time, depending on the human's position with respect to the viewer, the most appropriate impostor image is selected and displayed on a quadrilateral dynamically orientated towards the viewer. Unfortunately, since virtual humans are animated objects, they present a trickier problem in comparison to static objects. As well as rendering the virtual human from multiple positions, multiple frames of animation for each viewpoint need to be rendered. This greatly increases the amount of texture memory required.

Aubel et al. [2000] proposed dynamically generated impostors to represent the virtual humans. This approach uses less memory than pre-generated impostors, since no storage space is devoted to any impostor image that is not actively in use. Unlike dynamically generated impostors for static objects, where the generation of a new impostor image depends solely on the camera motion, animated objects also have to take self-deformation into account. The solution of Aubel et al. to this problem is based on the sub-sampling of motion. By simply testing distance variations between some preselected points in the virtual human's skeleton, the virtual human's impostor is updated only if the posture has significantly changed.

Recently, Ulicny et al. [2004] succeeded in rendering complex scenes involving thousands of animated individuals at interactive frame rates. They achieved this by storing pre-computed deformed meshes for a frame of animation in OpenGL display lists and then carefully sorting them using the OpenGLSceneGraph 3D toolkit to take cache coherency into account. In [de Heras Ciechomski et al. 2004], they improved on their performance by using 4 level-of-detail meshes for their model, thereby achieving a frame rate several times higher.

2.2 Virtual Human Variation

Ulicny and Thalmann [2001] state that a crowd composed of the same individuals with the same behaviour would not be convincing, even if each such individual, viewed in isolation, would be very realistic. They highlight that, compared to the simulation of single virtual humans, multi-agent systems pose different conceptual and technical requirements and constraints for the design of the system. The main technical challenge is the increased demand on computational resources, as a direct result of the increased number of virtual humans in the simulation. In comparison with single-agent simulations, the main conceptual difference is the need for efficient variety management at every level, whether it is visualisation, motion control, or animation.

Therefore, to successfully model real crowds, virtual humans in a virtual crowd should look, move, and react differently to each other. By adding even subtle variations in the way each agent looks and moves, the realism of a virtual crowd can be greatly increased. With respect to using impostor techniques for rendering virtual humans, Tecchia et al. use multi-pass rendering to modify the colour of certain areas of the pre-generated impostor images of the virtual humans to enhance crowd variety [2002b]. Ulicny et al. [2004] create several template virtual human meshes, which are then modified by applying different textures, colours and scaling factors to allow variation.

2.3 Level Of Detail

The fundamental idea behind Level of Detail (LOD) is that, when a scene is being simulated, an approximate simulation model is used for small, distant, or unimportant objects in the scene. For each frame, the appropriate model or resolution is selected usually based on the object's distance to the camera [Luebke et al. 2002]. Brogan and Hodgins [2002] use simulation LOD to control the movement and actions of large groups. By providing a simplified version of a physically simulated character as a simulation LOD, they were able to simulate large groups by dynamically switching between these LODs. O'Sullivan et al. [2002] describe a framework which incorporates levels of detail for crowds and groups. In addition to using subdivision surfaces for increasing and decreasing the geometric detail of characters, they propose the idea of conversational and social behaviour LOD. Random keyframe animations are chosen for characters not highly rated, while more sophisticated motions that are synchronised with speech are applied to more salient characters.

The area of LOD provides a computationally efficient solution for the simulation of crowds. With respect to our system, we use two LODs for our virtual human representation: an impostor and a geometric representation which will be described in the next section.

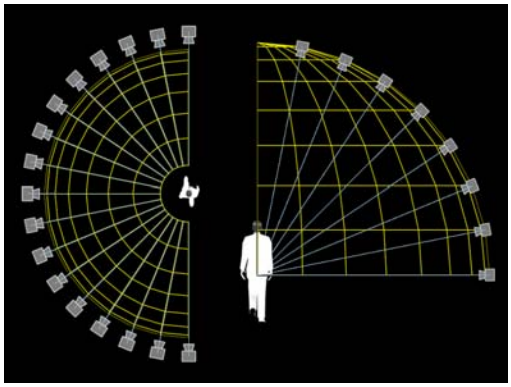

Figure 2: Generation of Impostor from 17 by 8 Viewpoints

3 Virtual Human LOD Management

For our virtual human's highest LOD representation, we use a two-layered model consisting of the skeleton and the skin. The skeleton is an articulated structure consisting of a series of interconnected joints. A single mesh is used to represent the skin layer and allows each vertex to be assigned a set of influencing joints and a blending weight for each influence (which is done in 3D Studio Max). Using a plug-in written in MaxScript, the skeletal, mesh and keyframe data are exported from 3D Studio Max into our system specific file formats.

We use pre-generated impostors for the virtual human's lowest LOD, which involves replacing a 3D object with an image of the object mapped onto a quadrilateral. This is advantageous mainly because it avoids the cost associated with rendering the object's full geometry. There are a number of reasons why we chose this approach over using low-resolution geometrical models. Firstly, less triangles need to be rendered when using an impostor instead of a low-resolution mesh. Secondly, automatic tools used to generate low-resolution meshes sometimes do not give the required results, thus necessitating a lot of time-consuming editing by hand. Finally, switching between two meshes of different resolutions can be quite noticeable as a result of the silhouettes not matching.

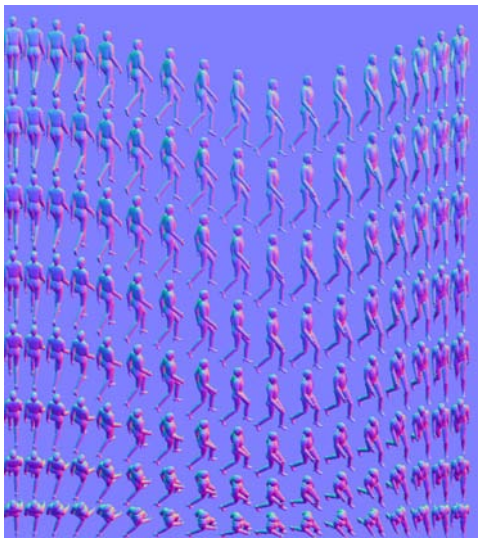

Figure 3: Normal Map Image for a Frame of Animation

A MaxScript plug-in was written, based on the work of Tecchia et al., to render the impostor images of the mesh from within 3D Studio Max. Before using the plug-in, the mesh's skeleton is animated with a simple one-second walk loop. The plug-in allows:

- **The generation of an impostor detail and a normal map image for a particular camera viewpoint.**
  These images are generated from 17 by 8 camera viewpoints as shown in Figure 2. It should be noted that for generation of the impostor detail image, the diffuse colour of the mesh's material is set to white and the texture of the head's material is grey-scaled to facilitate the introduction of variety (which will be illustrated in Section 4). Each pixel in the normal map image encodes the direction in which that particular point is facing with respect to the camera.
- **The encoding of the impostor detail image's alpha channel to allow variety.**
  Each pixel in the impostor detail image's alpha channel is encoded with a specific alpha value associated with the material ID of the material at that particular point.
- **The combining of each impostor detail and normal map image for a particular frame of animation into a single image of 1024*1024 pixels.**
  This is done for ten keyframes in the case of the walk animation (as shown in Figure3).

3.1 Switching between Virtual Human LOD Representations

The main aesthetic problem with using an impostor to represent a virtual human is that, once the human is close to the viewpoint, the impostor's flat and pixellated appearance becomes quite obvious. As stated in [Ulicny et al. 2004], this makes impostors a good approach for far-away humans that do not need detailed views. However, our main contribution is to allow a virtual human to switch to a higher LOD representation, based on some selection criteria, which should greatly improve the realism of the simulation. Our approach is to switch between a virtual human's impostor based representation and the mesh that was used to generate the impostor, based on an impostor image pixel size to impostor texel size ratio.

Given the set of viewing parameters used in the generation of the impostor images in 3D Studio MAX, we can compute the size of an impostor image's pixel as shown in Figure 4. In this figure, θ is the camera's field of view, d_{cam} is the distance along the viewing direction from the camera to the virtual human model, and x is the resolution of the impostor image. Using these values, the size of an impostor image's pixel can be calculated using Equation (1).

$$Pixel_{size} = \frac{tan^{-1}(\frac{\theta}{2}) \times 2 \times d_{cam}}{x} \quad (1)$$

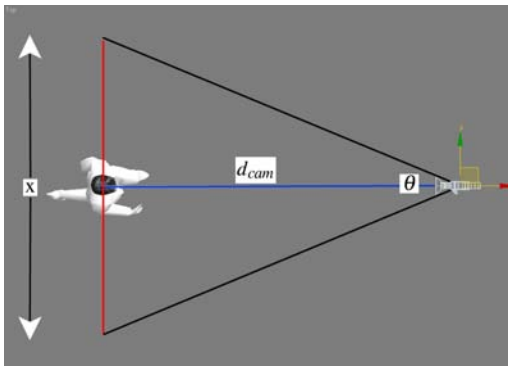

Figure 4: Calculating Impostor Image Pixel Size

A texel (TEXtured ELement) is the basic unit of measurement when dealing with texture mapped 3-D objects. We propose that the switch between the impostor and mesh representation of the human model should happen when the ratio of the pixel size of the impostor image to the texel size of the impostor's texture mapped quad equals a certain threshold. The ideal ratio at which the switching should happen is at 1:1, as aliasing starts to occur when the texel size is greater than the pixel size, resulting in the stretching of the texture on the impostor's quad. Given the set of viewing parameters of the system and using the properties of similar triangles, the distance d_{switch} at which the pixel to texel ratio is equal to one, can be calculated using Equations(2)(3). Note that θ is the camera's field

of view, $d_{nearplane}$ is the distance along the viewing direction from the camera to the near plane, and x is the resolution of the screen in pixels.

$$Texel_{SIZE} = \frac{(2 \times d_{nearplane} \times tan^{-1}(\frac{\theta}{2}))}{x} \quad (2)$$

$$d_{switch} = \frac{d_{nearplane} \times Texel_{size}}{Pixel_{size}} \quad (3)$$

At run-time, if the virtual human is within the view-frustum and the distance of the human from the viewer is less than d_{switch}, the system switches from the impostor based representation to the mesh representation. The appropriate frame of animation for the mesh is selected based on the impostor's current frame allowing a seamless transition.

4 Real-Time Rendering of the Virtual Human

4.1 Rendering of the Geometric Representation

Depending on the impostor's current frame of animation, the position and orientation of each bone is updated based on the corresponding keyframe. Once the skeletal structure has been updated, the positions of all the bones affecting each vertex are used to calculate the final position of the mesh's vertices. By pre-calculating and storing each vertex position and normal transformation for each frame of animation, we replace the complex, articulated skeletal model with a simple rigid mesh or "pose". This avoids the cost of deforming the mesh, and allows the entire model to be stored relative to a single bone or matrix. To improve the rendering speed of a pose, we minimise OpenGL state changes [Shreiner et al. 2004], as well as encapsulating the pose's data in a vertex buffer object (VBO). Vertex buffer objects allow data to be stored in high performance memory on the server side and therefore increase the rate of data transfer.

To improve the variety of our models, we use a set of different human meshes and change their appearance by using different "outfits". Outfits define a set of colours for the mesh's skin and clothes, where each colour is associated with a specific material ID. In 3D Studio MAX, the diffuse colour of each mesh's materials is set to white and each material is tagged with a specific ID to define whether it represents skin or a particular type of clothing (e.g. trousers, shirt, or jacket). When the mesh is rendered in the system, the diffuse colour of a material is changed depending on the material's ID and the colour associated with that ID, which is defined by the outfit being used. In the case of a material that uses a texture map, the texture is grey-scaled in 3D Studio MAX to allow colour modulation without losing detail. Several different outfits can be defined for each human model mesh allowing great variety with minimal memory overhead.

4.2 Rendering of the Impostor Representation

The main problem with using a pre-generated impostor approach is the consumption of texture memory. In order to render a dynamically lit impostor, an impostor detail image and a normal map image are required (for each frame of animation). Since the RGBA impostor detail image contains four channels (1024*1024*4 bytes) and the RGB normal map image contains three channels (1024*1024*3 bytes), this results in 7MB of texture memory being required. By using DXT3 texture compression, the memory requirements are reduced by a factor of four for RGBA images and by a factor of six for RGB images, resulting in only 1.5MB (1024*1024*4*1/4 + 1024*1024*3*1/6 bytes) of texture memory for each frame.

Given the amount of texture memory required by the system, we need a method to improve the variety and visual interest of large crowds of impostors while keeping memory usage to a minimum and ensuring that rendering speed is uncompromised. To improve variety, changing the colours of an impostor's clothing and skin is a method that is simple and yet has high visual impact when viewed in a crowd. While a multi-pass method as described in [Tecchia et al. 2002b] achieves this goal, it does so by performing a rendering pass for every different region of colour that needs to be changed. For improving realism, interactive lighting of impostors is highly desirable. In addition to aesthetic considerations, this is essential for a system that allows the impostor to switch to a geometric representation. By using a per-pixel dot product between the light vector and a normal map image, Tecchia et al. compute the final value of a pixel through multi-pass rendering and require a minimum of five rendering passes. However, multi-pass rendering can have a detrimental effect on rendering time, which limits both the number of impostors that can be shaded in real-time as well as the number of regions that can be changed when adding variety.

Our contribution in this area is that we improve upon existing impostor techniques for adding variety by taking advantage of recent improvements in programmable graphics hardware to perform an arbitrary number of colour changes in one pass. Since the colouring regions are encoded in the alpha channel (as described in Section 3), this number is limited only by that channel's precision. Our further contribution is the real-time shading of the impostors in a single pass, implemented in programmable hardware.

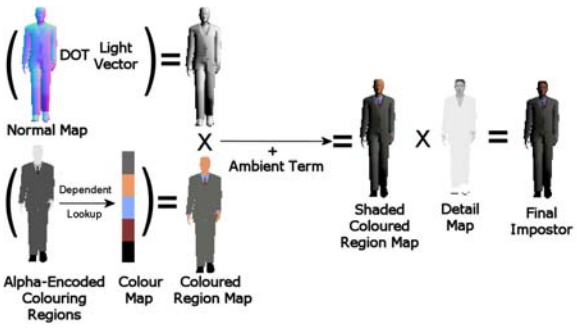

Figure 5: Impostor Colouring Sequence

4.3 Real-Time Impostor Shading using Programmable Graphics Hardware

As we are presenting a hybrid system that switches between two representations, it is crucial that there is no difference in the shading of each representation for the interchange to be imperceptible to the viewer. As the geometric representation is shaded using the fixed function pipeline, the shading of the impostor in hardware needs to match this. With respect to our system, we are using a single directional light with materials that have no specular, emission or shininess properties, resulting in the simplification of the OpenGL lighting Equation (4).

$$Vertex\_Colour = Ambient_{LightModel} \times Ambient_{material} + MAX((Vector_{light} \cdot Normal_{vertex}), 0) \times Diffuse_{light} \times Diffuse_{material} \quad (4)$$

The lighting of the impostor representation has been implemented in hardware using both texture shaders and register combiners [SGI b], and vertex and fragment programs [SGI c] [SGI a], depending on available hardware. This involves Equation (5), wherein the per-pixel dot product of the light vector and the pre-generated normal map is multiplied with the coloured region map (which will be discussed in the next section) to produce a "shaded coloured region map". This result is added to an ambient term, and multiplied with the detail map to yield the final lit, coloured pixels. The overall shading and colouring sequence is illustrated in Figure 5.

$$Pixel\_Colour = (Ambient_{LightModel} \times Ambient_{material} + MAX((Vector_{light} \cdot Normal_{Map}), 0) \times Colour_{Map} \times Diffuse_{light})) \times ImpostorDetail_{Map} \quad (5)$$

4.4 Adding Variety with Programmable Graphics Hardware

We exploit the programmability of graphics hardware to efficiently increase the variety and interest of each impostor. In order to match the virtual human's geometric representation, the impostors must also be able to change colour, depending on human model and outfit materials. We achieve this by storing distinct material IDs in the alpha channel of the impostor detail image upon generation, and use these IDs to address a changeable colour map at run-time. While this would not be possible with the fixed function pipeline, programmable texture addressing allows texture indirection to be employed, meaning the output of one texture lookup can be used as the texture coordinates of a subsequent lookup. Therefore we perform a lookup on the detail map, using the alpha-encoded material IDs to address a colour map texture that can be altered to match the outfit of the virtual human currently being rendered. The colour maps used are one-dimensional textures with one pixel per colour, meaning many different outfits can be stored with negligible memory usage. An added bonus is that the impostor colouring is controlled entirely by artist-drawn textures, allowing a quick and easy method of producing many different colour maps that are realistic and suitable to the model being rendered (Figure 6).

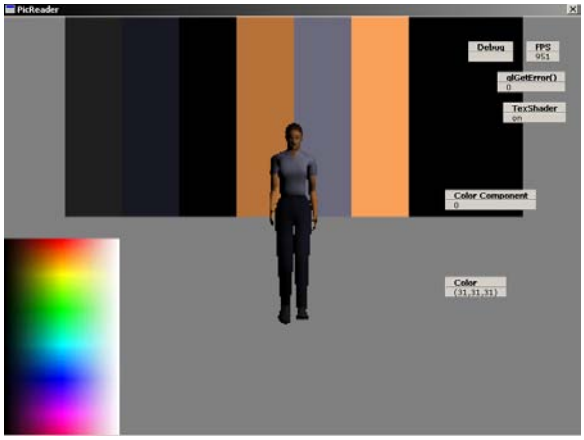

Figure 6: Generating Colour Maps to Add Variety

5 Implementation of Crowd System in an Urban Environment

To examine the real world performance and usability of our crowd system we incorporate it into an existing virtual city simulation [Hamill and O'Sullivan 2003]. This is a highly detailed model of Dublin city centre, covering several square kilometres, implemented as an OpenGL first person perspective application.

Performance of the crowd simulation in an urban setting can be greatly increased due to the densely occluded nature of the environment. We utilise view frustum culling as a first order step to eliminate those humans not potentially on screen. We further make use of hardware accelerated occlusion culling, with a grid based method, to cull large sections of the crowd when the camera is near ground level. We divide the world into a regular grid of human-height nodes and store which humans are present in each node. Having initially rendered the static environment, we perform occlusion queries on the bounding volume of these nodes allowing us to rapidly discard those nodes hidden by the environment, and thus the humans within them.

While frustum and occlusion culling decrease the rendering workload, there are still overheads associated with updating the positions of thousands of humans in motion. To lighten the workload we pause the updating of humans that have not been visible for more than five seconds. This time delay prevents temporal artefacts becoming noticeable amongst the nearby humans when performing rapid camera rotation.

To avoid the need for complex collision detection algorithms but still allow the virtual humans to navigate the city, we implement a series of walkable area maps. These are primarily stored as 1bit BMPs to allow low memory consumption and rapid lookup of the binary walkable data, but additional modes are available allowing height field information, potential field data, or colourisation data to be stored.

Stencil buffer shadows are used in the city simulation to enhance realism. Our shadow technique is shared by both the city simulation and the crowd. It operates by projecting the geometry / impostor onto the ground plane, rendering into the stencil buffer. A single semi-transparent quad is rendered over the whole scene (where the stencil buffer has been set) resulting in realistic blended shadows. While this method is similar to that employed by Loscos et al. [Loscos et al. 2001], our use of the stencil buffer instead of darkened textures allows shadows to blend realistically with the underlying world and each other without blocking or z-buffer fighting.

5.1 Reducing Texture Thrashing in a Virtual City Environment

In the case of populating a virtual city with crowds using impostors, the number of human models that can be used is limited, otherwise texture thrashing becomes a problem. In addition to each human model requiring 1.5MB of texture memory for its impostor, the city model will also require a certain amount of texture memory. Therefore, as the number of types of human models increases, texture thrashing will occur much sooner as a result of the extra texture memory being consumed by the city model. It should be noted that

in the case of real-time applications where the camera is fixed, say at eye-level, only 17 viewpoint images are needed for each frame of animation, and therefore texture thrashing is less of a problem. Since we wanted to implement a more generic system, where the camera can view the city from any viewpoint, 17 by 8 viewpoints are needed for the impostor.

However, as only a subset of the viewpoints in the textures is being used when rendering a frame, we propose splitting the impostor detail and the normal map images into eight separate smaller "elevation" images containing the set of viewpoints for each elevation. An application was written to facilitate the creation of these elevation images. The application reads in the 17 viewpoint images for a particular elevation and, based on the sum of the viewpoint images' area, the minimum dimensions of the elevation image are calculated. Once the viewpoints have been read in, the application allows the user to organise the viewpoints within the new elevation image. Unfortunately, since the dimension of each viewpoint image varies, it is not guaranteed that they will all fit within the minimum dimensions and therefore these sometimes have to be increased. Once the user has got all the viewpoint images to fit, the new elevation image is exported as shown in Figure 7.

Figure 7: Normal Map Elevation Image

The number of elevation images needed to render impostors using a particular human model type depends on the height of the camera and the distance of the camera to each impostor. Since buildings in a city environment generally occlude humans in the distance, all elevation images should never be needed simultaneously. The angle (θ_E) between the impostor and the camera around the horizontal axis, can be calculated using Equation (6), where h_{cam} is the camera height and d_{xz} is the distance on the x-z plane from the camera to the impostor. Using θ_E, the elevation image needed for that impostor can be calculated. From Equation (6), it can be noted that as camera height decreases, the number of elevation images needed is reduced dramatically.

$$\theta_E = tan^{-1}(\frac{h_{cam}}{d_{xz}}) \quad (6)$$

Taking advantage of the occluding nature of city environments, this method of separating impostor and normal map images for each elevation permits greater variety without texture thrashing as a result of each human model type consuming less texture memory.

6 Performance of Run-Time System

We measured the performance of the crowd rendering system by itself, and also within an urban simulation system to test its overall impact. All of our tests were performed using a PentiumIV 2.0Ghz processor, with 512Mb RAM and a GeForce 4 Ti4600 3D card with 128MB of video memory.

6.1 Performance of the Crowd Rendering System

We ran tests investigating how the number of virtual humans and the representation used affected the frame rate. These tests used an impostor and a geometric representation (consisting of approximately 2200 triangles) for 1, 10, 100, 250, 500 and 1000 virtual humans as shown in Figure 8. It should be noted that for each test, all of the virtual humans were fully lit but never frustum or occlusion culled and were therefore always on-screen.

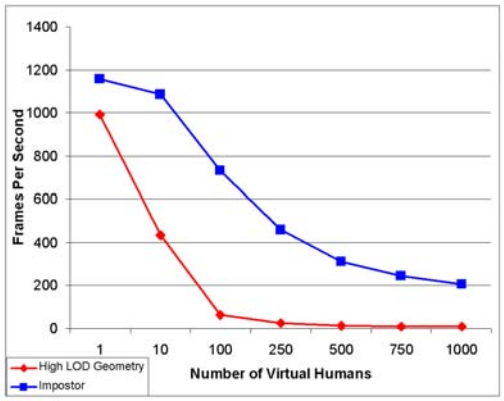

Figure 8: Impostor Vs Geometry

We also tested how using our two LOD representations affected the system's performance in comparison to just using an impostor representation. These tests were carried out for 1,000 - 10,000 virtual humans at 1,000 human intervals. A maximum of 10,000 virtual humans was chosen as this was considered to be the maximum amount that would be needed on-screen for scenes such as an army of characters or a stadium of spectators. In these tests, the number of virtual humans using the geometric representation was set to 100 to keep their rendering cost constant thus allowing the performance impact of using the impostors to be measured. Again, for each test all of the virtual humans were on-screen and lit as shown in Figure 9. The graph in Figure 10 illustrates that in the impostor/geometry case, the impostor representation has a minimal impact on the rendering time as the number of virtual humans increases.

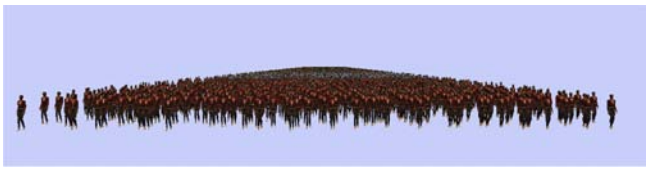

Figure 9: Frame Rate Tests for 10,000 Humans (9,900 Impostors 100 Geometry)

6.2 Performance in an Urban Simulation System

To provide a baseline, we recorded the frame rates during a 2000 frame walkthrough of the virtual city containing no humans. The city application contains over 100,000 Polygons and uses over 200MB of texture data, and so provides a good real-world application in which to test our crowds. We then added 5,000, 10,000, 20,000, and 30,000 virtual humans respectively to the environment to see how the frame rates of each walkthrough were affected in comparison to the baseline test. The selection criteria used for

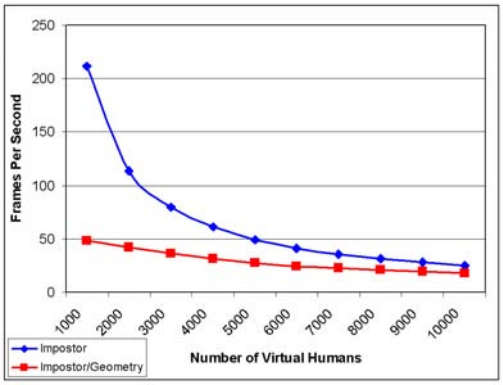

Figure 10: Impostor Vs Impostor/Geometry

switching between representations was a pixel to texel ratio of 1:1. It should be noted that in these experiments, the virtual humans were both frustum and occlusion culled, and lighting was enabled. As can be seen in Figure 11, interactive frame rates are maintained for crowds of up to 30,000 individuals.

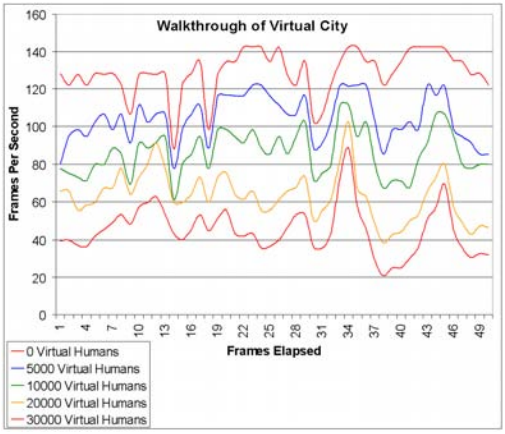

Figure 11: Walkthrough of Virtual City

Finally, to test the effect of splitting the impostor textures into separate elevation textures as opposed to using the original textures of 1024 by 1024 pixels, we recorded the frame rates during a 2000 frame walkthrough of the virtual city containing 10,000 individuals. These tests were run with 4, 6, 8, and 10 different human models for both types of textures. To avoid texture thrashing due to the city's models, the buildings were rendered without textures. As shown in Figure 12, once the number of human models exceeds four, texture thrashing becomes a problem when using the original textures. However, up to ten human models can be used without texture thrashing affecting the frame rate when the impostor textures are split into separate elevation textures as shown in Figure 13.

7 Conclusions and Future Work

We have presented a hybrid system for the real-time rendering of large-scale animated crowds. This system implements a LOD approach by using an image based representation for virtual humans in the distance, and switching to a geometric representation once the human is within a certain distance threshold based on a texel

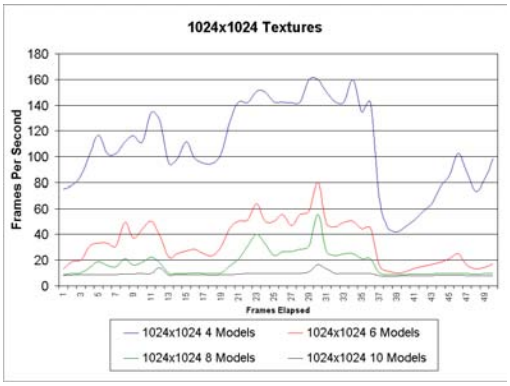

Figure 12: Frame Rates Using 1024x1024 Textures

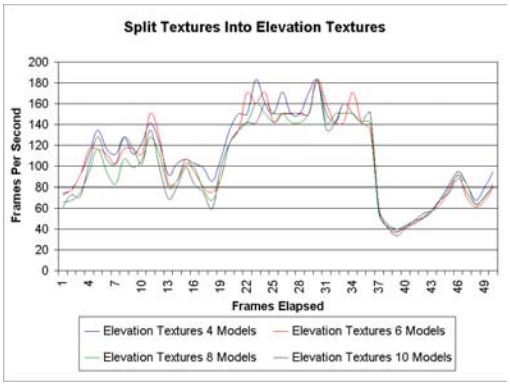

Figure 13: Frame Rates Using Elevation Textures

to pixel ratio. The shading of the impostors and the introduction of variety into the virtual human's geometric and impostor model is achieved at run-time through programmable graphics hardware. We have shown that our system allows large crowds to be animated at interactive frame rates.

Our results so far have convinced us that human impostors are a excellent substitute for geometry, not only because of proven rendering efficiency gains, but also in terms of visual fidelity. At certain distances, it is virtually impossible to determine whether the high-resolution model or the impostor is being rendered. The visual realism of our impostor representations is one of the clear advantages this method has over other approaches, such as using low polygon models. Up to now, we have used a texel to pixel ratio of 1:1 to decide when to switch virtual human representations, but from our observations it is clear that this is a conservative estimate. For example, when a single virtual human is rendered against a high constrast background, the interchanges are more easily detected and the 1:1 ratio is a reasonable estimate of the perceptible threshold. However, when the same human is placed within the complex urban environment, with many background details, it becomes almost impossible to notice the change until the model is very close to the camera. These informal observations now need to be evaluated empirically and we are currently running a set of perceptual experiments to find appropriate thresholds and to evaluate the factors affecting the perception of human models and their motions for a range of representation schemes.

This paper mainly focussed on rendering techniques for large animated crowds. The current behaviour of the virtual crowds is fairly simplistic and the impostors are limited to the motion that was used

in their generation. With our crowd rendering system in place, we plan to extend our work into implementing a scalable system that provides our virtual humans with more believable behaviours using a variety of motions. By implementing matrix palette skinning using programmable hardware, this would allow subtle but important variations in the motions used to animate the geometric models with minimum overhead. For example, this could be used to animate a human looking around carefully before crossing a road.

Futhermore, the splitting of the human impostor into separate body parts (such as the head, the arms, the legs, and the torso), would allow the mixing of different pre-generated impostor animations. For example, different arm or head gestures could be used depending on the virtual human's behaviour. With the latest generation of graphics cards having 256Mb of RAM and the imminent spread of PCI-Express which greatly increases bus bandwidth, the current issue of texture thrashing will become less of a problem.

To improve on the behaviour, Level of detail AI (LODAI) techniques will be investigated. LODAI reduces the high CPU demands of AI by approximating the behaviour of agents who are rated with a low-level of importance. A LODAI approach could be used to determine whether a virtual human exploits areas such as pathfinding level of detail, pathfinding cheating, AI update frequency and collision avoidance.

References

AUBEL, A., BOULIC, R., AND THALMANN, D. 2000. Real-time display of virtual humans: Levels of details and impostors. *IEEE Transactions on Circuits and Systems for Video Technology 10*, 2, 207–217.

BROGAN, D., AND HODGINS, J. 2002. Simulation level of detail for multiagent control. *In International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 199–206.

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

HAMILL, J., AND O'SULLIVAN, C. 2003. Virtual dublin - a framework for real-time urban simulation. *Proc. of the Winter Conference on Computer Graphics 11*, 1-3.

LOSCOS, C., TECCHIA, F., AND CHRYSANTHOU, Y. 2001. Real-time shadows for animated crowds in virtual cities. *Proceedings of the ACM symposium on Virtual reality software and technology*, 85–92.

LUEBKE, D., WATSON, B., COHEN, J., REDDY, M., AND VARSHNEY, A. 2002. Level of detail for 3d computer graphics. *Elsevier Science Inc*.

MUSSE, S. R., AND THALMANN, D. 2001. A hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics 7*, 2 (April-June), 152–164.

O'SULLIVAN, C., CASSELL, J., VILHJÁLMSSON, H., DINGLIANA, J., DOBBYN, S., MCNAMEE, B., PETERS, C., AND GIANG, T. 2002. Levels of detail for crowds and groups. *Computer Graphics Forum 21*, 4.

SGI. Nv_fragment_programs. *http://oss.sgi.com/projects/ogl-sample/registry/ARB/fragment_program.txt*.

SGI. Nv_register_combiners. *http://oss.sgi.com/projects/ogl-sample/registry/NV/register_combiners.txt*.

SGI. Nv_vertex_programs. *http://oss.sgi.com/projects/ogl-sample/registry/ARB/vertex_program.txt*.

SHREINER, D., KUEHNE, B., TRUE, T., AND GRANTHAM, B. 2004. Performance opengl: Platform-independent techniques. *In SIGGRAPH '04 Course*.

TECCHIA, F., AND CHRYSANTHOU, Y. 2000. Real-time rendering of densely populated urban environments. *Proceedings of the Eurographics Workshop* (JUNE), 83–88.

TECCHIA, F., LOSCOS, C., AND CHRYSANTHOU, Y. 2002. Image based crowd rendering. *IEEE Computer Graphics and Applications 22* (March/April).

TECCHIA, F., LOSCOS, C., AND CHRYSANTHOU, Y. 2002. Visualizing crowds in real-time. *Computer Graphics Forum 21* (December).

ULICNY, B., AND THALMANN, D. 2001. Crowd simulation for interactive virtual environments and vr training systems. *Proceedings of Eurographics Workshop on Computer Animation and Simulation 2001*, 163–170.

ULICNY, B., DE HERAS CIECHOMSKI, P., AND THALMANN, D. 2004. Crowdbrush: Interactive authoring of real-time crowd scenes. *Proceedings of ACM SIGGRAPH Symposium on Computer Animation* (August).

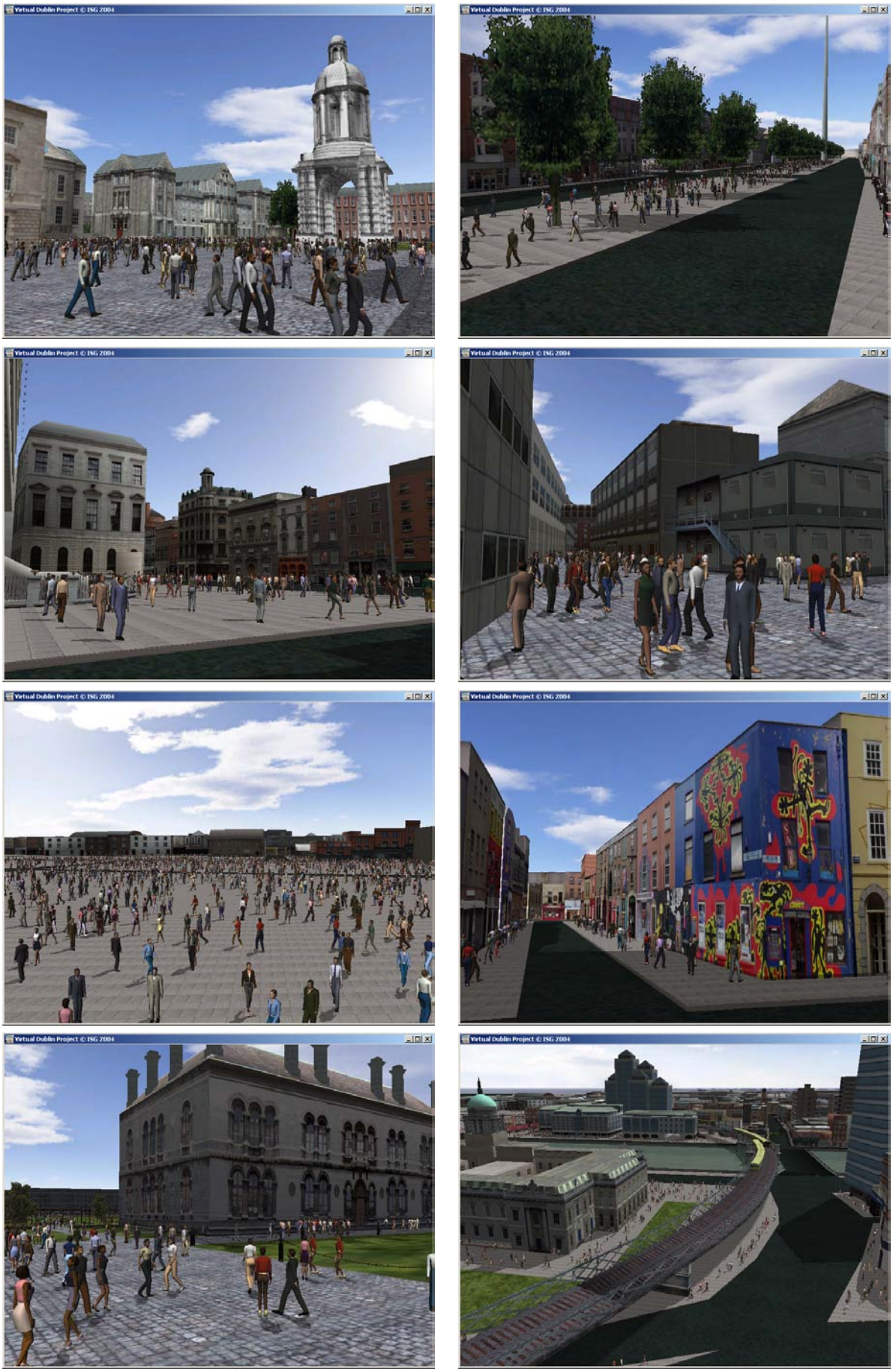

Figure 14: Geopostor Crowds in Virtual Dublin