PIPELINE FOR POPULATING GAMES WITH REALISTIC CROWDS

Rachel McDonnell, Simon Dobbyn and Carol O'Sullivan

Graphics, Vision and Visualisation Group
Trinity College Dublin
Ireland
email: Rachel.McDonnell@cs.tcd.ie

Figure 1: *Realistic crowds in virtual Dublin.*

KEYWORDS

Real-time Crowds, Impostors, Cloth simulation, Perceptual Metrics

ABSTRACT

With the increase in realism of games based in stadiums and urban environments, real-time crowds are becoming essential in order to provide a believable environment. However, realism is still largely lacking due to the computation required. In this paper, we describe the pipeline of work needed in order to prepare and export models from a 3D modelling package into a crowd system. For our crowd, we use a hybrid geometry/impostor approach which allows thousands of characters to be rendered of high visual quality. We use pre-simulated sequences of cloth to further enhance the characters appearance, along with perceptual metrics to balance computation with visual fidelity.

INTRODUCTION

In the games industry, real-time crowds are increasing in popularity due to the emergence of new level of detail (LOD) techniques. Currently, the humans that occupy crowd systems in games use skinned meshes for their clothing, often resulting in rigid and unnatural motion of the individuals in a simulated crowd. While there have been advances in the area of cloth simulation, both offline and in real-time, interactive cloth simulation for hundreds or thousands of clothed characters would not be possible with current methods. We addressed this problem in previous work by devising a system for animating large numbers of clothed characters using pre-simulated clothing.

In Dobbyn et al. [Dobbyn et al. 2006], we added realism to our crowd simulations by dressing the individuals in realistically simulated clothing, using an offline commercial cloth simulator, and integrating this into our real-time hybrid geometry/impostor rendering system ([Dobbyn et al. 2005]). We have also improved the quality of the impostors in the system by using perceptual metrics described in [Hamill et al. 2005; McDonnell et al. 2005; McDonnell et al. 2006; McDonnell et al. 2007a] in order to drive the impostor rendering.

In this paper, we discuss the necessary steps to advance from a single character in a modelling package such as 3D Studio Max, to a perceptually guided crowd of thousands of clothed characters in a real-time system. This useful step by step guide will enable others to incorporate presimulated deformable clothing and perceptual metrics into their crowd systems. Unlike previous work, we discuss in detail the challenges that we faced when incorporating clothing into the system and the practical use of our perceptual metrics. Our results show a system capable of rendering large realistic clothed crowds in real-time.

BACKGROUND

Real-time crowds pose a difficult problem as there may be limited resources available at run-time. As the number of characters increases, traditional techniques rapidly become overwhelmed and are unable to sustain interactive frame-rates. Generally, it is the number of polygons that the graphics engine can display per frame that limits the number of characters that can be displayed in a real-time crowd. Therefore, character meshes with a high number of polygons need to be simplified in order to achieve real-time display rates. Instead of simplifying all of the characters to a low level of detail, different levels of detail of the characters' meshes can be used in the scene. Detailed meshes are used when the object is close to the viewer, and coarser approximations are substituted for distant objects (e.g., as in [Funkhouser and Squin 1993]). This idea of using different levels of detail for objects in a scene is an established technique. As early as 1976, Clark [Clarke 1976] described the benefits of representing objects within a scene at several resolutions.

Recently, two main approaches to real-time crowds have emerged in the crowd rendering community. One is to use low resolution geometry, and the other is to use image-based representations for the characters in the crowd. Ulicny et al. [Ulicny et al. 2004] replaced full geometrical models with lower resolution ones, and were able to create complex scenes with thousands of characters. De Heras Ciechomski et al. [de Heras Ciechomski et al. 2004] significantly improved performance by using four discrete level-of-detail meshes for the humans in their crowd. Yersin et al. [Yersin et al. 2005] increase the number of humans significantly while still having varied animations by switching between dynamic and static meshes, depending on the distance from the camera.

Another approach to reducing the rendering costs of large crowds is to replace the entire geometry of the character with two-dimensional representations (impostors). Aubel et al. [Aubel et al. 1998] were the first to use this technique for crowds of virtual humans. They describe their dynamic impostor as "a simple textured plane that rotates to continuously face the viewer." A snapshot of the virtual human is textured onto the plane. The texture is updated from time to time, depending on the movement of the human or camera. The snapshot is produced at run-time by rendering the virtual human to an off-screen buffer. The polygonal model is then replaced with a single plane with this texture pasted onto it. Tecchia et al. [Tecchia et al. 2002] were able to render a virtual city with thousands of walking humans on a standard PC by using pre-generated impostors to represent their humans. They improved on existing techniques by shading the impostors in real-time.

One of the main disadvantages of the previous approaches was that the impostors looked pixellated when viewed up close. Dobbyn et al. [Dobbyn et al. 2005] presented the first hybrid crowd system to solve the problem of degraded quality of impostors at close distances. They did so by building an impostor system on top of a full, geometry-based human animation system, and switching between the two representations with minimal popping artifacts. They also improved on previous impostor techniques using graphics hardware to en-

hance the variety and realism of the impostors. Alternatively, Pettre et al. [Pettre et al. 2006] used three levels of detail, thus combining the strengths of the animation quality of dynamic meshes with the performance of static meshes and impostors.

The depiction of cloth in computer graphics has occupied researchers and movie-makers since the 1980's. Today, the need for realistic clothing has become more and more important for computer generated sequences in movies. In the CG research community, three types of physically-based models have been used for cloth simulation: Elasticity-based methods (e.g., [Carignan et al. 1992; Yang and Magnenat-Thalmann 1993]), particle-based methods (e.g., [Breen et al. 1994]), and the mass-spring model (e.g., [Provot 1995]). Implementing realistic cloth dynamics in real-time game applications still represents a significant challenge for game developers. Simulating cloth deformation requires much computation and is a complex process both in terms of the dynamics simulation and the collision detection for the changing cloth shape. In the animation research community attempts have been made to produce more reliable real-time clothing for characters (e.g., [Vassilev et al. 2001; Cordier and Magnenat-Thalmann 2005]).

While generating cloth dynamics in real-time is becoming more advanced, the simulation groups or crowds of characters with simulated clothing would not be possible with these techniques. All previously described crowd systems used skinned meshes to clothe the humans in their crowds. In [Dobbyn et al. 2006] we describe how pre-simulated cloth sequences can be used to create large numbers of clothed characters. Perceptual metrics are also of extreme importance in order to make the most out of a crowd. In [Hamill et al. 2005; McDonnell et al. 2005; McDonnell et al. 2006] we developed metrics for impostors and low resolution geometry, in order to balance visual fidelity with performance.

In this paper, we describe the techniques necessary for creating a perceptually guided crowd of clothed characters using pre-simulated cloth sequences and perceptual metrics, starting from just one character in a modelling package.

PIPELINE

We present the pipeline of work needed to achieve real-time crowds of clothed characters. In [McDonnell et al. 2006] we performed a rigorous series of psychophysical experiments in order to determine the most appropriate representations to use for depicting the deformations of cloth. We found that a hybrid combination of impostor and detailed geometry meshes would be most suited to our purpose since participants were unable to tell the difference between this type of crowd and a crowd containing all high resolution geometry.

The system is an extension of the crowd system detailed in [Dobbyn et al. 2005], and this paper describes all of the alterations necessary to convert that crowd of skinned characters, to a crowd of characters with deformable clothing.

There are three main stages in the pipeline process (see Figure 2):

- **Stage 1**: Model Prepping: The first stage involves modelling the character and adding a suitable animation for use in a crowd. The character's clothes are then constructed and fitted, and finally simulated in response to the underlying motion. Our novel technique detailed in [Dobbyn et al. 2006] for constructing a cyclical cloth animation is also applied here.
- **Stage 2**: Exporting the Data: The second stage is concerned with exporting the human mesh, skeletal animation, cloth mesh and simulation, and impostors into a suitable format for use in the system.
- **Stage 3**: Crowd Rendering: The final stage involves details on how the crowd is rendered in real-time.

  The following three sections will describe the stages in detail. We also present some

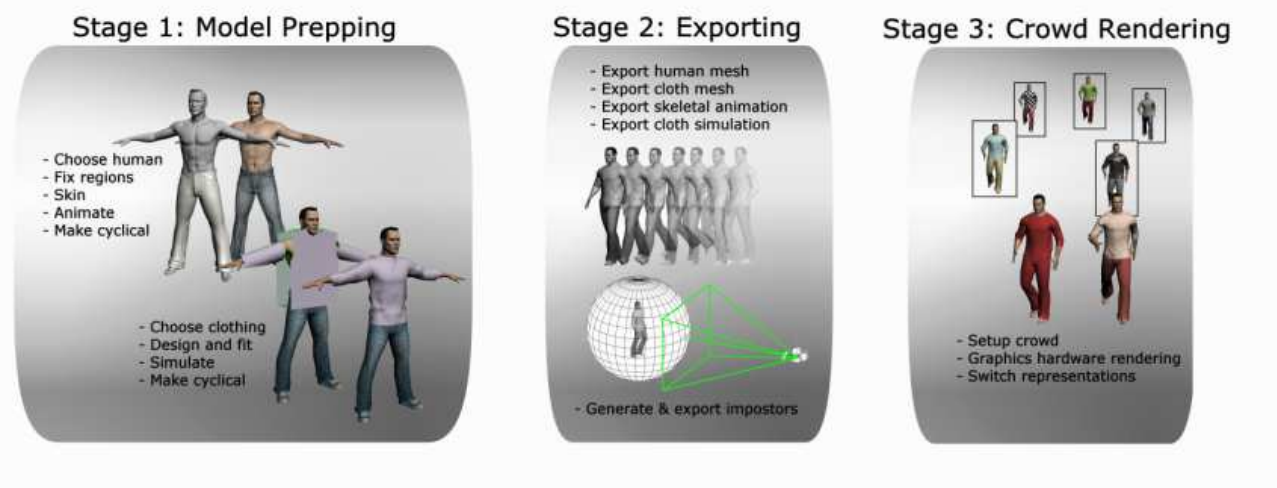

Figure 2: *Pipeline stages.*

images of the final crowd system integrated a Virtual model of Dublin city [Hamill and O'Sullivan 2003].

STAGE 1: MODEL PREPPING

Stage 1 of the process is concerned with the processing that is necessary in order to prepare a virtual character to be exported into the system.

Preparing the Human

Choosing a suitable model is the first thing that needs to be done. Characters in a crowd should look different from each other but should have the same overall style (e.g., a cartoon style character and a photorealistic character will not look good together in a crowd).

We discovered in [McDonnell et al. 2005] that characters of lower resolution around the joints are not as capable as those with more detail of displaying subtle motion variations. Therefore, the character should be detailed enough for both realism and the accurate deformation of the mesh around the joints in order to produce good animations. However, when choosing how detailed the character will be, it should be remembered that, for a hybrid system, a number of the high level of detail characters will be displayed on-screen at one time, as well as many low level of detail characters. Therefore, the detail of the character is limited to what is achievable in real-time. Our characters ranged in detail from roughly 2000 to 9000 polygons, depending on the number of characters needed for the crowd. However, we also found in [McDonnell et al. 2005] that the resolution that we were using for our highest characters was actually perceptually equivalent to lower resolution characters. Therefore, there is a tradeoff between the number of polygons necessary for the virtual human to look good in appearance and the number necessary for displaying subtle motion information. This should be remembered when choosing the resolution of the high resolution character. We will use the example of a female character model called *Angela* with 6500 polygons for the remainder of our description of the pipeline process.

Figure 3: *(a) material IDs assigned to mesh, (b) mesh with greyscaled textures and diffuse colours set to white, (c) mesh with skeleton fitted.*

Once a suitable character is chosen, it can be altered in one of the packages for 3D mod-

elling. We used 3D Studio Max but any other modelling package with built-in scripting language or custom designed system could be used. *Angela* was purchased as a correctly texture mapped mesh. However, in order to prepare her for exporting, some alterations had to be made.

Depending on the method used for colour modulation for the impostors in the system, two different approaches can be used at this stage to prepare the character's mesh. The first is to group the different coloured regions of the mesh together and use different materials for each region. This method was used by Dobbyn et al. [Dobbyn et al. 2005] for the purpose of easily specifying the different colour regions when generating the impostor images and the mesh can be rendered using the OpenGL fixed function pipeline. The second method is to apply a single texture map image to the character, with different colour regions specified in the texture's alpha channel. This technique was used by [de Heras Ciechomski et al. 2005] for the preparation stage of their characters for a crowd of low resolution geometry. For our clothed crowds, we use this approach in order to add pattern variety (detailed in *Exporting the Impostor*). We will describe both techniques for completeness.

Grouping Triangles
The character's triangles are first organised into groups, where each group is a body part that can be coloured differently from its surrounding parts. For *Angela* we had 8 different groups: eyes, head and shoulders, arms, fingernails, legs, top, shorts, and shoes (Figure 3(a)). Each of these groups was assigned different diffuse materials or texture maps. The reason for organising the triangles in this way is that it improves rendering when exported into the real-time system, due to the minimization of OpenGL states. Also, it allows for colour modulation, as each different group can have a palette of colours assigned to it, in order to create colour variety when duplicating characters in the crowd. The diffuse colour of each material is set to white, and each texture map is greyscaled to allow for colour modulation without the loss of detail (Figure 3(b)).

Specifying Regions in a Single Texture Map
The alternative method is to apply a single texture map to the character. Some models are purchased with a single texture map, but if not, it can be custom designed in an application like Photoshop to suit the character. The basic idea is to combine all of the texture maps into a single compact texture map (Figure 4(left)). The coloured areas with no detail are simply areas that have a single diffuse colour. The alpha channel of this texture is then manually encoded with different colour regions (Figure 4(right)). Since we are using a single pass to render the impostors in the system (see [Dobbyn et al. 2005]), as many regions as the user requires can be specified at this stage. Once this custom designed map has been created, it can be applied to the character. The texture coordinates of the character need then to be matched up to the texture, which can be a time-consuming task. This matching can be accomplished in 3D Studio Max using the *Unwrap UVW* modifier.

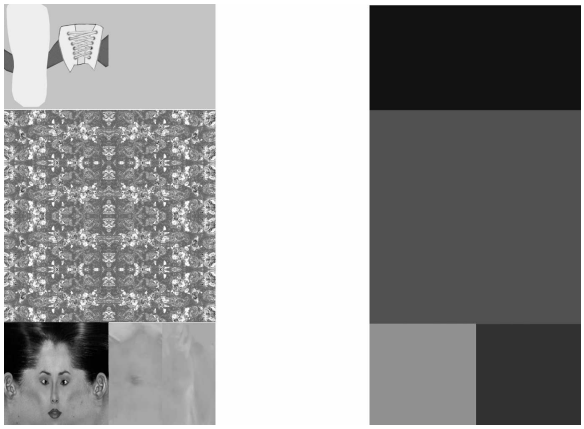

Figure 4: *(left) Example of a single greyscaled texture map for Angela, (right) Corresponding alpha channel.*

Skinning and Animating
The character now needs to be skinned, in order for an animation to be applied. Typically,

a biped skeleton is first fitted to the character (Figure 3(c)), and then the vertices of the character mesh are weighted and linked to the different bones of the skeleton. This can be achieved in 3D Studio Max using the *Physique* or *Skin* modifier. We used both motion capture from our Vicon system and keyframing in 3D Studio Max to obtain the animation for our character. The choice of animation depends on the type of crowd scene that is being produced. The amount of animation data also has to be restricted due to the memory limitations for the display of large crowds. In our system, an animation is typically one second long and then looped in the system to create long sequences. The animations should be cyclical in order that flickering animation artifacts do not occur when looping the animation in the system.

The amount of floor space that can be captured using a motion capture system is often quite limited. Therefore, capturing long sequences of humans walking is difficult, unless a treadmill is used. We can typically capture about 4 steps of a human walk, and do some post-processing in 3D Studio Max in order to make it suitable for use. A candidate cycle is first found in the motion captured walk, e.g., where the left foot touched the ground first to the next time it touches the ground. Although this motion is cyclic, the pose of the character at the first frame is likely to be a bit different to that at the last frame, which would cause artifacts when looped. To avoid this, we paste the pose at the start frame to the end frame and try to interpolate the motion curves to make it fit without loosing too much of the individuality and naturalness of the captured walk.

Care must be taken when capturing motion from actors and applying it to virtual characters. In [McDonnell et al. 2007b], we found that walks captured from male actors applied to female virtual characters are rated as ambiguous, as are female walks on the man. This implies that applying motion captured from actors of the opposite sex to the character will produce confusing or unsatisfactory results in general. Interestingly, synthetic neutral walks (such as those generated using 3D Studio Max's *footsteps* modifier) are considered male when viewed on the man and female when viewed on the woman. This implies that neutral walks may be applied to male and female virtual characters as the appearance of the character takes precedence over the motion in determining the sex of the character.

Preparing the Deformable Clothing

The commercial software that we used ([Clo]) provided tools for constructing garment segments, and for changing the different material parameters, along with accurate collision detection. We simulated many different types of clothing using this software, including silky skirts, stiff dresses, sweaters and shorts. Cloth simulation is still limited in what it can achieve, and therefore the choice of clothing and materials is often compromised depending on the animation of the virtual human. For example, a character performing a back-flip in a tight fitting long dress would not produce nice results - cloth by its nature is resistant to stretching, but not to bending. We found that a lot of tuning was necessary to produce nice cloth animations, for the human animations that we wanted.

Template creation
In [Grob et al. 2003; Keckeisen et al. 2004; Volino et al. 2005] and in most commercial packages, 3D virtual clothes are constructed from planar garment segments (often referred to as 'patterns', however we will not use this term to avoid confusion with the 'pattern' or motif variety detailed later). As this is the way real clothes are manufactured, virtual garments can therefore be designed to match real clothing. It also makes texture mapping particularly easy as correct texture coordinates can be obtained automatically at the planar rest state.

The shape of the segment is usually drawn by the user using splines or imported from a software package designed for cloth manufacturers

like Modaris[1] or PAD[2]. Seams and seam connections are then specified by the user. A seam is usually an edge on a piece of fabric and can be attached to another seam using virtual sewing threads. The planar segments are then positioned around the virtual character at approximately correct positions. A good initial position for the clothing is very important and can determine how well the clothing fits. In order to achieve this, the segments should first of all not collide with anything (the virtual character or each other). They should also be as close as possible to the virtual character's body and the sewing between seams should not penetrate the character. The start pose of the character can be changed in order to satisfy these rules; usually a T-pose is best.

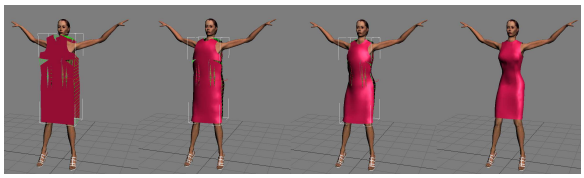

Figure 5: *Fitting a garment using ClothFX.*

Once the segments and sewing are lined up, the garment can be joined along the seam lines. In [Keckeisen et al. 2004] sewing is achieved by merging the segments along the seam lines. For every pair of vertices along the seams, the matching triangles are coupled and the seams are moved halfway between the two original points, as shown in Figure 5. The segment pieces are attached and attain a shape influenced by the form of the body.

Material Modelling

The mass-spring model is the most commonly used technique for deforming garments. It consists of a simple particle system, where the particles correspond to the vertices on a cloth mesh. These particles are connected by different springs: structural springs for tension, diagonal springs for shearing, and interleaving springs for bending. The forces exerted by the particles are determined by the type of spring that connects them. Forces for structural springs are very large, whereas bend and shear forces are small, which can lead to contradicting effects. In order to have a more general system, the mass-spring system can be thought of as a network of interwoven threads, where a thread is a chain of structural springs. Different threads can then interact at mass points. More complex forces are necessary to model this kind of behaviour.

The Kawabata Evaluation System (KES) for fabric is a technique used in the garment industry for measuring fabric mechanical properties through normalised procedures [Kawabata 1980]. In this system, five experiments are conducted using different instruments, and from these experiments 15 curves are obtained, which allow 21 parameters of fabric to be determined. These experiments measure shearing, bending, friction, compression, and deformation. These energies can be modelled in the particle system using a rectangular grid, where each particle interacts with its four direct neighbours.

In commercial software, coefficients that describe these 21 parameters can be changed in order to produce cloths that behave differently. For example, for a material like silk the resistance to bending would be set to low, whereas it would be set to high for a stiffer material like burlap.

Good collision detection is also necessary when modelling clothing. When clothing a virtual human with deformable garments, full and self collision detection need to be implemented in order for the clothing not to penetrate the human or itself. Commercial cloth simulation packages usually have full and self collision built-in. This is the most time-consuming task of cloth simulation, and is a major reason why real-time simulation of clothing is so difficult.

Cyclical Cloth Sequences

In order to prepare the clothing generated for our characters to be exported into the system, the animation had to be cyclical. As previ-

[1] www.lectra.com/en/pds/modaris_fashion.html
[2] www.padsystem.com/

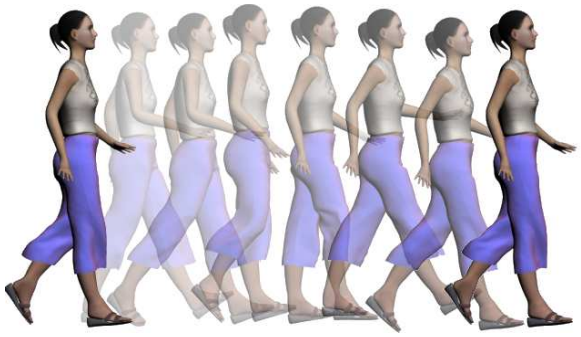

Figure 6: *Example of cyclical cloth. The cloth at the first frame is the same as the cloth at the last frame.*

ously described, in a real-time crowd system, the characters' animations are often cyclical in nature, so that they can be smoothly linked to allow them to move in a fluid manner. Cyclical animations are commonly obtained by manually altering the underlying skeletal motion so that they loop in a realistic looking manner. However, creating looping animations for characters with pre-simulated clothing is a more difficult task, as manual cleanup of the cloth to make it cyclical is very time-consuming, particularly for very deformable items of clothing like skirts, and can result in unrealistic effects.

In [Dobbyn et al. 2006], we describe an automatic method for generating cyclical sequences of cloth simulation. Firstly, a very long animated sequence was created by repeating the animation of the human many times and simulating the cloth in response to the repeating animation. The long cloth sequence needed to be searched using a distance metric that took into account all of the vertices on the cloth mesh between the two frames of animation, in order to find one correctly cyclical loop. This resulted in a two-pass algorithm. The first stage used a distance metric based on mesh vertex comparisons to find candidate cycles, and the second used edge image comparisons on the candidate cycles to produce the final cyclical cloth animation (see Figure 6). This method should be used at this point in the pipeline before proceeding to export the animations.

STAGE 2: EXPORTING THE DATA

At this point, we have pre-simulated the deformation of both the virtual human's skin mesh using linear blend skinning and its cloth mesh using the physical simulator, based on the motion of its underlying skeleton. The cloth and human motion sequences are cyclical and the character should also have its triangles organised into the appropriate groups or has a single texture with specific alpha mapped coloured regions. The character is ready to be exported from 3D Studio Max into files that can be loaded into the crowd system.

Exporting the Geometry

The virtual human was exported in the same way as in [Dobbyn 2006]. A new plug-in was written to export the keyframes of the cloth mesh. This plug-in simply stored the world-space position of each vertex with respect to the root object for each keyframe of animation in an organised manner. By pre-calculating and storing the deformation of the cloth mesh in poses, this avoids the cost of simulating the clothes at run-time. However, this also means that all animations are pre-determined and cannot be changed at run-time, which limits the motion variation in the crowd.

Exporting the Impostor

Generating the impostor representation of our character involves capturing two types of images from a number of viewpoints around the model: a detail map image to capture the detail of the model's diffuse texture, and a normal map image whereby the model's surface normals are encoded as an RGB value. For more information on how these images are generated and rendered in the system see [Dobbyn et al. 2005]. This section will describe any improvements to the algorithms and the necessary changes that were made in order to incorporate deformable cloth.

Generating Images

The normal maps in [Dobbyn et al. 2005] took considerable time to generate, as per-pixel look ups and operations were needed, so we im-

Figure 7: *Mesh after each vertex has been automatically painted with an RGB value corresponding to its normal.*

proved the algorithm using a less computationally intensive technique. A copy of the character's mesh and cloth at the current frame was first needed. Each vertex normal was first converted into eye-space coordinates, to find the normal with respect to the camera, and then converted into an RGB colour (using the equations described in [Dobbyn 2006]). Per-vertex colouring was then used to paint the RGB colours onto the vertices of the copied meshes (3D Studio Max's *VertexPaint* modifier was used to do this). These vertex colours were interpolated over the polygons, creating a character mesh with normal map colours (Figure 7). The normal map image was then generated by rendering an image of this mesh, from the current viewpoint. Per-vertex colouring and interpolating are operations that are performed very quickly, as they are supported by graphics hardware. This meant that the image could be produced almost immediately, without the need for slow per-pixel operations.

Incorrect back-facing polygons was another issue which arose when implementing cloth impostors. The previous impostor plug-in did not account for back-facing polygons as it was not a noticeable problem for skinned clothing, which clung tightly to the body and did not turn inside-out at any point. However, for flowing garments, this occurred quite regularly and resulted in incorrect normals in the impostor image. A simple approach was used to solve this, by testing the direction of the normal and flipping if it was back-facing.

The new technique for adding hardware assisted pattern variety to the impostors that we developed in [Dobbyn et al. 2006] involves a slightly different process for generating impostor images. The detail map image is replaced with a texture coordinate map or *UV map*. This is similar to a normal map just described. However, this time it is the texture coordinates that are converted into an RGB colour and then painted onto the vertices of the mesh. Similar to the normal map images, these images were generated for each viewpoint.

Reducing Texture Memory Consumption
Tecchia et al. [Tecchia et al. 2002] saved on texture memory when generating impostors by exploiting the symmetric nature of the human body performing a walk animation (both the animation and the body are symmetric). They halved the memory needed by mirroring the animation: 32 samples were acquired by sampling the mesh at 16 different points along one side, and mirroring these images to produce the samples on the other side. This worked very well for skinned humans, as the human and the clothing could be made symmetric quite easily (i.e., by mirroring one side), and would remain so for the animation. This approach could not be taken for generating impostors of humans wearing deformable clothing, as the folds in cloth are rarely symmetric in nature and mirroring them would produce artifacts at run-time. We tried generating a mirrored cloth animation using tools in 3D Studio Max, but it involved too much manual tuning and the results were not convincing. Therefore, the human was sampled on both sides to create the cloth impostor images, which increased texture memory consumption, but avoided artifacts.

Ideally, impostors would be generated at very small intervals around a sphere, the same number of times that a polygonal model would be updated, which would allow seamless transitions between the images. However, as we are using pre-generated textures, texture memory

consumption prevents choosing such a dense sampling, so there is a need to pick an optimal number of viewpoint images to generate. In [McDonnell et al. 2006], we performed a psychophysical experiment that determined the optimal impostor update frequency, balancing texture memory consumption with visual fidelity.

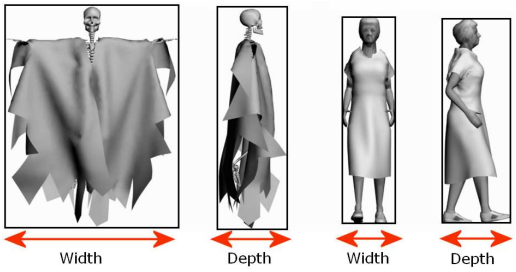

Figure 8: *(left) character with large width to depth ratio, (right) character with small width to depth ratio.*

It was found that the amount of deformation in the clothing did not affect the update rate, where characters with flowing cloth needing the same number of updates as those with stiff clothing. However, we found that the width to depth ratio of the characters did affect the update rate, where characters with large width to depth ratios needed more updates than those with smaller ratios (Figure 8).

For normal pedestrian characters (characters with small width to depth ratios), a viewpoint update rate of 17° is necessary to guarantee with high probability that users will not notice viewpoint changes of the impostors. This corresponds to 21 images that need to be generated at equal spacing around the character. We suggest rounding to the nearest even number of images (22) in order to include the direct front and the direct back images, particularly in applications where a front-on view would be most noticeable. For other characters whose width to depth ratios are large, a viewpoint update rate of 9° is advised. This corresponds to 40 images around the character. The number of impostor images to be generated at this stage in the pipeline should be chosen based on these metrics.

The number of frames of animation that are generated is also a factor to consider when reducing memory consumption. For memory critical systems such as real-time crowds using impostors, the fewer frames required to make an animation appear smooth the better. In [McDonnell et al. 2007a] we found that for a range of different types of animation, no more than 40 frames (or poses) per second was necessary for participants to find the animation smooth. We also found that background impostors could be displayed at 16Hz, with foreground geometry at 30 or 20Hz without observers noticing the difference. This results in the ability to store double the number of characters in memory than would be possible if the impostors were being displayed at 30Hz. Therefore, for every second of animation, 16 impostor frames should be generated.

STAGE 3: CROWD RENDERING

We are finally at the stage where all data is ready to be imported into the crowd system and used to display crowds of clothed characters. A number of alterations needed to be made to the crowd system in order to incorporate our perceptual metrics and the pattern variation technique. In this section, we will first give a broad overview of how the system was written, then we will look at how each of the levels of detail were rendered and how switching was achieved. This work was based on the approaches detailed in [Dobbyn 2006].

Setup

The framework was based on a perceptually driven LOD approach, whereby the viewer's perception is exploited to compute less accurate models when they would not be noticed. The highest level of detail was the mesh model and the lowest was the impostor, as previously described. The system is a C++ application running on the Win32 API, where the OpenGL

rendering library is used as the core component of the rendering subsystem. The number of humans in the crowd is first specified and they are then given random initial positions in the scene. Algorithms can be implemented to make sure that no human intersects, or that they are all oriented in a certain way. Each crowd individual is then randomly allocated a human template model, and variety is added by randomly choosing a specified *outfit*, which is a set of colours for each of the different parts of the body.

Rendering the Geometric Human and Cloth Models

The system is optimised by taking advantage of the fact that crowd individuals perform a default walk animation. Previously, we used static meshes, where the deformation of the mesh is pre-calculated. However, with advancements in graphics hardware we now calculate the deformation of the mesh on the GPU using linear blend skinning. The cloth animation is stored as pre-baked poses directly from 3D Studio Max, as described previously. The rendering speed is improved by using Vertex Buffer Objects (VBOs) to store the key-frames of animation for both human and cloth poses. The idea behind VBOs is to provide buffers (regions of VRAM) accessible through identifiers. A buffer is made active through binding (in a similar manner to display list binding). This allows graphics drivers to optimise internal memory management and also to choose the most suitable type of memory to store the buffers. At run-time, the correct VBO pose is selected and rendered, depending on the current frame of animation of the virtual human.

Adding colour variation to the mesh involves first creating different outfits, which specify the different colours to be used for each different region in the mesh. For example, an outfit for Angela would consist of: eyes coloured blue, head and shoulders coloured pale pink, arms pale pink, fingernails red, legs pale pink, top white, shorts blue, and shoes white. Many different outfits can be specified for each different template human as described in [Dobbyn 2006].

As mentioned before, there are two techniques for setting up the virtual humans' colours. We will now describe the two corresponding techniques to colour the human at run-time.

The first technique grouped together the triangles of the different regions of the mesh and then tagged them with IDs. At run-time, the OpenGL fixed function pipeline is used for rendering the mesh, where the diffuse colour of each mesh region is changed depending on its material ID and corresponding outfit colour. This was the technique used in [Dobbyn et al. 2005] and is suitable for meshes that will be used alongside detail mapped impostors.

The second technique is needed to match the geometry with the UV mapped impostors. This method should be used if texture variation is important to add variety, as in the case of adding different designs to clothing. This technique uses programmable graphics hardware to add variation. As described previously, a single texture map was applied to the character, where the alpha channel of the texture specified the different colour regions. The shading sequence for the geometry is shown in Figure 9.

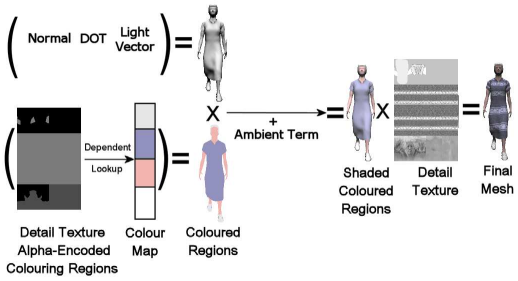

Figure 9: *Geometric mesh shading sequence.*

Rendering the Impostor

In order to render the impostor, the correct viewpoint image needs to be found. Using the camera position and the virtual humans position and direction, the most suitable viewpoint image can be calculated and retrieved from the large texture containing the pre-generated

viewpoint images. This image is then mapped onto a quadrilateral. To dynamically orientate the quadrilateral towards the viewer, the amount to rotate can be calculated using the camera and virtual humans position. See [Dobbyn et al. 2005] for detailed descriptions of the equations.

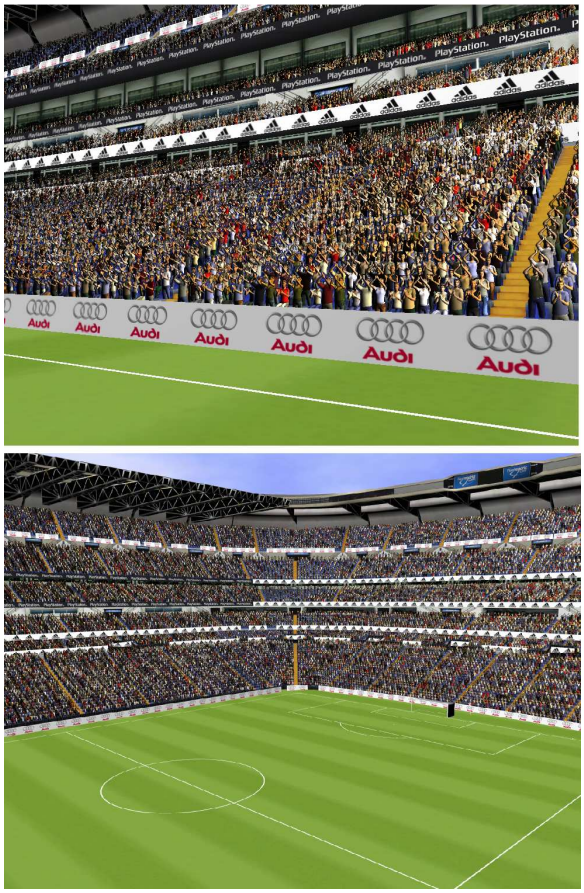

Figure 10: *Examples of our stadium crowd.*

In the case of crowds that do not move within the virtual environment, such as those found in sports games, viewpoint selection and the orienting of the billboard for each character can be done on the vertex processor. Billboards of individuals can be batched together in a single vertex buffer object and rendered in a single draw call. For more details see [Millán and Rudomín 2006]. It should be noted that smaller numbers of animation frames will result in bigger batch sizes, since only individuals using the same impostor texture can be batched together. Additionally, since the camera is typically limited to the playing field in sports games, pregenerating viewpoints from behind the character is not necessary, thus reducing the amount of texture memory consumed by the impostors (Figure 10).

Switching Between Representations

Ulicny et al. [Ulicny et al. 2004] noted that impostors were an efficient approach for rendering crowds of far-away humans, but that their pixellated appearance when displayed close to the viewer prevented them from being used for detailed crowds. Dobbyn et al [Dobbyn et al. 2005] solved this problem by using a hybrid impostor/geometry system, similar to that described in this paper (Figure 11). In crowd scenes where the humans and the camera are moving, the distance from impostors to viewer will change throughout the viewing time. Thus, a switching mechanism was employed to allow impostors to switch to geometry and vice versa when they reach a certain distance.

In Hamill et al. [Hamill et al. 2005], we found a perceptual metric for the optimal pixel to texel ratio at which to switch representations, which ensures that users will not notice the switch. This ratio was one-to-one, which means that geometry can be switched to impostor representation when one of its texels is displayed on one pixel on the screen. A seamless transition is achieved by matching the pose of the impostor to the geometry at the point of switching. This technique did not need to be altered for our clothed characters, as the poses of the cloth were inherently linked to the human poses.

In [McDonnell et al. 2006], we validated this metric by performing a system experiment which examined participant's perception of level of detail representations in different sized crowds. We found that participants did not notice the difference between a hybrid crowd of impostor/geometry and a crowd of all high resolution geometry, when the camera was zooming in and out of the crowd, which meant that popping between representations was not noticed.

Figure 11: *(top) Crowd Scene, (bottom) Crowd scene displayed in wireframe, the impostors are displayed as green quadrilaterals.*

DISCUSSION

In this paper, we provided a useful step by step approach to realising crowds of clothed characters incorporating perceptual principles. This pipeline should be of use to game developers, to create and/or improve the quality of their crowds.

Our technique of adding pre-simulated clothing improved the realism of our crowd system. The visual quality of the clothed crowd is maintained by creating cyclical cloth motions to avoid discontinuous motion artifacts. Additionally, the use of the impostor's UV map complements the use of impostors with its mesh representation in a LOD crowd system, since it allows the matching of texture and colour variation between the cloth and skin. In Figure 1 we show our virtual model of Dublin city inhabited by crowds of pedestrian characters wearing deformable clothing. All of the walk cycles were captured using a motion capture system, and pattern variety was added using our UV mapping technique.

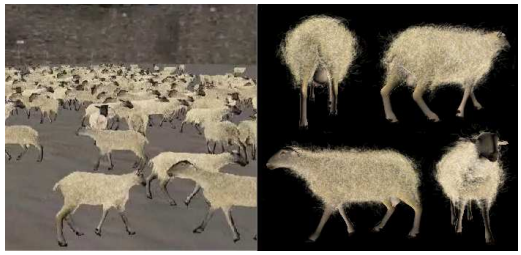

Figure 12: *Sheep impostors*

In [Skrba et al. 2006], we extended the techniques described in this paper to produce animal impostors, by adding highly realistic pre-simulated fur to sheep and dogs. Figure 12 shows a screen shot from this system. The pipeline could also be used to create realistic characters with pre-simulated hair, by replacing the cloth creation stage with hair simulation.

The main limitation at present is that all of the animations are pre-baked and do not allow for animation variety. Currently our impostors are only generated for a single walk cycle animation, and are also restricted to "pre-baked" poses at 10 frames per second. This is because an image must be created for each viewpoint of the human for each frame of the animation, which represents a significant memory overhead. As we wish to imperceptibly switch from geometry to impostor, and vice versa, we are therefore currently restricted to the simple walk cycle for the high-detail models in our crowds also. Therefore, while high visual fidelity of appearance can be achieved, the variety of motion in our crowds is restricted. To improve upon this, we would like to investigate efficient animation database formats for storing character motion.

BIOGRAPHY

Rachel McDonnell is a postdoctoral researcher at the Graphics, Vision and Visualisation Group in Trinity College Dublin where she recently finished her PhD entitled "Realistic Crowd Animation: A Perceptual Approach". Her research interests include perceptually adaptive graphics, real-time crowd simulation, virtual human animation and cloth simulation.

References

Aubel, A., Boulic, R., and Thalmann, D. 1998. Animated impostors for real-time display of numerous virtual humans. In *Proceedings of the First International Conference on Virtual Worlds*, 14–28.

Breen, D. E., House, D. H., and Wozny, M. J. 1994. Predicting the drape of woven cloth using interacting particles. In *Proceedings of ACM SIGGRAPH*, 365–372.

Carignan, M., Yang, Y., Thalmann, N. M., and Thalmann, D. 1992. Dressing animated synthetic actors with complex deformable clothes. *Computer Graphics 26*, 2, 99–104.

Clarke, J. H. 1976. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM 19*, 10, 547–554.

ClothFX, cloth simulation software. Size8Software, 2004.

Cordier, F., and Magnenat-Thalmann, N. 2005. A data-driven approach for real-time clothes simulation. *Computer Graphics Forum (Eurographics 2005) 24*, 2, 173–183.

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)*, 9–17.

de Heras Ciechomski, P., Schertenleib, S., Maïm, J., and Thalmann, D. 2005. Reviving the roman odeon of aphrodisias: Dynamic animation and variety control of crowds in virtual heritage. *VSMM*, 601–610.

Dobbyn, S., Hamill, J., O'Conor, K., and O'Sullivan, C. 2005. Geopostors: a real-time geometry / impostor crowd rendering system. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 95–102.

Dobbyn, S., McDonnell, R., Kavan, L., Collins, S., and O'Sullivan, C. 2006. Clothing the masses: Real-time clothed crowds with variation. In *Eurographics Short Papers*, 103–106.

Dobbyn, S. 2006. *Hybrid Representations and Perceptual Metrics for Scalable Human Simulation*. PhD thesis, University of Dublin, Trinity College.

Funkhouser, T. A., and Squin, C. H. 1993. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of ACM SIGGRAPH*, 247–254.

Grob, C., Fuhrmann, A., and Luckas, V. 2003. Automatic pre-positioning of virtual clothing. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics*, 99–108.

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.

Hamill, J., McDonnell, R., Dobbyn, S., and O'Sullivan, C. 2005. Perceptual evaluation of impostor representations for virtual humans and buildings. *Computer Graphics Forum 24*, 3, 623–633.

Kawabata, S. 1980. The standardization and

analysis of hand evaluation. *The Textile Machinery Society of Japan*.

Keckeisen, M., Feurer, M., and Wacker, M. 2004. Tailor tools for interactive design of clothing in virtual environments. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, 182–185.

McDonnell, R., Dobbyn, S., and O'Sullivan, C. 2005. LOD human representations: A comparative study. *Proceedings of the First International Workshop on Crowd Simulation*, 101–115.

McDonnell, R., Dobbyn, S., Collins, S., and O'Sullivan, C. 2006. Perceptual evaluation of LOD clothing for virtual humans. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 117–126.

McDonnell, R., Jörg, S., Hodgins, J. K., Newell, F., and O'Sullivan, C. 2007. Virtual shapers & movers: Form and motion affect sex perception. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Applied Perception in Graphics and Visualisation (to appear)*.

McDonnell, R., Newell, F., and O'Sullivan, C. 2007. Smooth movers: Perceptually guided human motion simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (to appear)*.

Millán, E., and Rudomín, I. 2006. Impostors and pseudo-instancing for gpu crowd rendering. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, 49–55.

Pettre, J., de Heras Ciechomski, P., Maïm, J., Yersin, B., Laumond, J.-P., and Thalmann, D. 2006. Real-time navigating crowds: Scalable simulation and rendering. *Computer Animation and Virtual World (CAVW) Journal - CASA 2006 special issue 17*, 445–455.

Provot, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, 147–154.

Skrba, L., Dobbyn, S., McDonnell, R., and O'Sullivan, C. 2006. Animating dolly: Real-time herding and rendering of sheep. In *Proceedings of Eurographics Ireland Workshop*, 7–12.

Tecchia, F., Loscos, C., and Chrysanthou, Y. 2002. Visualizing crowds in real-time. *Computer Graphics Forum (Eurographics 2002) 21*, 4, 753–765.

Ulicny, B., deHeras Ciechomski, P., and Thalmann, D. 2004. Crowdbrush: interactive authoring of real-time crowd scenes. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 243–252.

Vassilev, T., Spanlang, B., and Chrysanthou, Y. 2001. Fast cloth animation on walking avatars. *Computer Graphics Forum 20*, 3, 260–267.

Volino, P., Cordier, F., and Magnenat-Thalmann, N. 2005. From early virtual garment simulation to interactive fashion design. *Computer-Aided Design 37*, 6, 598–608.

Yang, Y., and Magnenat-Thalmann, N. 1993. An improved algorithm for collision detection in cloth animation with human body. In *Proceedings of the First Pacific Conference on Computer Graphics and Applications*, 237–251.

Yersin, B., Maim, J., de Heras Ciechomski, P., Schertenleib, A., and Thalmann, D. 2005. Steering a virtual crowd based on a semantically augmented navigation graph. In *Proceedings of the First International Workshop on Crowd Simulation*, 169–178.