A VIRTUAL MANNEQUIN SERVICE USING THE FIGMENT SCHEME

James N. Anderson and Mervyn A. Jack{{bios}}
The University of Edinburgh
Edinburgh, Scotland

Editor: Hal Thwaites

ABSTRACT

Until recently, the possibility of a "virtual mannequin" service for telepresence shopping systems has been unrealistic due to the number and complexity of calculations required for the modelling of physical clothing items. In this paper, the authors present the FIGMENT scheme (Fast Implementation Garment Modelling environmENT) which involves a four point strategy (a simplified physical model, collision volume approximation, progressive meshes and a hybrid rendering algorithm) to reduce the quantity and complexity of the computations involved, bringing simulation times from the realm of hours to seconds while maintaining an acceptable level of accuracy and quality in the results.

1. Introduction

The Teleshoppe project [18] is concerned with the application of advanced multimedia and virtual reality technologies in order to simulate the 'touch and feel' of physical shopping, thus enhancing the usability of the user interface for a telepresence shopping experience. Part of this research includes the development of a 'virtual mannequin' - a computer- generated, animated representation of the consumer exhibiting the same body dimensions and appearance, which performs tasks within the virtual environment on his or her behalf [2]. The primary action to be performed by the mannequin is that of modelling clothes, helping users to gain an impression of how particular items of clothing (in different colors, sizes and styles) might look on their own 'person'.

Modelling virtual clothing is a nontrivial procedure due to the number and complexity of the calculations necessary to obtain accurate results. Previous attempts to simulate the physical behavior and appearance of cloth have performed the calculations on a long term basis. However, such an approach to modelling clothing would be inappropriate for an interactive shopping service; any usable service which allows consumers to view different items and combinations of clothing during a shopping session needs to provide results in a considerably shorter space of time. A radical shift in perspective is required to reconcile these demands.

The necessary compromise has been made within the project by trading accuracy, or level of detail, against speed of computation. Reasoning that high quality photographic detail can already be provided by two-dimensional images and movies, the emphasis within the virtual mannequin service is that of context based visualization. The users, rather than being restricted to a static and impersonal view, will be able to see clothing items in the context of (1) bodies with their own physical dimensions, (2) their own skin color, hair style, etc., (3) different garment sizes, colors and styles, (4) other clothing items and (5) different environmental conditions, e.g. lighting. The advantages afforded by such technology will act as a useful, and perhaps indispensable, supplement to noninteractive two-dimensional media. By using reduced complexity (more discrete) models of clothing items in combination with simplified and optimized algorithms to calculate the dynamics of the cloth, a dramatic reduction in simulation time can be achieved. For example, a female mannequin can be clothed with a 1560 polygon dress in less than 100 seconds using a 200MHz Pentium platform.

This paper provides an overview of the FIGMENT scheme (Fast Implementation Garment Modelling environmENT) at the heart of the virtual mannequin service. The FIGMENT scheme incorporates a four point approach towards optimizing the computations and reducing the overall simulation times while maintaining an acceptable level of accuracy and quality in the rendered results. Firstly, a simplified physical model is used to represent the dynamic interactions between discrete sections of cloth and their surrounding environment. Secondly, collision volume approximation methods are used to represent the solid surface of the mannequin. Thirdly, all or part of the clothing item models are replaced with so- called progressive mesh representations, speeding up the early stages of the simulation. Finally, a hybrid rendering algorithm is used to ensure that layers of clothing can be rendered without appearing to penetrate one another or the surface of the mannequin.

2. FIGMENT Physical Model

A number of different methods have been developed for simulating the behavior of cloth [4,5,6,23,25]; the algorithms used within the FIGMENT scheme are based on those employed by MIRALab at the University of Geneva [6,25]. The clothing items are defined as meshes of triangular sections of material; the smaller the sizes of the sections, the more accurate the simulation, but the longer is takes to compute. The physical behavior (e.g. draping, folding) of the cloth meshes is parameterized using standard measurements of material properties: thickness, density, Young's modulus and Poisson's coefficient. In this way, different kinds of fabric can be easily simulated without having to create arbitrary and potentially meaningless constants. During each iteration of the simulation, the physical forces on each section of cloth are calculated according to the current position of the cloth and the appropriate change in cloth dynamics is applied.

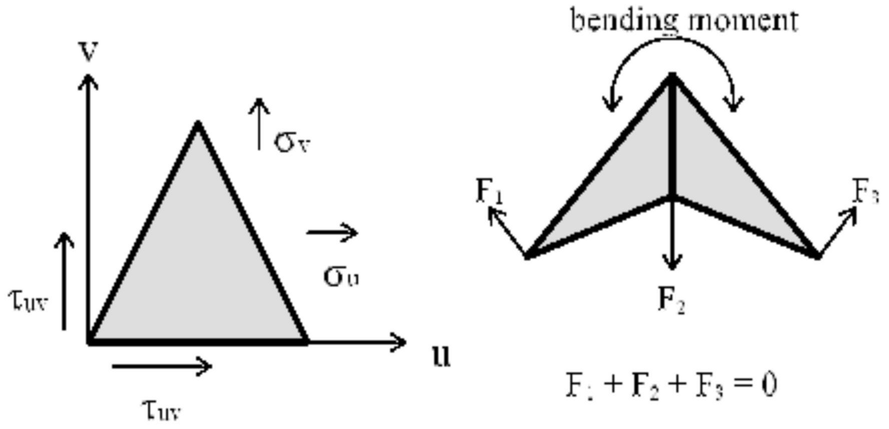


Figure 2.1: Elastic stress components and bending forces on cloth sections.

Two types of elastic forces are computed for each section of cloth (Figure 2.1). Firstly, the internal stresses of stretching (σ_u and σ_v) and shearing (τ_{uv}) are calculated according to the distortion of the section from its unstressed dimensions. From these components, the resultant forces acting on each corner of the section can be computed. Secondly, the bending forces occurring at the joints between sections are calculated according to the approximated curvature of the cloth at each joint. For each type of force, only first order physical effects are considered, since second order dynamics (such as viscoelasticity) contribute little to the final position of the cloth as it is draped over the mannequin body.

Other forces acting on the cloth sections include gravity, friction with the surface of the mannequin, and air resistance. Frictional forces are computed for each node within the cloth mesh when it comes into contact with the surface of the mannequin. The force and direction with which the cloth collides, and a specified frictional coefficient, allow the opposing force to be calculated. The simple model of air resistance computes an opposing force for each node of the cloth mesh proportional to the velocity of that node.

A simple wind effect has also been implemented within the model which can aid the user in gaining a better impression of the type of material used in clothing items (e.g. the difference between lighter or heavier cotton blouses). Effective results are produced by applying a sinusoidally varying force of specified direction, amplitude, and frequency to the sections of the cloth while changing the phase of the sinusoid at random moments (Figure 2.2). The force applied to each section is proportional to the orientation of that section with respect to the wind direction; the effects of wind 'shadowing' are not taken into account.

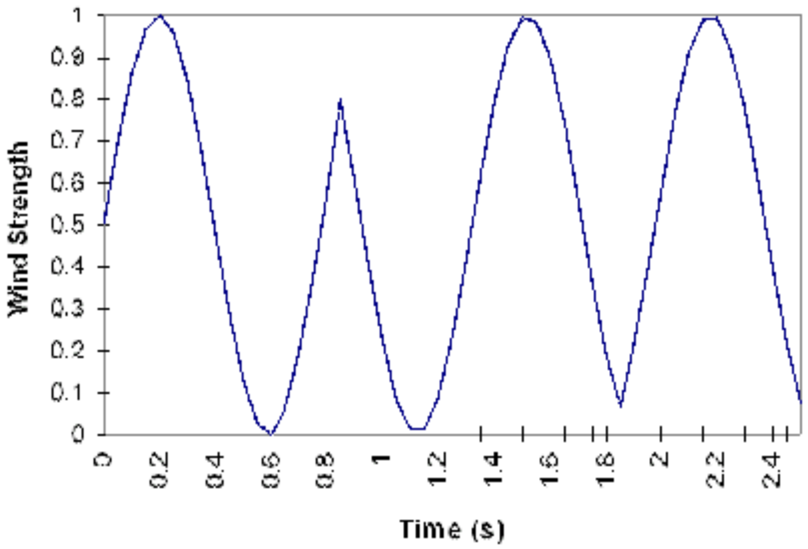


Figure 2.2: Example of wind variation.

The physical model consists of the dynamic forces described above acting on the individual cloth sections. At the end of each iteration, when the net force on each node of the cloth mesh has been computed, an improved Euler method is used to approximate the change in velocity and position for time $t+\delta t$ (where δt is the virtual time interval between iterations). In order to reduce the overall computation time required to model the dynamic behavior of the clothing items, two options exist: either to reduce the resolution of the cloth meshes (i.e. increase the size of the discrete sections) or to increase the virtual time interval between iterations. The former can only be done to a certain extent before the quality of the simulation is degraded beyond the point where it is usable as a realistic modeling service. The latter approach, although tending to degrade the accuracy of the simulation, is favorable. However, larger values of δt can introduce instability into the physical model; large forces can appear within and between the discrete sections of cloth leading to oscillations with catastrophic results.

Two techniques have been implemented within the FIGMENT scheme, therefore, to allow larger time intervals to be used for simulations. The excessive internal forces are predominantly due to temporary distortions in the cloth mesh which incur high magnitude stretching/shearing forces across individual cloth sections. The values of elastic strain for each section computed at the beginning of each iteration can therefore be clamped in order to keep them within a realistic range. Then, at a higher level, a limiting function is applied to the net acceleration computed for each node of the mesh at the end of the iteration. Two such limiting functions have been considered (Figure 2.3). The first is 'smooth' (based on the hyperbolic tangent function) but more computationally demanding; the second function simply 'clamps' excessive magnitude to a specified maximum value. In practice, there has been found to be no discernible difference in the results when using either function, and so the latter is used in preference.

The effect of these instability-countering measures have been analyzed by running comparisons between simulations with small δ_t (hence, most accurate) and simulations with large values and using the above limiting functions. The results have indicated that the measures successfully avoid instability in faster simulations without producing any visible difference in the results.

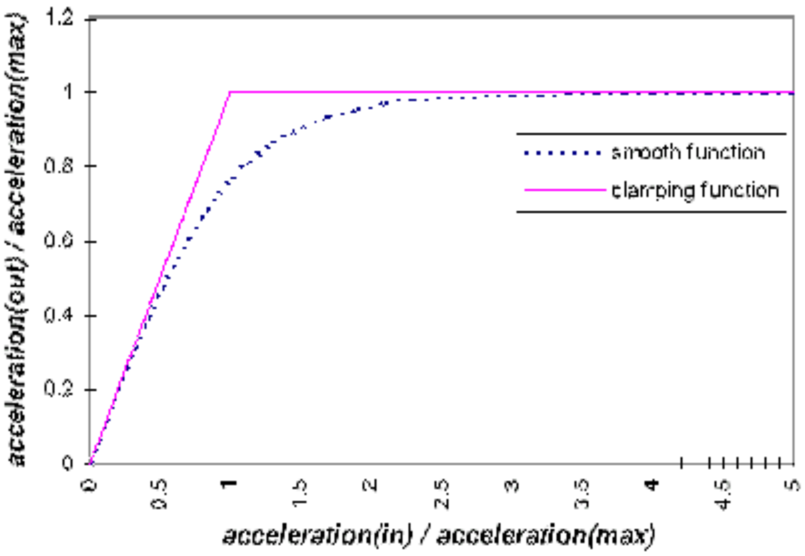


Figure 2.3: Limiting functions.

3. Collision Volume Approximation

Two categories of collision handling arise within cloth simulation: collisions between the cloth and the mannequin, and cloth self collision. Although fast algorithms for self collision detection have been successfully implemented in long term cloth animation [26], due to the high number of calculations required and the complex nature of such detection algorithms this aspect of the physical modeling is not incorporated within the FIGMENT scheme, which instead relies on the natural stiffness of the cloth to avoid serious self penetration problems. This approach has proved adequate for the purposes of this application due to the use of a hybrid rendering algorithm (see below) which ensures that intersecting layers or sections of cloth are still rendered correctly. Collision detection between the cloth and the mannequin would normally have to be performed on a facet-to-facet basis, accounting for over 90% of computation time in many cases. Although fast and efficient algorithms have also been developed for this purpose [12,19,20,27], a radical change of approach has been required to bring this time consuming portion of the computation as near to the realtime realm as possible. Hence, the methods described below were devised and implemented, with favorable results. Each method is one of collision volume approximation: appropriate parts of the overall mannequin model are approximated by geometrical objects, the surfaces of which are described by a number of parameters. These objects by their nature allow (1) simple determination of whether a vertex has penetrated the surface of the object, (2) computation of a corrected position for the vertex on the surface and (3) estimation of the normal vector at that location. A combination of analytic and iterative methods are used in each case to determine the parameters for the object which best represent the shape of the original part (with respect to the modeling of clothing).

3.1 CAPSULE APPROXIMATION METHOD

While initially trying to formulate a fast and effective way of handling collisions with the mannequin, it was noted that the hierarchically jointed mannequin models were comprised of geometrically similar objects where each object could be approximated by a cylindrical shape, of varying dimensions, with more or less rounded ends (Figure 3.1). The forearm, for example, can be represented by a long shape with less rounded ends, the neck by a shorter, fatter shape, and the hips and waist by a wider, more rounded shape. The calculations required to detect the penetration of such a geometrically simple volume are considerable less than for any arbitrary polyhedron. Similarly, the calculations employed to correct and respond to collisions are straightforward. It is worth comparing Hubbard's sphere approximation method in these respects [11,12,13,14], although there are significant differences. In the case of FIGMENT, only one structure is tested for penetration, i.e. no progressive refinement is performed, and the collision response is computed with respect to the approximation structure itself.

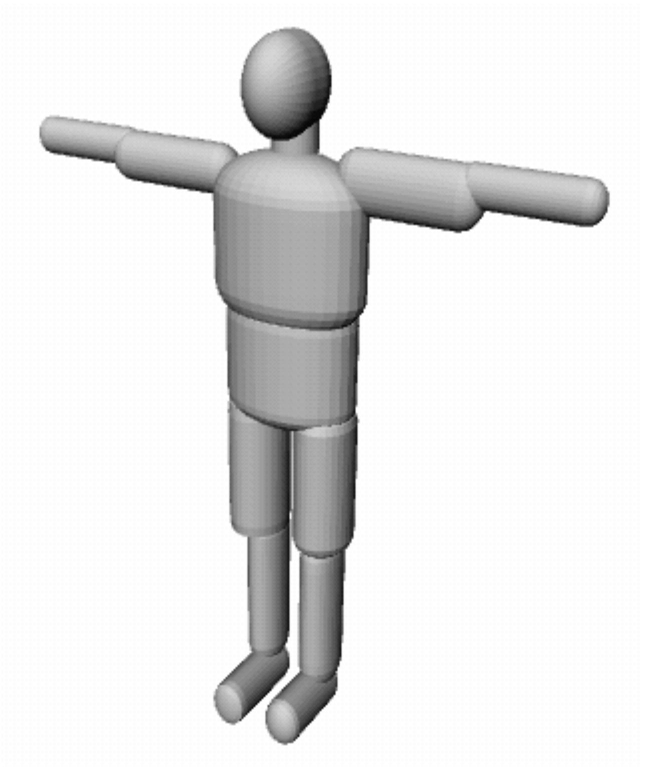


Figure 3.1: Representation of mannequin using simple geometric shapes.

This collision approximation structure is referred to here as a 'capsule' object, although it has developed to a more complex form since its initial conception in order to provide a more accurate representation of the body parts of the mannequin. The originally conceived capsule object was defined by only one parameter r, defining the degree of rounding at the ends of the object which was then rescaled to fit the dimensions of the body part. The present form of the capsule structure, however, is defined by the following 12 parameters:

- r1 and r2, separate rounding factors for each end of the capsule (Figure 3.2)
- tx and tz, which define tapering factors in the x- and z-dimensions (the capsule being aligned vertically), where a positive value indicates a widening at the top of the capsule (Figure 3.3)
- x, y, z, the dimensions of the capsule (of unit value by default)
- ox, oy, oz, the offset of the center point of the capsule (of zero value by default)
- Θ, the angle of tilt of the capsule (Figure 3.4)
- Φ, the angle of rotation (around the y-axis) of the tilting axis (0 degrees is x-axis, 90 degrees is z-axis, Figure 3.4)
- Further details regarding the progressive development of the capsule approach used in FIGMENT, and the introduction of each set of parameters, are provided elsewhere [3].

Solving the problem of finding the best fitting capsule object for each body part using a purely analytic method would be unrealistic due to the relatively high number of parameters and the complex relationships between them.

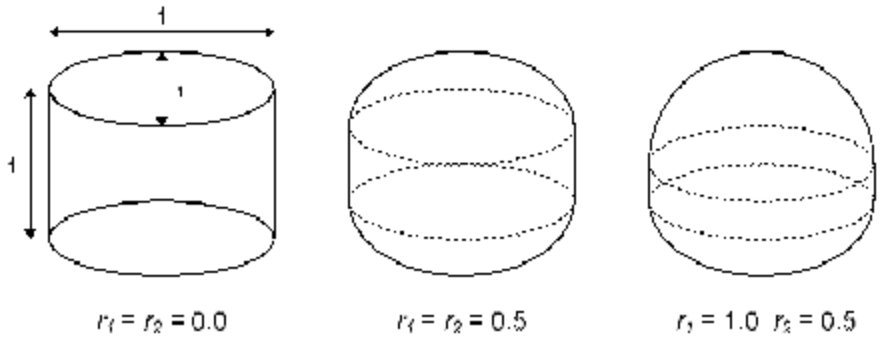


Figure 3.2: Variation of parameters r1 and r2 in unit capsule.

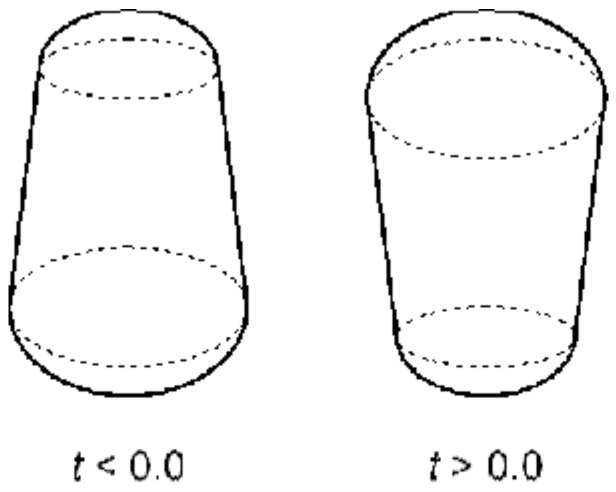


Figure 3.3: Tapering effect of parameters tx and tz.

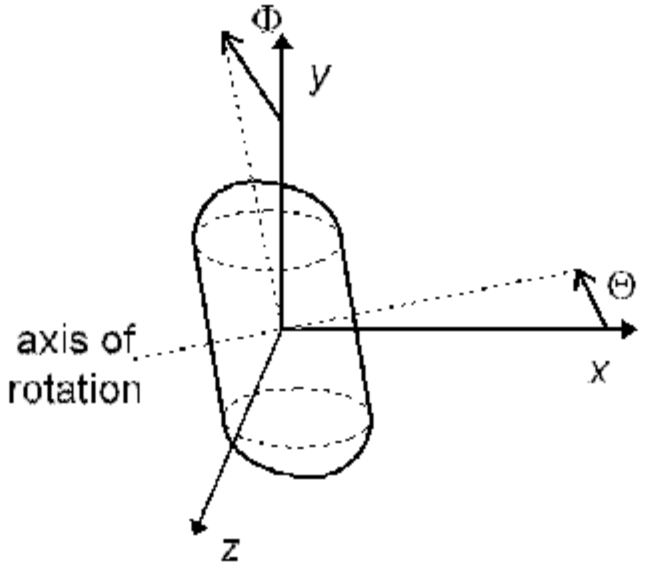


Figure 3.4: Tilting of capsule according to parameters Θ and Φ.

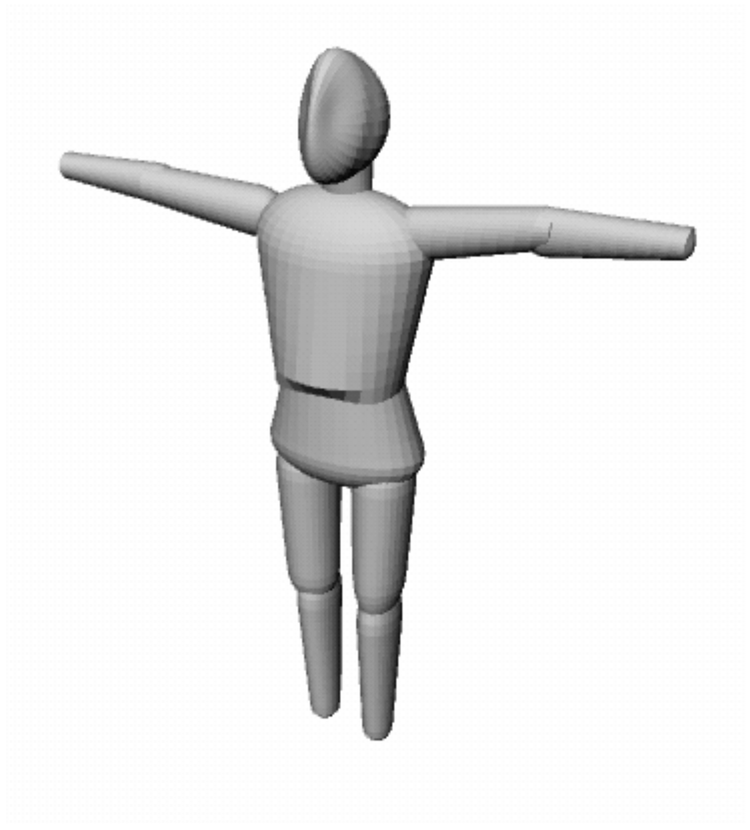


Figure 3.5: Representation of male mannequin.

Instead, a form of genetic optimization algorithm has been found to be the best iterative method for the task. Genetic algorithms lend themselves particularly well to this type of problem [8], and the type of algorithm adopted is an adaptation of that used by Louchet [16,17] for a different application. Full details of the algorithm have been published previously [3] but a brief summary follows. A population of identical capsules is created with default parameters. Each iteration then proceeds in four stages:

1. 1. A cost function is evaluated for each capsule to indicate the extent of discrepancy between the surfaces of the capsule and the corresponding body part. In practice, this function is biased so that capsules tend to enclose the original surface rather than not; the draping of cloth is more determined by convexities than concavities in the surface of the mannequin model.
2. 2. The members of the population (capsules) are sorted into order of increasing cost.
3. 3. A mutation stage occurs during which a certain proportion of the bottom (higher cost) members of the population are replaced with 'mutated' copies of those above; one randomly chosen parameter of each copy is varied randomly (according to an approximately normal distribution) within specified limits.
4. 4. A crossover stage occurs in which a certain proportion of the members above those replaced during the mutation stage is replaced with 'crossed over' versions of those above; each parameter being taken from a parent member randomly chosen from the top portion of the sorted population. The selection of parent members in stages 3 and 4 is biased towards those with lower cost.

The results of using the best fitting algorithm for a typical male mannequin are shown in Figure 3.5.

The greatest advantage of the capsule approximation method is the low number of calculations required to detect and respond to vertex collisions, the exact details of which are described elsewhere [3]. Briefly, it can be observed that all but the rounding and tapering parameters correspond to simple geometric transformations and can thus be incorporated into the overall transformation applied to the unit capsule. Checking for vertex penetration proceeds by applying the inverse transformation and then removing the effect of taping on the x and z coordinates. Once a bounding box test has been applied, the y-coordinate of the transformed vertex is examined to determine whether the cylindrical section or one of the rounded (hemispherical) sections should be considered. Detecting the penetration of a cylinder or hemisphere by a vertex, translating it to the surface, and calculating the normal vector at that point, is then straightforward.

3.2 RADIAL DEPTH APPROXIMATION METHOD

The second of the two collision methods employed by the FIGMENT scheme provides a considerably more accurate representation of the surface of the body parts. However, although the calculations required for handling collisions are still relatively simple and fast, computation generally requires over twice as much time as for the first method. The choice between these methods for cloth modeling depends therefore on the power of the rendering platform, the level of detail required, and even the shape and significance of individual body parts within the model.

The formulation of the radial depth approximation method relies on similar observations regarding the three-dimensional shapes being approximated; in this case, that the objects are generally enclosed surfaces, not unduly concave, which allow clear cross sections to be taken along one or more lateral axes (for example, this axis would be approximately vertical for a lower leg part). If a suitable axis is chosen, then a series of cross sections of the surface of the object may be obtained at regular, specified intervals. These cross sections can be represented as functions of 'radial depth' on a polar coordinate system; by translating these functions to Cartesian coordinates, it can be seen that their periodic nature lends itself to harmonic (e.g. Fourier) analysis and thus to approximation at varying resolutions.

In this way, each body part can be reduced to a series of harmonic coefficients which can be used to reconstruct an approximation of the shape of the original object. The geometric structure of this approximation allows relatively straightforward calculations to determine firstly whether a vertex (i.e. a node of the cloth mesh) has penetrated its surface and, if so, the nearest position on the surface (for collision correction) and the surface normal vector at that point (for collision response). As with the capsule approximation method, the total calculations required for collision detection and response are entirely unaffected by the complexity of the original model, e.g. the number of polygons used.

A combination of analytic and iterative (genetic) techniques are employed in order to find the best fitting volume for each body part. An example of this approximation method applied to a mannequin model is shown in Figure 3.6.

3.3 ISSUES

It will be observed that not all parts of the human anatomy lend themselves well to the capsule approximation method. In practice, however, this can be overcome by splitting the relevant parts (e.g. feet, female chest) into more appropriate shapes and using multiple capsule approximations. In contrast, the radial depth approximation method copes effectively with all parts of mannequins without further division.

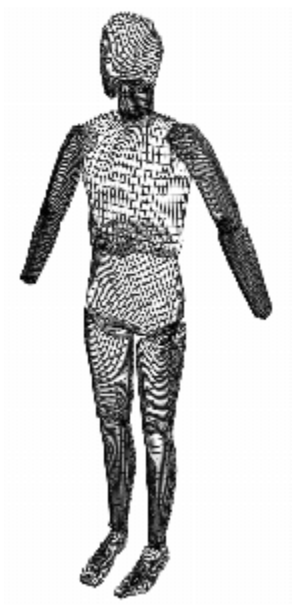


Figure 3.6: Male mannequin represented by radial depth approximation objects.

Since the two methods provide approximations of the surface of the mannequin, a particular approach has been adopted within the FIGMENT scheme toward dealing with protrusions of the model from its collision volume. The solution taken has been to force the full enclosure of the vertices of the body parts by the corresponding approximations. This is appropriate since the 'skin' of the mannequin is generally obscured by the clothing and, in any case, the shape of the mannequin is provided by the collision volume and thus the 'fitting' of the clothing is not distorted by adjustments to the surface of the actual model.

It will be noted that the objects used for a collision volume must overlap to some extent. The algorithm adopted to handle the cases of vertices which penetrate two or more objects during physical simulation is detailed elsewhere [3].

The two principle, and considerable, advantages of using these volume approximation methods are as follows. Firstly, the geometric structures of the collision objects require relatively few calculations for collision detection and response. Secondly, deep and discontinuous penetrations of the collision volume are handled equally well as those occurring continuously near the surface (in contrast to a vertex-to-facet collision detection method). In practice, this means that the separate meshes of clothing items do not have to be placed at a distance from the mannequin and 'seamed' together around it. Rather, the entire garment may be placed roughly over the model and the penetrating vertices of the meshes will be corrected ('pushed out') in the first few iterations of simulation.

In addition, since the collision objects correspond directly with the jointed body parts of the mannequin, the application of geometric transformations to animate the mannequin and to adjust its dimensions and proportions is quite straightforward.

4. Progressive Meshes

The time required to perform the physical simulation of a mannequin wearing an item of clothing will be proportional to the complexity of the clothing item, e.g. the number of polygons in its mesh. As stated previously, it is possible to reduce the complexity of highly faceted models with a permissible loss of accuracy, but at a certain point the increased discreteness of the cloth surface will result in unacceptable effects. These include collision detection discrepancies (the cloth appears to penetrate the mannequin) and loss of fidelity with respect to the nature of the cloth (the apparent folding, stretching and continuity of the fabric). At this point the mannequin no longer functions as a usable representation of reality.

It is only in the final stages of simulations, however, as the cloth comes to rest around the mannequin, that the perceived level of detail is important. In the initial period of simulation, the predominant dynamics of the mesh are those of gross collision correction and free fall as the cloth assumes its general position, rather than the more precise dynamics of folding and creasing as it comes to rest. Hence, a lower complexity model is acceptable during this initial period which may be progressively increased in complexity until the desired level of detail is achieved.

A considerable number of mesh simplification techniques have been developed in the past. A specific approach might involve fitting smooth surfaces to the polygon mesh [15] and then resampling to obtain a homogeneous mesh of the desired complexity, a retiling technique [24], or possibly multi-resolution analysis [7]. However, since the cloth simulation of FIGMENT is performed by an iterative approximation of the dynamics of the discrete sections of the fabric, and because there is no straightforward relationship between the individual facets of progressively more complex meshes in the above methods, they are unsuitable for this particular application.

Other mesh decimation methods can be used which require only slight topological changes between successive complexities of meshes, usually proceeding by collapsing edges [1,9,10,21,22] within the mesh. The differences between the various algorithms are with regard to choosing which edges to remove in order to maintain, as closely as possible, the appearance and structure of the original mesh. The method described by Hoppe [10] is particularly relevant to this application because of its emphasis on continuity and progression in the decimation process, and for this reason the FIGMENT scheme has adopted a modification of that particular algorithm.

4.1 MESH DECIMATION ALGORITHM

The FIGMENT mesh decimating algorithm, as with Hoppe [10], is based on an energy minimization principle. The method successively collapses edges within the mesh into a single vertex (Figure 4.1), in each case attempting to choose the transformation which will leave the resultant mesh as close to the original mesh as possible, with respect to their three-dimensional surfaces.

The algorithm initially defines two sets of sample points, X and X'; the first are taken from the overall surface of the original mesh, the latter from the perimeter (nonshared) edges of the mesh. Then, during each iteration of the decimation process, the algorithm considers in turn every edge of the mesh for a possible collapse transformation. Three possible collapses are considered: the two cases whereby the edge is collapsed into either of its end vertices, plus the case whereby the edge is collapsed into a vertex at its center point. An energy function E is computed for each case, and the transformation is performed which will produce the resultant mesh with the minimal energy value. The energy function is formulated to take into account the deviation of the resultant surface from the sample point set X, the deviation of the resultant perimeter from the sample point set X', and the extent to which the collapse of the particular edge in question tends to reduce the overall homogeneity of the mesh. Without this latter consideration, the decimation of a typical cloth mesh would result in large flat areas consisting of a few large sections in contrast to high number of small sections remaining at points of extreme curvature or creasing. Such a mesh is less suitable for the purposes of even and realistic cloth modeling.

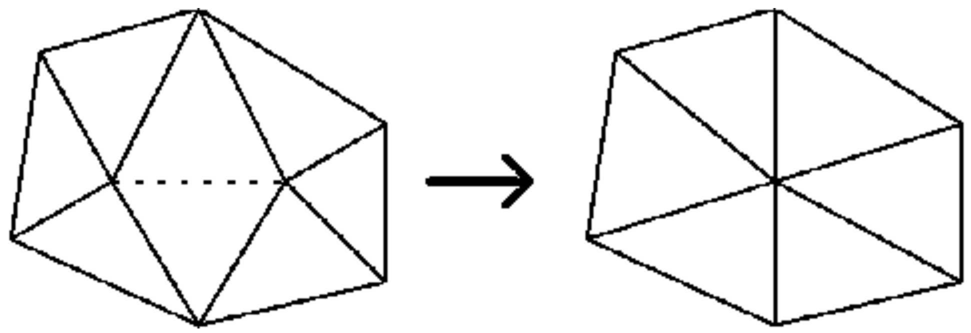


Figure 4.1: Edge collapse transformation.

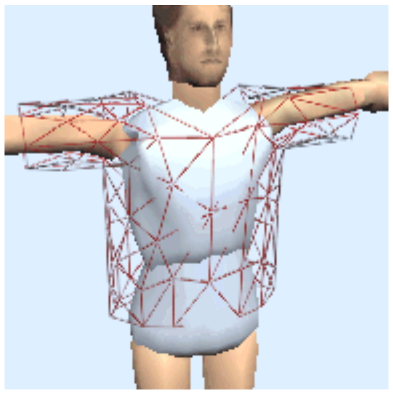
4.2 MODELLING WITH PROGRESSIVE MESHES

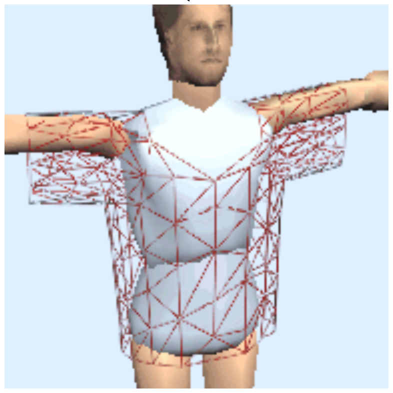
Once the optimal series of edge collapse transformations has been determined, further preprocessing is required before using the progressive meshes for modelling in order to specify exactly how the original cloth mesh may be reconstructed during the physical simulation.

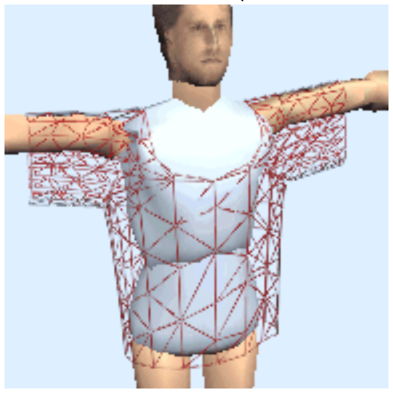
The reverse transformation to an edge collapse corresponds to a vertex split operation, and certain information is required (e.g. which vertex to split, the positions of the new vertices, etc.) in order to perform this transformation on a progressive mesh at each stage of reconstruction. For a progressive mesh used in a cloth modelling simulation, however, this process is more complex than that required for a static three-dimensional model. Firstly, since the mesh is in a state of dynamic deformation, the reconstruction algorithm must estimate suitable initial positions for any newly created vertices, rather than simply being provided with absolute coordinates. Secondly, the algorithm must not only reconstruct the geometry of the visible clothing item, but also information regarding the 'at rest', i.e. unstressed, state of the item. This is required so that the current elastic strain on each triangular cloth section can be computed.

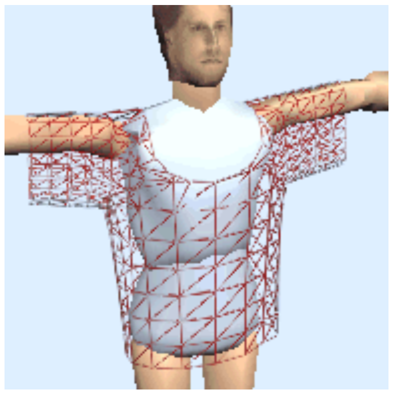
Taking these additional matters into account, therefore, the preprocessing software of the FIGMENT scheme takes the original clothing item model, along with the series of optimum edge collapse transformations computed by the decimation software, and outputs a 'mesh reconstruction script'. This script is then used by the main cloth modelling software to progressively increase the complexity of the initial (decimated) mesh, one vertex split operation at a time, until the full detail of the original is reached (Figure 4.2). It is possible, of course, to begin or end the simulation at intermediate states of reconstruction and to vary the rate at which the progression occurs, depending on the accuracy required of the results.

The advantages of this aspect of the modelling scheme must be evaluated in terms of the overall time reduction achieved for simulations and the inaccuracy incurred. As a typical example, the progressive mesh modelling of a 1000 polygon T-shirt item can be speeded up by over 40% (compared to nonprogressive modelling) while only introducing a final (at rest) mean deviation for its vertices of 0.033% (0.0005 units) of its vertical dimension of the item (1.51 units).


Initial state of mesh (159 sections, 320 ips)


State after 1000 iterations (345 sections, 148 ips)


State after 2000 (520 sections, 96 ips)



State after 3000 iterations (703 sections, 71 ips)

Figure 4.2: Example of a progressive mesh item, showing the development of individual cloth sections. (ips=iterations per second)

5. Hybrid Rendering Algorithm

The majority of realtime graphics libraries use a depth buffering method to correctly render facets which obscure others to some extent from view. The FIGMENT scheme is implemented using the Silicon Graphics Open Inventor 3D graphics library, which in turn uses the OpenGL rendering library [28]. The depth buffering method of OpenGL can cause problems, however, when rendering a clothed mannequin since the body of the mannequin can appear to protrude through the fabric when sections of cloth come close to, or slightly penetrate, the skin surface. The same problem occurs when using multiple layers of clothing, particularly in the absence of cloth-to-cloth collision detection.

The solution to this problem in FIGMENT has been to implement a hybrid rendering algorithm, based on depth buffering and depth sorting methods, by extending the Open Inventor library. In Open Inventor, virtual scenes are implemented using hierarchically arranged nodes which specify objects and environmental factors in the scene (Figure 5.1). During each rendering pass, child nodes are traversed from left to right and prior to the remaining siblings of the parent. The FIGMENT scheme extends the library with three custom nodes which allow correct rendering of modelled clothing.

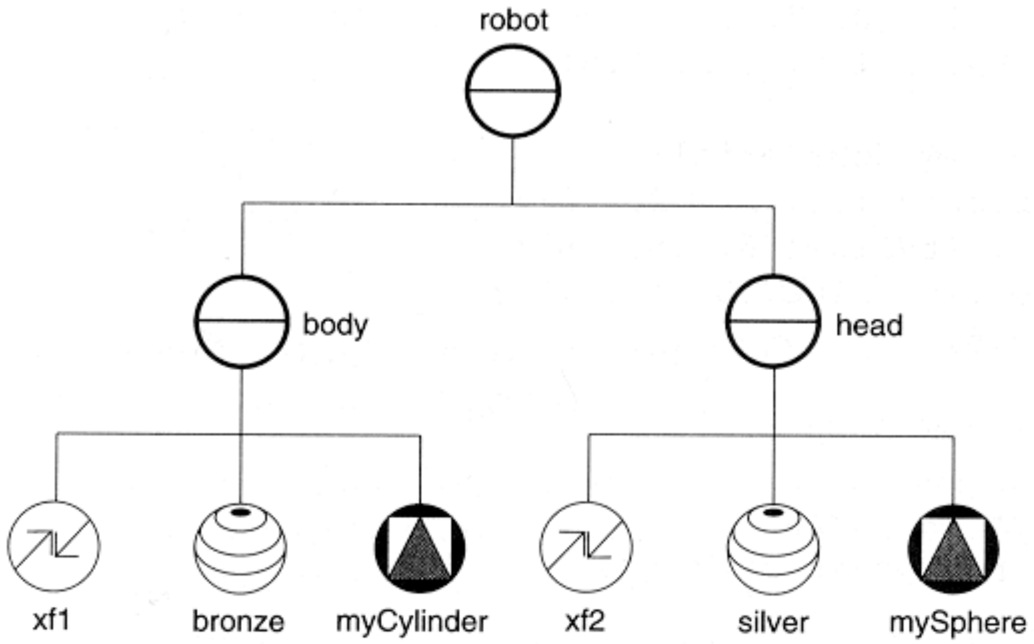


Figure 5.1: Example of an Open Inventor scene graph. (Source: The Inventor Mentor)

The first, Clothing Layer, replaces the Indexed Face Set node of the library which defines a 3D faceted surface. The definition of the surface remains the same, allowing straightforward substitution of nodes, but before rendering each facet of the set, the normal vector of the facet is examined to determine whether the facet faces substantially towards, or away from, the viewpoint. Those facets facing away from the viewpoint are rendering as normal using the depth buffer, but those facing towards are assumed to be enclosing all objects which have been previously rendered along that line of view and so are drawn entirely without reference to previous depth buffer values (i.e. obscuring all coinciding pixels) and replacing those values within the buffer. In addition, all facets in the set are depth sorted according to their center points to ensure that obscuring folds within a single cloth mesh are correctly rendered. Figure 5.2 illustrates the improvement in rendering. With depth buffering alone areas of the body are visible (e.g. at the shoulders and knees) and the clothing items appear to intersect; the modified method avoids these problems.

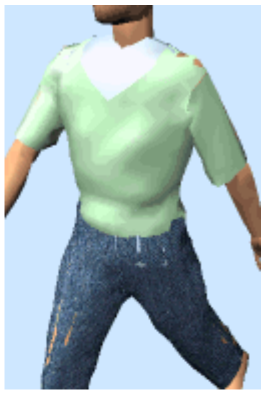


Figure 5.2a: Mannequin rendered with depth buffering alone.

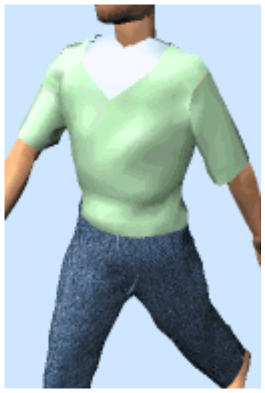


Figure 5.2b: Mannequin rendered with modified rendering method.

A problem arises when using this node, however, if an item of clothing is comprised of two or more 'seamed' cloth meshes. Those meshes which appear 'later' in the scene graph will always be rendered such as to obscure those which have been rendered already. Therefore, facets of one mesh (e.g. the sleeve of a shirt) can appear 'in front' of those of another mesh (e.g. the body of the shirt) which is directly between the former mesh and the viewpoint. A second custom node has been implemented to overcome this problem. The Depth Sorted Group node extends the standard Group node of the library but rather than traversing its children from left-to-right during rendering, the node computes the distances of its children from the viewpoint and then traverses in decreasing order of magnitude so that those meshes appearing 'behind' others are rendered first. Since the distance computation is dynamic, the entire scene can be rotated while still being rendered correctly with respect to the changing viewpoint.

There is a further issue which must be overcome due to the fact that the nodes comprising the mannequin model occur earlier in the scene traversal than those of the clothing items. For certain poses of the mannequin, parts of the body can be obscured by areas of cloth which do not enclose them. For example, a forearm which appears to the viewer to be in front and outside of an upper body clothing item can be drawn over when rendering the item (Figure 5.3a). A third node has been implemented, therefore, which helps remove this problem by making use of OpenGL's stencil buffer. The stencil buffer can be written into during rendering and the values in it used to mask out subsequent graphics drawing. The node, Stencil Separator, derived from the Inventor Separator group node with one additional boolean field, simply determines whether objects below it will be drawn into the stencil buffer or not during a scene rendering traversal. In addition, an extra boolean field has been added to the Clothing Layer node definition to determine whether or not its facets should be masked by the stencil buffer. A stencil mask can thus be created during rendering from those body parts which should not be obscured and then used to prevent those facets from clothing meshes which ignore the depth buffer from drawing over them (Figure 5.3b).

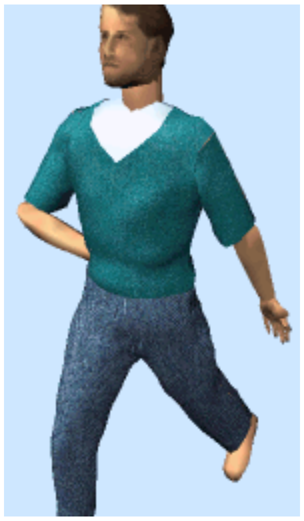


Figure 5.3a: Clothed mannequin without stencil buffer implementation.

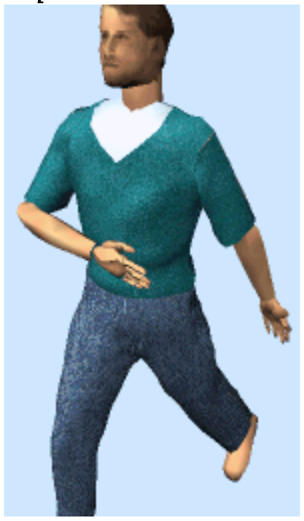


Figure 5.3b: Clothed mannequin with stencil buffer implementation.

If the viewpoint is changed then the combination of the body parts needing to define the stencil mask and the cloth meshes which should be masked must be recomputed to maintain a correct rendering. This involves simply setting the boolean fields in the appropriate Clothing Layer and Stencil Separator nodes.

6. Results

Figures 6.1, 6.2, and 6.3 illustrate the results of using the FIGMENT scheme for three different scenarios of clothing modeling. The clothing items themselves have been simply created using a 3D CAD package. In each case, the capsule method of collision approximation has been used with nonprogressive meshes. The timings indicated below refer to the duration of the simulation on a 233 MHz Pentium platform. See also the video clips on the CDROM:
demo1.avi or demo1.rm
demo2a.avi or demo2a.rm
demo2b.avi or demo2b.rm
demo3.avi or demo3.rm

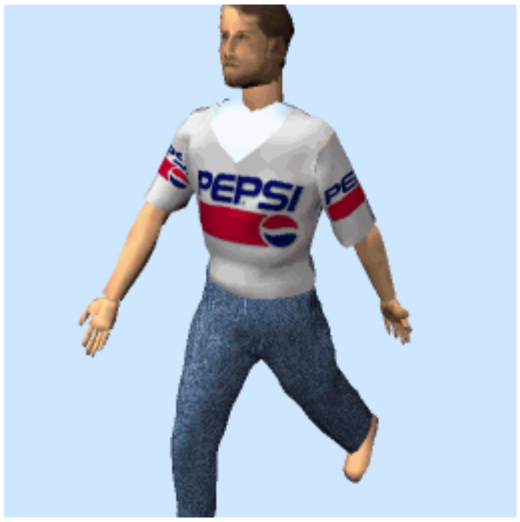


Figure 6.1: Example of a clothed mannequin. (165 seconds)

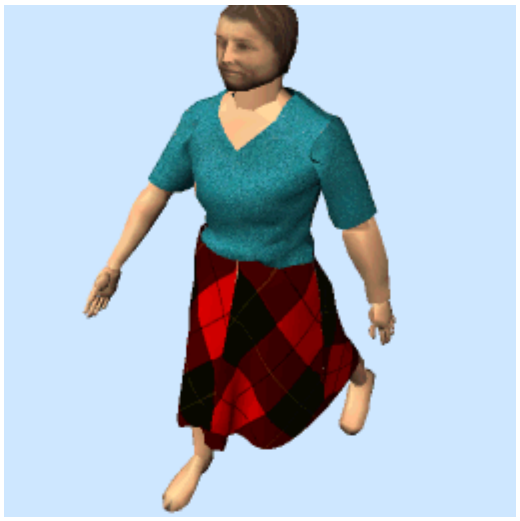


Figure 6.2: Example of a clothed mannequin. (145 seconds)

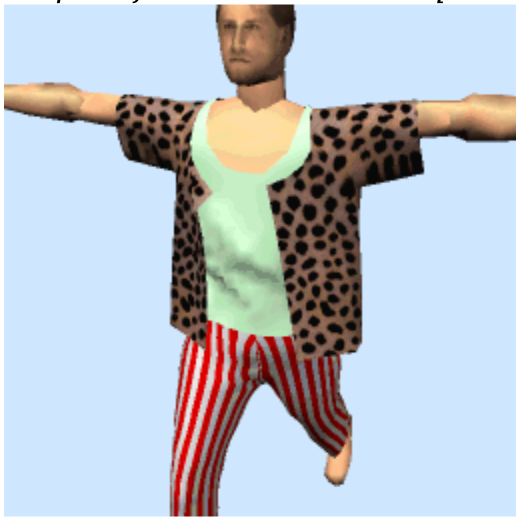


Figure 6.3: Example of a mannequin with multiple layers of clothing items (210 seconds). (145 seconds)

7. Conclusions

This paper has presented an overview of the FIGMENT scheme, an integrated collection of algorithms and techniques to enable the fast modeling of clothing items on a virtual mannequin such that a useful and usable service may be maintained without sacrificing the quality and accuracy of its results.

The authors are planning to perform extensive usability trials in the future with a fully operational modeling service based on real catalogue items, and ultimately hope to see such a system incorporated into a commercial online shopping service.

8. Acknowledgements

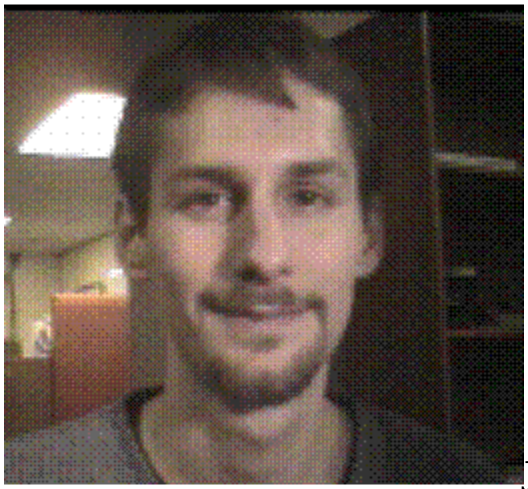
This work was funded as part of the Teleshoppe project, supported as part of the European Commission's ACTS Program (Project AC099).

REFERENCES

[1] M. Algorri, F. Schmitt, "Mesh Simplification", Computer Graphics Forum (proc. EUROGRAPHICS'96), August 1996.
[2] J. Anderson, M. Jack, "Dressing the Virtual Teleshoppe User", Proceedings of the European Conference on Multimedia Applications, Services and Techniques (ECMAST'96), pp. 293-303, May 1996.
[3] J. Anderson, M. Jack, "A Fast Collision Detection Technique for Modelling Virtual Clothing", Proceedings of the Eurographics UK Chapter 15th Annual Conference, pp. 75-88, March 1997.
[4] D. E. Breen, D. H. House, M. J. Wozny, "Predicting the Drape of Woven Cloth using Interacting Particles", Computer Graphics (proc. SIGGRAPH'94), pp. 365-372, August 1994.
[5] D. E. Breen, D. H. House, M. J. Wozny, "A Particle-Based Model for Simulation the Draping Behavior of Woven Cloth", Textile Research Journal, Vol. 64, No. 11, pp. 663-685, November 1994.
[6] M. Carignan, Y. Yang, N. Magnenat Thalmann, D. Thalmann, "Dressing Animated Synthetic Actors with Complex Deformable Clothes", Computer Graphics (proc. SIGGRAPH'92), pp. 99-104, August 1992.
[7] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes", Computer Graphics (proc. SIGGRAPH'95), pp. 173-182, August 1995.
[8] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, 1989.
[9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, "Mesh Optimization", Computer Graphics (proc. SIGGRAPH'93), pp. 19-26, August 1993.
[10] H. Hoppe, "Progressive Meshes", Computer Graphics (proc. SIGGRAPH'96), pp. 99-108, August 1996.
[11] P. Hubbard, "Real-Time Collision Detection and Time-Critical Computing", Workshop on Simulation and Interaction in Virtual Environments, July 1995.
[12] P. Hubbard, "Interactive Collision Detection", Proceedings of the 1993 IEEE Symposium on Research Frontiers in Virtual Reality, pp. 24-31, October 1993.
[13] P. Hubbard, "Collision Detection for Interactive Graphics Applications", IEEE Transactions on Visualization and Computer Graphics, pp. 218-230, September 1995.
[14] P. Hubbard, "Approximating Polyhedra with Spheres for Time-Critical Collision Detection", ACM Transactions on Graphics, Vol. 15, No. 3, July

1996.
[15] V. Krishnamurthy, M. Levoy, "Fitting Smooth Surfaces to Dense Polygon Meshes", Computer Graphics (proc. SIGGRAPH'96), pp. 313-324, August 1996.
[16] J. Louchet, "An Evolutionary Algorithm for Physical Motion Analysis", Proceedings of the British Machine Vision Conference, September 1994.
[17] J. Louchet, X. Provot, D. Crochemore, "Evolutionary Identification of Cloth Animation Models", Eurographics Workshop on Animation and Simulation, pp. 44-54, September 1995.
[18] I. McKay, M. Jack, L. Parker, J. Andrews, O. Villemaud, J-P. Lefevre, "Teleshoppe - Usable Teleshopping with Multimedia and Virtual Reality", Proceedings of the European Conference on Multimedia Applications, Services and Techniques (ECMAST'96), pp. 149-165, May 1996.
[19] M. Moore, J. Wilhelms, "Collision Detection and Response for Computer Animation", Computer Graphics (proc. SIGGRAPH'88), pp. 289-297, August 1988.
[20] M. K. Ponamgi, D. Manocha, M. C. Lin, "Incremental Algorithms for Collision Detection Between Solid Models", IEEE Transactions on Visualization and Computer Graphics, 1997, to appear.
[21] R. Ronfard, J. Rossignac, "Full-Range Approximation of Triangulated Polyhedra", Computer Graphics Forum (proc. EUROGRAPHICS'96), pp. 67-76, August 1996.
[22] W. Schroeder, J. Zarge, W. Lorensen, "Decimation of Triangle Meshes", Computer Graphics (proc. SIGGRAPH'92), pp. 65-70, August 1992.
[23] D. Terzopoulos, K. Fleischer, "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture", Computer Graphics (proc. SIGGRAPH'88), pp. 269-278, August 1988.
[24] G. Turk, "Re-tiling of Polygonal Surfaces", Computer Graphics (proc. SIGGRAPH'92), pp. 55-64, July 1992.
[25] P. Volino, M. Courchesne, N. Magnenat Thalmann, "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects", Computer Graphics (proc. SIGGRAPH'95), pp. 137-144, August 1995.
[26] P. Volino, N. Magnenat Thalmann, "Efficient Self-Collision Detection on Smoothly Discretised Surface Animations using Geometrical Shape Regularity", Computer Graphics Forum (proc. Eurographics), pp. 155-166, August 1994.
[27] B. Von Herzen, A. H. Barr, H. R. Zatz, "Geometric Collisions for Time-Dependent Parametric Surfaces", Computer Graphics (proc. SIGGRAPH'90), pp. 39-48, August 1990.
[28] J. Wernecke, "The Inventor Mentor", (Addison-Wesley), 1994.

BIOGRAPHIES

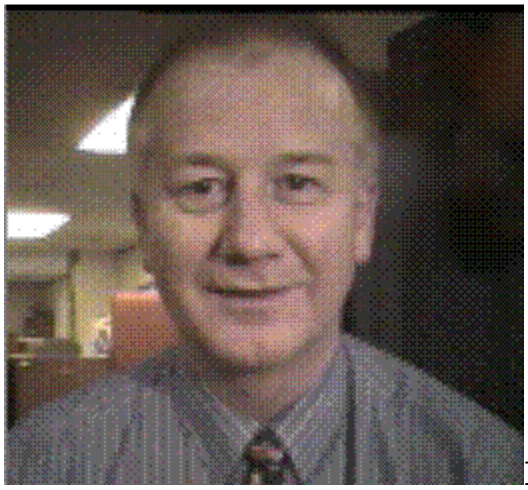


James Anderson

James Anderson is a Research Associate at the Centre for Communication Interface Research, University of Edinburgh. A graduate of the University, he is currently writing his Ph.D. thesis which concerns the implementation and usability of a 'virtual mannequin' system. His research interests include virtual consumer applications, modelling virtual clothing, and implementing shared-space environments with VRML and Java.

Contact information:

James Anderson
Centre for Communication Interface Research
University of Edinburgh
80 South Bridge, Edinburgh, EH1 1HN
SCOTLAND
Phone: (0131) 650 8230
Fax: (0131) 650 2784
Email: jad@ccir.ed.ac.uk
Web: www.ccir.ed.ac.uk/~jad



Mervyn Jack

Professor Mervyn Jack is Professor of Electronic Systems at the University of Edinburgh and was Director of the Centre for Speech Technology Research from 1989 until 1994. Since March 1994 he has been Director of the new Centre for Communication Interface Research (CCIR) at the University. His research interests include work on usability engineering and spoken language engineering, particularly for Internet and automated telephone services where he is leader of the UK project DIALOGUES 2000 to establish industry standards for the user interface in automated telephony. He has experience of working with a range of companies including BT, BSkyB, GUS, Videotron, Royal Bank of Scotland, Italtel, Barclays Bank and MFI on problems of user interface design.

Professor Jack has wide experience in European collaborative projects. He leads a team of 20 researchers and 5 Ph.D. students at the CCIR. An author of over 200 scientific papers and 5 textbooks, Professor Jack is a Fellow of the Institution of Electrical Engineers and a Fellow of the Institute of Acoustics.

Contact information:

Mervyn A. Jack, Director
Centre for Communication Interface Research
University of Edinburgh
80 South Bridge, Edinburgh, EH1 1HN
SCOTLAND
Phone: (0131) 650 2785
Fax: (0131) 650 2784
Email: maj@ccir.ed.ac.uk

{{Return}}

Beginning Previous Next Contents