

93 sh.

A THREE-DIMENSIONAL HUMAN BODY MODELING SYSTEM

119629

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
In Partial Fulfillment of the Requirements for
the Degree of Master of Science in Computer Engineering,
Computer Engineering Program**

**T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**

**by
Derya PAKALIN**

**July, 2002
İZMİR**

119629

Ms.Sc. THESIS EXAMINATION RESULT FORM

We certify that we have read the thesis, entitled “**A THREE-DIMENSIONAL HUMAN BODY MODELING SYSTEM**” completed by **DERYA PAKALIN** under supervision of **PROF. DR. ALP KUT** and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Prof. Dr. Alp KUT
Supervisor

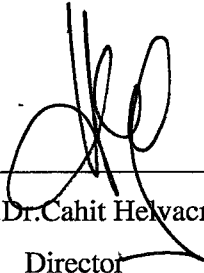


Committee Member
Doç. Dr. Yalçın ÇEBİ



Committee Member
Prof. Dr. Ahmet KAŞLI

Approved by the
Graduate School of Natural and Applied Sciences



Prof. Dr. Cahit Helyacı
Director

ACKNOWLEDGMENTS

I am grateful to my thesis supervisor, Prof. Dr. Alp KUT, who has contributed his time and energy to improve this thesis despite his busy schedule. Thank you for your encouragement, direction, and insight.

I am deeply indebted to my friend, Mustafa KASAP, who shares his knowledge by me. He has supported me in hundreds of ways through the development. I would like to thank Tanzer ONURGİL for his valuable discussions, suggestions and helps. He suggested correct explanations for my observations and made accurate predictions about the system.

Lastly, I want to express lots of thanks to my husband, Kökten Ulaş BİRANT, for his contributions and supports. He has an original view of the world and is great to laugh and joke with.

Derya PAKALIN

ABSTRACT

The focal point of this thesis is to study on human body modeling systems. It is related about the construction of complete, three-dimensional representations of the normal male and female human bodies in the computer environment. In this study, a human body modeling system has been developed to define 3D models of the whole human body. The system reads a new type file HBM (Human Body Model) as input and uses triangular meshes to generate a parametric surface model to describe the human body surface. In computer graphics, triangular meshes are a very standard way of representing 3D surfaces sets of any object. The system also shows a set of conceptual parameters to allow the users to modify the body size of the models in a controllable way. A significant design issue of such a system is to find a reasonable compromise between visual realism, ease of control and rendering speed. For reasons of the inherent difficulty of accurately modeling an entity as complex as the human body, the system requires improved computer graphics and modeling techniques, complex algorithms, faster machines and 3D scanners.

ÖZET

Bu tezin temel noktası üç boyutlu insan vücudu modelleme sistemleri üzerine çalışmaktır. Normal kadın ve erkek insan vücutlarının tümüyle ve üç boyutlu olarak bilgisayar ortamında oluşturulmasıyla ilgilidir. Bu çalışmada, tüm insan vücudunu üç boyutlu olarak tanımlamaya yarayan bir insan vücudu modelleme sistemi geliştirilmiştir. Sistem girdi olarak HBM (Human Body Model – İnsan Vücudu Modeli) adlı yeni bir dosya tipini okumakta ve insan vücudunun yüzeyini tanımlamak için üçgen ağ yapısından oluşan parametrik bir yüzey oluşturmaktadır. Üçgen ağ yapısı, bilgisayar grafikleri alanında herhangi bir nesnenin üç boyutlu olarak yüzeyini betimlemede kullanılan standart bir yöntemdir. Sistem ayrıca kullanıcılara, modellerin vücut ölçülerini kontrollü bir şekilde değiştirmeleri için parametreler sunmaktadır. Böyle bir sistemin tasarımında önemli olan görsel gerçekçilik, kontrol kolaylığı ve sunum hızı arasında kabul edilebilir bir uzlaşım sağlamaktır. İnsan vücudu gibi kompleks bir nesnenin doğru bir şekilde modellenmesinin zorluğundan dolayı, sistem gelişmiş bilgisayar grafikleri ve modelleme teknikleri, kompleks algoritmalar, hızlı bilgisayarlar ve 3D tarayıcılar gerektirmektedir.

CONTENTS

	Page
Contents	VI
List of Tables	IX
List of Figures	X

Chapter One

INTRODUCTION

1.1 What Is Human Body Modeling?	2
1.2 Critical Design Issues.....	3
1.3 The Steps of the Research.....	3
1.4 Application Areas.....	5
1.4.1 Medical Applications	5
1.4.2 Clothing Industry	5
1.4.3 Media and Entertainment	6
1.4.4 Visual Conferencing.....	7
1.5 Thesis Organization	7

Chapter Two

PREVIOUS WORKS

2.1 Related Works.....	8
------------------------	---

2.2	Existing Techniques for Human Body Modeling	10
2.2.1	Geometric Modeling	10
2.2.2	Mesh Based Modeling	11
2.2.3	Human Body Modeling from Video Sequences	12
2.2.4	Spherical Harmonic Modeling	14

Chapter Three

MODELING APPROACH

3.1	Modeling Technique	16
3.2	Data Structure.....	18
3.3	Basic Features for the Triangles.....	20
3.4	Generating 3D Meshes from Data File	24
3.4.1	HBM File Format.....	28
3.4.2	DXF File Format.....	31
3.5	Modification of the Body Size of the Human Model by Using Parameters	38

Chapter Four

PRESENTATION OF COMPUTERIZED HUMAN MODELS

4.1	Two Dimensional (2D) vs. Three Dimensional (3D) Presentation.....	43
4.2	Body Presentation Criteria	44
4.3	Computer Graphic Technique.....	49
4.4	Transformation Operations	50
4.5	Constructed OpenGL Scene Component	52
4.5.1	The Events of the OGLPnl Component	55
4.5.2	The Methods of the OGLPnl Component	56
4.5.3	Creating Components in Delphi.....	62

Chapter Five
DESCRIPTION OF THE PROGRAM

5.1	Programming Environment.....	64
5.1.1	Using OpenGL with Delphi	64
5.2	Program Capabilities.....	65

Chapter Six
CONCLUSIONS & FUTURE WORKS

6.1	Summary	74
6.2	Future Works.....	77
References		78

LIST OF TABLES

Table 3.1 Comparison results of HBM and DXF files according to the file sizes.....	27
Table 3.2 Comparison results of HBM and DXF files according to the file loads	27
Table 3.3 Basic measurements for normal woman body sizes according to the results of the English researches (Taylor, Shoben, 1984)	40
Table 5.1 Normal body size values for women	72



LIST OF FIGURES

Figure 1.1 The steps of the research.....	4
Figure 2.1 Geometric model of the upper human body (Ertürk, 1999)	10
Figure 2.2 Geometric model of the human hand (Rehg, Kanade, 1994)	11
Figure 2.3 Quadrangular mesh (on the left), triangular mesh (on the right) (Buxton et al., 2000)	12
Figure 2.4 Four frames of a video sequence (D'apuzzo et al., 1999).....	13
Figure 2.5 Arm models of frames 1, 9 and 13 from the images in Figure 2.4 (D'apuzzo et al., 1999)	13
Figure 2.6 The recovered animation model (D'apuzzo et al., 1999)	13
Figure 2.7 An example of a human head constructed from scanned polygonal model (on the left) and the corresponding SH model (on the right) (Ertürk, Dennis, 1999)	15
Figure 3.1 An example of dinosaur object constructed by sample triangles	17
Figure 3.2 The wire-frame human body models; a woman and a man.....	18
Figure 3.3 Body coordinate system.....	20
Figure 3.4 GetTriangle Function.....	21
Figure 3.5 PutTriangle Procedure	21
Figure 3.6 GetTriangleCount Function	22
Figure 3.7 ClearTriangleList Procedure.....	22
Figure 3.8 DeleteTriangle Procedure	23
Figure 3.9 InsertTriangle Procedure	23
Figure 3.10 IndexOfTriangle Function	24
Figure 3.11 FindTriangle Function	24
Figure 3.12 3DFACE Group Codes	25

Figure 3.13 An example for 3DFACE entity	26
Figure 3.14 DefineProperties Procedure	29
Figure 3.15 ReadTriangle Procedure	30
Figure 3.16 WriteTriangle Procedure	31
Figure 3.17 An example for HEADER section.....	33
Figure 3.18 An example for CLASSES section.....	34
Figure 3.19 An example for BLOCKS section	35
Figure 3.20 An example for ENTITIES section	36
Figure 3.21 An example for OBJECTS section.....	37
Figure 3.22 An example for TABLES section.....	38
Figure 3.23 Example woman models with 36 and 48 body sizes	41
Figure 3.24 Example woman models with 65 cm. and 55 cm. waist sizes	42
Figure 3.25 Necessary vertices modifications	42
Figure 4.1 Body coordinate system.....	43
Figure 4.2 Body parts of an example model (a) hip (b) waist (c) upper torso (d) right upper leg (e) right lower leg (f) right foot (g) neck (h) head (i) right shoulder (j) right upper arm (k) right lower arm (l) right hand (m) left upper leg (n) left lower leg (o) left foot (p) left shoulder (q) left upper arm (r) left lower arm (s) left hand.....	47
Figure 4.3 Virtual women with 3436 triangles (on the left) and with 17535 triangles (on the right).....	48
Figure 4.4 Quadrangular mesh (on the left) (Buxton et al., 2000), triangular mesh (on the right)	49
Figure 4.5 Translation of the model.....	51
Figure 4.6 Rotation of the model	52
Figure 4.7 Scaling of the model.....	52
Figure 4.8 Events of the constructed OpenGL component	53
Figure 4.9 Properties of the constructed OpenGL component.....	54
Figure 4.10 Create Constructor	56
Figure 4.11 CreateParams Procedure.....	57
Figure 4.12 Resize Procedure	57
Figure 4.13 Paint Procedure.....	58

Figure 4.14 ExceptionGL Procedure.....	58
Figure 4.15 ErrorCheck Procedure	59
Figure 4.16 Swap Procedure	59
Figure 4.17 WMCreate Procedure	59
Figure 4.18 WMDestroy Procedure	60
Figure 4.19 WMERaseBkgnd Procedure	60
Figure 4.20 CreateGLContext Procedure.....	61
Figure 4.21 DestroyGLContext Procedure	62
Figure 5.1 The main form of the program.....	65
Figure 5.2 Properties Page of the program.....	66
Figure 5.3 A sample computerized human model with Points, Frames and Polygons structures	67
Figure 5.4 A sample computerized human model with different thicknesses	68
Figure 5.5 Parts Page of the program.....	69
Figure 5.6 Objects Page of the program.....	70
Figure 5.7 Modifications Page of the program	71
Figure 5.8 An example of human body model with 48 body size	72
Figure 5.9 About Form of the program.....	73

CHAPTER ONE

INTRODUCTION

Representation of bodies of real humans has been a challenging and active area in computer graphics. Advances in graphic displays have made it possible to produce highly realistic 3D human models. 3D rendering provides the possibility of showing the body from various points of view. There is an increasing demand for computer-based models in numerous applications; for example, medical research, clothing industry, virtual conferencing, realistic visualization for use in games, and lately, even e-commerce applications.

This study is concerned with the problem of building 3D models to represent human bodies. It requires knowledge about human body topology. A human body is characterized by very complicated geometrical and topological characteristics. Because of this, most of the effort was to solve the computational and algorithmic difficulties of human body shape. The aim of the study is the manufacturing of affordable high-performance human body modeling system that can be used without requiring an expert operator.

The developed system in this study takes a set of triangles as input and uses them in the derivation of a fast surface generation process. It must be noted here that, the term ‘surface’ tends to mean a polygonal representation of the object in computer graphics area. In the case where the object of interest is a human body, the problem is the extraction of symbolic information (geometry, topology, shape, and semantics) from 3D data in a fast and robust manner.

Technique in the developed system is based on triangular mesh modeling. Triangular meshes are connected meshes that are topologically dual of triangulation. Skull, skin and muscles can be represented with triangular meshes, characterized with a standard vertex to vertex connectivity. Modification of the surface parts may be easily achieved due to the local nature of triangular meshes.

1.1 What Is Human Body Modeling?

Modeling is the process of creating a resemblance of a system such as human, animal etc. A model can consist of mathematical equations, procedural rules or graphical processes. It is possible to produce computer programs that implement the model.

During the past 30 years, modeling and simulation have seen a renewed interest. Today, the process of making models constitutes essential elements in project design and management.

Modeling is the one of the biggest advantages of the computers. It provides the saving of the money and time and speeds up the necessary processes. Today, there is a wide area usage of 3D modeling techniques. Systems that can be modeled range from the very simple to the extraordinarily complex. Human body modeling takes a place in the top layers of the 3D modeling. Human body modeling is the creation and manipulation of human computer models that provide a very realistic visualization of human beings.

The researches believe that digital human modeling is a highly important research topic for industrial applications. In recent years, there has been an explosion of research on new ways of capturing, understanding and representing the human bodies. There have been several workshops and conference tracks in this area.

The human plays the very important role in many systems as a subsystem. These systems usually need human models that can ideally appear and behave like real humans.

1.2 Critical Design Issues

A critical design issue of a human modeling system is to find a reasonable compromise between visual realism, ease of control and rendering speed.

Realism depends on believable and natural appearance. For visual reality, the primary goal is good visualization. The current computer graphic techniques made it possible to develop realistic virtual humans. One of the goals of this study is to capture realistic appearance together with approximate shape information.

The other goal of the study is to develop a new approach that can be used without requiring an expert operator. The system should be easy to use for an ordinary user.

In order to achieve rapid rendering, it is important to determine what kind of detail can be considered sufficient. In other word, it is necessary determine which body parts will be modeled more accurately than others. For example, it is not necessary to model the head and hand parts in clothing sector. The system can have a modular model that consists of different sub models for different parts of the body. The number of triangles is also affects the rendering speed. If there are too many triangles, the process of building human body models might not complete rapidly. It is also important to use efficient algorithms to present the model. Data structure used for the representation also affects the rendering speed.

1.3 The Steps of the Research

The research consists of four steps: (i) examination of previous studies, (ii) observation of human shape, (iii) developing a modeling approach, and (iv) presentation of computerized models of the human bodies, as seen in Figure 1.1.

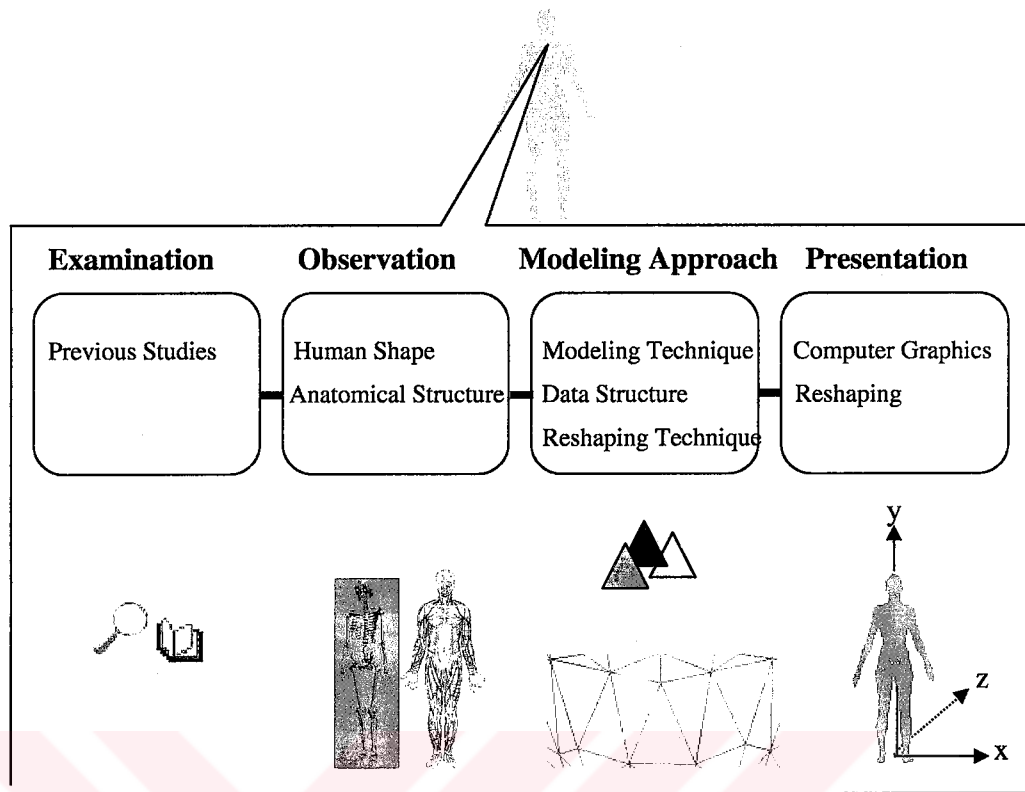


Figure 1.1 The steps of the research

As the first phase of the study, the existing studies related with the topic of human body modeling are examined. Modeling approaches proposed by the previous works are surveyed and their strengths and weaknesses are examined considering the visual quality and modeling capabilities. These works prove that it is possible to build a model that describes the surface of a real human body.

The second phase was the observation of human shape and anatomical structure of human bodies. Because the uses of the approaches for human body modeling require knowledge about body topology.

Developing a modeling approach step includes constituting a technique for modeling, finding a suitable data structure for the representation and constituting a technique for reshaping the model.

The last step consists of the usage of computer graphic techniques for the representation and the modification of the body size of the model to get a desired body size.

1.4 Application Areas

A computerized model of the human can be used in many different application areas, for example, in medical research, clothing industry, realistic visualization for use in games, virtual conferencing. In the most of the applications, many stages are needed such as observation of human operator, scrutiny the data in order to remove artifacts, placing landmarks, fill in missing data, taking measurements. These processes are normally slow and tiresome.

1.4.1 Medical Applications

Visualization and Modeling for medical applications arise from 3D optical whole body scanners (for example, Cyberware, Hamamatsu, Telmat) capable of rapidly capturing accurate data on human body size and shape. Such scanners have an important impact on areas that are related to health and fitness research such as: analysis and treatment of posture and skeletal problems, treatment of burns and injuries, prosthetic and cosmetic surgery, body kinetics and performance analysis.

It is possible to calculate whole-body measurements from the surface representations. These may be used in medical applications to determine body composition (for example, lean and fat percentages) that it may be used in order to diagnose and monitor nutritional disorders, design appropriate diets, monitoring and prediction of body growth, and body shape analysis.

1.4.2 Clothing Industry

Clothing designers, manufacturers and retailers constitute an important part of industry that is interested in human body modeling, especially in clothing design, manufacturing, and e-commerce. A human body modeling system enables the

customers to visualize themselves in a garment before purchasing it. It provides ensure customer satisfaction and reduces their turnover rates, whereby customers can see exactly how the clothes will fit on them, and even have a choice of sizing and styling modifications. It is possible not only to try on different types of clothes, but also to fit different sizes. By this way, it helps the designers to design clothes customized to an individual's body shape. In addition, data can be sent to the cutting equipment in order to purchase the item exactly as desired.

In clothing industry, designers firstly determine a model, then prepare a pattern for the desired model, then sew a sample according to the prepared pattern, then try it onto a person who has necessary body sizes. From now on designers can see their designs as three-dimensional and they can now appraise it. If they don't like some parts of the model, they should change unlike parts of the model from the pattern, and then they should make all previous steps again. Similarly, if they want to add some parts to the model or remove some parts from the model, they should return the first process and should do all processes again. A human body modeling system gets rid of these repetitions. Because designers can see their own designs as three-dimensional on a computerized human model on the screen without sewing a sample and trying it onto a real human. They can find an answer to the 'what if' question. So they can correct errors or unlike parts of the model easily. In addition, designers don't lose their times by repeating the processes again and again. So designers can quickly pass to the production step from the design step. Furthermore, computerized systems are faster and more consistent, accurate and manageable than humans. In addition, experimental changes can be made freely since the computerized systems quickly incorporate modifications into the display of the cloth. As a result of this, prototype costs can be reduced.

1.4.3 Media and Entertainment

In the gaming industry, digital representations of human bodies are currently very popular demand. Some applications render the replacement an actor in a motion picture by a visual representation of another actor. Some animator methods are used in the special effects industry.

1.4.4 Visual Conferencing

The use of virtual humans as part of a conferencing application is an area of increasing research attention. In advanced virtual conferencing systems, virtual humans are used to represent the people taking part in a meeting in a shared environment. The system use machine vision to control a realistic virtual human in order to make virtual meetings more like physical ones. The realistic representation of users gives more natural visual feedback. In some visual conferencing applications, the video sequence of the user's face is continuously texture mapped on the face of the virtual human.

1.5 Thesis Organization

The thesis is organized as follows: Chapter 2 represents the previous studies related with the topic of human body modeling. Chapter 3 explains modeling approach developed and used in this thesis. Chapter 4 defines the presentation of computerized human models. Chapter 5 describes the 3D HBM (Human Body Modeling) Program which has been implemented for realizing our designed study. Chapter 6 concludes the thesis with interpretations about the work done and explains future works.

CHAPTER TWO

PREVIOUS STUDIES

Building 3D models of human bodies has started in late eighties, when the first 3D scanning devices appeared in the market. Then many approaches have been developed following by this hardware appearance. These modeling approaches are surveyed and their strengths and weaknesses are examined considering the visual quality and modeling capabilities. This chapter presents the previous studies related with the topic of human body modeling.

2.1 Related Works

Since the early days of 3D scanning technology, the HUMAG (Human Measurement, Anthropometry and Growth) group at Loughborough University, headed by Peter Jones, has been working on generating an approach to Human Body Modeling. The early stages of this work contained the use of the LASS (The Loughborough Anthropometric Shadow Scanner) scanner to capture accurate data on human body size and shape. Then, the captured data was sequentially cleaned, segmented, and represented as a series of B-spline curves. They have been also survey on anthropometry which is the study of the measurement of the human body. 3D surface anthropometry surveying includes the acquisition, indexing, transmission, archiving, retrieval, interrogation and analysis of body size, shape, and surface together with their variability throughout growth and development to adulthood.

For a number of years, Adrian Hilton et al. at Surrey University have been working on the building 3D models of individual people. In their study, a person's 2D silhouette information that provided from the vertical view color images is

mapped onto the 3D model. They constructed a 3D animated model of a clothed persons whole-body shape and appearance from a set of orthogonal view color images. The reconstructed model of a person is then used to capture simple 3D movements from a video image sequence.

With respect to human bodies, significant works have been carried out by the LIG/EPFL group (headed by Daniel Thalmann) and the MIRAlab group (headed by Nadia Magnenat-Thalmann) in Switzerland. Their works are essential for modeling human bodies from video sequences, surface reconstruction, animation, articulation, and rendering with morphing surfaces.

Works on human body modeling has also been going on for a number of years in the Computer Science Department of University College London. Laura Dekker et al. developed a system capable of identifying up to 90 significant landmarks on the body surface. They use these landmarks to calculate sets of anthropometric measurements on the body. In their works, data is captured by a Hamamatsu whole body range scanner in order to generate quadmesh representations of human bodies. They also improved the model of the torso using a B-spline approximation.

Pascal Fua et al. have been also work on the process of building complete and realistic animation models of humans from video sequences. Their image capturing system is composed of three synchronized CCD cameras and a frame grabber which acquires a sequence of triplet images. From the video sequences, they extract two kinds of 3D information: a 3D surface measurement of the body parts for each triplet and 3D trajectories of points on the body. Their approach to surface measurement is based on multi-image matching, in which the adaptive least squares method is used. The 3D information extracted from the video sequences can be used to reconstruct the animation model of the original sequence.

There are also many examples like the work of Ong and Gong for the generation of a dynamic human model using hybrid representations in hierarchical PCA space,

as well as the work of Nurre on the automatic segmentation of the human body into its functional parts.

2.2 Existing Techniques for Human Body Modeling

Many human body modeling techniques are developed in recent years due to advanced computer technology. This section reviews the existing techniques for human body modeling.

2.2.1 Geometric Modeling

Geometric modeling is the description of object shape by the collection of components that are characterized by geometry. In three-dimensional applications, object shape is built by using well-defined blocks such as cylinders, spheres, cubes, etc. While this technique is successful for simple geometric objects, it fails to represent objects with more complex shape like human body. Because it hides most of the detail while representing complex shapes. It only provides a general idea of the object shape. While an example of upper human-body model with generalized cylinders is displayed in Figure 2.1, an example of human-hand is showed in Figure 2.2.

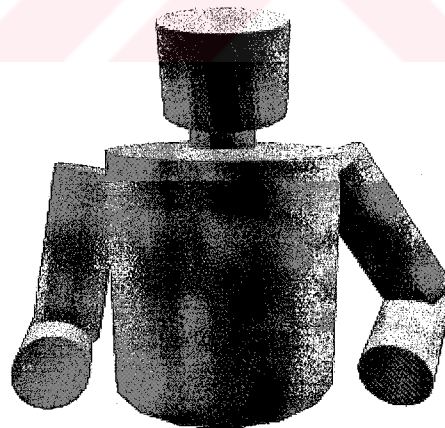


Figure 2.1 Geometric model of the upper human body (Ertürk, 1999)

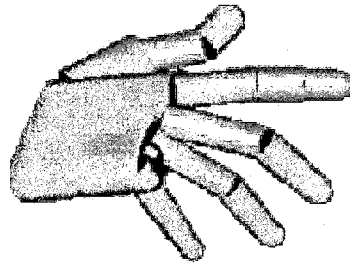


Figure 2.2 Geometric model of the human hand (Rehg, Kanade, 1994)

2.2.2 Mesh Based Modeling

There are now a number of 3D scanners, such as Hamamatsu, Telmat, Cyberware. Many techniques have been developed for converting the collection of data points delivered by such scanners into a surface representation as a mesh. Mesh representation is convenient for visualization and a number of applications, qualitatively such as texture mapping or quantitatively such as surface area measurement.

Mesh-based modeling is based on the idea of breaking the entire data set into a number of small, simple and easily manageable subsets. Sample points in those simple subsets located on the object surface are connected to each other with polygonal facets. These constructions can be used to model complex shapes in an easy way. The construction of complex shapes requires large amount of data to define it fine detail.

Mesh-based modeling is quite standard in human body modeling, because of their simplicity in the use. It has the advantage of been easy to formulate and (usually) computationally cheap. Thus there has been a considerable amount of work on modeling the human body surface this way. The mesh can be constructed from different topologies such as quadrangle, triangular. While quadmesh structure losses of detail, triangulated mesh structure provide very good and acceptable detailed information on the surface. It is even possible to carry out operations like Gaussian smoothing on a triangular mesh. On the other hand, the connectivity of the

quadrangles is more regular than triangle ones. This property makes easy the modification of the shape. Figure 2.3 compares two body surfaces constructed by quadrangles and triangles.

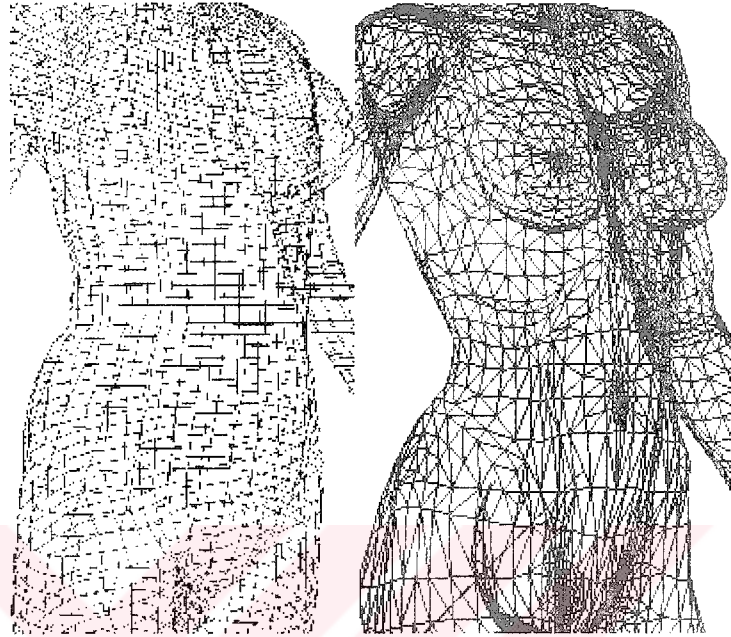


Figure 2.3 Quadrangular mesh (on the left), triangular mesh (on the right) (Buxton et al., 2000)

2.2.3 Human Body Modeling from Video Sequences

This method includes the extraction of 3D information of the shape and realistic movement of the body using video sequences. These video sequences of a moving person are acquired with a multi-camera system. Then they are used to recover joint locations during the movement and recover shape information. By using the recovered shape and motion parameters, the model is reconstructed to realize the original movement. Figure 2.4 shows an example of the video sequences of the arm of a person.

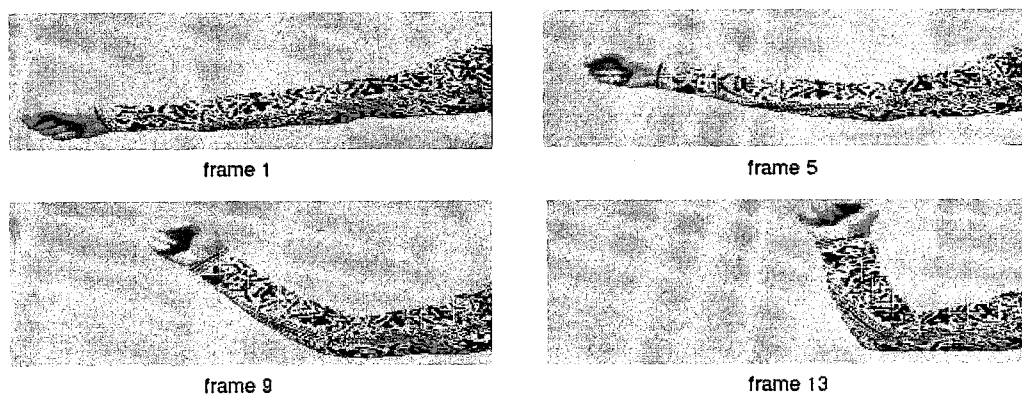


Figure 2.4 Four frames of a video sequence (D'apuzzo et al, 1999)

Figure 2.5 shows the reconstructed the positions and shapes by fitting the model of the right arm to the stereo information obtained from the images in Figure 2.4. Then the joint angles can be used to animate the model, as shown in Figure 2.6.

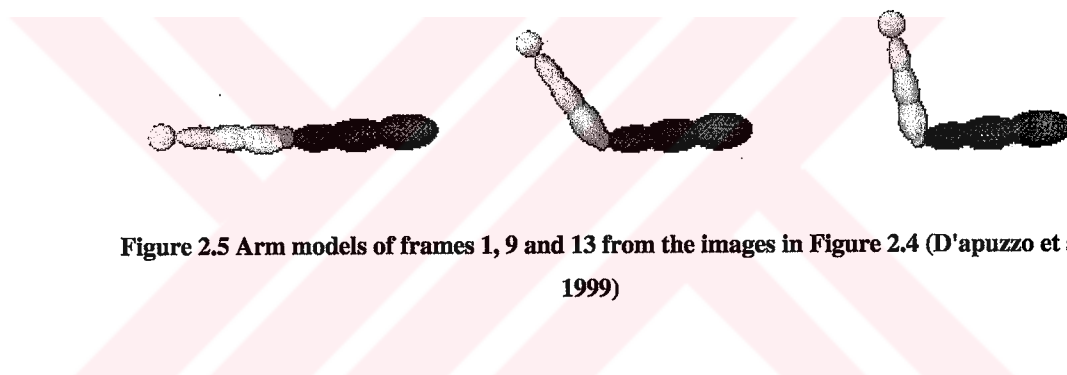


Figure 2.5 Arm models of frames 1, 9 and 13 from the images in Figure 2.4 (D'apuzzo et al., 1999)

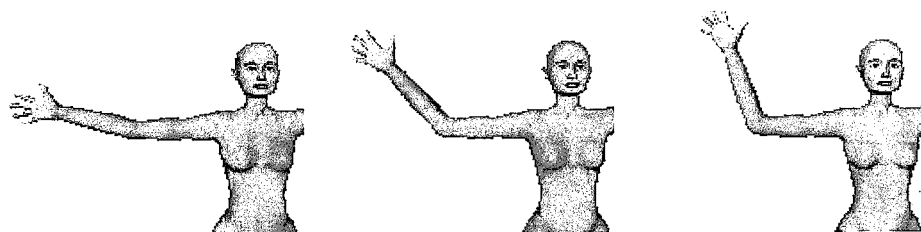


Figure 2.6 The recovered animation model (D'apuzzo et al., 1999)

2.2.4 Spherical Harmonic Modeling

Spherical harmonic (SH) shape representation is also one of the techniques available for the human body modeling. Spherical harmonic modeling is mainly the extension of SH techniques for shape representation. It is derived from Laplace's equation being solved in the spherical coordinate system.

“Spherical harmonics provide high rate of 3D shape compression with a compact formulation. The representation of each individual object is unique. Creation of specific shapes is complicated, as the harmonics are not related to the resultant shape in an intuitive way”. (Ertürk, 1999)

Operating from range data the SH representation ignores fine surface detail and retains the general shape suitable for texture mapping. The SH surface is represented with a limited number of bases function coefficients, analogous to Fourier series decomposition. It is shown that the total data amount to represent any human head SH model can be limited to a few thousand bits, with quantized and variable-length coded parameters. The SH models can be reconstructed from the parameters in wire-frame format at the desired polygon resolution and post-processed with conventional graphics utilities. (Ertürk, Dennis, 1999)

Figure 2.7 shows an example of a human head constructed from scanned polygonal model (on the left) and the corresponding SH model (on the right).

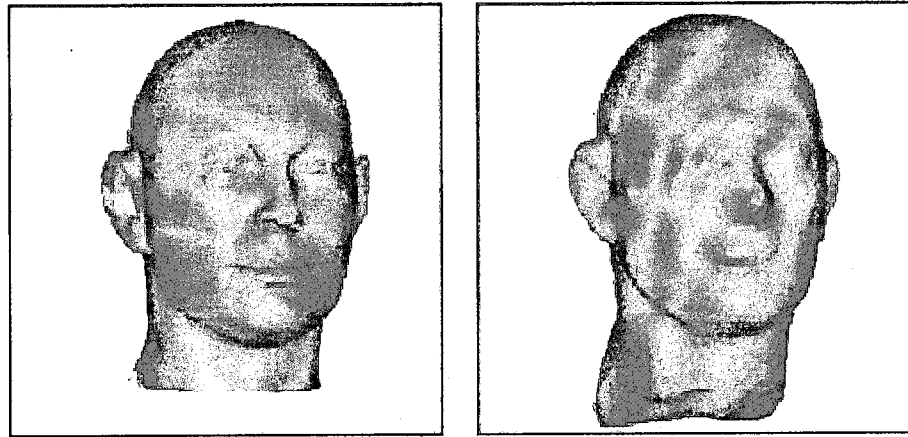


Figure 2.7 An example of a human head constructed from scanned polygonal model (on the left) and the corresponding SH model (on the right) (Ertürk, Dennis, 1999)



CHAPTER THREE

MODELING APPROACH

In this chapter, the modeling approach used in the application developed in this thesis will be described in detail. Then we will have a detailed look into the data structures, which stores this information and their relation with each other. After then the basic features for the triangles will be explained. Next section describes the format of the new file type HBM (Human Body Model), which is special for developed application. Finally, modification of the body size of the model will be clarified.

3.1 Modeling Technique

Technique in this study is based on the definition of the skin as a collection of triangles and these triangles are modified by particular functions. Triangulated models are constructed by sample triangles located on the object surface that are connected to each other with polygonal units. They can be used to model not only human bodies but also any kind of objects such as plane, car, and dinosaur. Figure 3.1 shows an example of dinosaur object, which is constructed by sample triangles in our program. In this technique, there is a large amount of data that are involved if the shape has fine details.

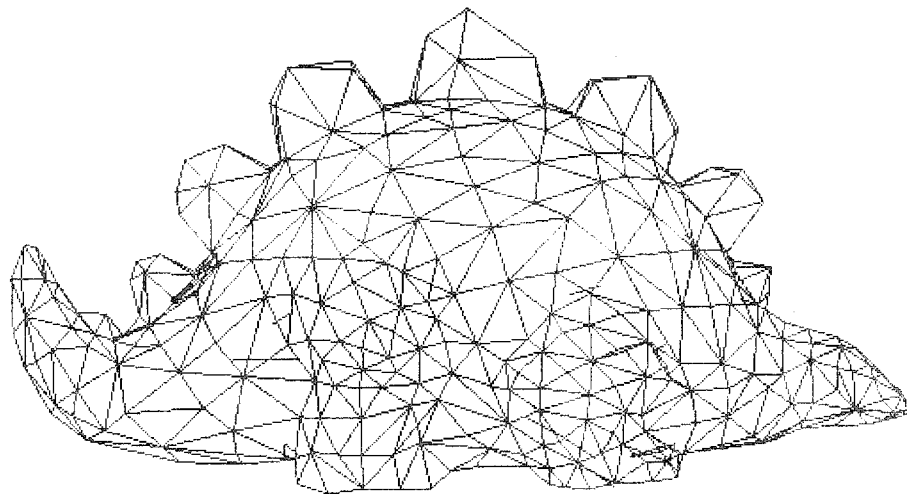


Figure 3.1 An example of dinosaur object constructed by sample triangles

A triangular mesh is an adequate representation for many applications and scales of display. Such techniques can provide very good results within certain contexts and can handle scanner data. This technique models only skin, not muscles and bones. The points of the triangles on the surfaces of the skin can be modified by some simple functions. The careful definition of functions and parameters is very important.

Some of the results are showed on Figure 3.2. Results are presented for a woman and a man. Data sets in this figure are obtained from MetaCreations Poser 4.0 program.

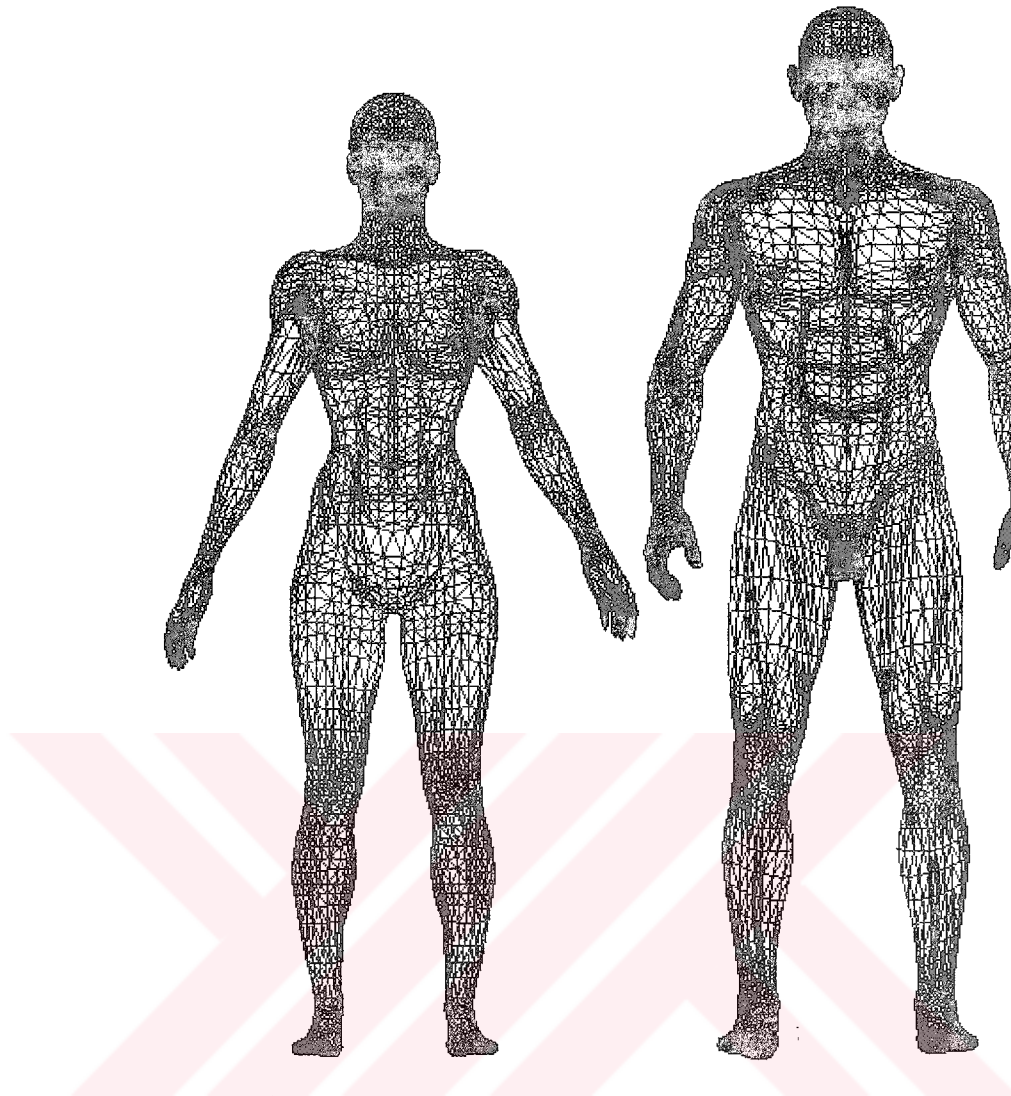


Figure 3.2 The wire-frame human body models; a woman and a man

3.2 Data Structure

The key element of human body modeling is to find a suitable data structure for the representation. In this study, triangulated meshes have been used to generate a parametric surface model to describe the human body surface. It must be noted here that, the term *surface* tends to mean a polygonal representation of the object in the computer graphics area. Our study also contains complex computational algorithms for operations on human body shape.

Usually mesh-based techniques are used for processing on human body graphics. Triangular mesh is a very standard way of representing 3D surfaces sets of any object. It is an adequate representation for many applications and scales of display. Such techniques can provide very good results within certain contexts and can handle scanner data. They can be represented by manageable and expandable data structures. The most attractive property of this technique is that the points of the triangles can be generated from real humans by body scanners. It is also possible to carry out operations like Gaussian smoothing on a triangulated mesh or diffusion and curvature flow. The definition of the triangles in the program is as follows:

```
TTriangleItem = Record
  TriSelected    : Boolean;
  TriPart        : Integer;
  x1, y1, z1,
  x2, y2, z2,
  x3, y3, z3    : GLfloat;
End;
TTriangleItemList: array [0..MAXINT div 64] of TTriangleItem;
```

As indicated above, the attributes of each triangle provided by the system are as follows:

- *TriSelected* Boolean value is used to show user selected triangles with different color. For this variable, the default value is false.
- *TriPart* indicates the identification number of the part of the human body. By using this information, triangles are classified as belong to which part of the human body. For example, if the value of *TriPart* variable for a triangle is 1, this means that this triangle belongs to hip section of the body. If its value is 2, this means that this triangle belongs to waist section the body, and so on. It means that the triangles in the same part of the body have the same *TriPart* value.
- *Triangulating point sets* consist of floating values of x1, y1, z1, x2, y2, z2, x3, y3, z3. Data sets are in a common coordinate frame, where the primary

axis is body width left to right (x), the second axis corresponds to body height (y), and the tertiary axis corresponds to depth (z), as shown in Figure 3.3.

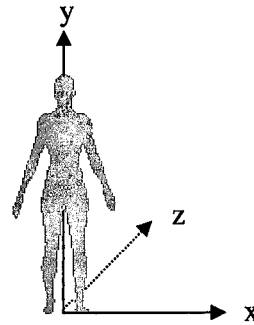


Figure 3.3 Body coordinate system

TTriangleItemList is the list of the triangles, which are used to construct the surfaces of the human bodies.

3.3 Basic Features for the Triangles

Basic features provided in the program about the triangles include:

- *GetTriangle* Function: This function returns the triangle indicated by an index. In order to prevent a possible error, the index is firstly checked to determine whether it is less than zero or it is greater than number of triangles. Then the triangle indicated by an index is returned as output. *GetTriangle* function is shown in Figure 3.4.


```

FUNCTION TEntityList.GetTriangle(Index: Integer): TTriangleItem;
var
  T : TTriangleItem;
begin
  if (Index < 0) or (Index >= FTriangleCount) then Error(@SListIndexError, Index);
  with FTriangleList^[Index] do
  begin
    T.TriSelected := TriSelected;
    T.TriPart:=TriPart;
    T.x1 := x1; T.y1 := y1; T.z1 := z1;
    T.x2 := x2; T.y2 := y2; T.z2 := z2;
    T.x3 := x3; T.y3 := y3; T.z3 := z3;
  end;
  Result := T;
END;

```

Figure 3.4 GetTriangle Function

- *PutTriangle* Procedure: This procedure puts the given triangle into indicated location of *TriangleList*. In order to prevent a possible error, the index is firstly checked to determine whether it is less than zero or it is greater than number of triangles. Then the given triangle is placed into indicated location of *TriangleList*. *PutTriangle* procedure is shown in Figure 3.5.

```

PROCEDURE TEntityList.PutTriangle(Index: Integer; const T : TTriangleItem);
begin
  if (Index < 0) or (Index >= FTriangleCount) then Error(@SListIndexError, Index);
  TriangleListChanging;
  with FTriangleList^[Index] do
  begin
    TriSelected := T.TriSelected;
    TriPart:=T.TriPart;
    x1 := T.x1; y1 := T.y1; z1 := T.z1;
    x2 := T.x2; y2 := T.y2; z2 := T.z2;
    x3 := T.x3; y3 := T.y3; z3 := T.z3;
  end;
  TriangleListChanged;
END;

```

Figure 3.5 PutTriangle Procedure

- *GetTriangleCount* Function: This function returns the number of existing triangles when it is called. *GetTriangleCount* function is shown in Figure 3.6.

```

FUNCTION TEntityList.GetTriangleCount: Integer;
begin
    Result := FTriangleCount;
END;

```

Figure 3.6 GetTriangleCount Function

- *ClearTriangleList* Procedure: This procedure deletes all triangles from the *TriangleList* if the number of triangles is different from zero. *ClearTriangleList* procedure is shown in Figure 3.7.

```

PROCEDURE TEntityList.ClearTriangleList;
begin
    if FTriangleCount <> 0 then
        begin
            TriangleListChanging;
            Finalize{FTriangleList^[0], FTriangleCount};
            FTriangleCount := 0;
            SetTriangleListCapacity(0);
            TriangleListChanged;
        end;
    END;

```

Figure 3.7 ClearTriangleList Procedure

- *DeleteTriangle* Procedure: This procedure deletes the triangle indicated by an index from *TriangleList*. In order to prevent a possible error, the index is firstly checked to determine whether it is less than zero or it is greater than number of triangles. Then the indicated triangle is deleted from *TriangleList*. *DeleteTriangle* procedure is shown in Figure 3.8.

```

PROCEDURE TEntityList.DeleteTriangle(Index: Integer);
begin
    if (Index < 0) or (Index >= FTriangleCount) then Error(@SListIndexError, Index);
    TriangleListChanging;
    Finalize{FTriangleList^[Index]};
    Dec{FTriangleCount};
    if Index < FTriangleCount then
        System.Move{FTriangleList^[Index + 1], FTriangleList^[Index],
            (FTriangleCount - Index) * SizeOf(TTriangleItem)};
    TriangleListChanged;
END;

```

Figure 3.8 DeleteTriangle Procedure

- *InsertTriangle* Procedure: This procedure inserts the given triangle into indicated location of *TriangleList*. In order to prevent a possible error, the indicated location is checked to determine whether it is less than zero. *InsertTriangle* procedure is shown in Figure 3.9.

```

PROCEDURE TEntityList.InsertTriangle(Index: Integer; const T: TTriangleItem);
begin
    TriangleListChanging;
    if FTriangleCount = FTriangleCapacity then GrowTriangleList;
    if Index < FTriangleCount then
        System.Move{FTriangleList^[Index], FTriangleList^[Index + 1],
            (FTriangleCount - Index) * SizeOf(TTriangleItem)};
    with FTriangleList^[Index] do
        begin
            TriSelected := T.TriSelected;
            TriPart:=T.TriPart;
            x1 := T.x1; y1 := T.y1; z1 := T.z1;
            x2 := T.x2; y2 := T.y2; z2 := T.z2;
            x3 := T.x3; y3 := T.y3; z3 := T.z3;
        end;
    Inc{FTriangleCount};
    TriangleListChanged;
END;

```

Figure 3.9 InsertTriangle Procedure

- *IndexOfTriangle* Function: This function returns the index of the given triangle. It calls the *FindTriangle* function which searches the given triangle in *TriangleList*. If the *FindTriangle* function returns false (this means that the triangle is not found in the *TriangleList*), *IndexOfTriangle* function returns -1 to indicate this situation. *IndexOfTriangle* function is shown in Figure 3.10.

```

FUNCTION TEntityList.IndexOfTriangle(const T: TTriangleItem): Integer;
begin
  if not FindTriangle(T, Result) then Result := -1;
END;

```

Figure 3.10 IndexOfTriangle Function

- *FindTriangle* Function: This function searches the given triangle in *TriangleList*. If the given triangle is not found in the list, it returns false as an output, otherwise it returns true. *FindTriangle* function is shown in Figure 3.11.

```

FUNCTION TEntityList.FindTriangle(const T: TTriangleItem; var Index: Integer): Boolean;
var
  I : Integer;
begin
  Result := False;
  for I := 0 to FTriangleCount - 1 do
  begin
    with FTriangleList^[I] do
    begin
      TriSelected := T.TriSelected;
      TriPart:=T.TriPart;
      if ((x1 = T.x1) and (y1 = T.y1) and (z1 = T.z1) and
        (x2 = T.x2) and (y2 = T.y2) and (z2 = T.z2) and
        (x3 = T.x3) and (y3 = T.y3) and (z3 = T.z3)) then
      begin
        result := true;
        Index := I;
        exit;
      end;
    end;
  end;
end;
END;

```

Figure 3.11 FindTriangle Function

3.4 Generating 3D Meshes From Data File

The program uses HBM (Human Body Model) files, which are special for developed program, to set and get data sets. Triangulating data sets consist of floating values of x1, y1, z1, x2, y2, z2, x3, y3, z3. The program can also read standard DXF (Drawing Interchange Format) files. DXF files enable the interchange of drawings between CAD systems and other programs. HBM files have two main advantages over DXF files: (i) the rendering time of DXF files is longer than the time for HBM files, (ii) the size of DXF files are almost twice larger than HBM files.

Translation of DXF files to HBM files and HBM files to DXF files is also possible in the program. During the first translation, *3dface* entities in DXF are divided into two parts to construct triangles. In the second case, triangulating data sets in HBM files are combined to construct *3dface* entities. It must be noted here that a *3dface* entity in a DXF file is used to define four corner vertices of the rectangular.

The group codes related with the *3dface* entities are given in Figure 3.12. As shown in this figure, the codes (10, 20, 30) indicate the first point of the quadrangle (x1, y1, z1), the codes (11, 21, 31) indicate the second point of the quadrangle (x2, y2, z2), the codes (12, 22, 32) indicate the third point of the quadrangle (x3, y3, z3), and the codes (13, 23, 33) indicate the fourth point of the quadrangle (x4, y4, z4).

Group Codes	Description
100	Subclass marker (AcDbFace)
10	DXF: X value of first corner (in WCS) Application: First corner (in WCS)
20, 30	DXF only: Y and Z values of first corner (in WCS)
11	DXF: X value of second corner (in WCS) Application: Second corner (in WCS)
21, 31	DXF only: Y and Z values of second corner (in WCS)
12	DXF: X value of third corner (in WCS) Application: Third corner (in WCS)
22, 32	DXF only: Y and Z values of third corner (in WCS)
13	DXF: X value of fourth corner Application: Fourth corner (if only three corners are entered, this is the same as the third corner) (in WCS)
23, 33	DXF only: Y and Z values of fourth corner
70	Invisible edge flags (optional; default = 0): 1 = First edge is invisible 2 = Second edge is invisible 4 = Third edge is invisible 8 = Fourth edge is invisible

Figure 3.12 3DFACE Group Codes

An example to the *3dface* entities is given in Figure 3.13. As shown in this figure, the quadrangle is constructed with the points ((-3.19, 2.58, -1.31), (-2.78, 2.57, -1.34), (-2.78, 2.87, -1.34), (-3.19, 2.88, -1.31)).

```

3DFACE
 8
 0

10
-3.19
20
2.58
30
-1.31

11
-2.78
21
2.57
31
-1.34

12
-2.78
22
2.87
32
-1.34

13
-3.19
23
2.88
33
-1.31
 0

```

Figure 3.13 An example for 3DFACE entity

Table 3.1 shows the results of the comparison of example HBM and DXF files according to the file size. As shown in this table, the sizes of the HBM files are nearly fifth-fifth smaller than DXF files. For example, while the DXF file for 17535 triangles takes 3.29 MB size in the memory, the HBM file with the same data set has only 1.7 MB size. The reason of this situation is that DXF files have many *special words* like SECTION, ENTITIES, 3DFACE, ENDSEC, \$EXTMAX, \$EXTMIN etc., and many *group codes* like 0 (the start of an entity, table, table entry, or file separator), 8 (layer name), 10 (X value of the primary coordinate), 20 (Y value of the primary coordinate), 30 (Z value of the primary coordinate), 50-59 (angles), 62 (color number) etc. These special words and group codes occupy a place in the memory. In opposite, HBM file only store the necessary attributes of the triangles. The detailed description of the HBM file format will be explained in the next section.

Table 3.1 Comparison results of HBM and DXF files according to the file sizes

Number of Triangles	DXF File	HBM File	Descending Ratio In File Sizes (%)
3436	664 KB	344 KB	% 48.19
4170	804 KB	416 KB	% 48.25
17535	3.29 MB	1.7 MB	% 48.32
28820	5.39 MB	2.8 MB	% 48.05
32298	6.05 MB	3.14 MB	% 48.09
34471	6.46 MB	3.35 MB	% 48.14

Table 3.2 shows the results of the comparison of example HBM and DXF files according to the file load time. As shown in this table, the load time of the HBM files are very very smaller than DXF files. For example, while the DXF file load for 17535 triangles takes 8232 milliseconds, the HBM file load with the same data set takes only 70 milliseconds. The reason of this situation is that DXF files have many *special words* like SECTION, ENTITIES, 3DFACE, ENDSEC, \$EXTMAX, \$EXTMIN etc., and many *group codes* like 0, 8, 10, 20, 30, 50-59, 62 etc. The reading process of these special words and group codes takes many times. In addition, DXF file is read line by line. In opposite, attributes of the triangles are only read from the HBM files. In addition, data set necessary for human body modeling is read from HBM files block by block. In order to take loading times, GetLocalTime(SystemTime) command is used in the program.

Table 3.2 Comparison results of HBM and DXF files according to the file loads

Number of Triangles	DXF File	HBM File	Ratio in the file loads (1/x)
3436	3254 ms.	26 ms.	1 / 125
4170	3515 ms.	30 ms.	1 / 117
17535	8232 ms.	70 ms.	1 / 117
28820	12207 ms.	120 ms.	1 / 101
32298	13439 ms.	126 ms.	1 / 106
34471	14251 ms.	131 ms.	1 / 108

3.4.1 HBM File Format

HBM (Human Body Model) file has a special file format to read and write data by using our new class, *TFilePacket*. This class is used to read/write data sets into/from HBM files. *TFilePacket* class is declared as:

```
Type
TFilePacket = class(TComponent) {This descends from TComponent}
Public
    FCountTriangle: Integer;
    FTriangle: array [0..MAXINT div 1024] of TTriangleItem;

    Procedure DefineProperties(Filer:TFile);override;
    Procedure ReadTriangle(Reader:TReader);
    Procedure DefineProperties(Writer:TWriter);
End;
```

When a class is declared, its immediate ancestor can be specified. In this declaration, *TFilePacket = class(TComponent)* declares a class called *TFilePacket* that descends from *TComponent*, which is implemented in the classes Unit of Borland Delphi 6.0. “A class type automatically inherits all of the members from its immediate ancestor. Each class can declare new members and can redefine inherited ones, but a class cannot remove members defined in an ancestor”. (Delphi Help, 2001) Hence *TFilePacket* contains all of the members defined in *TComponent* and in each of *TComponent*’s ancestors. “All of the classes from which a component inherits are called its ancestors; the component is a descendant of its ancestors”. (Delphi Help, 2001)

While *FCountTriangle* holds the number of triangles, *FTriangle* includes the triangulating point sets of the triangles to be read or write them.

DefineProperties procedure designates methods for storing a data on the file stream. “*TComponent* overrides the *DefineProperties* method defined in *TPersistent* to define "fake" Top and Left properties. These are defined so that components that are not controls can be manipulated at design-time. However, the Top and Left properties are hidden, that is, they are not published, because only controls appear at run-time”. (Delphi Help, 2001)

“*DefineProperties* is virtual; descendant classes can override it. When overriding *DefineProperties*, be aware that the Ancestor property of *Filer* might be set and that this property can determine whether or not it is appropriate to write properties. *DefineProperties* is called automatically as part of the component streaming system; do not call it directly” (Delphi Help, 2001). *DefineProperties* procedure is shown in Figure 3.14.

```
PROCEDURE TFilePacket.DefineProperties(Filer: TFiler);
begin
  inherited;
  Filer.DefineProperty (
    'FilePacket', ReadTriangle, WriteTriangle, True);
END;
```

Figure 3.14 DefineProperties Procedure

ReadTriangle procedure reads triangles from the associated stream. It uses *TReader* file object. *TReader* is a specialized filer object that reads component data from an associated stream. The methods used in this procedure are *ReadInteger* to read an integer-type number from the reader object's stream), *ReadBoolean* (to read a Boolean from the reader object's stream), *ReadFloat* (to read a float from the reader object's stream), *ReadListEnd* (to read a start-of-list marker from the reader object's associated stream), *ReadListBegin* (to read an end-of-list marker from the reader object's associated stream). *ReadTriangle* procedure is shown in Figure 3.15.

```

PROCEDURE TFilePacket.ReadTriangle(Reader: TReader);
var
    i      : Integer;
    Count  : Integer;
begin
    x.ClearTriangleList;

    Count:=Reader.ReadInteger;
    Reader.ReadListBegin;
    for i:=0 to Count-1 do
        begin
            t.TriSelected:=Reader.ReadBoolean;
            t.TriPart:=Reader.ReadInteger;
            t.x1:=Reader.ReadFloat;
            t.y1:=Reader.ReadFloat;
            t.z1:=Reader.ReadFloat;
            t.x2:=Reader.ReadFloat;
            t.y2:=Reader.ReadFloat;
            t.z2:=Reader.ReadFloat;
            t.x3:=Reader.ReadFloat;
            t.y3:=Reader.ReadFloat;
            t.z3:=Reader.ReadFloat;
            x.AddTriangle(t);
        end;
    Reader.ReadListEnd;
END;

```

Figure 3.15 ReadTriangle Procedure

WriteTriangle procedure writes triangles to the associated stream. It uses *TWriter* file object. *TWriter* is a specialized filer object that writes data to its associated stream. The methods used in this procedure are *WriteInteger* (to write an integer-type number from the writer object's stream), *WriteBoolean* (to write a boolean from the writer object's stream), *WriteFloat* (to write a float from the writer object's stream), *WriteListEnd* (to write a start-of-list marker from the writer object's associated stream), *WriteListBegin* (to write an end-of-list marker from the writer object's associated stream). *WriteTriangle* procedure is shown in Figure 3.16.

```

PROCEDURE TFilePacket.WriteTriangle(Writer: TWriter);
var
    i : Integer;
begin
    Writer.WriteInteger (x.TriangleCount);
    Writer.WriteListBegin;
    for i := 0 to x.TriangleCount-1 do
        begin
            Writer.WriteBoolean (x.Triangles[i].TriSelected);
            Writer.WriteInteger(x.Triangles[i].TriPart);
            Writer.WriteFloat (x.Triangles[i].x1);
            Writer.WriteFloat (x.Triangles[i].y1);
            Writer.WriteFloat (x.Triangles[i].z1);
            Writer.WriteFloat (x.Triangles[i].x2);
            Writer.WriteFloat (x.Triangles[i].y2);
            Writer.WriteFloat (x.Triangles[i].z2);
            Writer.WriteFloat (x.Triangles[i].x3);
            Writer.WriteFloat (x.Triangles[i].y3);
            Writer.WriteFloat (x.Triangles[i].z3);
        end;
    Writer.WriteListEnd;
END;

```

Figure 3.16 WriteTriangle Procedure

3.4.2 DXF File Format

“Drawing Interchange Format (DXF) files enable the interchange of drawings between AutoCAD and other programs” (DXF Reference, 1995). “All implementations of AutoCAD accept the DXF format (however, backwards compatibility is not guaranteed). DXF files contain a complete description of the AutoCAD drawing. Because much of the data in an AutoCAD drawing does not have an equivalent object type in other programs”. (DXF Reference, 1995) This section describes the format of DXF files. Because our application can process these files to obtain the points of the triangles on the human body surface.

Fundamentally a DXF file is contains of pairs of *group codes* and associated *values*. *Group codes* indicate the type of values such as 0 (the start of an entity, table, table entry, or file separator), 8 (layer name), 10 (X value of the primary coordinate), 20 (Y value of the primary coordinate), 30 (Z value of the primary coordinate), 50-59

(angles), 62 (color number). The *values* can be a string, an integer, or a floating-point value. Each group code and value has its own line in the DXF file.

A DXF file is organized into sections. “Each section starts with a group code 0 followed by the string, SECTION. This is followed by a group code 2 and a string indicating the name of the section (for example, HEADER). Each section is composed of group codes and values that define its elements. A section ends with a 0 followed by the string ENDSEC”. (DXF Reference, 1995) The overall organization of a DXF file is as follows:

- **HEADER Section:** General information about the drawing is found in this section. For example; \$EXTMAX 10, 20, 30 (X, Y, and Z drawing extents upper-right corner), \$EXTMIN 10, 20, 30 (X, Y, and Z drawing extents lower-left corner), \$PDSIZE 40 (Point display size), \$PLINEWID 40(Default polyline width), \$TEXTSIZE 40 (Default text height). An example for Header section is given in Figure 3.17.

```

HEADER
9
$ACADVER
1
AC1009
9
$INSBASE
10
0.0
20
0.0
30
0.0
9
$EXTMIN
10
0.0
20
-6.124491
30
0.0
9
$EXTMAX
10
221.583297
20
132.0
30
31.842929
9
$TEXTSIZE
40
0.2
9
$TEXTSTYLE
7
STANDARD
9

```

Figure 3.17 An example for HEADER section

- **CLASSES Section:** This section holds the information for application-defined classes such as 0 <Class DXF name>, 1 <C++ Class name>, 2 <Application name>, 90 <Class version number>, 280 <was-a-zombie flag> (Set to 1 if class was not loaded when this DXF file was created, and 0 otherwise). An example for Class section is given in Figure 3.18.

```

0
SECTION
2
CLASSES
0

CLASS
1
<class dxf record>
2
<class name>
3
<app name>
90
<flag>
280
<flag>
281
<flag>

...

0
ENDSEC

```

Figure 3.18 An example for CLASSES section

- **BLOCKS Section:** The BLOCKS section of the DXF file contains block definition and the entities used in the drawing. “The format of the entities in this section is identical to those in the ENTITIES section. All entities in the BLOCKS section appear between block and endblk entities. Block and endblk entities appear only in the BLOCKS section. Block definitions are never nested (that is, no block or endblk entity ever appears within another block-endblk pair), although a block definition can contain an insert entity”. (DXF Reference, 1995) An example for Blocks section is given in Figure 3.19.

```

BLOCKS
0
BLOCK
8
0
2
TITLEB
70
0
10
0.0
20
0.0
30
0.0
3
TITLEB
0
LINE
8
BORDER
10
0.0
20
0.0
30
0.0
11
17.0
21
0.0
31
0.0
0
LINE
8
ENDBLK

```

Figure 3.19 An example for BLOCKS section

- **ENTITIES Section:** This section contains the graphical objects (entities) in the drawing. An example of existing graphical objects are 3DFACE, 3DSOLID, CIRCLE, ELLIPSE, LINE, LWPOLYLINE, MLINE, MTEXT, POINT, POLYLINE, RAY, REGION, SHAPE, SOLID, SPLINE, TEXT, VERTEX. In this thesis *3dface* entities in DXF are divided into two parts to construct triangles. An example for Entities section is given in Figure 3.20.

```

ENTITIES
0
LINE
8
CON3
10
72.0
20
72.0
30
3.0
11
72.0
21
24.0
31
3.0
VERTEX
8
RAIL
10
153.91944
20
26.08056
30
15.585271
70
64
0

```

Figure 3.20 An example for ENTITIES section

- **OBJECTS Section:** Non-graphical objects in the drawing are found in this section. All objects that are not entities or symbol table records or symbol tables are stored in this section. Examples of entries in the OBJECTS section are dictionaries such as Named Object Dictionary, MLINestyle Dictionary. An example for Objects section is given in Figure 3.21.


```

0
SECTION
2
OBJECTS
0

DICTIONARY
5
<handle>
100
AcDbDictionary

3
<dictionary name>
350
<handle of child>

0
<object type>
... <data>

0
ENDSEC

```

Figure 3.21 An example for OBJECTS section

- **TABLES Section.** The TABLES section contains several tables that they can contain a variable number of entries. Each table appears between TABLE and ENDTAB labels. Table name and table headers may follow TABLE label. Then table entries are written to the DXF file. An example for Tables section is given in Figure 3.22.

```

TABLES
0
TABLE
2
VPORT
70
8
0
VPORT
2
*ACTIVE
70
0
10
0.0
20
0.0
11
1.0
21

....

VPORT
2
3A
70
0
10
0.0
20
0.0
11
0.5
21

...

ENDTAB

```

Figure 3.22 An example for TABLES section

3.5 Modification of the Body Size of the Human Model by Using Parameters

The current body size of human model may be represented as a set of conceptual parameters. These parameters should be easy to understand for the ordinary user such as the length of the body, size of the height, chest, waist, hip etc. These parameters may allow the users to modify the body size of the models in a controllable way. In other words, desired body sizes may be obtained by changing these parameters. It is possible to calculate whole-body measurements from the surface representations.

When the user changes one of the parameters, the model is reconstructed immediately. In order to reconstruct the model, the differences should be calculated between the current body size and the desired body size. After the calculation, changes are applied on the model by particular functions for each body part. The definition of functions and parameters is very important for visual realism. One of the aims of this study is to investigate functional relationships between user inputs and the model's shapes.

The parameters of the current model are determined by using geometrical properties such as coordinates of the triangles and distances between triangles, the distance between two points.

Some parts of the body can be modified independently of other parts. For example, the shape of the foot is not related to any other parts of the human body. On the other hand, some parts are largely influenced by the adjacent parts. For example, the neck, the waist, the left and the right upper arms can be affected by the modification of upper torso. This kind of interference between body parts must be considered in modifications. Otherwise human body shape will have unnatural appearance with unconnected parts.

In our approach, the body size of the woman model is modified according to the values in Table 3.3. This table shows the basic measurements for normal woman body sizes according to the results of the English researches Taylor and Shoben. For example, when the user selects 36 body size, the height of the woman model is set to 160 cm., the hip of the model is set to 87 cm., the chest of the model is set to 82 cm., the waist of the model is set to 60 cm.

English researches (Taylor, Shoben, 1984)

[illegible]

The left part of the Figure 3.23 shows an example woman model with 36 body size, the right part shows the same woman model with 48 body size.

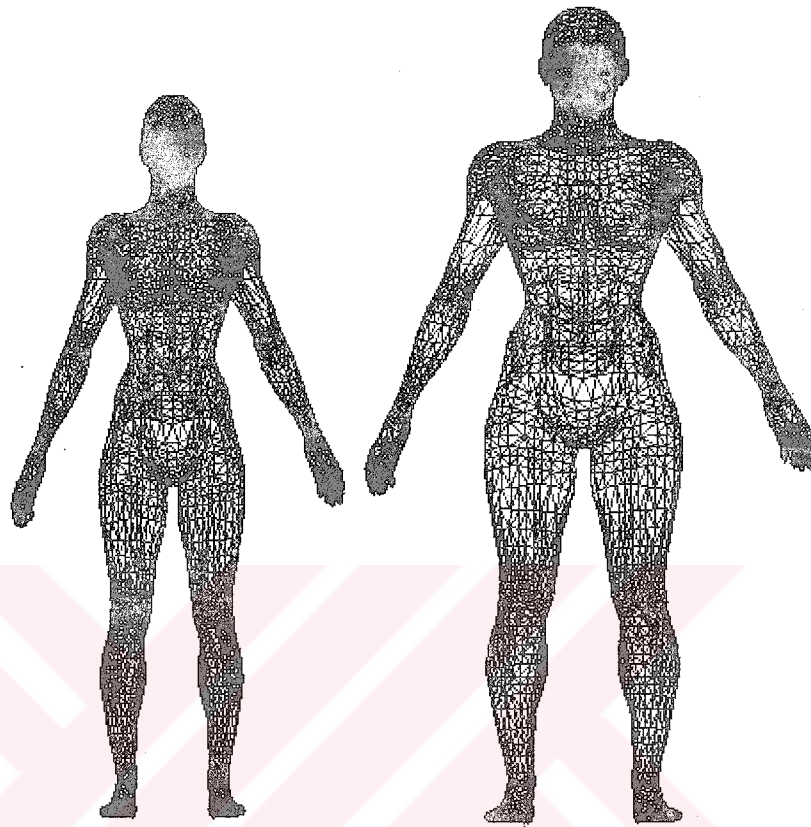


Figure 3.23 Example woman models with 36 and 48 body sizes

When the user changes the size of the one part of the model, the triangles in this part is modified without disconnecting the other triangles in adjacent parts. For example, the user can change the size of the waist from 65 cm. to 55 cm. as shown in Figure 3.24.

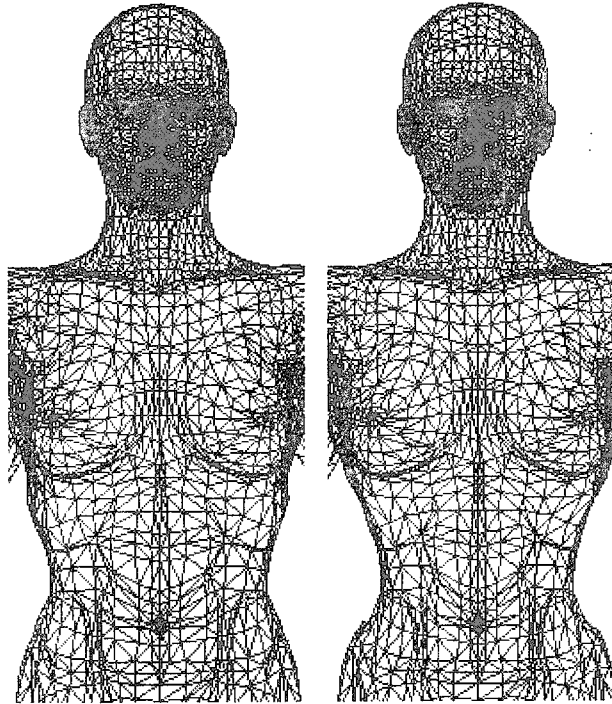


Figure 3.24 Example woman models with 65 cm. and 55 cm. waist sizes

In this situation, all vertices of all triangles in the selected part are modified. In addition, the connected vertices of the connected triangles in the adjacent part must also be modified. Figure 3.25 shows the necessary vertices modifications.

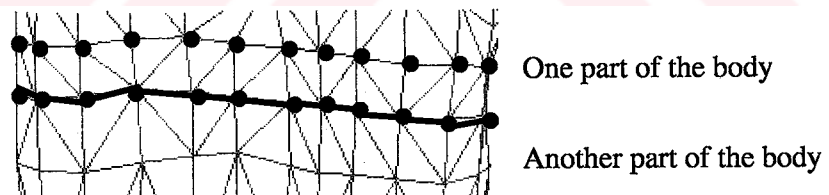


Figure 3.25 Necessary vertices modifications

CHAPTER FOUR

PRESENTATION OF COMPUTERIZED HUMAN MODELS

Presentation of computerized human models consists of the usage of computer graphic techniques for the representation and the modification of the body size of the model to get a desired body size.

4.1 Two Dimensional (2D) vs. Three Dimensional (3D) Presentation

2D and 3D are both ways of displaying data to the user. Both are viewed on a 2D screen, the monitor. The difference begins with the vertices of the mesh. While 2D point sets consists of floating values of $x_1, y_1, z_1, x_2, y_2, z_2$, 3D point sets consist of floating values of $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$. Data sets are in a common coordinate frame, where the primary axis is body width left to right (x), the second axis corresponds to body height (y), and the tertiary axis corresponds to depth (z), as seen in Figure 4.1.

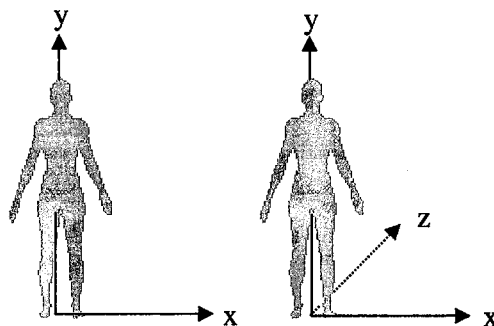


Figure 4.1 Body coordinate system

There has been a lot of work on the analysis of human body shape in 2D. The use of three dimensions enables the creation of highly interactive and flexible, user friendly applications. 3D objects can be viewed from any angle on the screen by doing a series of mathematical equations to rotate and scale them. When a human body has been modeled as 3D, users can view any side of the body that it provides more detail view. In addition, 2D images can be textured on to the 3D object to make it look more realistic.

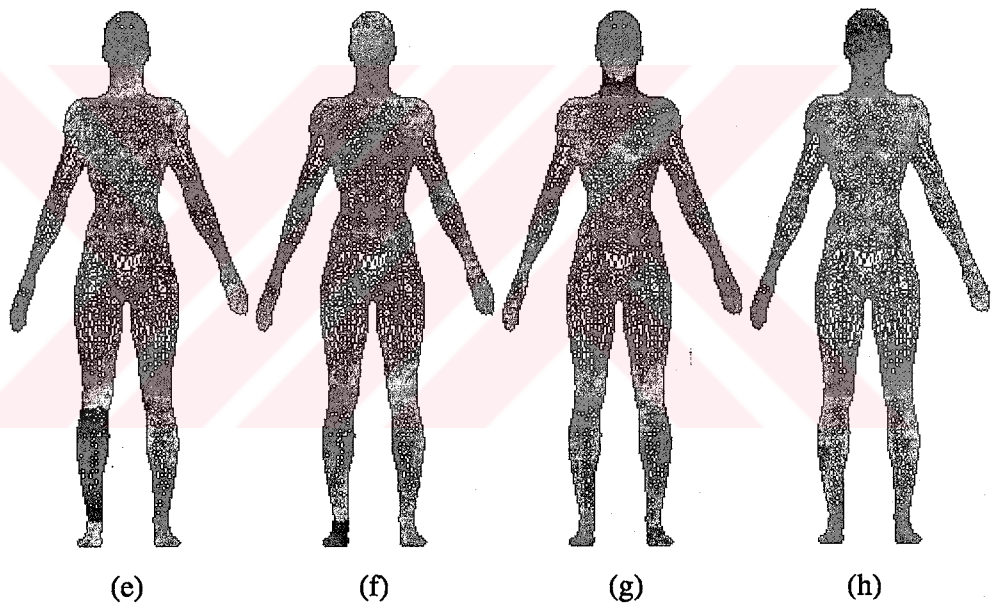
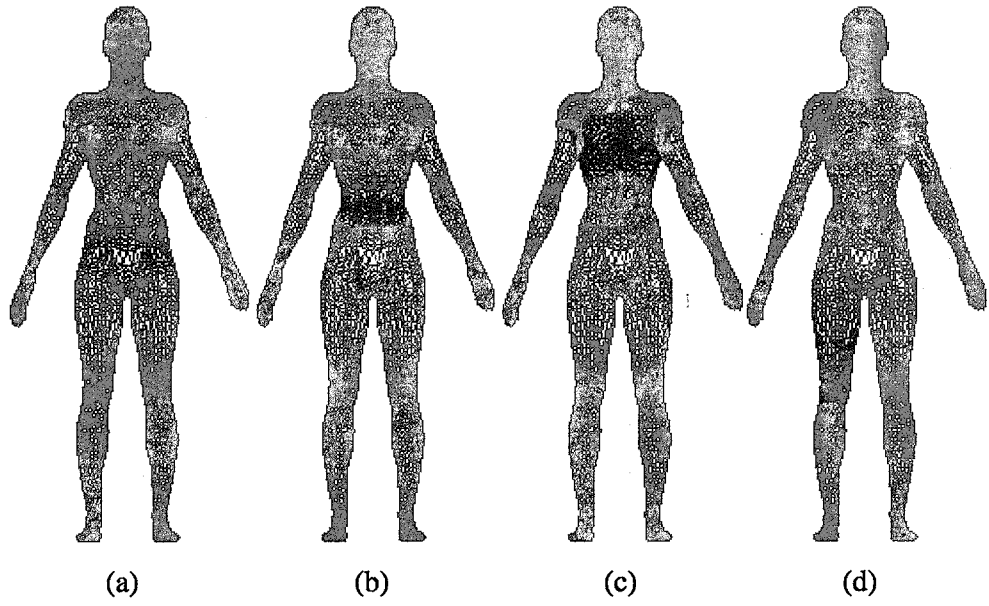
3D's flexibility can also be seen in the animation of 3D objects. To animate 2D objects would be very limiting in the angles they could be viewed from. 3D objects can be given processes for deformation and movement.

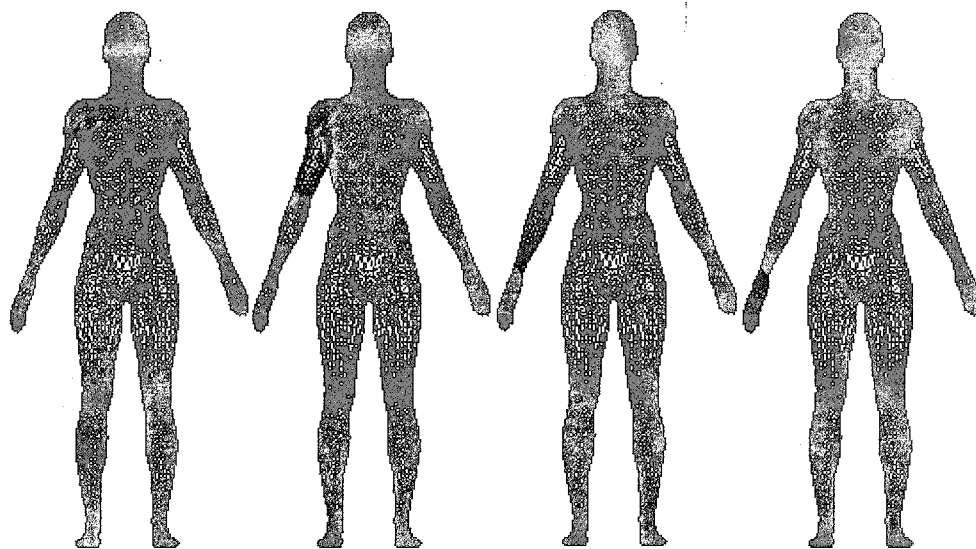
3D visualization is not always the optimal choice. Sometimes 2D holds some advantages of its own over 3D displaying methods. 2D is relatively faster to draw and process. 2D objects can be created more simply by using mathematical equations.

4.2 Body Presentation Criteria

The whole-body representation should be generated and rendered rapidly with no manual intervention. It should also allow subjects to interact with their 3D views (rotate, zoom, etc.) in real-time on a moderately powerful computer.

The models consist of some body parts like head, neck, upper torso, waist, hip, left and right upper arm, lower arm, hand, upper leg, lower leg and foot as shown in Figure 4.2. It is necessary to determine what kind of detail can be considered sufficient. In other word, it is necessary determine which body parts will be modeled more accurately than others. For example, it is not necessary to model the head and hand parts in clothing sector. The system can have a modular model that consists of different sub models for different parts of the body.



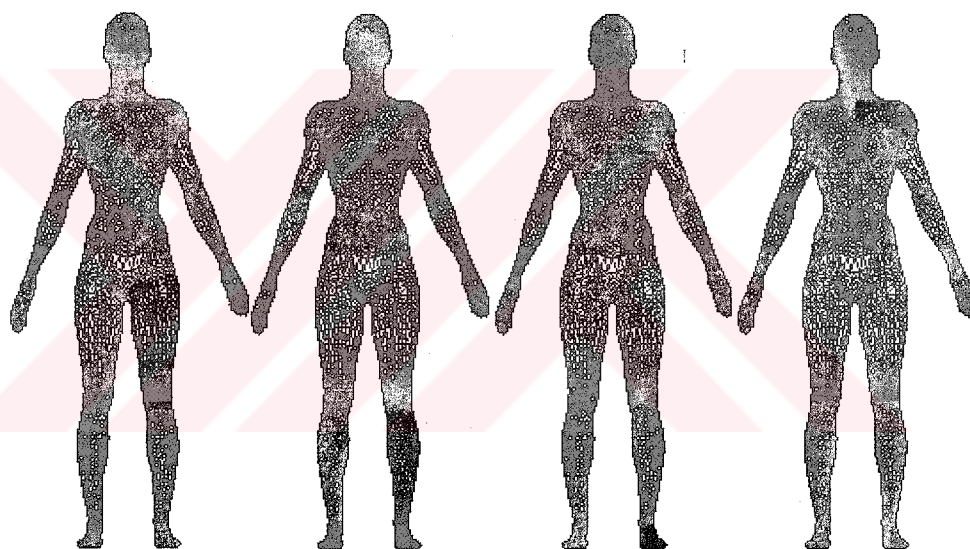


(i)

(j)

(k)

(l)



(m)

(n)

(o)

(p)

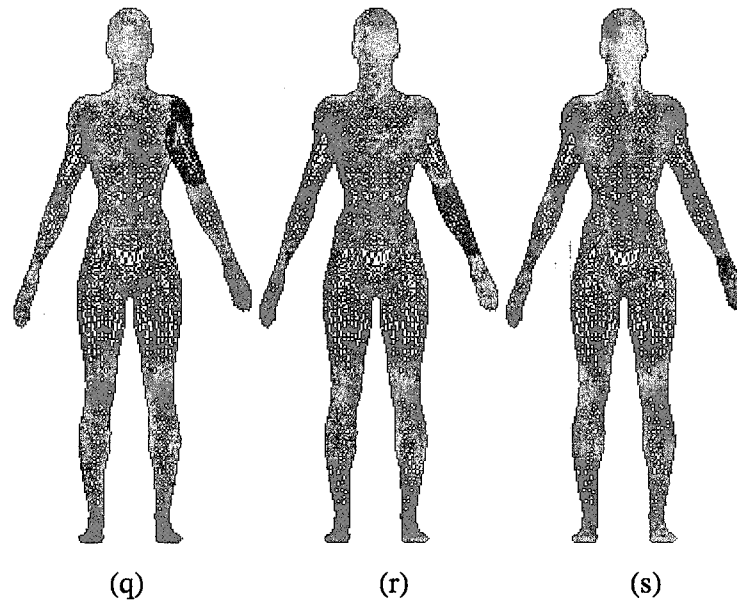


Figure 4.2 Body parts of an example model (a) hip (b) waist (c) upper torso (d) right upper leg (e) right lower leg (f) right foot (g) neck (h) head (i) right shoulder (j) right upper arm (k) right lower arm (l) right hand (m) left upper leg (n) left lower leg (o) left foot (p) left shoulder (q) left upper arm (r) left lower arm (s) left hand

In order to achieve computational efficiency as well as satisfactory visualization of human body models, it is important to consider the number of triangles. If there are too many triangles, the process of building human body models might not complete rapidly. If there are few triangles, the surface appears wrinkled or angular in an unnatural way. Whereas left side of the Figure 4.3 shows a virtual human made up of 3436 triangles, the right side shows a similar virtual human with 17535 triangles. So it is necessary to find a reasonable compromise between natural appearance and rendering speed.

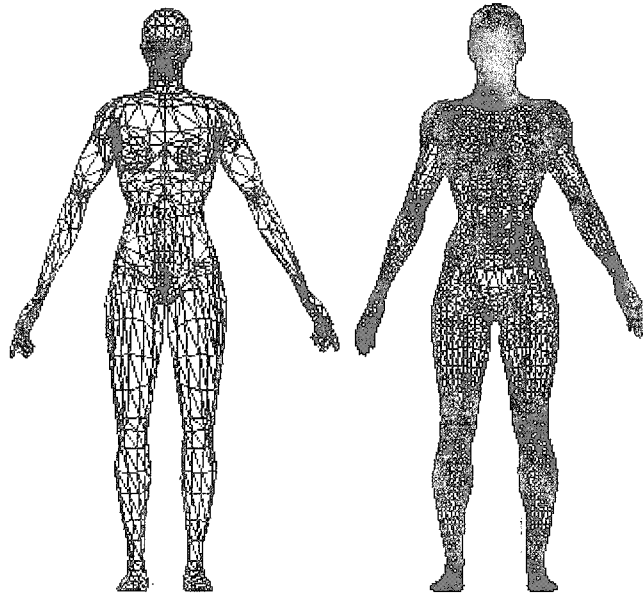


Figure 4.3 Virtual women with 3436 triangles (on the left) and with 17535 triangles (on the right)

In order to prevent unnatural appearance, two alternative methods can be applied. The first one is that new points can be generated by subdividing each triangle of the original mesh into smaller triangles. The result is still a mesh; however this subdivision can generate arbitrarily smooth surfaces. As a second method, points can be evaluated directly without subdivision. Blane et al. have demonstrated a least-squares approach for fitting implicit polynomial surfaces of very high degree to point data. Triangular meshes might also be chosen to construct the surface of human body rather than quadrangular meshes. Figure 4.4 compares two body surfaces constructed by triangles and quadrangles.

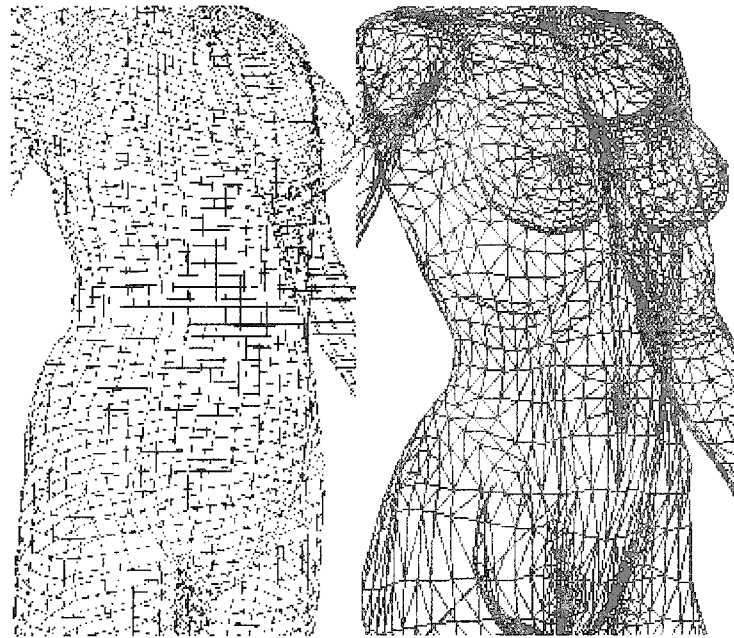


Figure 4.4 Quadrangular mesh (on the left) (Buxton et al. 2000), triangular mesh (on the right)

4.3 Computer Graphic Technique

OpenGL Graphic Interface Tool is used in this study to present computerized human models. Because OpenGL provides a powerful set of rendering commands, and all higher level drawing can be done in terms of these commands. It is a good technology to produce interactive three-dimensional graphic applications.

OpenGL graphic system is a software interface to graphics hardware (the GL stands for Graphic Library). This interface consists of approximately 120 distinct commands. These commands are used to specify the objects and operations needed to produce interactive two-dimensional (2D) and three-dimensional (3D) applications. It uses algorithms carefully developed and optimized by Silicon Graphics.

OpenGL is designed as a hardware independent interface. It can be implemented on many different hardware platforms. With OpenGL, we must build up our designed model from a set of geometric primitive-points, lines and polygons. So the

mathematical descriptions of the objects are necessary to construct shapes from geometric primitives. The basic geometric primitive used in this thesis is triangle. A triangle is drawn with OpenGL as follows:

```
glBegin(GL_TRIANGLES);
    glVertex3f(x1, y1, z1);
    glVertex3f(x2, y2, z2);
    glVertex3f(x3, y3, z3);
glEnd;
```

glBegin marks the beginning of a vertex list that describes a geometric primitive. The geometric primitive which is triangle in our case is indicated by `GL_TRIANGLES`. Then the vertices of the triangles are given to construct the triangle. *glEnd* marks the end of a vertex list.

With OpenGL, we can arrange the objects in three dimensional space and give some properties them. Some of the properties used in this thesis are providing the color of all the objects, translation of the objects, scaling of the objects, rotation of the objects, providing different viewing of the objects, changing the polygon modes of the objects.

Some limitations of OpenGL are:

- There is no direct support for printing OpenGL graphics to a monochrome printer or to a color printer with less than 4-bit places of color.
- Hardware palettes for various windows are not supported.
- Some OpenGL features are not implemented, including stereoscopic images, auxiliary buffers, and alpha bit planes.
- OpenGL has nothing to do with sound, joy sticks, etc.

4.4 Transformation Operations

Model transformation is the manipulation of objects in the clipping area translating, rotating, or scaling it. We can view a model in any orientation by transforming it in three-dimensional space.

Appropriate matrix stack must be manipulated to control model transformation for viewing and project the model onto the screen. Translating, rotating and scaling are performed using the commands `glTranslatef()`, `glRotatef()` and `glScalef()`. All three commands are equivalent to producing an appropriate translation, rotation, or scaling matrix.

Translating is the process of changing the object's position from one place to another as shown in Figure 4.5. `glTranslate{fd}(x, y, z)` command multiplies the current matrix by a matrix that moves an object by the given x , y , and z values.

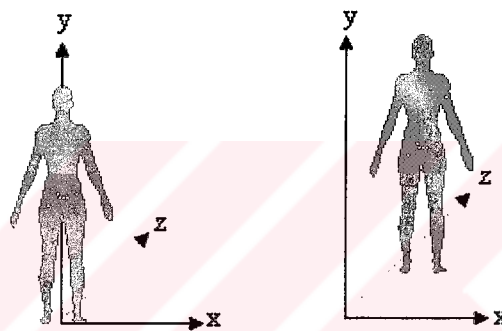


Figure 4.5 Translation of the model

Rotation is the process of rotating object in its position as shown in Figure 4.6. `glRotate{fd}(angle, x, y, z)` multiplies the current matrix by a matrix that rotates an object in a counterclockwise direction about the ray from the origin through the point (x, y, z) . The *angle* parameter specifies the angle of rotation in degrees.

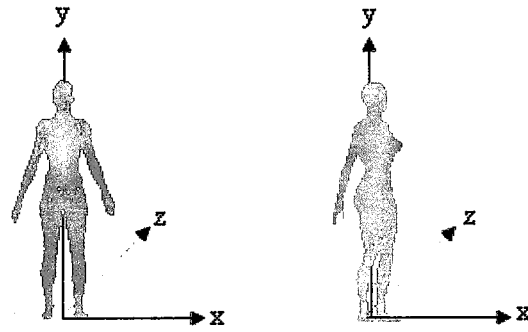


Figure 4.6 Rotation of the model

Scaling is changing the size of the object, as shown in Figure 4.7. `glScalef(x, y, z)` multiplies the current matrix by a matrix that shrinks an object along the axes. Each x , y , z coordinate value of the points in the object is multiplied by the corresponding x , y , z arguments.

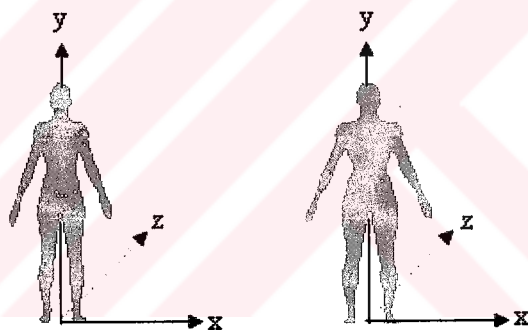


Figure 4.7 Scaling of the model

4.5 Constructed OpenGL Scene Component

Instead of writing OpenGL code in low level, we can handle most required OpenGL commands related with the drawing window in a component. By using this component, we can change the properties of the drawing window virtually. It may be very helpful to add the component that allows arranging properties such as coordinate system, alignment, projection, double buffering properties. Then desired OpenGL rendering code can be written by clicking related event from the event

handler. In addition, the repetitions can be prevented by using it in the new application developments.

In order to obtain these expectations, a component is constructed with the name OGLPnl. In the application, the patterns are drawn on the panel of this component. It changes a Delphi panel into an OpenGL window to set up the environment. Events of this component consist of *AfterWGLCreate*, *OnClick*, *OnDbClick*, *nMouseDown*, *OnMouseMove*, *OnMouseUp*, *OnPaint*, *OnResize*. Event handler of the component is shown in Figure 4.8. Many properties are also provided to pass as parameters, such as *ColorBits*, *DepthBits*, *PixelFormat*, *Projection*, *StencilBits* etc. All properties of this component are shown in Figure 4.9.

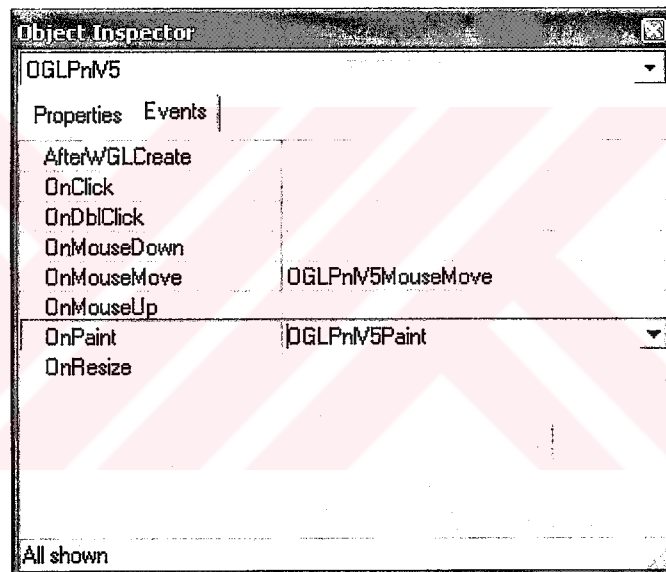


Figure 4.8 Events of the constructed OpenGL component

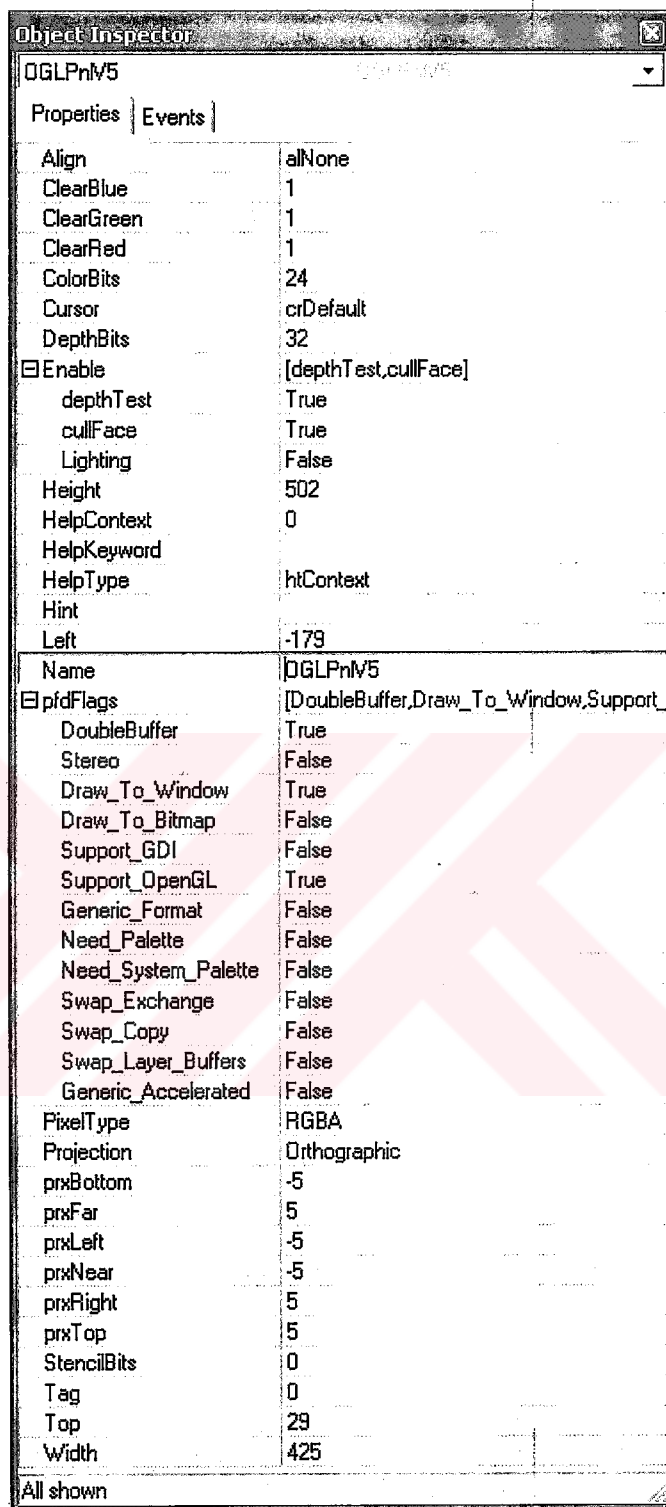


Figure 4.9 Properties of the constructed OpenGL component

4.5.1 The Events of the OGLPnl Component

Events of the component are listed in following.

AfterWGLCreate event occurs when the component is reloaded. It can be used to implement any special processing that should occur as a result of reloading.

OnClick event occurs when the user presses and releases the left mouse button with the mouse pointer over the component panel. It can be used to implement any special processing that should occur as a result of pressing and releasing the left mouse button.

OnDbClick event occurs when the user double-clicks the left mouse button when the mouse pointer is over the component panel. It can be used to implement any special processing that should occur as a result double-clicking the left mouse button.

OnMouseDown event occurs when the user presses a mouse button with the mouse pointer over the component panel. It can be used to implement any special processing that should occur as a result of pressing a mouse button. This event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations.

OnMouseMove event occurs when the user moves the mouse pointer while the mouse pointer is over a component panel. It can be used to respond when the mouse pointer moves after the panel has captured the mouse. This event may be needed to trace the position of the mouse over the component.

OnMouseUp event occurs when the user releases a mouse button that was pressed with the mouse pointer over the component panel. It can be used to implement special processing when the user releases a mouse button. Some information about the clicking area can be taken and used in the application. This event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations.

OnPaint event is required to draw shapes on the panel. OpenGL drawing commands are written in this event.

OnResize event occurs immediately after the control is resized. It can be used to make any final adjustments after the panel is resized.

4.5.2 The Methods of the OGLPnl Component

Component methods are procedures and functions built into the structure of a class. The methods of written in OGLPnl component are:

- *Create* constructor identifies the initial (default) values of some properties such as width (150) height (150), bevelwidth (1), color (clBtnFace), left (-1.0), right (1.0), bottom (-1.0), top (1.0), near (1.0), far (9.0). *Create* constructor is shown in Figure 4.10.

```

CONSTRUCTOR TOGLPnlV5.Create(AOwner: TComponent);
begin
    inherited Create(AOwner);
    ControlStyle := [csAcceptsControls, csCaptureMouse, csClickEvents,
        csSetCaption, csOpaque, csDoubleClicks, csReplicatable];
    Width       := 150;
    Height      := 150;
    BevelOuter   := bvRaised;
    BevelWidth   := 1;
    Color        := clBtnFace;

    FReady      := false;
    FProjection  := Orthographic;
    FLeft       := -1.0;
    FRight      := 1.0;
    FBottom     := -1.0;
    FTop        := 1.0;
    FNear       := 1.0;
    FFar        := 9.0;
    FColorBits  := 24;
    FDepthBits  := 32;
    FEnable     := [depthTest, cullFace];
    FFlags      := [DRAW_TO_WINDOW, SUPPORT_OPENGL];
END;

```

Figure 4.10 Create Constructor

- *CreateParams* procedure initializes a window-creation parameter named by Params. It is called automatically to initialize the window-creation parameters whenever the window for the panel needs to be created. For example window style is set as WS_CLIPCHILDREN or WS_CLIPSIBLINGS. *CreateParams* procedure is shown in Figure 4.11.

```

PROCEDURE TOGLPnlV5.CreateParams(var Params: TCreateParams);
begin
    inherited CreateParams(Params);
    Params.Style := Params.Style or WS_CLIPCHILDREN or WS_CLIPSIBLINGS;
    Params.ExStyle := Params.ExStyle or 0; // Place Holder
    //Warning Do not use CS_OWNDC as recommended by others; causes repaint problems
    Params.WindowClass.Style := Params.WindowClass.Style or 0; // Place Holder
END;

```

Figure 4.11 CreateParams Procedure

- *Resize* procedure is called automatically immediately after the control's dimensions change. New dimensions are set in this procedure. *Resize* procedure is shown in Figure 4.12.

```

PROCEDURE TOGLPnlV5.Resize;
var
    statue : boolean;
begin
    statue := (Owner <> nil) and Ready;
    if not statue then exit;
    glViewport(0, 0, Width, Height);
    ErrorCheck('Smaller');
    statue := (Width<=oldWidth) and (Height<=oldHeight);
    oldWidth := Width;
    oldHeight := Height;
    if statue then paint;
END;

```

Figure 4.12 Resize Procedure

- *Paint* procedure calls OnPaint event to process opengl commands to draw the desired geometric primitives. *Paint* procedure is shown in Figure 4.13.

```

PROCEDURE TOGLPnlV5.Paint;
begin
    //p := MaskExceptions;
    if Assigned(FOnPaint) then FOnPaint(Self);
    //UnmaskExceptions(p);
END;

```

Figure 4.13 Paint Procedure

- *ExceptionGL* procedure is used to display an exception message with its physical address when an error is occurred. In other words, it displays a message box for exceptions that are not caught by application code. *ExceptionGL* procedure is shown in Figure 4.14.

```

PROCEDURE TOGLPnlV5.ExceptionGL(Sender:TObject;E:Exception);
var
    errorCode: GLenum;
begin
    ShowException(Sender,E);
    repeat
        errorCode := glGetError;
        if errorCode<>GL_NO_ERROR then
            showMessage(gluErrorString(errorCode));
    until errorCode=GL_NO_ERROR;
END;

```

Figure 4.14 ExceptionGL Procedure

- *ErrorCheck* procedure makes error checking. *ErrorCheck* procedure is shown in Figure 4.15.

```

PROCEDURE TOGLPnlV5.ErrorCheck(const s:string);
var
    errorCode: GLenum;
begin
    errorCode := glGetError;
    if errorCode<>GL_NO_ERROR then
        raise Exception.Create('Error in '+s+#13+
            gluErrorString(errorCode));
    END;

```

Figure 4.15 ErrorCheck Procedure

- *Swap* procedure swaps the buffers with the command `SwapBuffers(glDC)`. *Swap* procedure is shown in Figure 4.16.

```

PROCEDURE TOGLPnlV5.Swap;
begin
    SwapBuffers(glDC);
END;

```

Figure 4.16 Swap Procedure

- *WMCreate* procedure calls *CreateGLContext* procedure to create OpenGL render context. *WMCreate* procedure is shown in Figure 4.17.

```

PROCEDURE TOGLPnlV5.WMCreate(var Message: TWMCreate);
begin
    CreateGLContext;

    if assigned(FAfterWGLCreate) then
        FAfterWGLCreate(Self);
    END;

```

Figure 4.17 WMCreate Procedure

- *WMDestroy* procedure calls *DestroyGLContext* procedure to destroy OpenGL render context. *WMDestroy* procedure is shown in Figure 4.18.

```

PROCEDURE TOGLPnlV5.WMDestroy(var Message: TWMDestroy);
begin
    DestroyGLContext;
END;

```

Figure 4.18 WMDestroy Procedure

- *WMEraseBkgnd* procedure sets result of the WM_ERASEBKGND message to 1, which means true. This setting provides erasing the background of the window. *WMEraseBkgnd* procedure is shown in Figure 4.19.

```

PROCEDURE TOGLPnlV5.WMEraseBkgnd(var Message: TWMEraseBkgnd);
var
    Rect      : TRect;
    OldColor  : TColor;
begin
    if (csDesigning in ComponentState) then
    begin
        OldColor := Canvas.Brush.Color;
        Rect := GetClientRect;
        Canvas.Brush.Color := clBtnFace;
        Canvas.FillRect(Rect);
        Canvas.Brush.Color := OldColor;
    end;
    message.Result := 1;
END;

```

Figure 4.19 WMEraseBkgnd Procedure

- *CreateGLContext* procedure creates OpenGL render context. It clears the buffers, identifies the projection and modelview matrices with desired projection style, chooses the pixel format, enables lights etc. *CreateGLContext* procedure is shown in Figure 4.20.


```

PROCEDURE TOGLPnlV5.CreateGLContext();
var
    pfd: TPixelFormatDescriptor;
    FormatIndex: integer;
    pflags: TPFDFlags;
    flags: word absolute pFlags;
begin
    pflags := pfdFlags;
    fillchar(pfd, SizeOf(pfd), 0);
    with pfd do
        begin
            nSize      := SizeOf(pfd);
            nVersion   := 1;
            dwFlags    := flags;
            iPixelFormat := byte(PixelType);
            cColorBits := ColorBits;
            cDepthBits := DepthBits;
            cStencilBits := StencilBits;
            iLayerType := PFD_MAIN_PLANE;
        end; {with}
    glDC := getDC(Self.handle);
    FormatIndex := ChoosePixelFormat(glDC, @pfd);
    if FormatIndex=0 then
        raise Exception.Create('ChoosePixelFormat failed '+
            IntToStr(GetLastError));

    if not SetPixelFormat(glDC, FormatIndex, @pfd) then
        raise Exception.Create('SetPixelFormat failed '+
            IntToStr(GetLastError));

    GLContext := wglCreateContext(glDC);
    if GLContext=0 then
        raise Exception.Create('wglCreateContext failed '+
            IntToStr(GetLastError));

    if not wglMakeCurrent(glDC, GLContext) then
        raise Exception.Create('wglMakeCurrent failed '+
            IntToStr(GetLastError));

    if depthTest in Enable then
        glEnable(GL_DEPTH_TEST);
    if cullFace in Enable then
        glEnable(GL_CULL_FACE);
    if Lighting in Enable then
        glEnable(GL_LIGHTING);

```

Figure 4.20 CreateGLContext Procedure

```

glMatrixMode(GL_PROJECTION);
glLoadIdentity;
if Projection=ORTHOGRAPHIC then
    glOrtho(prxLeft,prxRight,prxBottom,prxTop,prxNear,prxFar)
else
    glFrustum(prxLeft,prxRight,prxBottom,prxTop,prxNear,prxFar);

glClearColor (FClearRed,FClearGreen,FClearBlue,1.0);

ErrorCheck('Loaded');

FReady := true;
Application.OnException := ExceptionGL;
END;

```

Figure 4.20 CreateGLContext Procedure (cont.)

- *DestroyGLContext* procedure destroys Opengl render context. *DestroyGLContext* procedure is shown in Figure 4.21.

```

PROCEDURE TOGLPnlV5.DestroyGLContext();
begin
    FReady := false;
    wglMakeCurrent(glDC,0);
    wglDeleteContext(GLContext);
    ReleaseDC(Self.Handle, glDC);
END;

```

Figure 4.21 DestroyGLContext Procedure

4.5.3 Creating Components in Delphi

Components are visual objects that can be manipulated at design time. All components descend from the *TComponent*.

A new component can be created in two ways:

- (i) Using the Component wizard
- (ii) Creating a component manually

There are several basic steps that you must follow whenever you create a new component. These steps are:

1. Creating a unit for the new component
2. Deriving the component from an existing component type
3. Adding properties, methods, and events
4. Registering the component with Delphi
5. Creating a package to install the new component in the Delphi IDE

After finishing the creation of the component, the new component includes the following files:

- A package (.BPL) or package collection (.DPC) file
- A compiled package (.DCP) file
- A compiled unit (.DCU) file
- A palette bitmap (.DCR) file

CHAPTER FIVE

DESCRIPTION OF THE PROGRAM

A simple 3D HBM - Human Body Modeling program has been implemented for realizing our designed study. The software reads HBM and DXF files as input and uses triangular meshes to generate a parametric surface model to describe the human body surface. The parameters in the program allow the users to modify the body size of the models. In this chapter, detailed description of the program will be given.

5.1 Programming Environment

System is developed under Borland Delphi 6.0 programming environment. The system that the software is implemented on is an Intel Pentium III 599 MHz CPU and 256MB RAM computer that runs Microsoft Windows XP as the operating system.

OpenGL Graphic Interface Tool is also used to present computerized human models. Because OpenGL provides a powerful set of rendering commands, and all higher level drawing can be done in terms of these commands. It is a good technology to produce interactive three-dimensional graphic applications.

5.1.1 Using OpenGL with Delphi

Using OpenGL with Delphi requires a 32-bit version of Delphi, and OpenGL interface unit. OpenGL graphic system is generally provided as a library in binary format. It can be linked to interactive two or three dimensional applications dynamically. On the Windows platform, there is a DLL (*opengl.dll*) in the system

directory. So Delphi can use OpenGL techniques like other DLL's. The place to perform the OpenGL commands is Delphi event handlers.

5.2 Program Capabilities

The program uses HBM (Human Body Model) files, which are special for developed program, to get data sets. Triangulating data sets consist of floating values of x_1 , y_1 , z_1 , x_2 , y_2 , z_2 , x_3 , y_3 , z_3 . The program can also read standard DXF (Drawing Interchange Format) files. The information about these files is given in Chapter 3. Figure 5.1 shows the main form of the program.

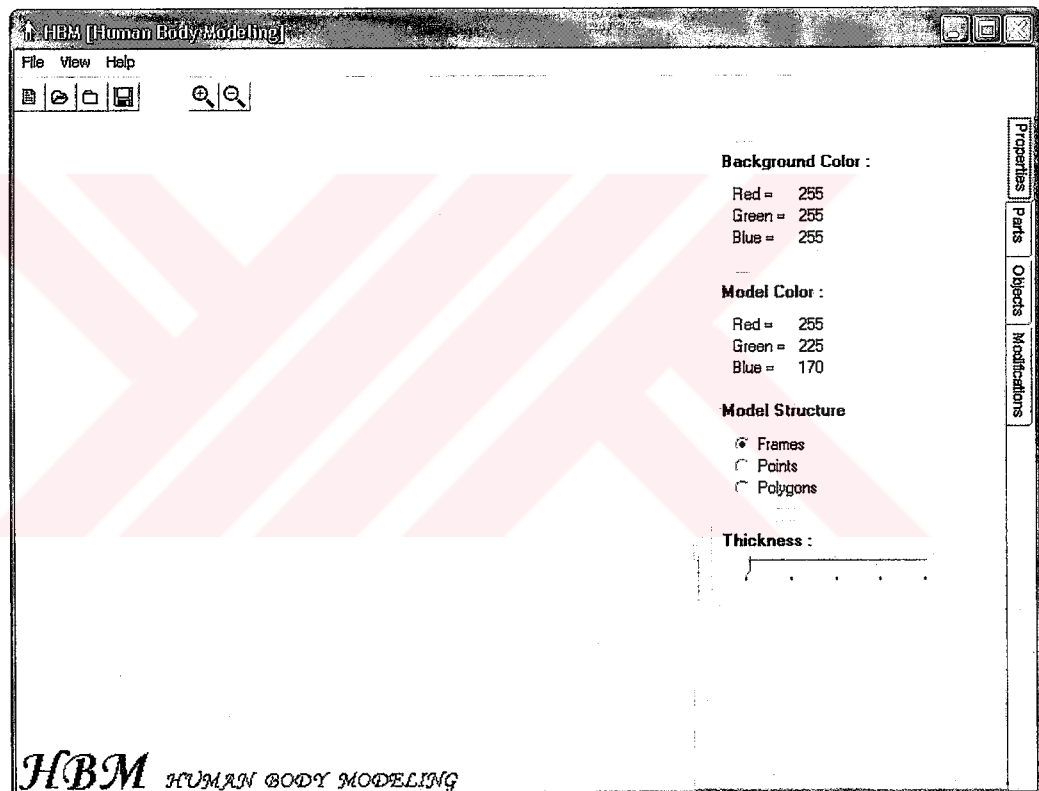


Figure 5.1 The main form of the program

The constructed model in the program can be rotated by mouse and zoomed by mouse. There is a tool bar above, which contains buttons for the basic functions such as new, open, close, save, zoom in, zoom out. There is also a file menu above the tool bar. This menu consists of file, view and help sections. File section includes file

operations such as new, open, close, save, import dxf file, export dxf file, exit from the program. View section allows the users to change some properties of the model, to view the parts of the model, to see the triangles on the model's surface, to modify the model's body size, and to zoom in and zoom out. Help section exists to help the users about the using of the program. This section also shows the about form of the program.

At the right of the window, there are pages of tools to modify the body. These pages are Properties Page, Parts Page and Objects Page, Modifications Page.

Properties Page allows the users to change some properties of the model as seen in Figure 5.2. These properties are background color, model color, model structure and thickness.

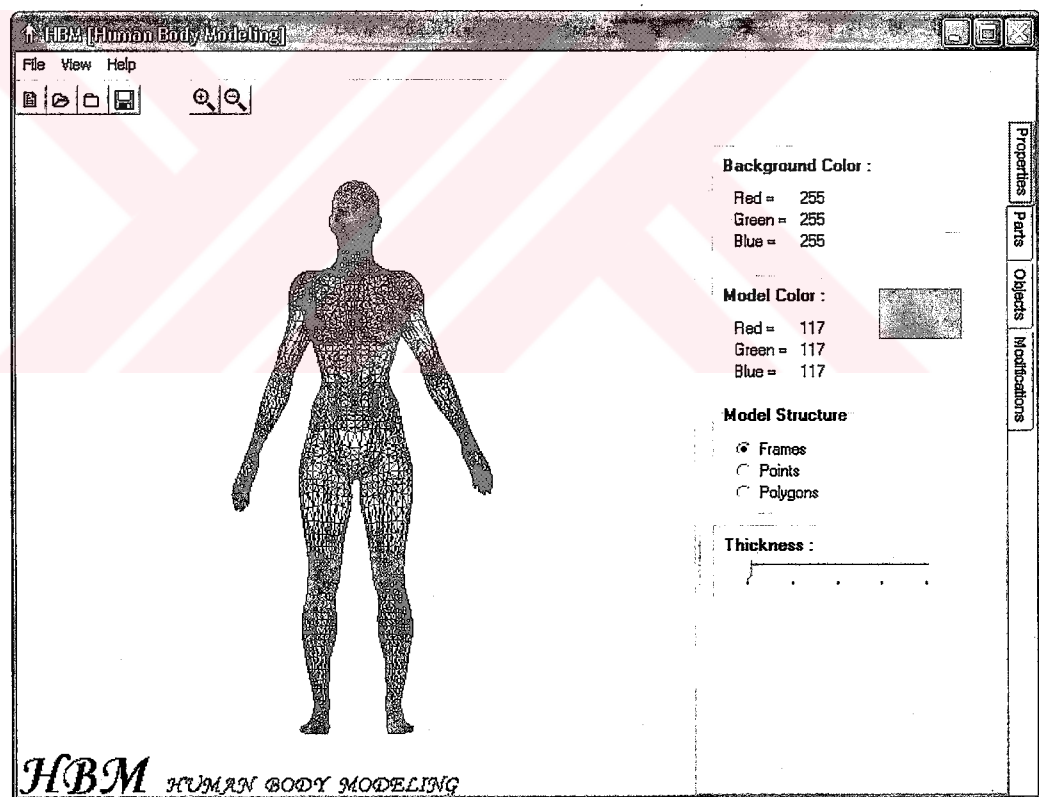


Figure 5.2 Properties Page of the program

Default background color is white. The users can be changed it by clicking on the colored panel. RGB mode correspondence of the color is also displayed on this page.

Default model color is skin color. The users can be changed it by clicking on the colored panel. RGB mode correspondence of the color is also displayed on this page.

The user can also change the model structure by selecting one the alternatives; Points, Frames or Polygons. In points structure, the model is constructed by sample points as shown in Figure 5.3 (a) In frame structure, the points located on the surface are connected each other with triangles as shown in Figure 5.3 (b) This typed models are called wire-frame models. Polygonal models are constructed by sample points that are connected to each other with polygonal facets as shown in Figure 5.3 (c).

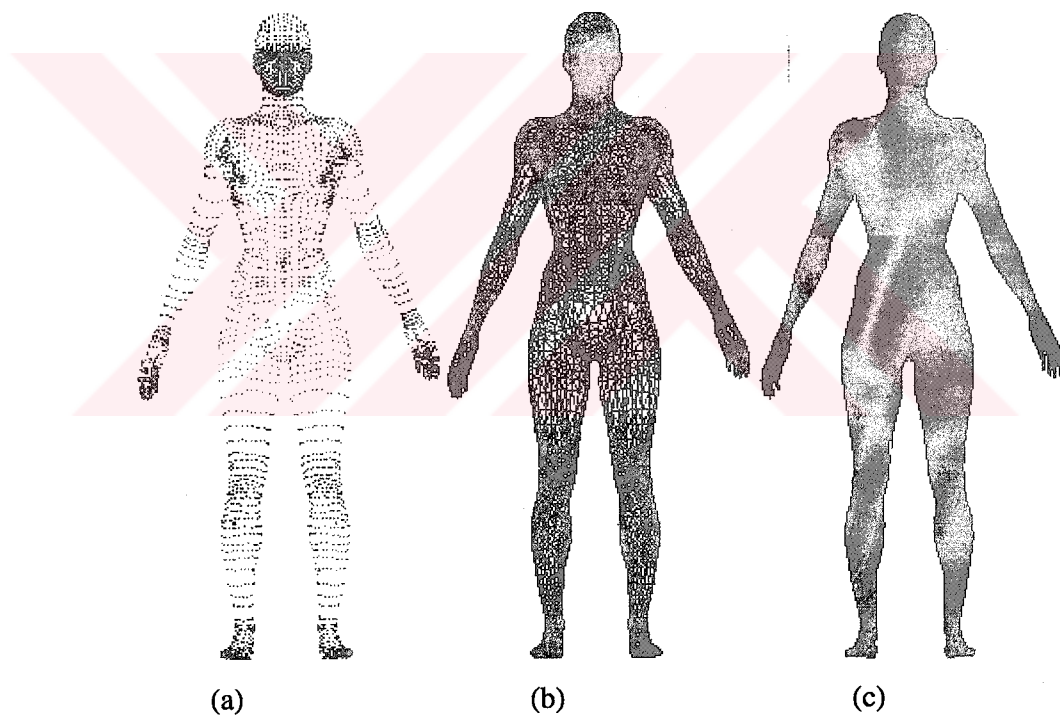


Figure 5.3 A sample computerized human model with Points, Frames and Polygons structures

The user can also change the thickness of the model structure as shown in Figure 5.4.

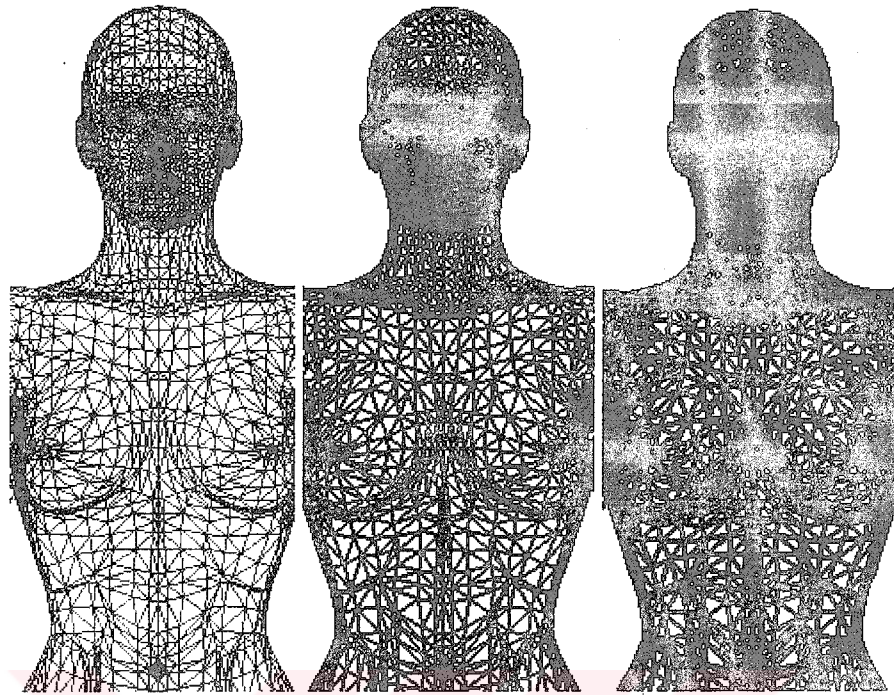


Figure 5.4 A sample computerized human model with different thicknesses

Parts Page shows the model's body parts like head, neck, upper torso, waist, hip, left and right upper arm, lower arm, hand, upper leg, lower leg and foot. Figure 5.5 shows a screen shot of parts page. The list box allows the user to select different parts of the model. When the user selects one of them, the triangles which construct this part are highlighted on the model. The user can also select more than one body parts at the same time.

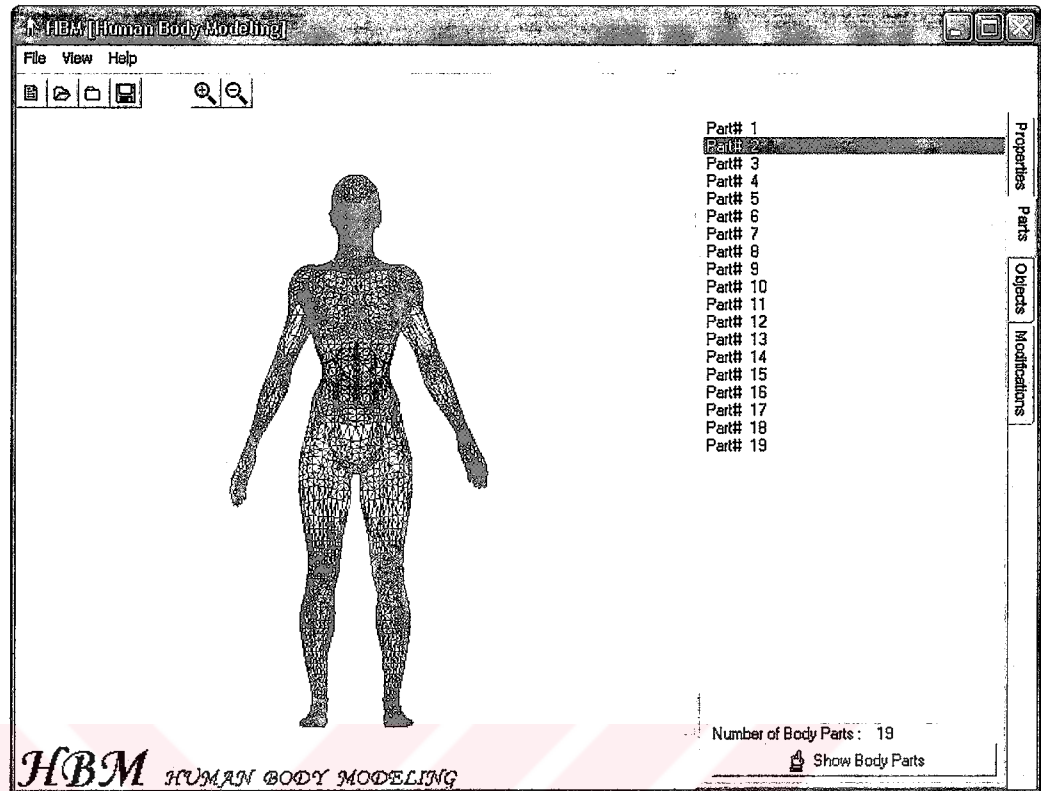


Figure 5.5 Parts Page of the program

Objects Page shows the triangles which generate the model to describe the human body. Figure 5.6 shows a screen shot of objects page. The list box allows the user to select the triangles. When the user selects one of them, this triangle is highlighted on the model. The user can also select more than one triangles at the same time.

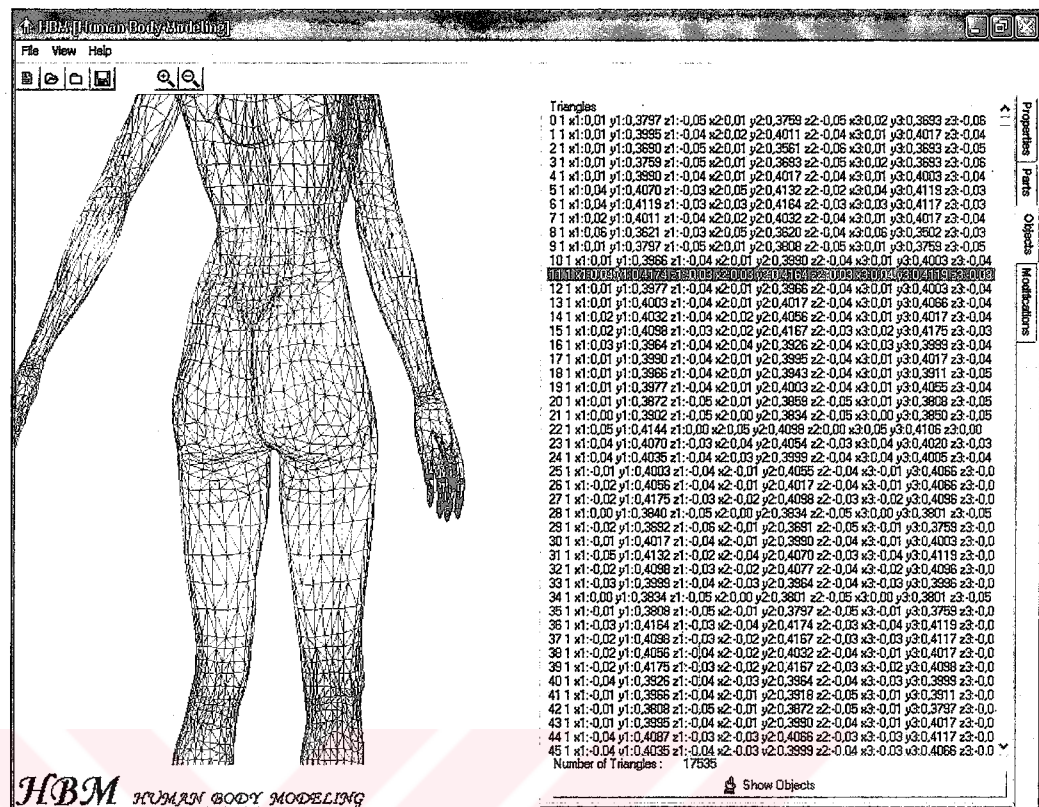


Figure 5.6 Objects Page of the program

Modifications Page shows the current size of human body model as a set of conceptual parameters. These parameters are easy to understand for the ordinary user such as the length of the body, size of the height, chest, waist, hip etc. These parameters allow the users to modify the body size of the models in a controllable way. In other words, desired body sizes can be obtained by changing these parameters. Figure 5.7 shows a screen shot of the modifications page.

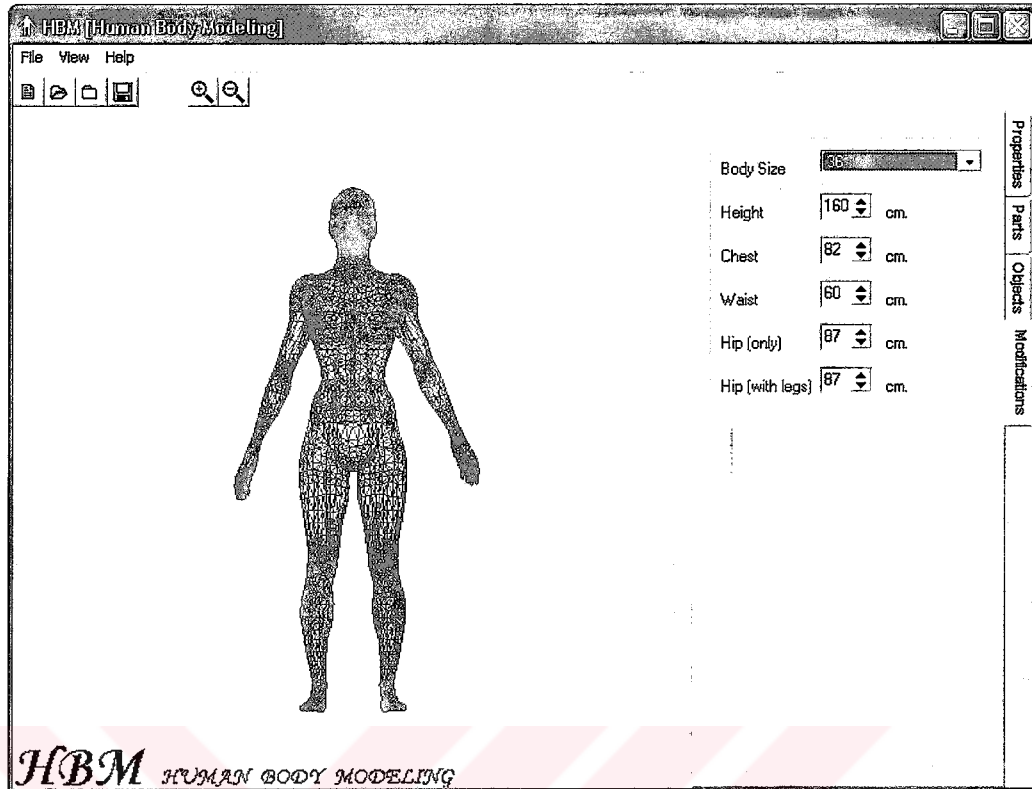


Figure 5.7 Modifications Page of the program

In modification page, SpinEdits allow the users to change the current body size. The first SpinEdit (body size) can be used to select the whole body size in 36, 38, 40, 42, 44, 46, and 48. The second SpinEdit (height) can be used to change the height of the model. The other SpinEdits (chest, waist, hip (only), hip (with legs)) can be used to change a special part of the model. For example, the fourth SpinEdit increases or decreases the size of the waist of the body. When the user selects one of them, the model is reconstructed immediately. Figure 5.8 shows a human body model with 48 body size.

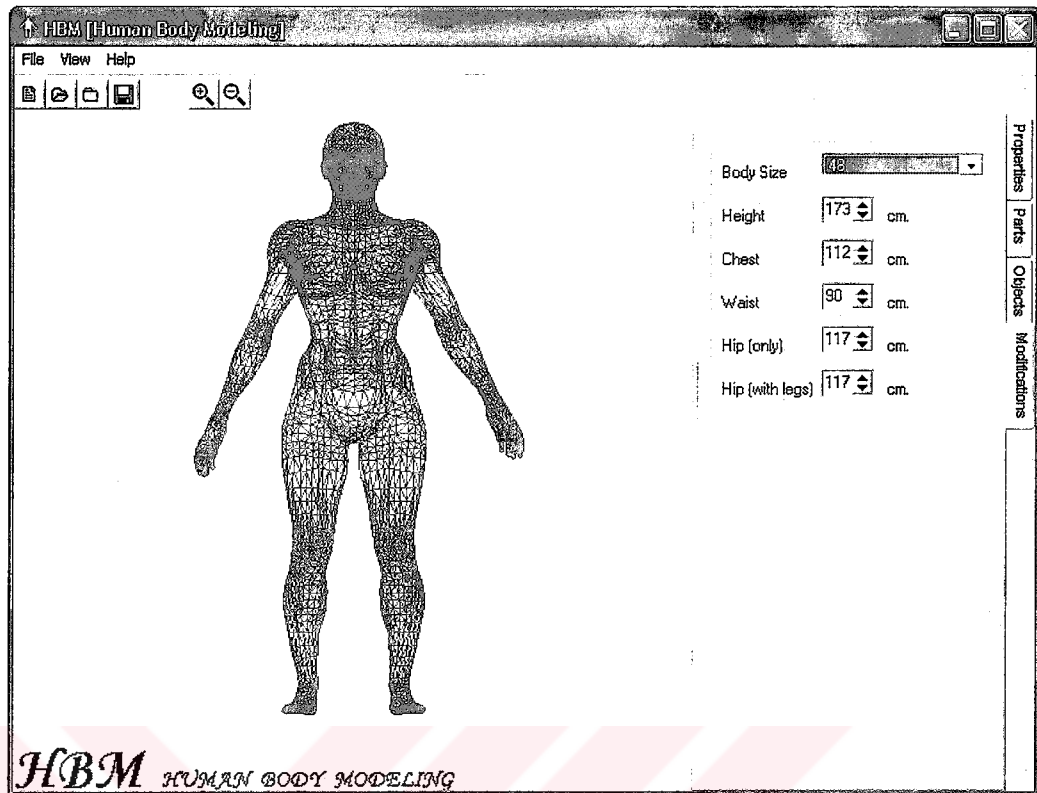


Figure 5.8 An example of human body model with 48 body size

The chosen body size sets the parameters according to the results of the English researches for normal woman body sizes. Table 5.1 shows the parameter values of possible body sizes. For example, when the user selects 36 body size, the height is set as 160cm., the size of the chest is set as 82cm., the size of the waist is set as 60cm., the size of the hip is set as 87cm..

Table 5.1 Normal body size values for women

		Body Sizes						
		36	38	40	42	44	46	48
Values (cm.)	Height	160	162	164	167	169	172	173
	Chest	82	87	92	97	102	107	112
	Waist	60	65	70	75	80	85	90
	Hip	87	92	97	102	107	112	117

Finally, About Form of the program is shown in Figure 5.9.

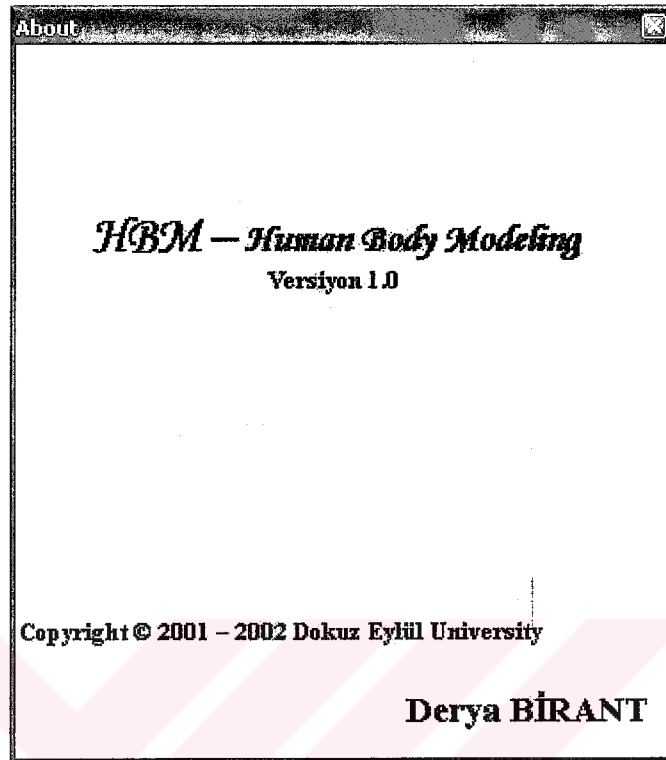


Figure 5.9 About Form of the program

CHAPTER SIX

CONCLUSIONS & FUTURE WORKS

6.1 Summary

This thesis performs on human body modeling systems. In other words, it is related about the construction of complete, three-dimensional representations of the normal male and female human bodies.

Human body modeling is the creation and manipulation of human computer models. The models provide realistic visualization of human beings. The models can consist of mathematical equations, procedural rules or graphical processes. There is an increasing demand for computer-based models in numerous applications; for example, medical research, clothing industry, virtual conferencing, realistic visualization for use in games, and lately e-commerce applications.

For reasons of the inherent difficulty of accurately modeling an entity as complex as the human body, the system requires improved computer graphics and modeling techniques, complex algorithms, faster machines and 3D scanners. In most applications, many stages are needed such as observation of human operator, scrutiny the data in order to remove artifacts, placing landmarks, fill in missing data, taking measurements. These processes are normally slow and tiresome. A significant design issue of such a system is to find a reasonable compromise between visual realism, ease of control and rendering speed.

For a number of years, Adrian Hilton et al. at Surrey University, the HUMAG (Human Measurement, Anthropometry and Growth) group (headed by Peter Jones) at Loughborough University, LIG/EPFL group (headed by Daniel Thalmann) and the MIRAlab group (headed by Nadia Magnenat-Thalmann) in Switzerland, Laura Dekker et al. at Computer Science Department of University College London have been working on the building 3D models of the human bodies.

In this study, a human body modeling system has been developed to define 3D models of the whole human body. The triangulated mesh technique is used for the representation of human bodies, which can be used in many different application areas. This technique is based on the definition of the skin as a collection of triangles and these triangles are modified by particular functions. Modifications by changing the parameters provide visually attractive results.

The developed system reads a new type file HBM (Human Body Model) as input to capture the data set. HBM is a special file for developed application. The usage of HBM files in human body modeling is more advantage than the usage of DXF file. HBM files have two main advantages over DXF files .

The first one is that the sizes of the HBM files are nearly fifth-fifth smaller than DXF files. The reason of this situation is that DXF files have many *special words* like SECTION, ENTITIES, 3DFACE, ENDSEC, \$EXTMAX, \$EXTMIN etc., and many *group codes* like 0, 8, 10, 20, 30, 50-59, 62 etc. These *special words* and *group codes* occupy a place in the memory. In opposite, HBM file only store the necessary attributes of the triangles.

The second one is that the load time of the HBM files are very very smaller than DXF files. The reason of this situation is that DXF files have many *special words* like SECTION, ENTITIES, 3DFACE, ENDSEC, \$EXTMAX, \$EXTMIN etc., and many *group codes* like 0, 8, 10, 20, 30, 50-59, 62 etc. The reading process of these *special words* and *group codes* takes many times. In addition, DXF file is read line by line. In opposite, attributes of the triangles are only read from the HBM files. In

addition, data set necessary for human body modeling is read from HBM files block by block. In order to take loading times, `GetLocalTime(SystemTime)` command is used in the program.

Defined data structure holds data sets which are in a common x, y, z coordinate frame. The attributes of each triangle provided by the system are *TriSelected* Boolean value, *TriPart* integer value, *Triangulating point sets* (floating values of x1, y1, z1, x2, y2, z2, x3, y3, z3).

The system shows a set of conceptual parameters to allow the users to modify the body size of the models. These parameters should be easy to understand for the ordinary user such as the length of the body, size of the height, chest, waist, hip etc. In order to reconstruct the model, the differences should be calculated between the current body size and the desired body size. After the calculation, changes are applied on the model by particular functions for each body part. While some parts of the body like foot can be modified independently of other parts, some parts are largely influenced by the adjacent parts like chest.

A simple 3D human body modeling program has been implemented for realizing our designed study. The program is developed under Borland Delphi 6.0 programming environment. OpenGL Graphic Interface Tool is also used to present computerized human models. A component, OGLPnl, is constructed to change the properties of the drawing window virtually. Events of this component consist of *AfterWGLCreate*, *OnClick*, *OnDblClick*, *nMouseDown*, *OnMouseMove*, *OnMouseUp*, *OnPaint*, *OnResize*. The methods of OGLPnl component are *Create* constructor, *CreateParams*, *Resize*, *Paint*, *ExceptionGL*, *ErrorCheck*, *Swap*, *WMCreate*, *WMDestroy*, *WMEraseBknd*, *CreateGLContext*, *DestroyGLContext* procedures.

The important functions and procedures for the triangles are *GetTriangle* function, *PutTriangle* procedure, *GetTriangleCount* function, *ClearTriangleList* procedure,

DeleteTriangle procedure, *InsertTriangle* procedure, *IndexOfTriangle* function, *FindTriangle* function.

6.2 Future Works

As a future work, some special operations can be done when representing humps, scars or other unusual objects on the skin is desired. There should be many triangles for satisfactory representation of these properties. In opposite case, the surface appears wrinkled or angular in an unnatural way. In order to solve this problem, triangles can be subdivided in smaller triangles.

Other possible improvement of the program can concentrate on more sophisticated visual details, such as textures and bump mapping.

Finally, the models generated in this study might be used in many applications such as textile applications, diet application etc. For example, generated models might be used in monitoring of size and scaling of human bodies (body shape analyses) on the web. It can allow users to monitor their bodies by entered their body sizes. In textile sector, such a system can enable the customers to visualize themselves in a garment before purchasing it. It provides ensure customer satisfaction and reduces their turnover rates, whereby customers can see exactly how the clothes will fit on them.

REFERENCES

- Blane, M.M., Lei, Z., Civi, H., Cooper, D.B. (2000). The 3L Algorithm for Fitting Implicit Polynomial Curves and Surfaces to Data. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Vol.22, No.3 pp. 298-313.
- Buxton B., Dekker L., Douros I., Vassilev T. (2000). Reconstruction and Interpretation of 3D Whole Body Surface Images. Scanning 2000 Proceedings, Paris.
- D'Apuzzo, N., Plaenkers, R., Fua, P., Gruen, A., Thalmann, D. (1999). Modeling Human Bodies from Video Sequences. Videometrics VI, Proceedings of SPIE, Vol. 3641, San Jose, USA, pp. 36-47.
- D'apuzzo, N., Plaenkers, R., Fua P. (2000). Least Squares Matching Tracking Algorithm For Human Body Modeling. International Archives of Photogrammetry and Remote Sensing, Amsterdam, The Netherlands, Vol. 33, Part B5/1, pp. 164-171.
- Dekker, L. (2000). 3D Human Body Modelling from Range Data. PhD Thesis, University College London.
- Dekker, L., Douros, I., Buxton, B. F., Treleaven, P. (1999). Building Symbolic Information for 3D Human Body Modeling from Range Data. Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling, In IEEE Computer Society, Ottawa, Canada, pp. 388-397.

Douros, I., Dekker, L., Buxton, B. F. (1999). Reconstruction of the Surface of the Human Body from 3D Scanner Data Using B-spline. Proceedings of SPIE on Three-Dimensional Image Capture and Applications, San Jose, California.

Douros, I. (2000). Developing Techniques for Building Active Shape Models from 3D Scanner Data for the Representation of Human Bodies. 1st-year PhD Report, University College London.

DXF Reference, (1999), Autodesk, Inc.

Ertürk, S. (1999). Spherical Harmonic Shape Representation: Analysis-Synthesis and Applications to Video. PhD Thesis, University of Essex, U.K.

Ertürk, S., Dennis, T. J. (1999). Efficient Representation of 3D Human Head Models. Proceedings of 10th British Machine Vision Conference (BMVC'99), Nottingham, UK, Vol 2. pp. 329-339.

Foley, J.D., van Dam, A., Feiner, S.K. Hughes, J.F. (1990). Computer Graphics: Principles and Practice. Addison-Wesley Publ., Wokingham, England, ISBN 0-201-12110-7.

Hearn D., Baker M.P. (1993). Computer Graphics, Prentice Hall, London.

Hilton, A., Beresford, D., Gentils, T., Smith, R., Sun, W. (1999). Virtual People: Capturing Human Models to Populate Virtual Worlds, Proceedings of the Computer Animation, In IEEE International Conference on Computer Animation, Geneva, Switzerland, pp. 174-185.

Hilton, A. (September 1999). Towards Model-Based Capture of a Person's Shape, Appearance and Motion. mPeople Workshop Proceedings, IEEE-ICCV99 (International Conference on Computer Vision), Corfu, Greece, pp. 37-44.

- Jacobs, J.Q. (1999). Delphi Developer's Guide to OpenGL. Texas: Wordware Publishing Inc., ISBN 1556226578.
- Jones, P.R.M., Brooke-Wavell, K., West, G. (1995). Format for Human Body Modelling from 3D Body Scanning. HUMAG Research Group, Loughborough University, International Journal of Clothing Science and Technology, Vol.7, No.1 pp. 7-16.
- Ju, X., Werghi, N. and Siebert, J. P. (2000). Automatic Segmentation of 3D Human Body Scans, Proceedings of IASTED Int. Conference on Computer Graphics and Imaging 2000 (CGIM 2000), Las Vegas, USA, Vol. 314.
- Kalra, P., Magnenat-Thalmann, N., Moccozet, L., Sannier, G., Aubel, A., Thalmann, D. (Sep/Oct 1998). Real-time Animation of Realistic Virtual Humans. IEEE Computer Graphics and Applications, Vol. 18, No. 5, pp. 42-56.
- Kut, A., Kasap, M. (2002). 3D Graphics in OpenGL with Applications. Dokuz Eylül Üniversitesi - Mühendislik Fakültesi Basım Ünitesi, İzmir.
- Lee, S. (1999). Interactive Multiresolution Editing of Arbitrary Meshes. Eurographics'99, The Eurographics Association and Blackwell Publishers, Vol. 18, Number 3.
- Li, P., Jones, P.R.M. (1994). Anthropometry-Based Surface Modeling of the Human Torso. HUMAG Research Group, Loughborough University, Proceedings of the 1994 ASME International Conference and Exhibition, Minneapolis, pp. 469-474.
- Nurre, J. H. (1997). Locating Landmarks on Human Body Scan Data. Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, Canada, pp. 289-295.

Ong, E., Gong, S. (1999). A Dynamic Human Model Using Hybrid 2D-3D Representations in Hierarchical PCA Space, Proceedings of 10th British Machine Vision Conference (BMVC'99), Vol. 1, pp. 33-42.

Plaenkers, R., Fua, P., D'Apuzzo, N. (1999). Automated Body Modeling from Video Sequences. Proceedings of the IEEE International Workshop on Modelling People, Corfu, Greece.

Rehg, J., Kanade, T. (1994). Visual Tracking of High DOF Articulated Structures: An Application to Human Hand Tracking. Proceedings of European Conference on Computer Vision, Stockholm, Sweden, Vol. 2, pp. 35-46.

Szijártó, G. (2001). Interactive Human Face Modelling. Proceedings of Central European Seminar on Computer Graphics (CESCG 2001), Bratislava.

Taylor, P.J., Shoben, M.M. (1984). Grading For the Fashion Industry - The Theory and Practice. Hutchinson & Co. Ltd., Londra.

Thalmann, D. (1993). Human Modelling and Animation. Eurographics '93 State-of-the-Art Reports.

www.metacreations.com, 30 March 2002, 1.10 pm.