AN APPROACH TO HUMAN SURFACE MODELLING USING CARDINAL SPLINES

by

Chin Wah Tony Chung

BSc, Simon Fraser University, 1984

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
February 1987

APPROVAL

Name: Tony C. W. Chung

Degree: Master of Science

Title of Thesis: An Approach to Human Surface Modelling using Cardinal Splines

Dr. Arthur Liestman, Chairman

Dr. Thomas Calvert
Senior Supervisor

Dr. Thomas K. Poiker

Dr. Binay Bhattacharya

Dr. John Dill
External Examiner

10 February 1987
Date Approved

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

AN APPROACH TO HUMAN SURFACE MODELLING USING CARDINAL SPLINES

Author:

(signature)

CHUNG CHIN WAH TONY

(name)

March 23, 1987

(date)

Abstract

This thesis involves the design and implementation of an interactive surface modelling system for the human body. The system provides an efficient and convenient way for an artist to specify the posture and the shape of a computer displayed three-dimensional human figure.

The human figure is represented as connected tubes, each tube consisting of connected surface patches and each surface patch being drawn using cardinal splines. The posture is specified by changing a set of transformation parameters which affect the orientation of each tube while the shape is specified by changing a set of control parameters which affect the geometric components of the cardinal splines. A hierarchical data structure for organizing the orientation and shape parameters is developed. This representation provides a smooth human body for animation and rendering programs.

The thesis also addresses the problems which arise when the body changes shape due to human motion and particular attention is given to the representation of the surface of a joint. An algorithm for displaying a bent tube around the joint is presented. Additional shape parameters are introduced to solve the problem of shape deformations as the joint is bent. The solution is to deduce the shape parameters for any orientation based on the shape parameters of a set of standard orientations. The standard shapes and orientations are built using the human surface modelling system.

Acknowledgements

There are many suggestions, help, moral support and effort from many people. In particular, they are my supervisor Dr. Tom Calvert, the committee, and the LCCR graphics team Gary Ridsdale, Nigel Protter, Susan Hewitt, Severin Gaudet and Victor Tso. Also special thanks to our dancer Clee.

Also thanks to department secretary Ethel and Robin for doing a lot of paperwork and LCCR system analyst Ed Bryant for helping in the use of SCRIBE and PICTOQMS.

Finally, I thank again my supervisor for correcting my grammar and style in writing this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Using a computer to generate human figures has many obvious advantages and there are many applications in almost all disciplines; however, the problem of modelling a 3-dimensional human figure in the computer is rather difficult. Human modelling is indeed an interesting and rewarding topic in computer graphics since it advances the state of the art in solid modelling, rendering, modelling of movement paths and programming techniques.

Human modelling involves two distinct aspects: 1. Movement Specification — from low level specification of joint locations to high level specification using natural language. 2. Shape Specification — from an unrealistic stick figure to a realistic picture of a smooth shaded solid human body. A realistic image of a human body can be drawn on a computer in the same way that an artist paints a picture, but in 2-D painting there is no easy way to specify a new posture for the figure or to view the figure from a new direction. This thesis describes an elegant design tool for the representation of the human body. This allows both simple movement specification and realistic display (See Fig. 1-1). The major objective is to display a smooth human body regardless of the posture of the figure.

After some exploration of old techniques, a new method of modelling a human body is presented. Basically, curved patches are connected together to form a human shape. The data structure for describing each patch and the way each patch is connected

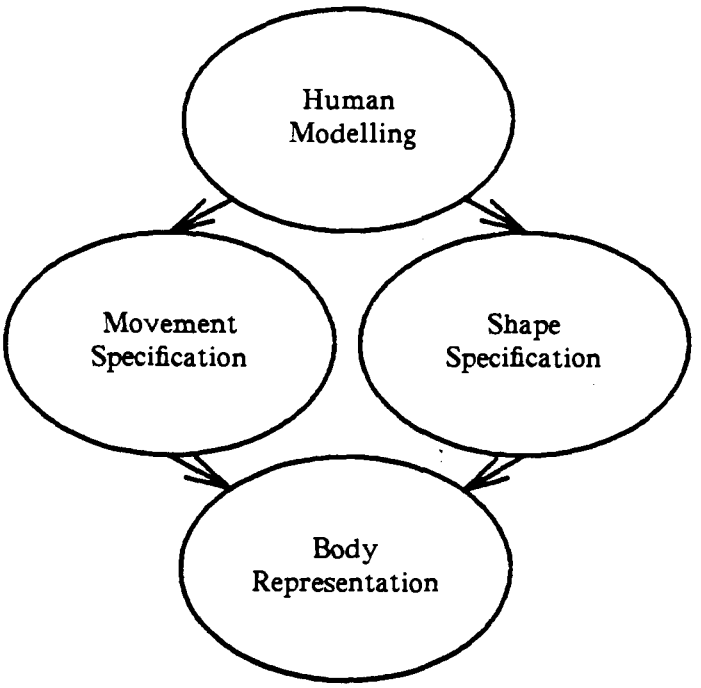


Figure 1-1: The Structure of the Human Modelling System

together constitutes the major design work. It is also important to consider the fact that the body changes shape as it moves, especially in the area around the joints. Thus the surface modelling method must provide a robust representation of the changing surface shape.

For the human surface modelling system described in this thesis, there are six phases of the software development cycle: requirements, specification, design, implementation, verification and maintenance (See Fig. 1-2). Requirements are the desired properties of the system; specification defines the input, output and error handling plus the functional and data abstraction of the system; implementation and verification are the processes of coding and testing the design (description of algorithm and data structures); maintenance is any change or improvement after the system is released.

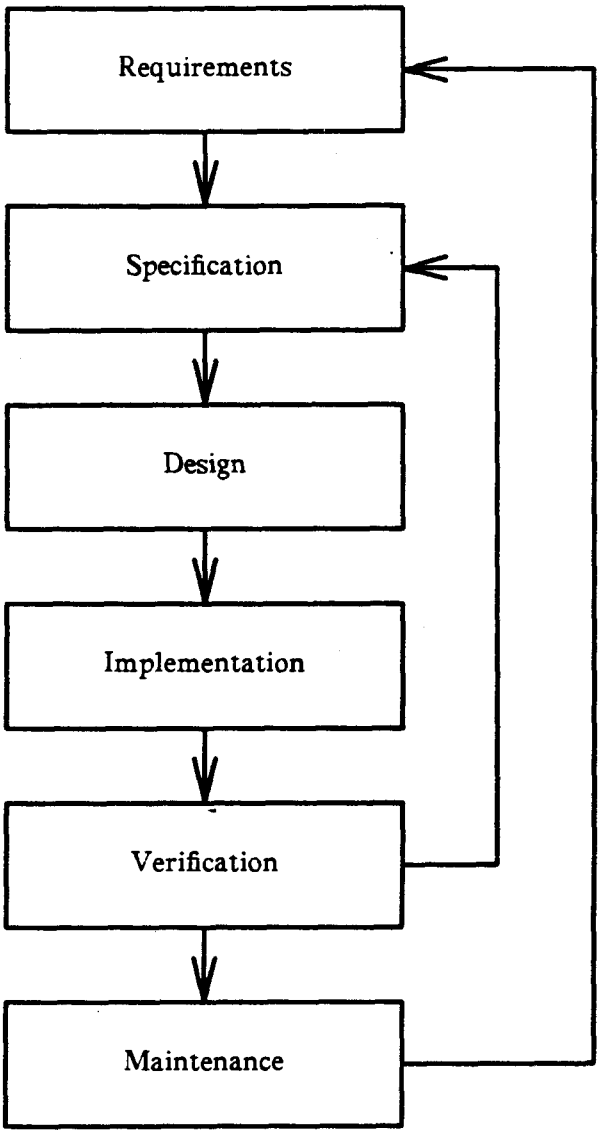


Figure 1-2: Software Development Cycle

After some extensive testing, the system may need to be re-specified to achieve a better design. Also, as the user requests more capacity in the future, the requirements may be change. Then the whole system will be needed to re-organized. This thesis describes each phase and the analysis from one phase to another.

Chapter 2

Requirements of a Human Body Modelling System

2.1. Overview

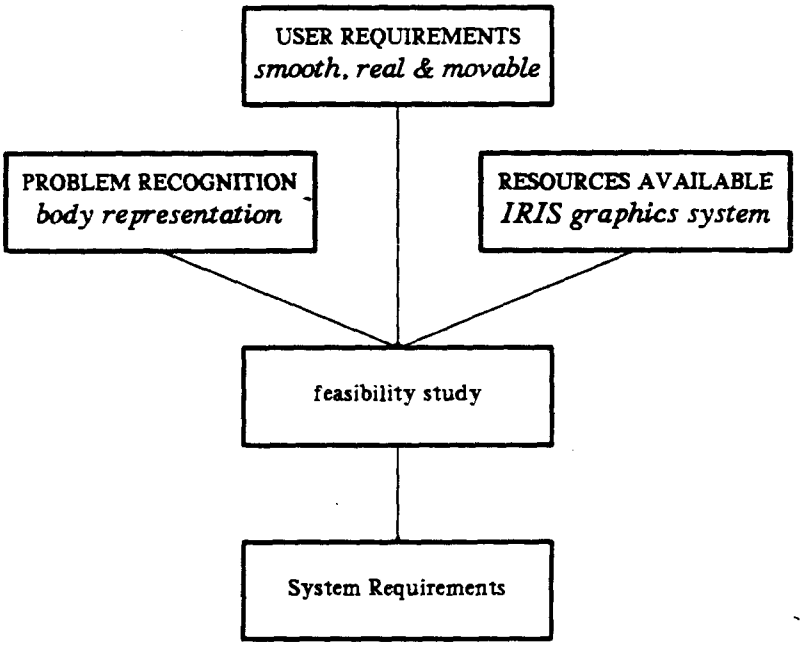


Figure 2-1: Task Analysis

Task Analysis (Figure 2-1) involves interviews with the users and understanding their needs and problems. Then, based on the constraints of time, money and people, a subset of the most important requirements are identified and implemented.

2.2. Analysis of user models

There are many applications for computer generated human bodies. These occur in the fields of medicine, kinesiology, fine arts, biomechanics, man-machine systems design and education among many others. A complete human modelling system aims at serving the following three closely-related user groups (see Fig. 2-2) : people that study human movement, people that study human representation and people that study human-environment interactions.

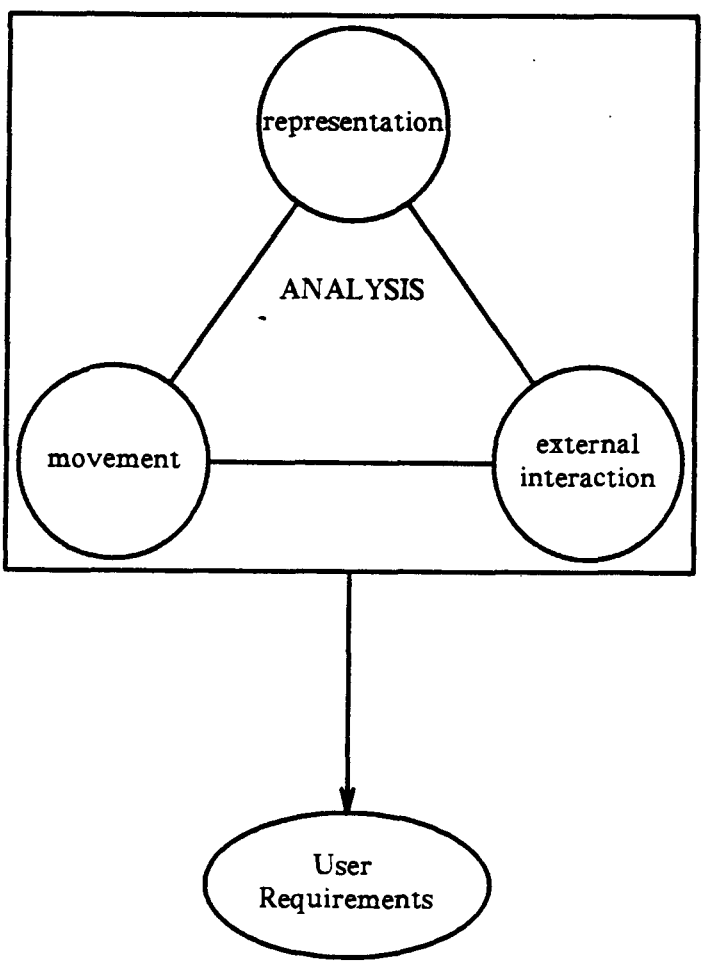


Figure 2-2: Analysis of User Models

2.2.1. Human movement

Physiatrists, physiotherapists, dancers, sports coaches and kinesiologists among others study human movement and expect the computer to help characterize performance. They require the ability to specify a posture that is restricted by physical human constraints. Inputs may be digitized with the ability to edit them later. They want to store the sequences of movement, re-display them with any speed and analyse them from different viewing positions. The exact measurements of body positions and joint locations are important [Calvert 82].

The importance of recording sequences of motion is particularly clear to dancers [Ridsdale 86]. Since there are so many types of dances and there are so many alternate tools or notation systems to record them, there is a need to use the computer in a natural and effective way to model movement. It may also be cheaper to use the computer to plan out the movement of a dance to be performed by live dancers.

2.2.2. Human representation

In computer graphics, scientists would like to have a representation of the human body to illustrate the techniques of rendering and movement transformation. In computer vision, the goal is to identify an object [Pentland 86], and this requires an unambiguous representation of the human body to distinguish it from other objects.

Art, education and entertainment can also involve some kind of animation.[1] Scenes that are impossible to create and motions that are impossible to perform can all be

---

[1] communication of an idea that deviates from the exact model [Parke 82].

simulated by using the computer. It is obvious that animated pictures, possibly in the form of video, can have a better effect on the customers or audience than still pictures. In all of these applications it is important to properly represent the human body.

2.2.3. Human-environment interaction

Study of many real-world situations (like crash studies) requires some sort of interaction of the human body with the external environment [Badler 78]. It is obvious that it is much safer to simulate a human when studying a dangerous environment; it may also be more economic. Calculation of how the human body reacts when external forces are applied is difficult but of interest in biomechanics. An exact model of a human body is needed here in order to extract useful information from simulation and dynamic analysis [Armstrong 86].

2.3. Problem Recognition

The human body can be modelled as an articulated structure of segments. There are two difficulties in human body shape modelling [Badler 79a]:

1. The shape of human body is irregular, therefore the human body cannot be composed of simple geometric primitives like spheres, ellipsoids or polygons which the computer can work with efficiently. The body requires a robust representation; this needs massive memory and complicated operations which are time-consuming and subject to computation error.

2. The body shape changes as the body parts move relative to each other. Thus the human body, in addition to being a relatively complex articulated structure, is not rigid. Thus the problem is more difficult than modelling an automobile or a robot arm.

2.4. Evolution of Human Body Modelling

2.4.1. Stick Figure

A stick figure is always the first step in developing any human modelling system. The only required information is the joint locations and the way the segments are connected together. The segments are straight lines and serve as a skeleton or basis for future development. The only merit of this representation is its simplicity which is important for real-time display and development; however, it has many obvious disadvantages.(see [Badler 82], [Badler 78]) Therefore, solid modelling is needed.

2.4.2. Solid Modelling

The problem of modelling the human body is an example of the more general solid modelling problem which is important in computer-aided design (CAD), computer graphics and computer vision. Techniques for solid modelling have been treated extensively in the literature. A recent review by Requicha [Requicha 80] summarizes the following approaches:

1. Pure primitive instancing,
2. Spatial occupancy enumeration,
3. Cell Decomposition,
4. Constructive Solid Geometry (CSG),
5. Sweeping,
6. Interpolation,
7. Boundary Representation (BR).

Among them, the CSG and BR methods are considered to be the most important for solid modelling in general and for human body representation in particular.

2.4.3. Constructive solid geometry

CSG is a binary tree representation in which the two *sons* are geometric solids while the parent node is a boolean operation like union, intersection or difference. Each *leaf* is a simple geometric primitive; these include spheres, cylinders, ellipsoids [Herbison-Evans 82], polyhedrons, superquadrics [Barr 84] or other geometric objects.

Generally the body segments are formed from a union of these primitives. The sphere is the most successful primitive [Badler 79b], since it is easier to compute and manipulate. Its projection on a screen is always a circle. If the center of sphere is the center of a joint, it simplifies the modelling of joint movements. However, in order to improve smoothness, the number of spheres needs to be large; and the resulting surface is still bumpy. As pointed out earlier, no nameable[2] geometric object can represent an irregular solid efficiently and properly.

2.4.4. Boundary Representation

BR uses vertices, edges and faces to model a solid. A wire-frame model is a special case of BR.

The simplest and most common way to represent human surfaces is by approximating them with planar polygons and applying shading techniques to make the connected edges of the polygons look smooth [Newman 79]. This requires a large number of polygons and the shape can deform during joint movement in a way which is highly undesirable [Catmull 72].

Surface patches can be used instead of polygons to represent the human shape. One

---

[2] sphere, box, ellipsoid .... etc

major advantage is that a relatively small number of patches are required and thus it is easier to manipulate them; however, hidden surface elimination and shading are much more difficult [Catmull 75] [Rogers 76].

2.5. Modelling solids which change shape

Relatively little attention has been given to the modelling and representation of solids which change shape over time. In CAD/CAM for example, very complex models are used for machinery (e.g automobiles and airplanes) but with very few exceptions, the individual components of the model do not change shape. Thus, modelling the human body presents a unique challenge.

Most of the human modelling systems have ignored the changes in shape. Current approaches in human modelling concentrate on modelling individual limb segments and ignore discontinuities at the joints. The only simple method which gives a reasonable result is the bubble-figure approach of Badler and his colleagues [Badler 79b]. This allows a reasonably good model for the whole body if a sufficiently large number of spheres are used (over 800 in our bodies and over 2,000 in the bodies by Badler's group). But this approach does not provide for any change in limb segment shape.

The obvious solution is to calculate an individual surface patch to model a joint at every possible joint orientation. Related work has been carried out by those investigating the automatic creation of blending surfaces for connecting mechanical parts in CAD. A blending surface is a patch that smoothly connects two surfaces. Figure 2-3 is a projected view of two cylinders and the blending surface.

There are many approaches for creating blending surfaces. One approach by Rossignac is the use of a canal surface [Rossignac 85]. A canal surface is the

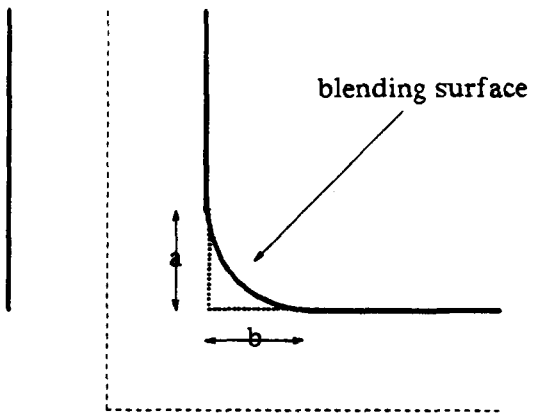


Figure 2-3: Blending surface

envelope of a family of spheres that move along a space curve while maintaining contact with the two surfaces to be blended. Another approach by Hoffmann and Hopcroft is the potential method [Hoffmann 85]. This method can be illustrated by an example; in figure 2-3, the blending surface is an ellipse $\frac{(g-a)^2}{a^2}+\frac{(h-b)^2}{b^2}-1$. Here, g $= x^2 + y^2 - r^2$ and h $= y^2 + z^2 - r^2$ are the surfaces of the two cylinders. The constants a and b control the width of the blending surface. The resulting polynomial is of degree 4 compared to that for a canal surface which results, in the simplest case, in a polynomial of degree 16. A further extension is the projection potential method which yields blending surfaces not obtainable by the simple potential method.

Their methods are based on algebraic implicit functions[3] which are difficult to

---

3

$$\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n} a_{ijk}x^i y^j z^k = 0$$

compute and subject to computational error [Goldman 83]. Also, their works only consider simple geometric objects (quadric surfaces) and for limited orientations (perpendicular and co-axial).

The best way to create a free-form surface is to use parametric functions [Clark 81]. Eight advantages are listed in [Mortenson 85]. One of the most promising parametric functions is the non-uniform rational B-spline [Tiller 83].

$$S(s,t) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} B_{i,k}(s)B_{j,l}(t)\,h_{ij}\,p_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} B_{i,k}(s)B_{j,l}(t)\,h_{ij}} \qquad 0 \leqslant s \leqslant 1,\ 0 \leqslant t \leqslant 1$$

The p_{ij} are called control points which form a convex hull enclosing the surface. The $B_{i,k}$ is the ith-basis polynomial of order k (degree k - 1) while the $B_{j,l}$ is the j^{th}-basis polynomials of order l (degree l - 1) and they are called blending functions. The h_{ij} are called homogeneous coordinates[4] which allow the representation of circular arc and straight line as opposed to ordinary non-rational splines. A blending function is recursively determined using the Cox-de Boor algorithm [de Boor 78] based on a given knot vector[5] . The introduction of a non-uniform[6] knot vector gives a more flexible and powerful tool in designing a surface and subdividing a patch. However, it is not really that intuitive for a user (without any mathematical background) to interactively change shape parameters other than the control points (that can be

---

[4]Analogy to homogeneous space in which if a point in 4-D is (hx,hy,hz,h), then the projection to 3-D is by dividing h to get (x,y,z).

[5]$\{t_q\}$ such that $0 = t_1 = t_2 = \ldots = t_k < t_{k+1} \leqslant t_{k+2} \leqslant \ldots \leqslant t_m < t_{m+1} = \ldots = t_{m+k} = 1$

[6]$\{t_q\}$ is called uniform knot vector if there exists some positive real number d, such that t_{v+1} - t(v) = d for all $k \leqslant l \leqslant n$, otherwise, it is nonuniform.

displayed). Also, the response time is slow unless special hardware is designed. A much simpler method uses cardinal splines which are less smooth; however, for the shape of a limb segment, they are sufficient. Cardinal splines will be explained in more detail in Chapter 4.

My approach is similar to that used by Parke for modelling the human face [Parke 82] [Pearce 86]. Parke used more than 50 numerical parameters to describe the rotation of jaws, length of the chin and other facial features. The movement is done by interpolation of two extreme shapes of facial features. My approach use cardinal splines instead of polygons and has less parameters. Also, the parameters are more intuitive in describing the shape especially around the joint.

2.6. Summary

The desirable properties of a body modelling system are:

1. A user-friendly interactive environment

    - input and control should be simple, easy to understand and flexible.
    - real-time feedback and a help facility should provide useful information displays at all times.

2. A flexible viewing capacity

    - the ability to look at the body model from any positions.
    - the ability to have multiple viewports.
    - the ability to display a 3-D smoothed human figure.

3. Posture Specification[7]

---

[7]Note: this is different from Movement Specification which involves notation, duration of motion, ... etc.

- the ability to define a new posture for the figure by flexing the limbs interactively

- the ability to read from a description file to change the posture

4. Shape Specification

- after flexing the limbs, the shape around the joint should alter smoothly without any gaps or unusual features.

- the ability to interactively change the local shape of the figure to improve realism.

Note that many body modelling systems can more or less satisfy the above properties. An example is a hybrid CSG system that uses spheres and ellipsoids as primitives. It will allow the user to re-position the center of any sphere, to change the number of spheres and to alter the size of each sphere. Then an ellipsoid can be used to enclose a segment to improve smoothness. We have chosen to use a system with bi-cubic surface patches where the user can change the parameters of the mathematical formula to achieve the necessary shape (This will be briefly explained in Chapter 4). Chapter 3 will describe *the body modelling system* which provides the user with a tool to model a perfectly smooth body.

Chapter 3

The Specification of the Human Modelling System

3.1. Overview

To specify a graphics system, we need to partition it into modules [Carson 83], describe the relationships between the modules and describe the functional behaviour of each module. For a graphics system, one of the most important elements is the user interface [Green 81] [Carey 82]. This thesis is about a 3-D interactive modelling system, so the user control and system feedback require careful consideration.

The specifications of the graphical objects, the layout of the screen and the user interface will be described in this chapter; all of these result from extensive testing and analysis. The analysis will be postponed until Chapter 5 "*Implementation & Verification*" Section " Development of the User Interface ".

3.2. System Overview

The system is partitioned to be three sub-systems:

- DBMS- Data Base Management Sub-system.
  - files I/O : load or save a data tree from or to a file.
  - access : retrieve information from a node.
  - update : update information for a node.
  - setup : transforming a data tree to graphical objects.
- UIMS- User Interface Management Sub-system.
  - layout : control the number of viewports and their sizes.
  - pick : map the data to an input device so that graphical objects can be identified and their parameters can be updated.
  - help : provide information and system feedback.
- Application Sub-system.
  - model builder : edit and design the shape of the human body.
  - posture fixing : arrange the posture of the figure.
  - viewing specification : control all the viewing parameters.

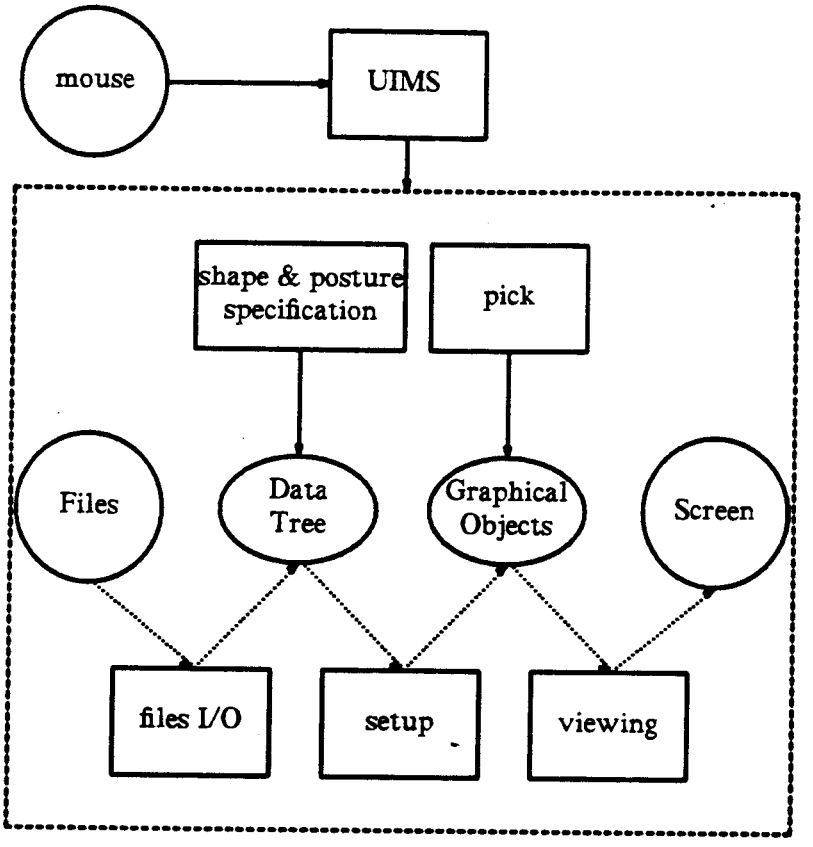


Figure 3-1: System architecture

3.3. Graphical Objects Description

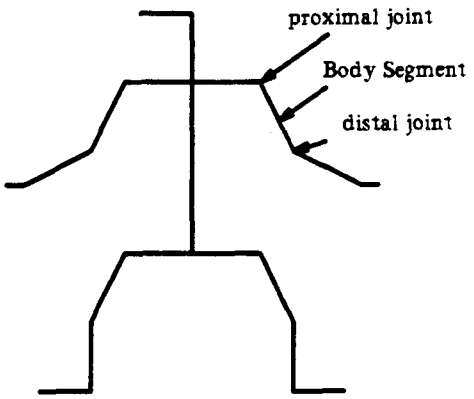


Figure 3-2: An articulated structure made up of segments

The human body is displayed as an articulated structure of white lines (called the segments) with a grid of closed curves enclosing the whole figure. The grid lines are vertical (parellel to the segment) curves which intersect with horizontal (perpendicular to the segment) closed curves (called the *rings*). Also, white dots (called the control points) are dispayed at the intersections of the curves. For better viewing, two colors are chosen for the grid of closed curves. The curves at the front of the segment are light blue while the curves at the back are light red.

The grid outlines the surface of a true human being. Initially, the displayed body segment is like a cylinder with six vertical curves and six *rings* per segment; however, it can be interactively changed to look like a realistic human body.

A more formal description of the graphical objects follows:

geometry

segment : 1. proximal end (x,y,z)
2. distal end (x,y,z)

ring: 1. size (radius)
2. center (x,y,z)

conrol-points : 1. (x,y,z)

relationship

A *ring* is a closed curve connecting a list of control-points.

A segment consists a list of *rings*.
The center of each *ring* is along the axis of the segment.
Let p_{ij} be the j^{th} control point of the i^{th} *ring*.
Then there exists curve that connect p_{ij} and $p_{i+1,j}$ for all possible values of i,j.

A human is an articulated object of segments.
The proximal end of a segment is equal to the distal end of the previous segment.

orientation (rotation along x, y and z)

segment : the origin of the coordinate system is at the proximal end with the z-axis parellel to the previous segment.

ring: the origin of the coordinate system is at the center with the z-axis parellel to the current segment.

3.4. User Interface Design

3.4.1. Screen Layout

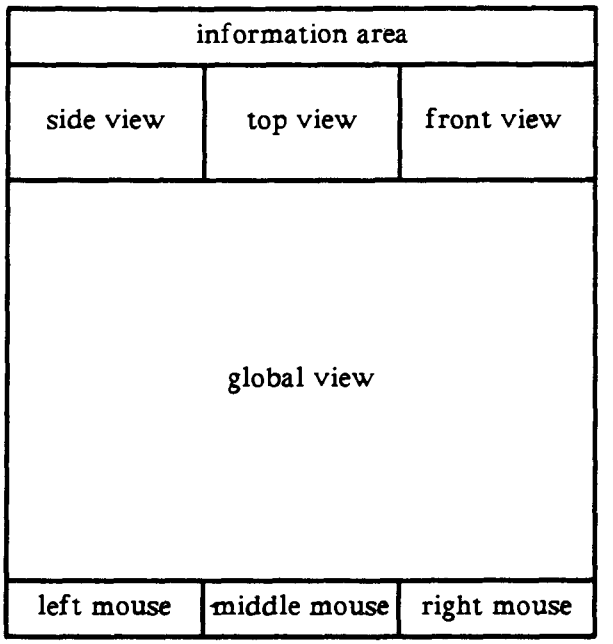


Figure 3-3: Layout of the viewports

The top area is for information display while the bottom area is to inform the user of the effects of holding the mouse buttons. The side, front and top views are orthogonal projections while the global view is a perspective projection.

The background is in light green to be comfortable to look at.

3.4.2. Description of each Mode

There are several modes for the system. Basically, they are divided into four categories namely, viewing, shape modelling, posture specification and utilities.

Viewing

- CHANGE VIEW - For orthogonal viewports, this mode is for changing the view by translating the whole figure. In the global viewport, this mode is for changing the eye position while looking at the origin.

- MOVE OBJECT - This mode is to rotate, translate and scale the whole figure so that different part of the figure can be exposed to the viewer.

Utilities

- PICK - This mode is for selecting options and for picking a graphical object (either a segment, a *ring* or a control point).

- HELP - This mode is for displaying information and for explaining how to use the system.

Posture specification

- MOVE JOINT - this mode is for changing the orientation of a picked segment. Note that when the orientation of a segment changes, all segments attached through the distal joint will need to be updated. For example, when the elbow is moved, both the lower arm and the hand will be displaced.

- POSTURE I/O - This mode is for loading a pre-defined posture or saving the current changed posture.

Shape specification

- DRAG POINT - This mode is for changing the local shape by dragging a control point in 3-D space.

- CHANGE SEGMENT - This mode is for changing the global shape of the segment to make it either longer or shorter and either thinner or fatter (changing the diameters of the *rings*).

- MOVE RING - This mode is specially designed for modelling the joint. A user can tilt the *ring* and move the center of the *ring* more toward the joint to get a better shape at the joint. Fig 3-4 is a projected view of two segments with *rings*. It resembles the flexing of limbs. Tilting the *rings* at the joint eliminates the overlapping of control points.

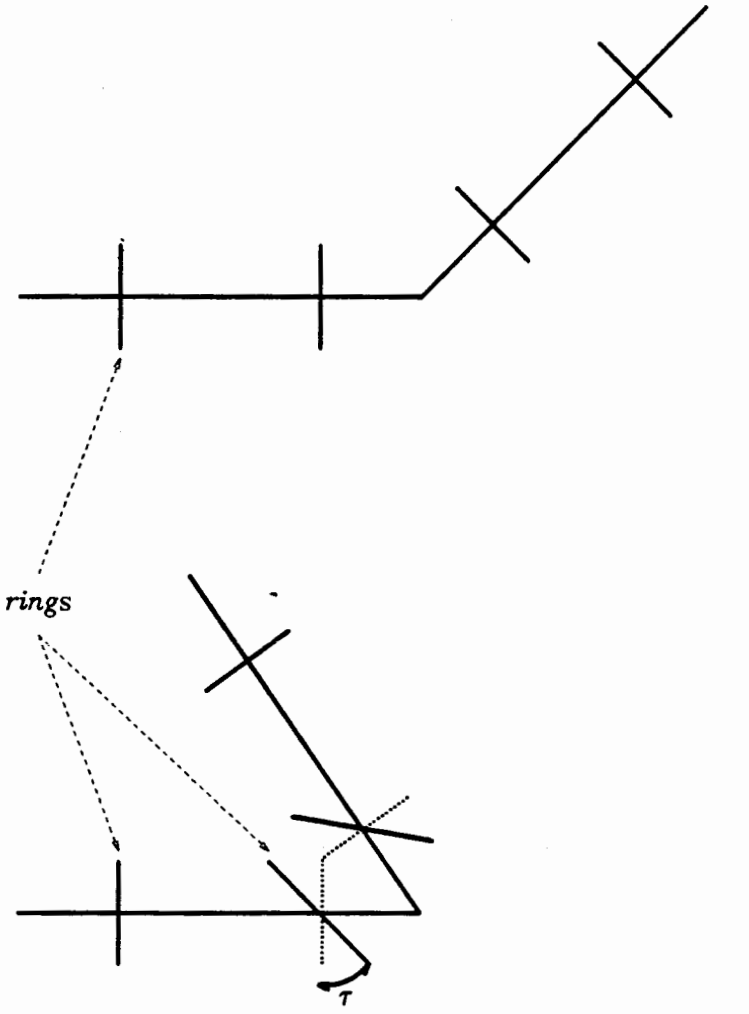


Figure 3-4: Tilt effect on the *rings*

- CHANGE RING - This mode is for changing the local shape by expanding the *ring* to have a local bumpy shape effect while rotating the *ring* to have a twisting effect. ω is the angle of twist.

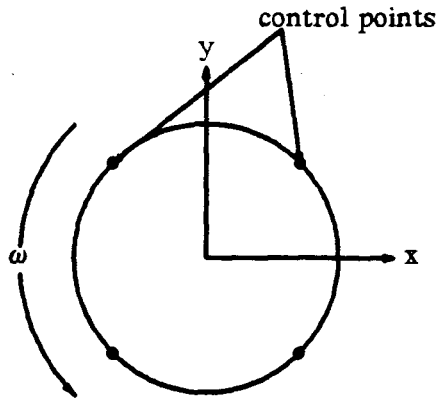


Figure 3-5: Twist effect on a *ring*

- SHAPE I/O - This mode is for loading a pre-defined shape for the current orientation or saving the current shape after it has been changed.

3.4.3. Control & Flow Mechanism

The control device is a mouse with three buttons on it. The left mouse button is for control while the right and middle mouse buttons are for complementary actions (for example, to zoom in and zoom out). The control button is used to activate a pop-up menu for changing mode and for picking graphical objects. A graphical object is picked by pointing the cursor (by moving the mouse) at the desired object and pressing the control button. Also a pop-up menu will be displayed with different menu-items (see table 3-1) depending on what graphical object is picked. In order to enter a mode, a user has to move the mouse until the desired menu-item is highlighted and then hold down the control button. The menu will disappear and the user has to continue holding down the control button in order to remain in that mode. By releasing the control button, the user is returned to PICK mode.

cursor position	menu items	mode
at a *segment*	move joint edit shape	MOVE JOINT CHANGE SEGMENT
at a control point	drag point edit shape move ring	DRAG POINT CHANGE RING MOVE RING
inside a viewport		CHANGE VIEW
at any boundary	posture I/O shape I/O help	POSTURE I/O SHAPE I/O HELP

Table 3-1: The effect of holding the control button

In general, for any mode the complementary buttons and motion of the mouse are used to change the parameters (geometric properties and orientations) of a picked graphical object. The values of the updating parameters will be displayed on the information area while the effects of pressing the buttons in this mode will be displayed on the bottom area.

3.4.4. System Feedback

The following explains the cursor shape which is the major system feedback:

hourglass - time-consuming process, please wait.
'X' - implies file does not exist.
'R' - implies file is readable.
box - object is affected by the motion of the mouse.
arrow - cursor can be moved freely to point at an object.

The 'arrow' shape is for PICK mode, the 'box' shape is for other modes while the 'X' and 'R' shapes are for the I/O mode.

During I/O, the cursor position is mapped to a data file which the user can write and read. So if the cursor is 'X' shape, the user knows he cannot load data.

Chapter 4

Detailed Design

4.1. Bi-cubic Surface Patches

Any point on a bi-cubic parametric surface patch can be determined using three independent bi-cubic polynomials for each of x, y and z.

In algebraic form a bi-cubic polynomial p is expressed as:

$$p(u,v) = \sum_{i=1,j=1}^{4} a_{ij} u^{4-i} v^{4-j} \quad 0 \leqslant u,v \leqslant 1$$

By specifying 16 constraints, the 16 a_{ij}'s can all be determined. The algebraic form can be transformed to matrix tensor form (see [Clark 81] [Mortenson 85]):

$$p(u,v) = [u^3 \; u^2 \; u \; 1]\,\mathbf{M}\,\mathbf{G}\,\mathbf{M}^t\,[v^3 \; v^2 \; v \; 1]^t$$

where

$$\mathbf{G} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix}$$

G is called the geometric matrix which controls the shape of the surface. The p_{ij}

are points in 3-D space[8] called the control points.

M is another 4x4 matrix called a basis matrix which determine how G controls the shape. Cardinal spline is chosen such that

$$\mathbf{M} = \begin{bmatrix} -a & 2-a & -2+a & a \\ 2a & -3+a & 3-2a & -a \\ -a & 0 & a & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The cardinal matrix is derived from solving the following constraints for a parametric cubic curve[9] with four control points $\{p_1, p_2, p_3, p_4\}$:

$C(0) = p_2$

$C(1) = p_3$

$C'(0) = a(p_3 - p_1)$

$C'(1) = a(p_4 - p_2)$

The constant a controls the tightness of the curve (as a increases, the tangent vector at the corners increases). The common practice is setting $a = 0.5$.

The advantages of using a Cardinal Spline are:

1. The control points lie on the surface (p_{32}, p_{22}, p_{23} and p_{33} are the endpoints of the surface patch (Figure 4-1)):

---

[8] this implies that for implementation, there should actually be three matrices : G_x, G_y and G_z.

[9] $C(t) = [t^3 t^2 \; t \; 1]\mathbf{M}[p_1 \; p_2 \; p_3 \; p_4]^T \qquad 0 \leqslant t \leqslant 1$

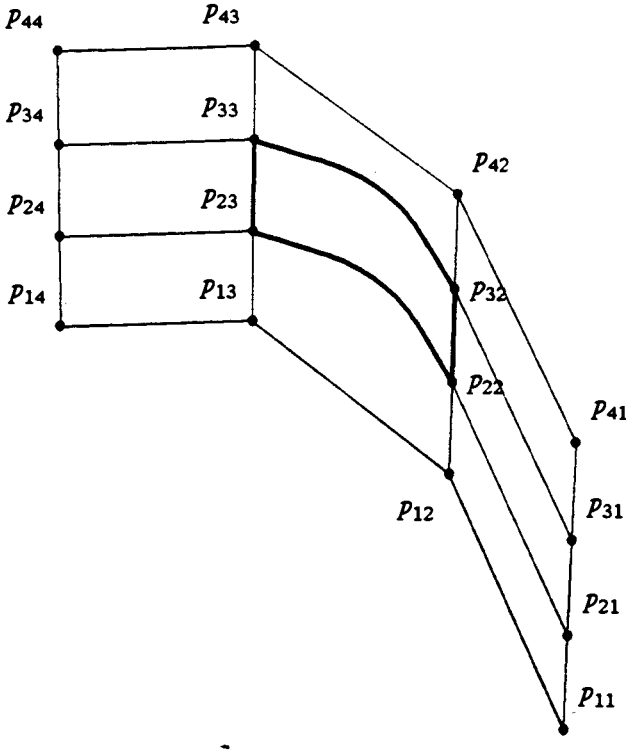


Figure 4-1: A patch defined by 16 control points

- they are more intuitive to look at and pick.
- they can be digitized directly from a human body.
- they can be used directly for approximating the body as the vertices of many polygons.

2. Two patches can be connected easily by sharing control points. Note that at the common boundary of two patches, their tangents are equal.

3. The normal vectors at the corners are easy to compute. This is useful for rendering algorithms.

4. A B-spline will decrease the number of control points; however, since a human body is not perfectly and regularly smooth, a B-spline will not improve very much. Also, the control points of a B-spline do not lie on the surface which makes the scene more messy to look at in comparison to a cardinal spline. The user interface specification only describes the

intersections of curves; thus a user does not have to have knowledge about splines.

The whole surface of a human figure can be represented by connected patches. Each patch is defined by a 4x4 array of control points. Thus, connecting patches together requires careful re-positioning and the sharing of many control points.

4.2. Data Abstraction

The data structure for the representation of a human body is a tree:

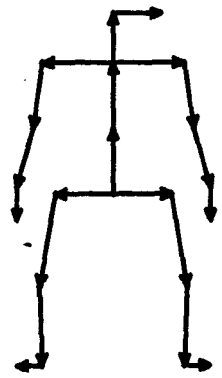


Figure 4-2: The human body as a tree

The terminology of graph theory will be used from here on. The human is a directed *tree* such that every segment is an *edge* and every joint is a *node*.

A tree can be implemented by using two links. One link is from parent to son and through the son to descendents. The other link is from the first son to the last son.

There are all sorts of algorithms or manipulations possible for trees; for example see [Tremblay 76]. Consider a recursive algorithm for Pre-order Traversal of a tree:

```
algorithm traverse node
begin
  if (node is not  LEAF) then
     begin
        process all the sons of this node.
        traverse  all the sons of this node.
     end
end.
```

One possible *process* is to retrieve the transformation matrix for this node and apply the matrix multiplication to each son. This has the effect of transforming all points in object coordinates to world coordinates (see Figure 4-3):

$$[x_w\ y_w\ z_w\ 1] = [x\ y\ z\ 1]M_d{*}M_{d-1}\ ....M_1$$

where (1,2,....,d-1,d) is a path from root to node at depth d and M_i is the transformation matrix for the i^{th} node. M_i transforms the object coordinates of the i^{th} node to the object coordinates of the $(i-1)^{th}$ node (the parent of the i^{th} node).

For saving space and better control, no matrix is stored. Only the parameters that set up the transformation matrix are stored. Using C-syntax the data structure is as follows:

```
struct  node {

  struct node down;       /* down-link to son;*/
  struct node next;       /* cross-link to brother; */


  Object primitive;       /* graphical object */

                          /* matrix parameters;*/
  float tx, ty, tz;       /*    translation */
  Angle rx, ry, rz;       /*    rotation    */
  float sx, sy, sz;       /*    scale       */
  }
```

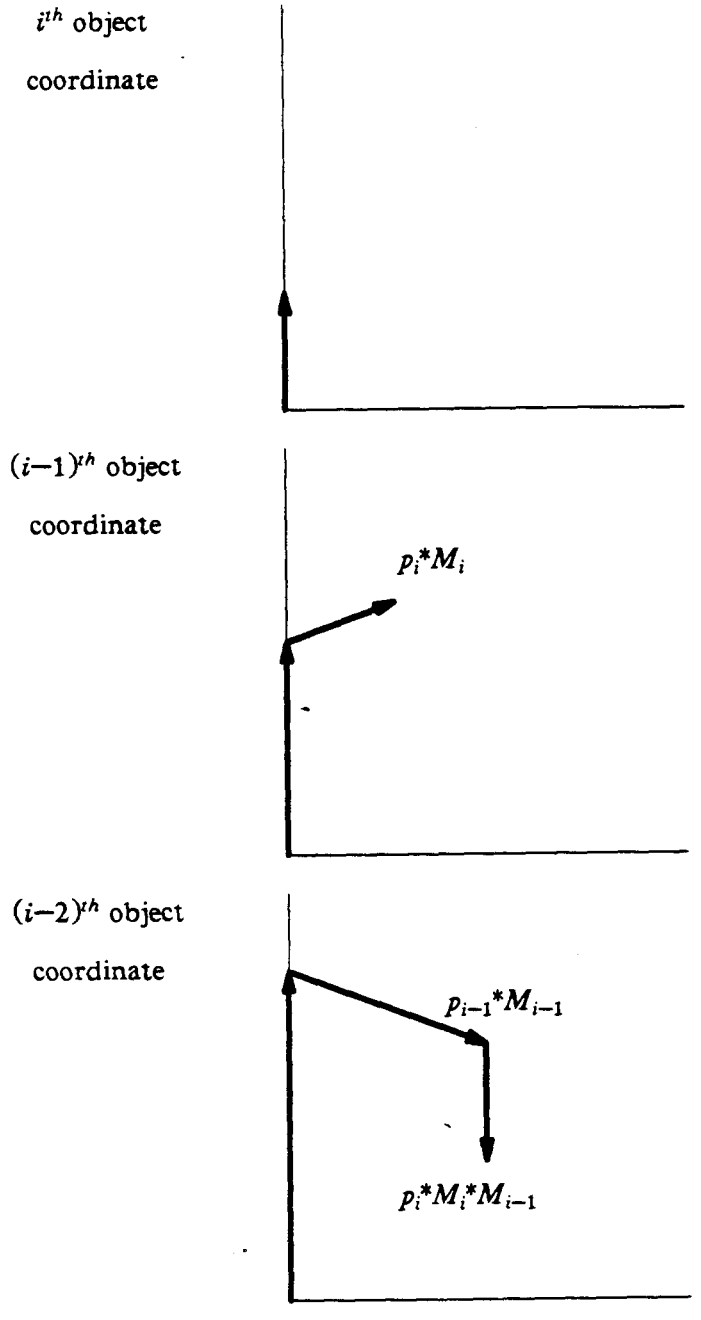


Figure 4-3: Matrix transformations of segments

A segment is also a tree since it has a hierarchical data structure:

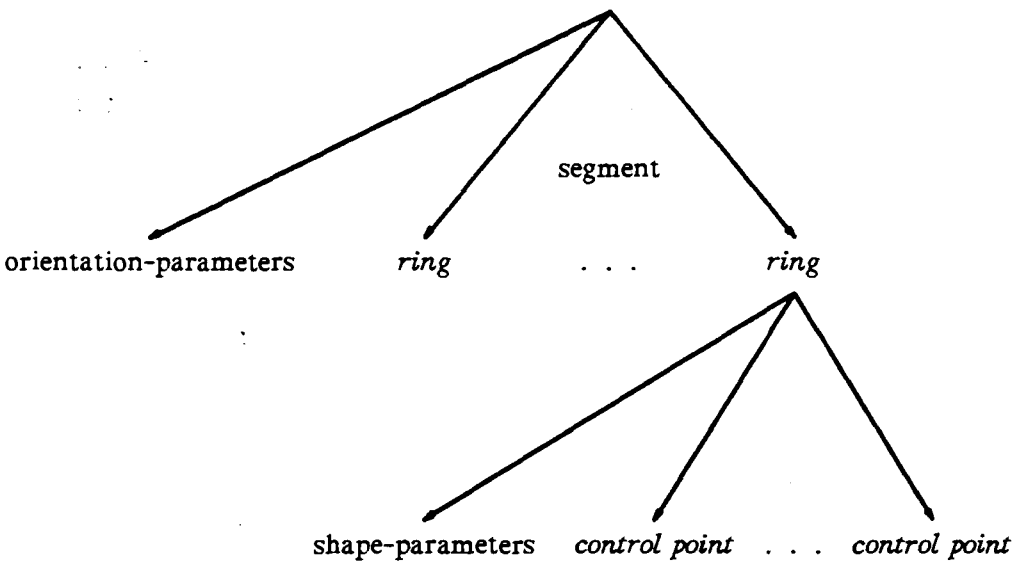


Figure 4-4: Segment as a tree

A larger tree is formed as we stick these sub-trees of "segments" to each node of the tree in fig 4-2.

Note there are three node types : segment, *ring* and control-point. Each node has its own object coordinate system and the transformation of the current object coordinate system to the parent coordinate system is done as follows: A control point is at the origin in its own object coordinate system and is translated to the *ring* coordinate system by (tx,ty,0). Then every points in the *ring* coordinate system are transformed to the segment coordinate system by (0,0,tz).

Before translation, a *ring* can be subject to scaling and rotation. The (rx,0,rz) specifies the tilt and twist of the *ring* while the (sx,sy,0) specifies the size of the *ring*. Then the translation (0,0,tz) specifies the center of the *ring* along the line segment.

In the object coordinate system of a segment, there is a `line` from (0,0,0) to (0,0,z) and a bunch of transformed control points. These control points are used to create the graphical primitive `tube` (see Fig. 4-4). A `tube` is a pipe of connected bands and each band is a ring of connected patches similar to the shape of a rubber band.

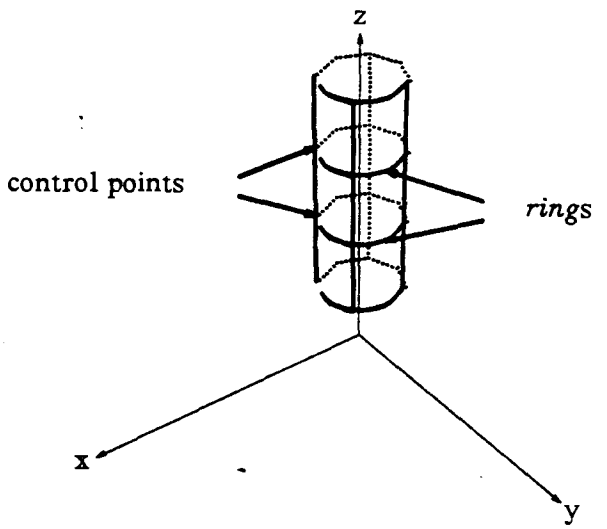


Figure 4-5: A *segment* in an Object Coordinate System

The corresponding primitive and matrix parameters for a node type are summarized as follows:

Node Type	corresponding primitive & parameters	
segment	line, tube	rx,ry,rz
ring	-	rx,rz,sx,sy,tz
control-point	point	tx,ty

The rx,ry and rz are the orientation-parameters. The rx, rz, sx, sy, tz, tx and ty are the shape-parameters since they specify the location of a control-point which affects the shape of a *tube*.

4.3. Algorithms

4.3.1. Overall logic

The overall logic of the interactive modelling system is:

```
algorithm DRIVER
begin
  CREATE_OBJECT from the root of tree;
  DISPLAY (object);
  while (request := user input) is not quit
        begin
          perform action according to request;
          if re-CREATE_OBJECT
             re-DISPLAY (object);
        end
end.
```

DISPLAY is a standard graphics procedure while CREATE_OBJECT is a recursive function that builds a graphical object.

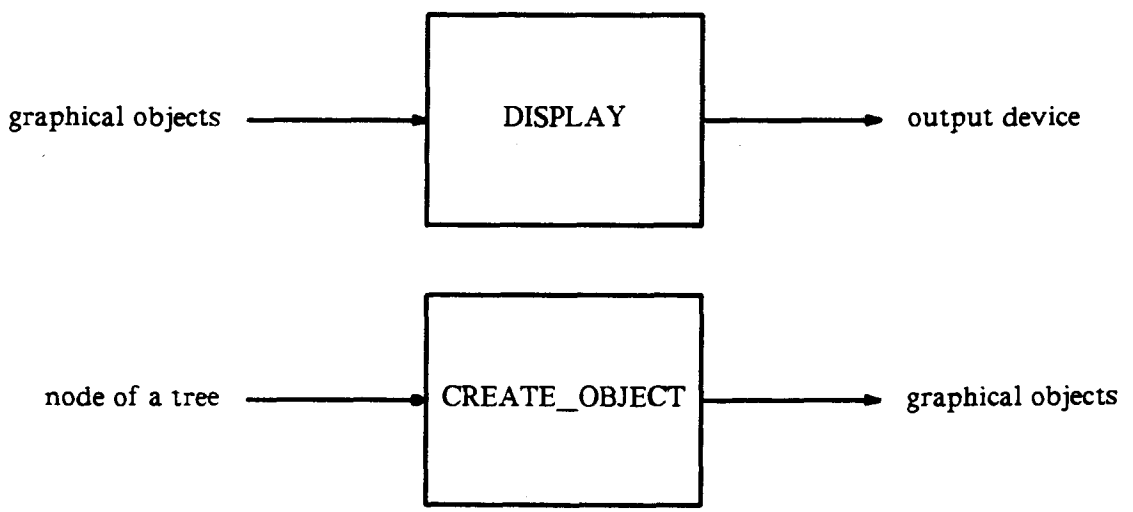


Figure 4-6: Modules of DISPLAY & CREATE_OBJECT

The working of the above modules is now explained in more detail:

```
algorithm DISPLAY(object)
begin
   For each viewport { front, side, top and global }
        begin
        compute its viewing transformations;
        compute its 3D-window for clipping;
        map object to screen coordinates;
        end
end.

algorithm CREATE_OBJECT
begin
    if (node is LEAF)
        get primitive;
    else
        begin
              generate transformation matrix M;
              get primitive for this node & transform it by M;
              CREATE_OBJECT for all sons & transform them by M;
        end
end.
```

The IRIS graphics library [IRIS 86] provides commands for getting simple primitives:

```
move( x1, y1, z1); draw( x2, y2, z2)   /* line */
pnt( x1, y1, z1);                      /* point */
patch( geomx, geomy, geomz);           /* surface patch */
```

The geomx, geomy and geomz are the geometric matrices described in section 4.1. To get a primitive *tube* requires a new routine that calls the procedure *patch*. Before continuing, the notation to be used is defined as follows:

m is no. of rings per segment;
n is no. of control points per *ring*;

$\mathbf{M}_{r_i}$ is the transformation matrix for the i^{th} ring;

p_{ij} is the j^{th} control point of the i^{th} ring
in the *ring* coordinate system;

T is a mxn array of control points
in the segment coordinate system;

G is the geometric matrix;

For implementation, there should be three geometric matrices and three arrays of **T** for each of x, y and z. For convenience, the following algorithms use **G** and **T** only.

The primitive *tube* is created like this:

```
algorithm CREATE_TUBE(segment)
begin

    GENERATE_TABLE(segment)
    for i := 1 to m-2 do
        for j := 2 to n-1 do
            begin
            G := EXTRACT_4x4_from_T(i,j);
             draw the patch(G);
            end
end.
```

The table of mxn control points is generated like this:

```
algorithm GENERATE_TABLE(segment)
begin

  for i := 1 to m do
        for j := 1 to n do
             T[i][j] := p_ij*M_r_i;

end.
```

The **G** is computed like this :

```
algorithm EXTRACT_4x4_from_T(i,j)
begin

          G[][1] := T[i-1,i,i+1,i+2][j-1];
          G[][2] := T[i-1,i,i+1,i+2][j];
          G[][3] := T[i-1,i,i+1,i+2][mod(j+1,n)];
          G[][4] := T[i-1,i,i+1,i+2][mod(j+2,n)];
          { at j=n,n-1 the extracting is revolved back to
            first, two columns (see Figure 4-7) }
end.
```

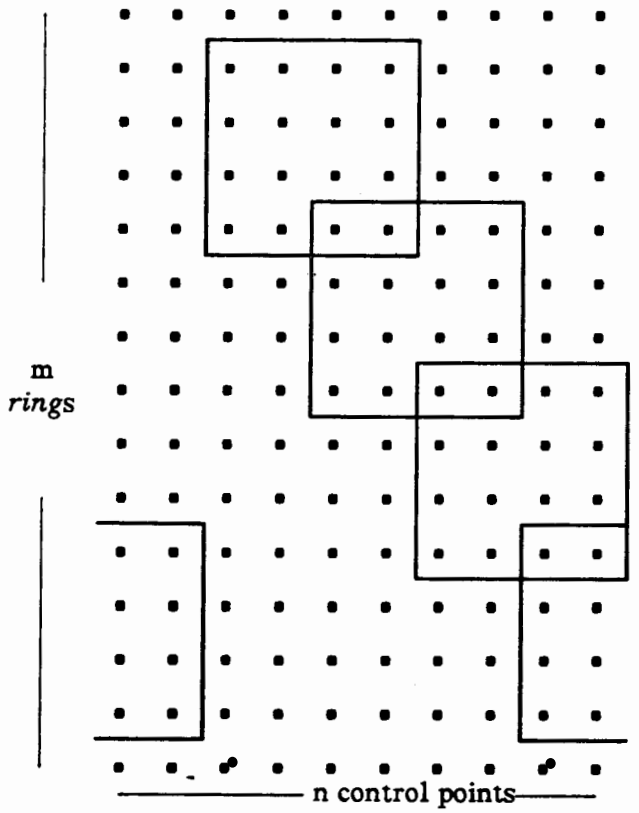


Figure 4-7: Extracting 4x4 control points

For example in figure 4-4, the `tube` is formed by 6 *rings* with 4 control points per *ring*.

4.3.2. The graphical primitive "tube"

The algorithm for creating the primitive tube is presented; however, it does not solve the problem of drawing patches at the joint, because creating surfaces around the joint requires control points from two segments.

In order to draw patches between two segments, as we built a table from the current segment we extract some *rings* from the previous segment and apply the reverse transformation matrix (from object coordinate of parent node to the object coordinate of the son) of the current segment to their control points to get the first few rows for the table (see Figure 4-8).

$$[x_t\ y_t\ z_t\ 1] = [x\ y\ z\ 1]*M^{-1}$$

The reason for this reverse transformation is that during CREATE_OBJECT, **M** will be applied to the current primitive, so it is necessary to cancel this effect since control points of the parent segment should not be affected by **M**, the transformation matrix of the *son* segment.

So we now have a new algorithm for GENERATE_TABLE:

```
algorithm GENERATE_TABLE(segment)
begin
 { FILL the first k rows of TABLE with control points from parent
   segment }
    for i := (m-k) to m do
        begin
                get M_ri from the i^th ring of parent segment;
                get M^-1 from the current segment;

                for j := 1 to n do
                    T[i][j] = p_ij*M_r*M^-1 ;
```

```
        end

{ FILL the rest of TABLE with control points from son segment }
   for i := 1 to (m-k+3) do
        begin
              get M_ri from the i^th ring of current segment;
              for j := 1 to n do
                   T[i][j] = p_ij*M_ri ;
        end
end.
```

Now the *tube* is not necessarily shaped liked a right cylindrical because it may have a bend.

There are also a number of minor problems such as the number of *rings*, k to be extracted from the previous segment, the number of points per *ring* and the number of *rings* per segment.

In order to fill a 4x4 geometric matrix, the minimum number of *rings* per segment is 4 and minimum number of *points* per *ring* is 4. The value for k should be greater than 2 so that the previous segment will not extract information (the first *ring*) from the current segment in order to generate a 4x4 matrix. Therefore basically, the problem of displaying a *tube* is solved.

So far the discussion is based on the assumption that p_{ij} are already properly positioned. The next section will dscribe how to ensure p_{ij} are correctly positioned.

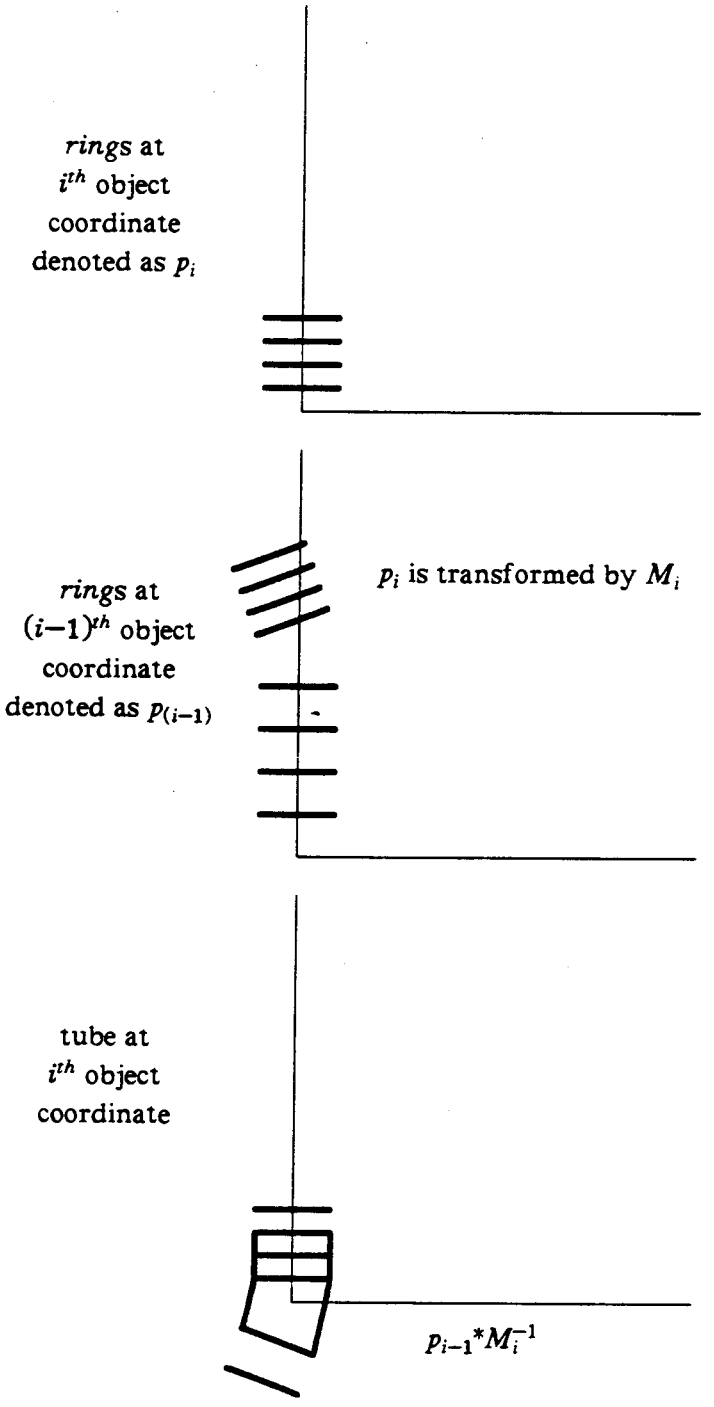


Figure 4-8: Construction of a `tube`

4.3.3. Solving the joint problem

The two major actions are "shape editing" and "posture specification". Both of them require re-construction of the primitive tube (calling the routine CREATE_TUBE). "Shape editing" directly modifies the primitive *tube*, while "posture specification" indirectly affects the primitive *tube*. If we do not re-construct the primitive *tube*, then "posture specification" will modify only the orientation matrix of a segment. The re-construction of the sub-tree (calling the routine CREATE_OBJECT) will be very fast since matrix multiplications are performed by hardware. However, our objective is to have a joint which is both smooth and anatomically accurate, so re-construction of the primitive tube is necessary. However, the process of *tube* construction is really slow so an efficient algorithm is needed. If a human was not "movable", then all control points could be transformed to the world coordinate system first and then surface patches could be directly created. However movement is necessary, so the surface patch has to be created in a segment coordinate system, subject to orientation-transformation and then transformed to world coordinates. This is why the joint modelling problem is so complicated.

The idea used here to solve the joint problem is to first use the interactive system to build standard shapes for different orientations of the joint. Then the shape for any orientation can be interpolated from these standards. This is similar to key-frame interpolation for motion [Kochnanek 84]. Now instead of interpolating joint positions, the parameters for controlling the shape are interpolated.

The shape is defined by the matrix parameters rx (tilt), rz (twist), sx (width), sy (height) and tz (center) of the *rings* that affects the control points. Let SHAPE(rx,ry) be a function that returns the these parameters when it is given the

orientation of the segment. The orientation is specified by the two angles of rotations rx, ry.

Consider a table of mxn standard shapes,

	β_1	β_2	...	β_n
α_1	p_{11}	p_{12}	...	p_{1n}
α_2	p_{21}	p_{22}	...	p_{2n}
	.	.		.
	.	.		.
	.	.		.
α_m	p_{m1}	p_{m2}	...	p_{mn}

each entry p_{ij} represents one of the shape parameters and corresponds to the orientation at rx = α_i and ry = β_j.

Also,

$$\forall_{1 \leq i<m} \quad -360 \leqslant \alpha_i < \alpha_{i+1} \leqslant 360$$

$$\forall_{1 \leq j<n} \quad -360 \leqslant \beta_j < \beta_{j+1} \leqslant 360$$

Note that the values of α_i and β_j are limited by the possible movement of a limb as in figure 4-9.

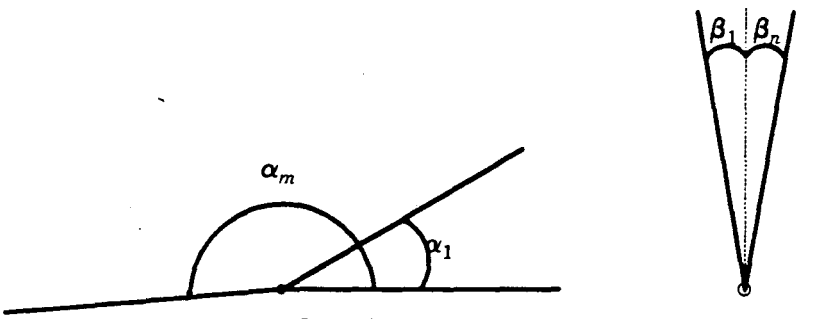


Figure 4-9: Orientations of a limb

The left-hand side represents the rx, for example in the flexing of an arm. The right-hand side represents ry, the lateral movement.

The algorithm is like this:

algorithm SHAPE(rx,ry)
begin
find a,b such that $\alpha_a < rx < \alpha_{a+1}$ and $\beta_b < ry < \beta_{b+1}$.
interpolate using

$$p(rx,ry) = \sum_{i=a-k}^{a+k+1} \sum_{j=b-k}^{b+k+1} f_{ij}(rx,ry)\, p_{ij}$$

end. (4.1)

The function f_{ij} can be implemented in many ways (this will be illustrated in the next chapter), but it must satisfy the constraints that $shape(\alpha_i,\beta_j) = p_{ij}$. We must also consider the problem of choosing k so that $a-k$ and $b-k$ cannot be smaller than 1 while $a+k$ and $b+k$ cannot be greater than m and n respectively. k determines the number of standard parameters p_{ij} are used.

For example, for a bi-cubic spline,

k=1 (i.e. there are 4x4 p_{ij}'s) and $f_{ij}(u,v) = B_{i,4}(u)B_{j,4}(v)$

where $B_{i,4}$ and $B_{j,4}$ are blending polynomials of degree 3; and rx, ry are normalized to u and v as follows:

$$u = \frac{\alpha_{a+1} - rx}{\alpha_{a+1} - \alpha_a}$$

$$v = \frac{\beta_{b+1} - ry}{\beta_{b+1} - \beta_b}$$

The idea is very simple and can provide any desired accuracy but the problem is the tedious work involved in finding the standard shapes for all orientations. The error is $O(1/n+1/m)$ so that if the number of standard shapes increases, the accuracy

of the interpolated shape increases. Therefore this is an accuracy-work tradeoff problem. Another point is the need for real-time performance which requires that the re-construction of a shape (i.e. the *tube*) be carried out as fast as possible. To interpolate each of rx, rz, sx, sy and tz is time-consuming. To interpolate directly from control points will be fast; however, this contradicts the original idea of having these extra parameters for better control in shape editing.

Chapter 5
Implementation and Verification

5.1. Incremental development

A sandwich incremental implementation [Page-Jones 80] approach has been used. First, the top-lever driver was built; this was followed by debugging the viewing system and interface system using a simple object (stick figure). At the bottom-level, utility routines were coded and debugged, while the algorithm for displaying the primitive tube was tested. Finally, the top-level and bottom-level systems were integrated for testing, and more capacity was added. One important observation is that the initial design is bound to be modified a great deal. This is because lots of unpredictable problems come up during the testing phase.

The Tree DBMS discussed previously is not going to be implemented since a working Human Data Manipulation System has already been developed. This system is more efficient since it does a more specific job; however, it is difficult to maintain because it is less general. Abstractly, the data structure of the human tree is still retained.

The human data manipulation system contains functions that access, update and traverse the data tree. The manual pages of these functions are in Appendix A.

5.2. Development of the User Interface

To simplify user control, only one physical device is used, namely the mouse. There are three buttons on the mouse that the user can press and the horizontal and vertical motion of the mouse can map to two values. Thus there are 2^4 controls; this is too many for a user to remember, so only a few combinations are used.

It is often necessary to change three parameters at the same time (e.g. x,y,z), but since mouse motion can only change two values, two buttons are needed to increment and decrement the third parameter. Thus the remaining button can be used to activate a pop-up menu which allows a wide range of selections.

The index finger is the most natural and most frequently used finger to press a button. Therefore, as a person grasps the mouse, the only other free finger that will be used is the middle finger. So if the index finger is reserved for holding the left button down, then the middle finger can be used to press the right and middle buttons. As it is unnatural to find another free finger, the right and middle buttons are used more effectively for complementary controls. (e.g. zoom in/zoom out; rotate clockwise/rotate anticlockwise) Keeping this in mind, the user interface will be consistent and simple.

One of the basic actions to be carried out using only the mouse is to pick a graphical object and change the values of rx, ry, rz, sx, sy, sz, tx, ty and tz. One way of picking is to move the cursor to the object and to hold down a button to attach the object to the mouse controlled cursor. Once it is picked there is no way to re-pick another object or do other selections; the button has to be released again in order to exit. There are nine parameters for placement and orientations but only

three can be changed at a time; the user should select which three parameters are to be changed before the object is attached. One alternative is to move the cursor to the object and press a button which activates a pop-up menu to select the set of parameters to be changed. Both methods have some merit; the choice depends on how the user wants to structure the task.

In reality, using the motion of a mouse to map two values is not easy. It is difficult to hold one value constant since a user cannot easily move the mouse absolutely horizontally or vertically. The problem is much more serious, when mapping two rotations for the segment. One alternative is to map the mouse motion to one value only and use the buttons to choose which directions to rotate. Yet this is inconsistent with other control mechanisms. Another alternative is to display the changing values to aid the user. Also, two xyz-axes are drawn at the joint for relative reference. One xyz-axis is fixed while the other axis is rotated with the vertical motion of the mouse.

When an object is picked, it is attached to the mouse. So the cursor should disappear and it should no longer attach to the mouse. Nevertheless, there is a time lag problem since the system does not always response fast enough to follow the mouse motion. A solution is to maintain the cursor display but changes it to a box shape; when the system is performing computation, the cursor changes to an hourglass shape which means that it will not be affected by movement of the mouse. When a computation is short, this will be un-noticeable. Also, the cursor is initially at the center of the screen and will be moved with the mouse motion.

A dynamic menu has the advantages of less load on user memorization and less need for the user to move his eye to other static menu items while concentrating on

modelling the object. Status messages are very useful but will sometimes divert the attention of the user while looking at the object. A better feedback technique is to change the cursor shape (for example during time consuming computation which requires the user to wait). The main point is that the feedback should be as close to the eye contact as possible and the feedback should be obvious to understand.

Most systems will ask the user to key in the filename for saving data but again it is inconsistent to the guideline of the proposed interface design. To achieve a consistent user interface, the filename is only a number which is mapped to the cursor position on the screen, that is, the mouse can be used to select a filename.

There is still a problem in visualizing a 3-D object on a screen because any single view is two dimensional. To alleviate the viewing problem, four viewports are used as in many other 3-D modelling systems: these are front view, top view, side view and a global view. However, as number of curves increase, the scene becomes very messy to look at. A better model is to use some means of hidden-line removal but the required computation time inhibits real-time user interaction. Using multiple colors or even an alternation of two colors can also be very confusing. So one color is assigned to the front half of the segment curves and another color is assigned to the other half. Moreover, when a segment is picked, only its parent and its descendent segments will be displayed so that changes are faster and there is less details to look at.

5.3. Interpolation vs Other alternatives

Interpolation requires standard shapes which are tedious to built. So our objective is to minimize this effort or find other alternatives in solving the joint problem.

One approach is to put a sphere around the joint and compute the intersection with the two connected primitive tubes. In practice, the rendered image around the joint is usually still not smooth enough, so this approach is inadequate.

Recall that the two *rings* closest to the joint affect the shape at the joint. So one approach is to resolve the two *rings* into one *ring*. This was tested and the displayed shape was not smooth enough. So this approach is inadequate and should not be examined further.

The basic problem is that regardless of what method is used, the only way to test the method is to look at the displayed shape; the evaluation is rather subjective. The interpolation of standard shapes has been chosen because it is the most simple and accurate method to solve the joint problem.

Recall the interpolation formula (4.1) in algorithm SHAPE:

$$p(rx,ry) = \sum_{i=a-k+1}^{a+k+1} \sum_{j=b-k}^{b+k} f_{ij}(rx,ry)\, p_{ij}$$

The objective is to minimize the human effort, that is to use as small a number of standard shapes as possible. So only four standard shapes are considered ($k = 0$). The simplest formula is:

$$p(u,v)=(1-u)vp_{21}+uvp_{11}+u(1-v)p_{12}+(1-u)(1-v)p_{22}$$

where rx and ry are normalized to u and v.

A more complicated formula uses a bi-cubic spline with the geometric matrix :

$$\mathbf{G} = \begin{bmatrix} p_{11} & p_{11} & p_{12} & p_{12} \\ p_{11} & p_{11} & p_{12} & p_{12} \\ p_{21} & p_{21} & p_{22} & p_{22} \\ p_{21} & p_{21} & p_{22} & p_{22} \end{bmatrix}$$

Although this is a cubic interpolation, the effect will not be good since **G** is duplicated from four values only. So a different method is implemented:

$$p(u,v)=(w_1p_{21}+w_2p_{11}+w_3p_{12}+w_4p_{22})/\sum_{i=1}^{4} w_i$$

where

$$w_1=1/((1-u)^2+v^2)$$
$$w_2=1/(u^2+v^2)$$
$$w_3=1/(u^2+(1-v)^2)$$
$$w_4=1/((1-u)^2+(1-v)^2)$$

This is a weighted sum formula in which the weights are the distances between the (u,v) and the corners (Figure 5-1). The idea is based on the inverse-square law. Again the validity can only be justified by looking at the displayed shape.

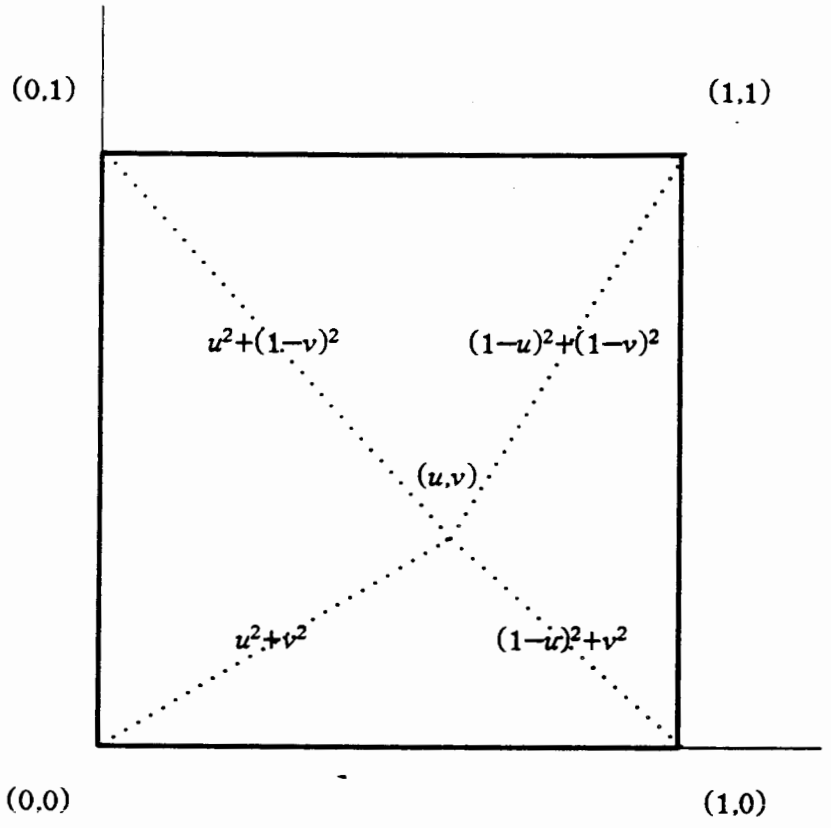


Figure 5-1: Square Distances of (u,v) from the corners

Chapter 6

Results

6.1. Using the system

A figure is modelled by using the following steps:

1. *pick* a segment
2. specify the orientation
3. model the shape around the joint.
4. use *expand* or *contract* to change the size of all the *rings*.
5. achieve a perfect shape by dragging each point step by step.
6. save it in a file.

The above steps can be elaborated. In step one, a segment has to be picked first by pointing the arrow cursor at the segment (a white line), pressing the control button to activate a pop-up menu and selecting the menu item "move joint". For better speed and viewing, all other unrelated segments or objects will disappear.

For step two, during interactive changes in the orientation of a segment, its primitive tube is removed to achieve a real-time interaction (See Figure 6-1).

After this is finished, the primitive tube is re-constructed and displayed. There are several algorithms used in re-creating the primitive tubes; one algorithm takes 3-4

seconds to compute while another algorithm is faster but the generated shape is not perfect. Without the use of interpolation, the re-created shape around the joint is imperfect when the angle between the two segments is small (This is due to overlapping of two *rings*. Recall Figure 3-4).

By dragging a point, the user can model the shape, but it is quite a tedious job. So tilting is introduced to solve the joint problem (step three). With interpolation, tilting is done automatically. The shape of the joint is mainly controlled by two *rings* from the two connected segments. Yet tilting (rx) alone is insufficient to correct the shape around the joint, it requires the *rings* (tz) be moved toward the distal joint. The distance between the center of the *rings* is inversely proportional to the angle between the two segments. Also, as the joint angle becomes smaller, the angle of tilt becomes larger. Moreover, the sy (scaling in y-direction) will be changed to ensure the control points remain the same distance with the segment.

Step four provides a facility to change the size of a *ring* in one step which is much more convenient than dragging the points of this *ring* one by one. Again, having a facility to change the global shape of a segment (several *rings*) in one step is really convenient.

For step five, the dragging of a control point is easy but looking at the effect on a 2-D monitor is difficult. So in addition to multiple viewports, the user has the choice to hide some of the curves interactively (see Figure 6-2). Also, the user may want to quit and look at the object from different directions. Then he may want to re-pick the same point or some other points for dragging or tilting the ring, so a fast mechanism to do this is beneficial. Thus, after exiting from viewing (or other modes) the arrow cursor will move back to the previously picked point, so that the user can

re-pick it again. This is a good general approach since during shape modelling, a user will usually concentrate on only a small region of the shape.

The final step is to save the data in a file. If the shape is still not perfect, then the data can be retrieved again and the modelling process can be started from step three since the orientation is already specified.

6.2. How good is the modelling system?

The system has been tested by modelling a leg with particular attention being paid to the knee.

First, based on a set of pictures of real human leg taken at different orientations and angles, the computer-displayed leg was modelled to be as close to the real leg as possible. These are the standard shapes.

In the top picture of figure 6-4, the abnormal shape around the knee is due to overlapping of control points. Using interpolation, there is a substantial improvement in the interpolated shape; however, there is not much difference between the example using three standard shapes and that using five standard shapes. This is because the standard shapes themselves are not perfect; indeed, it is very tedious to built a good standard shape.

The first standard shape is modelled at maximum flexing (20°). Then this leg is straightened (at 180°) and modelled. So some *rings* are still tilted (which is unnecessary). Then the standard shape at 100° is interpolated from the above two standards and then modelled and saved. Similarly, other new standard shapes can be interpolated from these old standards and then modelled and saved. If the above

procedure is continued, there will come to a point where additional modelling is unnecessary for an interpolated shape.

In figure 6-5, instead of displaying one leg, additional legs (these are the standard shapes) are displayed on top for references. This ensures that the size of the modelled leg is similar to the standard shapes.

In figure 6-7, the complete human figure is displayed but only the right leg is properly modelled. Gouraud shading is used here; the four vertices of each polygon and the intensity at each vertex are computed from the geometric matrix. This provides some indication of the results which can be achieved using this modelling system.

The experience gained in working with these examples clearly indicates that the approach developed in this thesis is viable. It would be beyond the scope of the thesis to take the time to use the system for a comprehensive model.

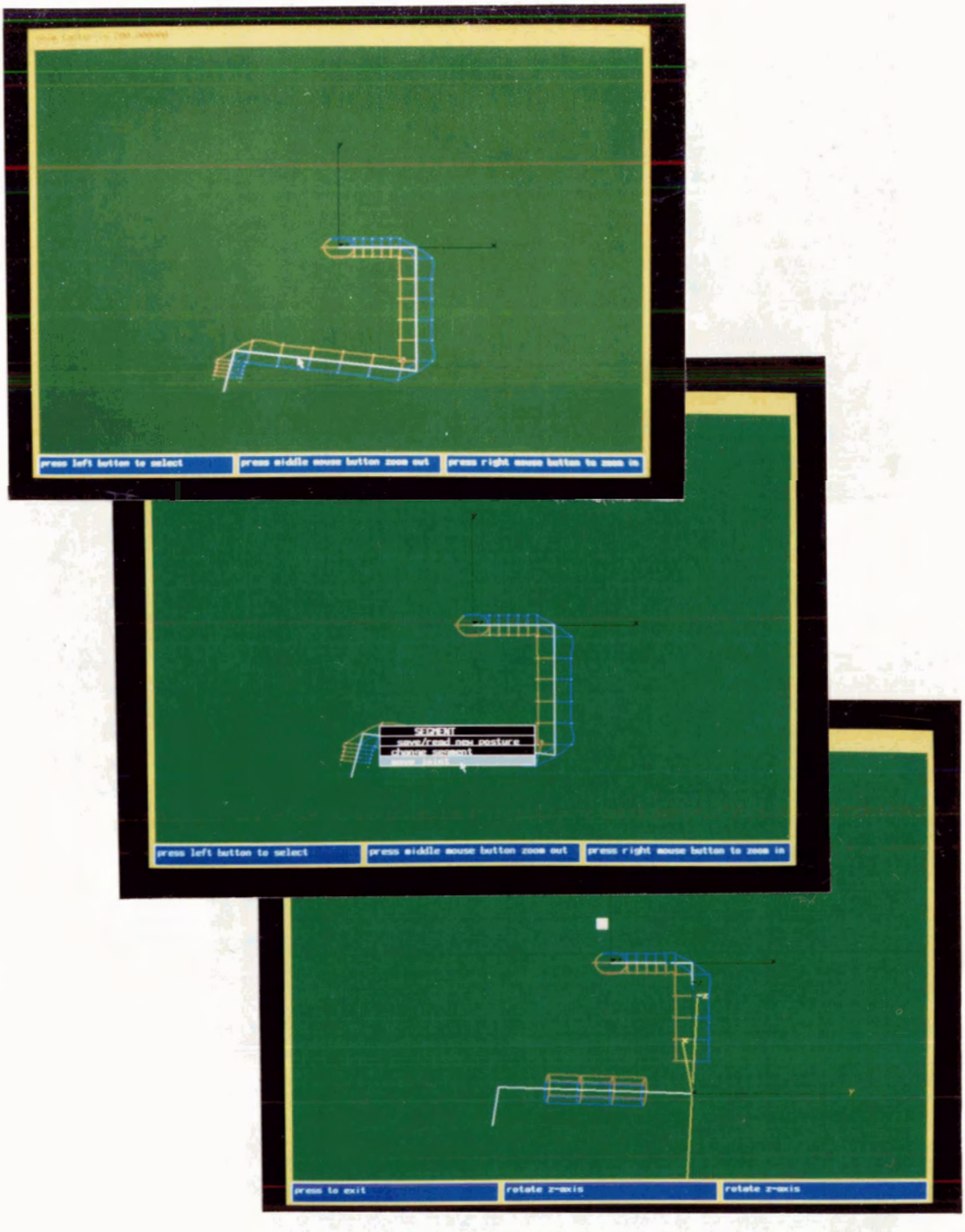


Figure 6-1: Move joint

Top : cursor at a segment.
Middle : activate a pop-up menu.
Bottom : movement of mouse mapped to rx & ry.

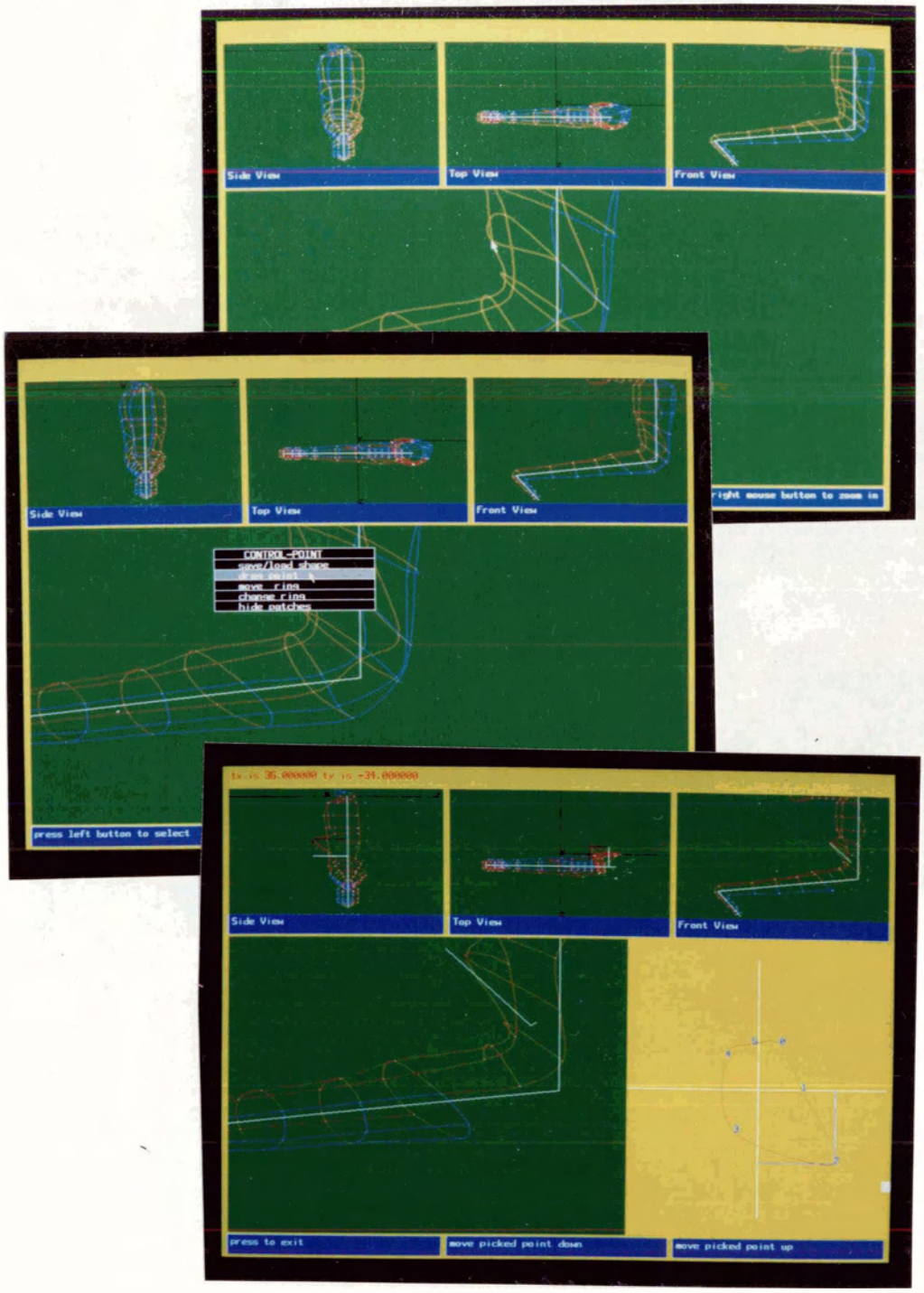


Figure 6-2: Drag point

Top : cursor at a point.
Middle : activate a pop-up menu.
Bottom : movement of mouse mapped to x & y.

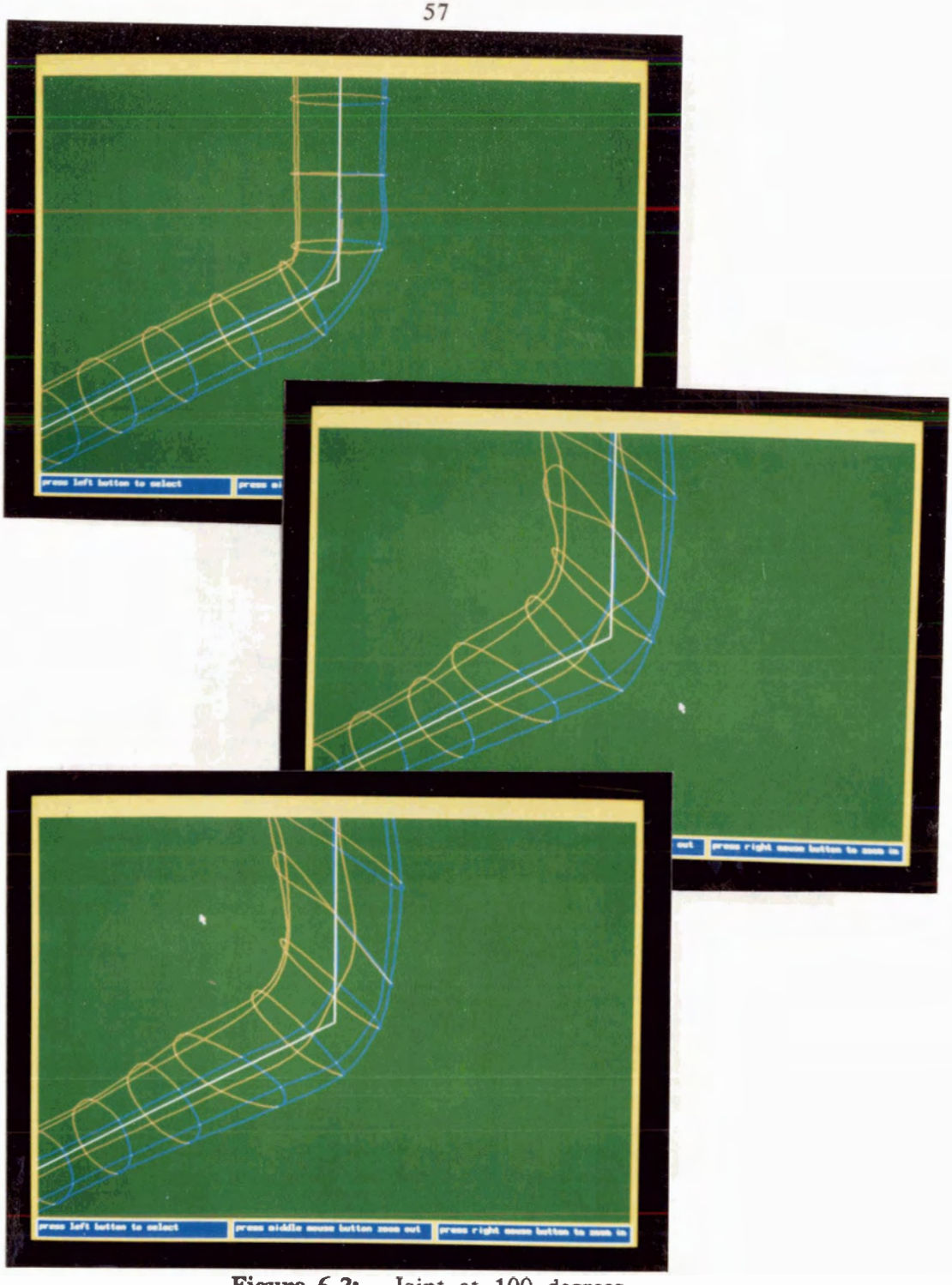


Figure 6-3: Joint at 100 degrees

Top : no interpolation.
Middle : interpolate using 3 standards.
Bottom : interpolate using 5 standards.

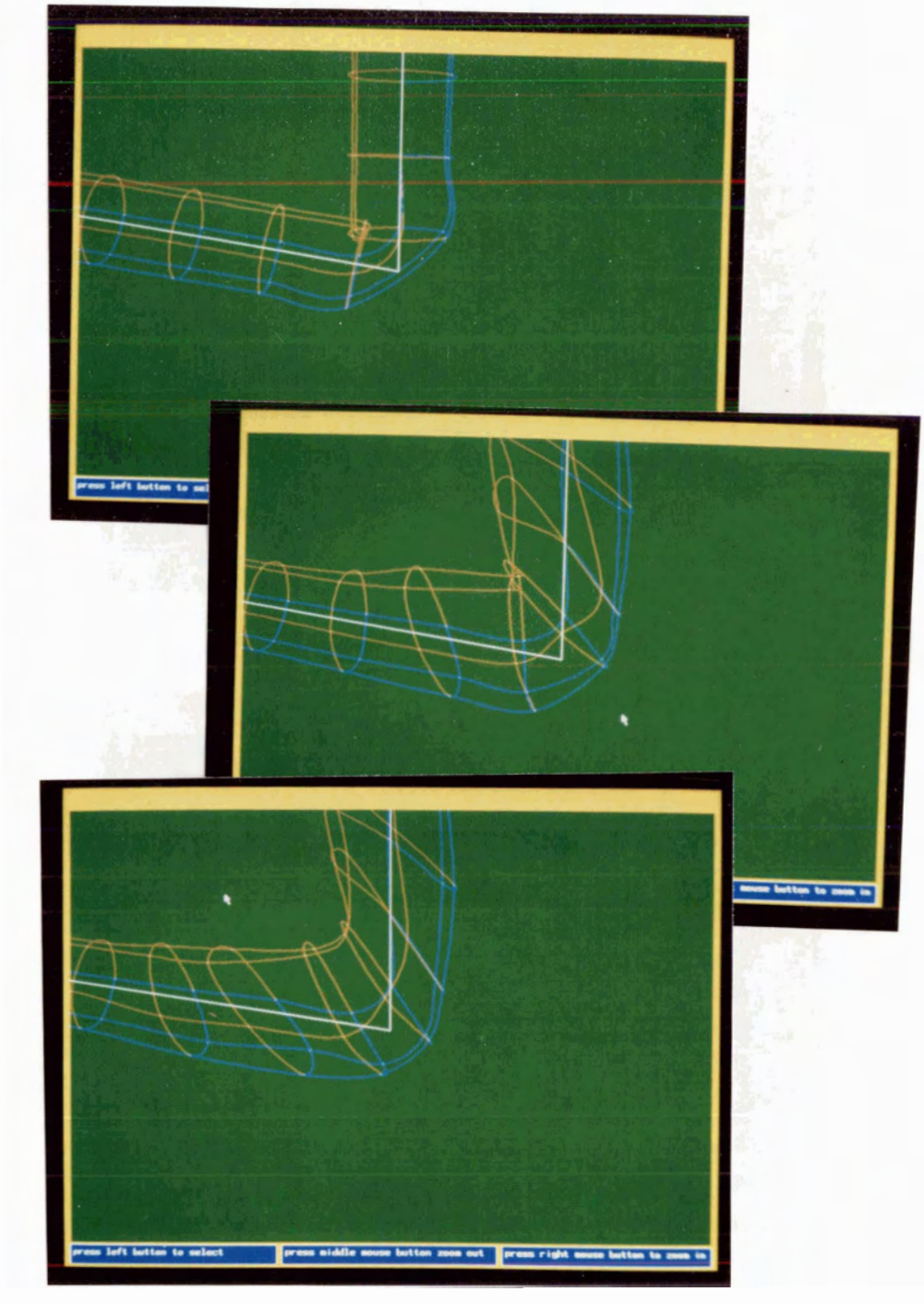


Figure 6-4: Joint at 80 degrees

Top : no interpolation.
Middle : interpolate using 3 standards.
Bottom : interpolate using 5 standards.

Figure 6-5: Standard Shapes

Top : standard shapes at 180°, 100°, 20°, 140°, 60°.
Bottom : standard shapes at 180°, 20°, 100°.

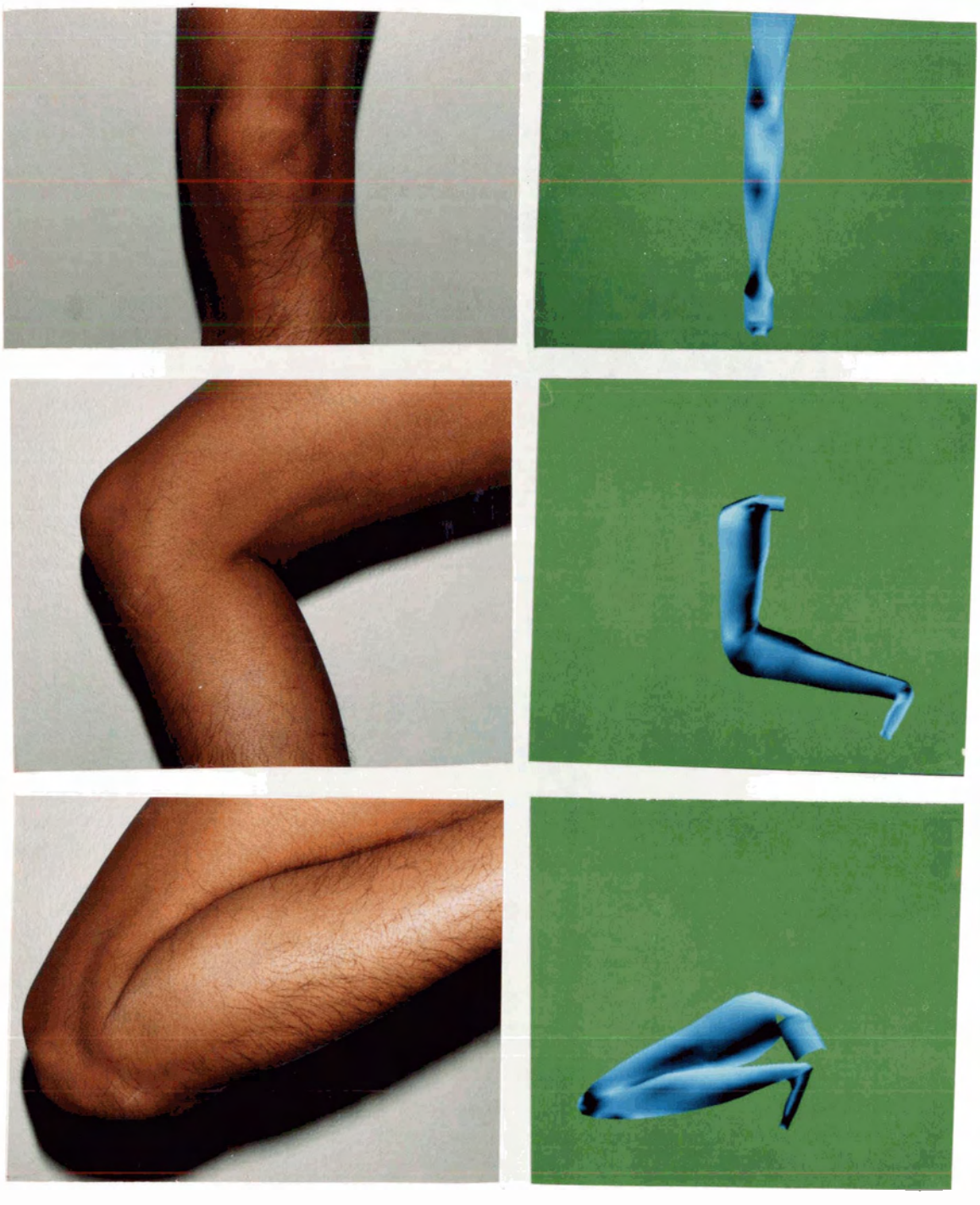


Figure 6-6: Comparison with real human leg

left : real human legs.
right : genterated legs.

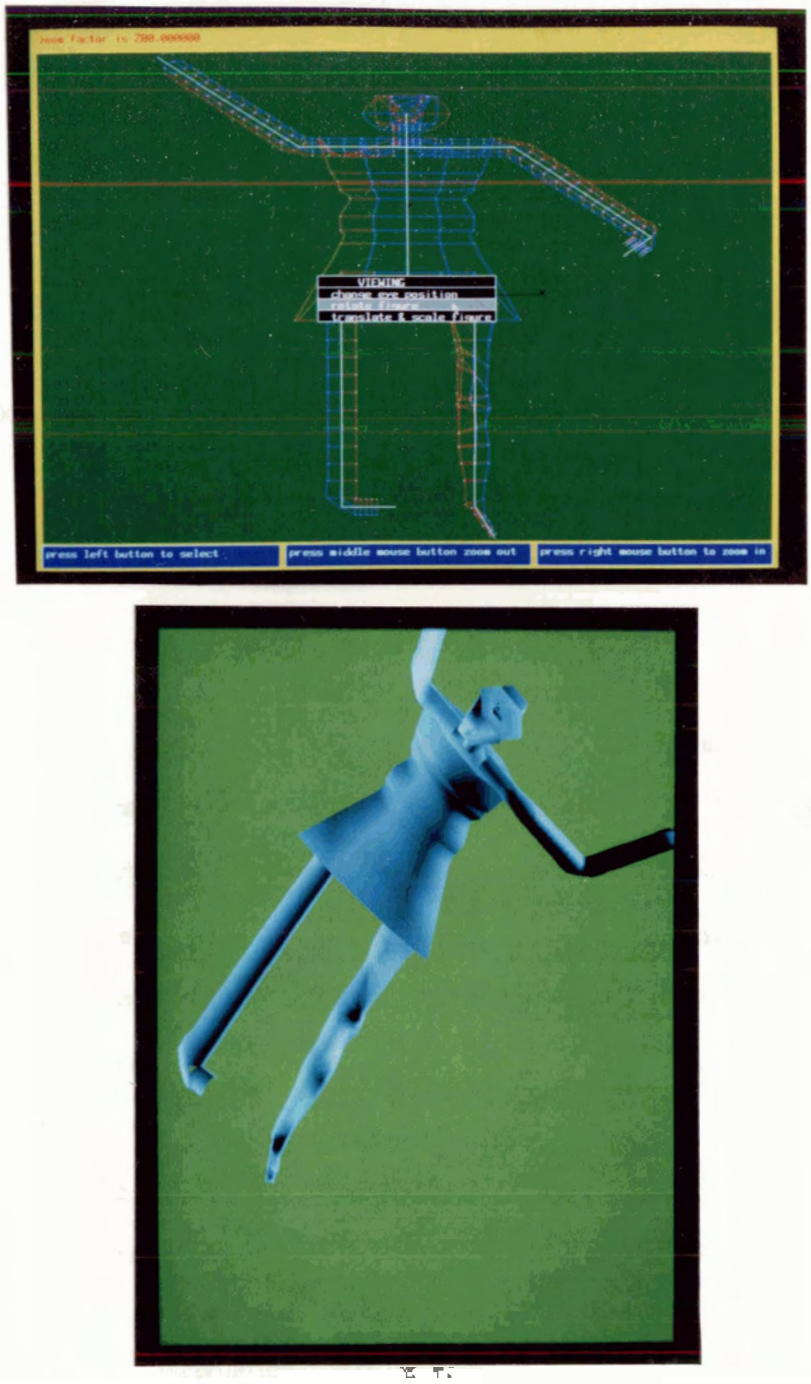


Figure 6-7: Complete human figure

Chapter 7

Epilogue

7.1. Shape Interpolation in general

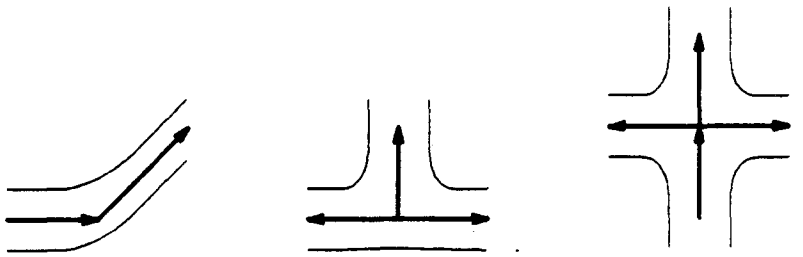
In general, when using splines a shape can be defined purely by control points. The addition of parameters like tilt, twist or displacement provides higher level control.

Currently, the only parameters that change the shape are rx and ry. The reason is that most of our previous work has been based on a stick figure or disconnected segments so rx and ry are sufficient to specify the orientation. In fact, rz should also be considered; the twisting effect due to rz has a significant effect in real bodies. Also, the shape does not depend purely on orientation of the current segment but also on its connected segments. This can be illustrated more clearly by example. It is obvious if we try to twist our lower arm. The shape around the elbow depends on the rx of lower arm and rz of upper arm; while the shape around the shoulder depends on the rx,ry,rz of upper arm and the rx,ry of the connected shoulder bone.

So the more general interpolation formula is :

$$p(t_1,t_2,\ldots,t_n) = \sum_{k_1=1,k_2=1\ldots,k_n=1}^{m} f_{k_1}(t_1)\, f_{k_2}(t_2)\ldots f_{k_n}(t_n)\, p_{k_1 k_2 \ldots k_n}$$

The algorithm for displaying a primitive tube only works for a list structure. Since a human figure is a *tree*, an algorithm to display a continuous surface at a



Type 1 Type 2 Type 3

Figure 7-1: Types of *joint*

branch (see Figure 7-1) is needed. A human has one "type 2" joint (around the hip) and one "type 3" joint (around the the neck and shoulders) (see Figure 3-2). The rest of the joints (elbow, knee and shoulder joint) are "type 1". Note that this categorization does not distinguish the different motion associated with different joint, and the movement of a shoulder joint is very different from the movement of an elbow.

Obviously, the problem of *exactly* displaying the whole human figure has not been solved in this thesis! Further research is required.

7.2. Future Enhancements

The user interface can certainly be improved. First, as the user becomes more familiar with the system, the simple interface becomes "expert unfriendly". A mechanism for the user to define his own interface is desirable [Olsen 86]. The physical device will include the keyboard. So while the right hand is moving the mouse, the left hand can choose over 40 keystrokes, which provides considerable power. Also, the user is allowed to modify the entries of the pop-up menu and to define the action of any keystroke, thus, eliminating the use of pop-up menus and making the interface much more flexible. Another useful feature is "undo" as it is very easy to make a mistake in modifying the shape.

During the building of a standard shape, a control point can be dragged in three directions : x, y and z; it may be easier to restrict the dragging of a control point to the x and y directions only. The effect of dragging in the z direction can be achieved by inserting a *ring* on the current *ring* and dragging a point on this new *ring*. This means that a sub-division algorithm is needed.

The human body is approximately symmetric; for example, the left arm is usually a mirror image of the right arm[10] . Therefore we can really save space if a node can point to the same subtree forming a hierarchical ring structure [Tremblay 76].

The incorporation of constraints on movement specification and shape specification would be useful. This extra information can be stored into a constraint data base.

---

[10] except for tennis players!

7.3. Digitizing

Although it is tedious to build standard shapes, they can be provided from digitized data. Also, digitized shapes provides a basis for verifying the accuracy of the shape-interpolation formula. Statistical analysis can be used to compare the interpolated shape and digitized shape. Yet there are technical problems in getting digitized data and transforming the data to control points. The problems are similar to the problems found in defining digitized motion paths.

Another alternative is to use a video-camera and get a bit-plane image of a real human. This image can be used as the background of the modelling system. So an artist can interactively model the displayed shape to look like the digitized shape.

Whether the data is from digitizing or from other programs (e.g dance composer program), the description should be-in the form :

```
(segment_id  rx ry rz shape-description)

shape-description =
                        ( ring1 tz ( x1 y1 x2 y2 ....)
                          ring2 tz ( x1 y1 x2 y2 ....)
                                .
                                .
                                .
                          ringm tz ( x1 y1 x2 y2 ....))
```

If a *ring* can be considered as the horizontal cross-section of a body segment with no tilt and twist, then the (x,y) pairs of control points can be digitized along the outline of the cross-sections of a body segment. A *ring* is translated to the segment coordinate system by tz which can be measured between the proximal joint and distal joint.

7.4. Applications

The human shape modelling system will provide basic data for other application programs. As discussed above, we have been particularly interested in producing high quality rendered images for use in animation.

The system can also be used to model other articulated objects like plants, insects and animals. We have recently been approached for assistance in modelling dinosaurs.

7.5. Conclusion

The problem of modelling the shape around a human joint is examined in this thesis. An interactive human modelling system has been designed and implemented as a test-bed for solving the joint problem. Also, in presenting the design of the software and setting out the software development process, the concept of user interface design, system partitioning and programming techniques have been examined in detail and illustrated. Structured design [Page-Jones 80] provides a guideline in software development but it is not a one step utopia. It has been quite successful in designing inventory or accounting systems. However, an interactive graphics system is different from a business system and there is little previous work on interactive 3-D modelling for the human body or anything else. The system is designed to be maintainable and portable so that the user interface sub-system and the human data base sub-system can be used by other application programs. Furthermore, the tree data-structure for representing a human and its shape is so general that it can be used for representing other articulated irregular solids. It also has the advantages of allowing a fast re-specification of the orientation of each segment and a fast re-construction of the patches around each joint.

After developing a method to display a *tube*, One solution to the problem of primitive re-construction due to joint movement has been developed. Several algorithms were evaluated with the interpolation of standard shapes being chosen the best solution to solving the joint problem. The more general problem of shape interpolation is discussed and suggestions are made for future work.

Appendix A

Human Data Manipulation Sub-system

The Human Data Manipulation Sub-system is a library of routines for accessing and updating the human graphical data. It is for programs that are based on the IRIS-graphics system.

In order to use these routines, an application program has to include the files "hdms.h" and be compiled with "hdms.o". The functions are :

Node Type	update functions	access functions	transversing functions
segment	setJointX setJointY setJointZ setRotations	getJointX getJointY getJointZ getRotateX getRotateY	nextJoint previousJoint firstJoint
ring	setTilt setHeight setScaleX setScaleY	getTilt getHeight getScaleX getScaleY	nextRing previousRing firstRing
control_point	setPointX setPointY setPointZ	getPointX getPointY getPointZ	nextPoint previousPoint firstPoint

SetJointX, setJointY and setJointZ are not implemented to set the orientation. Instead, a new function setRotation(rx,ry) is used because it is more easier to specify two angles for a segment.

NAME

getRotateX, getRotateY, getJointX, getJointY, getJointZ, --- get data from a joint

SPECIFICATION

```
#include <gl.h>
#include <hdbs.h>

Angle getRotateX(id)
Jointid id;

Angle getRotateY(id)
Jointid id;

Coord getJointX(id)
Jointid id;

Coord getJointY(id)
Jointid id;

Coord getJointZ(id)
Jointid id;
```

DESCRIPTION

The getRotateX returns the angle rotate along x-axis and the getRotateY returns the angle rotate along y-axis. These two anlges specify the orientation of a segment which is initally at origin and aligned with the z-axis.

The getJointX, getJointY, getJointZ return the coordinates of a joint

NOTE: getJointX, getJointY and getJointZ are macros.

NAME

getTube, getPointX, getPointY, getPointZ - get the data for specifying shape

SPECIFICATION

```
#include <gl.h>
#include <hdbs.h>

Coord getPointX(id)
pointKey id;

Coord getPointY(id)
pointKey id;

Coord getPointZ(id)
pointKey id;

Object getTube(id)
Jointid id;
```

DESCRIPTION

The getTube function returns a graphical object for display. The shape of the tube depends on the control points which are indirectly depend on the tilt and twist of *rings*. The getPointX, getPointY, getPointZ return the coordinates of a control point while the getTilt, getTwist return the angles of tilt and twist.

NOTE: getTube, getPointX, getPointY and getPointZ are macros.

NAME

setRotation -- update the data of specifying orientations and shapes

SPECIFICATION

```
#include <gl.h>
#include <hdbs.h>


setRotation(id,rx,ry)
Jointkey id;
Angle rx,ry;

setTilt(id,value)
ringKey id;
Angle value;

setTwist(id,value)
ringKey id;
Angle id;

setTube(id)
Jointkey id;
```

DESCRIPTION

The setRotation specifies the orientation of a segemnt. The coordinates of the joint is also set too.

The setTube creates the graphical primitives for the current segment.

Name

firstJoint, nextJoint, previousJoint, firstRing, nextRing, previousRing, firstPoint, nextPoint, nextPoint, nextPoint - traversing the tree using KEY manipulation.

SPECIFICATION

```
#include <gl.h>
#include <hdbs.h>


Jointid nextJoint(currentJoint,branch)
int branch;
Jointid currentJoint;

Jointid previousJoint(id)
Jointid id;

ringKey firstRing(id)
Jointid id;

Jointid nextRing(id)
ringKey id;

Jointid previousRing(id)
ringKey id;

poinKey firstPoint(id)
ringKey id;

Jointid nextPoint(id)
pointKey id;

Jointid previousPoint(id)
pointKey id;
```

DESCRIPTION

To traverse the list of joints, control-points and rings, use nextJoint, nextPoint and nextRing respectively. Since the data structure of joints is a tree, the parameter

"branch" of nextJoint is used to distinguish the path. For convienience, previousJoint, previousJoint and previousRing are used for traversing backwards.

Note the first control point of a *ring* is accessed by firstPoint, the first *ring* of a joint is accessed by firstRing and the first joint or the root is specified in the file human.setup (see initHuman).

NAME

initHuman - initialize the data base system.

SPECIFICATION

```
#include <hdbs.h>
initHuman(setup_file, posture_file, shape_file);
*FILE setup_file;
*FILE posture_file;
*FILE shape_file;
```

DESCRIPTION

Set up the data base for human data.

FILES

*.setup - files that describe all external environments.

*.posture - files that describe the posture for the human.

*.shape - files that describe the shape for each segment

Appendix B

User Interface Management Sub-system

The User Interface Management Sub-system is a library of routines for getting user input from physical device and for controlling the system feedback. It is for programs that are based on the IRIS-graphics system.

Since this sub-system is still under development, only several routines are presented. In order to use these routines, an application program has to include the files "mouse.h" and be compiled with "mouse.o".

NAME

initmap3, map3 - map the mouse device to three parameters.

SPECIFICATION

```
#include <mouse.h>

map3(arg1,arg2,arg3)
int *arg1, *arg2, *arg3;

initmap3(max1,min1,max2,min2,max3,min3)
int max1,min1,max2,min2,max3,min3;
```

DESCRIPTION

The map3 attached the mouse device to affect the values of three specified parameters. The first parameter will be depend on the vertical motion of the mouse. The second parameter will be depend on the horizontal motion of the mouse. The third parameter is affected by the complementary buttons (the middle mouse button and the right mouse button).

There is a limit for each of the three parameters to have. The limits are set by by initmap3.

NAME

make_layout - control the number of viewports on the screen.

SPECIFICATION

```
#include <mouse.h>

make_layout(number_of_views,figure)
int number_of_views;
Object figure;

setViewport(id, l, r, b, t, view)
portKey id;
ScreenCoord l, r, b, t;
Viewing view;
```

DESCRIPTION

The make_layout will create viewports for a graphical object.

Each viewport has to be specifed by the setViewport routine which set size of the viewport and use either orthogonal view or perspective view.

References

[Armstrong 86] Armstrong, W.W., Green M. and Lake R.
Near-Real-Time Control of Human Figure Models.
Graphics Interface '86 :147-151, May, 1986.

[Badler 78] Badler N.I., O'Rourke J. and Toltzis H.
A Human Body Modelling System for Motion Studies.
Technical Report 13, University of Pennsylvania, July, 1978.

[Badler 79a] Badler, N.I. and Smoliar S. W.
Digital Representations of Human Movement.
ACM Computing Surveys 11(1):19-38, March, 1979.

[Badler 79b] Badler, N.I., O'Rourke J. and Toltzis H.
A Spherical Representation of a Human Body for Visualizing Movement.
Proceedings of the IEEE 67(10):1397-1402, Oct., 1979.

[Badler 82] Badler, N.I.
Guest Editor's Introduction: Human Body Models and Animation.
IEEE Computer Graphics and Applications 2(9):6-8, November, 1982.

[Barr 84] Barr, A.
Global and local deformations of solid primitive.
Computer Graphics 18(3):21-30, 1984.

[Calvert 82] Calvert T., Chapman J. and Patla A.
Aspects of the Kinematic Simulation of Human Movement.
IEEE Computer Graphics and Applications 2(9):41-52, November, 1982.

[Carey 82] Carey, T.
User Differences in Interface Design.
Computer 15(11):14-20, November, 1982.

[Carson 83] Carson, G.S.
The Specification of Computer Graphics Systems.
IEEE Computer Graphics and Applications 3(6):27-43, Sept., 1983.

[Catmull 72] Catmull, E.
A system for computer generated movies.
In *1972 ACM Annual Conf.*, pages 422-431. ACM, New York, 1972.

[Catmull 75] Catmull, E.
Computer display of curved surfaces.
In *IEEE Conf. Computer Graphics, Pattern Recognition, and Data Structure*, pages 11-17. IEEE, New York, 1975.

[Clark 81] Clark, J.H.
Parametric Curves, Surfaces and Volumes in Computer Graphics and Computer-Aided Geometric Design.
Technical Report 221, Standford University, November, 1981.

[de Boor 78] de Boor, Cox.
A Practical Guide to Splines.
Springer-Verlag, New York, 1978.

[Goldman 83] Goldman, R.N.
Two approaches to a Computer Model for Quadric Surfaces.
IEEE Computer Graphics and Applications 3(6):21-26, Sept., 1983.

[Green 81] Green, M.
A Methodology for the Specification of Graphical User Interface.
Computer Graphics 15(3):99-108, Aug., 1981.

[Herbison-Evans 82]
Herbison-Evans D.
Real-Time animation of Human Figure Drawings with Hidden Lines Omitted.
IEEE Computer Graphics and Applications 2(9):27-34, November, 1982.

[Hoffmann 85] Hoffmann C. and Hopcroft J.
Automatic surface generation in computer aided design.
The Visual Computer 1:92-100, 1985.

[IRIS 86] *IRIS User's Guide*
Version 2.1 edition, Silicon Graphics, Inc, 2011 Stierlin Road, Mountain View, CA 94043, 1986.

[Kochnanek 84] Kochanek H.U. and Bariels R.H.
Interpolating Splines with Local Tension, contiuity, and Bias Control.
Computer Graphics 18(3):33-41, 1984.

[Mortenson 85] Mortenson, M.E.
Geometric Modeling.
John Wiley & Sons, 1985.

[Newman 79] Newman, W.M., and Sproull, R.F.
Principles Of Interactive Computer Graphics.
McGraW-Hill, New York, 1979.

[Olsen 86] Olsen, D.R.
An Editing Model for Generating Graphical User Interfaces.
Graphics Interface '86 :66-70, May, 1986.

[Page-Jones 80] Page-Jones, M.
The Practical Guide to Structured Systems Design.
Youron Press, New York, 1980.

[Parke 82] Parke, F.I.
Parameterized Models for Facial Animation.
IEEE Computer Graphics and Applications 2(9):61-70, November, 1982.

[Pearce 86] Pearce A., Wyvill B., Wyvill G. and Hill D.
Speech and Expression.
Graphics Interface '86 :, May, 1986.

[Pentland 86] Pentland A.P.
Part Structure for 3-D sketching.
Graphics Interface '86 :223-228, May, 1986.

[Requicha 80] Requicha, A.A.G.
Representations for rigid solids : theory, methods and systems.
ACM Computing Surveys 12(4):437-464, Dec., 1980.

[Ridsdale 86] Ridsdale G., Hewitt, S. and Calvert, T.W.
The Interactive Specification of Human Animation.
Graphics Interface '86 :121-130, May, 1986.

[Rogers 76] Rogers, D.F. and Adams, A.A.
Mathematical Elements for Computer Graphics.
McGraw-Hill, 1976.

[Rossignac 85] Rossignac, J.
Approximation of canal surfaces with standard CSG primitives.
PhD thesis, Rochester Univ., 1985.

[Tiller 83] Tiller W.
Rational B-Splines for Curve and Surface Representation.
IEEE Computer Graphics and Applications 3(6):61-69, Sept., 1983.

[Tremblay 76] Tremblay J.P. & Sorenson, P.G.
An Introduction to Data Structures with Applications.
McGraw Hill, 1976.