A Search Facility for a New Zealand Sign Language Dictionary

Shuyi Scott

Submitted in partial fulfillment of the requirements for Bachelor of Science with Honours in Computer Science.

Supervisor: Peter Andreae

Date: 17 October 2003

Abstract

This report is submitted for comp489. It is an honours report on a search facility for a New Zealand sign language dictionary. This search facility is based on the three dimensional representation of motion. The graphical user interface is designed to allow a user to interactively control the character from five different views by using mouse and keyboard. The interactive control techniques allow a user to search for the meaning of a sign by specifying a "broad" description of motion. The motions with similar features will be grouped together. Four example movement shapes have been investigated and implemented. These are, line, curve, circle and a group including wavy and zigzag shapes. A prototype with limited search capabilities has been developed for demonstration and testing.

Acknowledgement

I'd like to thank my supervisor, Peter Andreae for his great help with my project, as well as his understanding and kindness. I'd also like to thank the school of mathematics and computer science for providing me with the scholarship for the honours degree year.

1. Introduction

The purpose of this project is to investigate the design of a new interface for searching a sign language dictionary that will help sign language learners to use the dictionary easily. This search facility uses interactive control techniques to allow a user to specify search requirements in a "broad" description of motion to retrieve the search result of a sign or signs from the database.

1.1. New Zealand Sign Language.

Sign Language is not a series of gestures and mime only. It is a real language. It has its own vocabulary and grammatical structure. A comprehensive and meaningful idea can be expressed by combining phonetic handshapes, movements, hand and finger orientations with various non-manual movements or features. This is quite different from spoken languages. Sign Language doesn't involve talking and listening. It is a visual-gestural language. The only way for deaf people to learn a sign language is visually. Therefore, movement and position components are very important in sign language.

1.2. Limitations of the existing New Zealand Sign Language Dictionary.

Victoria University (Deaf Studies unit, Applied Linguistics Department) and the New Zealand Deaf Association produced a dictionary of New Zealand Sign Language. This dictionary collects more than 4000 signs that are used in New Zealand. There are 129 handshapes, grouped into 27 categories in the dictionary according to dominant handshapes (usually the right hand). Within each handshape category, signs are ordered according to the location on the body. A standard transcription notation known as Hamburg Notation System (or simply Hamnosys), developed by the Centre for German Sign Language, also attempts to specify each of the signs. Hamnosys describes the handshapes, orientations, body locations and movements of signs. Movement is a salient feature of many signs. The movement involves describing a shape with an articulator, eg a hand or finger. A sign might, for example, involve a sweep of the hand from a particular start position forming a circle.

This existing dictionary is a good tool to help sign language learners study New Zealand Sign language. However it has the following features that limit its usefulness:

- The dictionary includes drawings of each sign as images and in Hamnosys. Therefore the physical book is large. It is difficult to carry and use.
- In an attempt to naturally index signs, the dictionary orders signs based on properties (handshapes and initial position) of a sign, not on an alphabetic ordering in English.
- Look up by sign in the dictionary is slow and in fixed order: first by handshape and then by starting position. This is not natural for novices.
- The "English to New Zealand Sign Language" lookup simply uses an English-order index with references back to entries in the "Sign Language to English" section.

- The dictionary doesn't address searching for a sign by the movement. In the dictionary, movements are represented by arrows and symbols on the drawings rather than by actual movement. It is understandable that the indexing structure of a paper dictionary ignores the movement completely, despite its importance to signs。 Learners of sign language may find it easier to remember positions and movement rather than handshapes and orientation.

Due to the above features, the dictionary is not easy to use and finding the English translation of a sign or signs can be very slow.

1.3. The task: enable searching for a sign by specifying movement and positions

The task of this project is to develop a more natural and efficient system that enables a user to search for signs by specifying movement and positions. The system accepts user specified movement and position as search requirements of signs and categorises the inputs to allow a query of the dictionary and display of the matching sign or signs on the screen.

To provide a natural interface it was decided to provide a 3D figure of a human character that would respond to user input by performing the required movements. The most natural user interface therefore allows the user direct manipulation of the human figure to specify the movements. That is, these movements are to be specified by the user simply moving the figure's hand in the desired 3D movement. This direct manipulation interface will be easier to use than a point/click/menu/select/type interface.

The central component in this task is therefore;

- Be able to specify movement of signs by using mouse movements to directly manipulate a human figure on the screen rather than keyboard or written description.

1.4. Key issues:

To meet the project goals outlined above, this project had to address the following key issues:

- How to specify 3D movements and positions with a standard computer interface with a mouse and a 2D screen
- How to animate the figure as the user moves the mouse in 2D, while giving the appearance that the user is simply leading a 3D human figure's hand.
- How to acquire or construct a sufficiently good human figure.
- How to allow the user to input signs which involves both hands.

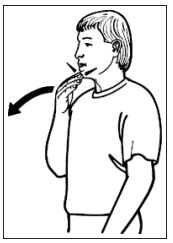
1.4.1. 3D movement, 2D mouse and 2D screen

The central problem in this project is how to naturally represent a 3D movement on the 2D screen with a 2D mouse.

Figure 1, below, shows a couple of signs from the New Zealand Sign Language Dictionary to illustrate typical 3D movements. The sign in the first figure shows a two-part motion of the dominant hand acting in relation to the other arm. The sign in the second figure shows a one handed motion.

Thankyou: thank you, thanks

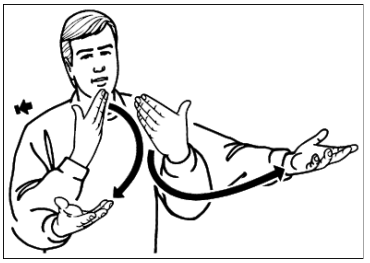
Use:
(V) Mary thanked him for the present.
(Phr) Thank you for the flowers.
(Npl) Thanks for your help.



Recipe:
(Dir)
The right hand is raised so that the fingertips touch the chin, palm facing the signer, blade left, fingers curved, and is moved forward/down.

Show: demonstrate, display, exhibit, prove, reveal

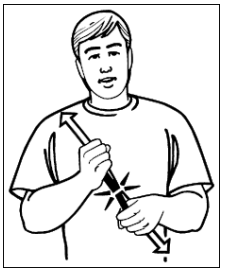
Use:
(V) I showed my mother the letter. Lucy will demonstrate how the machine works. He revealed why my car wouldn't start. Columbus proved that the Earth is not flat.



Recipe:
(1 or 2)
Both flat hands are held up, some way apart at upper chest level, palms up/back/out, blades down/back/in. The hands are moved down/forward/out to palms up, blades back/in.
{Body leans right.}

Bank: bank account

Use:
(Nc) Liz opened an account at the bank. Which bank do you use?
(Phr) My bank account doesn't have much in it.



Recipe:
The left fist is held out, palm facing the signer, blade down/left. The right fist, palm facing the signer, blade down/left, is moved sharply down/left so that the blade strikes the thumb joint of the left fist as it moves up/right. Then the left fist moves down/left as the right moves up/right.

Figure 1: Typical signs from the dictionary

The following observations about the properties of signs and movements were critical to the addressing the design issues in the project:

- Although there are signs with movements in all directions and many positions, a very large number of the signs involve movement in a single plane or surface.
- Although this surface may be different for different signs, there is a natural correspondence between the comfortable viewing angle on the screen and the movements required. Therefore by allowing the user to change their view angle on the figure, and connecting this with the selection of the surface for the movement, the user will always be able to specify the movement by a 2D movement transformed onto the appropriate surface.
- The natural constraints of the human body and the use of Hamnosys to specify the signs mean that we only need a coarse specification of the movement, with a fairly limited number of view angles, in order to identify the desired motion.

1.4.2. Inverse kinematics: keeping bodies together

One of the issues this project had to address is movement control. Most of the available computer packages for 3D that support 3D human characters allow them to be posed. Posing is usually accomplished by bending elbows and moving arms at the shoulder, ie, the desired position is accomplished by specifying or controlling the joint angles. Movement in this way by specifying joint angles in a connected model is called “kinematics”. For this project, this is an unnatural and inadequate approach. This project involves the inverse problem: To move the hand to a desired position, or sequence of positions in a movement, what are the appropriate angles for joints? Work from robotics and some computer animation on this “inverse kinematics” problem has been adapted and implemented to allow the user to manipulate a 3D computer model of a human character. This is described in more detail below.

1.4.3. Signs with both hands

One of the problems in this project is to move both arms to show a sign. According to New Zealand Sign Language dictionary, the majority of signs involve either only one hand or both hands moving in mirror or parallel actions. Therefore, one mouse is adequate to perform these signs. This leads to a solution to this problem which is to categorize the signs into one of these types: “One hand”, “two hands parallel”, “two hands mirrored” and “two hand others”. As explained in section 2.4, in New Zealand Sign Language the “Two hand others” is also simple. While one hand, the dominant hand, signs a movement, the other, non-dominant hand, maintains a fixed position. This type is therefore implemented as two steps. First move the left arm (non dominant arm in this prototype) to its position, and then move the dominant arm.

1.4.4. 3D Graphics

The selection of 3D models and 3D graphics rendering package is also a big issue in this project. Graphics applications have been developed over recent years that allow representation of 3D shapes as computer models or for rendering. There are two ways to acquire a 3D figure: get an existing one or build one yourself by using commercial 3D packages. Implementation of a 3D human figure is easier than previously. A custom designed figure would provide the most pleasant user interface, but it would need substantial work to design the figure which is beyond the scope of this project. Also good 3D packages are expensive and can be difficulty to learn. On the other hand, pre-constructed 3D computer models representing human characters are available from a number of sources and in many styles.

Selection of the graphic figure and graphics applications was limited by the following;

- The budget could not stretch to big packages & expensive figures
- Goals for this version only require a natural look to the figure rather than an accurately realistic figure.
- A figure must be in a format that can be read and manipulated
- The requirements for the figure must not increase the complexity of the project substantially.
- Time constraint of this project

Due to the above limitations, a free downloadable graphic figure was used. This figure has become known as Hanna. Hanna consisted only of an exoskeleton of graphic components such as limbs, coat, etc. As downloaded, these were positioned to create the look of a 3D figure.

Java3D has been used in this project to provide a convenient and portable 3D display capability. Java3D provides the display interface but does not directly support the representation of connected 3D objects as required. Data structures were developed and implemented in Java to add the capability to model the human character as a 3D connected model and implement the inverse kinematics solutions controlling the display of 3D graphics. These representations of the human character, inverse kinematics and user interactions developed for this project are described below.

1.5. Related work and extensions

There are three kinds of work directly related to this project. The first involves other sign language dictionaries that provide a “search by movement” facility; the second involves work on inverse kinematics problems and the third on interactive 3D graphics with human figures.

1.5.1. Electronic Sign language dictionaries.

Related work in this area can be classified into two types. One type includes web-based, on-line projects that allow multiple users to access the system at the same time and query the system, e.g. [7] and [8]. These currently provide limited vocabularies. The other type includes stand-alone CD based

single-user systems. These can be installed or run from a CD and provide more supporting functions and vocabularies than the first type, e.g. [9] and [10].

The most common approaches used to provide a search facility for signs include the following:

1. Typing an English word,
2. Selecting a subject category,
3. Browsing alphabetically the English translation,
4. Displaying search results as video clips or graphics.

Some other approaches involve:

5. Clicking on an English phrase.
6. Selecting a question type.
7. Searching by parameters such as handshapes, movements, positions etc.
8. Viewing the component breakdown of each sign.
9. Displaying results as animated images.

These systems use a combination of text, still graphic and moving graphic images to display signs. Searching for a sign, given its English equivalent, is straightforward using these computer dictionaries. It is slow and difficult to search for a sign by specifying the position and components of the sign as this requires selection of the desired movements and positions from long lists of those available. This doesn't meet the goal of this project which is to explore an interface design that is easy to use and effective to find a sign.

While use of video clips to display signs makes it easier to see the movement, but it makes it very slow to scan through multiple signs searching for a particular sign or feature. The natural way that people scan rapidly is lost.

1.5.2. Related work from robotics: Inverse kinematics

What is the best way to allow the user move an arm to show signs? The commonly used technique is to solve the kinematics problem associated with controlling the movements of robots, human or animals. In [2], the author said, "Kinematics studies the geometric properties of the motion of points regardless of their masses or the forces acting on them." Inclusion of mass is a degree of detail not required for the project goal.

There are two types of kinematics: forward kinematics and inverse kinematics. Forward kinematics is setting the state vector (joint angles) to particular values. The result is the articulator position. Inverse kinematics is a way to meet the need to animate a figure with joints into a position. Inverse kinematics starts with the articulator desired location and solves for the state vector (joint angles). Setting this state vector, these angles for the figure's joints, positions the articulator at the desired position. Inverse kinematics fits our purpose in wanting to lead the articulator to a position.

The mathematical framework for kinematics and inverse kinematics has been explained thoroughly in [5]. This area has been explored by a number of

people in recent years. There are essentially two approaches to solving the inverse kinematics problem. In some circumstances the kinematics of the structure are simple enough to solve analytically. In some cases, either because of the complexity, or for other reasons, numerical solutions are more appropriate. Numerical approaches tend to represent the joints angles as a state vector and optimise a cost function based on the desired position.

In [1], the authors outline an analytical approach to solve the inverse kinematics of the human arm. They derive the geometry for positioning the joints on the arms of a human figure to achieve a desired position. This article is most relevant to this project. In this project, with some modifications, I adapt the solution in [1] to solve the inverse kinematics problem to move the arms and articulator to the desired positions and through the desired movements.

In [2], the author proposed an approach to include numerical solution of inverse kinematics into Java3D models. While this project implements a direct solution approach rather than a numerical one, the design of Java programming structures and method of inclusion in the Java3D structures implemented here is based on that in [2].

1.5.3. Related work in 3D Graphics

Recent work in 3D graphics, possibly connected to the big budget movie industry, has created a range of low-end graphics packages including both 3D connected models and 3D graphics rendering.

There are thousands of 3D figures on the Internet. They are created for different purposes and driven by different motivations. Most of them are for sale, but some are free. There are a number of commercial 3D packages (Maya or 3D Max etc.) which can be used to build a 3D figure or render human figures and motion. These packages are expensive and can be hard to use. In order to reduce the cost and complexity of the project, I acquired a free 3D figure. This figure is not designed to suit our purpose; however it is sufficient for this project. Demonstration of the principles of the project goal does not require a high degree of detail in the figure.

The combination of 3D models and 3D graphics rendering seen in the available packages is a symptom of different goals. The most common applications of these 3D graphics and modelling capabilities are to provide accurate movement of figures under computer control, eg robotics and animated characters in movies, games etc. The movement is precisely specified and controlled by the computer to achieve an accurate result or desired visual effect.

In contrast, the natural control required for the goals of this project leads to two differences;

- Movement starting and ending positions have “broad” meaning. The distinction between one position and another is not precise. For example:

an area between a nose and a mouth is classified as both nose position and mouth position when forming a query description.

- Broad movement types can be grouped together, e.g.: zigzag movement and wavy movements, distinct in the dictionary, are considered as the same type of movement.

1.6. Organisation of the following sections

The remainder of this report is divided into six main sections. Section 2 addresses the design ideas of the user interface for specifying motion input. Section 3 illustrates how the Java3D structures (scene graph) are constructed to allow Java3D to render our figure. Section 4 explains the technique for recognising the starting position and ending position of movements by defining coarse regions on the figure. Section 5 analyse the movements and classify them into four common movements in the New Zealand Sign Language Dictionary. The last section describes the implementations and algorithm of this project.

2. User Interface Design for Motion Input

An emphasis of the project is on the human computer interface (HCI) for searching by signs. Because New Zealand Sign Language involves simultaneous movements of the body in three-dimensional space, it imposes a challenge on the HCI design. Maximising the benefit to the users will require careful HCI overall and detail design.

One of the interesting, important problems in the project is the difficulty of specifying a 3D motion of hands with a 2D mouse. The solution to this problem is to use a user-controlled animated human figure, with motion constrained to be on certain surfaces, to specify the search criteria. The user chooses the surface just by turning the figure round to get a good view of the movement.

Much of the user interface design was dictated by the requirements to construct a search query based on movement and position.

2.1. Requirements for a search query based on movement & position

New Zealand sign language doesn't make fine distinction in motion. This is a reflection of the fact that human gestures are a rather imprecise art. Hamnosys made a compromise and defined 17 directions of movements. In the prototype, we adopt these 17 directions. Analysis of the signs in the dictionary and described by Hamnosys suggests;

- a great majority of signs can be specified with only a few parameters, including;
  - broad descriptions for the start and finish positions of movements
  - a very limited number of directions of movements (17)
  - The movement of any one sign tends to be restricted to a surface significantly less than all 3D space.
  - The surface including the movement changes from sign to sign

    - hand shape is not important for many movements
- movements involving both hands form a few subcategories
    - the two hands, and arms, track each other in parallel movements
    - the two hands, and arms, track each other as mirror images
    - actually only the dominant hand movement is of interest, specifying only the broad position of the non-dominant hand
- The movement usually involves only one point, known as the articulator. Eg the hand, or finger traces the movement. Articulators are nominated points on hands which are lead by the mouse when the user performs a movement shape.

2.2. Requirements on the user interface

The user interface must therefore allow specification of the list above to construct a query. Particularly, with the goal of naturalness, the user must be able to

- Nominate an articulator; eg: the hand
- move it to the start position
- move the movement required, seeing the figure execute the action on the screen
- finish at the movement's end position

This is achieved in the natural sense of holding the figure's hand and leading it through the movement. These leads to some simple requirements;

- graphical interface
- realistic (enough) human figure to recognise the signs
- appearance of 3D movement and positioning
- appearance of natural movement for the human figure depicted
- simple, efficient, easy to use, not require typing, not require user training,

2.3. Requirements for specifying 3D motion

Specifically to specify the query the interface needs to;

- Allow the user to change the view angle to one appropriate for the sign of interest
- Allow positioning of the non-dominant hand
- Translate the user mouse 2D motion into a movement on an appropriate 3D surface. These surfaces are not planar.
- Must be able to select the hand, forearm, or upper arm for positioning and movement as the articulator.
- The mouse movement must be captured in sufficient detail to determine the shape of the movement.

2.4. User interface design issues & solutions

A screen shot of the current user interface version is displayed in figure 2. It is designed specifically to meet the above requirements for constructing a query. The right panel shows the figure that the user can control to specify a motion. The left side window will show the search results. Search results have not been implemented yet, because the focus of the project is the specification of

the desired motion. This area currently shows debugging information and partial results.

In this prototype, right hand is the dominant hand; the left hand is the non dominant hand. The current interface will allow the user:

- to move the dominant hand to a starting position, then do the motion.
- or, in the case of two arms (independent), move the other arm to its position, then move the dominant arm through the motion.

Operation of current interface can be described below.

- Click the drop down menu to select hand arrangements: one hand, two hands mirror, two hands parallel or two hands others;
- If it is a one hand sign or two hands dependent sign (mirror or parallel), move the dominant hand to the start position, and release the mouse to identify the start position.
- If it is a two hands independent sign, move the non dominant hand to the start position, and then release the mouse to identify the start position.
- Click the mouse to pick the dominant hand and do the motion.
- Releasing the mouse signals the stop position.

The design issues & solutions for the part of the user interface for specifying the movement of a sign is described below.

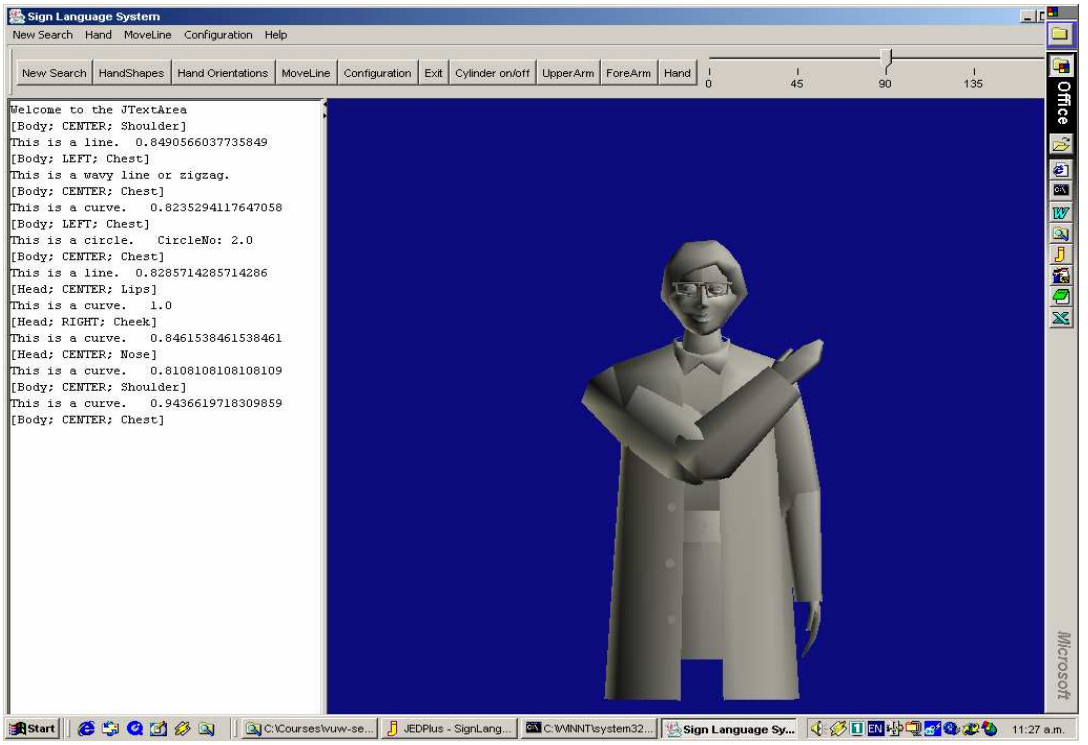


Figure 2: The interface of sign language system

2.4.1. 3D representation of a sign instead of 2D representation.

One of the major design decisions made in this project is to use 3D representation of a sign instead of 2D representation. This decision meets the

requirements on the user interface described above. Movements in sign language are visual and spatial. 2D representation is not enough to present signs clearly. This is especially so when a user is not familiar with the symbols used in sign language. 3D representation can show a sign from different angles and the transformation of a 3D object can be done on the fly. 3D representation is close to reality, it's natural, better and easy to navigate. Although 2D representation has high quality and is fast, the speed of rendering a shape is not the most important factor and quality of 3D display is sufficient in this project. Therefore it is a better choice for developing a computerized search facility for sign language than 2D representation. It supports the user's learning goal efficiently.

2.4.2. Categorized features to facilitate search.

In the dictionary, handshapes and body location are used to look up the meaning of a sign. According to Hamnosys, four categories are used to describe sign languages signs. They are handshapes, orientations, locations and movements. Within the scope and goal of this project the following simplifications are made;

- Hand shape is ignored as not central to the goal of capturing movement,
- Wrist orientation is similarly not central to the goal and is supported, but not changeable from this version of the user interface,
- Only the upper body is displayed, as this is the most interest.
- Starting and ending positions are the most important elements of a motion.
- One-hand movements and two-hand movements have been separated. In terms of two-hand movement, we divide them into three sub groups: two hands mirror, two hands parallel and two hands independent.

The above simplifications solve the problem of selecting of the desired movements and positions from long lists of those available and reduce the movements to a relatively small set of values for matching to the sign language code book. They are sufficient and effective to distinguish the different signs in the sign language dictionary. It is a compromise between good usability and efficiency. It protects a user from selecting features from a long list in order to find a meaning of a sign.

2.4.3. "7 DoF" of movements of arms for showing movements.

Movement of the figure that looks natural is an issue with human figures as even the arms have 7 degrees of freedom (7 DoF). A shoulder has three degrees of freedom. An elbow has one degree of freedom. A wrist has 3 degrees of freedom. However, only 6 degrees of freedom are needed to place a hand at all positions and orientations. This is close to our everyday life.
In positioning the hand in space, several degrees of freedom are not used. These offer a freedom to make the action more natural. In this project, we are only interested in the position of the articulator, joint angles that are not required for a particular movement, as unused degrees of freedom, are implemented but fixed during the movement.

2.4.4. The combination of view angle and surface to Specify motion

Movements in sign language are not random movements in three dimensions. For example, a sign might involve a sweep of the hand upwards to the left of the signer. Such a movement is restricted to a simple vertical plane radial from the user to the left. In a more complicated movement, the articulating hand, might cross in front of the signer. In this case the articulator movement might start to the right and cross to the left. Since the hand must pass in front of the body, the movement may start on a radial plane then effectively move on a circle or ellipse round the body.

Examination of the sign language dictionary suggests that a vast majority of the movements can be expressed adequately by movement of the articulator on a few very simple surfaces. This is also consistent with Hamnosys. These surfaces are not all planar. The implementation of these surfaces in this project is called “movement surfaces”.

When describing a particular movement, the user moves a 2D mouse on the flat screen. The system translates the sequence of mouse position into appropriate 3D positions on the movement surface in the virtual world of the human character.

Most motions are on a surface, but different signs have different surfaces. When using the user interface, the user will most naturally choose a view angle that will allow them to see and perform the sign. There is therefore a natural correspondence between the view angle and the movement surfaces. The view angle is used to select the appropriate movement surfaces.

Therefore the user interface must allow changing the view angle in order to let the user face the desired surfaces. This use of the combination of view angle and surface to specify motion has not been found elsewhere and appears new with this project.

View angle selects subsets of movement planes in a way natural to mouse movement. These subsets include radial planes to the front and/or sides and surfaces close to the body as the articulator passes in front of the body. When the hand moves close to the body to define a sign the surface needs to specify positions in close proximity to the body, for example finger pointing to cheek.

Body surfaces (and intersections of surfaces) are selected by viewing angle and articulator positions. Mouse movements are free to move articulators within these selected surfaces. In this project, we have defined two main body surfaces which have the shapes of a circle and an ellipse if we look at the body from the top. The circle defines a cylinder about the head. The ellipse similarly creates an elliptical cylinder about the upper body.

The movement described by the mouse therefore appears in 3D in the planes over and round body. This motion described in 3D form from the 2D mouse can now be parsed and classified to construct the query.

Five view angles have been implemented to specify all the locations and movements according to Hamnosys. The approach taken here could provide more directions but keep the project (and UI) complexity low only those required by Hamnosys were implemented.

The simple approach using the mouse and movement surfaces allows starting position, ending position and movements to be specified by using the mouse alone. We think this approach meets the naturalness requirements of the user interface. It is easier to understand and navigate than other approaches requiring menu selection and keyboard. It is natural and realistic. A user can use a mouse to drag a hand to a desired position and show the action. A user can have a direct visual result of the actions, for example direction, curve, line movement etc. There is no long list of positions and movements to select.

2.4.5. "Broad" sense of description of search requirements.

In this project, we not only search for a sign which matches the search requirements precisely, but also search for those signs which roughly match the search requirements. One purpose of users is search for the meaning of signs after they have seen them. The approach taken here meets the requirement of constructing a query which is mentioned in section 2.1.

In this prototype, starting positions and stopping positions are the nearest points in 3-dimensional space within a particular range. For example: The hand or articulator is pointing to the nose, the starting position will be interpreted as a position between eyes and mouth. Therefore, we divide the figure into a number of rectangles for identifying the positions.

The same thing applies to movement. Zigzag movements and wavy movements will be treated the same type of movement in constructing a query.

2.4.6. Reasons for choosing hands without fingers.

Initial design idea was to show fingers on each hand. By careful consideration, we think that this approach is unnecessary in this short project, figure details are not the focus. At any one time only the movements of the articulators are of interest. In this prototype we are interested in movements and only need to be able to use an articulator on each hand.

3. Java 3D Scene Graph Design

To allow animation, the character should be a connected set of components so that, for example when the upper arm moves, the graphic for the lower arm should be moved and follow. The downloaded figure components were not in this form. To allow simple replacement of the figure for one more appropriate or sophisticated in the future it was decided to construct Java classes to effectively restructure the figure into a connected model. The lowest Java classes hide the source and actual structure of the figure. This section gives an overview of the architecture used in Java3D to describe the 3D model

and how the inverse kinematics is included. More detail of the Java3D data structures and design are included in the section 6.

3.1. Java3D scene graphs:

Java3D provides a hierarchical tree data structure called a scene graph to describe the items in a 3D scene. The hierarchical structure is well suited to extension to describing connected models and hence well suited to the human figure required for this project.

Nodes of the tree structure include;

- TransformGroups (TG) describe translations, rotations, etc
- BranchGroups (BG) attachment points
- Leaf Nodes (LN) define properties, shapes, behaviors of the items in a 3D scene.

The data structure forms a tree structure without loops. There is only one root and one path between the root to each leaf node. This path defines the state information of any leaf as it defines the transformations from the root to the leaf.

The following subsections briefly describe aspects of the design of scene graph in this project and the structure used to add inverse kinematics capability into the Java3D scene graph.

3.2. Scene graph design and adaptation of the character.

The downloaded animated figure that we are using is not entirely suited to this job, but it is simple and adequate for this version. It only has the human figure as a collection of shapes that form an exoskeleton. It has no hierarchy or attachments. Adaptation for the Java3D scene graph requires that some parameters be derived, e.g. length of limbs. The information on the figure file format is hidden in Java objects. Therefore, this figure should be easy to replace in future versions if required.

The appendix B shows how the scene graph is used to represent the geometric arrangement of the character. (Only the right arm is shown: the left arm will be the same.)

In the scene graph, Leaf nodes with “S” inside triangles represent the shapes for rendering. Group nodes with “TG” or “BG” inside circles describe the structure of the tree. Each TransformGroup object controls one of the transformations that make up the figure and its allowable motion. They are scale, translation and rotation.

In adapting Hanna, transformations need to be added to attach the limb shapes at the required points. For example, the elbow and forearm must attach to the lower end of the upper arm. Transform groups are included in designed Java limb objects to allow attaching subsequent limbs to “the other end of the limb”. Limb objects also encompass shapes for rendering. Joint objects hide abstraction to transform groups, allow angles to be used in terms of human joints, regardless of co-ordinate system. Limbs & joints provide the start of a

mechanism for easing substitution of the figure with more appropriate and complex one for later stages. Such could include work on figures based on bones and skins.

3.3. Inverse Kinematics to move arms.

Actually, the problem in this project is the inverse kinematics – knowing the path of the end point of an arm, work out what the joints are. A design issue was how to best build inverse kinematics into the Java3D scene graph structure. In the path from root node to leaf node, the sequence of subsequent transforms of transform groups encountered describes the rotations and translations relative to those above. This is well suited to forward kinematic control of a connected model, as in many robotics problems.

Positioning the hand relative to the shoulder requires setting the various joints of the shoulder, elbow and wrist. Following [1], the inverse kinematics implemented defines limb positions in terms of the joints, not Java3D transform groups. Joints translate these into the implementation using transform groups.

An inverse kinematics controller to position a hand will therefore control more than one joint and more than one transform group in the scene graph tree structure. The design of the scene graph and associated controller was adapted from [2].

The detailed calculation of rotation angles for the shoulder and elbow can be found in [1]. Some changes in the mathematics have been made in the implementation for this project for convenience.

In this prototype, as in the paper, it is considered the kinematics works on two joints: shoulder joint and elbow joint. The wrist joint is considered fixed when calculating the angles to move the upper arm and forearm together as an unused degree of freedom. By doing this, the complexity of the project is reduced but is still adequate for our purpose. Control of the wrist is not required for this project. The wrist joint and transform groups are implemented, but not controllable in this version.

The coordinate system is also a little different, having adopted that which came with the downloaded human figure model. The coordinate system used is such that the origin is approximately at the figure's centre, the z axis is vertically upwards, the positive x axis points out directly to the figure's left, leaving the positive y axis pointing directly behind the figure. When our character is facing the user, the co-ordinate system is as shown in figure 3 below and the origin is at the center with y pointing into the page.

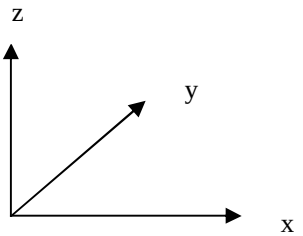


Figure 3 Coordinate system in the prototype.

4. Analysis of movements

The aim of this project is to search for a sign by specifying motion inputs. Two important motion inputs are positions for start and end, and shapes of the movements. These features form part of the motion query language.

4.1. Define coarse regions on the character for positioning

One of the features that distinguish different signs is the starting and ending position of a movement. Two important facts guide the recognition of these positions used to generate motion queries. The first is that these positions can be adequately recognised with coarse regions. The second is that the user input with the mouse is imprecise.

Because it is not easy to distinguish adjacent areas and human movements are not precise, sometimes it is hard for a user to specify the exact starting and ending positions with a mouse. It is not practical to accurately define all the positions the same as those in the dictionary. For example, it's difficult to define the boundary of left nose and centre of nose. It's better to ignore left and centre when identifying positions on the nose. The same thing happens with movements starting or ending near the neck. Therefore, the result of parsing should consider this and take care of both left and centre when translating into the motion query language used to search a sign in the database. It is also difficult to define the boundary between a nose and a mouth. However, it is much easier for a user to position the mouse on large areas, such as left shoulder, centre of shoulder and right shoulder.

By considering this, a coarse grid is effectively placed over the figure to identify the positions. See figure 4. This grid is constructed in the 2D mouse co-ordinate system. For any particular view the possible mouse movement area is divided into small rectangles about the figure. This approach is simple, easy to implement and understand. It provides a coarse positioning sufficient to meet the requirements of the project.

Currently this position recognition ability has been implemented only when the character is facing the user. This proves the concept in the prototype version without taking excessive project time on implementation.

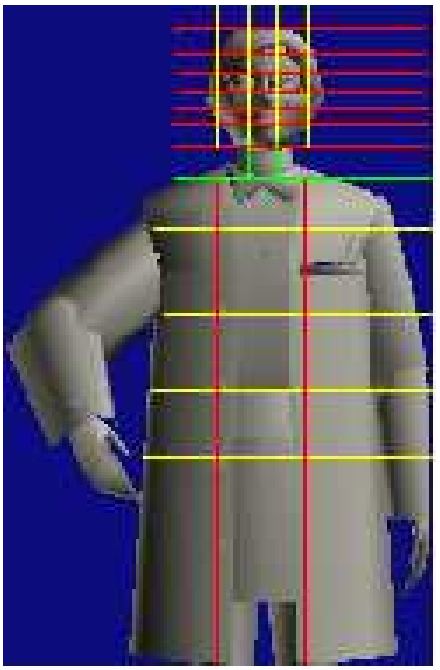


Figure 4: Coarse region for positioning

Therefore we can collect the position data and translate them into the position description format for forming the motion queries.

4.2. Movement Analysis

Data from mouse movements are collected and used to calculate parameters to categorise the movements into one of the coarse movement shapes expected. In this project only four movement shapes are implemented. These are line, curve, circle and a group including wavy and zigzag shapes. Because this is a prototype for exploration and demonstration, the four typical movements are sufficient to demonstrate the capability. They are the most common signs in New Zealand Sign Language Dictionary. Most signs involve one of these four movements. By considering fewer motions, implementation and test are shortened.

4.2.1. Analysis parameters

The mouse movement, combined with any parameters defining the movement surface, therefore represent a reduced complexity description of the movement. This project takes the approach of collecting data from the mouse movement and then identifies the movement type rather than taking data from the resulting 3D movement.

Much of the information to distinguish the four shapes is present in the direction of the mouse movement and trends of this. The prototype user interface was used to perform mouse movements and the data collected. A number of parameters have been studied using this mouse movement data.

The most useful of these parameters is a measure of instantaneous angle. At the end of a mouse movement, the program has a sequence of positions in the

screen co-ordinate system. The angle of movement between adjacent positions is calculated.

These angle values are considered as the most appropriate parameters for this project for classification of movement shape. The four movements of this project, mentioned before, have distinctly different features to their angle values. These features can be seen easily when we plot the angle values. The derivative of this data versus time is also used.

The following section illustrates some of the raw data collected from sample mouse movements and outlines the derivation of parameters to aid classification of shape.

4.2.2. Trends of angle values are used for analyse the movement.

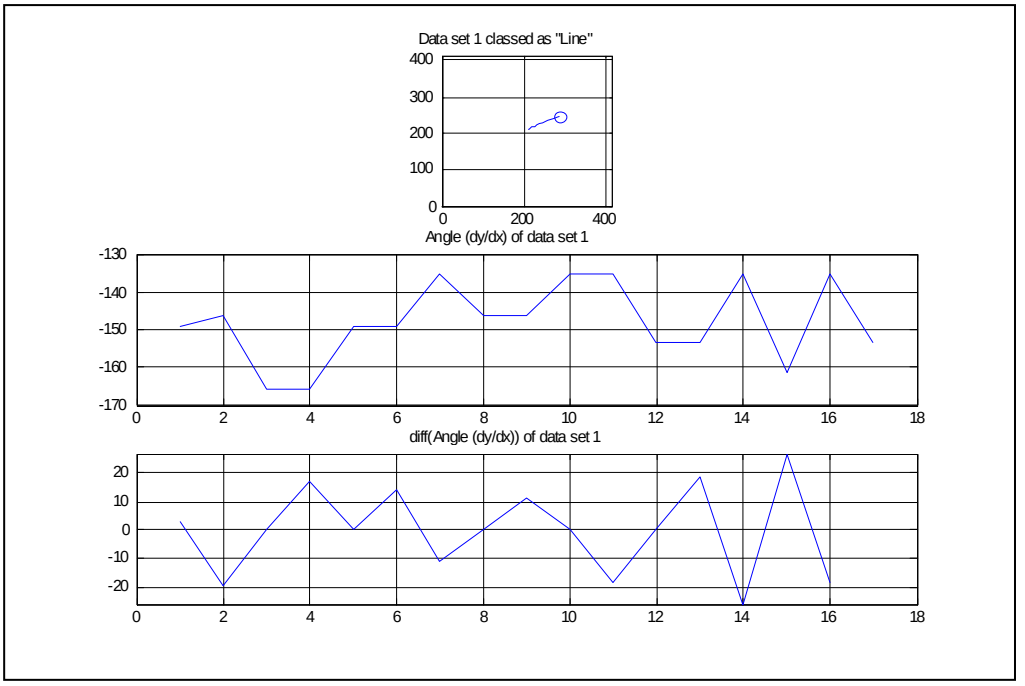
Each of the following four figures represents one example of the four different movement shapes: Line, Curve, circle and wavy&zigzag. These captured examples of real mouse movements have been used to set arbitrary thresholds on the parameters for classification of the movement shape.

In each figure, the top graph shows the shape of a mouse movement. It simply plots the collected points in order, connected by straight lines. The start is marked with a small circle. The middle graph plots the angle values of a movement relative to the screen coordinate system. The bottom graph plots the derivative of these angle values.

4.2.2.1. Parameters for line movements

Figure 5 below illustrates angle values for a line movement. As would be expected the angle values do not vary much. A computer generated line would be straight. The points would progress from start position to stop position without any variation in direction angle. The human user moving the mouse will have some variation in angle. The variation of individual angle samples in the movement is normally within 50 degrees (25 degrees either side). The general trend of the angle values is a horizontal straight line about the mean direction. The variation of the derivative values is normally between 20 degrees and –20 degrees. These variations are caused by the user not moving the mouse straight. The irregular rates at which the mouse routines generate events reporting position also affect the derivative data.

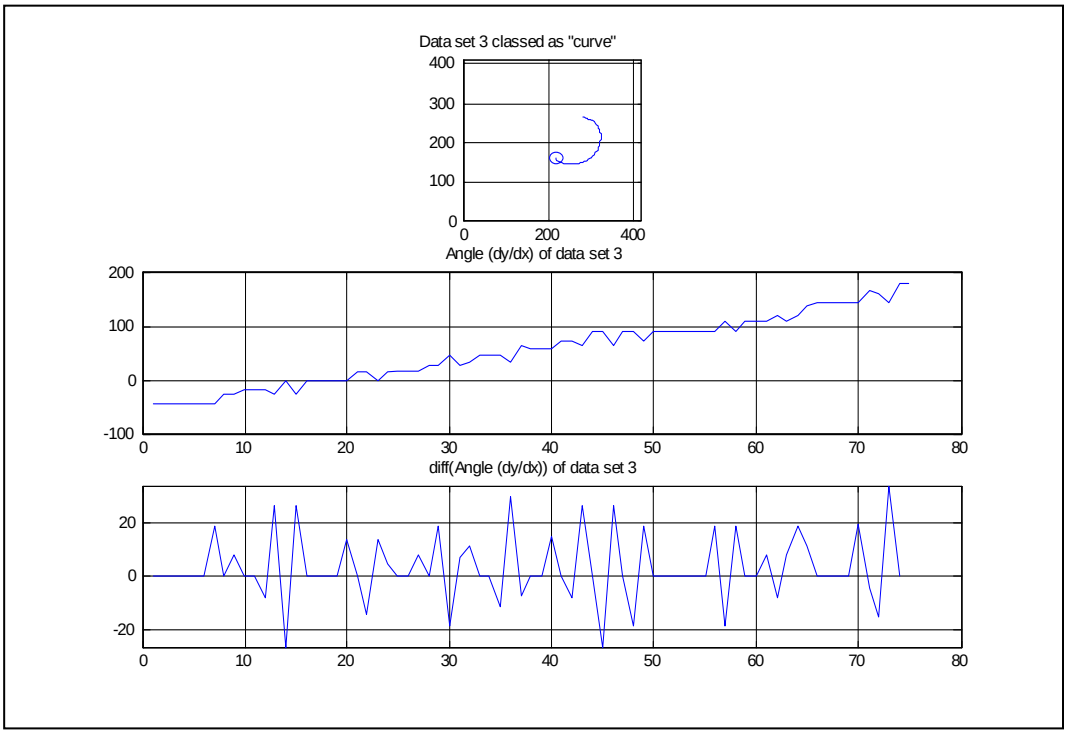
Figure 5: Parameters for line movements



4.2.2.2. Parameters for curve movement

Figure 6 below shows an example of a curve movement. The difference between the highest angle value and the lowest angle value is normally within 360 degrees. The curve does not complete a revolution. The general progression of the line of the curve causes a trend of angle values is either up or down, depending on rotation direction. The difference between the highest value of the derivative and the lowest value of the derivative values can be above 20 or below 20.

Figure 6: Parameters for curve movements



4.2.2.3. Parameters for circles

Figure 7 below shows an example of a circle movement. The difference between the highest angle value and the lowest angle value is more than 360 degrees. A circle completes more than one revolution. The general trend of angle values is either up or down, but it is towards one direction. The difference between the highest value of the derivative and the lowest value of the derivative values can be above 20 or below 20.

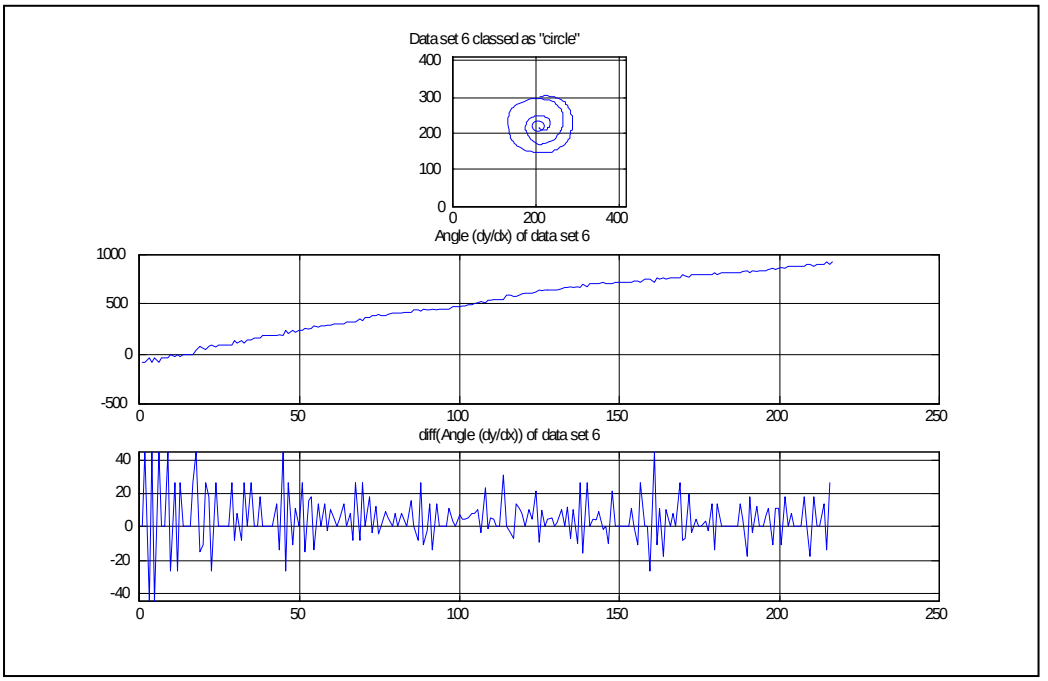
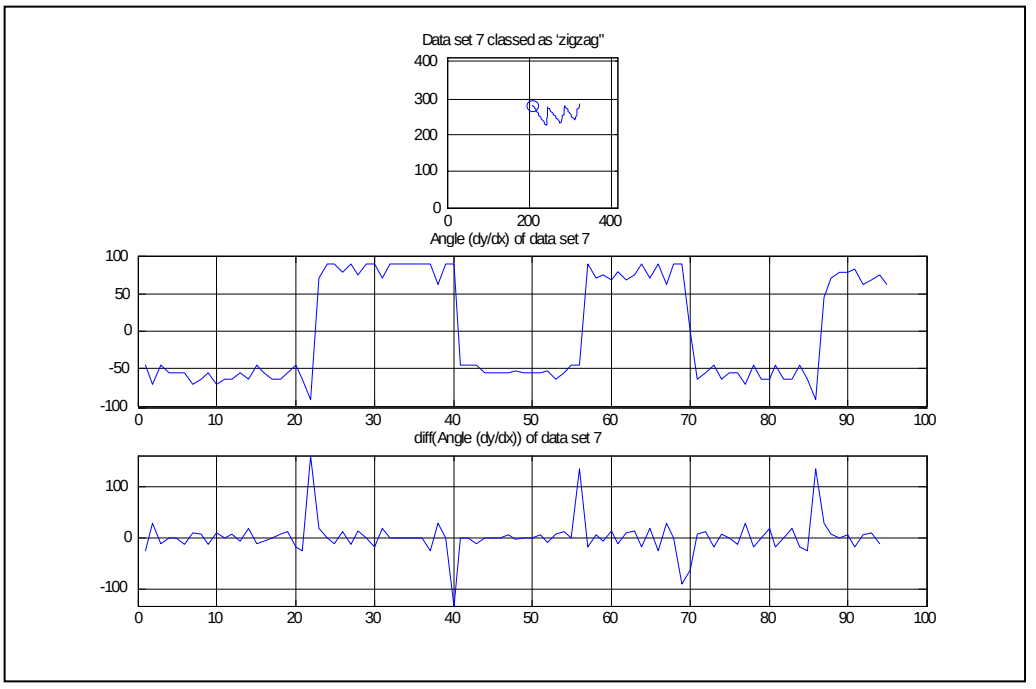


Figure 7: Parameters for circle movements

4.2.2.4. Parameters for zigzag and wavy lines

Figure 8 below is for a zigzag&wavy movement. The difference between the highest angle value and the lowest angle value is normally more than 360 degrees. The signs of angle values are changing alternatively. This indicates the direction change in a movement. There are some sudden jumps in the derivative. The difference between the highest value of the derivative and the lowest value of the derivative values can be above 20 or below 20.

Figure 8: Parameters for zigzag and wavy lines movements



4.2.3. Error reduction technique:

The graphs in the previous section illustrate the typical variation in the angles from a mouse. The data shows significant variation element by element while the trends are much clearer. Ways to smooth the individual parameter samples were explored.

The effect of using non-adjacent values was explored as a way to better measure the trend and reject some of the fluctuations seen in individual values.

The calculation of angle for adjacent points was compared with values at several separations, termed the "span value". Span values of 1 (adjacent), 2, 4, 8 and 16 were compared, with results illustrated below. A span value 4 was selected experimentally as the optimum value because it reduces the error and still keeps the characteristics of the data sets. See figure 9 below.

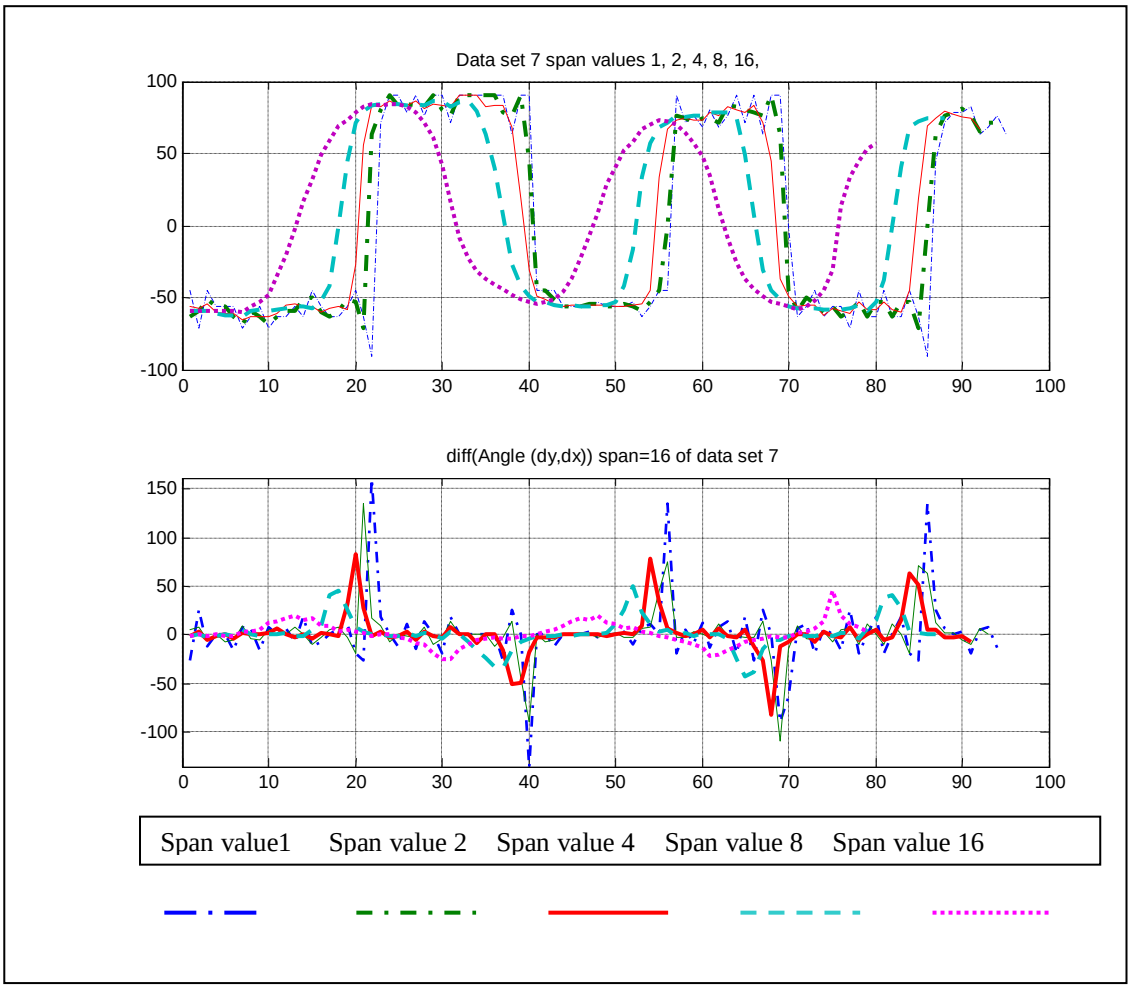


Figure 9: Span values of 1 (adjacent), 2, 4, 8 and 16 comparisons

4.2.4. Sudden jumps in angle with line movements.

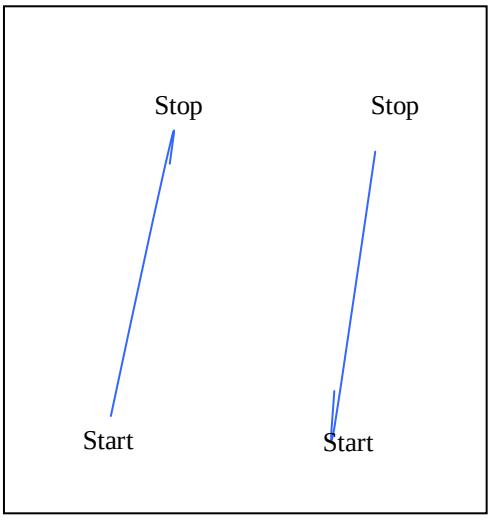


Figure 10: Sudden jumps in angle with line movements

When the user attempts to trace a line, it is common to have a short section (maybe one mouse point) in the wrong direction, as the mouse movement gets started or comes to a stop. This causes as sudden jump in the angle data. The trace on the left illustrates a jump at the finish of a line. The trace on the right shows one at the start. These jumps have a nature different to the random movement fluctuations along the movement shape. They are therefore dealt with separately in this prototype. This kind of error was found experimentally when data was collected for the analyses above.

This is a side-effect of using the angle as a parameter. The angle parameter is susceptible to small mouse movements in the wrong direction. It has no sense that the movement was a small distance. Beyond the scope of this project would be to look at improved processing of the angle data or alternative parameterisations of the movement.

4.2.5. Threshold value for angle value jump detection set by experiment.

A threshold value 72-degree is used to indicate a sudden angle value jump in a movement has appeared. This threshold value was derived by experiment. It was chosen by considering many experimental results. This threshold value depends on the mouse movement on the screen. Actually this experiment to set threshold value should be done by many real users and derive the appropriate values.

5. Implementation and algorithm

This section provides details of implementation of the character, the specific algorithms used for movement shape parameterisation, classifications and for the various inverse kinematics.

5.1. Implementation of the figure as a connected model

Java classes were located to read the 3D graphics file of the downloaded figure, Hanna. The classes, representing the figure graphic elements, created from this were manually analysed to determine which classes were which limb and their positions. Each component in the downloaded format was already placed appropriately for display: that is the fore arm appears placed at the end of the upper arm. Java3D effectively creates a new coordinate system for each component. Java classes were implemented to encapsulate each of the required graphic components into an object with the attachment point, or joint, at the origin. Additionally each was given a translation to allow the next limb to be placed at a new origin at the other end of the limb. For example the upper arm attaches at the shoulder. A limb is created that contains;

- The graphic component for the upper arm
- A translation to move the shoulder joint of the upper arm graphic component from its downloaded position to the origin
- A translation representing the shift from shoulder to elbow, to allow attachment of the elbow joint, ie the length and direction of the upper arm limb.

In this way the details of implementation of graphics, individual sizes and shifts are hidden from the programs that just want to manipulate the limbs. These classes are designed in a way compatible with the Java3D representation of 3D scenes, the scene graph. The body components, limbs etc, included in the scene graph structures can be rendered by Java3D without special treatment.

5.2. Dummy elements in the scene graph as anchors.

Java3D includes some convenient functions that manipulate transform groups directly from mouse movements. These functions allow programs using forward kinematics to control joints directly from the mouse with very little custom programming. Inverse kinematics control is, understandably, not supported directly. These functions use a reference to the controlled transform group to obtain the appropriate co-ordinate scheme.

In this project some of these functions provide a convenient way to monitor mouse movement. We use these by placing dummy nodes in the scene graph as reference points /anchors for the function to control and to provide the required reference to co-ordinate frames. Java3D compilation of scene graph will maintain efficiency of the graph by removing static elements.

5.3. Analyze the mouse co-ordinate values and distinguish four different movements: Line, curve, circle, zigzag and wavy.

The algorithm is based on angles of lines between mouse points. The angles are calculated with atan2(dy,dx) an hence are limited to ±180 degrees. That is, the data has jumps of 360 degrees when the angle reaches maximum and

minimum values. These angles are unwrapped in the processing to remove these jumps and replace the data with angles that span multiple revolutions.

The algorithm for distinguishing four different movements: Instantiation of parser object

```
Parser (angles) {
    Initialize two vectors to store unwrapped angles and derivatives;
}
Test () {
        While (have vector of more than 4 angles) {
        Calculate and store angle values and derivative values with span value of 4;
    }
     Initialise a circle object, a curve object, a line object, a wavyzigzag object;
     Each object returns a quality factor as a degree of match between the data & it's
     characteristics
     Determine the shape according to the quality factors;
}
```

5.4. Move the whole arm by using inverse kinematics.

In this project, I adopt the inverse kinematics approach which is explained thoroughly in [1] to move the articulator on a hand to a desired position. That is, the user leads the figure by the hand, and the inverse kinematic routines, and 3D rendering by Java3D, create the impression that the upper arm and forearm simply follow naturally. The wrist represents an unused degree of freedom and does not change angle during this action. The figures 11 and 12 are only slightly adapted from those of [1]. In the following two sections, I have summarized the mathematical solutions in [1] as they are implemented in the Java code in this project. For detailed explanation of the geometrical derivation, please refer to [1].

The figures adopt the convention for identifying points on the figures, from [1], S for shoulder, E elbow, W wrist and A for articulator and "· "represents vector dot product.

The arms of the figure are represented by translations and graphics connected at the ends by rotations representing joints. To move all the pieces of the arm about the shoulder to position the hand, we solve for the angles required at the shoulder (spherical polar co-ordinates), the elbow (a single angle) and the wrist angle (here fixed).

The inverse kinematics routine for this case, calculates each of the rotation angles involved in the joints in turn. Figure 11 shows the geometry which solves for the upper arm at the shoulder. Figure 6.2.2 assists the explanation of the elbow angle.

5.4.1. Polar angles for moving upper arm in shoulder co-ordinate system

The joint from shoulder to upper arm is modelled as two spherical polar angles. Drawing a line to the desired position, Phi is the angle between the negative z axis and the line. Conventional definition has phi to the positive z axis. Here this is modified to the negative z axis to better match the mathematics of [1]. Theta is the angle between this line projected into the xy plane and the x axis as

a rotation about the z axis. The Java3D implementation of this then passes the calculated polar angle to the shoulder joint. The shoulder joint implements the polar co-ordinates by transforming them to Java3D transform groups.
These polar angles are then
Polar angle from z axis: $\varphi_A + \omega_0 + \pi/2$;
Polar angle from x axis: $\theta_A + \theta_0 - \pi$;

The calculation, for the above two angles, is derived from figure 11:
$\omega_0 = \pi/2 - \varphi_0$;
$\varphi_0 = \pi/2 - \arccos [(SE \cdot SA') / (|SE| * |SA'|)]$;
$\theta_0 = \arccos [(SA \cdot SA') / (|SA| * |SA'|)]$;
The following two polar angles are calculated by referring figure 12:
θ_A =atan2(A.y,A.x);
φ_A =atan2(sqrt(A.x*A.x+A.y*A.y),A.z);

In [1] the authors point out that this calculation fails in two cases. This prototype doesn't handle the two cases; therefore, it will fail if either of the two cases happens. These two cases are:

1. |SA| = 0, the articulator on the hand is positioned at the shoulder joint. (painful)
2. |SA'|= 0, the articulator on the hand and the wrist lie on perpendicular axes through the shoulder.

Fortunately these are not conditions that occur frequently, and in each case the correct positioning is easily added to the inverse kinematics code.

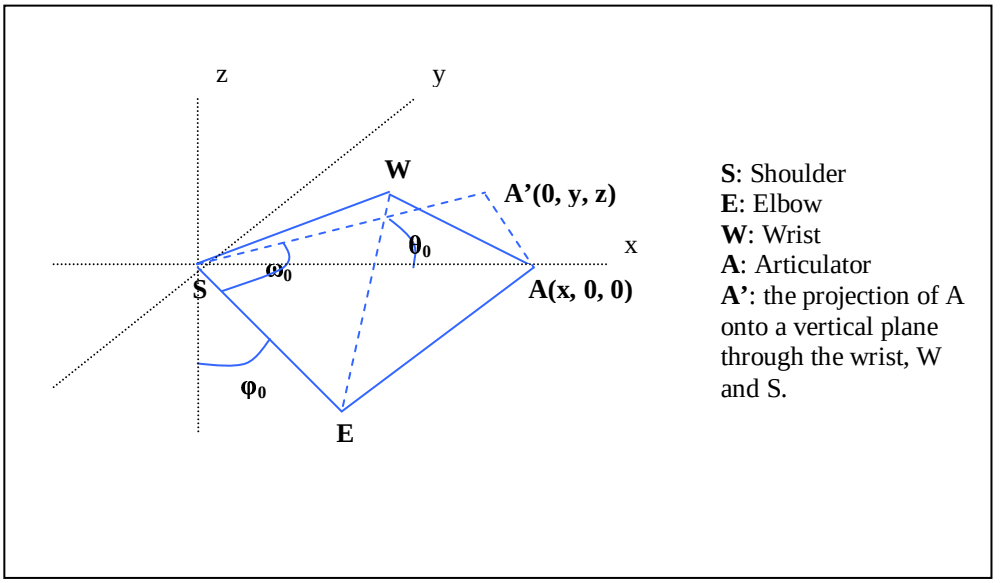


Figure 11: Shoulder coordinate system

5.4.2. Elbow angle for moving forearm in elbow co-ordinate system:

The forearm is then positioned relative to the upper arm, setting the elbow angle. As illustrated in figure 12, this is easiest seen with the elbow at the origin.
Elbow angle: $-(\pi - \gamma)$;

Calculation for the above angle:

$\gamma = \pi/2 - \delta$; δ= arc sin (|EA|/|EA'| * cos α) + ε ;
α = arc cos [(|SE| * |SE| + |EA| * |EA| - |SA| * |SA|) / 2 * |SE| * |EA|];
A = (x, y, z); S = (|SE| * cos δ, 0, |SE| * sin δ); E = (0, 0, 0);
A'= (x, 0, z) = (- |EA'| * sin ε, 0, |EA'| * cos ε);

As pointed out in [1] this calculation also fails in two cases.

1. |EA'| = 0; meaning that the articulator on the hand is pointing at the elbow. This prototype doesn't deal with this problem, but as it is non-physical for most body shapes it is not considered a problem.
2. Abs [(|EA| / |EA'|) * cos α] > 1. This means the point is unreachable, ie beyond reach with this arm length. This prototype version takes care of this situation. Setting the magnitude of the acos() argument to 1 returns the best possible solution: the arm points towards the desired point, appearing to try naturally to follow the users positioning.

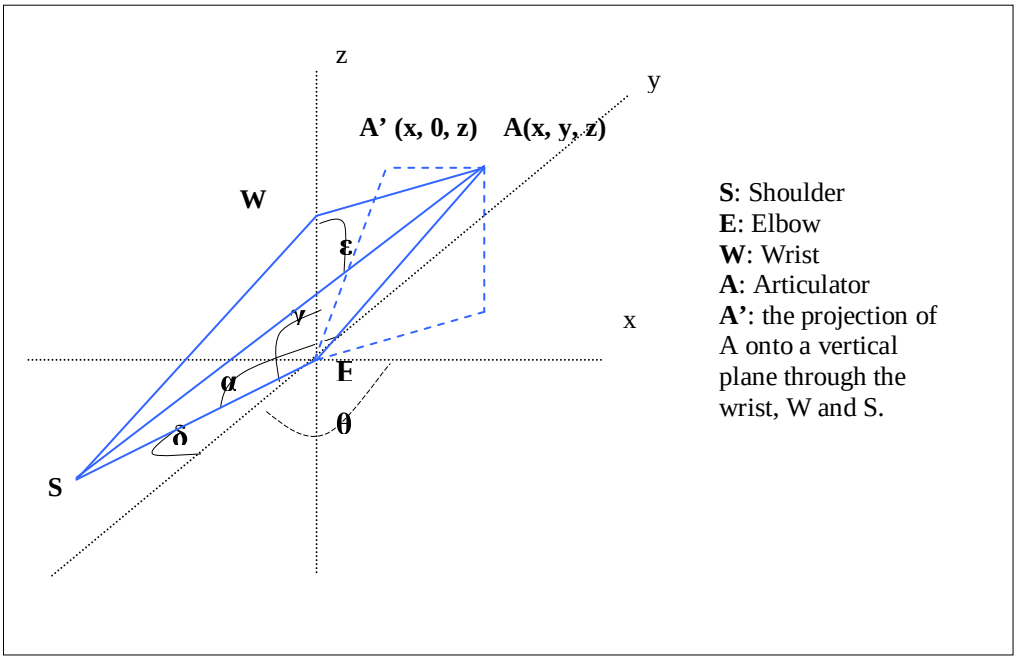


Figure 12: Elbow coordinate system

5.5. Movement surfaces.

The articulators on the hand(s) move on the defined movement surfaces. Surfaces over and round the character's body define sign language articulator positions on and near the body.
This section explains implementation of movement surfaces in this prototype. All the movement surfaces are defined in x and y, irrelevant of the z dimension. That is, they are vertical planes or right prisms with z as an independent variable. Each surface is active only over a region of the body. Within the range of z values over which a particular movement surface is active, the surface shape is independent of the z value. Our explanation of the surface shape can therefore be limited to x and y dimensions.

In the project, the transformation of a mouse position on the screen into desired 3D coordinate system involves the following steps.

1) A mouse position P_1on the screen is transformed into a point P_2 on the image plate. In [6] image plate has been described as "An image plate is the conceptual rectangle where the content is projected to form the rendered image."
2) Point P_2 on the image plate is projected into the 3D point P_3 on a plane through the origin in the virtual world coordinate system. The origin being near the centre of the character.
3) After the above transformation, sometimes point P_3 needs to be replaced if it doesn't lie on the movement surfaces. A new Point P_4 is derived whenever it is necessary. This correction process is illustrated below.
4) The last step, P_4 is transformed into the appropriate point P_5 in the local coordinate system of the joint being moved. This could be shoulder, elbow or wrist coordinate systems which have the origins at the shoulder, elbow or wrist appropriate to the inverse kinematics routine required at the time.

The following three subsections 1, 2 and 3 explain step 3 above for view angles from three different sides.

5.5.1. The image plate is parallel to the xz plane.

The points on the image plate have been transformed onto the xz plane after the third step above. The desired movement surface for the mouse or the hand is along the track of ABCDE (thick black line) in the figure 13 below. There is no correction needed on the track of AB and DE. Corrections are only needed on the track of BCD. The mathematical explanation for calculating the corrected coordinate value (x, y) for a point on a screen after step 3 is explained below.

- For an ellipse:
  a and b are the major and minor axes of an ellipse;
  Define the following points; B(-a, 0); D(a, 0); C(0, -b);
  For a point on the x axis: $x = x_0$ and $y = 0$;
  Function for the ellipse: $x^2/a^2+y^2/b^2=1$;
  Solving the above function with $x = x_0$, we can get the corrected y value.

  For example:
  A point on a screen is transformed into a point P on the image plate. This point is transformed into a point P'(x_0, 0) in the figure 13 below after step 3, but we want to know the point P"(x_0, y). By solving the function for the ellipse with $x = x_0$ above, we can get value of y.
- For a circle, the same approach can be used. I replaced the function for an ellipse with a function for a circle.
  r is the radius of a circle;
  Function for the circle: $x^2 + y^2 = r^2$;

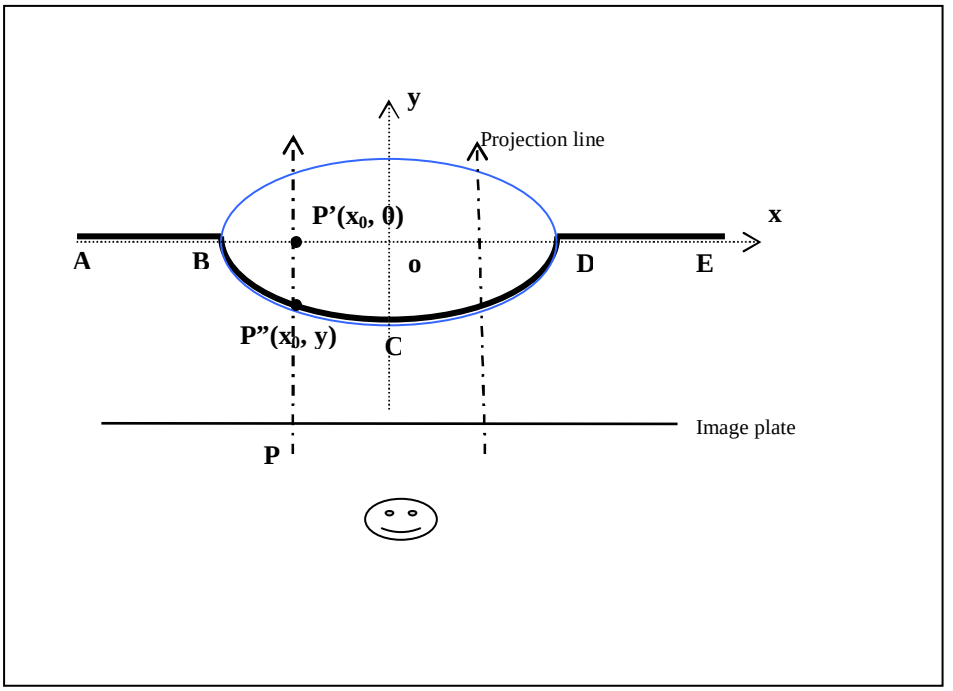


Figure 13: The image plate is parallel to the xz plane

5.5.2. The image plate is parallel to the yz plane.

When the image plate is on the left of the character, meaning that the user is viewing the character from the left, the points on the image plate have been transformed onto yz plane after the third steps above. The desired movement surface for the mouse or the hand is along the track of ABCD (thick black line) in the figure 14 below.

In this case there is no correction needed from C to D. Corrections are needed from C to positive y axis. The mathematical explanation for calculating the corrected coordinate value (x, y) for a point on a screen after step 3 is explained in the table 1 below.

- For an ellipse:
  a and b are the major and minor of an ellipse;
  B (-a, 0); C (0, -b);
  For a point on y axis, x = 0 and y = y_0;
  Function for the ellipse: $x^2/a^2+y^2/b^2=1$;
  Solving the above function with y = y_0, we can get the corrected x value.

  For example:
  A point on a screen is transformed into a point P on the image plate. This point is transformed into a point P' (0, y_0) in the figure 14 below after step 3, but we want to know the point P"(x, y_0). By solving the function for the ellipse with y = y_0, we can get value of x.

- For a circle, the same approach can be used. I replaced the function for an ellipse with a function for a circle.
  r is the radius of a circle;

Function for the circle: $x^2 + y^2 = r^2$;

- When the image plate is on the right of the character (the thick blue line), the same approach is used to derive the corrected coordinate value.
  Function for the ellipse: $x^2/a^2+y^2/b^2=1$;
  Solving the above two functions with $y = y_0$, we can get the corrected x value.

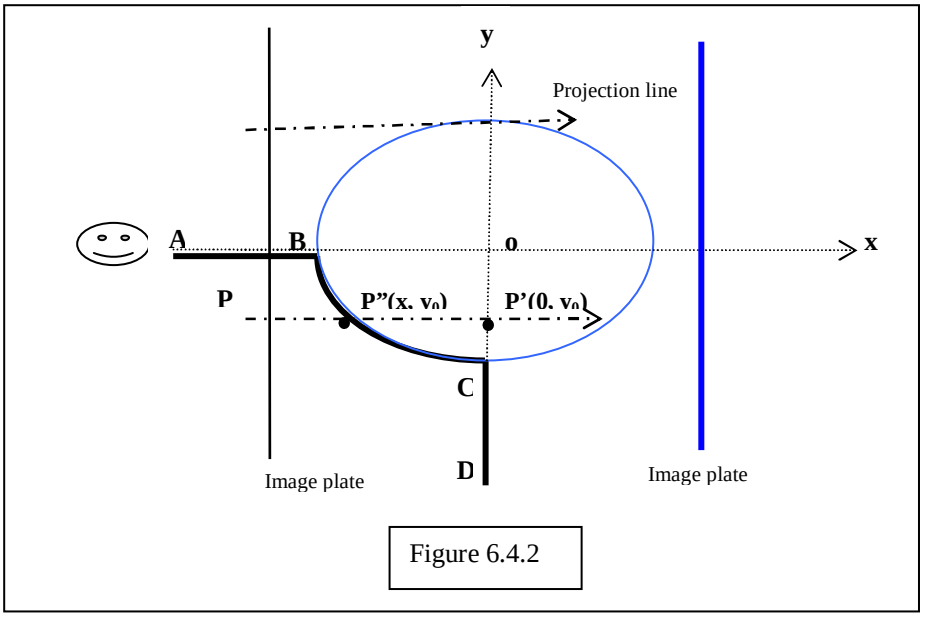


Figure 6.4.2

Figure 14: The image plate is parallel to the xz plane

Image plate on the left of the character	Ellipse shape	y range	y>=0	-b<y<0	y<-b
		Solution function	x = -a; y=0	$y = y_0$; $x^2/a^2+y^2/b^2=1$	no correction
	Circle shape	y range	y>=0	-r<y<0	y<-r
		Solution function	x = -r; y=0	$y = y_0$; $x^2+y^2=r^2$	no correction
Image plate on the right of the character	Ellipse shape	y range	y>=0	-b<y<0	y<-b
		Solution function	x = a; y=0	$y = y_0$; $x^2/a^2+y^2/b^2=1$	no correction
	Circle shape	y range	y>=0	-r<y<0	y<-r
		Solution function	x = r; y=0;	$y = y_0$; $x^2+y^2=r^2$	no correction

Table 1: The image plate is parallel to the xz plane

5.5.3. The image plate is parallel to the plane which goes through the origin and forms 135^0 with xz plane:

When the image plate is parallel to the plane which goes through the dashed black line in the figure below, the points on the image plate have been transformed onto this plane after the third steps above. The desired movement surface for the mouse or the hand is along the track of ABCDE (thick black line) in the figure 15 below.
There is no correction needed from D to E. Corrections are needed along the track of ABCD. The mathematical explanation for calculating the corrected

coordinate value (x, y) for a point on a screen after step 3 is explained in the table 2 below.

- For an ellipse:
  a and b are the major and minor of an ellipse;
  B(-a, 0); C(0, -b);
  For a point on the dashed black line, $x = v_0$ and $y = -v_0$;
  Function for the projection line: $y = x - 2v_0$;
  Function for the ellipse: $x^2/a^2+y^2/b^2=1$;
  Solving the above two functions, we can get the corrected x and y value.

  For example:
  A point on a screen is transformed into a point P on the image plate. This point is transformed into a point P' $(v_0, -v_0)$ in the figure 15 below after step 3, but we want to know the point P"(x, y). By solving the functions for the ellipse and the projection line above, we can get values of x and y.

- For a circle, the same approach can be used. I replaced the function for an ellipse with a function for a circle.
  r is the radius of a circle;
  Function for the circle: $x^2 + y^2 = r^2$;

- When the image plate is on the right of the character (the thick blue line), the same approach is used to derive the corrected coordinate value.
  Function for the projection line: $y = -x + 2v_0$;
  Function for the ellipse: $x^2/a^2+y^2/b^2=1$;
  Solving the above two functions above, we can get the corrected x and y value.

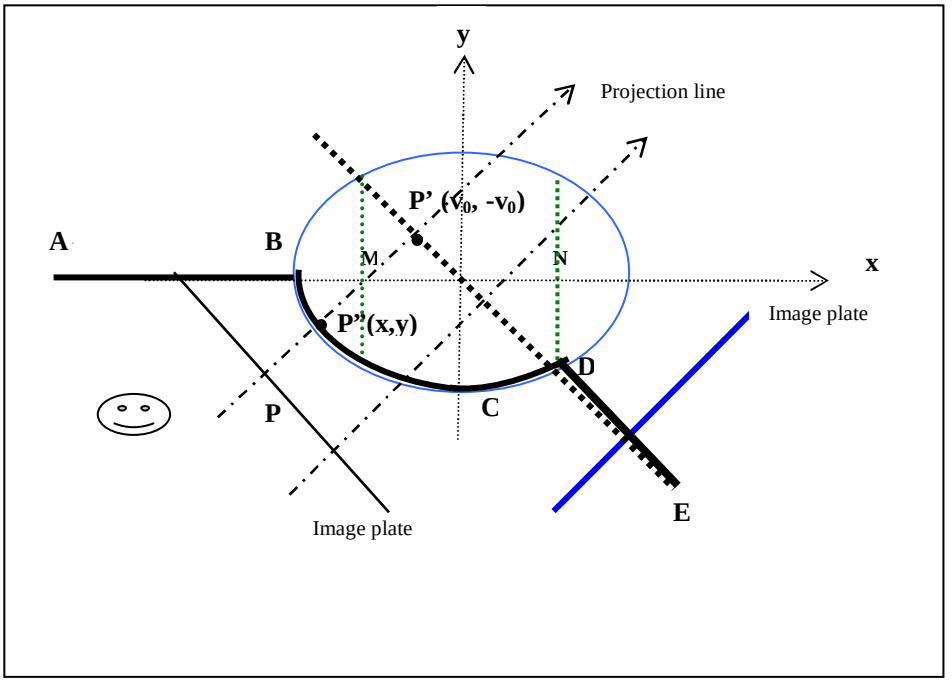


Figure 15: The image plate is parallel to the plane which goes through the origin and forms 135^0 with xz plane

Image plate on the left of the character	Ellipse shape	x range	$x<-a/2$	$-a/2<x<x_n$	$x> x_n$
		Solution function	$x = 2v_0$; $y = 0$	$y = x - 2v_0$; $x^2/a^2+y^2/b^2=1$	no correction
	Circle shape	x range	$x<- r/2$	$-r/2<x< x_n$	$x> x_n$
		Solution function	$x = 2v_0$; $y = 0$	$y = x - 2v_0$; $x^2 +y^2 = r^2$	no correction
Image plate on the right of the character	Ellipse shape	x range	$x>= a/2$	$x_n <x< a/2$	$x< x_n$
		Solution function	$x = 2v_0$; $y = 0$	$y = -x + 2v_0$; $x^2/a^2+y^2/b^2=1$	no correction
	Circle shape	x range	$>=r/2$	$- x_n <x< r/2$	$x<- x_n$
		Solution function	$x = 2v_0$; $y = 0$	$y = -x + 2v_0$; $x^2 +y^2 = r^2$	no correction

Table 2: The image plate is parallel to the plane which goes through the origin and forms 135^0 with xz plane

6. Conclusion

In this project, we have explored an interface design which uses a user controlled animated figure to specify motion input to search for a sign. It appears that this approach has distinct advantages. The user moves the hands to perform a sign which is natural, realistic, simple, efficient and easy to use. It doesn't require typing and user training.

Progress on this project required gaining familiarity on the following related areas: theory of kinematics and inverse kinematics from robotics, analytic inverse kinematics solution to arm movement and New Zealand Sign Language Dictionary.

A free figure was acquired for this project after investigating 3D graphics packages and 3D humanoid characters. The structures for connected models in Java3D have been designed and implemented. Java converter classes have been designed and implemented to adapt the character graphical representation to a connected model structure which can be used in the prototype system.

The theory of inverse kinematics has been adapted to control the movement of the 2D screen image to appear to be the arms of a 3D human character. An inverse kinematics implementation has been incorporated into the Java3D model representation.

A novel approach which uses movement surfaces to specify motion has been developed to describe 3D movements for sign language with the 2D mouse. Different movement surfaces appropriate to Hamnosys have been implemented. Therefore, actions can be performed on these surfaces. Users can move hands over the body of the character. Users can move hands in the shapes of straight lines, circles and ovals.

In addition, an approach which uses the combination of view angle and selection of movement surface minimises the required user actions. Users can see the

positions and motion of a realistic 3D figure by selecting one of five viewing angles through the interface.

The prototype version of the user interface was used to collect real mouse movement data. These data have been plotted and analysed to explore characteristics of the real user movements. These characteristics have been used to design a simple parameterisation to classify the movements into the types expected in sign language and represented in Hamnosys. Imperfections seen in the examples of real movement data have been used set thresholds on the detection of movement types and the classification of some events as common movement errors.

The work has been done in this project leads to some suggestions for the future work and improvements. These include:

- Replacing Hanna with a more sophisticated model with bones & deformable skin.
- Implementing the interfaces to allow user to select handshapes and orientations.
- Testing and developing the user interface with users.
- Enhancing parameterisation and classification algorithms.
- Implementing code to identify additional movement shapes and direction of movements from the recorded motion data.
- Grouping similar movement together to produce appropriate movement shape description which can be used in motion query language.
- Translating Hamnosys strings in the dictionary into appropriate description language for matching. That is, integrate this new user interface with the dictionary database.
- Developing motion query language based on the movement shape data.
- Retrieving matching results from the database by using motion query language.

References

[1] John McDonald, Karen Alkoby, Roymieco Carter, Juliet Christopher, Mary Jo Davidson, Dan Ethridge, Jacob Furst, Damien Hinkle, Glenn Lancaster, Lori Smallwood, Nedjla Ougouag-Tiouririne, Jorge Toro, Shuang Xu, Rosalee Wolfe. “A Direct Method for Positioning the Arms of a Human Model”, (http:// www.graphicsinterface.org/proceedings/2002/144/), 2002.

[2] Fred Klingerner. “Inverse Kinematics on the Java 3DTM Scene Graph”. Brock Engineering 2000, Website. http://www.Vmech.com, Current as of December 5, 2000.

[3] Graeme Kennedy, Richard Arnold, Pat Dugdale, Shaun Fahey, David Moskovitz. “A Dictionary of New Zealand Sign Language”. Auckland University Press with Bridget Williams Books. ISBN 1 86940 164 6.

[4] Sowizral, Henry, Kevin Rushforth, Michael Deering, The Java™ 3D API Specification, Website. http://www.javasoft.com/products/java-media/3D/forDevelopers/J3D_1_2_API/j3dguide/index.html, Addison-Wesley, Reading Massachusetts, 1995 ISBN 0-201-32576-4.

[5] McKerrow, P. J., “Introduction to Robotics”, Addison-Wesley Publishing Company, Singapore, 1991, ISBN 0-201-18240-8.

[6] By Sun Microsystems Java 3D Engineering Team, Java 3D API Tutorial, Website. Current as of October 2003. http://developer.java.sun.com/developer/onlineTraining/java3d/.

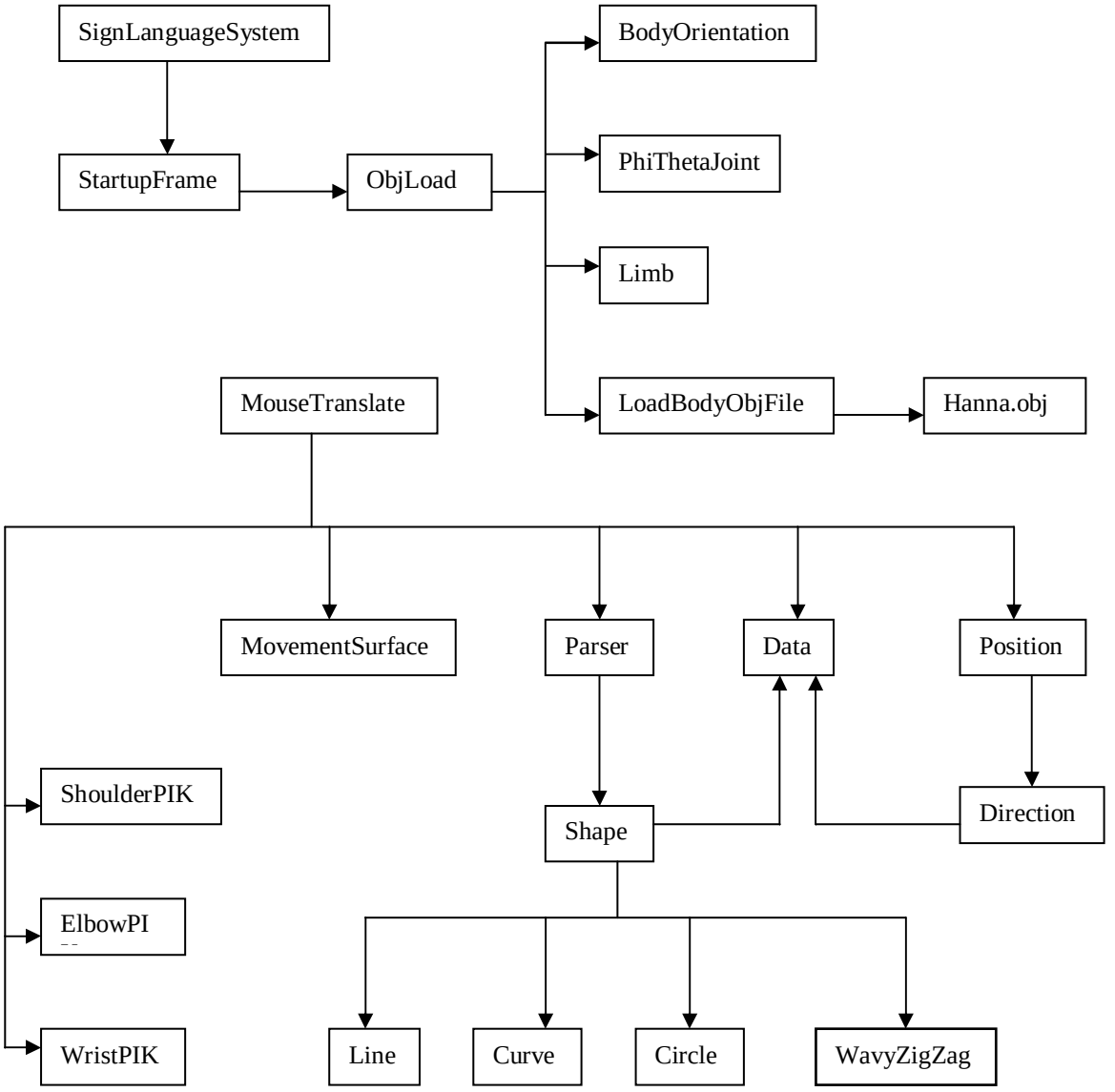
[7] By Comm Tech Lab, Michigan State University, “ASL Browser Web Site”, http://commtechlab.msu.edu/sites/aslweb/browser.htm

[8] Jolanta A. Lapiak, http://www.handspeak.com/

[9] “American Sign Language version 2.0”, Published by Multimedia 2000, updated and redesigned version of The American Sign Language Dictionary, by Martin L.A. Sternberg, Ed. D.

[10] By Comm Tech Lab, Michigan State University, “Personal Communicator CD-ROM” , http://commtechlab.msu.edu/products/asl/index.html

Appendix A: Class Diagram



Appendix B: Scene Graph