Virtual Human Modelling and Animation for Real-Time Sign Language Visualisation

by

Desmond Eustin van Wyk

A thesis submitted in fulfilment of the requirements
for the degree of
Magister Scientiae
in the Department of Computer Science,
University of the Western Cape

Supervisor: James Connan

December 2008

Virtual Human Modelling and Animation for Real-Time Sign Language Visualisation

Desmond Eustin van Wyk

Keywords

3D Computer Graphics
Open Modelling Animation Framework
Virtual Human Modelling Animation
Sign Language Visualisation
SignWriting
MakeHuman
Blender
Python
H-Anim
MPEG-4

Abstract

This thesis investigates the modelling and animation of virtual humans for real-time sign language visualisation. Sign languages are fully developed natural languages used by Deaf communities all over the world. These languages are communicated in a visual-gestural modality by the use of manual and non-manual gestures and are completely different from spoken languages. Manual gestures include the use of hand shapes, hand movements, hand locations and orientations of the palm in space. Non-manual gestures include the use of facial expressions, eye-gazes, head and upper body movements. Both manual and non-manual gestures must be performed for sign languages to be correctly understood and interpreted. To effectively visualise sign languages, a virtual human system must have models of adequate quality and be able to perform both manual and non-manual gesture animations in real-time. Our goal was to develop a methodology and establish an open framework by using various standards and open technologies to model and animate virtual humans of adequate quality to effectively visualise sign languages. This open framework is to be used in a Machine Translation system that translates from a verbal language such as English to any sign language. Standards and technologies we employed include H-Anim, MakeHuman, Blender, Python and SignWriting. We found it necessary to adapt and extend H-Anim to effectively visualise sign languages. The adaptations and extensions we made to H-Anim include imposing joint rotational limits, developing flexible hands and the addition of facial bones based on the MPEG-4 Facial Definition Parameters facial feature points for facial animation. By using these standards and technologies, we found that we could circumvent a few difficult problems, such as: modelling high quality virtual humans; adapting and extending H-Anim; creating a sign language animation action vocabulary; blending between animations in an action vocabulary; sharing animation action data between our virtual humans; and effectively visualising South African Sign Language.

February 25, 2009

Declaration

I declare that *Virtual Human Modelling and Animation for Real-Time Sign Language Visualisation* is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

Full name: Desmond Eustin van Wyk Date: February 25, 2009

Signed:_____

Acknowledgments

I thank God the Lord Almighty for everyday that I live, for everyone that has come into my life, for the challenges that has come my way, for the moments of confusion and inspiration, for EVERYTHING. This thesis would not have been possible without the open technologies Blender, Python, MakeHuman and SignWriting. My supervisor, James Connan, many thanks for the support, guidance, patience and the freedom you have given me while performing this research. Many thanks goes to the staff of the Computer Science Department and the many friends I have made here at UWC. Mom, Jenny, Nita and John, many thanks for the love and support you have given me all my life. I would also like to thank the Centre for Excellence at the University of the Western Cape sponsored by Telkom, Cisco and THRIP for the financial assistance they provided me.

Contents

Glossary

3D – Three Dimensional
API – Application Programming Interface
ASCII – American Standard Code for Information Interchange
ASL – American Sign Language
Auslan – Australian Sign Language
BAP – Body Animation Parameter
BDP – Body Definition Parameter
BSL – British Sign Language
CAESAR – Civilian American and European Surface Anthropometry Resource Project
COLLADA – COLLAborative Design Activity
CSG – Constructive Solid Geometry
DFFD – Dirichlet Free Form Deformation
DGS – German Sign Language
DOF – Degree of Freedom
FAP – Facial Animation Parameter
FDP – Facial Definition Parameter
FBAP – Face Body Animation Parameter
FFD – Free Form Deformation
FK – Forward Kinematics
GSL – Greek Sign Language
GPU – Graphics Processing Unit
GUI – Graphical User Interface
HamNoSys – Hamburg Notation System
H-Anim – Humanoid Animation Working Group
IEC – International Electrotechnical Commission
IK – Inverse Kinematics
ISO – International Organisation for Standardisation
JDL – Joint-dependent Local Deformation
JSL – Japanese Sign Language

LBS – Linear Blend Skinning
LoA – Level of Articulation
MEL – Maya Embedded Language
MPEG – Moving Picture Experts Group
MT – Machine Translation
NURB – Nonuniform Rational B-spline
OpenGL – Open Graphics Library
RFFD – Rational Free Form Deformation
SASL – South African Sign Language
SFFD – Surface oriented Free Form Deformation
SBML – Sign Bank Markup Language
SLNS – Sign Language Notation System
SLV – Sign Language Visualistion
SSL – Slovene Sign Language
SSD – Skeletal Subspace Deformation
STEP – Scripting Technology for Embodied Persona
SWML – Sign Writing Markup Language
VH – Virtual Human
VRML – Virtual Reality Markup Language
X3D – Extensible 3D Graphics
XML – Extensible Markup Language

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The South African Sign Language (SASL) Project at the University of the Western Cape is concerned with the translation of English to SASL and vice versa [74]. The primary goal of this project is to develop technologies that will allow the breaking down of the communication barrier between Deaf and hearing communities. This is being done by developing technologies that will allow for the creation of English and SASL Machine Translation (MT) and educational tools. The South African constitution recognises SASL as the official language of the Deaf [27]. Although this is the case, the Deaf community still have poor socio-economic opportunities and poor access to public and information services. This is mainly because the Deaf community is a minority group and because of the many misconceptions the hearing have about the Deaf and sign languages [27] [43].

Some of these misconceptions are: that there is only a single sign language; that sign languages are merely the visual-gestural representation of spoken languages; that linguistic studies of spoken languages can be applied to sign languagse; and that one can easily write sign language sentences using spoken words [43] [84]. These misconceptions lead to the idea that deaf persons can easily read and understand the written form of spoken languages [43] [84]. This is not the case, as was discovered by Holt [42]. Holt found that on average, hearing students at age 15 reached a reading level grade equivalent of 10, whereas Deaf and hard of hearing students at age 17 reached a reading level grade equivalent of 4.5 [42].

There are numerous sign languages throughout the world, each with its own vocabulary. These include American Sign Language (ASL) in Northern America, British Sign Language (BSL) in Great Britain, Japanese Sign Language (JSL) in Japan and SASL in

South Africa [37]. Sign languages are communicated in the visual-gestural modality, by the use of manual and non-manual gestures, having a grammar completely different from spoken languages. Manual gestures include the use of hand shapes, hand movements, hand locations and orientations of the palm in space. Non-manual gestures include the use of facial expressions, eye-gazes, head movements and upper body movements. Both manual and non-manual gestures must be performed for sign languages to be correctly understood and interpreted [27]. Apart from the different sign languages, SASL in South Africa possesses a high degree of lexical diversity which means that it varies across regions. Despite this lexical diversity, SASL has the same underlying grammar over all regions [27].

1.2 Motivation

To facilitate the communication between the Deaf and hearing persons, highly skilled interpreters have traditionally been used [27]. These interpreters tend to be very costly and it is a great effort to become a good interpreter that can translate between a spoken language and a sign language correctly and efficiently [27]. The use of an interpreter is not always appropriate and they need to be notified in advance when their services are to be required [27]. Another important fact to consider is that there will simply never be enough good trained interpreters that can assist the millions of deaf people [27] [37] [43].

An MT system that can translate between a spoken language, such as English, and a sign language, such as SASL, will solve the above problem of insufficient interpreters in South Africa. To visualise a sign language, an MT system must employ three dimensional (3D) computer generated virtual humans (VHs). Such a system can be used in many different application domains, such as Deaf telephony as well as English and sign language education. This will make it easier for the Deaf to access the various public and information services [24] [39] [43] [83] [95] [102] [107] [108].

The modelling and animation of VHs is very challenging and it requires a significant amount of time and money to develop a VH system that delivers adequate results. A workshop on "Accelerating Progress in Perceptive Animated Interfaces and Virtual Humans" organised by Ron Cole and his colleagues during April in 2004, highlighted both technical and social challenges with regard to progressing VH research [23]. It was noted in the workshop that "multimillion dollar systems did not result in community resources" and that "the expertise and infrastructure required to develop effective and scalable virtual human systems resides in just a few laboratories" [23].

1.3 Research Problem

To effectively visualise sign languages, a VH system must have models of adequate quality and be able to perform both manual and non-manual gesture animations in real-time. Ron Cole also notes the following in their workshop [23]:

> *The development of interfaces that incorporate virtual humans requires collaboration among researchers in many areas – psychologists, linguists, speech scientists, engineers and computer scientists with multidisciplinary expertise in human communication, interface design, speech and language technologies, dialogue modeling and management, computer vision and computer animation. While individual researchers, research labs and existing research communities represent knowledge and skills in each of these areas, no research community exists today that strives to focus the necessary multidisciplinary resources on research and development of perceptive animated interfaces incorporating virtual humans.*

Considering what was noted in the workshop by Cole [23] and the challenge to model and animate virtual humans (VHs), we formulate our research question as follows: *How do we model and animate VHs of adequate quality to effectively visualise sign languages?* We hypothesise that the research question, with some challenges highlighted by Cole [23], can be overcome by developing a methodology and open framework that employs various standards and open technologies.

1.4 Research Goals

Thus our goal was to develop a methodology with an open framework by using various standards and open technologies to model and animate VHs of adequate quality to effectively visualise sign language. As stated above, this open framework is to be used in a MT system that translates from a verbal language such as English to SASL. The standards and technologies we used include MPEG-4 [1], H-Anim [3], MakeHuman [57], Blender [16], Python [70] and SignWriting Mark-up Language [87]. We found it necessary to adapt and extend H-Anim to effectively visualise sign language. The adaptation and extensions we made to H-Anim are in fact also the purpose behind an open framework which is to aid in "Accelerating Progress in Perceptive Animated Interfaces and Virtual Humans" [23]. We wish to make it clear to the reader that this research does not constitute linguistic research or the correct grammatical visualisation of SASL.

1.5 Thesis Outline

The following is a brief outline of this thesis.

Chapter 2 In this chapter we review literature and discuss concepts and techniques related to 3D VH modelling and animation and we highlight their respective advantages and disadvantages. We also discuss MPEG-4 Facial Definition Parameters (FDPs) facial feature points and H-Anim which are standards for modelling and animating VHs. Our use, adaptation and extension of H-Anim is also motivated. Later in the chapter we review related work and discuss their methodologies and the models they developed. We organised related work based on body, hand and facial modelling and animation.

Chapter 3 In this chapter a focused literature review on sign language visualisation is presented. We first discuss the use of sign language transcription systems. Our use of SignWriting Mark-up Language and its more compact form Sign Bank Mark-up Language in our open framework is motivated. Later we discuss sign language visualisation systems that use video and VHs with their advantages and disadvantages.

Chapter 4 In this chapter we discuss our developed methodology and the technologies we used to establish an open framework to model and animate VHs and to effectively visualise sign language. We discuss the adaptation and extension of H-Anim to develop a generic skeleton by imposing joint rotational limits, developing flexible hands and addition of facial bones based on MPEG-4 FDP facial feature points for facial animation.

Chapter 5 This chapter details all the experiments with results and discussions to evaluate our methodology and open framework. We begin the chapter by discussing the design of our experiments after which we perform experiments on body, hand and facial animation.

Chapter 6 This is the final chapter of this thesis in which we provide concluding remarks and the main contributions of our research. Advantages and disadvantages to our methodology and framework is provided as well as some recommendations for future work.

1.6 Summary

This chapter gave some background on the Deaf, sign languages and the SASL project. We provided motivation for the SASL project and how it is to address the poor socio-economic opportunities and poor access to public and information services by the Deaf. This is to be done with an MT system that employs VHs and can translate between English and SASL. The modelling and animation of VHs however, is very challenging and it requires a significant amount of time and money to develop a VH system that delivers adequate results. The goal of this thesis was to develop a methodology and establish an open framework by using various standards and open technologies to model and animate VHs of adequate quality to effectively visualise South African Sign Language.

Chapter 2

Virtual Human Modelling and Animation

The previous chapter highlighted the research problem and goals of this thesis. In this chapter we first discuss concepts and techniques related to 3D virtual human (VH) modelling and animation with their respective advantages and disadvantages. These include: model representation; model creation and acquisition; and model parameterisation and deformation. We then discuss VH modelling and animation standards such as MPEG-4 Facial Definition Parameters (FDPs) and H-Anim. Our use, adaptation and extension of the H-Anim standard is then motivated. A survey of related work follows that is divided into face, body and hand modelling and animation. We discuss the methodologies that related work adopted, models they developed and present results they obtained. We then end the chapter with a summary of our discussions.

2.1 Model Representation

There are two model representation schemes to represent and visualise VHs, or any object model, in 3D computer graphics [41]. The first is the boundary representation scheme in which models are represented by surfaces which include: polygons; subdivision surfaces; and curves and surface patches. The second is the space-partitioning or volume representation scheme in which 3D space is partitioned in a set of finite primitive volumes that include: metaballs; volume elements (voxels); and constructive solid geometry (CSG) [41]. The sections that follow discuss the above mentioned model representation schemes with their advantages and disadvantages to advocate our use of a polygon representation for VHs.

2.1.1 Boundary Representations

2.1.1.1 Polygons

The use of polygons far outweigh the use of other model representation techniques in 3D graphics and are used in many real-time applications such as games, simulations and virtual reality [11] [41]. Polygons, along with points[1] and lines,[2] are considered to be geometric primitives in 3D graphics as can be seen in Figure 2.1.

A polygon is defined as a set of lines that is fully connected to form a closed loop,[3] where a set of connected polygons are called a polygon mesh [105]. Polygons can be named by the number of edges they have, of which triangles (3 edges) and quadrilaterals (4 edges) are the most commonly used for polygon meshes and to model complex 3D objects [105]. Apart from being named, polygons can be classified as regular, irregular, convex, concave or crossed. These classifications can be valid or invalid, and determines whether polygons will be correctly displayed by a graphics library, for example such as OpenGL [105], which do not draw concave and crossed polygons properly.

A major disadvantage of polygons is that it takes a large number of polygons to approximate a curved surface and thus computationally expensive to model and display organic looking objects. This disadvantage is addressed by hardware acceleration to process large numbers of polygons (polygon faces) and subdivision surfaces.

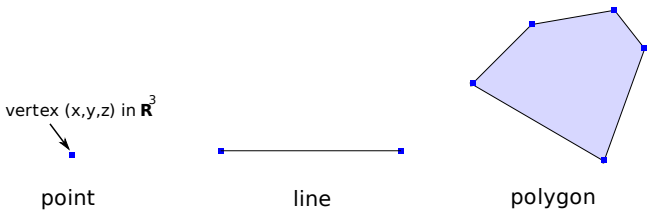

Figure 2.1: Geometric primitives: point, line and polygon.

2.1.1.2 Subdivision Surfaces

Subdivision surfaces are polygonal surfaces that are recursively refined to produce a smooth surface by using a subdivision scheme. Subdivision schemes work by subdividing

[1] A point, also known as a vertex, has no dimension and is represented by a set of co-ordinates in 2-space ($\mathbf{R}^2$) or 3-space ($\mathbf{R}^3$).

[2] A line is represented by a segment that connects two points.

[3] Lines of a polygon are known as edges and the area or closed loop known as a polygon face.

the original polygon mesh edges and adding new vertices at locations to approximate a curved surface. The most well known subdivision schemes include those of Catmull-Clark, Doo-Sabin and Loop [21] [29] [55].

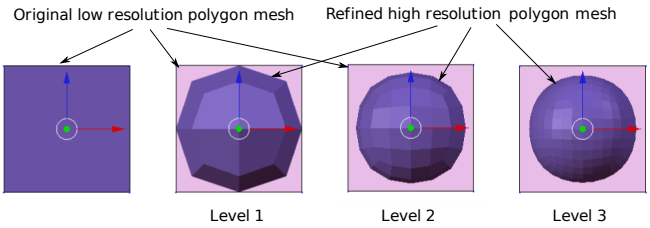

Figure 2.2: A cube at 3 different levels of subdivision using Catmull-Clark subdivison.

Sudivision surfaces are usually represented by a high resolution mesh with the original low resolution mesh used to control the high resolution polygon mesh (see Figure 2.2). In Figure 2.2 a cube is subdivided up to level 3, using the Catmull-Clark subdivision scheme. Advantages of using subdivision surfaces include simplification of the modelling process to create smooth surfaces and one can easily refine the entire surface globally or at local regions. Since subdivision surfaces are merely refined polygonal surfaces, as a surface is subdivided or refined, there is a reduction in display times [82].

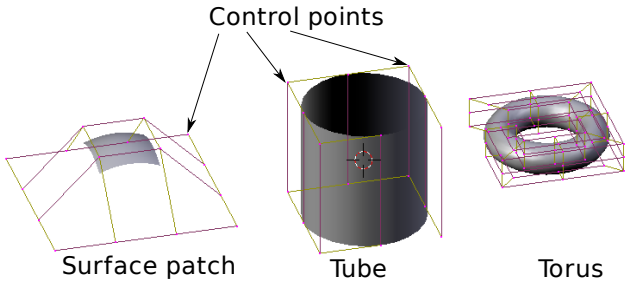

Figure 2.3: Examples of nonuniform rational B-splines (NURBs) surfaces.

2.1.1.3 Curves and Surface Patches

A significant amount of research has been performed to develop mathematically precise curved lines and surfaces [41]. Curved surfaces, also known as surface patches, are defined by two sets of orthogonal parametric functions of two variables [41]. These parametric functions have a set of control points that are also used to manipulate the surfaces they define. Parametric functions used to create curved surfaces include spline functions such

as Bézier splines, B-splines, beta-splines (β-splines) and rational splines which include nonuniform rational B-splines (NURBs) [41]. When using curved surfaces, a designer would normally create surface patches and align the borders of those patches to build a 3D model such as a face [14]. Examples of simple NURBs surfaces, which allows for exact representations of circles and ellipses to model complex shapes, can be seen in Figure 2.3.

Obvious advantages of using surface patches are that they are mathematically well defined and are able to represent smooth organic looking surfaces that are easy to manipulate through control points [41]. Disadvantages of surface patches include: slow performance, as they are complex mathematical representations; the lack of hardware acceleration to address their complexity; it is difficult to perform local refinement of a single surface patch to model very fine details; and undesired surface wrinkling and lines or cracks where surfaces align during animation [14].

2.1.2 Volume Representations

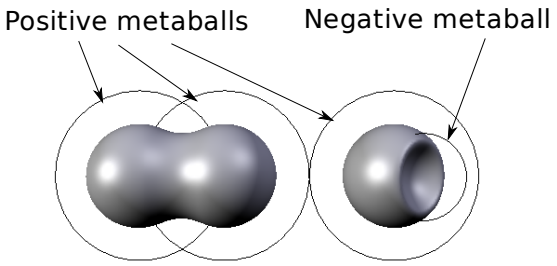

Figure 2.4: The two positive metaballs on the left seem to bond as molecules and their surfaces are smoothly blended together. A negative and invisible metaball on the right creates a dent in a positive metaball.

2.1.2.1 Metaballs

Metaballs, also known as "implicit" surfaces, are attributed to J. F. Blinn [18]. This model representation was developed by Blinn to display molecular models and to automatically simulate how molecules interact with each other [18]. Metaballs are defined by density functions in space, that can affect each other either positively or negatively [18]. A polygon surface for the density function is displayed, where the function equals a selected threshold. This threshold, along with the height and standard deviation of density functions, affects the polygon surfaces when combining metaballs. The height and standard deviation are

used to affect the stiffness or "blobbiness" of a metaball and whether it makes a positive contribution, such as a bump, or negative contribution, such as a dent [18]. Examples of metaballs can be seen in Figure 2.4.

Advantages of metaballs are: that they are mathematically well defined; they automatically affect each other based on their proximity to each other in space; and they appear soft, continuous and suitable to design VHs [88] [98]. Disadvantages of metaballs include slow display times and that they require tessellation into polygon meshes to enable texturing and improve visual realism [98].

2.1.2.2 Volume Elements

A model representation that is primarily used to visualise biomedical data sets is volume elements (voxels). In the use of voxels, space is partitioned into non-overlapping volumes, with each volume known as a voxel. This sampling of space is stored as an octree, where the depth of the octree relates to the resolution of the sampled space [41]. The deeper the octree, the better the resolution of the sampled space and quality of voxel models. Advantages of voxels include the use of simple data structures that aid in optimisation of data sets and a single value can be stored for voxels in a sub-region of space where the data is homogeneous [41]. A major disadvantage of voxels is slow display times even with the use of a modern graphics processing unit (GPU) [28].

2.1.2.3 Constructive Solid Geometry

Constructive solid geometry (CSG) is a model representation technique primarily used to model solid objects and have not found much application in VH modelling [11]. In CSG, primitive volumes, such as cubes, spheres and cones which can overlap and occupy any region of space unlike voxels, are used. Set operations such as union, intersection and difference are applied to these primitive volumes to build more complex objects. Complex objects are then represented as a binary tree of operations on objects [41]. Extensions to CSG such as volumetric-CSG have also been developed to visualise medical data sets [32]. Advantages of CSG include ease of computation of properties such as mass and volume and it is used for procedural modelling as discussed in Section 2.2.3 [41]. Disadvantages of CSG include its inability to represent curves exactly where objects intersect and slow display times [11] [40]. These disadvantages have been overcome by compact representations of curves where models intersect and CSG triangulation algorithms developed on a modern day GPU to display CSG models in real-time [40].

2.2 Model Creation and Acquisition

Techniques to create and acquire VH models or parts of VHs are discussed in the sections that follow. The discussion begins with manual labour intensive techniques, such as, interactive modelling and parametric modelling and later proceeds to nearly fully automated techniques, such as procedural modelling, photogrammetry and 3D scanning. These techniques are discussed with their advantages and disadvantages to advocate our use of MakeHuman [57], a parametric VH modelling tool that we discuss in Section 4.1.1.

2.2.1 Interactive Modelling

Interactive modelling is a manual labour intensive technique in which a designer uses a generic interactive modelling package such as Maya [8], Blender [16] or Truespace [20], to name but a few. In this thesis, Blender is employed to perform some interactive modelling. This is discussed in Section 4.1.2. During interactive modelling, any primitives, such as points, lines, polygons, curves, NURBs or metaballs, are used to model 3D objects [88]. These primitives can be duplicated, translated and some extruded to created higher order primitives, such as cubes, spheres, cones, arms, legs, heads or any other objects. These higher order primitives can then be used to build complete VH models. Designers have come up with clever techniques, such as using background images or profiles of objects they wish to model and use these as blueprints. The advantage of interactive modelling is that one has absolute control over every single point of a model in the modelling process. Generic interactive modelling tools have matured over the years and include features such as mesh mirroring, 3D sculpting and path based extrusion among many others to simplify modelling [8] [16] [20]. The disadvantages of interactive modelling are that it requires a fair amount of time to learn how to use a generic modelling package and artistic skills to create models of adequate quality.

2.2.2 Parametric Modelling

Parametric modelling of VHs can be regarded as a higher level specific modelling technique compared to interactive modelling. A template model of a VH can be developed with a generic interactive modelling package or acquired through 3D scanning (see Section 2.2.5). The template model can be divided into parts and parameterised with attributes such as width, length, size and others that range from a minimum to maximum value [77]. A database of minimum and maximum “morph” targets (see Section 2.3.2) can be modelled

using the template model which are then tied to these attributes [15]. Another approach is to use a database of scanned models with different proportions, which are used to build a target vector space that is tied to these attributes [77]. Upon manipulating these attributes, parts of the template model are transformed to the modelled "morph" targets or targets in a vector space [15] [77].

Two of the most popular parametric modelling tools to model VHs is the open source tool MakeHuman [57] and the proprietary tool Poser [80]. The advantages of parametric modelling are that: it is much simpler; less time consuming; and requires no artistic skills for end users as opposed to modelling with a generic interactive modelling package. Disadvantages of parametric modelling are that it takes years to develop a high quality template model with its targets database and one is bound by the number of attributes and targets that can be used which in turn limits freedom of expression.

2.2.3 Procedural Modelling

Procedural modelling is a semi-automatic modelling process in which a programmer writes procedures with a modelling language to model objects [62]. This type of modelling is usually applied to texture generation or GPU shader programs but was found to be of some use for VH modelling, although with unsatisfactory results [62]. Results are usually dependent on how well a model is paramaterised, the primitives used and the operators or functions available in a procedural modelling language.

As mentioned before, CSG is used in procedural modelling because complex objects can be represented as a tree of operations applied to primitive volumes. Many modern generic interactive modelling packages provide interfaces to programming languages such as Python, VBscript or Jscript as modelling languages [16] [20]. Some packages have their own embedded scripting languages, such as Maya with its Maya Embedded Language (MEL) [8].

Advantages that procedural modelling has over other techniques are: well defined parameterised and structured models as scripts; well developed scripts can be easily reused; one only needs to understand a script's input parameters; and resultant models can be dynamically regenerated with different characteristics dependent on a script's input parameters [62]. Disadvantages of procedural modelling is that one does not have instant visual feedback as with interactive or parametric modelling tools, which results in much trial and error to obtain desired results. Also, programming in general is difficult and operators or functions used in procedural modelling are of a low level and requires understanding of programming concepts during script development.

2.2.4 Photogrammetry

Photogrammetry is the process of estimating camera parameters and making measurements from images to model objects [68]. Multiple cameras, with different views of the object to be modelled, are set up and calibrated to capture images simultaneously [68]. Once images have been captured, camera parameters and target feature points, can be manually estimated or automatically extracted from captured images [68]. A model fitting process can then be used with these feature points to deform a generic model with corresponding source feature points [68] [69] [71] [97]. Some researchers have used only a single camera image but use additional means such as a ruler to aid in estimating camera parameters and locating feature points [6].

The best advantage of photogrammetry is that it enables one to model person specific models [6] [68] [69]. The more feature points used, the more accurate the fitting process and the generic model resembles a real person. Captured images can also be used as textures for added realism [68] [71]. Photogrammetry is considered a better alternative to 3D scanning since one can use inexpensive cameras [68] [69] [71]. The downside of photogrammetry is its inability to completely model certain features such as the eyes, teeth, tongue and ears. It is also not possible to model accurate human bodies in cases where there is occlusion of certain parts of the body [69].

2.2.5 3D Scanning

The use of a 3D laser scanner, to perform 3D scanning, is the most advanced method to automatically and accurately acquire real world surface data of objects. A 3D laser scanner collects thousands of data points of an object at a specified sampling rate and grid size by projecting a laser beam onto an object's surface. 3D Scanning technology was previously slow but it is now possible to scan a complete human body surface in a matter of seconds [25]. Information such as colour and weight can also be obtained while scanning [25]. The use of 3D scanning was applied to build an anthropometric database such as the Civilian American and European Surface Anthropometry Resource Project (CAESAR) [73]. A large database such as CAESAR along with feature points would be difficult if not impossible to create using other modelling techniques [73].

Disadvantages of 3D scanning include: expensive equipment; scanned data is complex and requires a significant amount of post-processing; it is not possible to create separate models for the hair, eyes teeth and tongue as with photogrammetry; and occlusion can also be a problem as with photogrammetry [6] [10] [53] [97] [99].

2.3 Model Parameterisation and Deformation

A VH model that was created or acquired in some way or another with the techniques discussed above is not of much use if it cannot be animated. To enable a model to be animated, it must first be parameterised with a deformation technique. There are several deformation techniques that enable one to apply a geometric transformation such as a translation, rotation or scaling to a model's data. These deformation techniques range from computationally inexpensive techniques which yield results of less quality or realism to computationally expensive techniques that have realistic results. Also, some techniques are more intuitive or simpler to use than others. In the sections that follow, we present some of the more popular model parameterisation and deformation techniques. Keeping our goal in mind, to model and animate VHs for real-time sign language visualisation, we omit the two most advanced and computationally expensive deformation techniques. These are muscle and physical based parameterisation and deformation. The reader is referred to Albrecht [6] and Ng-Thow-Hing [64] on these computationally expensive techniques.

2.3.1 Direct Parameterisation

Direct parameterisation is the first technique to be used to parameterise models for animation [66]. In this technique a model is parameterised by selecting certain points used to define the model either directly or by a density function on the surface that influence a region of points [66] [67] [103]. The model is then directly deformed or animated by applying transformations to the parameterised points. The advantage of direct parameterisation is that it is straightforward to use for facial animation and is computationally efficient [66] [67] [103]. The disadvantage of direct parameterisation is its difficulty to be applied to articulated figure models; it is time consuming to apply and is usually a trial and error process to obtain pleasing results [66] [103].

2.3.2 Morphing

Instead of selecting certain points directly or by applying a density function to parameterise parts of a model, a model can be completely parameterised as a whole and stored as a source "morph" or parameterisation [52] [68]. The same model can be taken and transformed to take on another form or appearance that can also be stored separately as a target "morph" or parameterisation. Morphing is then the process of transforming the

source parameterisation into the target parameterisation through interpolation [52] [68]. Each point from the source "morph" is interpolated to its associated point in the target "morph". Morphing has been applied to both facial and body animation and has proven to yield extraordinary results during animation [52] [68].

Advantages of morphing include: direct control over each and every point; by taking the model as whole, it is possible to model certain details, such as wrinkles for facial animation, which is difficult in direct parameterisation; one can perform optimisations, such as storing only indexes of transformed data points and their relative transformations as a target "morph"; and computational efficiency is directly proportional to the complexity of a model [97] [68] [52]. Disadvantages of morphing include the difficulty to create "morph" targets and a significant amount of time since direct user interaction is required; morphing targets have space requirements which are dependent on the complexity of the model in use; and source and target models must have the same topology. Lee et al. [52] have shown that one can adapt models with different topologies from different sources to a generic model, which is then used for morphing.

2.3.3 Free Form Deformation

Free form deformation (FFD) is a solid model deformation technique invented by Thomas Sederberg and Scott Parry [75]. This deformation technique can be regarded as a volume represented by a parallelepiped grid of control points (lattice) with the model that is to be deformed embedded within this volume [75]. The embedded model points are then parameterised by the control points on the lattice by a triple tensor product Bernstein polynomial [75]. As control points on the lattice are transformed by a geometric operation, so too are points of the embedded model. FFDs have been extended by some researchers to allow for arbitrary shapes and an arbitrary number of control points. Some of these extensions include: rational free form deformation (RFFD) that has been used for facial deformation [45]; Dirichlet free form deformation (DFFD) for hand deformation [44]; and surface oriented free form deformation (SFFD) [79] for body and hand deformation. These extensions of FFDs have proven to be both computationally efficient and produce good visual results, comparable to that of skeletal subspace deformation (SSD), which is discussed in the next section [79].

2.3.4 Skeletal Subspace Deformation

Skeletal subspace deformation (SSD) is currently the most popular technique to parameterise and deform articulated figures in real-time and is used especially in sign language visualisation systems [12] [24] [36] [51] [95] [106] [108]. As the name implies, SSD enables one to build an underlying skeleton with joints for an articulated figure, such as that of a VH. The skeleton is built as a tree of bones where a root bone contains the global co-ordinate frame for the entire skeleton (skeleton space). Each bone in the tree has its own local co-ordinate frame (bone space) which is affected by that of their parent and ancestors. A bone can be scaled and have degrees of freedom (DOFs) to perform rotations and translations.

Parameterisation of the data points of the skin model (surface geometry) of a VH is attached to the skeleton by assigning a weight to each point as to how much a bone should affect a point when the bone undergoes a geometric transformation. This parameterisation process is popularly known as "skinning" or "rigging" and can be performed through an interactive manual process known as vertex weight painting [26] or with an automatic skinning algorithm [13]. Initial implementations of SSD only allowed a single weight to be assigned to a vertex which resulted in very unrealistic deformation and displeasing results. This was overcome with linear blend skinning (LBS) which assigns multiple weights to points such that the weights sum to 1. These weight values are linearly blended depending on a point's distance from a bone [44] [13].

Baran and Popović implemented an automatic skinning algorithm which made some improvements to LBS. Their algorithm assigns bone weights to points based on a heat equilibrium over the surface geometry of an articulated figure [13]. In this thesis we employ the automatic skinning algorithm developed by Baran and Popović which was implemented in Blender [13] [16]. The algorithm by Baran and Popović was developed with three requirements in mind [13]:

- Generality: A skeleton can be used on different VH figures.
- Quality: Animation of skinned characters must be of high quality.
- Performance: The algorithm must be fast and perform skinning in under a minute.

These requirements are satisfied by the fact that their algorithm takes an articulated figure's surface geometry into account and also performs automatic skeleton fitting and simplification before assigning weights. The current implementation in Blender performs no skeleton fitting and simplification but does consider the surface geometry. In Chapter

4 we show how we take advantage of the current implementation. The reader is referred to [13] for an in depth discussion of the algorithm by Baran and Popović.

Advantages of SSD are that: it is simple to implement and exists in many interactive modelling packages; it can be used as a general deformation technique especially when using vertex weight painting to parameterise a VH model; it is simple and intuitive to use since one only rotates bones to pose or deform an articulated figure; and most importantly, it is computationally efficient [8] [13] [16] [20] [54]. SSD however serves only as an approximation to a real skeleton and has limitations such as the "collapsing elbow" and "candy wrapper" effects when performing bone rotations [54]. These limitations have been addressed by some researchers by extending LBS. These extensions take into consideration the underlying shape of joints [47] and developing a pose space that determines the desired deformation for a pose when rotating bones [54].

2.4 Standardisation

The different possible ways to parameterise and deform VHs for animation lead to difficulties for different research groups to share, reuse and transfer resources between them. These difficulties were highlighted in a report by Susan Duncan, Javier Mollevan, Eric Petajan and Jianxia Xue in the workshop organised by Cole [23]. The report pointed out the need to use a standard encoding scheme to encode animation databases. There are two standards used to parameterise and deform VHs that can serve as animation data encoding schemes. These are MPEG-4 Face Body Animation Parameters (FBAP) [1] and H-Anim [3].

2.4.1 MPEG-4 Facial Definition Parameters

The MPEG-4 FBAP specification for face and body animation is a closed ISO/IEC standard and formally specified in ISO/IEC 14496-2:2004 Information technology – Coding of audio-visual objects – Part 2: Visual [1]. The standard document defines a face body animation (FBA) object as a collection of nodes in a scene graph. The FBA object has a node to define a skeleton with a body definition parameter (BDP) set and a node to define a face with a facial definition parameter (FDP) set. The BDP set that defines a skeleton is controlled by a stream of body animation parameters (BAPs) to perform body animation. The FDP set defines facial feature points used to parameterise faces and is controlled by facial animation parameters (FAPs) to perform facial animations.

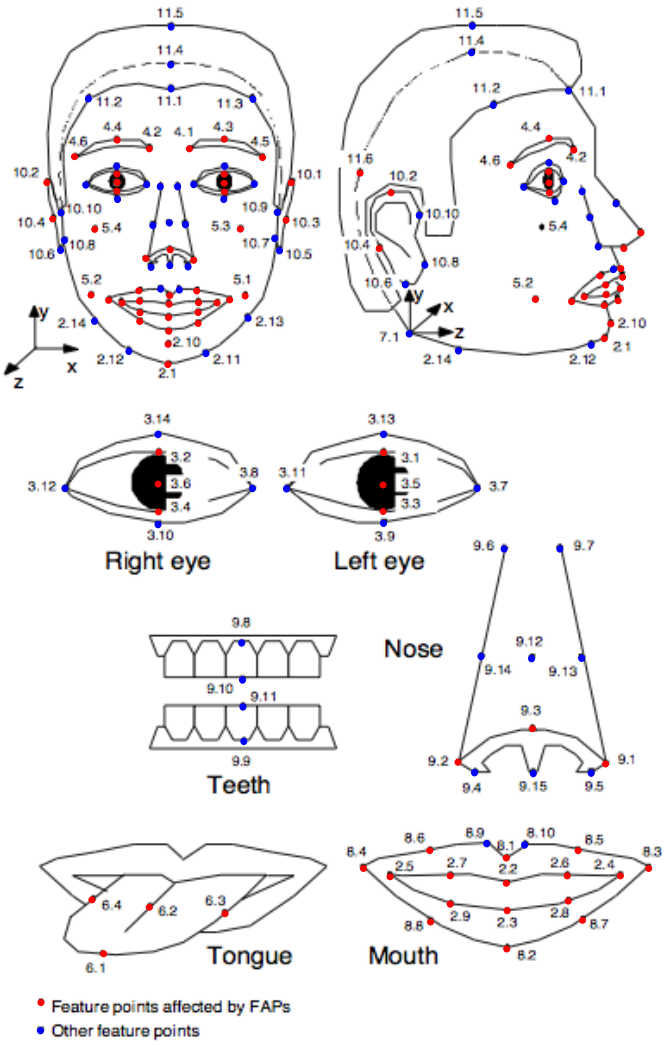

Figure 2.5: MPEG-4 Facial Definition Parameter set facial feature points [1].

In this thesis, we are interested in using the MPEG-4 FDP set facial feature points to extend the H-Anim standard which is based on the MPEG-4 BDP skeleton definition [3]. We therefore omit our discussion on the MPEG-4 BDP set and refer the reader to the specification document for more information on MPEG-4 BDP [1].

The MPEG-4 FDP set facial feature points can be seen in Figure 2.5 and defines two sets of feature points. The one set, red dots in Figure 2.5, is affected by FAPs and the other set, blue dots in Figure 2.5, defines standard face locations. FAPs represent displacements of the face that are based on "minimal facial actions and are closely related to muscle actions" [1]. Another popular encoding scheme that is also based on facial muscle actions is the facial action coding system FACS [30]. FACS, which we omit and

refer the reader to [30], has a very important disadvantage compared to MPEG-4 FAP. This disadvantage is the inability of FACS to encode detailed lip movements, whereas MPEG-4 FAP have detailed facial feature points to allow complex mouth shapes, such as visemes [1] [23] [30]. MPEG-4 also has disadvantages which are related to its direct parameterisation as discussed in Section 2.3.1.

2.4.2 H-Anim

H-Anim is an open standard and used for modelling Virtual Reality Markup Language (VRML) [2] and X3D [4] VHs to address the increasing need to represent humans in virtual environments [3]. The design goals of H-Anim are to ensure:

- Compatibility: Compliant browsers must implement the features of an H-Anim figure.
- Flexibility: The standard must make no assumptions on the types of applications it will be used for.
- Simplicity: The standard can be used in its simplest form and extended if the need arise.

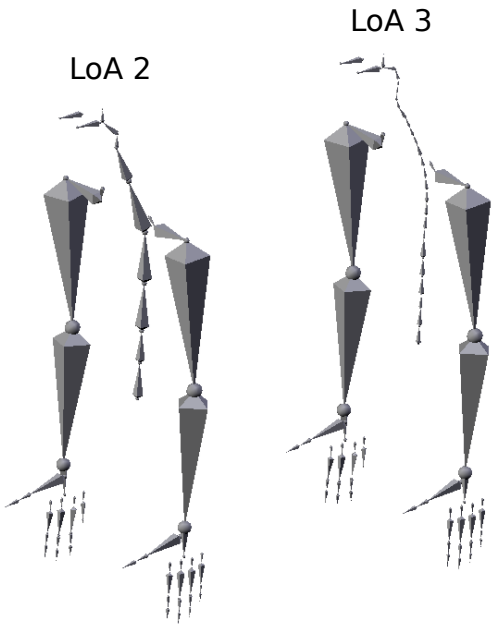

Figure 2.6: Upper body H-Anim LoA 2 and LoA 3 examples modelled from source data found in [3].

H-Anim specifies a VH skeleton in terms of hierarchal joint and segment nodes at four different levels of articulation (LoAs) that range from 0 to 3. These LoAs start with LoA 0 as the lowest that has only a single root joint and ends with LoA 3 that has a near realistic spine. In this thesis, we endeavour to adapt and extend the LoA 2, which include hand joints unlike LoA 1, and has a simpler spine than that of LoA 3. Figure 2.6 depicts H-Anim skeletons for LoA 2 and LoA 3 of the upper body that we modelled as bones in Blender from example data in the H-Anim [3] specification. As can be seen in Figure 2.6, the H-Anim skeletons are approximations to a real skeleton with a single bone for the skull and single bones to represent the carpals in the hands.

Although not required, the H-Anim specification suggests that body segments be built in place. The skin model of H-Anim VHs can be modelled as separate geometric segments, which closely follow the joint hierarchy that is computationally efficient but results in poor quality. It can also be modelled as a seamless skin geometry for improved results but requires more data and thus more processing [3] [9].

The H-Anim standard is very flexible but does have a few limitations that forces us to adapt and extend it. The standard does not specify joint centre locations or joint rotational limits. Elliot et al. [31] found that the hand bones in H-Anim did not provide enough flexibility for hand shapes found in sign languages. Another limitation is the fact that H-Anim only provides a simple set of bones for facial animation and it is suggested by the specification that the MPEG-4 FAP set be used for facial animation [3].

2.5 Virtual Human Models for Animation

In 3D computer graphics there is always the trade-off between visual quality or realism and display speed [11]. This trade-off is very apparent for VH modelling as it has advanced from modelling simple single layered models to complex multi layered models. Single layered models have only an outward skin surface model [68] [72]. Multi layered models strive towards realism and can include skin, muscles and a skeleton [6] [64]. Some researchers have gone to great lengths to model anatomically detailed VHs such as that of The Visible Human Project [93]. We are only interested in models of the upper body that have been developed for animation as our goal is to model and animate VHs of adequate quality with the necessary features to animate sign language gestures. In the sections that follow, we review methodologies employed and VH models developed for animation. Researchers aimed their efforts to create separate modules for facial animation, body animation and hand animation which dictates the structure of our review [44].

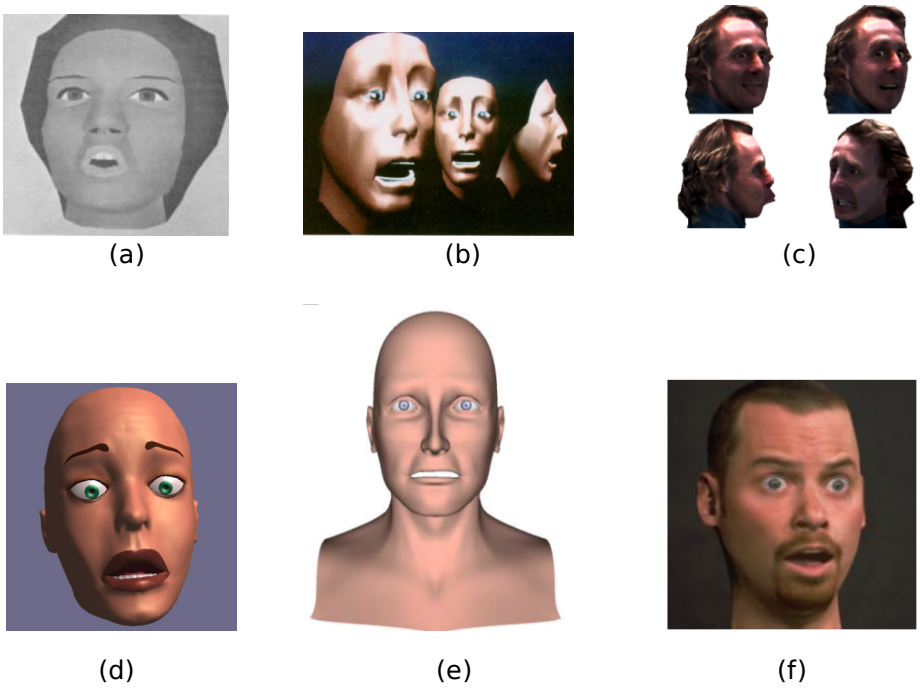

Figure 2.7: (a) Parke's facial model [66]. (b) Waters's facial model [100]. (c) Lee's facial model [53] . (d) Greta facial model by Pasquariello and Pelachaud [67]. (e) The facial model used by Barker [14]. (f) Pighin's facial model [68].

2.5.1 Face and Head Feature Models

Frederick Parke [66], one of the forerunners in facial modelling and animation, developed facial models as early as 1972. Parke used a polygonal representation for the skin and made a very important observation with regard to using polygonal models. He observed that a polygonal model of the face must be specially developed and optimised to allow for natural movement of the polygons to acquire natural looking results. The approach he took was to assume that the face is symmetric and painted a polygonal mask on one half of the face of an assistant. Photos of the assistant with different facial expressions was then captured from two views. Photogrammetry was then applied in the sense that measurements were taken from the photographs to develop a polygonal model with the best topology to represent a face. The resultant polygonal model had only 124 polygons and 202 vertices which was visually improved by applying a smooth shading algorithm. Parke further improved his model by interactively creating nostrils, models for the eyes, teeth and inside of the mouth [66]. To perform animation, Parke used morphing by storing the face topology and phases (morphs) of the face in files. The model developed by Parke can be seen in Figure 2.7 (a).

Waters [100] developed a polygonal facial model and a muscle parameterisation and deformation process to perform facial animation. His model was developed by also using a photogrammetry technique similar to that of Parke [66] [100]. To avoid unwanted artefacts, such as polygon intersection and "facet popping", Waters increased the polygon count at curving regions and triangulated his model. His initial model had no eyelids, eyes and teeth. These were modelled interactively by adding curves near the eye socket regions for eyelids, creating swept revolutions to model eyeballs and using sets of Bézier curves to model teeth [100]. It is not clear what mechanism was used to perform jaw rotation. The model developed by Waters can be seen in Figure 2.7 (b) with a surprised expression.

Lee et al. [53] proposed a methodology to model facial models for individuals by scanning their heads to acquire model and texture data. After scanning, their algorithm automatically fits a generic face model, which is parameterised with facial muscles, to the scanned data [53]. Animation of the face is then performed through a three layer physics based muscle process. Several improvements to facial animation is proposed by Lee et al. [53], such as algorithms to estimate the structure of the skull and to prevent the skin from penetrating the skull. They interactively developed models for the eyes as spheres, the eyelids as polygons and the teeth as NURBs. They also modelled the hair, neck and bust as polygonal models by extending the facial model to the boundaries of the scanned data. A very important feature, which is not discussed by Lee et al. [53], is the mechanism to perform jaw rotation to open the mouth. The model developed by Lee et al. can be seen in Figure 2.7 (c) with different facial expressions.

Pasquariello and Pelachaud developed Greta, a proprietary facial animation system that complies with the MPEG-4 specification [67]. Greta's facial model was manually modelled and is based on a polygonal surface that includes models for the eyes and teeth. During model development, particular attention was given to the mouth, sides of the mouth (nasolabial furrow), eye and forehead regions. Greta's model was parameterised with the MPEG-4 FDP facial feature point set by a function that assign weights of decreasing influence from a feature point. Locations that are affected by FAPs was subdivided to improve results during animation. Algorithms to produce auxiliary deformations such as bulges and wrinkles when manipulating FAPs were also implemented. One thing that is not clear is the mechanism they use to open the mouth. The final Greta model has a total of 15000 triangles and can be seen in Figure 2.7 (c) with a fearful expression.

Barker acquired a facial model that was modelled with NURBs surface patches [14]. His main goal was to develop a muscle parameterisation and deformation process for facial

animation based on that by Waters [100]. Animation in Barker's system is limited to the face area and there are no models for the teeth and tongue [14]. Also, Barker's system can not perform jaw and head rotations. Barker's facial system has many of the advantages and disadvantage related to NURBs as discussed in Section 2.1.1.3. A result of Barker's system can be seen in Figure 2.7 (e) with a fearful expression.

Pighin et al. modelled human faces by using a photogrammetry technique from a head model developed with an interactive modelling package [68]. They captured multiple images of a face from different views that were manually marked with an initial set of 13 feature points. These marked images were then used in an initial model fitting process by scattered data interpolation to deform a generic head model into an estimate of the captured head [68]. After initial fitting, they perform shape refinement by using an additional 99 feature points to deform the head model to closely match a captured face. Their deformed model is then textured with the captured images which are blended together on the surface of their model for added realism. Pighin's system has no separate models for the eyes, teeth, ears and hair. These features are individually textured in a separate process onto their model. To perform animation, they modelled and textured head models of a person with different facial expressions which are then used in a morphing process. The model data along with facial textures are morphed, which yielded highly realistic results as can be seen in Figure 2.7 (f).

Figure 2.8: (a) Badler's Jack figure model [66]. (b) The model developed by Kalra et al. [44]. (c) The optimised model by Seo et al. [76].

2.5.2 Articulated Figures

2.5.2.1 Bodies

In 1993 Badler et al. published a book on simulating humans [11]. The book is an accumulation of research they performed at the University of Pennsylvania to develop their VH system called Jack. Their primary goal was to model and animate VHs to be used by engineers whom design products for humans with improved ergonomics [11]. They developed what seemed to be a simple polygonal model that can interact with objects in a virtual environment. Despite the simple appearance, much time was invested into modelling VHs of realistic proportions and their skeletons with joint rotational limits by using an anthropometry database [11]. Their skin model had a total of 183 polygons which was attached to a skeleton with 69 segments (bones), 68 joints and 136 DOFs. Badler et al. also paid special attention to modelling the shoulder as it is one of the most complex parts to model and animate [11]. Jack became a proprietary system which now makes use of several anthropometric databases, including that of CAESAR [78]. The original Jack model can be seen in Figure 2.8 (a).

In 1998, Kalra et al. developed the only system with modules that could perform animation of the body, hands and face [44]. We limit our discussion to the model of their body without their face and hand models, since these are modelled as separate entities. The body is also discussed more in depth than the face or hands. An interactive modelling package named BodyBuilder was used to model the body on 3 layers for male and female figures [44]. The first layer of the body includes a skeleton with 32 joints and 74 DOFs along with a 6 DOF joint to position the skeleton. The second layer, which defines the volume of the body such as muscle and tissue, is modelled with metaballs and ellipsoids that are attached to the skeleton. The third layer, which represents the skin, is modelled using spline surfaces by performing ray surface intersection tests on the second layer. Since their body was modelled separately from the face and hands, they had difficulty to connect all the models together [44]. Kalra et al. found it necessary to convert their model to a triangle polygon mesh to take advantage of hardware acceleration and simplify the integration between the head, hands and body. A male figure developed by Kalra et al. with 14000 vertices and 13500 textured triangles is shown in Figure 2.8 (b).

Seo et al. [76] used a generic model with skin and skeleton from one of their previous projects to develop a methodology to quickly model VHs that are ready for animation. They employ the H-Anim LoA 2 to define the skeleton, which is attached to the skin model with LBS. Their choice for H-Anim LoA 2 is to allow them to use MPEG-4 BDPs

during animation. Seo et al. further implemented volume deformation of: the breasts by using a NURBs curve and directly controlling the curve control points; the belly (stomach) by using a Bézier curve and directly controlling the curve control points; and the bottom (buttocks) by using FFD [76]. To optimise their model for animation, Seo et al. employed an intelligent mesh decimation technique to reduce the number of vertices and polygons (faces) in their model. The mesh decimation leaves more vertices and faces at regions of high curvature such as the joints [76]. In Figure 2.8 (c) the optimised model with 4726 vertices and 8578 faces used for animation by MPEG-4 BDPs is shown.

Moccozet et al. [61] obtained scanned body data which was then pre-processed by triangulation and hole filling. After pre-processing a scanned model, H-Anim body feature points was used to automatically establish a correspondence with a generic model and its underlying skeleton [61]. Once a correspondence was established, the generic model was automatically fitted within the pre-processed model. The fitting starts with a coarse fitting process and is then refined with a fine fitting process, which was developed by Seo and Thalmann [77]. It should be noted that the skeleton of the generic model was of a low LoA that helped the fitting process [77]. An overview of the methodology developed by Moccozet et al. is depicted in Figure 2.9.

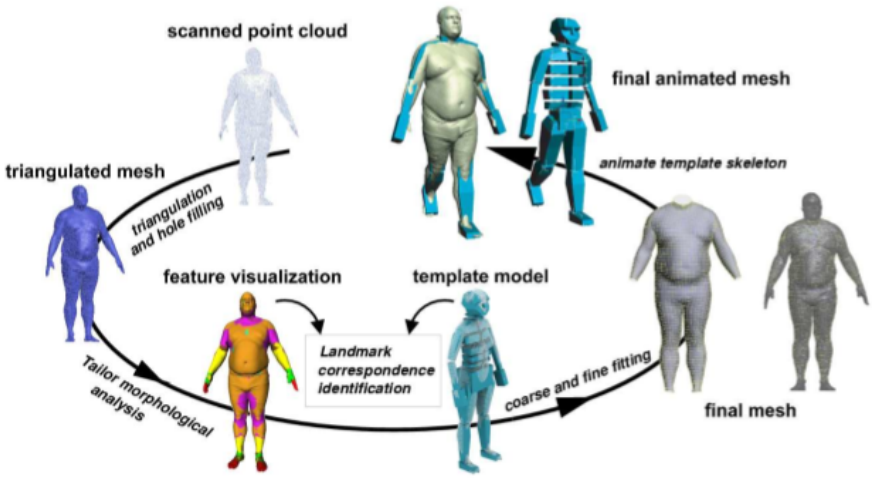

Figure 2.9: Overiew of the methodology developed by Moccozet et al. [61].

Wang and Ressler [99] developed a toolkit to convert CAESAR body models into seamless H-Anim compliant VHs. Only body models in the standing posture were used which was first pre-processed. The CAESAR bodies were simplified from 200000 vertices and 300000 polygon faces to have only 50000 polygon faces before being hole filled. Landmarks on the CAESAR bodies were then used to estimate joint centre locations for an

H-Anim skeleton. Wang and Ressler found that the CAESAR models were still too complex after pre-processing [99]. They therefore found it necessary to use a semi-automatic process to segment the bodies into parts. After body segmentation, a skinning algorithm similar to LBS was applied [99]. An overview of their methodology is depicted in Figure 2.10 (a) with an example of a skinned CEASAR body in Figure 2.10 (b).

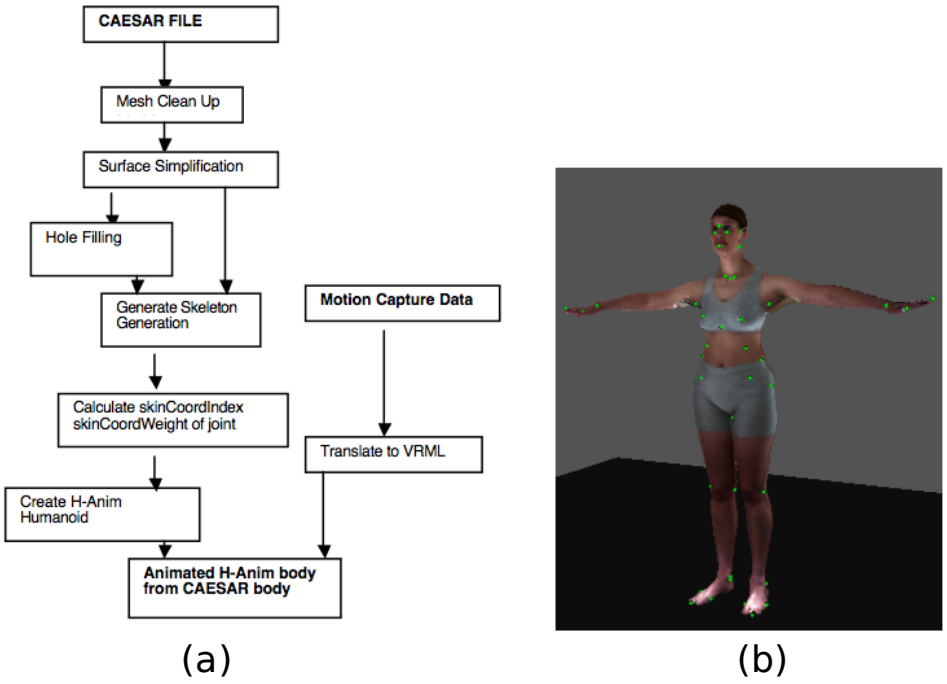

Figure 2.10: (a) Overiew of the methodology developed by Wang and Ressler. (b) Example of a skinned CEASAR body [99].

2.5.2.2 Hands

Most of the earlier work on hand modelling and animation focused primarily on interacting with objects in virtual environments. Thalman et al. designed a simple skeleton of the hand and employed the concept of Joint-dependent Local Deformation (JDL) operators to map surfaces onto a skeleton [56]. They also developed algorithms for collision detection, algorithms to simulate joint rounding and muscle inflammation. Although computationally expensive, visual quality of animations while grasping objects was acceptable. Thalman et al. was amongst the first to separate the geometric surfaces from the underlying skeleton thus resulting in two layers. This enabled them to employ a myriad of surface modelling techniques to further enhance visual quality [56].

Wan et al. also developed a virtual hand for object grasping [98]. They developed a

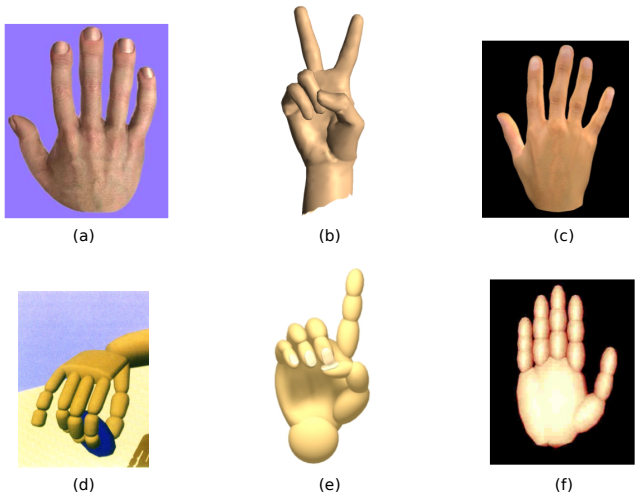

Figure 2.11: Hand models developed by different researchers: (a) Wan et al. [98]. (b) Albrecht et al. [6]. (c) Rhee et al. [71]. (d) Rijpkema [72]. (e) McDonald et. al [60]. (f) Van Zijl and Raitt [96].

3 layer model of the hand that includes skin, muscles and a skeleton. The geometry of the skin layer was modelled by using metaballs [98]. Metaballs are excellent for modelling organic surfaces but computationally expensive as discussed in Section 2.1.2.1. They therefore found it necessary to convert the metaball representation to a polygonal surface to realise an interactive application. Upon conversion to a polygonal surface, they found it necessary to apply texture mapping to improve the visual result of their hand model. Their muscle layer is based on Dirichlet Free-Form Deformation and used to deform the skin. The skeleton layer, instead of the muscle layer, serves as the primary animation control interface. The hand model developed by Wan et al. can be see in Figure 2.11 (a).

Albrecht et al. [6] developed a physics-based anatomical model of the hand from scanned data. Their model is also based on 3 layers with skin, muscle and a skeleton also like that of Wan et al. [98]. All layers have a geometric model to improve realism with the muscle model including pseudo muscles for deformation. The pseudo muscles are used to rotate bones by specifying muscle contraction values. Although Albrecht et al. achieve real time frame rates with high end hardware, their approach has two drawbacks. These are that their approach is computationally expensive and also it is not straightforward to specify muscle contraction values to achieve desired movements [6]. The hand model developed by Albrecht et al. can be seen in Figure 2.11 (b).

Rhee et al. [71] developed a technique to model human hands from surface anatomy. Their approach uses photogrammetry to automatically construct “person-specific” hand models from a single hand image presenting the palmar surface. Upon capturing a hand

image, a predefined generic hand model is deformed by employing scattered data interpolation and radial basis functions. Crease information of the palm and fingers is extracted and used to estimate joint centre locations. After modelling and skinning, where curve segment matching is also performed, the hand image is used as a texture to improve realism. Although their technique produces results that are visually appealing, they avoid animation and skin deformation [71]. The hand model developed by Rhee et al. can be seen in Figure 2.11 (c).

One of the first works to address the use of virtual hands for sign language visualisation is that of Steinback [83] that is applied to finger spelling. Steinback's hand model is based on the one developed by Rijpkema [72]. The model is highly simplified with only a single layer modelled as separate geometric segments that results in a highly unrealistic looking hand. The hand model developed by Rijpkema can be seen in Figure 2.11 (d)

McDonald et. al [60] aimed to developed an improved hand model for sign language visualisation but used the same modelling technique of separate geometric segments. Although McDonald's hand model has no embedded skeleton, it has realistic joint rotational limits and rotation correlation between finger segments. Van Zijl and Raitt [96] also modelled their hand as separate geometric segments but their goal was to develop a collision avoidance strategy for finger-spelling that is based on deterministic finite automaton (DFA). The hand model by McDonald et al. can be seen in Figure 2.11 (e) and that of Van Zijl and Raitt in Figure 2.11 (f).

2.6 Summary

The modelling of virtual humans is a very complex and time consuming process. There is a myriad of techniques to represent, model and parameterise VHs for animation. Two standards to parameterise and animate VHs are MPEG-4 FDPs and H-Anim. VH models developed thus far do not have all the necessary features to effectively visualise sign language. Due to the complexity of modelling and animating VHs, researchers found it necessary to develop separate modules to model and animate the face, body and hands which was then difficult to integrate. Earlier methodologies to model VHs adopted simple single layer models and was manually parameterised for animation. Later methodologies adopted complex multi layer models that are automatically or semi-automatically parameterised for animation.

Chapter 3

Sign Language Visualisation

In the previous chapter, our discussion was geared towards the modelling and animation of VHs. This was done as the majority of sign language visualisation (SLV) systems or frameworks do not discuss the methodologies and techniques they employ to model VHs. Also, some SLV systems use proprietary VHs systems or VH systems that are of poor visual quality that lack the necessary model features to effectively visualise sign language. In this chapter a focused literature review on SLV is provided. We first discuss sign language notation systems (SLNS) as they are part of many SLV systems and used as input to animate and control VHs. The use of Sign Writing Markup Language (SWML) as part of our open framework to visualise sign language is hereby motivated. Later we discuss SLV systems that use video with their advantages and disadvantages as well as discuss systems that use VHs with their advantages and disadvantages. We also give a brief overview of some SLV systems, along with the technologies, VH models and animation control inputs they employ. The chapter ends with a summary of our discussions.

3.1 Sign Language Notation Systems

A sign language notation system (SLNS) is a writing system to record sign languages for research or educational purposes [33] [81] [84]. There is no standard SLNS and there is much debate on their usefulness in an English to sign language translation system [43]. It was stated by Huenerfauth [43] that, "any symbolic representation of an ASL performance will omit some amount of detail, and choosing what details are acceptable to omit when developing an artificial encoding scheme for a natural language is a challenging and error-prone task". While this is a valid statement by Huenerfauth, the development of SLNSs led to the gathering of information on sign languages and a better understanding of the

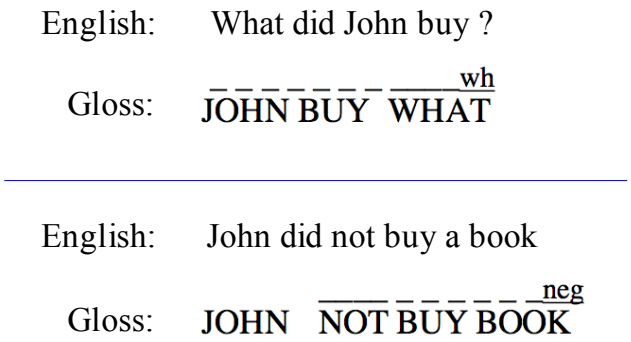

Figure 3.1: Glosses of American Sign Language and their English translation [5].

these languages and the Deaf [5] [33] [81] [84]. Also, with respect to SASL, Van Zijl [94] found that "there is almost no published information available". A SLNS would aid in gathering and publishing such information. The next sections discuss the most commonly used SLNS used in SLV with their advantages and disadvantages.

3.1.1 Gloss Notation

Gloss notation is employed not only in the transcription of sign languages but also to translate between different languages and hence the English glossing of sign languages [5]. A gloss, in the transcription of sign language, is written in all upper case characters and approximates an English word to a sign language gesture. Since gloss notation is used to transcribe sign languages on the word level, a significant amount of detail is left out, which leads to ambiguity [81].

Gloss notation has been further extended by different researchers. Some of these extensions include features, such as the marking and duration of non-manual gestures, as can be seen in Figure 3.1, by a line above the gloss [5]. Extensions by different researchers lead to non-standardised features which are still not enough for unambiguous interpretation. Because of non-standardisation and ambiguity, there are some difficulties in sharing knowledge with regard to sign languages [81]. In essence, gloss notation allows for easy transcription and basic translation but is not suitable for systems that aim to provide accurate SLV.

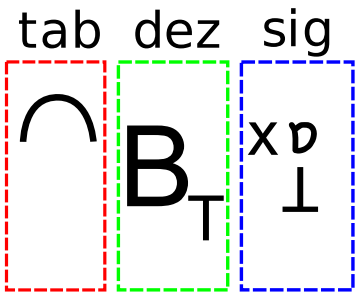

Figure 3.2: ASL transcription of "don't know" in Stokoe notation.

3.1.2 Stokoe Notation

In 1960, Dr. William C. Stokoe jr, published a paper on sign language structure and was one of the first to show that sign languages have linguistic features that showed them to be natural languages [84]. Through careful analysis, Stokoe showed that American Sign Language (ASL) contains both gestures and finger spelled English that are structurally different. Moreover, Stokoe introduced the concept of cherology which is equivalent to the phonology of spoken languages. Stokoe used the concept of cherology to develop a transcription system to aide in the study of sign languages or any gestural communication which was subsequently named Stokoe notation [84].

A Stokoe transcription of a sign language gesture is written from left to right and consists of three parts or cheremes (which are also referred to as phonemes) namely the tabula (tab), designator (dez) and signation (sig) as can be seen in Figure 3.2. As can be seen in Figure 3.2, these phonemes are written with the use of roman characters and iconic symbols designed by Stokoe himself. The tab phoneme relates to the positon relative to the body where a gesture is performed, such as parts of the face or trunk of the body. The dez phoneme relates to the configuration of one or both hands, such as their shape and orientation of the palm. The sig phoneme relates to the change of the tab or dez phonemes that results from signing [84].

To use Stokoe notation as input to SLV systems, some researchers have developed variants such as ASCII-Stokoe [58] to encode phomemes as ASCII characters. Since Stokoe notation is based on the concept of phonemes, it has a few advantages over the use of gloss notation when used as input to a SLV system. Some of these advantages include: clearly defined hand positions, hand shapes and palm orientations; transcriptions of successive or repetitive motions; and transcriptions of simultaneous motions. These advantages allow for the re-use of animation data when encoding phonemes as animations and leads to

smaller memory requirements as opposed to encoding animations as gloss [39] [49].

Stokoe notation also have disadvantages similar to that of gloss, such as that it leaves out a significant amount of detail with regard to non-manual gestures. Another limitation of Stokoe notation includes the fact that it has only a total of 19 hand shapes and 13 hand locations, which is much less than the actual number of hand shapes and locations found in sign languages today [84] [91] .

3.1.3 Hamburg Notation System (HamNoSys)

The Hamburg Notation System (HamNoSys) [92] was created at the University of Hamburg and is primarily used for research. HamNoSys, which is based on Stokoe, is also a phonetic SLNS with several successive versions and improvements [92]. It makes use of a much larger "alphabet" than Stokoe notation with more than 200 iconic symbols in HamNoSys 3 [90]. Several changes and modifications are made in HamNoSys 4 [91] with more symbols to transcribe non-manual gestures. The improvements to transcribe non-manual gestures in HamNoSys 4 includes symbols for the transcription of: shoulder movements; body movements; head movements; eye gaze; facial expressions; and mouthing [31] [91].

HamNoSys can be applied to the transcription of any sign language as it does not use a national diversified finger spelling [92]. An example transcription of the German Sign Language (DGS) gesture for "going to" can be seen in Figure 3.3. A complete description of the transcription in Figure 3.3 can be found in [48].

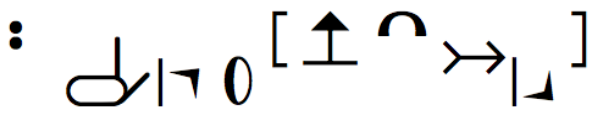

Figure 3.3: DGS transcription of "going to" in HamNoSys [48].

Although HamNoSys makes several improvements to Stokoe notation, it is still not without limitations. HamNoSys did not have a simple to use machine readable representation and its syntax was described as unwieldy by Kennaway [49]. It is also ambiguous in the sense that: it lacks default locations of the hands; it makes use of concepts such as "close to", "chest level", "fast" and "slow" with no exact values; and it does not specify duration of movements [49]. Moreover, the use of HamNoSys is limited to experts as it is difficult to use and possible to develop different transcriptions of the same gesture [49].

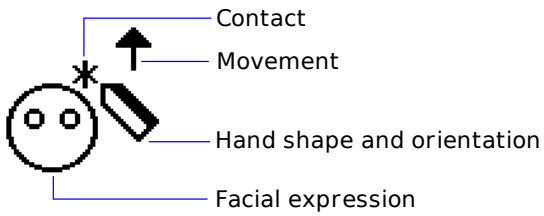

Figure 3.4: SASL transcription of "hello" in SignWriting.

3.1.4 SignWriting

SignWriting was invented by Valerie Sutton in 1974 and derived from her notation system to record body and dance movements [86]. Sutton developed SignWriting with the intention to record sign languages for research purposes [86]. Over the years, SignWriting evolved with the aid of many Deaf people and has proven to be easy to use and able to represent any sign language [86]. This led to SignWriting being widely accepted by different groups and it making its way into education [33] [86]. SignWriting can be used to transcribe sign language on different levels of detail and includes symbols for hand shape and orientation, movements, facial expressions, shoulder movements, contacts, space and punctuation [86]. The symbols are used in a pictograph called a sign box which is very different from the previously discussed SLNS. An example for the SASL transcription of "hello" using SignWriting can be seen in Figure 3.4. Features such as hand and contact locations are indicated by their arrangement in a transcription.

In this thesis, our goal was to employ Sign Writing Markup Language (SWML) [87], which is the Extensible Markup Language (XML) form of SignWriting. SWML was developed for the storage, processing and interchange of SignWriting texts [87]. Due to the popularity of SignWriting, Sign Bank Markup Language (SBML) [86], which is a variant of SWML, has been used as a medium to store SignWriting databases such as SignPuddle [87]. SBML has a document type definition (DTD) that produces a more compact XML representation than the DTD for SWML. Both SWML and SBML are attractive to use in a SLV system when one considers the popularity of SignWriting. This will aid in gathering and publishing information on SASL. SignWriting however does have limitations in that it includes: less hand shapes than HamNoSys; if suffers from ambiguity problems similar to that of HamNoSys such as motion duration; and it requires some study to be used effectively as one can also produce different transcriptions of the same sign language gesture [87]. Despite these limitations, Papadogiorgaki et al. [65] were highly successful in using SWML in a SLV system.

3.2 Sign Language Visualisation Systems

We define sign language visualistion (SLV), also known as sign language synthesis by Grieve-Smith [39], as a process that uses any visual medium, such as captured images, 3D rendered images or video to display dynamic sign language performances from a transcription. This transcription can be based on an SLNS or any formal description that can be used by a computer program to infer a sign language performance. The nature of the SLV problem is such that one is required to build a database or lexicon of sign language gestures [39]. SLV systems can be classified by the type of visual medium or internal data representation they use to represent such a database or lexicon. In the sections that follow, we first discuss video SLV after which we discuss VH SLV that uses animation data.

3.2.1 Video Sign Language Visualisation

Video SLV systems can easily capture videos of real people performing sign language, which is then stored as dictionaries. Visualisation is then performed by merely looking up and displaying video entries in a dictionary. Video based approaches have two primary advantages over VH SLV systems. The first advantage is the ease of capturing and acquiring a large lexicon of sign language gestures. The second advantage is being able to capture natural and realistic performances that incorporates both manual and non-manual gestures. There are however three very important problems that make the use of video based systems undesirable. The first problem is that the storage requirement for video of sufficient quality is high in that it requires large amounts of disk space [39] [83]. The second problem is that the transmission of video requires large amounts of bandwidth to make quality sign language material remotely available [63]. The third problem is the automatic joining and blending of videos to create natural new sign language output sequences which is extremely difficult [83]. A solution to the third problem of joining video segments was investigated by Krapez and Solina [50].

3.2.1.1 Investigating the Joining of Videos

Krapez and Solina developed a system to visualise Slovene Sign Language (SSL) by using gesture video segments [50]. The gesture video segments they captured, have a sign performer first assume a neutral pose, perform a gesture, and then assume the neutral pose afterwards. Their aim was to build a translation system that uses text as input from which a named sequence of sign language gestures were constructed. This named

sequence was then used to look up gesture video segments of sign language words. Once the sequence of videos have been looked up, they are automatically joined and blended together. To improve blending results between video segments, they introduced a function that uses four criteria to locate parts in video segments that are very similar. These criteria include [50]: palms start position; palms outside the start position; palms over the chest; palms close to each other. Their system could successfully join videos with satisfactory results although they needed to store palm locations for each frame of every video separately, which in turn increased the data storage requirement [50].

3.2.2 Virtual Human Sign Language Visualisation

Virtual Human (VH) Sign Language Visualisation (SLV) can be seen as the opposite of video SLV in the sense that it is more complex to create or capture animation data but in turn solves all three problems associated with video SLV systems mentioned in the previous section [39] [43]. VH SLV have several other advantages over video based SLV which include: VHs can be interchanged or have their appearances altered whereas video must keep with a single signer; visualisations can be viewed from different viewpoints; and the possibility to visualise advanced sign language features, such as classifiers that require dynamic locations in space [34] [39] [43] [65]. The creation and capture of animation data is discussed in Section 3.2.2.1 as it pertains to the problem of creating or capturing an animation dictionary or lexicon. Animation data is usually created with keyframing or captured with motion capture equipment. Both of these techniques have advantages and disadvantages. Other important disadvantage of VH SLV is the difficulty to model and animate VHs of adequate quality as discussed in Chapter 2, as well as to animate VHs on a higher level through an SLNS or formal description. In Section 3.2.2.2 we give a brief overview of VH SLV systems and discuss aspects such as the technologies, VH models and input for animation control they employ.

3.2.2.1 Creating Animation Dictionaries

Keyframing

Keyframing is the process of posing a VH by hand and storing DOF parameters, such as joint transformations (rotation angle, translation, scaling), morph targets or any type of animation control parameters as key values (poses) at specified frames [85]. Animation is then achieved by interpolating between key values over time. The skeleton of a VH can be posed by either forward kinematics (FK) or inverse kinematics (IK). Forward kinematics

is the process of specifying joint rotation angles in a forward manner beginning at an ancestor, joint moving all the way down to descendant joints [41]. Inverse kinematics is the process of placing an "end effector" (child or descendant joint) at an arbitrary location in space and automatically calculating joint rotation angles of ancestor joints to satisfy the "end effector's" location [41]. The advantages of keyframing are: less storage requirements to that of motion capture as keyframes can be optimised; it is inexpensive to create animation data; and some interactive modelling packages provide automatic keyframing features to aid in the process [8] [16] [20]. Disadvantages of keyframing are: it can become complex and time consuming as it is dependent on the number of animation parameters that must be controlled; modelling packages that can be used for keyframing have a learning curve and therefore require user training; and modelling packages do not provide dictionary look-up facilities that can aid in the creation of complex motions by combining stored keyframe data [8] [16] [20].

Motion Capture

Motion capture on the other hand is the automatic capturing or recording of animation control parameters from a live human performance by using specialised equipment. Earlier motion capture equipment consisted of a wearable suit or gloves with potentiometers to measure body and hand movements [85]. Later motion capture equipment employ optical markers attached to a performer and use specialised cameras to record face and body movements [85]. An obvious advantage of motion capture is the capture of realistic and lifelike human performances. Disadvantages associated with motion capture includes: expensive equipment; it is time consuming and difficult to set up and calibrate motion capture equipment; and captured motion data must be pre-processed else it is too difficult to use and edit [43] [48] [85].

3.2.2.2 Overview of Systems

SignSynth

SignSynth is an online SLV system developed by Angus Grieve-Smith at the University of New Mexico [39]. The system makes use of the older Web3D technology, VRML [2] for 3D visualisation. It employs ASCII-Stokoe as input to control a simple VH that is capable of manual and non-manual gesture animations. The ASCII-Stokoe is parsed with a Perl script which then provides input to a keyframe animation generation module. The animation generation module in turn builds a VRML file containing the VH with

animation data that is subsequently published on the internet [39]. The VH developed by Grieve-Smith can be seen in Figure 3.5 (a).

ViSiCAST and eSIGN

ViSiCast, with eSign being its successor, were European Union funded projects developed at the University of East Anglia to translate from English to BSL and other European sign languages [31] [48] [108]. A custom 3D rendering application was developed for ViSiCAST and a web plugin for eSIGN [108]. The VH used in ViSiCAST, namely Visia, was developed by Televirtual Ltd. Visia has a custom skeleton and can perform body and hand animations but is unable to perform facial animation. Guido on the other hand, also developed by Televirtual and used in eSIGN, is more advanced than Visia and can perform facial animation by using morphs [31]. There is not much information on the internal development of both Visia and Guido as both ViSiCAST and eSign are closed source projects. To control the VHs, a formal and proprietary XML representation of HamNoSys, namely SiGML, that uses keyframe animation, was developed as input to both VisiCAST and eSign [31] [49]. The animation generation module that takes SiGML as input also considers a VH's skin geometry, skeleton and surface feature points to visualise sign language from SiGML [31] [49]. The VH Visia from VisiCAST can be seen in Figure 3.5 (b) and Guido from eSIGN in Figure 3.5 (c).

Thetos

Thetos is a translation system that was developed at the Silesian University of Technology to translate from written or spoken Polish to Polish Sign Language [35]. Their system is implemented with OpenGL as a custom 3D rendering application to visualise sign language [105]. A simple VH with 15 DOFs in each hand that can only perform manual gestures (see Figure 3.5 (d)) was developed. To control and animate their VH, a formal "gestographic notation" that is used by the Polish Deaf community and to encode keyframe animations was employed [35].

The SYNENNOESE project

Karpouzis et al. [46] implemented a virtual signer tool for the SYNNENOESE project which is used to help in the education of Greek Sign Language (GSL). Their tool uses VRML [2] for 3D visualisation and H-Anim as a skeletal representation for their VH. There is not much detail on their VH and it is only capable of performing manual gestures. An

interpreter tool is used to convert GSL HamNoSys transcriptions into a keyframe based scripting language called Scripting Technology for Embodied Persona (STEP), which is subsequently used to control and animate their VH [46]. The VH used in the SYNENNOESE project can be seen in Figure 3.5 (e).

VSigns

VSigns was developed by Papadogiorgaki et al. [65] at the Informatics and Telematics Institute in Greece. They employed an MPEG-4 BAP player developed by École Polytechnique Fédérale Lausanne (EPFL) for hand and body animation and an MPEG-4 FAP player developed by EPFL and the University of Geneva for facial animation.The two MPEG-4 animation players were adapted and integrated with the head model attached to the body. Teeth were also later added to improve the appearance of the VH [65]. Their system makes use of SWML as keyframe based input that is interpreted and converted to MPEG-4 FBAP. The MPEG-4 FBAP in turn is generated as a VRML animation with an H-Anim compliant VH (see Figure 3.5 (f)) to visualise sign language [65].

Auslan Tuition System

The Auslan Tuition System was developed at the School of Computer Science and Software Engineering at the University of Western Australia as a teaching tool for Australian Sign Language (Auslan) [104] [106] . A custom 3D rendering application was developed using OpenGL for visualisation [105]. An earlier version of the system used a cartoon like VH skin and a skeleton with 39 joints that could only perform manual gesture animations. The skin was subsequently replaced with a VH skin modelled in Poser [80], which can be seen in Figure 3.5 (g). Their system was later extended with non-manual gestures by using facial morphs for facial animation on their new model [104]. A custom XML representation that stores keyframe animation data was developed for input to the system to visualise Auslan [104] [106].

SASL-MT

The South African Sign Language Machine Translation (SASL-MT) project that is to translate from English to SASL is being developed at the University of Stellenbosch [94]. The facial animation system by Barker [14], discussed in Section 2.5.1, was developed for the SASL-MT project and employs VRML for 3D visualisation. Also, a generic pluggable VH system that use H-Anim compliant VHs was developed by Fourie [34]. Fourie's

system was to be integrated with the facial animation system by Barker for a complete VH system. Fourie used Java 3D to implement the VH system and designed both an animation controller and SignSTEP as input to control and animate VHs. SignSTEP is based on STEP which is employed by Karpouzis et al. [46]. An H-Anim VH used in the system developed by Fourie can be seen in Figure 3.5 (h).

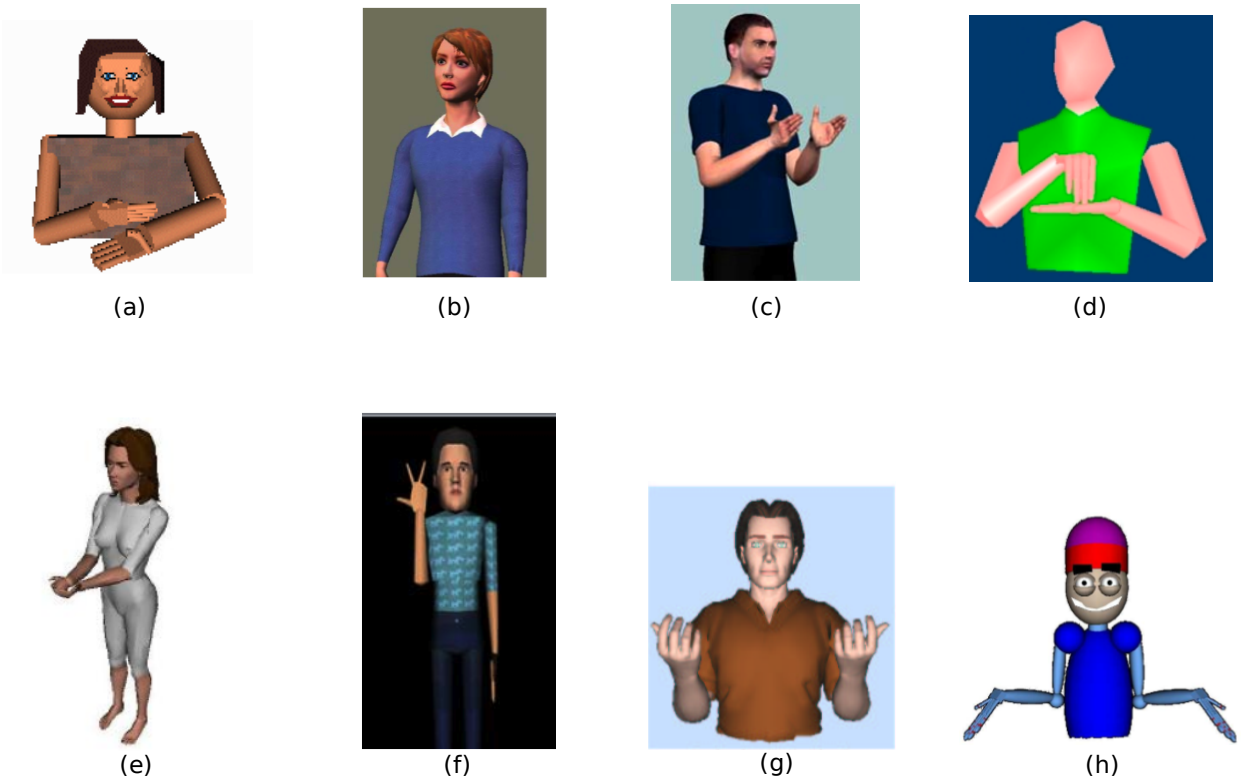

Figure 3.5: Virtual humans used by different sign language visualisation projects: (a) SignSynth [38]. (b) VisiCAST [108]. (c) eSIGN [108]. (d) Thetos [35]. (e) SYNENNOESE [46]. (f) VSigns [65]. (g) Auslan Tuition System [104]. (h) SASL-MT [34].

3.3 Summary

Sign language notation systems are used to record sign languages for research and educational purposes. All of the SLNS have limitations and need to be adapted to be used in an SLV system. The most attractive SLV is that of SignWriting as it is easy to use with a large user community. VH SLV systems have a significant number of advantages over SLV that uses video. Also, the majority of VH SLV uses keyframe animation data which is difficult to create but easier to manage and use.

Chapter 4

Methodology and Implementation

In Chapter 2, we established the need for an inexpensive methodology to simplify the modelling of quality VHs with the necessary features that can perform face, body and hand animation. The need for such a methodology was supported by Chapter 3 as SLV researchers do not concentrate on the challenges of modelling VHs. Their efforts are mainly focussed on employing standards and developing animation control for SLV. In this chapter, we present our experimental research approach and the methodology we developed to model and animate VHs of adequate quality with the necessary features that can perform face, body and hand animation. Our methodology is unique in the sense that we employ standards and open technologies to conceive a simple yet effective methodology and framework to model and animate VHs. We employ the H-Anim standard, that we adapt and extend with a slight variation of MPEG- 4 FDP facial feature points, to build a generic skeleton. This generic skeleton is then enhanced further by developing flexible hands and imposing joint rotational limits to ensure physically plausible poses. We also discuss animation controllers that we implemented in Blender. At the end of the chapter we provide a summary of our methodology.

4.1 Open Technologies

To conceive a methodology and open framework to model and animate VHs for SLV, we initially had to investigate various standards and open technologies in order to find ones that satisfy our requirements. Our requirements for standards and technologies are that these must be: open (open source) with active development and large user communities; flexible and extensible; allow us to model VHs with all the necessary features; easy to use; provide programmable interfaces and deliver good results in terms of visual quality

and real-time performance. The MPEG-FDP and H-Anim standards we discussed in Section 2.4.1 and Section 2.4.2 respectively have been widely adopted and satisfy some of our requirements despite their limitations. In the sections that follow, we discuss the open technologies we employ that we believe satisfy all our requirements in addition to adapting and extending the standards mentioned above.

4.1.1 MakeHuman

MakeHuman is an open source project developed by the MakeHuman Team [57] to address the need to easily model quality and realistic looking VHs. We decided to make use of MakeHuman as it is a completely free and open source parametric polygonal VH modeller (see Section 2.2.2) used by professionals to design VHs [57]. By employing MakeHuman, we free ourselves from the burden of modelling quality VHs from scratch or acquiring VH models of poor quality. One of the goals of MakeHuman is to develop an anatomically correct VH model that has only "*the necessary number of vertices and is optimised for animation*" [15].

MakeHuman has been in development for over a period of roughly 8 years with several improvements to both the user interface and the base (template) polygonal models being used. The earliest version of MakeHuman was developed as a Python script in Blender by Manuel Bastioni and had a polygonal model of the skin with approximately 7000 vertices [16] [15] [70]. Due to the complexity of the MakeHuman project and its slow performance as a Python script in Blender, later versions were developed as standalone C++ applications.

The version of MakeHuman we employ in this thesis, version 0.9.0, is a C++ application. The polygonal skin model in version 0.9.0, known as the K-Mesh by Kaushik Pal, was developed from anatomical references and includes separate polygonal models for the eyes, eyelashes, teeth and tongue [15]. The K-Mesh was designed to be sexually neutral and can be transformed to male or female figures. It has the following model geometry details [15]:

- Number of vertices: 10936
- Total number of faces: 10857
- Number of triangular faces: 470
- Number of square faces: 10387

The MakeHuman interface of version 0.9.0 along with the K-Mesh can be seen in Figure 4.1. MakeHuman 0.9.0 has a simple to use interface with a large database of modelling targets. It is also capable of posing VHs and exporting models in Wavefront object file format. At the time of writing, a new version of MakeHuman namely version 0.9.1 was released. MakeHuman version 0.9.1 has several improvements with a completely redesigned interface with the following features: Tetra-parametric GUI; Natural Pose System; improved polygonal mesh for subdivision; COLLADA[1] and Wavefront object export [57]. We refer the reader to the MakeHuman website for more details on these features [57].

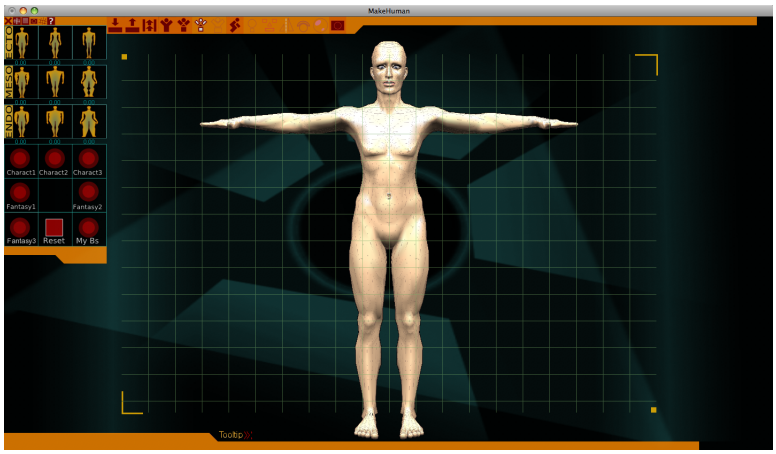

Figure 4.1: The MakeHuman interface of version 0.9.0 with the K-Mesh.

4.1.2 Blender

Blender is a free and open source generic interactive 3D modelling package that is being developed by the Blender Foundation [16]. Development of Blender began as a closed and internally used project by NeoGeo and Not a Number (NaN) in the late 1990s. It was subsequently released under the GNU General Public Licence after NaN declared bankruptcy in 2002 [17]. Blender is today one of the most active open source 3D projects with a large international user community. The large user community can be attributed to fact that Blender can be used for modelling, shading, animation, rendering, compositing and real-time interactive applications [16].

Blender, which is currently at version 2.48, is an advanced keyframe animation system and has a vast range of tools and features that include: vertex shape keys (morphing),

[1]COLLADA is an interchange file format for 3D applications.

Bézier, B-spline and quaternion curves for interpolation. The set of features implemented in Blender which are very attractive and allows it to satisfy our requirements are: its multi-window user interface; the skeleton (armature) system with scale, rotation and translation constraints; forward and inverse kinematics; Catmull-Clark subdivision surfaces; implementation of the automatic skinning algorithm by Baran and Popović (see Section 2.3.4); an embedded Python interpreter with an application programming interface (API); a state of the art internal game engine with its own Python API and a visual game logic editor [16]. With its advanced features and APIs, Blender can be looked at as an interface to 3D programming [16]. The sections that follow discuss aspects related to Blender's skeleton and animation system as well as Python and Blender's Game engine that we use in Section 4.5.

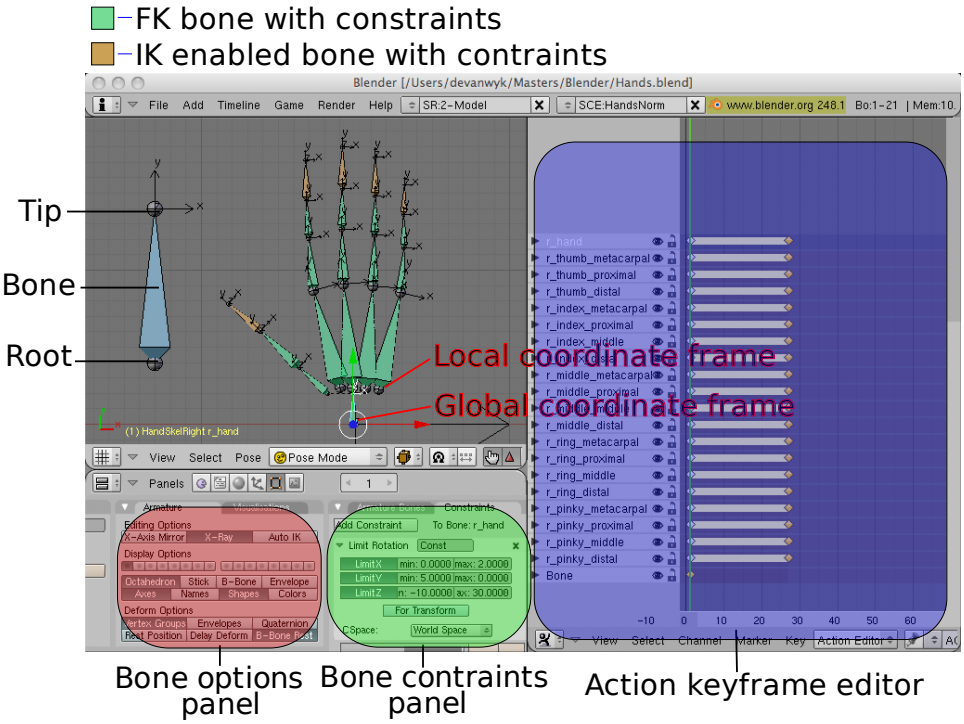

Figure 4.2: An example of a hand skeleton and Blender window configuration.

4.1.2.1 Blender's Skeleton and Animation System

Blender's skeleton system is much like the tree of bones described in Section 2.3.4. A skeleton in Blender has a root bone with the global or object co-ordinate frame for the entire skeleton (skeleton space) and a local co-ordinate frame for each bone (bone space). Each bone has a root and a tip, with the co-ordinate frame situated at the root where

rotation or articulation occurs. The root of a bone can thus be considered as a joint. A bone's co-ordinate system rotates with it and has the X- and Z-axes perpendicular to each other and to the vector from the root to the tip that represents the Y-axis (see Figure 4.2). As mentioned before, Blender is an advanced keyframe animation system with features such as constraints for its skeleton system, forward kinematics (FK), inverse kinematics (IK) and has several interfaces to aid in VH modelling and animation. An example of a hand skeleton and Blender window configuration that showcases the above mentioned features can be seen in Figure 4.2.

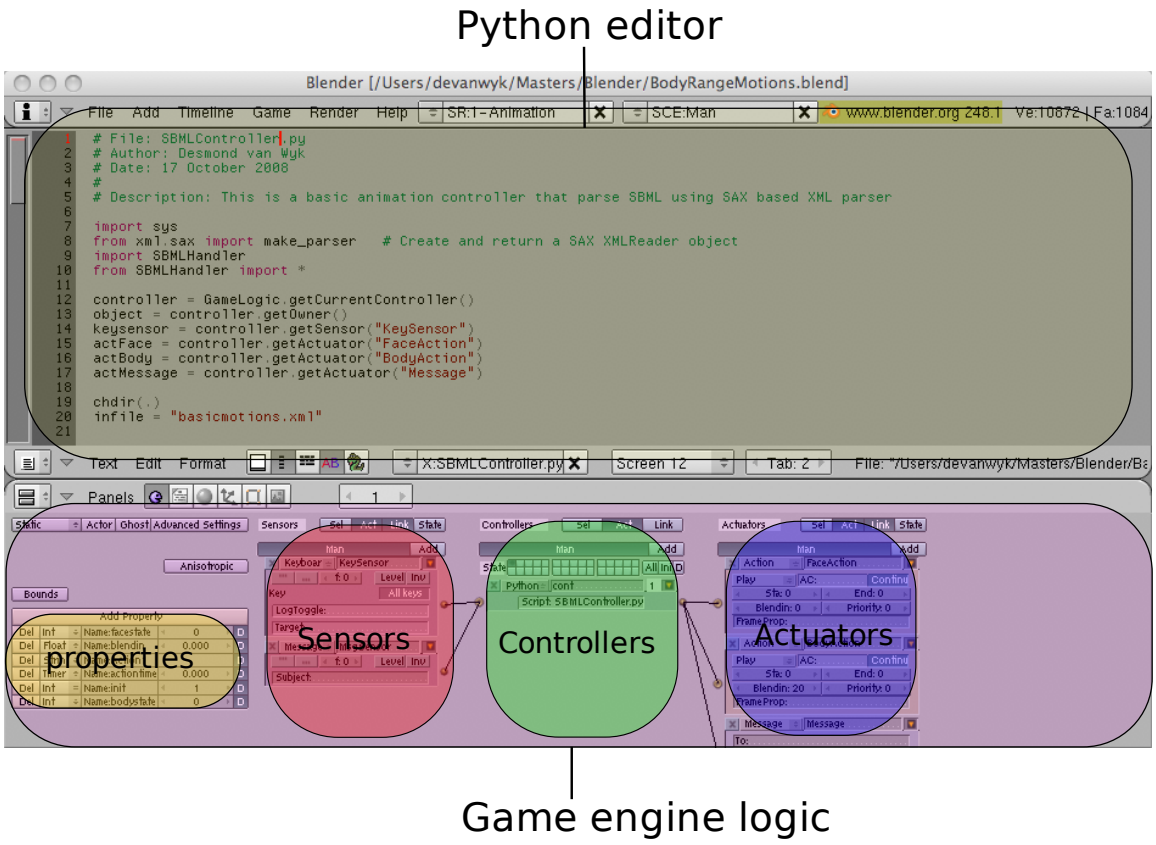

Figure 4.3: An example window configuration of Blender's Python editor and game engine logic.

4.1.2.2 Python and Blender's Game engine

Python is a dynamic object-oriented interpreted programming language that is being developed open source by the Python Software Foundation [70]. The Python language specification is very powerful yet simple and makes the language easy to learn [70]. Python has an extremely large library which is developed as modules that can be easily distributed.

It has modules for: internet data handling; operating services; data compression and archiving; networking; and structured markup processing such as XML to name but a few [70]. Also, what makes Python really attractive, is that it allows for extension and integration with other languages and tools [70]. As mentioned before, Python has been integrated into Blender and the Blender game engine [16].

The Blender game engine is a high performance game engine that exists within Blender and has advanced features such as: skeletal animation; replay of keyframe animations; game physics; and most importantly game logic which includes properties, sensors, controllers (AND, OR and Python controllers) and actuators for advanced animation control. All this can be edited in Blender's interface [16]. By integrating the game engine within Blender and also using Python, development time of complex projects can be significantly reduced. An example of a Blender window configuration that showcase Blender's Python editor and the game engine's game logic can be seen in Figure 4.3.

4.2 Modelling Process

The modelling process is the part in our methodology in which we employ MakeHuman and Blender as part of our framework to model VHs before parameterising and animating them. We first discuss the modelling we performed in MakeHuman after which we discuss the modelling we performed in Blender.

4.2.1 MakeHuman

In Section 4.1.1 we elaborated on MakeHuman and the base model's attributes except for the default pose the model assumes. The model assumes the "crucifixion" pose (see Figure 4.1), meaning it has its arms stretched out to the sides instead of hanging down as proposed by the H-Anim standard [3]. The "crucifixion" pose was proven through experience to be the best default pose to use when modelling and parameterising a VH model [15]. We therefore left the model in the "crucifixion" pose. Experimentation showed that we needed to open the mouth for correct parameterisation of the mouth and jaw region. Also, it was simpler to adapt and embed an H-Anim LoA 2 skeleton in the "crucifixion" pose with the jaw open (see Section 4.4). MakeHuman has hundreds of targets so we initially modelled only a single VH. Targets we changed for a more masculine appearance, without affecting the model's segment lengths are listed in Table 4.1.

MakeHuman target	Value
head_baby	0.20
jaw_open	0.20
neck_muscular	0.30
dorsi_muscular	0.50
pectoral_muscular	0.20
pectoral_forward	0.50
trapezious_muscular	0.30
r_shoulder_move_sideways_out	0.40
r_shoulder_move_sideways_in	0.50
l_shoulder_move_sideways_out	0.40
l_shoulder_move_sideways_in	0.50
r_upper_arm_fat	0.50
l_upper_arm_fat	0.50
r_lower_arm_fat	0.30
l_lower_arm_fat	0.30
abdomen_muscular	1.00

Table 4.1: MakeHuman targets changed to give a more masculine appearance.

4.2.2 Blender

After modelling in MakeHuman, we exported the model as a Wavefront object file that was subsequently imported into Blender. The model's scale was clamped to 30 units and was imported as separate objects by separating them by material.[2] By clamping the model's scale to 30 units, it is imported at a reasonable viewable size. Also, importing the model as separate objects by material results in polygonal models for the skin, eye balls, pupils with eyelashes, lips, teeth and tongue. While importing the model into Blender, we experienced some errors due to minor incorrectly defined material groups by MakeHuman's Wavefront object file exporter. Thus, some models had some of their vertices assigned to other models such as skin vertices assigned to the teeth model for example. All of the models were thus closely inspected and these minor importation errors were manually corrected. The models for the pupils and eyelashes were separated and the eyelashes were combined with the skin and lips. We found that the original eye models were too complex as these consisted of separate eyeball, iris and pupil models. We therefore replaced the original eye models with separate half spheres. The half spheres were given material colours to resemble eyes and translated to the locations of the previous

[2]A material consists of a set of properties that affects the shading of a model.

eye balls. Instead of developing separate models for the clothes, we applied a different material to the torso and upper legs of the model. The eyebrows were also created as a different material on the face of the skin. The resulting polygonal model of the combined skin, lips and eyelashes, as well as separate models for the eyes, teeth and tongue can be seen in Figure 4.4. Table 4.2 displays the geometric details of all the models.

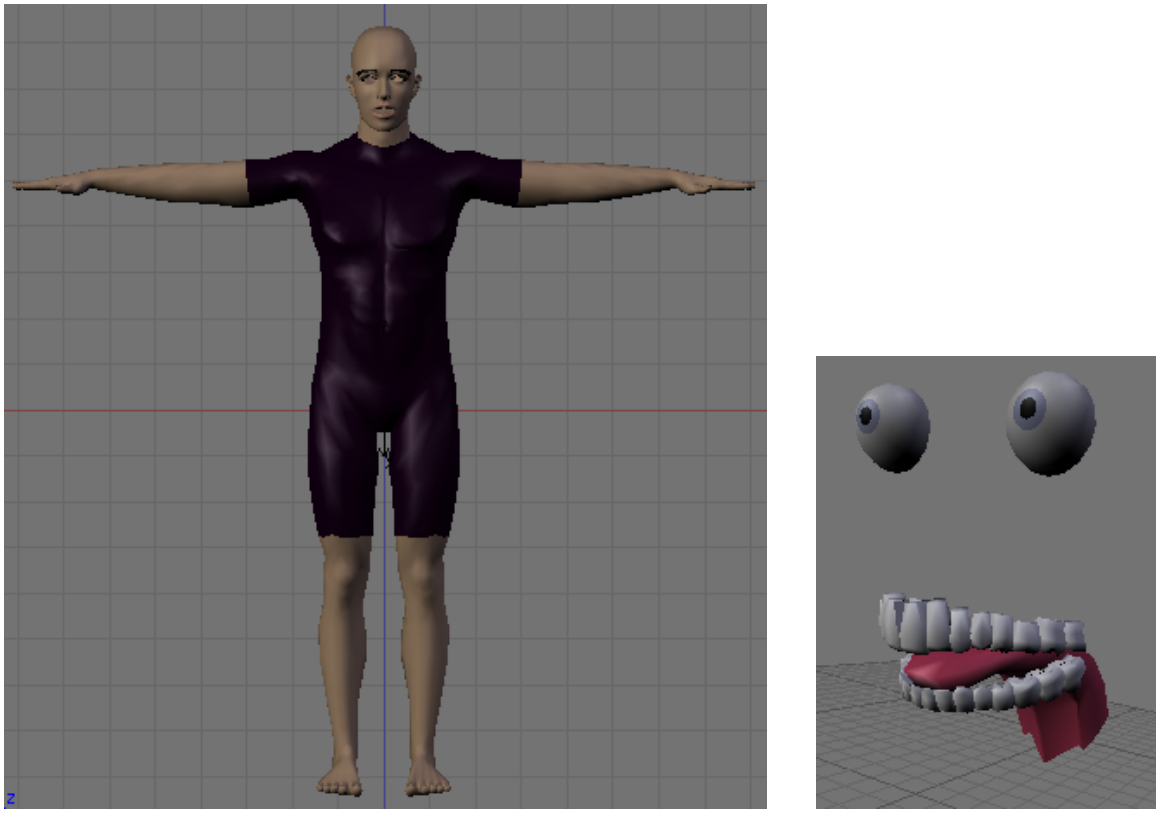

Figure 4.4: The resulting polygonal model with the skin, lips and eyes lashes combined and the separate eyes, teeth and tongue models.

Model	**Vertices**	**Edges**	**Faces**
Skin, eyelashes and lips	9247	18585	9302
Each eye model	129	256	128
Teeth	1215	2291	1140
Tongue	152	294	143

Table 4.2: Geometric details of the resulting models.

4.3 Adapting and Extending H-Anim

In Section 2.4.2 we discussed H-Anim and its limitations where we also motivated its adaptation and extension. In this section, we address those limitations by adapting and extending H-Anim to meet our requirements for a seamless generic parameterisation of a VH that can perform face, hand and body deformation. We wish to point out to the reader that we only take the structure and the naming interface used by the H-Anim standard and do not develop an H-Anim compliant application. Compliance to the H-Anim standard can easily be achieved by developing Python scripts in Blender. The H-Anim LoA 2 skeleton that we modelled in Blender (see Section 2.4.2), from the sample source found in [3], was used for adaptation and extension as this LoA 2 was successfully used by Seo et al. [76]. Our adaptation and extension process started with the body, then the hands and finally the face. We mirrored the bones of the skeleton while it was being developed to simplify the process.

4.3.1 The Body

The H-Anim LoA 2 skeleton was manually fitted inside the model from Section 4.2. From here on forth, we will use the naming convention followed by the H-Anim standard (name of joints) for Blender bones that can also represent segments [3]. All bones were given limits for their rotational DOFs to ensure physical plausible poses during animation [34]. Fitting of the skeleton started with the HumanoidRoot (root bone) and sacroiliac (pelvis bone) in the pelvis region of the model. The HumanoidRoot and sacrolliac both have 0 degrees (°) of rotational DOF.

The spine with 3 lumbar vertebrae bones (vl5, vl3, vl1), 3 thoracic vertebrae bones (vt10, vt6, vt1), 2 cervical vertebrae bones (vc4, vc2) as well as the skullbase bone was built vertically, aligned upwards, extending from the sacroiliac into the head of the model. The segment for the thoracic vertebra bone, vt6, was also designed to represent the breastbone (sternum). All the vertebrae including the skullbase bone were given limits of 20° for all their DOFs [101].

The sternoclavicular bones were placed perpendicular under the collar bone location, parallel to the vertebrae, but pointing in the opposite direction. The sternoclavicular bones were given limits of 0° for their DOFs. The acromioclavicular bones being perpendicular to the sternoclavicular bones, extends from the tips of the sternoclavicular bones up to the shoulder bones. Acromioclavicular bones articulate at the sternoclavicular bones and are responsible for the shoulder girdle (scapula and clavicle) movement.

These bones were given limits of rotational DOFs according to the values in Table 4.3 which was found by McClure et al. [59].

Axis	**Limit range**
X-axis	-5 – 50
Y-axis	-13 – 30
Z-axis	-13 – 24

Table 4.3: Limits of rotational DOFs in degrees (°) for the acromioclavicular adopted from McClure et al. [59].

The shoulder bones extend up to the elbow bones which in turn extend up to the wrist bones. Since the skeleton was mirrored and with swapped[3] rotational limits for mirrored bones, we only show rotational limits for the right side of the skeleton. Table 4.4 displays values for the shooulder, elbow and wrist we adopted from Boon [19]. The shoulder is a very complex joint that articulates with the acromioclavicular and has varying degrees of freedom for its axes of rotation depending on whether it is abducted or in the neutral position. A problem we experienced was that the neutral position for abduction of the shoulders was to have the arms hang down the sides of the body as in the H-Anim specification [3] [19]. To compensate for this, we first rotated the arms down by 90° to the sides and applied rotational limits from there onwards. Also, maximum values of rotational limits from Boon [19] were used for joint movements where lesser values prohibited certain movements. The technique employed by Boon [19] to measure joint rotational limits, is similar to that used by Cave and Roberts [22]. We used the approach by Cave and Roberts [22] for experimentation in Section 5.2 of Chapter 5.

Bone (joint)	**Limit X-axis**	**Limit Y-axis**	**Limit Z-axis**
shoulder	-90 – 80 (neutral abduction = 170)	-53 – 158	-45 – 135
elbow	0	-71 – 84	0 – 146
wrist	-73 – 71	0	-33 – 19

Table 4.4: Limits of rotational DOFs in degrees (°) for the shoulder, elbow and wrist adopted from Boon [19].

[3]The axes to which the rotational limits are applied are swapped (e.g X becomes -X).

4.3.2 The Hands

The human hands are complex articulated structures that we use to physically interact with the world around us and critical for sign language performances. Even though the hands of the H-Anim LoA 2 skeleton is simpler than that of a real skeleton, they are still powerful due to the flexibility of the H-Anim standard. In this section we address the limitations of the H-Anim hands as found by Elliot et al. [31].

Joint centre locations of the hands were manually estimated with the placement of the carpometacarpal joints (pinky0, ring0, middle0, index0, thumb1) closer to the wrist joint than that of the LoA 2 example [3] [6] [7]. Table 4.5 displays the limits of rotational DOFs for the joints in the right hand that we adopted from Albrecht [6] except for thumb1. The range of movement of the carpometacarpal joints cannot be easily measured and it is noted by Albrecht [6] that only the carpometacarpal joints namely thumb1, ring0 and pinky0 are rotational. Through experimentation, the carpometacarpal joint of the thumb namely thumb1, showed that it required greater rotational DOF (see Section 5.3).

Bone (joint)	**Limit X-axis**	**Limit Y-axis**	**Limit Z-axis**
thumb1	-180 – 20	-180 – 0	-90 –120
thumb2	-85 – 30	0	-5 – 5
thumb3	-90 - 60	0	0
middle0 index0	0	0	0
ring0	0	0	-5 – 5
pinky0	0	0	-5 – 5
middle1	-100 – 25	-7 – 10	-15– 15
ring1	-115 – 25	-5 – 25	-15 – 15
pinky1	-115 – 25	-5 – 18	-25 – 15
pinky2 ring2 middle2 index2	-110 – 5	0	0
pinky3 ring3 middle3 index3	-90 – 15	0	0

Table 4.5: Limits of rotational DOFs in degrees (°) for bones in the hand adopted from Albrecht [6] except for thumb1.

4.3.3 The Face

In Section 2.4.2 it was found that the H-Anim standard only has a simple set of facial bones to perform facial animation. Also, the H-Anim specification suggested using the MPEG-4 FAP set for facial animation. In this section, we extend the structure of the H-Anim LoA 2 skeleton by adding facial bones to our skeleton and model based on the MPEG-4 FDP facial feature points (see Section 2.4.1).

Bones were added manually by "snapping" them to vertex locations closest to MPEG-4 FDP facial feature points on the face of the skin model. The temporomandibular, which is part of H-Anim, was manually fitted in the centre of the jaw area of the model with an open mouth pose. Initial experimentation of parameterisation of the model indicated the need for extra bones on the face to avoid undesired deformations (see Section 5.4). Some extra bones, including bones located at MPEG-4 FDP facial feature points which are not affected by MPEG-4 FAP, we refer to as structural bones. Additional bones at the eye (l_eyelid_inner_up1 and l_eyelid_inner_up2) and eyebrow (l_brow_mid_in and l_brow_mid_out) areas were used as single MPEG-4 FDP feature points. The bones added for the face follow a different but more informative naming convention than that proposed by the H-Anim specification [3]. Also, a special child-parent relationship was established between the bones on the face, the temporomandibular bone and the skullbase bone to ensure the facial bones moved with the rest of the skeleton. A further addition was a separate skeleton for the tongue which was added as a child to the skullbase bone after parameterisation of the tongue. Bones for the tongue's skeleton were placed uniformly within the centre of the tongue starting from the base of the tongue to the tip and were named tongue1, tongue2, tongue3 and tongue4 (see Section 4.4.4). Table 4.6 displays the newly added facial bones that are children of the temporomandibular bone which in turn is a child of the skullbase bone. Table 4.7 displays the newly added facial bones that are direct children of skullbase. Table 4.8 displays the extra bones that we refer to as structural bones which are children of the skullbase bone.

Face area	**MPEG-4 FDP**	**Bone (joint) (parent: temporomandibular)**
Jaw	2.1	chin_bottom
	2.11	l_chin_corner
	2.12	r_chin_corner
	2.13	l_jaw_corner
	2.14	r_jaw_corner

continued on next page

continued from previous page

Face area	MPEG-4 FDP	Bone (joint) (parent: temporomandibular)
Inner lip	2.3	lip_mid_inner_low
	2.4	l_lip_inner_corner
	2.5	r_lip_inner_corner
	2.8	l_lip_mid_inner_low
	2.9	r_lip_mid_inner_low
Outer lip	8.2	lip_mid_outer_low
	8.7	l_lip_mid_outer_low
	8.8	r_lip_mid_outer_low
Cheeks	5.1	l_cheek_center
	5.2	r_cheek_center

Table 4.6: Facial bones that are children of the temporomandibular bone.

Face area	MPEG-4 FDP	Bone (joint) (parent: skullbase)
Eye region	3.1	l_eyelid_inner_up1 and l_eyelid_inner_up2
	3.2	r_eyelid_inner_up1 and r_eyelid_inner_up2
	3.3	l_eyelid_inner_low
	3.4	r_eyelid_inner_low
	3.5	l_eyeball_joint
	3.6	r_eyeball_joint
	3.7	l_eye_outer_corner
	3.8	r_eye_inner_corner
	3.9	l_eyelid_outer_low
	3.10	r_eyelid_outer_low
	3.11	l_eye_inner_corner
	3.12	r_eye_outer_corner
Eyebrow	4.1	l_brow_inner_corner
	4.2	r_brow_inner_corner
	4.3	l_brow_mid_in and l_brow_mid_out
	4.4	r_brow_mid_in and r_brow_mid_out
	4.5	l_brow_outer_corner
	4.6	r_brow_outer_corner
Cheeks	5.3	l_cheek_bone
	5.4	r_cheek_bone
Inner lip	2.2	lip_mid_inner_up
	2.6	l_lip_mid_inner_up
	2.7	r_lip_mid_inner_up

continued on next page

continued from previous page

Face area	MPEG-4 FDP	Bone (joint) (parent: skullbase)
Outer lip	8.1	lip_mid_outer_up
	8.3	l_lip_outer_corner
	8.4	r_lip_outer_corner
	8.5	l_lip_mid_outer_up
	8.6	r_lip_mid_outer_up
Nose	9.1	l_nostril
	9.2	r_nostril
	9.3	nose_tip
	9.4	r_nose_bottom_edge
	9.5	l_nose_bottom_edge
	9.6	r_nose_upper_edge
	9.7	l_nose_upper_edge
	9.12	nose_bump
	9.13	l_nose_lower_edge
	9.14	r_nose_lower_edge
	9.15	nose_edge_middle
Ears	10.1	l_ear_top
	10.2	r_ear_top
	10.3	l_ear_back
	10.4	r_ear_back
	10.5	l_earlobe_bottom
	10.6	r_earlobe_bottom
	10.7	l_ear_lower_contact
	10.8	r_ear_lower_contact
	10.9	l_cheek_upper
	10.10	r_cheek_upper
Forehead	11.1	forehead_middle
	11.2	r_forehead
	11.3	l_forehead

Table 4.7: Facial bones that are children of the skulbase bone.

Face area	Bone (joint) (parent: skullbase)
Upper Cheek and side of head	r_side_skull
	l_side_skull
	r_cheek_upper
	l_cheek_upper

Table 4.8: Extra structural bones that are children of the skullbase bone.

4.4 Parameterisation Process

After modelling and manually fitting our developed generic skeleton within the models, the last step before animation was to parameterise the models with the generic skeleton. Parameterisation of our models follows a semi-automatic sequential process to ensure models are completely and correctly parameterised. It is performed in the following order: skin, teeth, eyes and finally the tongue. There are a few conditions that a model must satisfy before parameterisation, to ensure correct deformations. These conditions we present with the associated part of a model that must be parameterised in the sections that follow.

4.4.1 The Skin

Parameterisation of the skin requires that the model be in the "crucifixon" pose with the jaw opened. Also, the skeleton for the tongue must be kept separate as discussed in Section 4.3.3 to avoid parameterisation of the jaw area with the tongue's bones. The entire skin model, which includes the eyelashes and the lips, as discussed in Section 4.2.2, is automatically parameterised.

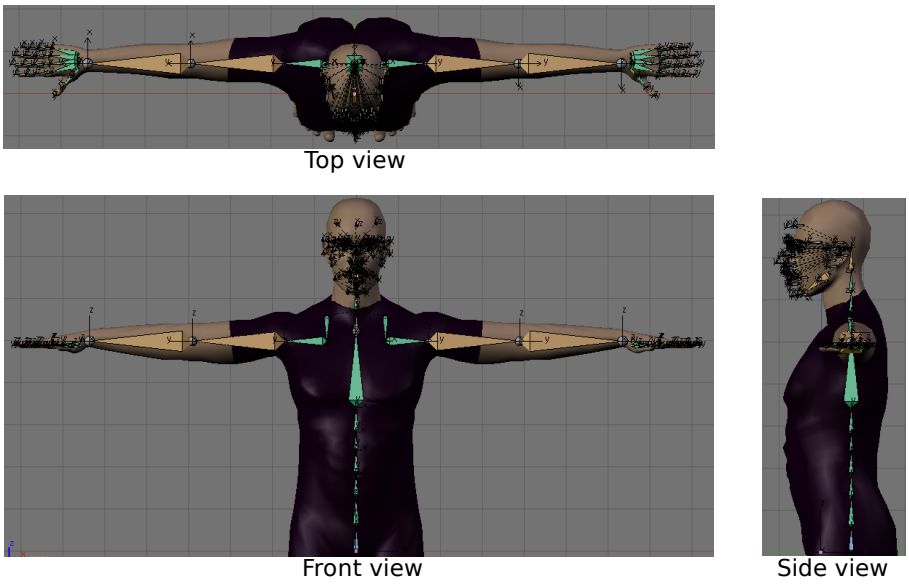

Figure 4.5: The skin manually fitted with the generic skeleton for automatic skinning.

We employ the automatic skinning algorithm developed by Baran and Popić, as discussed in Section 2.3.4, which was designed for articulated figures such as VHs. Since the current implementation of Baran and Popvić's algorithm in Blender does not perform

automatic skeleton fitting and simplification, it allows us to parameterise the face at the same time which is not an articulated figure. Figure 4.5 shows the model, along with the full generic skeleton, with different views for automatic parameterisation (skinning).

4.4.2 The Teeth

Parameterisation of the teeth also requires that the model of the jaw be open. It would not be possible to obtain a correct parameterisation if the jaw was closed, since the surfaces of the top and bottom teeth would intersect. The model for the teeth was thus manually parameterised by weight painting as it needs to be divided into the top set of teeth that remains stationary with the head (skullbase bone) and the bottom set of teeth that moves with the jaw (temporomandibular bone). Figure 4.6 (a) shows the parameterisation of the top teeth (red surfaces) with the skullbase bone and (b) the parameterisation of the bottom teeth (red surfaces) with the temporomandibular bone.

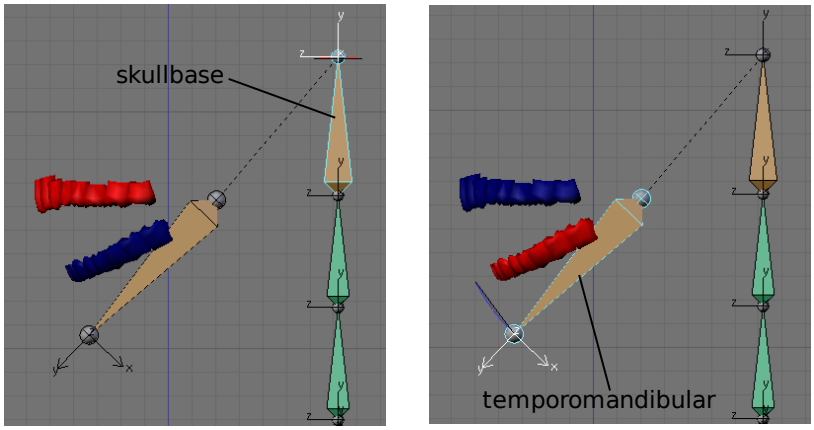

Figure 4.6: The teeth manually skinned with the skullbase and temporomandibular bones.

4.4.3 The Eyes

The eyes, which were separately modelled as half spheres, requires no special condition for parameterisation. These are also manually parameterised by weight painting the left eye model to l_eyeball_joint and the right eye model to r_eyeball_joint.

4.4.4 The Tongue

The tongue was given its own skeleton, as discussed in Section 4.3.3, which is first used to automatically parameterise the tongue model with Baran and Popović's algorithm. After parameterising the tongue with its own skeleton, it is attached as a child to the skullbase bone of the generic skeleton to finalise the parameterisation process. The model and skeleton for the tongue can be seen in Figure 4.7 after parameterisation and attachment to the skullbase bone.

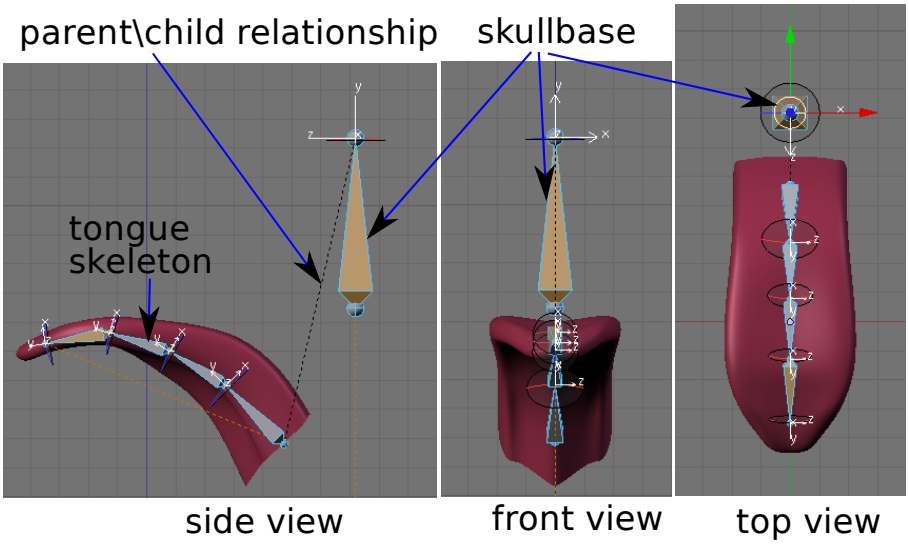

Figure 4.7: The tongue model and skeleton after parameterisation and attachment to the skullbase bone.

4.5 Implementing Animation Control

Animation control is the final feature required to realise a complete and open framework for the modelling and animation of VHs to visualise sign language. In Section 3.1 we discussed SLNS and motivated our use of SignWriting but most specifically its XML representations, which are SWML and SBML. Later, in Section 3.2.2.2, we provided an overview of different SLV systems and highlighted the different technologies, models and animation control input that they employ. In the sections that follow, we discuss how we designed and implemented animation control in the Blender game engine to visualise sign language, both interactively and procedurally. The Blender game engine is very flexible, as discussed in Section 4.1.2, with an integrated Python interpreter, APIs and a visual game logic editor. All the animation controllers we designed use game logic, such as

sensors, Python controllers, actuators and properties. The reader is referred to Appendix A for documentation on the GameLogic module which also contains a list of sensors and a list of actuators. From here on forth, we will refer to the game logic sensors, controllers and actuators by their names as in Appendix A.

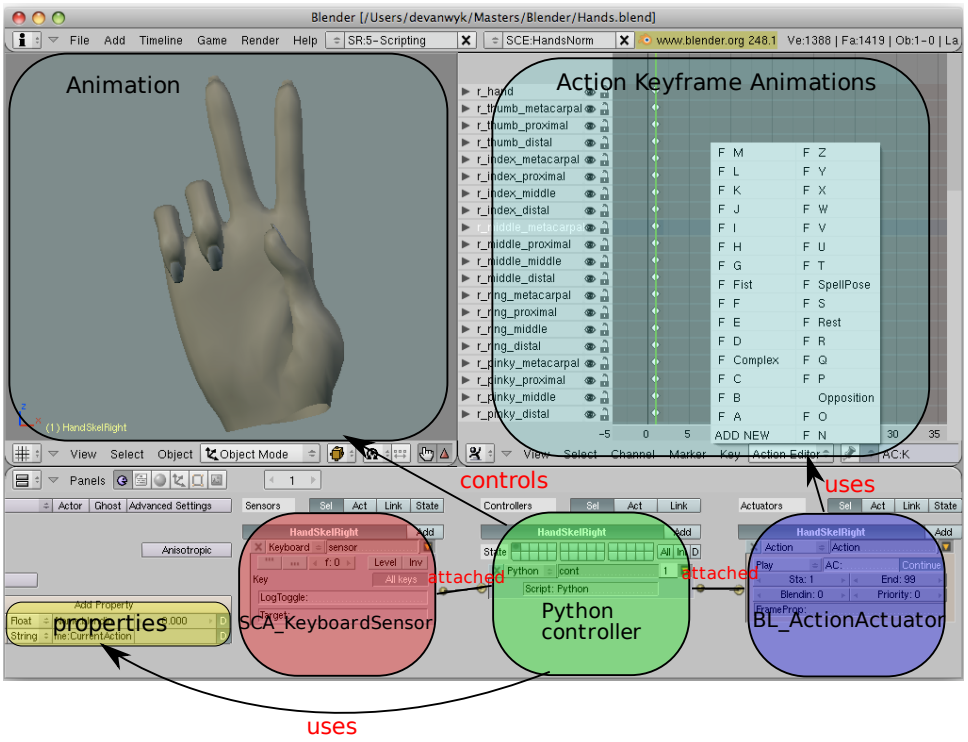

Figure 4.8: General design of interactive animation controllers displaying finger spelling animation.

4.5.1 Interactive Control

Three separate interactive animation controllers for the body, hand and face were designed to initially evaluate the Blender game engine and its Python APIs. All three animation controllers have the same design and each one has a Python controller attached to a SCA_KeyboardSensor and a BL_ActionActuator. A general design for all three animation controllers is depicted in Figure 4.8, which displays finger spelling animation. The SCA_KeyboardSensor waits for keyboard input, which is processed by the Python controller, that in turn activates the BL_ActionActuator to set and play an action keyframe animation. The BL_ActionActuator was specifically designed to be used for Blender's skeletal system that uses quaternions to perform rotations stored in action keyframe poses. Blending or transition between keyframe poses and different actions

therefore use quaternion interpolation. An advantage of using quaternions is that the problem of "gimbal lock", which is often encountered when using rotation matrices, is avoided [34].

The animation controllers for body and face animation are state based controllers that use state properties and loads different action keyframe animations sequentially until the last state and then back to the first state. The animation controller for the hand loads hand finger spelling action keyframe animations which depends on the letter of the keyboard being pressed. A "blendin" property is used by all three animation controllers to set the number of frames used to interpolate between animations.

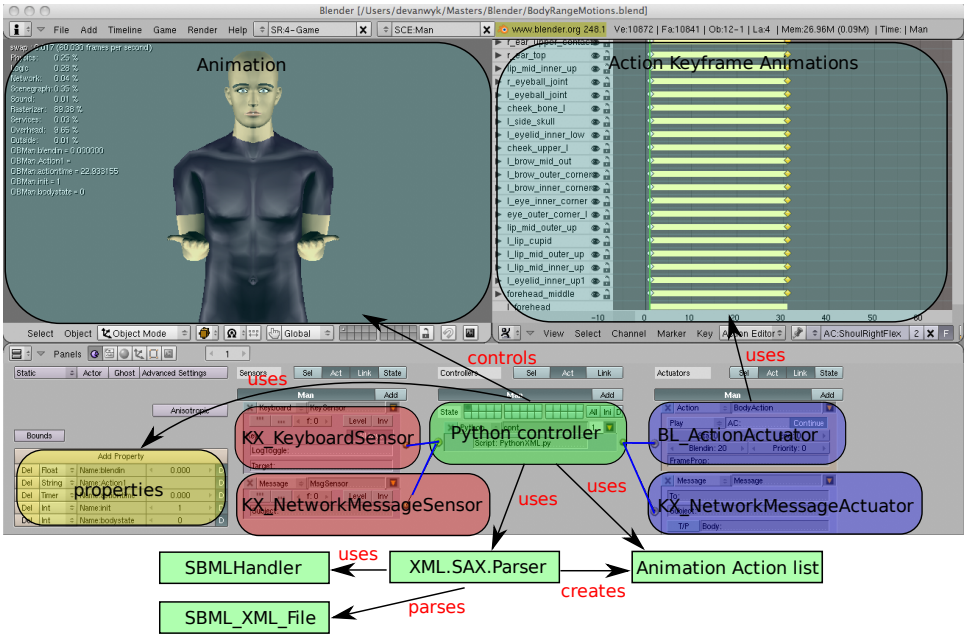

Figure 4.9: Design of the procedural animation controller displaying full virtual human animation.

4.5.2 Procedural Control

A procedural Python animation controller was designed to take SignWriting, in the form of an SBML file as input and to build an animation action list based on the contents of the file. The high level design of the procedural animation controller within a Blender window configuration is depicted in Figure 4.9. The procedural animation controller has nearly the same design as the interactive controller except that it is also connected to a KX_NetworkMessageSensor and a KX_NetworkMessageActuator as well as capable of parsing an SBML file with an extensible markup language (XML) parser.

We used the Python Simple API for XML (SAX) to create a parser and a SBMLHandler for parsing SBML files instead of a Document Object Model (DOM) parser [34]. The SAX parser is an event based parser which requires less memory than a DOM parser to create an animation queue or action list [34]. The KX_KeyboardSensor is used for starting the animation after which a KX_NetworkMessageSensor and KX_NetworkMessageActuator is used to establish a sequential animation loop to automatically activate the Python controller. The Python controller checks after each activation by the KX_NetworkMessageSensor if an action animation in the action list has finished being played by the BL_ActionActuator. The KX_NetworkMessageActuator sends a message to the KX_NetworkMessageSensor, regardless of whether an animation was finished, to reactivate the animation controller. Once an action animation has finished, the next action animation in the animation action list is loaded until the last animation after which it restarts from the beginning of the animation action list.

4.6 Methodology and Framework Overview

In Section 2.5 we reviewed the methodologies employed and VH models developed for animation. It was noted that researchers aimed their efforts to create separate modules for facial animation, body animation and hand animation which was then later difficult to integrate [44]. Also, methodologies developed by Moccozet et al. [61] and that of Wang and Ressler [99] are automatic or semi-automatic processes that employ scanned body data which required pre-processing and was still difficult to use for animation. Another important fact is that their methodologies employed skeletons which can only perform body (articulated figure) animation hence their discussion in Section 2.5.2.1. In this section we presented an overview of our methodology and framework to model and animate VHs with the necessary features that are of adequate quality to visualise sign language.

An overview of our methodology and open framework can be seen in Figure 4.10. Our methodology begins with modelling in MakeHuman, as discussed in Section 4.2.1. We then separate the skin, tongue and teeth models and also replace the eye models in Blender as discussed in Section 4.2.2 while correcting minor importation errors. After modelling, we follow the parameterisation process as discussed in Section 4.4 by using the H-Anim LoA 2 adapted and extended generic skeleton we developed in Section 4.3. Once we have our parameterised model, we can create action keyframe animations for our model and use these in the Blender game engine with Python and SBML for interactive

or procedural control as discussed in Section 4.5.

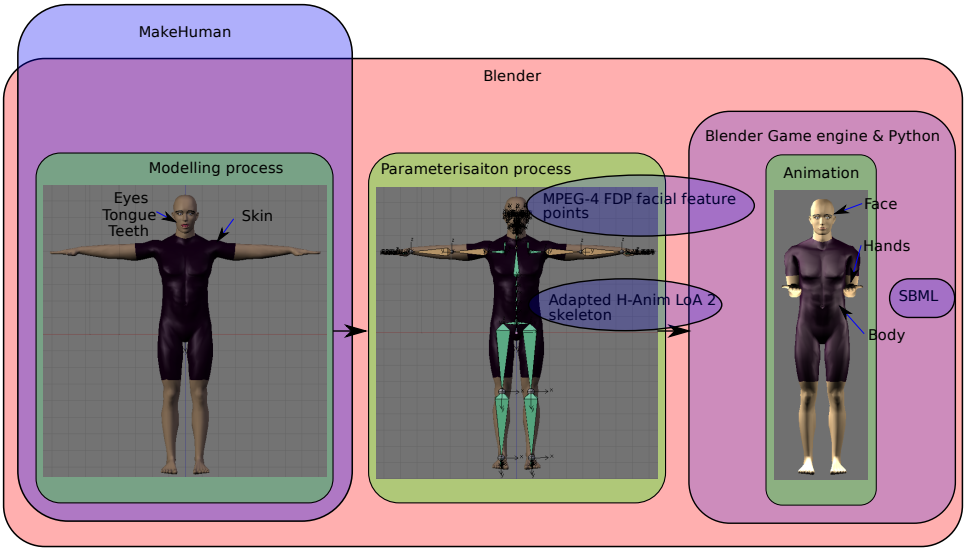

Figure 4.10: Overview of our developed methodology and framework.

4.7 Summary

In this chapter we discussed our methodology and open framework to model and animate VHs with the necessary features and which are of adequate quality to visualise sign languages. We elaborated on the open technologies we employed and how we used these to adapt and extend the H-Anim LoA 2 skeleton with joint rotational limits and facial bones to perform body, face and hand animation. The facial bones we added are in accordance with MPEG-4 facial feature points with minor variations. Finally, we discussed how we designed and developed interactive and procedural animation control that make use of SBML to realise a complete framework for sign language visualisation.

Chapter 5

Experiments, Results and Discussions

The previous chapter presented our research approach which resulted in a methodology and open framework to model and animate VHs for sign language visualisation. To remain true to our goals and answer our research question, in this chapter we experiment with our methodology and open framework to evaluate its efficiency and the quality of the models and animations it delivers. We begin by discussing the environment in which we perform experiments such as the hardware and software settings we use. Next we discuss the modelling of complete VHs with different appearances and proportions that we use in this chapter to perform experiments on body and face posing and animation. Also, we modelled three VHs that have hands of extreme minimum and maximum segment lengths to perform experiments on hand posing and animation. The purpose of multiple models of different appearances and proportions is to evaluate how easily we can share animation resources between different models with the assumption that it is possible with a generic skeleton.

5.1 Experimental Design

As mentioned in Section 2.5, in 3D computer graphics there is always the trade-off between visual quality and display speed. This trade-off is determined by the techniques for model representation and deformation, as well as the hardware employed [44]. We performed experiments on a MacBook Pro with a 2.16 GHz Intel Core 2 Duo processor, 1GB RAM and ATI Mobillity Radeon X 1600 graphics card. The MacBook Pro satisfies our requirement to develop a methodology and open framework on standard hardware

that can be shared easily.

The requirement for real-time sign language visualisation is an animation frame rate of between 15 and 25 fps [48]. In Blender's game engine, one can either enable all frames to render or clamp the maximum frame rate to 60 frames per second (fps). For all experiments performed, we enabled rendering of all frames to measure the maximum frame rates achieved during all animations. Experiments were separated into body, hand and face posing and animation, which we discuss later in the chapter. The sections that follow discuss the models we developed for the experiments. Moreover, we employed Catmull-Clark subdivision surfaces during hand and face animation experiments, discussed below, to further improve visual realism.

5.1.1 Modelling Multiple Virtual Humans

In the previous chapter we modelled a single VH while developing our methodology and framework. To fully evaluate our methodology and framework, we modelled three more VHs of different appearances and proportions. The same approach for modelling and parameterisation with our generic skeleton was followed, as discussed in the overview of our methodology, in Section 4.6. Minor importation errors into Blender, as discussed in Section 4.2.2, resulted in all four VH models having minor different geometric details, except for the eyes, that are displayed in Table 5.1.

VH	Model	Vertices	Edges	Faces
Man	Skin, eyelashes and lips	9247	18585	9302
	Teeth	1215	2291	1140
	Tongue	152	294	143
BigMan	Skin, eyelashes and lips	9217	18526	9275
	Teeth	1205	2275	1132
	Tongue	158	299	143
Woman	Skin, eyelashes and lips	9331	18714	9350
	Teeth	1215	2291	1140
	Tongue	152	294	143
Boy	Skin, eyelashes and lips	9273	18606	9300
	Teeth	1215	2291	1140
	Tongue	158	299	143

Table 5.1: Geometric details of the resulting models.

The first model we developed during our methodology we will from here on forth refer

to as Man, the second model as BigMan, the third model as Woman and the fourth model as Boy. Figure 5.1 depicts the four VHs we modelled and used in experimenting with posing and animation of the face and body.

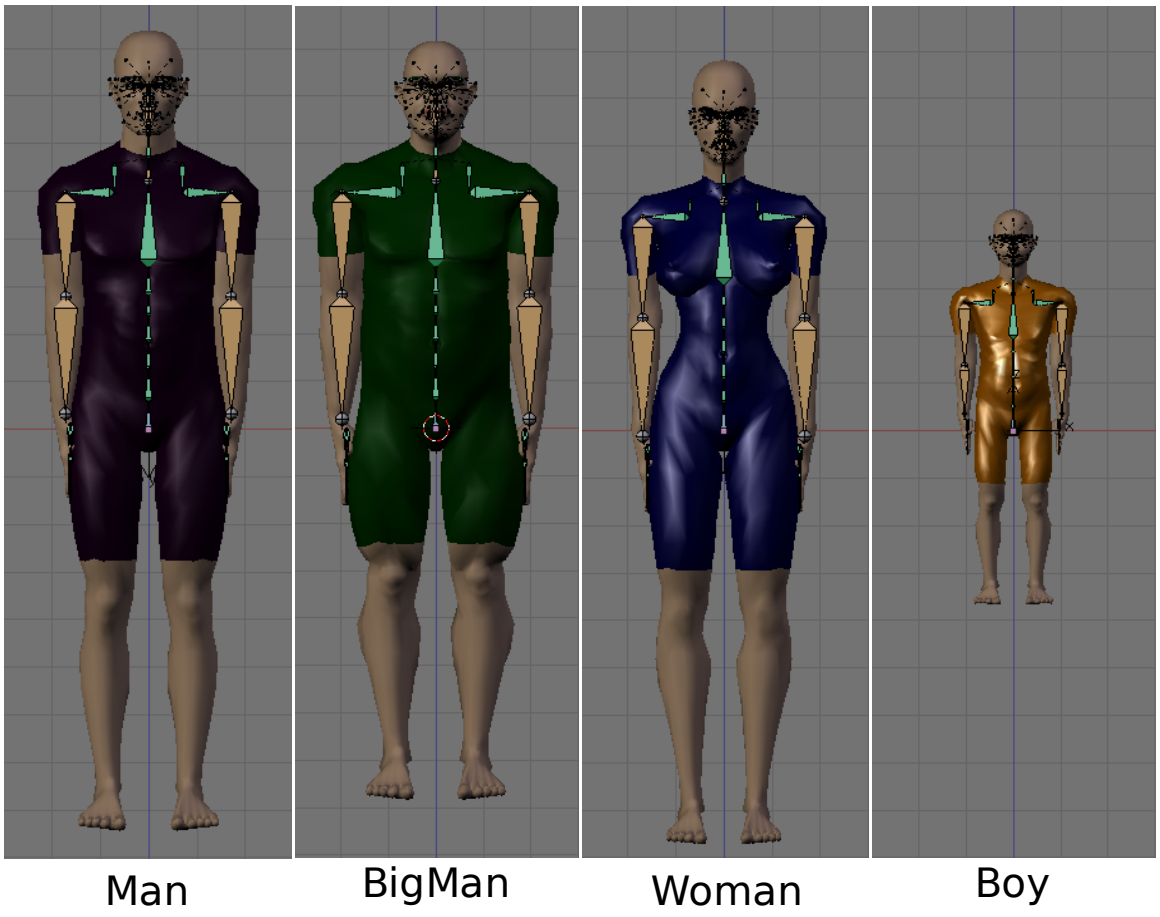

Figure 5.1: The four models we developed with our methodology and framework.

5.1.2 Extreme Modelling of the Hands

To fully evaluate if we can share animation resources between articulated figures, we modelled three virtual humans with different hand segment lengths, also following our methodology in Section 4.6. The hands were then separated from the bodies. The first model has unmodified hand parameters, the second model has hand parameters maximally set for short hands and the third model has hand parameters maximally set for long hands. For future reference, we will refer to the first model as Norm Hands, the second model as Short Hands and the third model as Long Hands.

5.2 Body Posing and Animation

Body posing and animation was the first experimentation we performed. For posing, we desired to know if we obtained quality parameterisations or skinning of the body during animation. Also, we wanted to evaluate if the joint rotational limits we employ for our generic skeleton ease the posing of VHs and if these prevent us from posing a character in non physically plausible poses. All models delivered quality parameterisations of which we will only provide results for Man in in this section. Certain bones were enabled as IK bones and was thus used to perform interactive posing of a VH. The bones we enabled as IK bones includes: r_shoulder, r_elbow, r_wrist, l_shoulder, l_elbow, l_wrist, vt1 and skullbase. The sections that follow provide posing results by body parts similar to the approach for measuring and recording joint function by Cave and Roberts [22].

5.2.1 The Spine

Cave and Roberts [22] state that the neutral position for the spine cannot be defined and it can perform: forward bending; extension; left and right lateral bending; and rotation with the pelvis fixed. Figure 5.2 displays results for posing the spine.

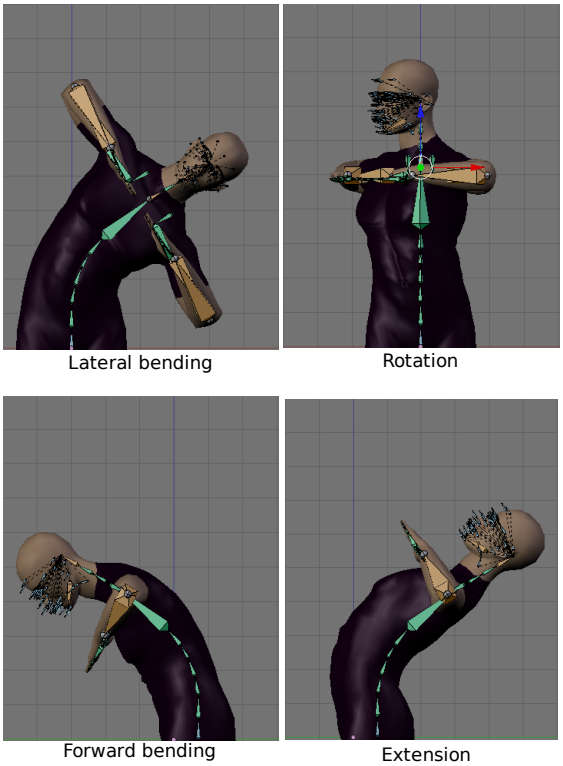

Figure 5.2: Posing of the spine.

5.2.2 The Neck

The neck has a neutral position and can perform: left and right rotation; extension; flexion; left and right lateral bending [22]. Figure 5.3 displays results for posing the neck.

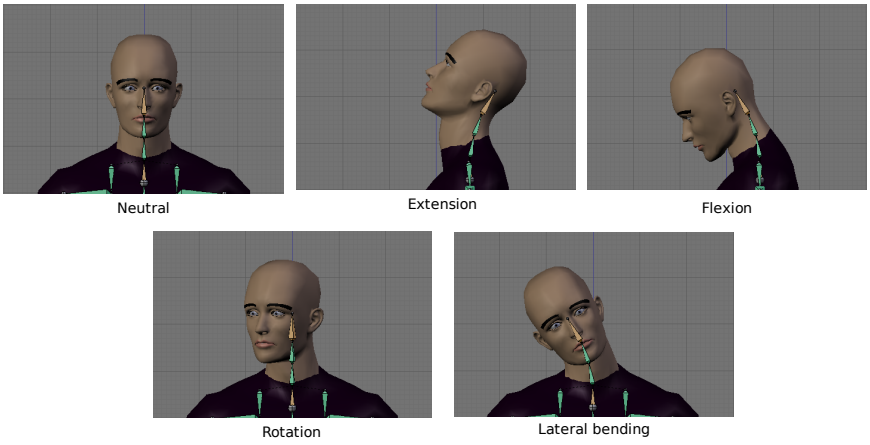

Figure 5.3: Posing of the neck.

5.2.3 The Shoulder

Cave and Roberts define the neutral position of the shoulder such that the arm hangs to the side and the elbow is flexed by 90° to have the forearm pointing forward [22]. The shoulder is capable of poses that include: extension; flexion; abduction; external and internal rotation in abduction; external and internal rotation in neutral; and elevation. Figure 5.4 displays posing results for the right shoulder by using the joint rotational limits from Boon [19].

Experimentation showed that the shoulder required greater rotational DOFs than that initially adopted from Boon. This is mainly due to the complexity of the shoulder joint as discussed in Section 4.3.1 and Blender's skeletal system having a rotational local coordinate system for its bones that swaps the roles of its axes. For example, a rotation of 90° about the Y-axis such as an internal or external rotation in abduction swaps the roles of the X- and Z-axes. Thus if the Z-axis had greater rotational freedom than the X-axis, once the roles are swapped, rotation that was initially intended to be about the Z-axis will be limited. To overcome this limitation and allow predictable posing, we used maximum values of rotational limits where lesser values prohibited certain movements, as mentioned in Section 4.3.1. Using maximum values do have a downside, as it allows for some physically implausible poses.

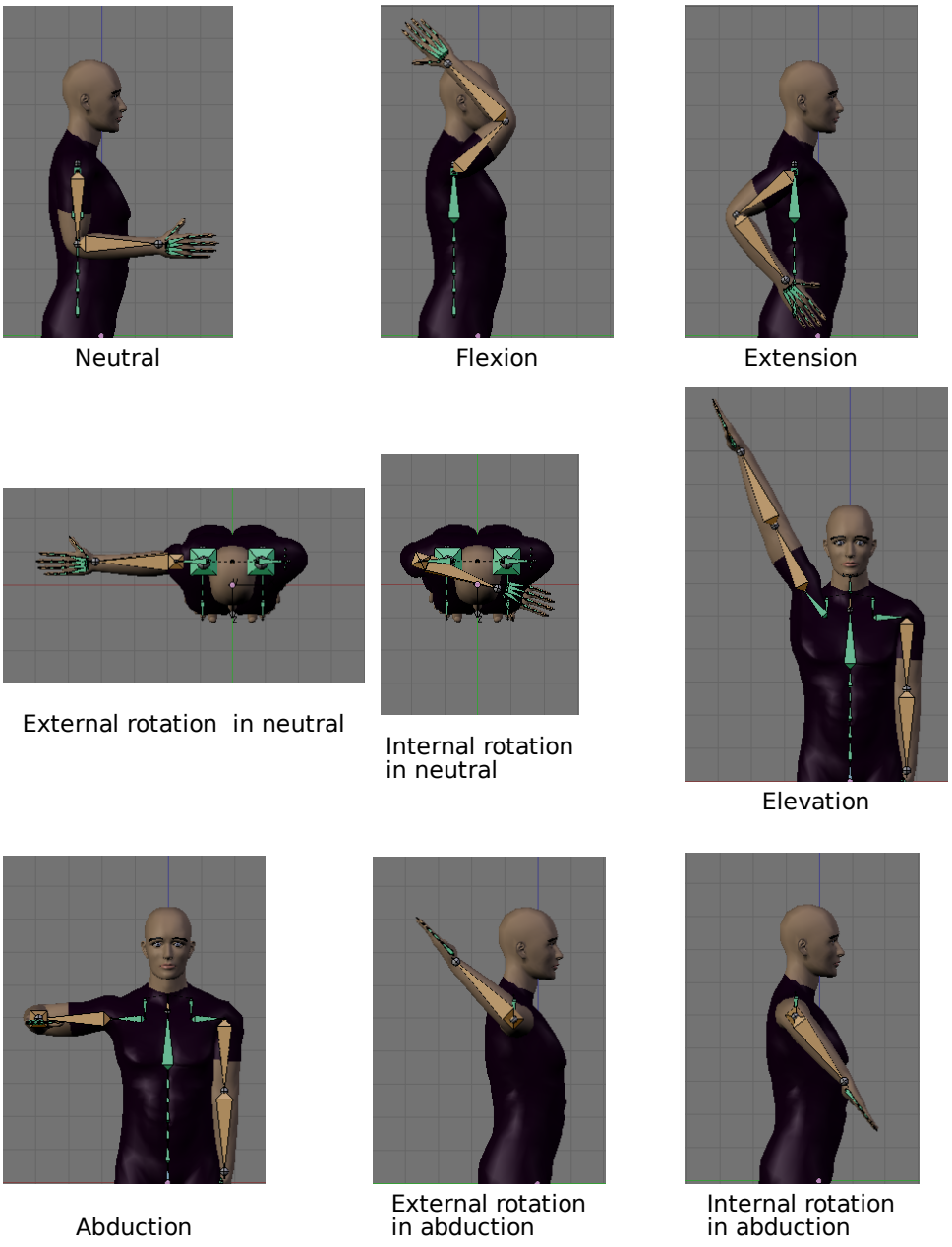

Figure 5.4: Posing of the shoulder.

5.2.4 The Elbow

The elbow's neutral position is with the forearm extended down the side of the body and is capable of poses that include: flexion; hyperextension; supination; and pronation [22]. Figure 5.5 displays results for the elbow except for hyperextension which is not allowed by the joint rotational limits from Boon [19]. The only problem we experienced with the elbow is that it suffers from the limitations of SSD (see Section 2.3.4).

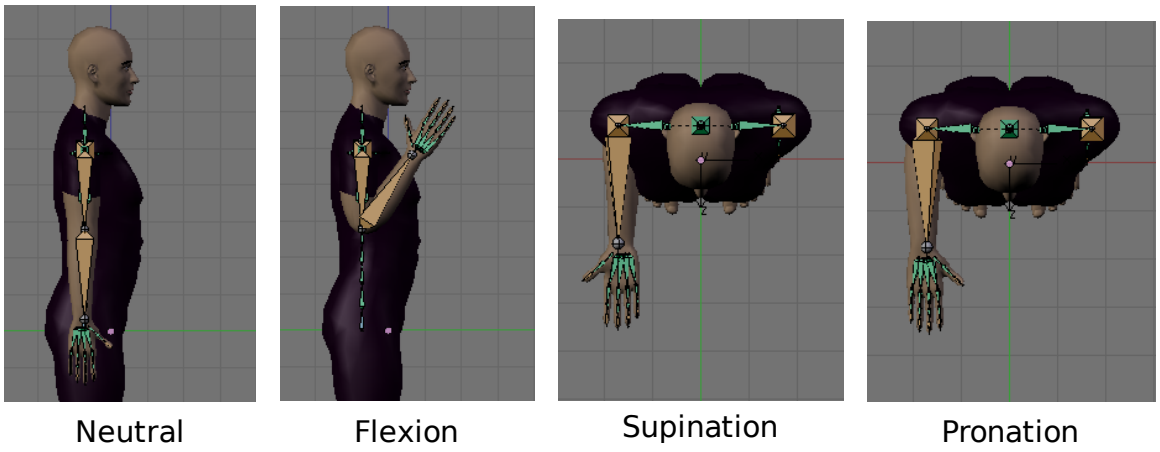

Figure 5.5: Posing of the elbow.

5.2.5 The Wrist

The wrist's neutral position is with the hand in line with the forearm and the palm facing down [22]. Figure 5.6 displays results for the wrist which is capable of poses that include: dorsiflexion (extension); palmar flexion; ulnar deviation; and radial deviation [22].

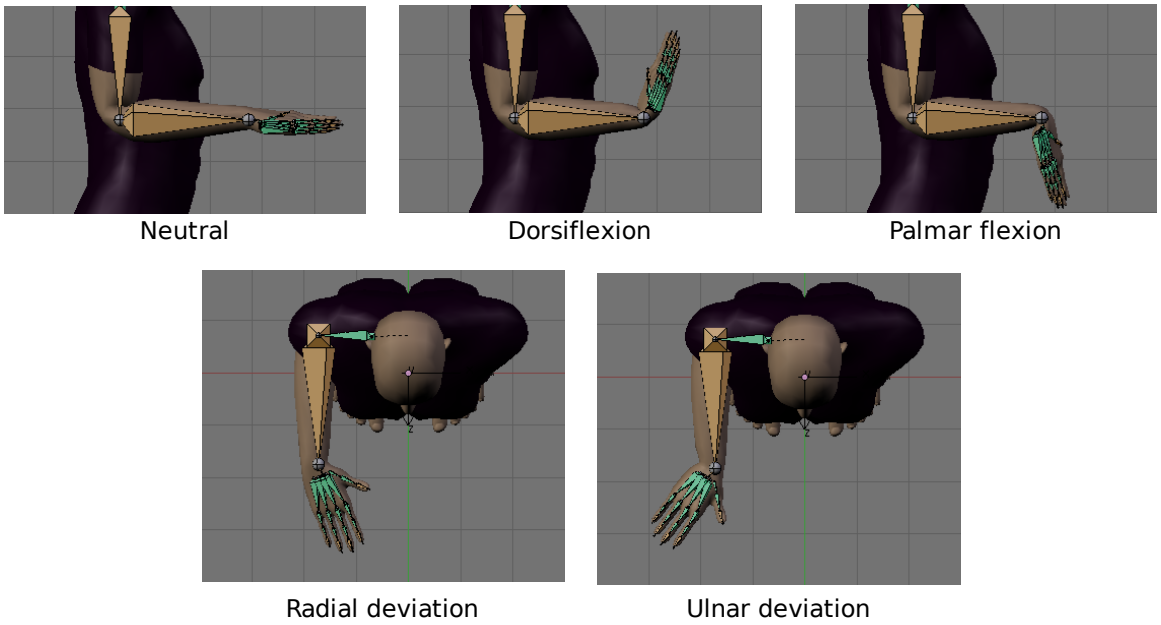

Figure 5.6: Posing of the wrist.

5.2.6 Animation

The ability to easily create, reuse and share animation data was the focus with regard to experimentation on animation. As mentioned in Section 4.1.2, Blender has features that enables one to easily create keyframe animations which is vitally important for SLV (see Section 3.2.2.1). All four VH models we modelled in Section 5.1.1 have more or less the same number of vertices, edges and faces. Also, all four models had quality parameterisations that can be animated with the implemented animation controllers that make use of the same keyframe animations that were created with Man. Figure 5.7 displays results of animations of all four VH models sharing the same animation data which were rendered at between 120 – 230 fps that satisfies real-time requirements.

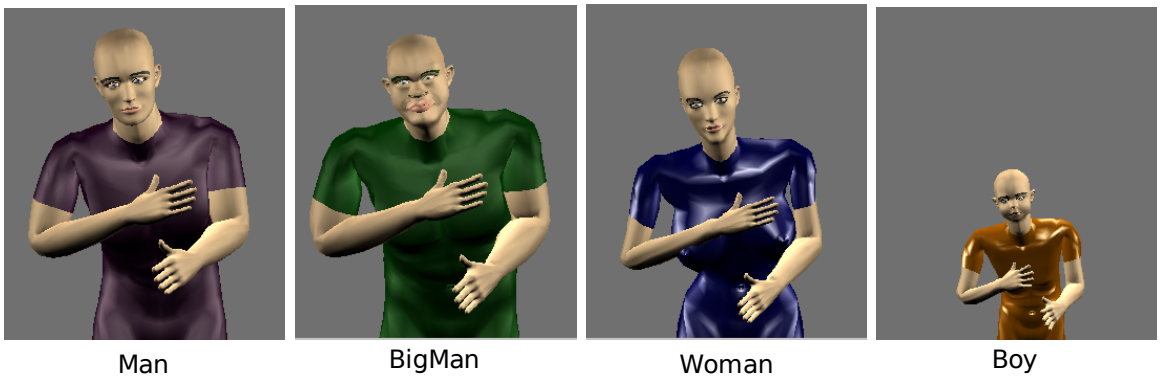

Figure 5.7: Sharing animation data created with Man between all four VH models.

5.3 Hand Posing and Animation

The hands are articulately more complex than the body parts we experimented with in the previous section and require special attention, as was given by related work in Section 2.11. The hand models we modelled in Section 5.1.2 were used to evaluate if we obtained quality parameterisations and if the joint rotational limits prevented the posing of physically implausible poses. Also, with regard to animation, we desired to know if it was possible to easily share animation resources between these complex articulated figures.

A keyframe animation database that represents the SASL finger spelled alphabet was created with Norm Hands. The finger spelling for the acronym SASL using Norm Hands can be seen in Figure 5.8. While creating the finger spelled alphabet, we experienced a similar problem as with the shoulder due to the complexity of the carpometacarpal joint

(thumb1). The thumb1 joint showed that it required greater rotational DOF than that used by Albrecht [6], which we increased to allow complete anteposition and opposition of the thumb to a finger. However by increasing the rotational limits, we allow unrealistic radial and palmar abductions. The finger spelling alphabet animation database was used with an interactive animation controller to interactively change between hand shapes and create finger spelling animations. The sharing of the animations between Norm Hands, Short Hands and Long Hands is displayed in Figure 5.9, which shows the SASL finger spelled letters 'S' and 'V'. Due to segment length in the different hand models, one may not always obtain the desired result when sharing animation data.

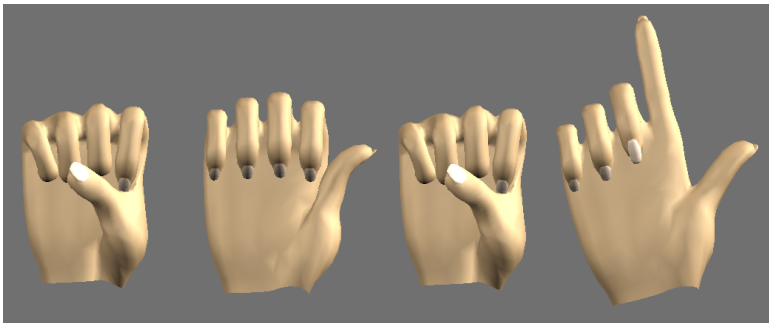

Figure 5.8: Finger spelling of "SASL".

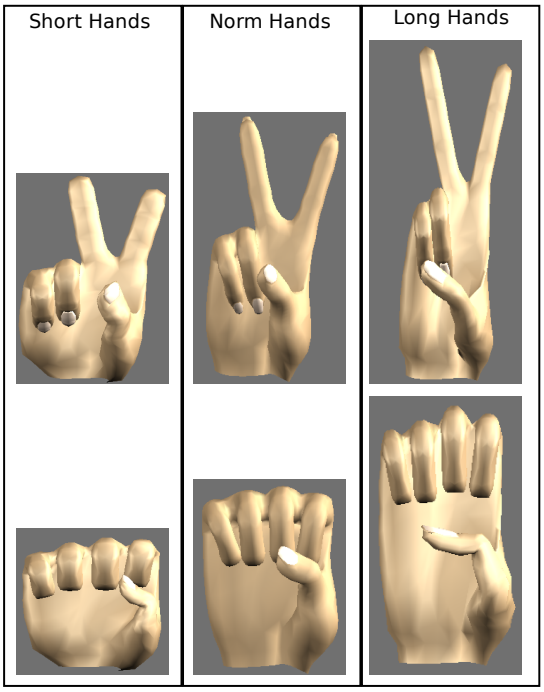

Figure 5.9: Hand shape posing and sharing animation data.

To improve visual results, we employed Blender's multi-resolution mesh features. We only took the right hand of Norm Hands and performed Catmull-Clark subdivision. In Table 5.2 we display the model data on 3 levels of subdivision and the frame rates achieved. Figure 5.10 shows that there is a significant increase in quality and slight differences in the palm area between the 3 levels of subdivision.

Level	Vertices	Edges	Faces	frame rate in fps
1	1388	2806	1419	330
2	5613	11205	593	269
3	22411	44782	22372	94

Table 5.2: Catmull-Clark subdivision hand models geometric details and frame rates.

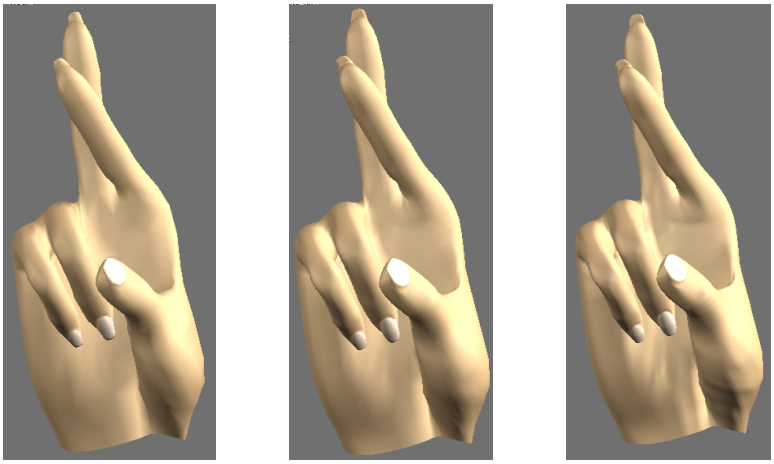

Figure 5.10: Catmull-Clark subdivision hand models at 3 different levels of subdivision.

5.4 Face Posing and Animation

Experimentation that focussed on facial posing and animation was also performed with the same reasoning as for the hands in the previous section. The facial bones we added in Section 4.3.3 were given no limitations to allow a designer complete freedom to design facial expressions. Facial expressions from the Thibologa Sign Language Institution booklet [89] was used to design facial expressions with the Man model. The results for the kind, smile, surprise and thoughtful facial expressions can be seen in Figure 5.11 where we share animation data between the four models. The same frame rates were achieved as for body animation since we used the same full body VH models.

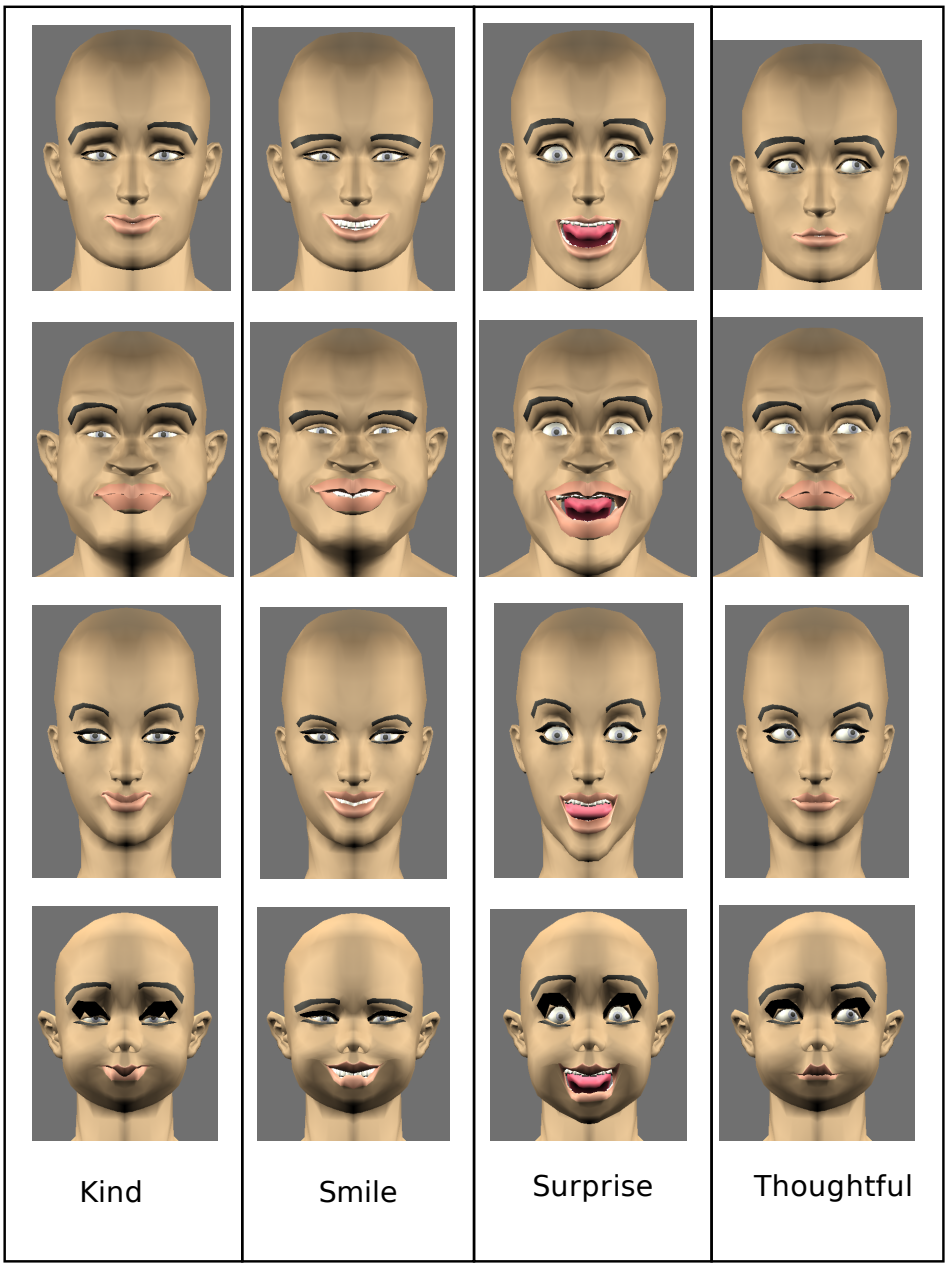

Figure 5.11: Facial animation data created with Man and shared between the other models.

To improve visual results for facial expressions, a similar approach to that taken with the hands was employed, using Blender's multi-resolution mesh features. Catmull-Clark subdivision was performed on the whole skin model and not just the hands. Since we do not apply subdivision to the eyes, teeth and tongue models, we provide the geometric properties of the skin, eyes, teeth and tongue models combined as given by Blender in Table 5.3 for 3 levels of subdivision as well as the achieved frame rates. Figure 5.12

displays facial posing and animation results for the BigMan VH model on 3 levels of subdivision. There is little visual difference between level 2 and level 3 of the subdivided BigMan VH and although the frame rate of level 3 was below the 15 fps requirement for real-time visualisation, it is still acceptable.

Level	Vertices	Faces	frame rate in fps
1	10838	10806	170
2	38639	38190	45
3	149009	148167	11

Table 5.3: Geometric details of Catmull-Clark subdivision of the skin at 3 levels combined with the eyes, teeth and tongue as well as the achieved frame rates.

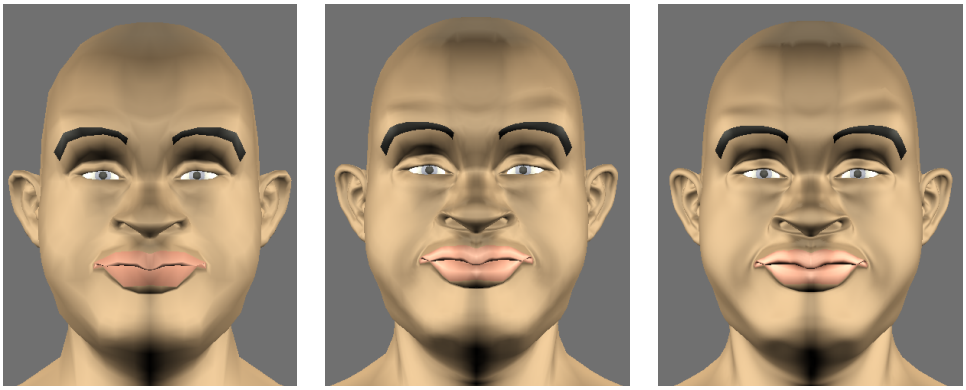

Figure 5.12: Catmull-Clark subdivision of the skin at 3 levels combined with the eyes, teeth and tongue.

5.5 Summary

This chapter presented experiments that we performed to evaluate our methodology and framework. Four VH models were created and used for evaluation throughout the chapter. We also modelled three pairs of hands which are complex articulated figures to evaluate the quality of the parameterisations and the ability to share animation data with consistent results. Catmull-Clark subdivision was performed on the hand models to further improve quality achieving acceptable frame rates. The same experimentation was performed with faces as with the modelled hands to evaluate the quality of the parameterisations of our models and the sharing of facial animation data between models.

Chapter 6

Conclusions

In this thesis we had two goals. The first was to model and animate VHs of adequate quality to effectively visualise sign languages. The second was to accelerate progress in perceptive interfaces for virtual humans by developing a methodology and open framework that can result in community resources. Our goals have been achieved as we have demonstrated in the previous chapter that our methodology and open framework deliver excellent results. In this chapter we conclude by discussing our contributions to VH modelling and animation, as well as SLV. We also discuss the advantages and disadvantages of our methodology and framework and provide some recommendations for future research.

6.1 Contributions

The research question we posed in Chapter 1 has been successfully answered as we have shown in the previous chapter. Our contributions to virtual human modelling and animation is a methodology and open framework to model and animate VHs with the necessary features that are of adequate quality to perform sign language visualisation. Overall, our methodology and framework includes:

- Open technologies with their associated interfaces that simplify the modelling and animation of seamless high quality VHs.
- A generic skeleton based on an H-Anim LoA 2 skeleton with joint rotational limits and flexible hands for body and hand animation.
- Additional facial bones to our generic skeleton that are based on MPEG-4 FDP facial feature points for facial animation.

- A semi-automatic model parameterisation process to parameterise skin, eyes, teeth and tongue models with our generic skeleton.
- Interactive and procedural animation controllers for virtual human animation and visualising sign language from SBML in real-time.

6.2 Advantages

Our methodology and open framework has the following advantages compared to approaches we reviewed in Chapters 2 and 3 that include:

- A straightforward approach to modelling and animating VHs of high quality with the necessary features that can perform body, hand and facial animation for sign language visualisation in real-time.
- We employ open technologies with large user communities that are actively developed that enable us to easily share resources which can be body, hand or facial animations.
- The interfaces provided by Blender enable us to easily create the above mentioned animation resources and we therefore avoid the difficulties experienced by researchers in Chapter 3.

6.3 Disadvantages

As no methodology or framework is perfect, we provide a list of disadvantages to our methodology and framework that include:

- The open technologies we employ require some user training to model and animate VHs.
- Our methodology requires some user interaction and is not completely automatic as the teeth and eye models require manual parameterisation.
- The procedural animation controller we implemented is only capable of performing sign language visualisation from a complete keyframe animation that represents a whole sign language gesture.

6.4 Recommendations

We are satisfied with our results and believe that we have contributed greatly to the fields of VH modelling and animation, as well as sign language visualisation. There are two sets of recommendations that we make that relate to VH modelling and animation and to that of sign language visualisation.

6.4.1 Virtual Human Modelling and Animation

At the time of writing, we employed an older version of MakeHuman as the latest version does not yet have the jaw open target that we require during modelling before parameterisation of VH models. The latest version is near completion and includes several improvements and new features that can be used in future work. Also, the latest version includes a Python API which makes automatic modelling, exportation and importation of models into Blender possible. Another improvement that can be made is automatic skeleton fitting and parameterisation that takes into consideration the facial bones. Clothes can also be created or modelled within Blender to create fully clothed VHs.

6.4.2 Sign Language Visualisation

Further improvements can be made to the procedural animation controller, as we can only perform sign language visualisation from a complete keyframe animation that represents a whole sign language gesture. A procedural animation controller can be developed that uses more BL ActionActuators in Blender to separate body, hand and facial animation control. By doing so, phoneme based SLV will be possible. Also, SWML instead of SBML can be used although SBML is much more compact than SWML.

6.5 Summary

This research has significantly advanced the field of perceptive interfaces for virtual humans. Previous work is divided between those who concentrated on developing separate systems for hand, body and facial animation and then attempted to integrate them and those who focussed on maximising realism at the expense of real-time performance. This research shows that it is possible to assemble state of the art 3D visualisation systems that seamlessly integrate hand, body and facial animation and that this can be done in

real-time. While these systems have a wide range of applications, they are particularly well suited for the real-time visualisation of signed languages.

Appendix A

Blender Game Engine Logic

A.1 Module GameLogic

This is the documentation for the GameLogic Module which was taken from and available at http://www.blender.org/ documentation/pydoc_gameengine/PyDoc-Gameengine-2.34/ GameLogic-module.html. There are only three importable modules in the game engine:

- GameLogic
- GameKeys
- Rasterizer

All the other modules are accessible through the methods in GameLogic.
Examples:

```
# To get a controller:
import GameLogic
co = GameLogic.getCurrentController()
# To get the game object associated with this controller:
obj = co.getOwner()

# To get a sensor linked to this controller.

# ‘‘sensorname" is the name of the sensor as defined in the Blender
interface.
# +------------------------------+ +-------------+
```

```
# + Sensor ‘‘sensorname" +--+ Python +
# +-----------------------------+ +-------------+
sens = co.getSensor(‘‘sensorname")
# To get a list of all sensors:
sensors = co.getSensors()
```

A.2 Sensors

- KX_NetworkMessageSensor
- KX_RaySensor
- KX_MouseFocusSensor
- KX_NearSensor
- KX_RadarSensor
- KX_TouchSensor
- SCA_KeyboardSensor
- SCA_MouseSensor
- SCA_PropertySensor
- SCA_RandomSensor

A.3 Python Controller

```
To get an actuator attached to the controller:
+--------+ +--------------------------------------+
+ Python +--+ Actuator ‘‘actuatorname" +
+--------+ +--------------------------------------+

actuator = co.getActuator(‘‘actuatorname")
GameLogic.addActiveActuator(actuator, True)
```

A.4 Actuators

- BL_ActionActuator
- KX_CameraActuator
- KX_CDActuator
- KX_ConstraintActuator
- KX_GameActuator
- KX_IpoActuator
- KX_NetworkMessageActuator
- KX_ObjectActuator
- KX_SCA_AddObjectActuator
- KX_SCA_EndObjectActuator
- KX_SCA_ReplaceMeshActuator
- KX_SceneActuator
- KX_SoundActuator
- KX_TrackToActuator
- KX_VisibilityActuator
- SCA_PropertyActuator
- SCA_RandomActuator

Bibliography

[1] ISO/IEC 14496-2:2004, Information technology – Coding of audio-visual objects – Part 2: Visual.

[2] ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004 – Virtual reality modelling language (VRML). [Online] Available at http://www.web3d.org/ x3d/ specifications/ vrml/ ISO-IEC-14772-VRML97/ [Accessed 1 November 2008].

[3] ISO/IEC FCD 19774:200x, Information technology – Computer graphics and image processing — Humanoid animation (H-Anim). [Online] Available at http://h-anim.org/Specifications/H-Anim200x/ISO_IEC_FCD_19774/ [Accessed 1 November 2008].

[4] ISO/IEC FDIS 19775-1.2:2008 — X3D Architecture and base components. Edition 2. [Online] Available at http://www.web3d.org/x3d/specifications/ISO-IEC-FDIS-19775-1.2-X3D-AbstractSpecification/ [Accessed 1 November 2008].

[5] D. Aarons. *Aspects of the syntax of American Sign Language.* PhD thesis, Boston University Graduate School, 1994.

[6] I. Albrecht. *Faces and hands: Modeling and animating anatomical and photorealistic models with regard to the communicative competence of virtual humans.* PhD thesis, Universitat des Saarlandes, 2005.

[7] American society for surgery of the hand. 2002. Hand anatomy. Available at: http://www.assh.org/Content/NavigationMenu/PatientsPublic/HandAnatomy/ Hand_Anatomy.pdf, [Accessed 1 November 2008].

[8] Autodesk. Autodesk Maya. [Online] Available at http://usa.autodesk.com/ sadsk/servlet/index?id=7635018&siteID=123112 [Accessed 1 November 2008].

[9] C. Babski and D. Thalmann. A seamless shape for hanim compliant bodies. In *VRML '99: Proceedings of the fourth symposium on virtual reality modeling language*, pages 21–28, New York, NY, USA, 1999. ACM.

[10] N. I. Badler, M. S. Palmer, and R. Bindiganavale. Animation control for real-time virtual humans. *Communications of the ACM*, 42(8):64–73, 1999.

[11] N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, USA, 1993.

[12] J. A. Bangham, S. J. Cox, M. Lincoln, I. Marshall, M. Tutt, and M. Wells. Signing for the deaf using virtual humans. *Speech and language processing for disabled and elderly people (Ref. No. 2000/025), IEE seminar on*, page 4, 2000.

[13] I. Baran and J. Popović. Automatic rigging and animation of 3D characters. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 72, New York, NY, USA, 2007. ACM.

[14] D. Barker. Computer facial animation for sign language visualization. Master's thesis, University of Stellenbosch, 2005.

[15] M. Bastioni. New mesh model in MakeHuman 0.8. 2005. [Online] Available at http://www.dedalo-3d.com/docs/2005-12-01-new-model.pdf [Accessed 1 November 2008].

[16] Blender Foundation. Blender. [Online] Available at http://www.blender.org [Accessed 1 November 2008].

[17] Blender (software) Wikipedia. [Online] Available at http://en.wikipedia.org/ wiki/ blender_(software) [Accessed 1 November 2008].

[18] J. F. Blinn. A generalization of algebraic surface drawing. *ACM transactions on graphics (TOG)*, 1(3):235–256, 1982.

[19] D. C. Boone. Normal range of motion of joints in male subjects. *The journal of bone & joint surgery*, 61(5):756–759, 1979.

[20] Caligari Corporation. trueSpace 7. [Online] Available at http:// www.caligari.com/ Products/ trueSpace/ tS75/brochure/ intro.asp?Cate=BIntro [Accessed 1 November 2008].

[21] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer aided design*, 10(6):350–355, 1978.

[22] E. F. Cave and S. M. Roberts. A method for measuring and recording joint function. *The journal of bone & joint surgery*, 18(2):455–465, 1936.

[23] R. Cole. Accelerating progress in perceptive animated interfaces and virtual humans. Final report of NSF-funded workshop, University of Colorado Boulder, 2004.

[24] S. Cox, M. Lincoln, J. Tryggvason, M. Nakisa, M. Wells, M. Tutt, and S. Abbott. Tessa, a system to aid communication with deaf people. *Proceedings of the fifth international ACM conference on assistive technologies*, pages 205–212, 2002.

[25] Cyberware. Whole body color 3D scanner (Model WBX). [Online] Available at http://www.cyberware.com/products/scanners/wbx.html [Accessed 1 November 2008].

[26] R. Dale. Introduction to character animation. 2006. [Online] Available at http://wiki.blender.org/uploads/6/66/Introduction_to_Character_ Animation_19_Sept_2006.pdf [Accessed 1 November 2008].

[27] DEAFSA. Deaf federation of South Africa. [Online] Available at http://www.deafsa.co.za [Accessed 4 June 2007].

[28] Z. Dong, W. Chen, H. Bao, H. Zhang, and Q. Peng. Real-time voxelization for complex polygonal models. In *Proceedings of the computer graphics and applications, 12th Pacific conference*, pages 43–50. IEEE Computer Society Washington, DC, USA, 2004.

[29] D. Doo. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In *Proceedings of interactive techniques in computer aided design*, pages 157–165, 1978.

[30] P. Ekman, W. V. Friesen, and J. C. Hager. *Facial Action Coding System (FACS).* A Human Face, Salt Lake City, 2002.

[31] R. Elliott, J. R. W. Glauert, and J. R. Kennaway. A framework for non-manual gestures in a synthetic signing system. *2nd Cambridge workshop on universal access and assistive technology, CWUAAT*, pages 127–136, 2004.

[32] S. Fang and R. Srinivasan. Volumetric CSG–A model-based volume visualization approach. In *Proceedings of the 6th international conference in central europe on computer graphics and visualization*, pages 88–95, 1998.

[33] C. M. Flood. *How do deaf and hard of hearing students experience learning to write using signwriting, a way to read and write signs?* PhD thesis, Educational Lingusitics, University of New Mexico, 2002.

[34] J. Fourie. The design of a generic signing avatar animation system. Master's thesis, University of Stellenbosch, 2006.

[35] J. Francik and P. Fabian. Animating sign language in the real time. *20th IASTED conference on applied informatics, Innsbruck, Austria*, 2002.

[36] F. Godenschweger and T. Strothotte. Modeling and generating sign language as animated line drawings. In *Assets '98: Proceedings of the third international ACM conference on assistive technologies*, pages 78–84, New York, NY, USA, 1998. ACM.

[37] R. G. Gordon. Ethnologue: Languages of the World, Fifteenth edition Dallas, Tex.: SIL International. Online version: http://www.ethnologue.com/, 2005.

[38] A. B. Grieve-Smith. SignSynth: A prototype articulatory sign-language synthesis application. [Online] Available at http://www.unm.edu/ grvsmth/signsynth/ [Accessed 1 November 2008].

[39] A. B. Grieve-Smith. SignSynth: A sign language synthesis application using Web3D and Perl. *Gesture workshop, London*, pages 134–145, 2001.

[40] B. Guenter and M. Gavriliu. Exact procedural CSG modeling for real time graphics. 2007. Available at http://research.microsoft.com/ bguenter/docs/template.pdf.

[41] D. Hearn and M.P. Baker. *Computer Graphics: C Version.* Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1996.

[42] J. A. Holt. Stanford achievement test - 8th Edition for deaf and hard of hearing students: Reading comprehension subgroup results. *American annals of the deaf*, 1993.

[43] M. Huenerfauth. *Generating American Sign Language classifier predicates for English-to-ASL machine translation.* PhD thesis, Computer and Information Science, University of Pennsylvania, 2006.

[44] P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Aubel, and D. Thalmann. Real-time animation of realistic virtual humans. *IEEE computer graphics and applications*, 18(5):42–57, /1998.

[45] P. Kalra, A. Mangili, N. M. Thalmann, and D. Thalmann. Simulation of facial muscle actions based on rational free form deformations. In *Computer Graphics Forum*, volume 11, pages 59–69. Blackwell Synergy, 1992.

[46] K. Karpouzis, G. Caridakis, S. E. Fotinea, and E. Efthimiou. Educational resources and implementation of a Greek Sign Language synthesis architecture. *Proceedings of the first international workshop on Web3D technologies in learning, education and training (LET-WEB3D)*, pages 8–15, 2004.

[47] L. Kavan and J. Žára. Spherical blend skinning: a real-time deformation of articulated models. In *I3D '05: Proceedings of the 2005 symposium on interactive 3D graphics and games*, pages 9–16, New York, NY, USA, 2005. ACM.

[48] R. Kennaway. Synthetic animation of deaf signing gestures. *Fourth international workshop on gesture and sign language based human–computer interaction, Springer series lecture notes in artificial intelligence. Springer Verlag*, 2001.

[49] R. Kennaway. Experience with, and requirements for, a gesture description language for synthetic animation. *Fifth international workshop on gesture and sign language based human-computer interaction*, 2003.

[50] S. Krapez and F. Solina. Synthesis of the sign language of the deaf from the sign video clips. *Elektrotehniski vestnik*, 66(4-5):260–265, 1999.

[51] T. Kuroda, K. Sato, and K. Chihara. S-TEL: An avatar based sign language telecommunication system. *International journal of virtual reality*, 3(4):21–27, 1998.

[52] W. Lee and N. Magnenat-Thalmann. Virtual body morphing. In *The fourteenth conference on computer animation*, pages 158–166, 2001.

[53] Y. Lee, D. Terzopoulos, and K. Walters. Realistic modelling for facial animation. In *Proceedings of the 22nd annual conference on computer graphics and interactive techniques*, pages 55–62. ACM New York, NY, USA, 1995.

[54] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th*

annual conference on computer graphics and interactive techniques, pages 165–172. ACM Press, 2000.

[55] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[56] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on graphics interface '88*, pages 26–33, Toronto, Ont., Canada, Canada, 1988. Canadian Information Processing Society.

[57] MakeHuman Team. MakeHuman. [Online] Available at http://www.dedalo-3d.com/index.php [Accessed 1 November 2008].

[58] M. A. Mandel. ASCII-Stokoe notation: A computer-writable transliteration system for Stokoe notation of American Sign Language. [Online] Available at http://www.speakeasy.org/m̃amandel/ASCII-Stokoe.html [Accessed 1 November 2008].

[59] P. W. McClure, L. A. Michener, B. J. Sennett, and A. R. Karduna. Direct 3-dimensional measurement of scapular kinematics during dynamic movements in vivo. *Journal of shoulder and elbow surgery*, 10(3):269–277, 2001.

[60] J. McDonald, J. Toro, K. Alkoby, A. Berthiaume, R. Carter, P. Chomwong, J. Christopher, M. J. Davidson, J. Furst, B. Konie, G. Lancaster, L. Roychoudhuri, E. Sedgwick, N. Tomuro, and R. Wolfe. An improved articulated model of the human hand. In *The Visual Computer*, volume 17(3), pages 158–166. Springer, 2001.

[61] L. Moccozet, F. Dellas, N. Magnenat-Thalmann, S. Biasotti, M. Mortara, B. Falcidieno, P. Min, and R. Veltkamp. Animatable human body model reconstruction from 3D scan data using templates. In *Proceedings of workshop on modelling and motion capture techniques for virtual environments, CAPTECH*, pages 73–79, 2004.

[62] C. Morkel and S. Bangay. Procedural modeling facilities for hierarchical object generation. In *Afrigaph '06: Proceedings of the 4th international conference on computer graphics, virtual reality, visualisation and interaction in Africa*, pages 145–154, New York, NY, USA, 2006. ACM Press.

[63] L. J. Muir and I. E. G. Richardson. Perception of sign language and its application to visual communications for deaf people. *Journal of deaf studies and deaf education*, 10(4):390–401, 2005.

[64] V. Ng-Thow-Hing. *Anatomically-based models for physical and geometric reconstruction of humans and other animals*. PhD thesis, Graduate Department of Computer Science, University of Toronto, 2001.

[65] M. Papadogiorgaki, N. Grammalidis, L. Makris, and M. G. Strintzis. Gesture synthesis from sign language notation using MPEG-4 humanoid animation parameters and inverse kinematics. In *Second international conference on intelligent environments 2006*, Athens, Greece, July 2006.

[66] F. I. Parke. Computer generated animation of faces. In *ACM'72: Proceedings of the ACM annual conference*, pages 451–457, New York, NY, USA, 1972. ACM.

[67] S. Pasquariello and C. Pelachaud. Greta: A simple facial animation engine. In *Sixth online world conference on soft computing in industrial applications. Session on soft computing for intelligent 3D agents*, 2001.

[68] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin. Synthesizing realistic facial expressions from photographs. In *International conference on computer graphics and interactive techniques*. ACM Press New York, NY, USA, 2006.

[69] R. Plankers, P. Fua, and N. D'Apuzzo. Automated body modelling from video sequences. In *Proceedings of the IEEE international workshop on modelling people*, page 45. IEEE Computer Society Washington, DC, USA, 1999.

[70] Python Software Foundation. Python. [Online] Available at http://www.python.org [Accessed 1 November 2008].

[71] T. Rhee, U. Neumann, and J. P. Lewis. Human hand modeling from surface anatomy. In *I3D '06: Proceedings of the 2006 symposium on interactive 3D graphics and games*, pages 27–34, New York, NY, USA, 2006. ACM.

[72] H. Rijpkema and M. Girard. Computer animation of knowledge-based human grasping. *ACM SIGGRAPH computer graphics*, 25(4):339–348, 1991.

[73] SAE International. Civilian American and European Surface Anthropometry Resource Project – CEASAR. [Online] Available at http://store.sae.org/caesar/ [Accessed 1 November 2008].

[74] SASL Project. Integration of signed and verbal communication: South African Sign Language recognition and animation, http://www.cs.uwc.ac.za/, http://www.cs.uwc.ac.za/.

[75] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics*, 20(4):151–160, 1986.

[76] H. Seo, F. Cordier, L. Philippon, and N. Magnenat-Thalmann. Interactive modelling of MPEG-4 deformable human body modelsmpeg-4 deformable human body models. In *DEFORM '00/AVATARS '00: Proceedings of the IFIP TC5/WG5.10 DEFORM'2000 workshop and AVATARS'2000 workshop on deformable avatars*, pages 120–131, Deventer, The Netherlands, The Netherlands, 2001. Kluwer, B.V.

[77] H. Seo and N. Magnenat-Thalmann. An automatic modeling of human bodies from sizing parameters. In *I3D '03: Proceedings of the 2003 symposium on interactive 3D graphics*, pages 19–26, New York, NY, USA, 2003. ACM.

[78] Siemens PLM Software. Jack human modeling and simulation: Virtual people, virtual places, real solutions. [Online] Available at http://www.plm.automation.siemens.com/en_gb/Images/tx 20jack 20fs 20W 203_tcm642-4952.pdf [Accessed 1 November 2008].

[79] K. Singh and E. Kokkevis. Skinning characters using surface-oriented free-form deformation. In *Graphics Interface*, pages 35–42, 2000.

[80] Smith Mirco Software. Poser: Discover the art of 3D figure design. [Online] Available at http://my.smithmicro.com/mac/poser/index.html [Accessed 1 November 2008].

[81] D. L. Speers. *Representation of American Sign Language for machine translation.* PhD thesis, Faculty of the Graduate School of Arts and Sciences, Georgetown University, 2001.

[82] J. Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH '98: Proceedings of the 25th annual conference on computer graphics and interactive techniques*, pages 395–404, New York, NY, USA, 1998. ACM.

[83] C. Steinback. CARPI: A database for computer generated fingerspelling. In *Twelfth annual technology and persons with disabilities conference.* Center on disabilities, 2001.

[84] W. C. Stokoe. Sign language structure: An outline of the visual communication systems of the American deaf. *Journal of deaf studies and deaf education*, 10(1):3–37, 2005.

[85] D. J. Sturman. A brief history of motion capture for computer character animation. *SIGGRAPH 94, Character motion systems, Course notes*, 1994.

[86] V. Sutton. SignWriting site. [Online] Available at http://www.signwriting.org [Accessed 1 November 2008].

[87] SWML. Sign writing markup language. [Online] Available at http://sign-net.ucpel.tche.br/swml/ [Accessed 1 November 2008].

[88] D. Thalmann, J. Shen, and E. Chauvineau. Fast realistic human body deformations for animation and vr applications. In *CGI '96: Proceedings of the 1996 conference on computer graphics international*, page 166, Washington, DC, USA, 1996. IEEE Computer Society.

[89] Thibologa Sign Language Institution. South African Deaf People and their Language. Booklet and DVD, 2007.

[90] University of Hamburg. HamNoSys 3.0. [Online] Available at http://www.sign-lang.uni-hamburg.de/Projekte/HamNoSys/HamNoSysErklaerungen/englisch/Contents.html [Accessed 1 November 2008].

[91] University of Hamburg. HamNoSys 4.0. [Online] Available at http://www.sign-lang.uni-hamburg.de/Projekte/HamNoSys/HNS4.0/HNS4.0eng/Contents.html [Accessed 1 November 2008].

[92] University of Hamburg. Sign language notation system. [Online] Available at http://www.sign-lang.uni-hamburg.de/Projects/HamNoSys.html [Accessed 1 November 2008].

[93] U.S. National Library of Medicine. The visible human project. [Online] Available at http://www.nlm.nih.gov/research/visible/visible_human.html [Accessed 1 November 2008].

[94] L. van Zijl. The South African Sign Language machine translation project. In *Proceedings of the 8th international ACM SIGACCESS conference on computers and accessibility (ASSETS06)*, pages 233–234, Portland, USA, October 2006.

[95] L. van Zijl and J. Fourie. Design and development of a generic signing avatar. In *Proceedings of graphics and visualization in engineering*, pages 95–100, Florida, USA, January 2007.

[96] L. van Zijl and L. Raitt. Implementation experience with collision avoidance in signing avatars. In *AFRIGRAPH '04: Proceedings of the 3rd international conference on computer graphics, virtual reality, visualisation and interaction in Africa*, pages 55–59, New York, NY, USA, 2004. ACM.

[97] T. Vetter and V. Blanz. A morphable model for the synthesis of 3D faces. In *Proceedings of the ACM SIGGRAPH conference on computer graphics*, pages 187–194, 1999.

[98] H. Wan, Y. Luo, S. Gao, and Q. Peng. Realistic virtual hand modeling with applications for virtual grasping. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on virtual reality continuum and its applications in industry*, pages 81–87, New York, NY, USA, 2004. ACM.

[99] Q. Wang and S. Ressler. A tool kit to generate 3D animated CAESAR bodies. *SAE transactions*, 114(7):835–842, 2005.

[100] K. Waters. A muscle model for animation three-dimensional facial expression. In *Proceedings of the 14th annual conference on computer graphics and interactive techniques*, pages 17–24. ACM New York, NY, USA, 1987.

[101] C. Wei-Min, C. Ching-Wei, Y. Wei-Cheng, C. Shih-Chang, C. Ti-Sheng, and C. Jian-Horng. Vertebral axial rotation measurement method. *Computer methods and programs in biomedicine*, 81(1):8–17, 2006.

[102] C. J. Wideman and E. M. Sims. Signing avatars. In *Technology and persons with disabilities conference*. Center on disabilities, 1998.

[103] A. Wojdeł and L. J. M. Rothkrantz. Parametric generation of facial expressions based on FACS. *Computer graphics forum*, 24:743, 2005.

[104] J. C. Wong, E. J. Holden, N. Lowe, R. Owens, I. Links, and G. Back. Real-time facial expressions in the Auslan tuition system. In *5th IASTED international conference on computer graphics and imaging. http://auslantuition. csse. uwa. edu. au/help. html*, 2003.

[105] M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL Version 1.1.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1997.

[106] S. Yeates, E. J. Holden, and R. Owens. An animated auslan tuition system. *Machine graphics & vision international journal*, 12(2):203–214, 2003.

[107] L. Zhao, K. Kipper, W. Schuler, C. Vogler, N. Badler, and M. Palmer. A machine translation system from English to American Sign Language. *Association for machine translation in the Americas*, 2000.

[108] I. Zwitserlood, M. Verlinden, J. Ros, and S. van der Schoot. Synthetic signing for the deaf: eSIGN. In *Proceedings of the conference and workshop on assistive technologies for vision and hearing impairment, CVHI*, Granada, Spain, June 2004.