Animating Avatars Using Motion Capture Data Extracted from Video Stream Data

Submitted in Partial Fulfilment for a

Bachelor of Science (Honours) Degree

at Rhodes University

by

Melissa Nadine Palmer

12 November 2001

Acknowledgements

My time at Rhodes University has been the most enjoyable and rewarding time of my life. There are many people who have contributed to shaping the person that I am today and to the final completion of this paper. It would be an impossible task to mention each of them individually, however I would like to give thanks to those who have played a considerable part in helping me to achieve the goals and ambitions that I have strived for.

I would firstly like to thank my supervisor, Professor Shaun Bangay, for all of his advice and support, throughout the year. His assistance and advice has been invaluable.

The research and this paper that I have undertaken this year, would not have been possibly without the resources of the Internet. Most of the papers that I have used and gained valuable information from have been obtained from the net. Thank you to all those people who have provided their research freely available and down loadable on the net.

Thanks also goes the members of the Calnet labs who have put up with my numerous funny questions, weird moods and who have, over the year, become truly great friends. Thanks especially goes to Pareen and Lindy for the great year that we have had. I would not have made it through this year without your support and friendship. Thanks for an excellent year guys, you are the bomb! I will never forgot the times we've enjoyed together this year. Just for you guys - "One quite one and nine loud ones!"

During my last four years at Rhodes I have been lucky enough to have been blessed with a caring and supportive group of friends. I am extremely grateful and appreciative to all of them, more than they will ever know. I would like to however take this opportunity to make a special mention of thanks to some of my closest friends Gen "Longbody" James, Kim "Bucket" Taylor, Kath "Gushells" Gush and Tanya "Skinny" Beukus. You guys have been there for me over the last few years. Thanks for all the help, support, friendship and good times that we have all enjoyed. You are truly wonderfully friends and its going to be, oh so, very difficult not to have you all around everyday in the future, but I know there will be many more good times to be had.

I would also like to thank my late "Granny Cleo" (Beryl Coates), and my Granny Jessie (Lilian Duane) I am extremely grateful for all the help and guidance they have given me A big thank you to my younger sister, Bronwyn for being there when I needed her. Thanks Bron.

Last, but definitely not least, my thanks goes to my parents for everything that they have done for me and allowed me to do. Firstly, thank you both for all the proof reading, corrections and feedback that you have supplied me with throughout the duration of this paper. Thank you for giving me the opportunity attend to some of the best schools in South Africa and for the opportunity to attend University, especially for allowing me to complete this final honours year. I would not have survived without your continuous encouragement and support over the years. Thank you especially for the support and encouragement during my years at University and especially this last year.

Abstract

Motion capture provides extremely realistic and real time results. The research and advancements that have been made in the field of motion capture have concentrated on the motion of human characters. Current motion capture techniques require the use of very specialised equipment, sensory markers and tracking devices. It is not always suitable or practical to use such equipment for the capturing of all moving objects. Motion capture for animals and other living creatures poses many complications.

This paper presents a different motion capture technique, that does not require any markers, body suits, or other devices attached to the performer. The only equipment needed is a hand held video camera. This technique will enable motion capture for complex motion of animals and other living creatures. The paper investigates current methods of motion capture and their feasibility in capturing complex animal motion. An efficient method of extracting 3D motion using a single hand held video camera is investigated. A motion capture application is implemented to view video data and extract skeleton and motion data, from the film footage captured with a camera. A model viewer application is created which models and views the data extracted from motion capture files in a three dimensional environment.

This paper provides the reader with a solution to motion capture for animals and other moving objects that have very complex motion, and where it is not possible to attach markers or other equipment to them. Various motion capture techniques and technologies are compared a discussed in conjunction to the technique proposed by this paper.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1 Introduction

1.1 Background

Motion Capture is the process of capturing motion from a real life object and mapping it onto a computer generated object. Most of the research that has been undertaken with motion capture has been for human characters. This is because it is easy to get human actors into a studio to perform for you. Studio motion can provide very detailed and realistic motion. This however is not appropriate for many situations. The studio limits the range of motion that can be obtained from the actor and it is not possible to record the motion of animals and other living creatures. Using a single hand held video camera enables one to record motion of a greater variety and in any location thus enabling virtually unlimited input into motion capture.

1.2 Aim

The aim of this project is to reduce the complexities involved in current technologies used for motion capture. Current technologies require the subject to be in a carefully setup studio and to have magnetic or optical markers placed on them. These methods are not suitable or practical for motion capture of animals and other creatures or objects such as, such as quadrupeds, insects, fish, birds and other living creatures or balls, boats and other such examples. This project investigates a simple method of extracting motion data from video stream sequences captured with a single hand held video camera.

The final goal of this project is to model and animate articulated quadruped avatars in a three dimensional environment. This is achieved through the collection of film footage depicting quadrupeds in their natural environment, where no attempt has been made to specially position the camera or the subject whilst filming. Still frames are extracted from these sequences of video data. Each of the frames are used to annotate the skeletal hierarchy and motion for the sort after quadruped. This information is stored for later use of modelling and viewing of these avatars in a three dimensional environment.

The project addresses the issues of extracting still frames from video sequences. The use of these frames to write and store motion capture information. A method of extracting motion data from motion capture files in order to model and view these avatars is also researched.

1.3 What is Motion Capture?

Motion Capture for computer animation is the process of obtaining the positions, orientations and offsets of certain points on real life characters, while they are in motion, in order to be able to use this information for animating characters in a computer generated 3D scenes. Motion Capture usually involves the use of markers that are placed on the actor. The actor then makes movements, which can then be filmed from several different angles, this will allow a 3D track of all the markers to be generated.

However, as motion capture can be used in numerous application areas such as entertainment, simulation systems and virtual reality, its purpose is not simply to duplicate the movements of an actor. The use and application of motion capture, if simplified and easily applicable to all types of motion and life, is virtually endless.

1.4 The History of Motion Capture?

The use of motion capture for computer animation is a fairly new concept, although motion capture in itself is nothing really new. Its history can be traced back as far as the late 1800's, to experiments performed by photographer Eadweard Muybridge and others, who used photography to analyse human movement for medical and military purposes. It was however not until about 1917 when Disney started using the technique of tracing animation from film footage of live characters, that the benefits of motion capture really came to be. (Sturman, 1994). This technique of using photographed characters is known as rotoscoping.

Motion capture for computer animation really only emerged in the late 1970's when computing power was on the increase and it become feasible to animate characters by computer. At this time the traditional techniques such as rotoscoping were adopted for computer motion capture techniques. (Unknown E, 2001)

All of the current motion capture techniques rely on specialized markers as will be discussed in the section 2.1 entitled 'Motion Capture Techniques'

1.5 Why use Motion Capture?

Motion capture provides animators a simple and convenient method for animation. Motion capture costs less and is far less time consuming, than any other animation technique, such as key framing, parametric systems, or scripting systems. Yet, it provides more realistic results than ever before.

Motion capture allows animators to place actors in scenes which may be impractical or too dangerous to perform in real life. It enables scenes that would normally have been prohibitively impossible to be performed. Such scenes may include the repeat of a performance where the props may have been damaged. Motion capture also allows for the impossible to be done. This may include scenes such as giving animals human like walking motion or actors dancing with objects such as chairs. (Tager, 1994).

1.6 Document Overview and Layout

This paper presents a motion capture technique, that does not require any markers, body suits, or other devices attached to the performer. The only equipment needed is a hand held video camera. The remainder of the paper has been structured as follows.

Chapter2:- discusses the various motion capture technologies and file formats that are available and being used, in the industry today. The chapter discusses the feasability of these techniques and formats for the capturing of motion for quadrupeds.

Chapter3:- describes the design of the applications that are developed in this project. The chapter begins by discussing the requirements of the system being developed, the system architecture is then outlined, it goes on to discuss the user interface design and closes with a discussion on the data structures that are set up and used.

Chapter 4:- discusses the languages and platforms that have been used for the implementation of the applications designed in chapter 3. The implementation for each of the applications is the discussed separately.

Chapter 5:- summarizes the results that have been achieved. It discusses the functionality of each application and various screen shots of the working application are shown.

Chapter 6:- concludes the information discussed in this paper and summarises the results and achievements for each of the applications.

Appendix A:- is a poster summarising the project, methodology, implementation strategies and results.

Chapter 2 Related Work

Most of the work that has been undertaken on motion capture has been for human performers. The reason for this is that we are able to attach markers and sensors to humans as well as being able to communicate easily with them, as to what we wish them to do. This is not possible with quadrupeds and other living creatures. It is not feasible to attach markers to them, animals do not understand us and it is difficult to place animals into a studio environment.

This chapter begins by discussing the various technologies and equipment that are available for motion capture. It goes on, to investigate the various file formates that are available to store skeleton hierarchies and motion data.

The chapter concludes with a discussion on the feasibility of these technologies and file formats for motion capture of quadrupeds.

2.1 Motion Capture Technologies

Over the past few years a number of technologies have emerged in the field of motion capture. In this section the main technologies used for motion capture today are discussed together with the pros and cons of their application. The section concludes with a discussion on the feasibility of these applications for quadruped motion capture. As well as discussing a method that may better be used for this.

The main motion capture technologies in use today include prosthetic, acoustic, optical and magnetic systems. The two systems notable above all others available today are the optical and magnetical systems.

2.1.1 Prosthetic (Mechanical)

Prosthetic motion capture is one of the earliest methods of capturing a live characters motion. This method requires strapping prosthetic devices to the performers body (Unknown E, 2001). A set of straight metal pieces, which are shaped in a human body form, are attached to the performers back. As the performer moves so this exoskeleton is forced to move as well. A series of armatures connect each of these metal pieces to one another, in turn the armatures are connected using rotational and linear encoder's. The encoder's are then connected to an interface which enables simultaneously reading of positional and rotational data from each of them. (Tager, 1994). The data obtained from the encoder's along with a set of trigonometry functions is used to analyse the and capture the performers motion. The figures below show various mechanical motion capture systems that are available.

Figure 2.1: Prosthetic Motion Capture system (Gypsy 2.5 (Analogus))

(Figure taken from http://www.gasiopeia.com/sig/sig2.html)

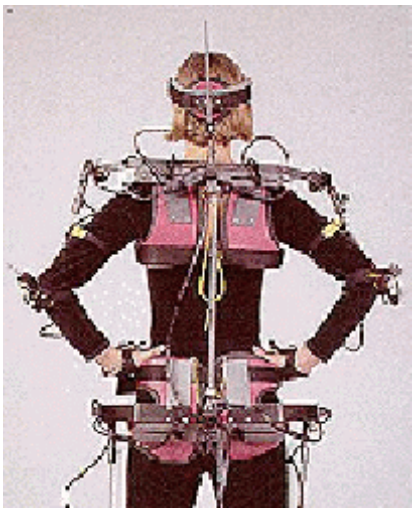

Figure 2.2: Prosthetic Motion Capture System

(Figure taken from http://www.cs.dartmouth.edu/~brd/Teaching/Animation/Papers/motioncapture.pdf)

There are other types of prosthetic motion capture devices which are for more specific motion capture such as the hands or faces, including gloves and mechanical arms. Figure 2.3 shows a diagram of the data glove used for motion capture of the hand.

Figure 2.3: 5DT Data Glove 5 (Gypsy)

(Figure taken from http://www.metamotion.com/motion-capture-hardware/motion-capture-hardware-gloves-Datagloves.htm)

The advantages of this method are that there is no interference of light or magnetic fields affecting the data. However the prosthetic devices restrict the performers motion capabilities to a large extent.

2.1.2 Acoustics

Acoustic tracking systems use high frequency sound waves to determine the position of objects (Casanueva, 1999). This method requires the use of three audio receivers which are placed in a triangle around the capture area. An array of audio transmitters are strapped at the joints of the performers body (Trager, 1994). Each of these transmitters sends out a click sound in a sequential manner. Each receiver uses the speed of sound to calculate the difference in time between transmission and reception of the sound emitted by a marker. This time is then used to calculate the distance from the transmitter to the receiver. The distances calculated by each of the three receivers are then triangulated to provide a 3D point in space for that particular transmitter. (Tager, 1994).

The acoustic method suffers from numerous disadvantages. Firstly the cables may hinder the actors motion capabilities, secondly the data for each joint cannot be collected simultaneously, thus subtle movements may have been made to the last sensor emitting a sound as compared to the first one being captured, and the size of the capture space is limited to the speed of sound and the accuracy is easily affected by echos and sound reflections. (Tager, 1994). The main problem associated with these systems is that "they require a straight line of sight at all times thus they suffer from markers being easily obscured" (Casanueva, 1999). Figure 2.4 shows an acoustics motion capture system from InterSence.

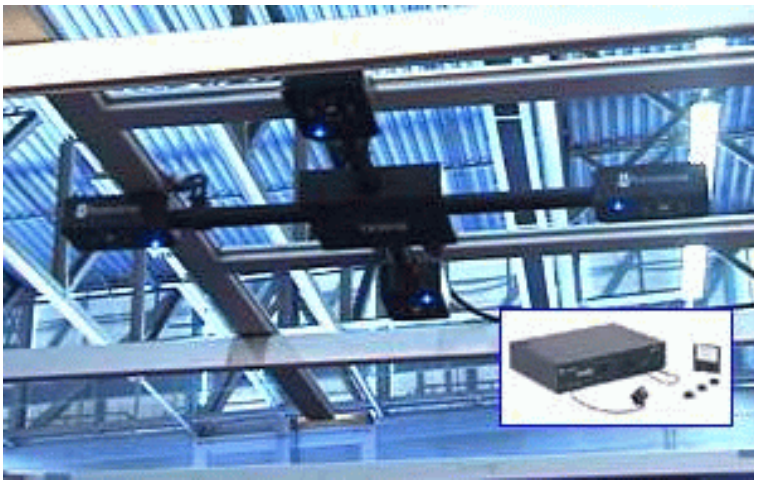

Figure 2.4: Acoustics Motion Capture System (IS-600 Mark 2 Plus (InterSence))

(Figure taken from http://www.gasiopeia.com/sig/sig2.html)

2.1.3 Magnetic

Magnetic motion capture systems require a centrally located magnetic transmitter. The performer wears an array of magnetic receivers which are placed on the performer's joints. These are connected to a computer via connecting cables. These sensors are capable of measuring their spacial relationship to the transmitter (Tager, 1994). Magnetic motion capture systems have the advantage that there is no shadowing or hiding of markers. However there are numerous disadvantages about this method. Magnetic motion capture have a limited range and data accuracy is reduced as the distance between the transmitters and receivers is increased. (Casanueva, 1999). The data can become noisy as the receivers are easily affected

by electrical interference and interference form other magnetic fields and metallic objects around the capture area. Another problem with magnetic systems arises when trying to capture the motion of several actors, when actors are in close proximity of each other, the magnetic fields of their receivers interfere with each other. Most magnetic motion capture equipment available requires wires to connect the transmitters and the receivers.

These wires will hinder the performers motion to a large extent. Magnetic motion capture also requires careful and detailed rehearsals to ensure that the actor is familiar with the constraints of the cables and the active space available (Furniss, 1999).

2.1.4 Optical

Optical systems require the placement of markers in strategic positions on the performers body. The markers used are either reflective in nature or infa-red emitting. Each of these markers are then tracked by an array of specially placed cameras (Garrison, 2000). The information is triangulated between the camera to calculate the position of the reflectors. Optical systems are very accurate and offer the performer the highest degree of freedom since there are no connecting cables restricting their movement capabilities. The drawback of optical systems is that they are prone to light interference (Tager, 1994) and they suffer from the shadowing or hiding of markers, which can cause false marker readings to occur. Shadowing problems occur when the path to a specific reflector is obstructed, this causes the marker to be obscured from the receiving camera/s. This causes a lose of data from these markers. (Casanueva, 1999). Markers can be obscured by the performer themselves or by other structures in the scene. (Tager, 1994). Figure 2.5 shows clearly what is meant by markers being obscured. One can see that the markers on the left hand-side of the performers body are not visible to the camera, as well as the right arm possibly obscuring other markers.

Figure 2.5: Optical Motion Capture System Showing hiding of Markers

(Figure taken from Luis, 1999)

2.1.5 Feasibility of Current Motion Capture Equipment for Quadruped Motion

Each of the technologies described above have their pros and cons. However each of them require the placement of some type of marker onto the performer, and they require the performer to be in a studio.

Although there have been some great enhancements in these areas, these technologies are best suited for biped motion of humans. They are not the best solution for capturing animal motion and the motion of various other objects. The capturing of live animal motion poses many difficulties, a major concern is the placement of markers on an animal. Locomation Studios using Vicon 8 have used a suit specially made for a horse. This has worked however this horses motion is still very restricted and they needed to use "a highly trained horse" (Tustin ,1999).

This project suggests the use of a very different technique. The technique suggested, is the use of a single hand held video camera to capture sequences of real life motion. Still images are acquired from this video data and used to capture motion data for computer generated avatars. Instead of using magnetic, optical or other such markers to gather this data, the still images can be used to mark-up the various joints and their motion. The use of a single hand held camera for motion capture will offer many advantages, however using

this method it is very difficult to obtain three dimensional coordinates for the system. Thus this method is not necessarily the most accurate. The advantages of this system are that moving objects such as animals may be filmed in their natural habitat without placing any markers or restrictions on them. There is no need for any specialised equipment or set up process, a single hand held video camera being all that is needed to capture motion. The animator is able to rewind and fast forward the tape in order to grab the exact frames of motion wanted. It is also possible to easily remove frames from the beginning and end of the sequence. This method also has the problem of some points being hidden from the camera. This is however is easily dealt with. It is easy enough for the user to be able to visualise where these point should be and mark them up. The difficulty in finding the three dimensional co-ordinates for a point is not the only disadvantage of this method. Other disadvantages are that when grabbing frames from the video sequence some images can be very blurred and the mark-up of the skeleton and motion may take more time than traditional motion capture techniques might.

Table 2.1 provides an comparison of the advantages and disadvantages for the various motion capture technologies in use today. It also shows how the use of a hand held camera as suggested in this paper compares to each of these technologies.

	Prosthetic	**Acoustic**	**Optical**	**Magnetic**	**Hand Held Camera**
Occlusions	**?**	**?**	**?**	**?**	**?** (but easily visualised by the user)
Restriction of Movement	**?**	**?**	**?**	**?**	**?**
Outside Interference	**?**	**?**	**?**	**?**	**?**
Markers Required	**?**	**?**	**?**	**?**	**?**
Studio needed	**?**	**?**	**?**	**?**	**?**

Table 2.1: Comparison of Motion Capture Technologies

2.2 Motion Capture Files Formats

The data that is being obtained by motion capture equipment needs to be stored in some sort of manner. There are many companies who have developed methods for storing this data. Each of these is listed below with a very brief discussion on each. The main two in the industry are the BioVision BVH format and the

Acclaim ASF/AMC format. These two are discussed in further detail. The section concludes with a discussion on the pros and cons of these formats.

2.2.1 BioVision Motion Capture Studio Formats

The BioViosion Motion Capture Studios have two file formats. The BVA format is an older version of the BVH format that is now more commonly used. The BVH is an improvement on the BVA format. BioVision has a third format known as the BioVision/Alias ASK/SDL format.

2.2.1.1 BioVision BVA

The BVA format is the first format developed by BioVision. This format however does not have any skeleton definition included in it. It is thus difficult to use effectively and consequently BioVision developed the BVH format (discussed in the next section). (Thingvold, 1999). The BVA format is the simplest of the formats that are available. (Lander, 1998).

There is no hierarchy definition for this format. Each bone's actual position is described for each frame. Each bone or segment has nine channels of motion. These include the x, y and z values for translation, rotation and orientation for each bone in each frame. Listing 2.1 shows a piece of a .BVA file. We see that the first line describes the first bone as the 'Hips'. The next line describes the number of frames of motion for this segment. The third line describes the sampling rate of the data. In Listing 2.1, one we see that there are 29 frames and the frame time is 30 seconds (per frame). Following this is a description of the channel types and the units for each. This is followed by the actual channel data. The number of lines of channel data will correspond to the frame count, specified by "Frames: 29" in the file. Each line will have nine values each. These values are followed by another segment block which will describe the next bone's information. (Lander, 1998 and Thingvold, 1999).

Segment: Hips
Frames: 29
Frame Time: 0.033333

XTRAN INCHES	YTRAN INCHES	ZTRAN INCHES	XROT DEGREES	YROT DEGREES	ZROT DEGREES	XSCALE INCHES	YSCALE INCHES	ZSCALE INCHES
8.03	35.01	88.36	14.78	-164.35	-3.41	5.21	5.21	5.21
7.81	35.10	86.47	12.94	-166.97	-3.78	5.21	5.21	5.21

... Repeats for a each frames
Segment: Chest
Frames: 29
Frame Time: 0.033333

XTRAN INCHES	YTRAN INCHES	ZTRAN INCHES	XROT DEGREES	YROT DEGREES	ZROT DEGREES	XSCALE INCHES	YSCALE INCHES	ZSCALE INCHES
8.33	40.04	89.69	-27.24	175.94	-2.88	18.65	18.65	18.65
8.15	40.16	87.63	-31.12	175.58	-4.08	18.65	18.65	18.65

.. Repeats for each frame
Segment: Neck
Frames: 2
... Repeats for each bone/joint in the skeleton

Listing 2.1: Sample BioVision .BVA File Format

2.2.1.2 BioVision BVH

BioVsion .BVH format has several key differences from the .BVA format. The most significant of these diffrences being that the .BVH file stores motion data for hierarchical skeletons.

The BVH file format is an ASCII file format. The .BVH file format replaces the earlier .BVA format developed by BioVision. .BVH stands for BioVision Hierarchical Data. This format provides skeleton information and motion data.

```
HIERARCHY
ROOT Hips
{
        OFFSET 0.00 0.00 0.00
        CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
        JOINT LeftUpLeg
        {
                OFFSET 0.0000 -0.0000 -2.8784
                CHANNELS 3 Zrotation Xrotation Yrotation
                JOINT LeftLowLeg
                {
                        OFFSET -0.0000 -17.1369 -3.0217
                        CHANNELS 3 Zrotation Xrotation Yrotation
                        JOINT LeftFoot
                        {
                                OFFSET -0.0000 -16.0556 -2.8310
                                CHANNELS 3 Zrotation Xrotation Yrotation
                                JOINT ltoes
                                {
                                        OFFSET 2.7557 -2.2772 -0.4015
                                        CHANNELS 3 Zrotation Xrotation Yrotation
                                        End Site
                                        {
                                                OFFSET 1.3778 -1.1386 -0.2008
                                        }
                                }
                        }
                }
        }
... Repeats for each bone/joint in the skeletal hierarchy
}
MOTION
Frames: 183
Frame Time: 0.033333
37.8224 40.0267 89.8209 1.8411 ... Repeats for each channel of motion
... Repeats for each frame
```

Listing 2.2: Sample .BVH File Format

As can be seen in Listing 2.2, which shows a fragment of a .BVH file, the .BVH file has two parts to it, the first section describes the hierarchy and the initial pose of the skeleton; and the second section lists the motion data. The hierarchical section begins with the word "HIERARCHY". Which signifies the start of the skeleton definition section. The very next line starts with "ROOT" which is then followed by the name of the root segment for the hierarchy that is to be defined. This is the parent bone for all other bones in the hierarchy.

The .BVH format now becomes a recursive definition. Each joint is contained within a set of braces. All of the joints that are within these braces are the children of the joint where the braces were opened. Each joint has an **OFFSET** and **CHANNELS** definition described for it following the **JOINT** and name definition. Each joint (bone) in the hierarchy contains information that is relative to only that joint (bone). The **OFFSET** describes that joint's displacement from its parent joint. (And as such the length of the bone between these two joints). These are xyz world coordinates. The **CHANNELS** line defines which of the bones parameters will be animating in the motion section of the file. The first parameter is the number of channels animated for this bone, followed by a list of the channels for animation. Only the root bone has positional data associated with it and the rest of the bones only have rotational data. (In this project we have assumed that the channels come in the order Zrotation, Xrotation, Yrotation). Each branch in the hierarchy ends with a joint named **End Site**. This is to determine the length of the last bone.

Following the hierarchy section is the motion section, which begins with the word **MOTION,** this section describes the animation of each bone over time. The line proceeding this begins with **Frames:** and then a number to represent the number of frames that the file describes. The next line begins with **Frame Time:** which defines sample rate of the motion data. Each line in the motion section represents one time frame. In each line there is a value for each of the **CHANNELS** described in the **HIERARCHY** section. (Lander, 1998 and Thingvold, 1999).

2.2.1.3 Biovision/Alias ASK/SDL

The Biovision/Alias ASK/SDL is a special variant of the BioVision .BVH file. This file was developed to accompany Alias' Poweranimator SDL files which contain motion information for transformation nodes in Alias Poweranimator. ASK stands for Alias Skeleton. ASK files only contain skeleton information, there is no channel or motion data stores in these files. The .ASK's format is virtually the same as the .BVH format except that in the .BVH file the offset of the children bones/segments is relative to their parent bone/segments position. In the .ASK format offset data is in global coordinates and there is no dependency on the parent bones/segments. (Thingvold, 1999)

2.2.2 Acclaim

Acclaim is a gaming company that has been involved with motion capture research for many years.

Acclaim has two file formats for describing and storing skeletal motion data. This format is regarded as the most complicated of the file formats. Acclaim separates the storing of the skeleton information and the motion into two separate files. The benefit of this separation being that one .ASF file is needed for a specific skeleton and multiple .AMC (movement) files can be used. The reason being that most motion sequences use the same skeleton definition, thus it is logical to store the skeletal information only once and various motion sequences in many other files. Various motion files will then be able to use one skeleton file definition over and over again. The .ASK file stores the skeletal information and the AMC file stores the motion data. The Acclaim format is a very technical and complex file format. (Lander, 1998).

2.2.2.1 ASF

.ASF stands for Acclaim Skeleton File. This file stores the base pose of the actual skeleton and its hierarchy. The .ASF format is a very technical and complex format. Lander (Lander, 1998) believes it is the most complicated of the formats available.

The .ASF format describes the joints and the hierarchy of the skeleton as does the first section of the .BVH format. The .ASF format however extends this to include data describing the skeleton in terms of its shape, hierarchy and the properties of each of the joints and bones. (Schafer, 1995)

The keywords of this format all begin with a colon "**:**". The file is divided up into various sections. Each section starts with a keyword and will continue until the next keyword is reached. Listing 2.3 shows a portion of the Acclaim .ASF format. The explication of this file below relies heavily on this listing.

```
# Comment Lines
:version 1.10
:name ASFHop
:units
  mass 1.000000
  length 2.540000
  angle deg
:documentation
  Example of an Acclaim skeleton
  To be used with "hop.amc"
:root
  axis xyz
  order tx ty tz rz ry rx
  position 0 0 0
  orientation 0 0 0
:bonedata
  begin
    id 1
    name LeftHipDummy
    direction 0.762291 -0.286258 0.58049
    length 12.5411
    axis 0 0 0 xyz
  end
  begin ... Description of the next bone begins
    id 2
    name LeftHip
    direction 6.12281e-017 -0.999964 -0.00843259
    length 42.6915
    axis 0 -60 -90 xyz
    dof rx ry rz
  end
  begin
    id 3
    name LeftKnee
    direction -0.00230476 -0.998652 -0.0518571
    ... same elements as bone above
  end
 ... Continues until all bones are specified
:hierarchy
  begin
    root LeftHipDummy RightHipDummy ChestDummy
    LeftHipDummy LeftHip
    LeftHip LeftKnee
    ... Continues until hierarchy is defined
  end
```

Listing 2.3: Sample .ASF File Format

The keyword **:version** indicates the version of this file format. The **:name** allows the skeleton to be named something different from the file name. The **:units** section defines all the values and units of measure used.

The **:documentation** section allows for documentation information about the file to be stored. This information will persist from one file creation to the next. The comment information may be easily changed and there is thus no guarantee that it will be retained when copying these files.

The **:root** section defines the parent joint of the entire hierarchy. The **axis** and **order** elements describe the order of operations for the initial offset and transformation of the root node. The **order** element specifies the channels of motion that are applied to the root and in what order they will appear in the .AMC file. The **position** element describes the root translation of the skeleton and the **orientation** element defines the rotation. The **position** and **orientation** elements are both followed by a triplet of numbers that are used for their description.

The **:bonedata** section contains a description of all of the remaining segments in the hierarchy. This section only describes each segment, there is no information on the relationships between any of the joints/bones here, this will be describe later in the :hierarchy section. Each bone is capsulated between **begin** and **end** statements. The **id** element provides each bone with a unique id. The **name** element provides a name for the segment. The initial rest position of the skeleton is described by the **direction** element, and the **length** element describes the physical length of the bone. These two elements combined provide the information needed for rendering the bone. The **axis** parameter is followed by a triplet of numbers that define the global orientation of the joint, these are proceeded there letters xyz which specify the order of the rotations. The Degrees of Freedom for each joint specifies the number of motion channels and the order in which they appear in the .AMC is describe by the **dof** element (if the segemnt does not have a **dof** element then it has no motion data associated with it). The **limits** element puts limits onto each bones dof. For each channel that appears there will be a pair of numbers inside parenthesis indicating the minimum and maximum allowed value for that channel.

The ast section is the **:hierarchy** section, which describes the hierarchy of the bones described in the :bonedata section. Each line in this block begins with the parent bone which is followed by a list of all of its children nodes.

2.2.2.2 AMC

The .AMC file contains the motion data for the skeleton that has been defined in an .ASF file. Each .AMC

file tied to a particular skeleton or .ASF file. In the .AMC file the data is grouped by frame. First the frame number is specified and is then followed by a number of lines, containing the motion information. Each line begins with the bone name followed by numbers describing the motion data. (Thingvold, 1999) The bones are in the same order for every frame. (Schafer, 1995).

```
# AMC file for hop.asf
:FULLY-SPECIFIED
1
root  -86.3268 -39.9639 -24.5891 -0.150874 -37.9647 -3.45307
LeftHip  0.0927953 -15.7659 9.79131
LeftKnee  -4.23651 6.25758 41.615
LeftAnkle  -8.86676 -0.778883 19.9955
LFoot  -0.0419846 0.663786 1.09843e-015
:
:
2
root  -86.3811 -39.9123 -25.4018 0.248165 -38.1775 -3.46523
LeftHip  0.25837 -15.9107 9.10496
:
:
```

Listing 2.4: Sample .AMC File Format

2.2.3 Other Motion Capture File Formats

There are various other file formats that are available, however the BioVision and Acclaim formats have become the industry standard and are the most popular formats used. Below is a brief discussion of various other file formats that are available.

2.2.3.1 Motion Analysis HTR

The Motion Analysis HTR (Hierarchical Translation-Rotation) was developed before the Accliam format was entered into the public domain. This format was created as an alternative to the .BVH format, which was the only existing hierarchical skeleton method at the time. This method was developed as a native skeleton format for the Motion Analysis skeleton generating software. The HTR contains everything necessary for a motion capture file it has lots of flexibility in data types and ordering and it has a complete basis pose specification where the starting point for both rotations and translations are given. (Thingvold, 1999)

2.2.3.2 LambSoft Magnetic Format BRD

The LambSoft BRD file format was devised to store data from the Ascension Technologies "Flock of Birds" motion capture system. Although the suffix "BRD" implies that it is only for the Flock of Birds system this format is capable of storing data from any magnetic system.(Thingvold, 1999)

2.2.3.3 Polhemous DAT Files

The dat file format devised by Polhemous is to support their real time magnetic motion capture system. (Thingvold, 1999)

2.2.3.4 Ascension ASC files

The ASC file format by Ascension Technologies is a format they created to store data from their magnetic motion capture systems. (Thingvold, 1999)

2.2.4 Feasibility of Current Motion Capture File Formats for Quadruped Motion

As has been mentioned previously although there are many motion capture file formats available the main two companies are BioVision and Acclaim with their .BVA, .BVH and .ASF, .AMC files respectively. The BioVision .BVH format is far simpler and easier to implement, it is for this reason that the .BVH format

is chosen as the file format on which to concentrate for this project.

Chapter 3 Design *(Development Environment)*

This paper proposes a method for motion capture that reduces the complexities and complications of current technologies. A method whereby motion data is extracted from 2D video sequences captured using a single hand held camera is proposed. This technique offers many advantages over current technologies. The advantages and disadvantages of this method for motion capture over other techniques has been discussed in section 2.1.5

This chapter begins with a description of the requirements of the system. It then goes on to discus the system architecture and the components involved in the system, the communication between these components and their functionality. This is followed by a description of the user interface for the system. The chapter ends with a discussion on the various data structures that are set up and used.

3.1 System Requirements

The system developed has two main requirements. The first being the viewing and capturing of motion data from video sequences, recorded with a single hand held camera. The second being the modelling and animation of articulated skeletons in a virtual environment. The design of the system developed to achieve these requirements is divided into two separate applications. Each application dealing separately with the above requirements.

3.2 Motion Capture Application *(Viewing and Capture of Motion Data)*

The Motion Capture Application has two parts to it. The first component deals with the digital presentation of continuously playing video data, that has been recorded to tape as well as the presentation of the captured frames as sequential still images. The second component deals with the annotation of the skeleton and motion information on each of the image frames are captured

The application thus deals with digitizing video data from video footage and the manipulation of the captured frames. This manipulation of frames is necessary in order for the user to be in a position to best be able to select images and frames which they wish to use for annotation of the skeleton.

The application must allow the user to specify a starting and a stopping point within the recorded video data and once a sequence of images has been captured the system must enable the user to switch between the viewing of the camera information and the viewing of these captured images as still frames. Whilst viewing the captured images the user should be able to easily navigate through them. The application must also deal with navigation between captured frames, the removal of a single frame from the sequence or the removal of the entire sequence and the saving of these frames. Once it has been decided which frames will be used to develop motion data files, the images need to be saved. When saving the images it is vitally important to keep the images in their sequential order.

The application then needs to enable the user to annotate the skeleton by using the mouse to clicks on the required joint/points of reference and build up a 3D hierarchical skeleton and show its 2D visualization as the user steps through each frame. In order to be able to successfully model the images, various data will need to be identified and recorded. The joint positions relative to one another, the positions of the joints in space and the rotation of the joints and limbs in the x, y and z planes. Using the clicks made by the user the application is required to calculate this information enabling the capturing of motion data that can be saved and replicated for later modelling and animation.

The accuracy and completeness of the animation and the ultimate success of the whole system will depend on the correct selection of frames, sequences and the ability of the user in setting up the skeleton and its motion. Only experience will determine the number of frames needed to best replicated the movements wanted. But as the system grows so more and more routines and sequences will interlink, enabling the animator to carry out unlimited visual sequences.

A model viewer application is then needed in order for the user to be able to view this captured motion data for computer generated characters.

3.3 Model Viewer Application

Having viewed the required video footage which is to be used for motion capture, selected the frames and sequences to be used and suitably annotated the skeletons on these frames, and written this data to a motion capture file, we are now in a position to carry out modelling and animation of the articulated skeleton.

The Model Viewer application is developed to deal with this second requirement. Modelling and animation of articulated skeletons is done by the extraction of data from motion capture file formats. As previously discussed we focus on the BioVision .BVH file format.

Requirements for the Model Viewer Application include being able to read and interpret the information in motion data file formats. The application needs to verify that the correctness of these files, while reading the information it also needs to store it to in some way for later use.

Once the reading and storing of the information from the motion data file is successful the application needs to use this data to model and animate articulated skeleton avatars in a virtual environment.

The user needs to be able to view the avatars motion frame by frame as well as viewing it playing continuously. The user should also be able to select the direction of playback. (Either forwards or backwards).

3.4 System Architecture

The framework for motion capture and model viewer applications, is composed of three basic modules which are supported by a graphical user interface (GUI).The modules include input, processing and output modules. The entire framework then works on manipulation of the various data structures. (Da Silva et. al., 1997).

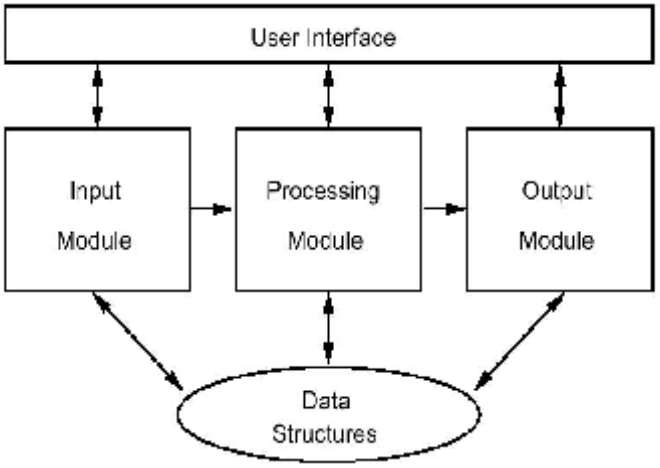

Figure 3.1: Basic System Architecture

3.4.1 Motion Capture Architecture

3.4.1.1 Input Module

The input module of the Motion Capture Application focuses on dealing with the video data information, from the video tapes recorded using a hand held video camera. As well as the sequential saving to disk of the images that have been captured. This module also deals with the users interaction of clicks made on the images. These clicks are used to build up the skeleton. The user needs to easily set up the initial pose skeleton as well as move the joints on each frame, in order for the motion of the skeleton to be recorded. Each of the clicks made by the user represents a joint, the application needs to calculate the parent child relationships between theses joints. Various input from the users part will be needed. The application will need to obtain information about the third co-ordinate from the user. The user will need to supply a name for each of the joints.

3.4.1.2 Processing Module

The processing module of the Motion Capture Application deals with the actual creation of these joints and the skeleton as the user clicks on the image frames. Once the skeleton is built up and as the user steps through each frame moving the positions of each joint, the application needs to deal with the calculation of

the rotational and orientation data for each joint.

3.4.1.3 Output Module

The output module uses the information that has been gathered to write to motion capture files. These files can then be used later in the modelling and animation of avatars. The format of file that the application developed for this paper is the BioVision .BVH format.

3.4.2 Model Viewer Architecture

3.4.2.1 Input Module

This module focuses on the reading and interpretation of motion capture file formats. From these files the application is able to obtain information about the hierarchical skeleton for which motion will be mapped to, the number of frames being described, the sample rate for this motion data and the actual motion data itself.

3.4.2.2 Processing Module

The processing module comprises of the set of tools for dealing with the data structures, in order to be able model articulate skeletons and to view the animation of these skeletons.

3.4.2.3 Output Module

This module deals with the rendering of the skeleton in a three dimensional virtual environment.

3.5 User Interface

The user interface for the Motion Capture and Model Viewer applications is a very important part of the application. The interface needs to be intuitive for the user to use and the user needs to be able to perform actions which can easily be undone. The interface needs to handle and prevent errors that may be made

by the user and needs to provide the user with feedback as to what state the application is currently in.

The design of the user interface is based on the model-view-controller (MVC) paradigm. The user interface is divided into three components. These components are virtually independent of one another. They are divided according to the separate functions that they perform as follows:

Model:-The model holds the data that is being manipulated and performs all of the computations

.

View:- The view is the actual interface that is seen by the user. The view displays the interface and the relevant data to the user.

Controller:- The controller responds to the events created by the user, it notifies the model and the view appropriately.

(Kenny, 1995)

The model component here represents the processing module in the basic framework described earlier. The view represents the user interface and the controller works to integrate these two components.

The advantage of using the MVC method is the separation of the code into the model, view and controller. This separation introduces many advantages, it makes it easier to change the interface without affecting the code for the model. When porting the application to other platforms only the code for the interface(view) needs changes, the underlying back-end or model is unaffected. Details of the application interface designed using the above paradigm follow.

3.5.1 User Interface (Motion Capture Application)

The interface for the motion capture application has three windows. The first window is used to display images to the user. The window is set to the size of the initial image being viewed. The user is unable to resize the window, this ensures that all of the images captured in a motion sequence are the same size. The images must be the same size so as to avoid any distortion in the visual representation of the skeleton to the user, during mark-up.

The second window allows the users to control the data source being displayed in this first window. It allows the user to switch between viewing the data playing from the camera and the images that have been captured. When the user is viewing the saved images they are offered the ability to be able to navigate through these images, remove frames or the entire sequence and lastly they are able to save the frame sequence. Figure 3.5 shows basic sketch of the interface proposed for this control window.

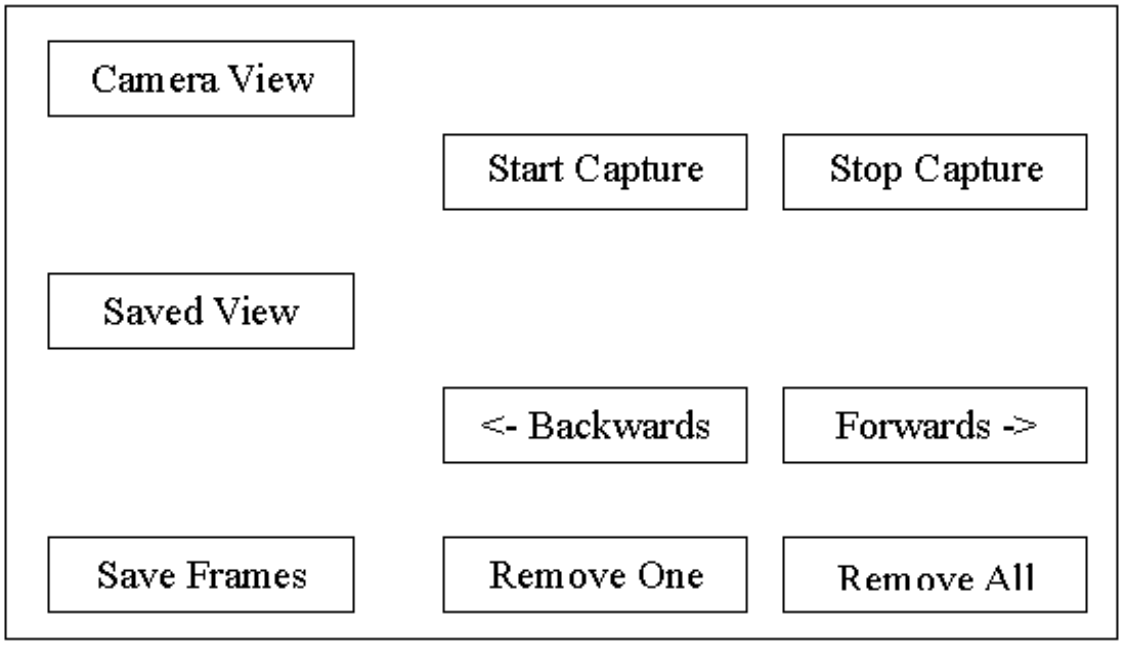

Figure 3.2: Proposed User Interface for the Control Window

The Camera View and Saved View buttons will allow the user to toggle between viewing the camera data and the captured data. The start button allows the user to specify a starting point of where the capturing should begin and the stop button allows the user to end this capturing process. The backwards and forwards buttons allow for navigation through the images when in saved view mode. The remove one button enables the removal of a single frame from a captured sequence and the remove all button allows for the entire captured sequence to be discarded. The save button invoke the saving of all the images that have been captured. The user need only specify a single file name hare and the application will save these images in their order.

The third window allows for the mark-up of the hierarchical skeleton and the motion data. This window allows the user to click on an image in order to create the joints and build up the skeleton. The user is able to load each of the frames sequentially until the final frame is reached. Figure 3.6 shows a sketch of the

interface for the Mark Up application.

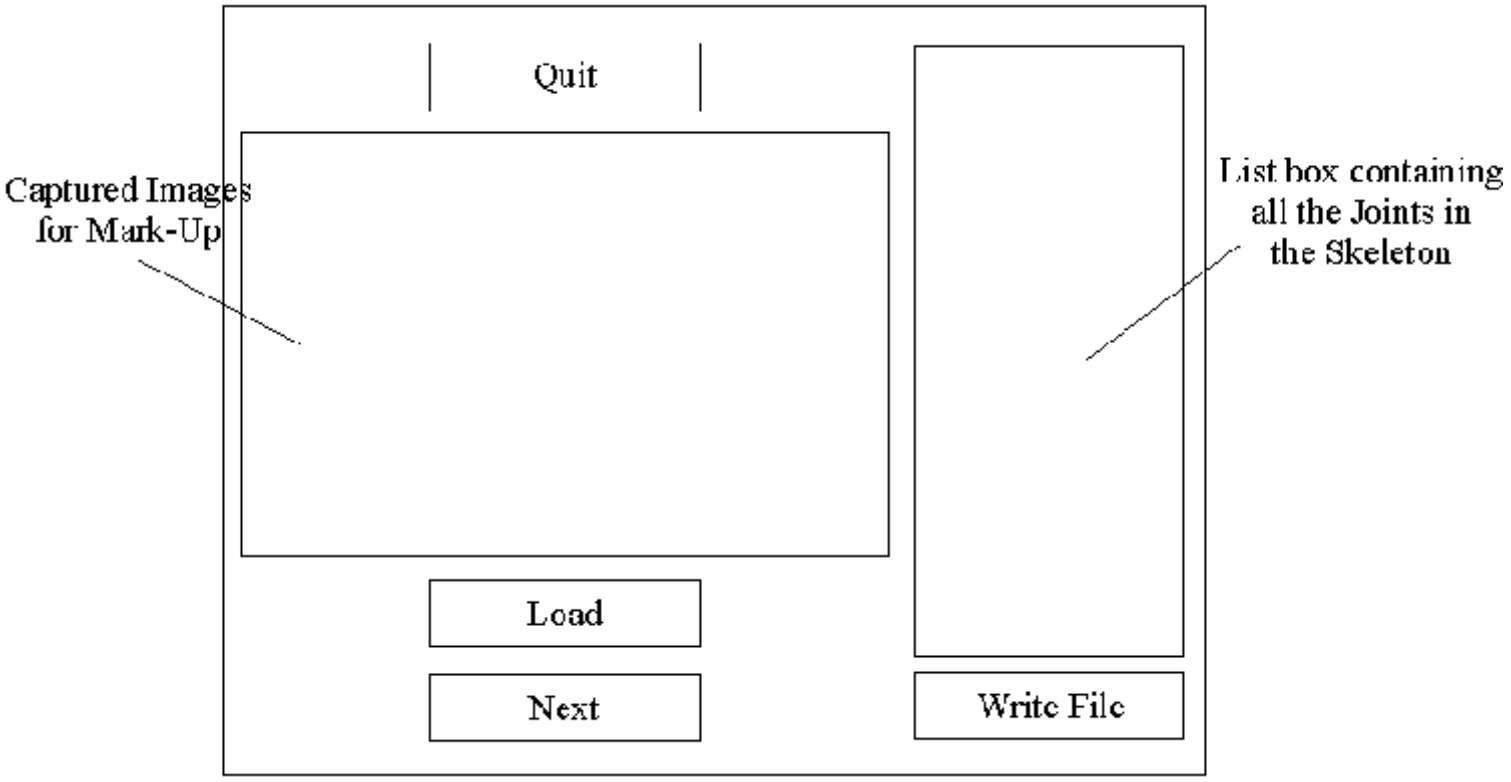

Figure 3.3: Proposed User Interface for Mark Up Window

The load button allows for the first image in a sequence to be loaded, and the next button allows for subsequent images to be loaded. The user is able to click on the rectangular area where the captured images are displayed. The co-ordinates of these clicks are used to build up the skeleton and it motion. On the right hand side is a list box which lists every joint in the skeleton. Once all of the images have been loaded and marked up the user is able to click on the write file button which will write and save a BioVision .BVH file.

3.5.2 User Interface (Model Viewer Application)

The interface for this application has been designed to mimic the interface of a video player. Users are familiar with the icons used on video players and their various action. The video player metaphor was chosen for its familiarity. This provided a simple and intuitive interface. The icons on a video player and those used for this application are very similar. Thus, the user need not learn meanings of any new metaphors. The icons used are a square to represent stop. Then the play buttons used are the traditional

triangle (of the video player), however the direction of the triangle represents the direction of playback. Two overlapped triangles represent continuos play and a single triangle represents stepping through the motion one frame at a time. The use of small buttons with icons, unclutter's the interface and allows for most of the screen to be taken up by the virtual world.

The model viewer interface is very simple. There are various navigation buttons which allow the user to choose the direction and mode of playback for the motion of the avatar in the virtual environment. The 3D world is shown in the bottom right of the window. Figure 3.7 below shows the proposed interface for the Model Viewer Application.

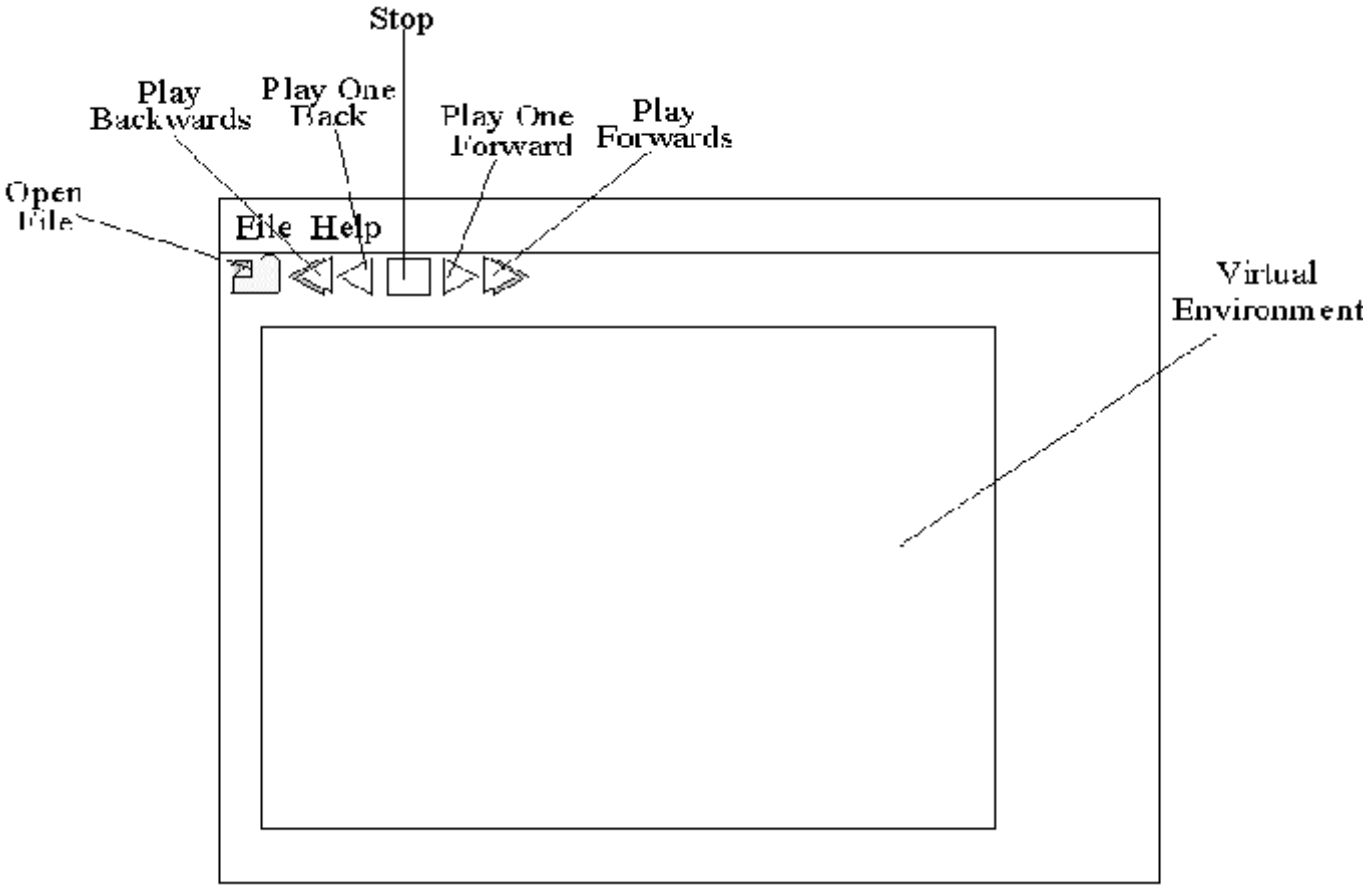

Figure 3.4: Proposed User Interface for the Model Viewer Application

The buttons in the top left corner allow for the user to select the direction and mode of play back. The virtual environment is displayed in the right bottom corner. The user is able to open various file by either using the file menus of the open button on the tool bar.

3.6 Design of the Data Structures Used

Both applications work on the same data structures. There are two underlying data structures. A hierarchical skeletal structure and a joint object. The skeleton is a tree like structure built up of a number of joints as shown in figure 3.1.

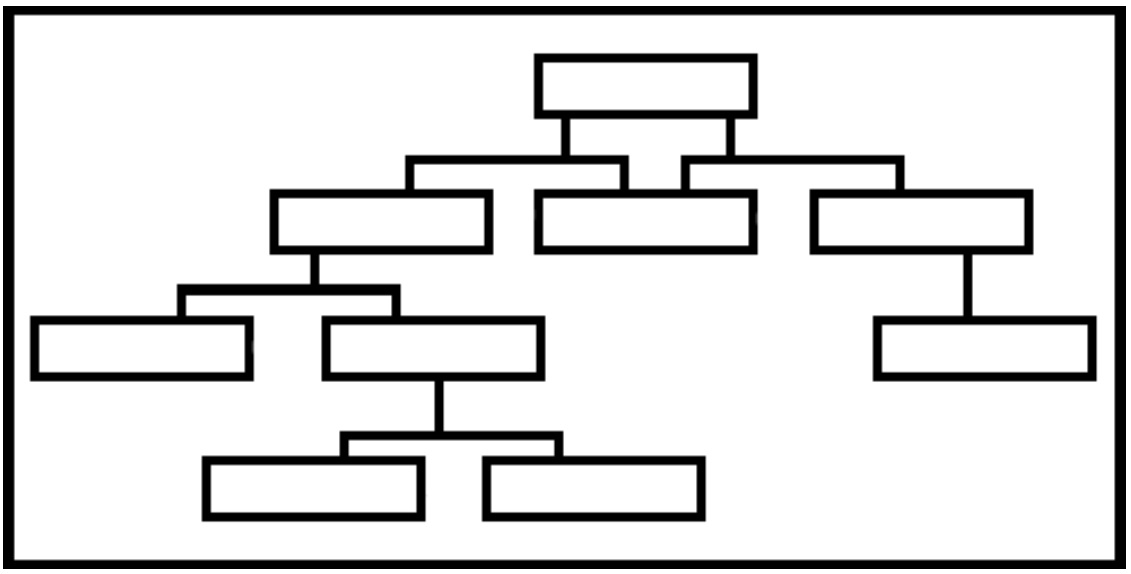

Figure 3.5: Shows the Hierarchical Skeleton Structure

Each joint has a single parent joint (excepting for the root joint) and a number of children joints. Each joint has its own set of attributes. These attributes describe the joint and they way in which it relates to other joints in the skeleton. Each joint has its own ID and name, an offset which is used to determine its position relative to its parent joint. Only the root joint will have positional data associated with it. Each joint has a number of animation channels which are used to calculate its orientation. Each branch of the skeleton is terminated with a joint that is named “End Site. Below is the Universal Modelling Language (UML) representation of these structures.

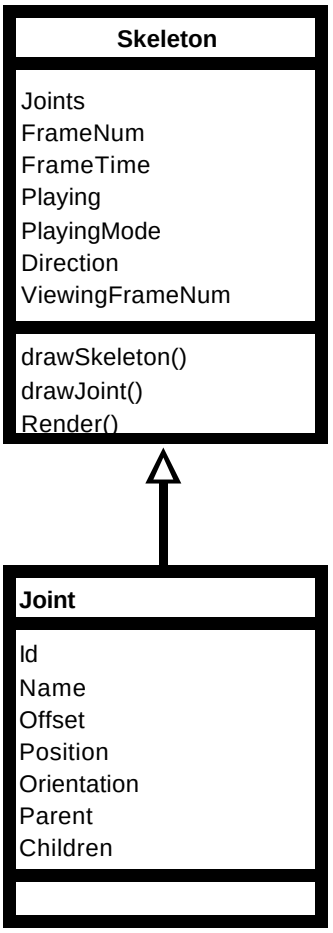

Figure 3.6: UML Description of the Data Structures Used

The ID attribute is use to associate the motion data with the correct joint. The motion information in a .BVH file is specified for each joint in order that the bones appeared in the first section of the file (the hierarchical definition of the skeleton see section 2.2.1) and not in the order that the joints appear in the skeleton structure.

The offset attribute for each of the joints is used to position the joints relation to their parent joints, this is used to create the visualization of a skeleton as shown in figure 3.3 instead of the true structure as shown above in figure 3.1

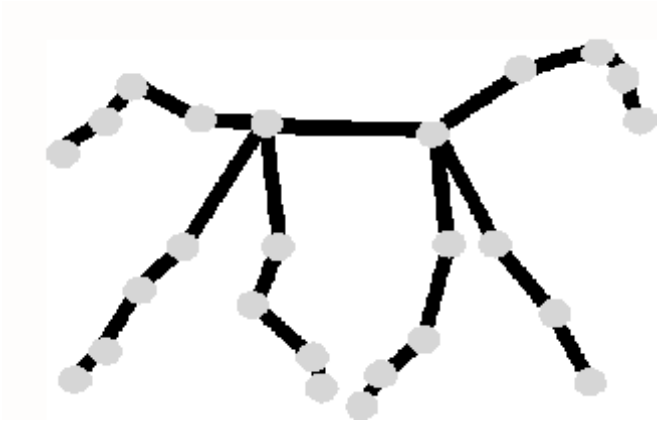

Figure 3.7: Visualisation of the Skeleton Structure (Using offset values to position joints)

The position attribute is used position the skeleton in the virtual world. Only the root joint will have positional data associated with it. The orientation attribute holds the orientation in the x, y and zplanes for that joint at a particular point in time. Each joint will have only one parent joint and may have any number of children joints.

Chapter 4 Implementation

This chapter introduces the reader to the implementation technologies used for the designs of this research project as discussed in chapter 3, as well as to provide justification for the choice of these technologies.

4.1 Implementation Languages and Platforms

4.1.1 Open GL

In order to more easily develop an interactive three-dimensional graphics application, without having to deal directly with the hardware the OpenGL graphics system software has been used OpenGL is a cross-platform standard for 3D rendering and 3D hardware acceleration. This software provides an interface to the graphics hardware of a computer. OpenGL allows the programmer to more easily create programs with moving three-dimensional objects. (Unknown A, 2001)

As discussed on the official OpenGL website, OpenGL has become computer graphics industry standard and most widely used, supported and documented 2D and 3D application programming interface (API). As such it is very inexpensive and easy to obtain information and help for. (Unknown B, 2001). It is for this reason that the choice was made to use OpenGL as the rendering language for this project.

OpenGL has been designed so that it is a hardware independent interface that can be implemented on different platforms. To achieve this, no commands for performing windowing tasks or obtaining user input have been included in OpenGL. Instead the programmer needs to work through the windowing system that controls the hardware they are using. (Neider *et. al*,1997).

4.1.2 Qt C++ GUI Application Development Framework

When using OpenGL it is necessary to make use of a windowing toolkit in order to deal with the windowing tasks and obtaining input from the user. Numerous toolkits are available. The better known two are GLUT (OpenGL Utility Toolkit) and Trolltech's Qt widget Library.

GLUT implements a simple windowing Application Programming Interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. However GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits and GLUT only supports C, C++ (same as C), FORTRAN, and Ada programming bindings there are no bindings for (Java[1] [TM]). It is for this reason that the choice was made to use the Qt-Library. (Unknown C, 2001)

The Qt-Library is a C++ Graphic User Interface (GUI) application development framework. Qt provides a cross-platform solution. Qt allows one to have a "dumb" GUI with generic code controlling it. Qt simplifies the task of writing and maintaining GUI (graphical user interface) applications. Qt is written in C++ and is fully object-oriented. Qt allows one to develop and maintain a single source code base. Qt also has an extension module for OpenGL. The Qt OpenGL extension makes it easy to add a full-fledged GUI to an OpenGL program. It lets you interface the OpenGL renderer to the Qt user interface. (Unknown D, 2001). This will in turn provides the ability to program the model in Java. Using native calls from the C++ GUI implementation to the Java model.

4.1.3 Java™ Technology

Java provides an easy-to-use language which helps the programmer to "avoid the pitfalls of other languages, such as pointer arithmetic and memory management, that affect code robustness" (Sun Microsystems, 2001). Java is an object orientated programming language, which helps the programmer to more easily visualize their program in real-life terms and it "provides a means to make code as streamline and clear as possible" (Sun Microsystems, 2001). The Java technology architecture also provides automatic garbage collection and offers a great deal of code security. Java is very well documented and the documentation is easily available on the net. It is for these reasons that Java is chosen as the implementation language

Java is normally compiled to Java byte code which is then interpreted by the Java Virtual Machine (JVM). This makes Java an interpreted language which makes it a bit slower then other languages.

Java on Greatdane, however has been complied using Gnu Compiler for the Java (GCJ). "GCJ is a portable, optimizing, ahead-of-time compiler for the Java Programming Language". (Unknown F, 2001). The GCJ compiler is able to compile Java source code directly into native machine code, or to Java bytecode and it can also compile Java bytecode to native machine code. (Unknown E, 2001). This enables Java programs to run faster. We use this method because of the fact that we are dealing with Virtual Reality. Computer graphics and Virtual Reality take up lots of computational power. We don't want to jeopardize this even further by using an interpreted language such as Java. We thus use the GCJ compiler to over come this problem.

Using Java and the GCJ complier we are able to gain the advantages offered to the programmer by Java as well as still having an executable program that is compiled to native machine code.

4.1.4 Linux

Because Qt, OpenGL and Java are all platform independent both Linux and Windows are would be appropriate operating systems. The Rhodes University Virtual Reality Special Interest Group (VRSIG) uses the Rhodes Virtual Realty System (RhoVeR). RhoVeR is currently on version Greatdane. RhoVeR is a parallel and distributed Virtual Reality (VR) system created by the Rhodes University Computer Science Department. RhoVeR is used in order to allow each of the members of the Rhodes VRSIG can develop their own application concurrently. RhoVeR operates uses the Linux Operating System. For this reason, the Linux system was used as opposed to Windows.

4.1.5 Integration of Qt, Java, OpenGL and Greatdane

Greatdane offers support for a virtual environment and programming 3D graphics. A number of classes are encapsulated within the Greatdane environment The following section discusses the classes, relevant to the applications developed for this project, and provides the reader with insight to the integration of the various programming languages used.

C++ code and Qt have been used to implement the view component of the MVC paradigm as discussed in section 3.3. Qt provides a signal/slot concept which allows for the communication between objects during runtime. When the user performs an event a signal is created which is in turn connected to a slot. These slots are used to call the various functions implemented in the model component.

The model has been implemented using Java. In order for these two components to communicate, native calls from the Java code are made to call and initialize the system interface.

4.2 Implementation Motion Capture Application

4.2.1 Motion Capture Component

The system requires two data sources. One for the viewing of data from the camera and another for the viewing of the saved images. Greatdane supplies a Device Component class that can be easily used for getting data without the complexities involved in watching ports and polling devices directly. Sources and sinks are connected to this component and thus data can then be easily transferred between it and the target component. We create a MotionCaptureComponent class which extends the DaviceComponent. As the user toggles between camera view and viewing of the captured images we connect the Component to either the port to which the camera has been connected or to the Vector holding the captured frames.

The images that have been captured off of the video are initially only kept in a Java Vector, this is so as to improve performance. The performance is improved because there is no need for the retrieval and saving of the images to and from disk. A vector also allows the programmer to dynamically add items whilst in run time, thus there is no need to specify the initial size for the capture data.

The order of the images captured is an essential property as they are to be used later for the extraction of motion data. Thus we see the advantages of using a Vector again. The Vector index is used for positional information for each frame. When saving the application uses the specified file name and automatically saves every frame with this name and its position in the sequence concatenated to it. This ensures that the ordering of the images is correct. The code segment in listing 4.1 below shows how this has been implemented.

```
for (int pos=0; pos<savedImages.size(); pos++) // savedImages is the Vector
                                               // containing the images that have been captured
 {
   VideoImage imageToBeSaved;
   imageToBeSaved = (VideoImage) savedImages.elementAt(pos);
   imageToBeSaved.writeppm(saveFileName+ pos);        //file name and its position
}
```

Listing 4.1: Code Segment for Saving Images to Disk

4.2.2 Mark Up Component

The Mark Up Component needs to gather information for the 3D skeleton from the 2D images as well as being able to render a 2D visualization of this skeleton on each of the frame for the user to be able to move each of the joints.

A clickable object has been implemented. This object allows for images to be loaded and displayed to the user. The object has properties that will deal with user clicks and its methods will communicate with the skeleton object in order to build up the skeleton using the users clicks. The user is able to perform various clicks on the images that are loaded. The position of the mouse clicks is used to calculate the x and y co-ordinates for each joint added to the skeleton, the user is prompted with an input box to specify the name of the joint and its value for the z co-ordinate.

With each click made by the user the skeleton data structure is built up. Although only the root joint for a .BVH file needs positional data, we obtain positional data as well as the rotations and orientations for each of the joints. The positional data for the joints is used to render the 2D visualisation of the skeleton on each of the frames as they are loaded.

On the first frame/image loaded the user specifies the initial pose of the skeleton, on the subsequent frames/images loaded the user is able to move these joints into their new positions and the application calculates the rotation and orientation values for each of these joints. The user interface has been designed in such a way so as to eliminate as much of the thought process needed, for building the skeleton, from the user as possible. The very first click made is always assumed to be the root joint. Each click proceeding this assumes the previous joint added to be its parent joint, until an end node is added. End nodes are

determined by joints who have the name “End Site”. Any click added after an end node will prompt the users to select a parent joint, from the list of available joints that are already in existence. Each joint is assigned an ID. This ID is used to associate its orientation and rotational information to it. This is all controlled by the Qt mousePressEvent() procedure which has been over loaded for the purposes of this project. The listing below shows how this process is governed.

```
//overloaded function form Qt
void mousePressEvent (QMouseEvent * click)
 {
   if (lastimage != NULL)
    {
      if (firstPic)
       {  Obtain the x and y co-ordinates for the mouse click
         1) if the click is first, it is assumed to be the root
         2) Subsequent joints assume the joint added immediately before it to be its parent
              joint
         3) Joints added that are not root joints, and have had an end node aded directly
            before the require the user to select a parent joint from the existing joints

         once all of the attributes for a joint have been found the model is updated and a new
       joint is added to th skeleton.

         counter keeps track of the number of joints added and their ID's
       }//end firstPic
      else if (pointSelected) //for subsequent images loaded
       {  //you want to now set the values for orientation
         int x = click->x();
         int y = click->y();
         emit orienClick(x, y, jointID, frameNum);
       }
 }//end mousePressEvent()
```

Listing 4.2: Code Segment for Adding Joints to the Skeleton
and Code to Initiate the Setting of the Orientation of the Joints

Once the entire skeleton has been marked up the user is able to click a button labelled next. This will invoke the next image in the sequence to be loaded. The manner in which these images was saved from the motion capture component ensures that the images are loaded in order. On any frame that is loaded after the first frame the user is unable to add joints to the skeleton and is only able to set the new positions for the joints, these positions are used to calculate the orientations and rotations.

The calculation of the orientations and rotations is achieved by only looking at the rotation around the z-

axis. This is because we are using a 2D image. The rotations in the x and y axises are kept at zero. Each joint regards the root joint (shown in green in figure 4.1) as its parent co-ordinate and itself as the child co-ordinate (the yellow joints in figure 4.1). These points are used to calculate the distances (x and y) completing the triangle formed between the joints, as shown in gray in figure 4.1. The function atan2(x, y) is used to calculate the arc tangent of the two variable x and y. This is the angle shown in orange in figure 4.1. The signs for each of the arguments is used to determined the quadrant of the result.. These angles are used to set the orientation and rotation for each joint. We recursively call this function until all of the joints have had their orientations calculated.

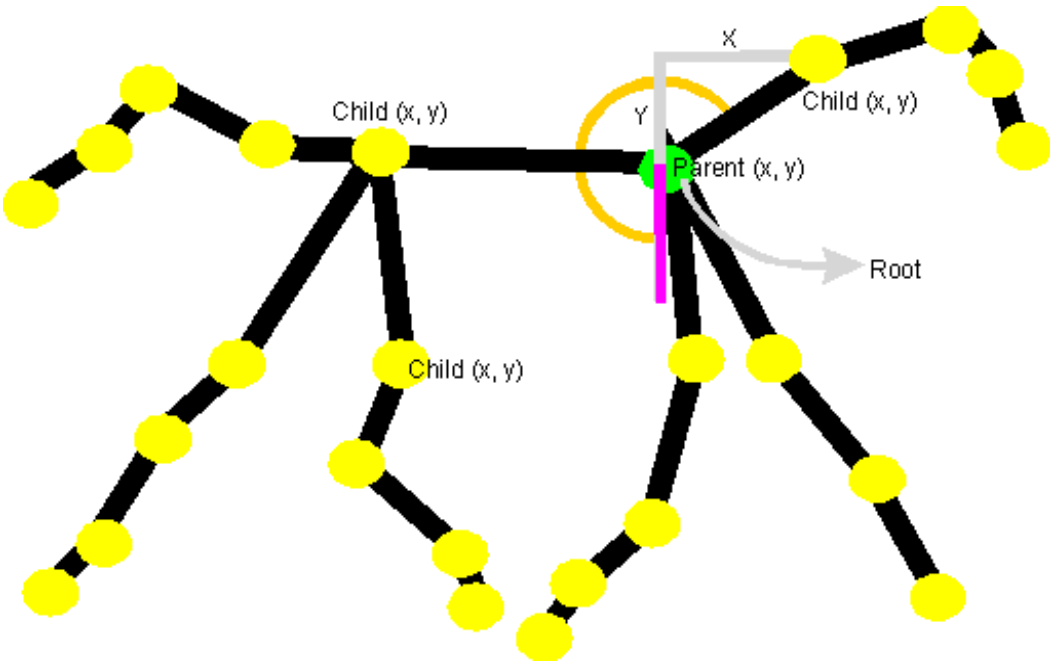

Figure 4.1: Diagram representing the Calculation of the Orientation for each Joint

Only the orientation for the z axis is calculated. We set the orientations for the x and y axis to be zero. This is because of the nature of the 2D image that is being use. On the 2D image we see the x and y axis thus any movement seen is around the z axis. The method used to implement this can be seen below in listing 4.3.

```
void setOrien(int x, int y, int jointID, int frameNum)
 {  //convert Radieans to Degrees ( * 180/3.141592654)
   double xAngle = 0, yAngle = 0, zAngle = 0;

   int rootX, rootY, childX, childY;

   1) get positional data for the root joint and the current child joint
   2) calculate the x and y distances between these tow points
   3) calculate the orientation around the z axis using atan2(a, y)

   4) set the orientation for that joint
 }
```

Listing 4.3: Implementation of the Calculation for the Orientations

4.3 Implementation Model Viewer Application

As has been described in detail in section 3.1.2 the Model Viewer Application is required to be able to read and interpret motion data file formats. For reasons discussed in section 2.2.4 this project concentrates on the BioVision .BVH file format.

A parser has been developed to handle the reading in of data from a .BVH file format. The parser is based on a Cocol description of a .BVH file. Listing 4.1 below is the Cocol description of the .BVH file format. From this description we see that all white space in the files is ignored. We also make the assumption that the ordering of the channels for each joint is in the order Zrotation, Xrotation,Yrotation,. This order defines the sequence in which the rotational operations need to be processed in the playback. Although the specification for the file states that they may be in any order, however this research has yet to encountered a BVH file whose order differs from this. The description describes the .BVH file reader that is intended to process the contents of a .BVH file. The parser is developed to recognise and interpret the structure of the .BVH file. "The 'CHARACTERS' section describes the set of characters that can appear in" (Terry, 2000:62) the strings within the file. "The 'TOKENS' section describes the valid forms" (Terry, 2000:62) that numbers and words may take and the 'PRODUCTIONS' section describes the structure of the file itself.

The method of parsing that has been used is top-down parsing by recursive descent. The parser in effect starts at the goal symbol, which in this case is the word "HIERARCHY", and tries to parse the rest of the

file by applying a set of appropriate production rules to the symbols being read in. This is done by looking at the next terminal in the string that it has been given to parse. We use a special case of parsing called LL(1) parsing. This terminology means that we are scanning the input string from left to right (first L), and applying the productions rules to the leftmost non-terminal string that we are trying to verify (second L), and looking only as far ahead as one terminal in the input string, in order to build up the skeleton information and help decide which production to apply next. Listing 4.4 shows the Cocol description of a BVH file.

```
COMPILER BVHReader
CHARACTERS
 digit    = "0123456789" .
 letter   = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz" .
 tab      = CHR(9).
 lf       = CHR(10).
 cr       = CHR(13).
 Xpos     = "Xposition".
 Ypos     = "Yposition".
 Zpos     = "Zposition".
 Xrot     = "Xrotation".
 Yrot     = "Yrotation".
 Zrot     = "Zrotation".

IGNORE tab + lf + cr

TOKENS
 name      = letter { letter }.
 number    = digit { digit }.

PRODUCTIONS
 BVHReader = "HIERACHY" Joints Motion.
 Joints    = ((("JOINT" | "ROOT")JointDef) | ("End" EndDef)).
 JointDef  = "{" Offset Channels {Joint} "}".
 EndDef    = "Site" "{" Offset "}".
 Offset    = "OFFSET" number number number.
 Channels  = "CHANNELS" number (number * ChannelType).
 CannelType = Xrot | Zrot | Yrot | Xpos | Ypos | Zpos.

 Motion    = "MOTION" "Frames:" number FrmTime MotionValues.
 FrmTime   = "Frame Time:" number.
 MotionValues = NumberOfChannels * FrmTime.
END BVHReader
```

Listing 4.4 : Cocol Description of a Motion Data .BVH file format

Each production in the BVH description deals with a separate section of the .BVH file. As we see from the description in section 2.2.1 this file is very recursive in nature. Thus we have used a recursive decent parser as discussed above. Each production in the description is mapped to a separate function. Each function will parse the input string until it has read in and stored that data into the appropriate structures,

it will then return the next terminal of the input string to the function for where it was called.
As each terminal is parsed by the parser the information is used to build up the skeleton hierarchy and the motion associated with it.

Vigorous error checking has been put into place. The parser will check for the correct format of the file, it will check the formatting of number type that have been read in. Any error that is encounter will be reported to the user and the file will not be loaded.

The rendering of the skeleton to the 3D virtual environment is controlled by a single thread. This thread constantly check the status of a 'playing mode' flag. The flag is set by the user from the Graphical User Interface (GUI). The user is able to view the playback of the animation continuously or frame by frame. They are also able to choose the direction of playback. Playback can either be forwards or backwards. This is easily implemented by having a single variable that constantly keeps track of the frame at which is being displayed to the user. This will either be incremented or decremented depending on the actions of the user.

The drawing of the skeleton is achieved by traversal of the skeleton data structure. We always begin with the root node and then recursively move done each of its children sub-trees until and end node is encountered. An end node is identified by joints that have the name "End Site". A drawSkeleton() function has been written which will translate and rotate the joint that it has been called with before calling the drawJoint() function which will actually draw the joint.

Chapter 5 Tests and Results

5.1 Motion Capture Application

5.1.1 Motion Capture Component

A collection of various motion sequences of animals was performed, with the use of a single hand held camera. We concentrated on capturing motion for horses. Horses offered varied sequences of motion. We have captured horses trotting, galloping and performing jumps. Motion for other quadruped animals such as buck, wildebeest, cats and dogs was also captured and digitized. For each of the sequences captured, the animals where completely unrestricted, no special positioning of the camera or animal was done. We captured motion sequences of animals that would have been impossible in a studio environment. The figures over the page show numerous screen shots of the various animal motion sequences that have been captured.

Figure 5.1: Digitized Video Data of Animal Motion

(a) Horse Trotting **(b)** Horse Walking **(c)** Playful Cat **(d)** Dog

5.1.2 *Mark Up Component*

The still video images captured using the above component can be used to build up the various skeleton hierarchies for these animals. The application uses a value specified by the user for the z co-ordinate. This method is not the best solution and there is much more research that can be taken in the area. Only the orientation in the z-plane is calculated, and the x and y orientation are kept at zero. This offers very limited extraction of motion. The figures that follow show the building up of a skeleton on each of the frames captures.

Figure 5.2: Mark Up Application Adding the Root Joint to a Skeleton.
(The initial click on the first image loaded is assumed to be the root joint)

Figure 5.3: Mark Up Application Adding the Joints to a Skeleton.

(Subsequent clicks assume the previous joint added to be their parent joint, unless and end node has been added, end nodes are identified by having the name 'End Site')

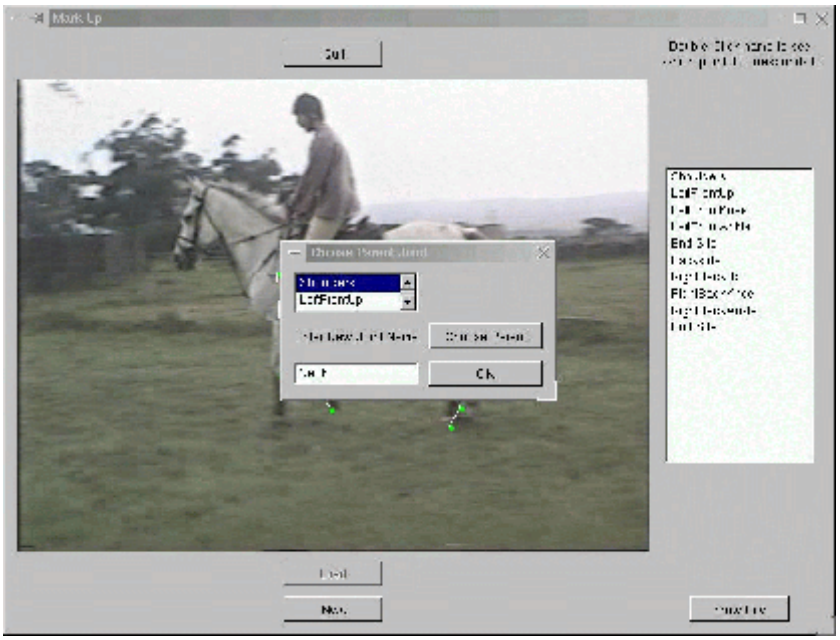

Figure 5.4: Mark Up Application Allowing a Parent Joint to be chosen for a New Joint Added to the Skeleton.

(Each joint added after an end node ('End Site') joint forces the user to select a parent joint for it from the existing joints added)

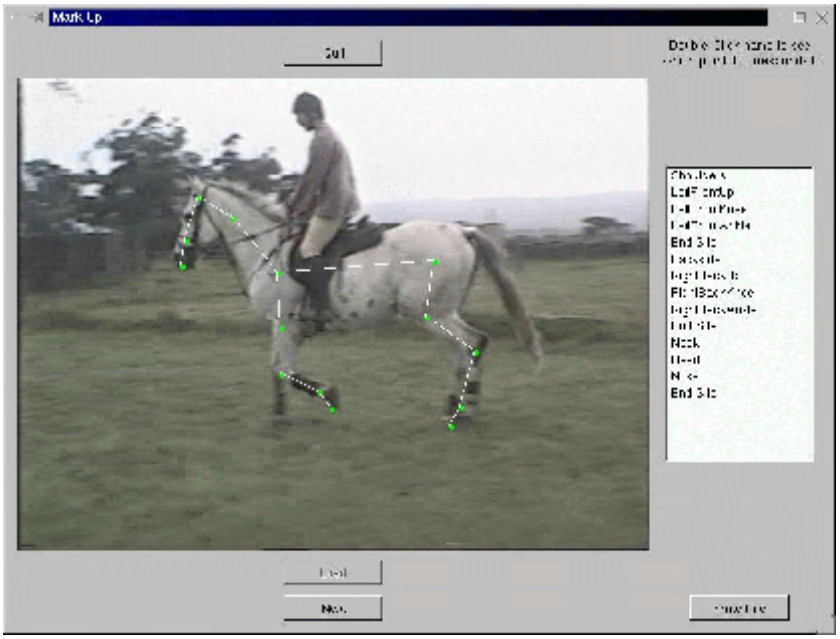

Figure 5.5: Mark Up Application the Mark Up of a Complete Skeleton.

5.2 Model Viewer Application

Various sample files have been freely available and downloadable from the internet. We have collected a variety of files, most depicting human motion. The Model Viewer application successfully parses these files and uses their information to model and animate various articulated skeletons. We have animated human skeletons performing dance sequences, playing basketball, performing karate and a horse skeleton. Below are various screen shots for a number of these skeletons.

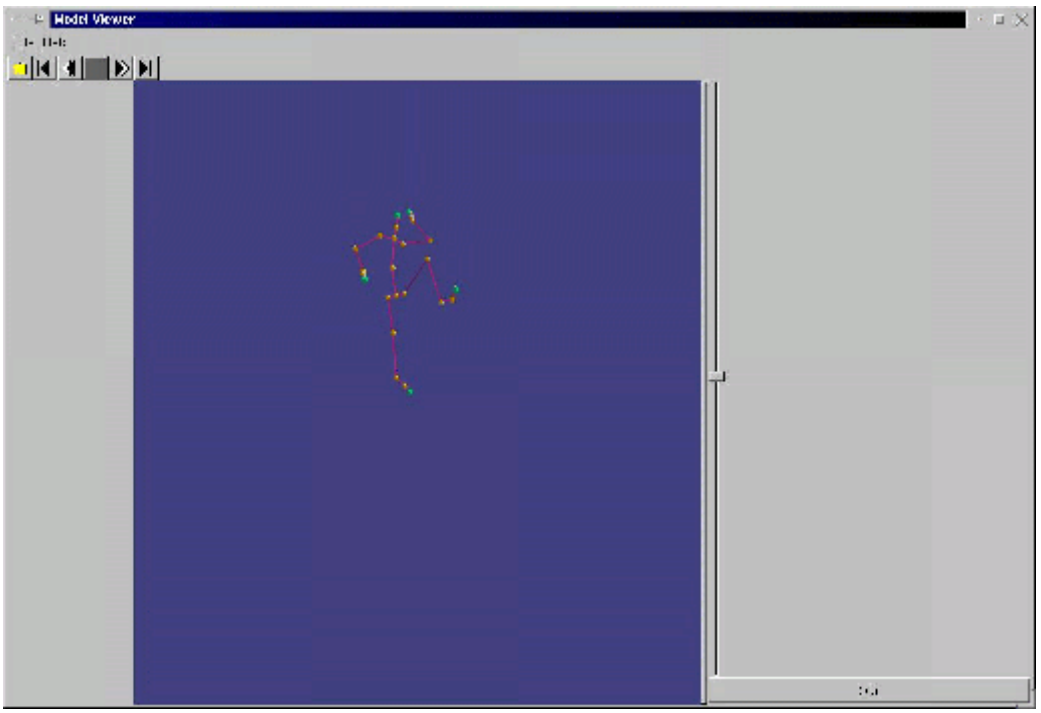

Figure 5.6: Model Viewer Application Animating a Human Avatar Performing a Sequence of Karate Movements.

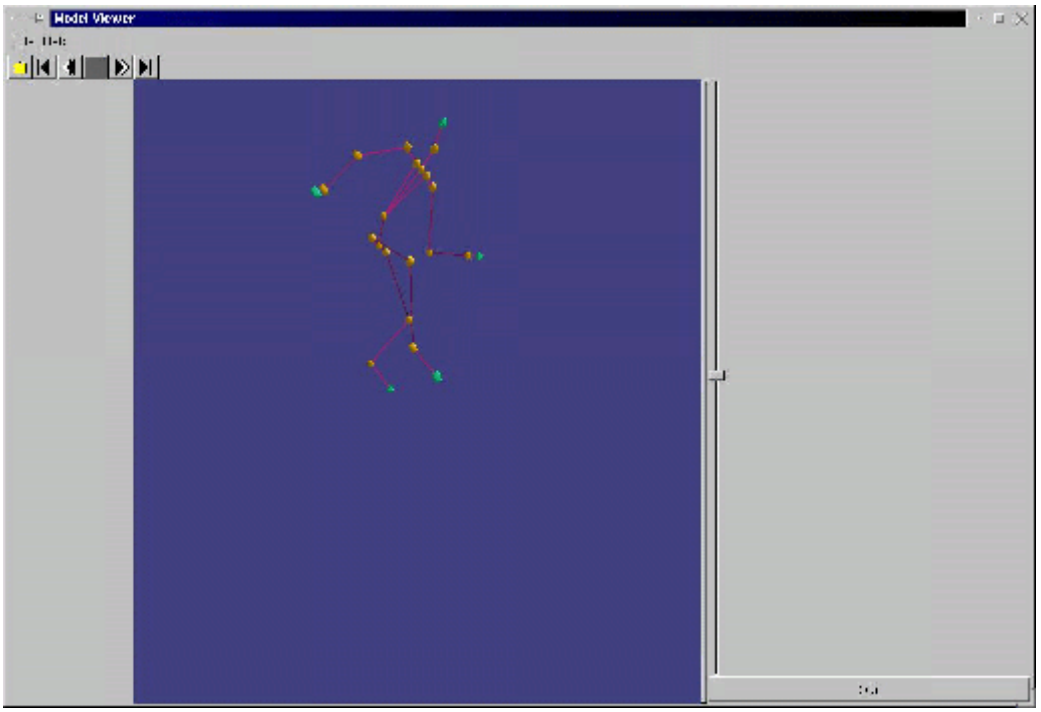

Figure 5.7: Model Viewer Application Animating a Human Avatar Bouncing a Basketball

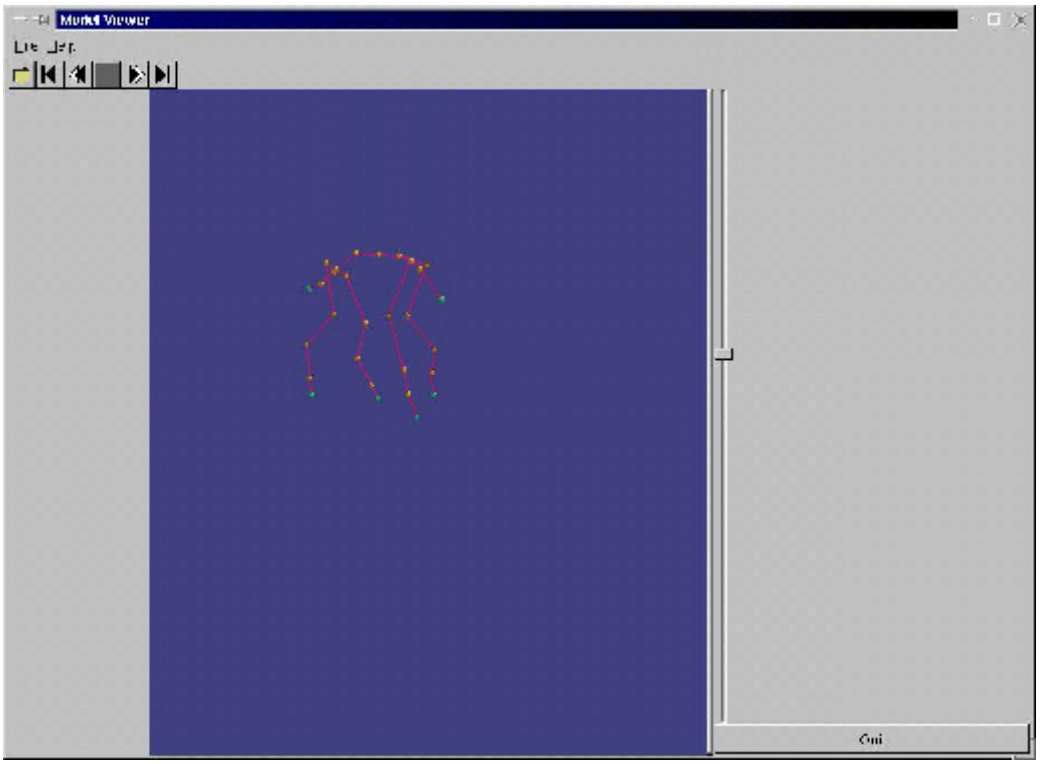

Figure 5.8: Model Viewer Application Animating a Horse Avatar.

Chapter 6 Conclusion

This paper presents an uncomplicated method of motion capture for quadrupeds and other living creatures. The technique used allows for a very simple form of motion capture which is cheap and versatile. The approach that has been used does not require any markers or any other devices attached to the body of the performer. We are able to film performers in their natural habit, where there is no specialised equipment or set up proses required. Motion capture data is has been used to model and animate articulated skeleton avatars in a three dimensional environment.

Two applications have been built based on the proposed requirements and user interface. They have both achieved promising results, however there is room for improvement in both of them.

The first of these applications is the Motion Capture application, used for the capturing of still images from video data, recorded with a single hand held camera, which can then be used for the annotation of the performers skeleton and motion. This application demonstrate recordings of various animals outside of the studio environment. The calculation of the missing dimension form the two dimensional video images has relied on careful calculation on the users part. The application is only able to calculate the angles for orientation in the z-plane. This application may be enhanced by adding the functionality of viewing various video formats. Such formats might include .mov, .avi and .mpeg. This would enable the gathering of motion with a single hand held camera as well as being able to download and gather motion scenes from video clips of the internet.

The second application, the Model Viewer, manages to parse motion data file formats very effectively. Using the parsed information, articulated skeleton models are developed and motion data mapped to them. The application manages to model various skeletal avatars as well as animate these skeletons in a three dimensional environment. Although this application has had very promising results, here to there is much room for improvement.

Both applications concentrate on the use of the BioVision .BVH file format. The system would be more

efficient and useful is the ability to work with many file formats where to be added.

This paper has provided the reader with a motion capture method for animals and other moving objects that have very complex motion, and where it is not possible to attach markers or other equipment to the actors. The system manages to use the information in motion data file to animate articulated skeleton figures in three dimensional environments.

References

BioVision. 1998 — **BioVision Optimum Human Performance Center** [On-Line] Available: http://www.biovision.com/

BioVision. 1998 — **Character Studio 2.0 BVH File Format Specification** [On-Line] Available: http://www.biovision.com/bvh.html

CASANUEVA, L. 1999 — **Minimal Motion Capture with Inverse Kinematics for Articulated Human Figure** Animation. [On-line]. Available: http://www.cs.ru.ac.za/vrsig/techdocs.html

DA SILVA, F. W, VELHO, L, CAVALCANTI, P. R. and GOMES, J. 1997 — **An Architecture for Motion Capture Based Animation.** [On-line]. Available: http://www.visgraf.impa.br/RefBib/Data/PS_PDF/silva97b/sib97.pdf

FURNISS, M. 1999 — **Motion Capture.** [On-line]. Available: http://media-in-transition.mit.edu/articles/furniss.html

GARRESON, 2000 — **Maximizing Your Investment With Motion-Capture Equipment.** [On-line]. Available: http://www.ascension-tech.com/news/articles/maximize.html

Gypsy Motion Capture. 2001 — **Gypsy Motion Capture** [On-Line] Available: http://www.metamotion.com/gypsy-motion-capture-system/gypsy-motion-capture-system.htm

KENNY, K. B. 1995 — **Tkauxlib: An Auxiliary Library for TK.** [On-line]. Available: http://ce-toolkit.crd.ge.com/tkauxlib/tkauxlib_1.html

LANDER, J. 1998 — **Working with Motion Capture File Formats**. [On-line]. Available: http://www.darwin3d.com/gdm1998.htm#gdm0198

LocoMotion Studios. 2001 — **Locomotion Studios** [On-Line] Available: http://www.locomotionstudios.com/frameset.html

META MOTION PAGES. 2001 — **Gypsy Motion Capture FAQ.** [On-line]. Available: http://www.metamotion.com/faq.htm

NEIDER, J., DAVIS, T, and WOO, M.. 1997. — **OpenGL Programming Guide.** [On-line]. Available: http://graphics.eecs.wsu.edu/cpts442/book.html

NOLDEN, R.1999 — **The KDevelop Programming Handbook.** [On-line]. Available: http://www.kdevelop.org/doc/programming/index.html#toc2

SCHAFER, M. 1995 — **Acclaim Skeleton File Definition V1.10.** [On-line]. Available: http://www.biomechanics-inc.com/software/asfamcspec.html

STURMAN, D. J 1994 — **A Brief History of Motion Capture for Computer Character Animation.** [On-line]. Available: http://www.css.tayloru.edu/instrmat/graphics/hypgraph/animation/motion_cap

ture/history1.htm

THINGVOLD, J. 1999 **Jeff's notes on motion capture file formats.** [On-line]. Available: http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/MoCapTOC.html

TRAGER, W 1994 **An overview of current input systems.** [On-line]. Available: http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/motion_capture/motion_optical.htm#An%20overview%20of%20current%20input%20systems

TUSTIN, C.A. 1999 **Locomotion Studios Horses Around with Vicon 8** [On-Line] Available: http://www.metrics.co.uk/animation/press/previous/locohorse.html

Unknown A. 2001 **OpenGL Overview.** [On-line]. Available: http://www.opengl.org/About/About.html

Unknown B. 2001 **OpenGL OverViews.** [On-line]. Available: http://www.opengl.org/Documentation/Documentation.html

Unknown C. 2001 **GLUT OpenGL Utility Toolkit.** [On-line]. Available: http://www.opengl.org/Documentation/GLUT.html

Unknown D. 2001 **Qt Documentation.** [On-line]. Available: http://www.trolltech.com/products/qt/qt.html

Unknown E. 2001 **History of Motion Capture.** [On-line]. Available: http://www.bergen.org/MotionCapture/

Unknown F. 2001 **The GNU Compiler for the Java Programming Language** [On-Line] Available: http://gcc.gnu.org/java/

Vicon Motion Capture. 2001 **Vicon Motion Capture.** [On-Line] Available: http://www.metrics.co.uk/animation/