Contents

Contents

List of Figures iii

List of Tables iv

1 Introduction 2
1.1 Introduction to 3D character animation 2
1.2 Objective 6
1.3 Contributions 6
1.4 Overview of the rest of the thesis 7
1.5 Summary 8

2 Related work 9
2.1 Overview 9
2.2 Modelling 9
2.3 Skeleton creation 11
2.4 Rigging 12
2.5 Animation 14
2.5.1 Key-frame animation 14
2.5.2 Inverse Kinematics 15
2.5.3 Motion capture 15
2.5.4 Alternative animation methods 16
2.6 Clothing 16
2.7 File formats 17
2.7.1 Cal3D 18
2.7.2 Collada 18
2.7.3 FBX 18
2.8 Summary 18

3 Rigging 20
3.1 Overview .................................................. 20
3.2 Skeleton Embedding ................................. 21
  3.2.1 Preparation process .......................... 21
  3.2.2 Embedding process ........................... 24
3.3 Skin Attachment ...................................... 25
3.4 Extending Pinocchio ............................... 26
  3.4.1 FBX integration ................................. 26
  3.4.2 User interaction ............................... 27
3.5 Summary .................................................. 27

4 Clothing ..................................................... 29
  4.1 Overview ............................................ 29
  4.2 Clothes creation .................................. 30
  4.3 Attachment to the character ...................... 30
  4.4 FBX integration and User Interaction ............... 31
  4.5 Summary ............................................ 32

5 Animation .................................................... 33
  5.1 Overview ............................................ 33
  5.2 OptiTrack .......................................... 33
    5.2.1 System Setup ................................ 33
    5.2.2 Calibration .................................. 34
    5.2.3 Motion Capture ............................... 34
  5.3 Animating the rigged character ................. 36
  5.4 Summary ............................................ 37

6 Results ......................................................... 38

7 Conclusions ................................................ 43

Bibliography ................................................... 44
List of Figures

1.1 Some Second Life avatars. ................................. 3
1.2 Emergency situations in virtual reality. .................... 4
1.3 Virtual Crash Dummy. .................................. 5

2.1 Blender example ........................................ 10
2.2 FiberMesh result ........................................ 10
2.3 Skeleton placed inside a scanned woman model .......... 12
2.4 Skin marked with colours simulating the bone weights. .... 13
2.5 Image from Animating Scanned Human Models .......... 14
2.6 Motion Capture ........................................ 16
2.7 Motion Doodles example sequence ....................... 17

3.1 Pinocchio skeleton embedding process .................... 20
3.2 Pinocchio process ....................................... 22
3.3 Simplified skeleton ...................................... 23
3.4 Embedding optimisation ................................ 24
3.5 Temperature distribution ................................ 25
3.6 Bone rotation ........................................... 26

4.1 Clothing process ......................................... 29
4.2 Mesh of a pair of jeans ................................ 30
4.3 Example of a vertex discarded by the cloth rigging algorithm. .... 31
4.4 Rigged Clothes ......................................... 32

5.1 Three steps of the OptiTrack calibration process ......... 35
5.2 Default skeleton for the OptiTrack system ............... 36

6.1 Skeleton weights applied on clothes. ....................... 40
6.2 Test: the process took more than usual, and the embedding is incorrect. 41
6.3 The skeleton is completely rotated, plus the feet joints are incorrect. 41
6.4 The joints of the arms are incorrect. ...................... 42
List of Tables

6.1 Test: Skeleton rotations on the X axis. ......................... 39
6.2 Test: Skeleton rotations on the Y axis. ......................... 39
6.3 Test: Skeleton rotations on the Z axis. ......................... 39
6.4 Test: Arm rotation. ............................................. 39
I would like to thank Dr. Bernhard Spanlang for his patience during the long development stage and the inestimable support and guidance given throughout the whole project. Also many thanks to Prof. Mel Slater, Dr. Marta Fairén, and Dr. Frederic Perez for reviewing this thesis; and to Clàudia, my family, friends and colleagues for all the support and love given while I was busy.

Barcelona June 25th 2008

Marc Boquet Bertran
Abstract

Using existing 3D modelling tools to create Avatars that can be animated is a cumbersome manual task that can take a modeller days or even weeks.

In a process often referred to as rigging, a modeller places a skeleton inside a 3D mesh and maps skeletal segments to the corresponding parts of the mesh to create a character that smoothly deforms if the skeleton is moved.

The focus of this master’s thesis is to create and evaluate methods that can help a modeller in the rigging process and that can also carry out the process fully automatically. Currently, if the resulting quality is an important requirement, the skeleton fitting and skin attaching are done by hand. Recently, a method has been published that automates the process given a 3D mesh and a skeleton. However, this method only works on a single manifold mesh and it does not work well on all mesh shapes. In addition, this method allows little or no hand-tuning when it comes to correcting problems in the mapping to achieve better results.

Therefore, the aim of this thesis is to find a method that builds on an automatic method but allowing the user to easily modify and guide the process. We aim to improve the automatic method, firstly, by evaluating different optimisation methods used to carry out skeleton insertion and surface mapping and, secondly, by extending the method to handle non-manifold meshes to allow us to rig not only a character’s skin but also characters dressed in one or more layers of virtual clothing. Using automatically or manually extracted body landmarks to identify the different parts of a character’s anatomy we aim to guide our method to improve the quality of the rigging.

The resulting data will be ready to use with motion capture data in a real time animation library, in order to allow us to evaluate the quality of the animations in an interactive immersive 3D virtual environment.
Chapter 1

Introduction

The objective of this Master’s Thesis is to provide modellers with a comprehensive method of rigging, clothing and animation of virtual characters. Rigging is the process of creating the necessary information for a static character to be animated. It will be described in detail through the following chapters.

This chapter starts with a short introduction to the basics of 3D character animation, its use in various fields and the different requirements that each of these fields present with respect to the resulting animated virtual human, often referred to as an ‘avatar’. In addition, a description of the process of creating avatars is given. Afterwards, the objective of this thesis is presented, as well as a brief summary of the contributions made. The chapter finishes with an overview of the rest of the chapters.

1.1 Introduction to 3D character animation

This section reviews some of the many areas that take advantage of 3D avatars, focusing on the different requirements of every field. In addition, the creation process of a 3D character will be described.

Application areas

One of the main fields where the presence of animated 3D characters is more noticeable is the film industry. The growing number of movies produced using computer graphics contributes to innovations in the field. Some film making companies such as Pixar have research groups working on new techniques of modelling, animation or illumination among others, and the improvements made by this industry often benefit other fields. The requirements for movies are usually
quite high, the characters must be nearly perfect, both for their appearance and their movements.

The presence of avatars has also been growing in the other major industry of entertainment, video games. For many years now, there are an increasing number of games that make use of 3D graphics, which require in most cases the creation of new animated characters from scratch. As in the case of movies, the characters in video games often need to look very real while permitting interactive frame rates, and their movements have to be sufficiently accurate.

Online worlds, such as Second Life [49], represent another approach to the subject. While still being considered as a type of video games, its requirements with respect to the avatars differ. These applications must allow quite a high number of people to connect together from any part of the world, and the movement of the avatars must be close to real time. The fact that the data must be able to travel through the internet may require the 3D models to have a low resolution, otherwise the amount of data transferred could be prohibitive for the application to run smoothly. Figure 1.1 shows Second Life characters; the low polygon level is noticeable, most of the detail is offered by the textures.

![Figure 1.1: Some Second Life avatars.](image)

Another field where there is a heavy use of animated characters is virtual reality, mostly in the research world. There are many projects that include 3D avatars, from psychological experiments [14, 30, 33] to telecommunication [11] or crowds rendering [36]. Figure 1.2 shows an example of a psychological experiment developed by Spanlang et al. [33] which the goal of which was to identify the impact of interactivity of virtual characters on participants in an emergency
situation. Furthermore, some recent researches [31] have demonstrated that in virtual reality visual realism does not influence as much as it may seem with respect to the user perception.

![Emergency situations in virtual reality.](image1.png)

There are many other industries that take advantage of the emergence of three-dimensional animated characters. Some aeroplane companies use virtual characters to give safety instructions, the motor industry uses virtual crash test dummies before performing expensive tests with real cars, as shown in Figure 1.3, where the physics-driven approach is manifested. Many architects show their creations incorporating 3D avatars to add realism, which is also the case for the design process of clothes designers. These are only a few examples of the increasing number of real world applications for virtual characters.

**Creation process**

Three-dimensional character animation for their use in virtual environments is currently an arduous and highly time-consuming manual task. The character geometry, often represented by a 3D triangle mesh, can be created from scratch by using current modelling software such as *3ds Max* [42] or *Blender* [44]. It can also be acquired using body scan methods, from which the 3D representation of a real body is obtained [9]. Other methods such as *Teddy* [18] allow easy creation of 3D models interactively using a sketching method. These and other methods are reviewed in the related work chapter, section 2.2.

Once the geometry is created, the next step is to define its internal skeletal structure, frequently described as a graph where articulations are shown as vertices and bones are represented by edges. This representation allows the user to move and rotate the different parts of the skeleton in a comfortable way. Never-
theless, the skeleton is not attached to the actual character, so there is a need to specify which parts of the surface are attached to which bone, a process called rigging or skinning. Manual rigging is a tedious task, which requires assigning weights to all the bones with respect to every polygon of the mesh. Fortunately there are some automatic methods to establish the rigging, such as Pinocchio [2].

Once the rigging is done, the character moves along with the skeleton, but it may need clothes, and these will have to move too. For this step, the tool created by Spanlang et al. [34] was used, which is capable of computing adjusted clothes to a given model. The next step is to attach the clothes to the skin in order to be able to animate them too, using the previously computed weight information from the bones and the skin. After these steps, the user should be able to animate the model, using motion-capture or other animation methods, as explained in chapter 5.

**Different needs, different solutions**

As we have seen, there are many areas where virtual characters may be needed. Depending on the field of work, however, the final quality of this avatar may be more or less important compared within the context of the whole project. For example, in most of the current video games, their final success depends in a great measure on the quality of their graphics and animations. This fact justifies the high number of hours spent on the creation of the characters. In some other fields, however, the job of modelling and animating virtual characters is not that
significant for the success of the project. For instance, researchers in virtual reality or other areas may need easy-to-create characters in order to perform the actual study. These characters have to be animated, but a quality loss on the final animation may be justified by the saved amount of time. The current automatic techniques do not offer a global approach that minimises the manual effort needed to accomplish the whole process, thus compelling modellers to require several days working on the animation of a single avatar.

1.2 Objective

This master’s thesis presents a fast and easy-to-use method for animating 3D characters automatically while allowing the user to tweak the results if necessary. This method should be able to automatically provide a correct result in most of the cases, as well as it should offer an easy way of manual tuning of the process when necessary. The result should be saved in a well known file format, directly usable in most professional modelling and animation systems. There are some existing methods of automatic rigging, but user interaction is difficult, and their integration with current modelling software is nonexistent. This process is applicable to polygonal biped models, either human or fantasy characters. It could also be applied to other models such as quadruped, but unfortunately the lack of free 3d models for these kind of characters has prevented us to test it.

1.3 Contributions

The following contributions have been made in this thesis:

- An integration of the Pinocchio automatic rigging method with the FBX file format [50], allowing to incorporate the results in some of the mostly used modelling and animation software.

- Attaching clothes to the rigged character. An existing technique of cloth creation has been used to obtain static clothes, and the deformation has been applied to them so the clothes move along with the character. In order to achieve more realistic results, textures have been added to the clothes.

- A separation of the different steps of the rigging process in order to let the user review or modify the automatic result after each step. It is also allowed to run each step of the process individually.

- The resulting character has been animated using motion-capture. We provide and discuss an analysis of the different results based on various parameters used during the rigging process.
1.4 Overview of the rest of the thesis

The structure of the rest of this thesis is outlined in the following.

Chapter 2: Related work

In this chapter existing methods of rigging and animation of 3D characters are reviewed. The different parts of the process –skeletal placement, rigging, clothing and animation– are examined separately, and the advantages and disadvantages of the different approaches are analysed. Finally, one of these methods is chosen as a basis for the thesis.

Chapter 3: Rigging

In this chapter the automatic rigging process is explained and reviewed in detail. First, the fundamentals of the mathematical theory behind the Pinocchio library \[^2\] are given, then some implementation details are shown. Furthermore, we describe the integration of this existing library with our software, using the Software Development Kit (SDK) of the FBX file format \[^50\].

Chapter 4: Clothing

In this chapter the method used for the creation of the clothes is described. The clothing system from Spanlang and others \[^34\] is used to obtain static clothes virtually tried on geometry obtained by a whole body scanner. The clothes must be attached to the skin in order to allow their deformation along with the skeletal model. The procedure to find the correct attachment for the clothes meshes is explained.

Chapter 5: Animation

In order to review the results of the process, realistic animations are needed. Several animations have been captured and applied to the resulting models. This chapter gives details of the motion-capture technique that has been used to obtain the animations that will be applied to the character, the Optitrack software Arena \[^32\].

Chapter 6: Results

Some results are shown and analysed, searching for inconsistencies in the various animations and different configuration options used. We present a set of techniques to identify these inconsistencies in order to avoid them.
Chapter 7: Conclusions and Future Work

This chapter presents the conclusions of the thesis. Also possible lines of further research are suggested.

1.5 Summary

In this chapter an overview of the different fields using 3D characters has been given, as well as a short description of their creation process and a review of some of the creation techniques. Also, the objective of this thesis has been presented, along with the contributions made. The chapter has ended with an overview of the rest of the chapters.

The next chapter will present the existing research work in the field, briefly reviewing the most important techniques with respect to this thesis.
Chapter 2

Related work

This chapter describes some of the existing methods of character modelling and animation in 3D, stating their advantages and disadvantages. Solving some of these disadvantages is an objective of this thesis, and this chapter briefly describes how they are addressed.

2.1 Overview

Professional modellers and animators use 3D software such as Maya [43], 3ds Max [42] or LightWave [47] for the creation of new characters, and they tend to do a great part of the process manually. This situation is not likely to change in the next few years, as this software gives modellers the freedom they need to manually create high-detail models for their use in movies or video games, and most of these 3D artists prefer to sculpt their creations contributing with their human touch, rather than having a machine doing all the work [10]. However, this approach is not affordable by amateur animators or researchers that may need an easy-to-create animated character; therefore some other approaches have appeared to provide a faster way of creating and animating an avatar, even though the final result may not be as refined as if performed by a skilled modeller.

2.2 Modelling

The first step in the creation of an animated character is modelling. Starting from scratch the objective is to obtain a three-dimensional triangle mesh representing a real-life model, in this case a person. A brief review of some of the several existing methods is given in this section.
CHAPTER 2. RELATED WORK

NURBS and Subdivision Surfaces

Most modelling tools offer various methods for the creation of 3D characters. A modeller may use basic shapes or NURBS \cite{26, 25} and modify them by extrusion, merging, symmetry and other tools in order to define the surface of the model. Another option is to begin with a basic polygon or a NURBS surface and apply the subdivision surfaces technique \cite{7}, plus other modifiers to achieve the desired result. Figure 2.1 shows half a face modelled using these tools, and finally applying symmetry to get the other half. These techniques among many others offer a set of high-profile tools for modellers and are found in most professional 3D software, either commercial such as 3ds Max \cite{42}, Maya \cite{43} and LightWave \cite{47}, or OpenSource software such as Blender \cite{44}.

Intuitive sketching tools

Recently some user-friendly systems for the creation of 3D characters have been developed to fill the gap of modelling tools for non-professionals. In 1999 Igarashi et al. presented Teddy \cite{18}, a sketch-based modelling tool, which allows the user to create a character using an intuitive sketching interface. The modeller draws several 2D freeform strokes, and the system interactively creates a 3D
model, allowing different operations on the surface such as extrusion and bending. Other methods such as ShapeShop [29] in 2005, SmoothSketch [19] in 2006 and FiberMesh [22] in 2007 improved its results and user interface. These approaches are relatively new, and they still need to improve their robustness and results, but some really interesting models can be created with this early software, as for example the fish in Figure 2.2, in which the strokes used for its modelling are shown.

Body Scanning

A completely different approach to mesh creation is body scanning. Human whole body scanners have been produced by Cyberware, TC2, Virtronics, Wicks and Wilson, Hamamatsu and others, often for anthropometric purposes or for applications in the fashion and clothing industries [9]. Such systems may be used to produce scans of as many people as needed. The data they provide is a 3D point cloud representing the body skin or with a fit underclothes, and surface reconstruction techniques are used to create a mesh. However, the obtained mesh usually is not a clean mesh as it may have holes or isolated pieces, and often it needs to be treated by some algorithm like Poisson Filter [20] in order to obtain a noise-free model.

2.3 Skeleton creation

In order to animate a model it is necessary to create a skeleton, which represents the internal structure of the character and will be used to define the movement of its polygons. It is formed by a series of rigid objects (bones) connected together by joints. Each of these joints may be able to rotate in 1, 2, or 3 orthogonal directions, which define the degrees of freedom (DOF) of the joint; translation can also be applied. Usually the rotation and translation of joints is defined using a local coordinate system, so each joint knows its position and translation with respect to its parent, and the root joint has global position and rotation information, which will affect all the skeletal hierarchy. In anthropomorphic characters, the skeleton often is a simplified representation of a real human skeleton, with fewer joints, that allows the avatar to adopt human-like poses and to perform realistic movements.

Professional 3D modelling software provide tools for animators to manually define a skeleton inside a character, as well as to translate and rotate its joints once created. These rotation and translation tools are very useful, and thus the work made in this thesis relies on them to allow user interaction, as described later in section 3.4.2. Nevertheless, these software do not help the user to overcome the tedious task of creating and placing the skeleton inside the model, which has
to be done by hand and for every single model. Section 3.2 describes the solution to this problem that has been implemented.

2.4 Rigging

Once the skeleton is placed the next task is to perform the rigging. Rigging is the process of attaching each polygon of the character mesh to the corresponding skeleton bones, defining which parts of the model move when a bone is translated or rotated. The attachment is controlled by a weighting system, in which every polygon is attached to one or more bones with a given weight, which determines the influence of each bone over the current polygon. Each polygon has to know by which bones is influenced and how much. As it happens with the previous steps, most existing techniques focus on professional users. Commercial software offer some methods for the definition of the rig by creating influence areas of each bone and attaching the polygons which are included in this area. This method, however, must be performed manually and requires a high level of knowledge of the software, as well as several hours of tedious work. Figure 2.4 depicts the manner in which 3ds Max works with the skeleton weights, showing in different colours the weights affecting the skin of a character automatically rigged using the method presented in this thesis.

Recently some researchers have worked on techniques to simplify skeleton placement and rigging, mostly aimed at providing fast results to non-expert modellers.

In 2003 Oliveira and others introduced a technique for automatically creating and animating models obtained from human whole body scanned data [23], de-
developing a layered model in which the underlying skeleton, simplified surface and mapping of the surface to the skeletal structure were generated without manual intervention. The algorithm uses a set of surface landmarks, usually provided by the scanning software, in order for it to generate the skeleton. This skeleton, however, is always created from a fixed template, and does not allow using a custom skeleton that may fit better the user needs. The process of mapping bones to the skin is made by dividing the surface mesh layer and applying a surface growing technique, using a distance function to generate the weights to the surface. First, bisector planes are positioned at most bone joints, perpendicular to the plane formed by bone pairs, as shown in Figure 2.5. Afterwards, in order to map every mesh polygon to its corresponding bone, the algorithm selects the polygons at the end of the extremities and expands through all the adjacent vertices until a bisector plane is found, and these polygons are marked as belonging to the same bone segment. A similar procedure is applied for the rest of bone segments, until every polygon is attached to a bone. In order to avoid artifacts when moving the skeleton, triangles near a joint are weighted so that they are influenced by both neighbouring bones. Figure 2.5 depicts the result of this surface grown regions.

In 2007 Baran and Popović developed Pinocchio [2], a method for automatic animation of 3D characters. Given a 3D triangle mesh and a generic skeleton Pinocchio computes the fitted skeleton and calculates its attachment to the character skin, thus allowing its animation using any skeleton-based animation method. The following section reviews some of these animation methods. More precisely, Pinocchio computes the skeleton joints position as an optimization problem, which objective function is fitting the skeleton correctly into the character while preserving as much as possible the shape of the original skeleton. This complicated problem is discretised in order to be able to solving it, since trying to solve it using continuous optimisation would be prohibitive. More details on this technique are given in chapter 3.
Other research in rigging methods has also focused on more specific approaches. For example in 2006, Orvalho, Zacur and Susin developed a technique for transferring a facial rig to different face models [24], using a sophisticated skeleton and muscle structure based on a face labelling method. This is only an example of a large variety of research works based on the rigging concept.

2.5 Animation

In this section various animation techniques are described. First, traditional methods are described, such as keyframe and inverse kinematics. Afterwards, a popular approach often termed motion capture is reviewed. Finally, a recent technique on editing motion capture using intuitive user interaction is briefly described.

2.5.1 Key-frame animation

Animation of 3D characters may be done in several ways. Professional software like MotionBuilder [48] include a technique called keyframe animation [35, 17], that interpolates bones position and rotation between user-defined key states. This interpolation method provides the animator with complete control over the character movement, but the definition of the key poses must be done manually, meaning that all the bones affected by the pose change must be translated and rotated carefully, to avoid unnatural poses or unwanted deformations of the character skin. This problem is caused by the fact that there are no constraints on
the bones. For example, if an animator wants to move an arm of the character, the shoulder will have to be rotated, and the elbow and the wrist probably will have to be adjusted too, taking care not to achieve an anatomically wrong position. The subsequent interpolation between the two key positions will create the intermediate states, generating an animation.

2.5.2 Inverse Kinematics

These deficiencies in the basic key-frame animation method led to the incorporation of inverse kinematics techniques [51, 16], adopted from the existing research and application in the field of robotics. This method adds a series of constraints to the character skeleton to achieve realistic poses with less effort than using basic rotation and translation of joints. To summarise, inverse kinematics work with a hierarchical kinematic chain of joints, some of which may be defined as end-effectors, usually the ones at the end of a chain. These end-effectors are the joints over which the animator has control, translating them directly to any location, and the inverse kinematics will solve the necessary rotations for the rest of the joints. No joint rotations must be given at all, the method takes care of the hard work of positioning the whole skeleton to match the translation constraint given by the user. As presented by Chadwick and others in their Critter system [8], this technique combined with the keyframe interpolation contributes to relieving the animator from some of the more tedious aspects of creating new animations by hand.

Because of these advantages nowadays most of the professional animation software includes inverse kinematics and key-framing techniques as the default animation mode, since it reduces in great measure the time and effort needed to create a realistic animation.

2.5.3 Motion capture

Currently most professional animators prefer to create their animations manually by keyframing, as said in the beginning of this chapter. However, as motion capture methods improve, their use in professional animations is increasing, and every day we can see more motion-captured characters in video games and movies. Recent examples are the hyper-realistic motion capture movie Beowulf, and the video game Guitar Hero III, which features 3D characters of famous guitar players that have been animated using motion capture (Fig. 2.6).

Researchers have been working on motion-capture for various years now [21, 4], and also several works based in motion capture have appeared recently presenting very diverse work, such as keyframe animation assisted by motion capture [27], virtual reality motion capture based games [15], and physics-based dynamic interaction in motion capture animations [52]. In this thesis motion capture has
been used to obtain the animations that are used on the character. In Chapter 5 the motion capture process is described in detail.

### 2.5.4 Alternative animation methods

Recently some alternative methods for defining 3D character animation have appeared, with the objective to create user-friendly tools for novice users, following the philosophy of the alternative modelling techniques described in section 2.2 such as *Teddy*. One of these methods is *Motion Doodles* [40], presented by Thorne and others in 2004. In this research work the authors present a sketching system that allows the user to create 2D and 3D realistic animations by drawing a sequence of lines, arcs, and loops directly in the scene. Six types of possible tokens (simple strokes) are defined, which combined form a set of 18 possible types of 2D motions. These tokens are used for the segmentation of the continuous drawing made by the user, and the motions they form are extracted and put together to generate a continuous animation. Currently the set of possible motion types in 3D is quite limited, due to ambiguity problems when dealing with 3D strokes, however, the originality of this approach is noticeable, and it may lead to further research and possible full-featured sketching tools.

### 2.6 Clothing

In the process of modelling a 3D character, it is very common to first model the character without clothes, and afterwards create the clothes for it. These clothes have to be adjusted to the character and move along with it when animation is applied. The research done in clothes animation involves collision detection and realistic movement among other subjects.

Initial research in physically based cloth modelling was made in the late eighties by Terzopoulos et al. [37, 39, 38], introducing the finite element method in

![Figure 2.6: Slash in a motion capture session for Guitar Hero III](image-url)
this field. Ten years later, Baraff and Witkin solved some of the initial problems of this technique [1].

In 2001 Vassilev, Spanlang and Chrysanthou developed a technique for fast cloth animation on walking avatars [41], which using a mass-spring model [5] achieved physically plausible simulation of animated clothes at a rate of three to four frames per second. The method presented a velocity directional modification method to deal with the elasticity problem of the original mass-spring model. They also presented an image-based collision detection technique that allowed a fast cloth-body collision detection.

More recent approaches make use of new resources such as the programmable graphics hardware. The work by Rodriguez-Navarro and Susin [28] in 2006 presented a finite elements method (FEM) technique for cloth simulation running on the GPU. The use of the GPU allowed to run cloth-mesh collision detection as well as self-collision, using an image-based approach.

In Chapter 4 the clothing method that has been used in this thesis [34] is described in detail.

## 2.7 File formats

Many file formats allow saving the information needed for an animated 3D character, which involves saving at least the meshes, textures, skeleton, rig and animation. Most of the applications have each own format, defined in a different manner, and most of them without a public specification. Recently, however, there has been an effort by some companies to create compatible and well-specified file formats, even offering software development kits (SDK) for allowing users to read and write their own models in these formats. These file formats and their application programming interfaces (API) have been tested and analysed in terms of ease of use, understandability and compatibility with current modelling and animation software.
2.7.1 Cal3D

Cal3D [45] is an opensource skeletal-based animation library providing its own file format, originally developed for its use in video games and currently being maintained by the opensource community. Despite initial efforts like importers/exporters for the major modelling software, its development is in standby since 2006 and the community is not very active. Furthermore, its integration with current software is very poor.

2.7.2 Collada

Collada [46] is an interchange file format, defined in XML and implementing all the features needed in a 3D application. It is widely supported, so most of the current tools have importers and exporters of Collada files. Its main use is in video games, but also is widely used in other fields. It has a very active community and a solid user base, formed by modellers and animators as well as developers that use its open source SDK. Collada also contains some other packages other than the file format, such as physics support or definition of shader effects for the visualisation.

2.7.3 FBX

The FBX [50] file format was originally developed by Kaydara, a 3D company that was bought by Autodesk. Since the acquisition, the FBX became widely supported by professional software, and has a free to use proprietary SDK. As for its features, it is very similar to Collada. And like Collada it has a large number of users, but its major weakness is the fact that it is not opensource and so its lack of an active community supporting it.

The file format that would have been more interesting to use in this thesis is Collada, for its large user base and its promising future. However, since the software used for character modelling and animation was from Autodesk (3ds Max and MotionBuilder), the Collada files had to be loaded and saved using a plug-in. This plug-in for export and import was tested and presented some trouble, thus finally the decision was to use FBX, which is native on Autodesk products and have a highly documented SDK as well, although not such a big user community. A more detailed description of the incorporation of this file format in the thesis can be found in section 3.4.1.

2.8 Summary

This chapter has shown a review of related work on the whole process of modelling, rigging, and animation of a 3D character, explaining briefly some of the
related work. Other related work such as cloth animation has also been briefly reviewed, as well as the different file formats for 3D content storage.

In the next chapter a detailed description of the rigging process is given, as well as implementation details for the integration made with the FBX file format.
Chapter 3

Rigging

This chapter describes the method used for automatic skeleton embedding and character rigging, presented in 2007 by Baran and Popović [2, 3]. This method was chosen among other automatic rigging systems because of its outstanding results and its ease of use. This implementation, called Pinocchio, was released as a C++ library including source files, and it is used in this thesis as a basis for further work in the subject. The last sections of the chapter explain the method used to integrate the technique with current software, and how user interaction has been included in the process.

Figure 3.1: Pinocchio skeleton embedding process

3.1 Overview

The rigging process presented in this chapter fits a given skeleton template into a given 3D character mesh. The resulting output includes the mesh and the fitted
and rigged skeleton, saved in a well known file format in order to allow further editing and animation in popular 3D software. Figure 3.1 represents this process graphically. For simplification of the embedding, there are some requirements for the input data. The character must be a single connected mesh. Also, the skeleton and mesh must be in a similar pose and orientation, and should have similar proportions. Size differences between the skeleton and the mesh do not affect the result, as a rescaling is always applied to fit inside a unit cube.

The algorithm first finds an optimal skeleton embeddings fitted into the input mesh. This new skeleton maintains the same structure and roughly the same pose as the input skeleton. Afterwards, it computes the attachment of the new skeleton to the skin, and saves the result.

3.2 Skeleton Embedding

Given a character mesh and a template skeleton Pinocchio adjusts the skeleton to fit inside the model, by resizing and positioning its bones and joints. This adjustment is made by treating the issue as an optimization problem, the objective of which is to compute the skeleton adjustment that fits better inside the character while maintaining as much as possible its resemblance with the original skeleton. Solving this using continuous optimization is infeasible, since for a skeleton with \( n \) joints this would mean solving a \( 3n \)-dimensional problem with a complex objective function.

What the authors propose is a discretisation of the problem, constructing a graph with potential joint positions represented by vertices and potential bone segments represented by edges. The construction of the graph is accomplished by creating spheres centred on the character’s approximate medial surface, and then connecting the sphere centres forming a graph (Fig. 3.2). Afterwards, this graph is used to find the optimal embedding of the skeleton with respect to a discrete penalty function. The resulting skeleton is treated then by a continuous optimisation to achieve a better embedding.

3.2.1 Preparation process

Discretisation

In order to find the medial surface, the algorithm constructs a trilinearly interpolated adaptively sampled signed distance field on an octree [12]. At first it computes a kd-tree to evaluate the distance from all the possible points to the surface, and then the distance field is constructed from the top down. Precisely, the algorithm starts from a single octree cell, and splits only the cells that intersect the mesh volume until the exact distance is within a tolerance of the interpolated distance. The authors propose a tolerance \( \tau \) of 0.003, because it
Figure 3.2: Approximate Medial Surface, Packed Spheres, and Computed Graph.

presents the best relation between speed and precision. The distance field is then used to compute an approximate medial surface (Fig. 3.2, left image). To accomplish this, the octree is traversed, inspecting the nearby points for each face of each cell and computing the gradient vectors for their adjacent cells. These points are added to the medial surface only if the vectors form an angle of more than $120^\circ$, in order to only adding those points in which the skeleton is more probably to be placed. In addition, points within a distance to the surface minor than $2\tau$ are also discarded for the same reason.

Once the medial surface is created, its points are sorted by their distance to the surface and used to calculate the packed spheres, starting with the points which are farther from the surface. If the current point is outside all the previously added spheres a new sphere centred at the point is added, using as a radius the distance to the surface. This way, the resulting spheres will be the largest ones, and no sphere will contain other sphere centres (Fig. 3.2, centre image).

Finally, the graph of potential joint and bone positions is constructed by connecting sphere centres. An edge is added between two spheres if the spheres intersect. An additional condition has been used to add spheres that do not intersect but are important to the structure. This condition is that the distance from any point of the edge to the surface must be at least half of the radius of the smaller sphere, and the additional edge must be in the Gabriel graph [13] of the sphere centres. This way, spheres that do not intersect but are necessary will be added, as seen in Figure 3.2 (centre and right) with the neck and left shoulder spheres, which do not intersect but an edge between them is created.

Before proceeding to the embedding step, Pinocchio performs a precomputation of the shortest paths between all pairs of vertices in the graph of potential joint positions.

**Simplifications**

The graph of potential joint positions will be used in the next step of the process to fit the skeleton into the mesh. However, since the given skeleton may have an
undetermined number of joints, the optimisation problem would be intractable if too many positions had to be computed. In order to avoid this potential problem, the algorithm first reduces the skeleton. The simplified version is constructed by eliminating all degree two joints, thus dramatically reducing the complexity of computations that will be performed. Figure 3.3 shows the differences between the two skeletons. These simplification does not affect the final result, since after the embedding the skeleton is reconstructed by splitting the bones in accordance with the proportions of the initial skeleton. This is done by taking the shortest path between the joints in the constructed graph, and adding the needed joints in concordance.

**Discrete Penalty Function**

The requirements for a good skeleton embedding are quite complex, as it should have the proportions, bone orientations and size similar to the given skeleton. Furthermore, the paths representing the bone chains should be disjoint. To overcome the difficulty of designing such a complex penalty function, Pinocchio uses independent penalties and follow a learning procedure to find a good global penalty function combining them.

The following penalty functions have been designed:

- Penalisation of short bones.
- Penalisation of embeddings which directions between embedded joints differ from those in the given skeleton.
- Penalisation of bone chains sharing vertices.
- Penalisation of bone chains of zero length.
- Penalisation of improperly oriented bone segments.
• Penalisation of degree-one joints that are not embedded farthest from their parent.

• Penalisation of joints that are close to each other but are not close in the graph hierarchy.

The global penalty function $f$ is a linear combination of these $k$ simple penalty functions: $f(V) = \sum_{i=1}^{k} \gamma_i b_i(V)$, where $\gamma_i$ represents the weight that the penalty function $b_i(V)$ has been given by the learning procedure, being $V$ the tuple of vertices of the skeleton.

The learning procedure to assign the weights to each penalty function is based on the theory of support vector machines [6]. A full description of the learning procedure used in the finding of the global penalty function can be found in [2].

3.2.2 Embedding process

Discrete Embedding

It is computationally difficult to compute a discrete embedding that minimises the general penalty function. To solve this problem the authors propose the use of an estimation of a lower bound on the function from a partial embedding, and use a branch-and-bound method to extend the embedding. A priority queue is used to keep the partial embeddings ordered by their lower bound estimates. The best partial embedding is taken from the queue at every step, extending it to the next joint, and pushing the result or results back to the queue. In this manner, the first complete embedding taken from the queue is the optimal one. This is basically the A* algorithm on the tree of possible embeddings.

Embedding Optimisation

![Computed skeleton and the result of embedding refinement](image)

Figure 3.4: Computed skeleton and the result of embedding refinement

The optimal embedding created by the discrete optimisation may have some problems, such as not fitting nicely inside the character and having the smaller
bones incorrectly oriented. The latter happens because these small bones do not influence enough when the optimisation is run. A refinement step with a simple penalty function solves these problems in most of the cases. Figure 3.4 shows a result of the refinement, which is found by using a gradient descent method.

### 3.3 Skin Attachment

Once the skeleton is placed inside the model, it needs to be attached to the skin so when a joint is rotated or translated the mesh acts accordingly. To this end, it is necessary to compute the weights of the bones for each vertex.

Some conditions have to be met in order for the weights to be correct:

- The weights should be independent of the mesh resolution.
- The distribution of these weights should be smooth.
- The width of a transition between two bones meeting at a joint should be roughly proportional to the distance from the joint to the surface.

For these properties to be satisfied, a temperature equilibrium analogy has been used. The character volume is treated as a heat-conducting body. For each bone, its temperature is marked to be 1 while keeping the temperature of the other bones at 0, and then the weight of this bone at each surface vertex is taken from the equilibrium temperature. Figure 3.5 shows this equilibrium in a simplified 2D bone; Figure 3.6 shows an example with a 3D character, where a deformation artifact can be seen in the elbow, probably caused by the low resolution of the mesh.

![Figure 3.5: Temperature distribution on bone before and after rotating the joint.](image)

The distribution of the temperature over the bones is found by solving the heat equilibrium on the surface. In addition, for some vertices the heat from the nearest bone is also transferred.

Once the vertex weights have been found, the skin is attached to the skeleton, and when a bone moves the mesh will follow its movements based on the weight value of this bone on each vertex of the skin mesh.
3.4 Extending Pinocchio

One of the main objectives of this thesis was to have an automatic rigging system for its future use in other applications, such as having a fast way of creating animated characters for research in virtual reality or other fields. In most cases, character rigging had to be done manually, wasting many hours of real research because of the complexity of current rigging tools. Pinocchio addresses the rigging process in a faster way than any modeller, and the results are very good for most of the applications. The method, however, does not deal with real-world possibilities, being its main problem that it describes the structure of the skeleton hard-coded in the source code of the library, leaving the potential end-user with no possibilities to use a different character pose or a different skeletal structure. As recompiling the code every time that a different skeleton is not feasible, the solution offered in this thesis to allow this user interaction is to use an already available file format to perform all the input and output. This way, the user may continue using the regular modelling and animation tools and, when needed, perform this rigging process with minimal effort.

3.4.1 FBX integration

As explained in section 2.7, Autodesk FBX [50] is the chosen file format for implementing input and output in the rigging and animation process. Its SDK, which is publicly available, has been used in this thesis to develop an extension to the Pinocchio library, in form of a stand-alone command line program written in C++.

As most of the 3D file formats, FBX is structured in a hierarchical form. The root node, called the Scene, is unique and encapsulates all the information for the visualisation of a three dimensional animated scene, including the characters, 3D objects, cameras, lights, background, and any other object that may appear in a 3D environment, as well as internal aspects such as object modifiers or animation parameters. Among all these available elements only a few are relevant to our implementation, as we only need to deal with character meshes, skeletons, and
their attachment.

### 3.4.2 User interaction

The program that has been developed is thought as a helping software for researchers that work with 3D characters, obtained either by body scanning or by modelling. These avatars need to be animated, and possibly this animation will come from motion capture.

This situation led to the design a system that may receive different kinds of models as well as different types of skeletons. In addition, it must allow some kind of user guidance of the process, in order to fix possible deficiencies when dealing with a specific model or skeleton. The developed software allows the input of a skeleton in FBX format plus a character mesh, and its process is the following.

- The skeleton is parsed and converted into the internal format of the Pinocchio library. The mesh is also passed to the library.
- The skeleton embedding process is executed. After this step the skeleton is fitted into the character mesh.
- At this point the process is paused. An intermediate output containing the mesh and the fitted skeleton is handed to the user in FBX format. The program waits for the user to confirm this intermediate state, or to modify the fitting if there is some problem.
- The FBX file is re-loaded in order to incorporate any modifications made by the user.
- The rigging step is executed. The result containing the rigged mesh is saved in a different FBX file, for if the user wants to change something and rerun only the rigging step.

As seen, this application can be used as a rigging method for relieving the hard work, allowing to spend quality time working with the usual tools. This is a simplified version of the application behaviour, as it not includes the clothes processing. In the next chapter the modifications made in order to include the clothes are described.

### 3.5 Summary

In this chapter a description of the Pinocchio rigging library has been presented. The method creates a rigging for the wanted mesh, but does not allow much user interaction. This interaction is achieved by the software described in the last section, which integrates the powerful rigging method with the common tools a user may need, plus allowing some interaction with the process.
However, the resulting character is naked, as the process detailed in this chapter only works with simple connected meshes. Hence, a method for adding virtual clothes to the character may be needed. In the next chapter such a method is reviewed.
Chapter 4

Clothing

In the previous chapter the process of rigging of a 3D character has been detailed. If the character needs clothes they will need to follow its movements. This chapter deals with the process of creating adjusted clothes for an avatar and allowing them to move.

4.1 Overview

Section 4.2 describes the method used for the creation of the clothes for a 3D character. Afterwards, in section 4.3 the method for attaching the clothes to the character is detailed. At the end of this process, the clothes will be fit to the character and will move along with the avatar, which will be saved in a popular file format, as described in section 4.4.

Figure 4.1: Clothing process. First, a pair of jeans pre positioned around the body geometry. Second and third, a jacket pre positioned and during the sewing simulation. Last, back view of the completed simulation.
4.2 Clothes creation

The adjusted clothes for the characters have been created using the technique by Vassilev, Spanlang and Chrysanthou [41]. In this paper the authors present a modified mass-spring model for the adjustment of clothes patterns. A format for specification of the clothes is presented, being able to define different types of cloth in 2D. This 2D specification is used to create a mesh (Fig. 4.2). The cloth is then incrementally adjusted to the skin, using the mass-spring model, and accelerating the process with the use of graphics hardware for cloth-body collision processing. In Figure 4.1 an overview of the steps of the process is shown.

![Figure 4.2: Mesh of a pair of jeans ready for simulation.](Image)

More results of the cloth creation are shown in section 6.

4.3 Attachment to the character

Once the clothes are computed to nicely fit the model, they need to be attached to the character and must be rigged using the same skeleton in order to follow its movements.

In most cases the clothes consist of a mesh of 3D triangles, defining the basic shape. Some details are achieved by additional meshes, which are used for pockets or other supplements. These meshes are grouped together forming a single piece of cloth, and should be treated as a single object to avoid possible inconsistencies in the rig that has to be computed. This situation discards the possibility to use the same rigging method used for the character, because it can only deal with closed meshes forming a volume. The clothes have to be attached to the skeleton using a different approach.

The method that has been developed is based on a very simple concept. It uses the information that has already been computed to calculate the attachment of the clothes. After the previous steps of the process we know which polygons are affected by which bones, and also the value of the weights for every one of
them. Furthermore, we can assume that the skeleton is correct for the mesh, and since the clothes are correct for the mesh, it should be possible to use the same weight information to calculate the attachment of the clothes. In order to know which weights have to be used, the algorithm finds for each vertex of the cloth the closest vertex, in terms of euclidean distance, belonging to the character which is oriented similarly. To decide if the condition is met, the normal vectors of both vertices are found, and a threshold angle is used to compare the normals. By testing we found that the best results are provided by setting a threshold angle of $45^\circ$, but since it may depend on the characteristics of the mesh, the user has the option to use a different threshold. Figure 4.3 shows a simple example: the white vertex has to be rigged, but its nearest vertex (red) does not have a similar normal, so the rigging will be performed using the other one (green), although it is further away.

![image](image.png)

Figure 4.3: Example of a vertex discarded by the cloth rigging algorithm.

In Figure 4.4 the result of rigging the same character without clothes and with them is shown. The right elbow and the left knee have been rotated to demonstrate the good behaviour of the clothes rigging. Further images of rigged clothes, plus comparisons of using different thresholds, and performance analysis are shown in section 6.

### 4.4 FBX integration and User Interaction

The rigged clothes should be saved in the same file that contains the rigged character, in order for the user to be able to apply the animations to the dressed character without any post processing of the results. To this end, clothes are processed in the same program execution as the rigging of the character; then all
the results are saved together in the same FBX file. Clothes are included as new nodes in the hierarchy of the 3D scene, specifically as direct descendents of the character mesh. This way, any global transformation applied to the character mesh will also be applied to the clothes, and the clothes will remain well fitted. The number of clothes that may have been processed is not a problem, since the FBX file format allows the definition of an unlimited number of objects in the scene, and also has no restrictions on its hierarchy. In section 3.4.1 more details of the FBX file format have been given.

If the results are not the desired, the user is allowed to define a different normal angle threshold for the cloth rigging. Also it is possible to run only the cloth rigging, this way the user is able to try the results of different thresholds without having to run the whole rigging process. To do so, the application loads the previously computed FBX, which included the rigged character, and performs the attachment of the clothes as explained in the previous section.

4.5 Summary

In this chapter the cloth creation system used in this project has been briefly reviewed. Afterwards, the approach used for attaching the clothes to the previously rigged character is described. The integration with the whole rigging and animation system is reviewed as well.

In the following section a description is given on the animation method that has been used in this master’s thesis. Furthermore, the integration with the information accomplished in the previous chapters will be made.
Chapter 5

Animation

5.1 Overview

This chapter presents the animation method that has been used to integrate with the results of the rigging and clothing of 3D characters. The chosen technique is motion-capture –often abbreviated mocap– because of its realistic results and its easy integration with current animation software.

5.2 OptiTrack

OptiTrack [32] is a complete and affordable motion-capture system. It is commercialised by NaturalPoint, a company specialised in tracking systems, which also offers other solutions such as a head-tracking display for video game players and an infrared eye-tracking system for PCs. Many other manufacturers, like Vicon, also offer motion-capture systems, but to our knowledge OptiTrack is the system with a better quality-price ratio.

5.2.1 System Setup

The OptiTrack system is formed by a varying number of USB cameras connected to a personal computer and a set of reflecting markers which will be worn by the subject. The recommended configuration is of no less than 6 cameras, and offering optimal results with 8 cameras or more, but it may depend in a great measure on the conditions of the setup scenario, as well as possible reflections or blocking objects, which may have a negative effect on the precision of the animations. The cameras are able to capture infrared light as well as black and white images. The main difference from other systems is that the cameras have a small CPU which allows them to perform live image processing such as applying
a threshold on the image, or blocking certain areas where reflections are present. The goal is to have a final image where the points to be tracked (markers) are easily recognisable, and so the only things that have to be transferred to the PC are the 2D positions of the markers. This feature allows several cameras to be connected to a single PC, as very low bandwidth is used. In addition the PC does not need to process any image, only direct points, so the processor is more capable of dealing with the animation. The system includes a specific software to gather the information given by the cameras and compute the movements, and this application is also used for calibrating the cameras before starting the actual capture.

First of all, the cameras have to be distributed throughout the room, taking care in avoiding as much as possible blocking objects and cameras seeing other cameras, and trying to avoid points seen by only one or two cameras. In this thesis a setup of 8 cameras has been used.

5.2.2 Calibration

Once the cameras are positioned and connected to the PC, they may need some adjustments. The software provides a tool for reviewing what is seen by each camera, allowing to adjust some parameters, such as ignoring some invalid points caused by reflections, or tweak the image processing made by the cameras. Afterwards, the OptiTrack software asks to perform a calibration process, in order for the PC to learn interpreting the data which it receives from all the cameras. In Figure 5.1 an overview of the calibration process is shown. First of all, the user has to move a single marker around the room, trying to reach all the possible points seen by each camera. This process is helped by an interface showing the path drawn so far (Fig. 5.1 top). The PC gathers all the data received and perform a series of calculations that will indicate the relative position of all the cameras. Once these positions are known, a reference system must be created, in order for the software to know where the floor is and how is it oriented with respect to the cameras. This is done by placing three markers on the floor forming a triangle (Fig. 5.1 centre). The last step is the reviewing of the calibration process, in which the software shows a picture of all the camera frustums and the effective motion capture volume (Fig. 5.1 bottom). In this step the user is allowed to move the markers through the scene to confirm that the positions are well computed.

5.2.3 Motion Capture

Once the system is calibrated, the user dresses in the mocap suit and places all the markers on it, guided by the software. Then a skeleton calibration needs to be done in order for the system to know the physical characteristics of the
Figure 5.1: Three steps of the OptiTrack calibration process
user. To this end, the subject is asked to stand in T-pose and the software computes a skeleton from the positions of the different markers. Some skeleton parameters may be adjusted to fit better the user, such as total height, shoulder width, and bone lengths. An example of a skeleton used by OptiTrack is shown in Figure 5.2. Once the user confirms the skeleton, the system is ready to start the motion capture. The calibration and skeleton creation steps may be saved for their use in following motion capture sessions.

The actual motion capture process is simple. The user chooses the recording time, and during the recording session the software saves all the data received by the cameras, while simultaneously showing a default character moving on screen. Once the motion capture is finished, the animation can be exported to the popular Biovision Hierarchy (BVH) format [32], which is supported by most animation software.

![Default skeleton for the OptiTrack system](image)

Figure 5.2: Default skeleton for the OptiTrack system

### 5.3 Animating the rigged character

The character that has been created in the previous chapters is now rigged, dressed up and ready for the animation process. As told, the animation will be obtained using motion capture data. The rigging and clothing process has been executed using the skeleton from OptiTrack, this way the animation data obtained in this chapter can be used directly on the rigged character.
The motion capture system that has been used allows different methods of using the animation data in any system:

- Data may be exported to BVH file format, and then imported into any 3D animation software.

- An OptiTrack plug-in for MotionBuilder is available, allowing real-time motion capture with a custom character.

- A Software Development Kit (SDK) offering animation streaming data is also available. Using this library, any software is able to obtain the animation data in real-time during motion capture. Applications such as immersive virtual environments may use this SDK for implementing real-time feedback to the user.

The animation software that has been used to animate the character is MotionBuilder [48]. This application makes use of a predefined complex skeletal structure to which all the movements are applied, and offers a mapping tool in order to transfer the movements to any custom skeleton. The skeleton used by MotionBuilder is very complete in order to allow the maximum number of details when animating the character, but also allows the use of a subset of the available joints. This is what has been done in this thesis: the rigged character processed using the OptiTrack skeleton has been mapped to the MotionBuilder control skeleton, and the animation from motion capture has been applied straightforward. Results and analysis of this animation step are shown in chapter 6.

5.4 Summary

In this chapter a review of a popular motion capture system was presented, and details on its usage were given. The integration of animations with the work made on previous chapters was described.

In the next chapter results and tests of the automatic animation system will be shown.
Chapter 6

Results

This chapter presents some results obtained using the method described throughout the whole thesis. Some pictures of the different steps of the process are shown. In addition, some tests have been executed and its results are presented.

This thesis aims to provide an easy-to-use tool for non-professional modellers. To this end, the process has been tested from a practical point of view, with the objective to discover its limits. Following this criteria, the following tests have been made on the character rigging step:

• Orientation limits. The input skeleton is rotated previous to the processing in different angles and axis.

• Pose variation limits. Extremities are rotated so the initial pose of the skeleton differs from the character one.

• Possibility to use different skeletons. From a basic "stick man" to a very complex skeleton have been tested.

Results from the clothing step are also reviewed.

Orientation limits

The input skeleton may not be always oriented exactly like the character mesh. In order to know which are the limits in which a good embedding is found, different orientations of the same skeleton have been tried. In tables 6.1 to 6.3 the results are shown.

Pose variation

The pose that is normally used for the skeletons is the so-called T-pose. However, it is possible that the mesh is not in this exact pose, but may have the arms in
a different position, which is often the case with 3D body scans. The following test show which are the limits of this pose difference for the algorithm to work properly. In those situations where the approach does not work the user is still allowed to modify the original skeleton and run again the process.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Rotation</th>
<th>Finished?</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>15°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>X</td>
<td>30°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>X</td>
<td>45°</td>
<td>No</td>
<td>–</td>
</tr>
<tr>
<td>X</td>
<td>90°</td>
<td>No</td>
<td>–</td>
</tr>
<tr>
<td>X</td>
<td>180°</td>
<td>No</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6.1: Test: Skeleton rotations on the X axis.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Rotation</th>
<th>Finished?</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>15°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>Y</td>
<td>30°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>Y</td>
<td>45°</td>
<td>No</td>
<td>–</td>
</tr>
<tr>
<td>Y</td>
<td>90°</td>
<td>Yes</td>
<td>see Fig.6.2</td>
</tr>
<tr>
<td>Y</td>
<td>180°</td>
<td>Yes</td>
<td>see Fig.6.3</td>
</tr>
</tbody>
</table>

Table 6.2: Test: Skeleton rotations on the Y axis.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Rotation</th>
<th>Finished?</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>15°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>Z</td>
<td>30°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>Z</td>
<td>45°</td>
<td>No</td>
<td>–</td>
</tr>
<tr>
<td>Z</td>
<td>90°</td>
<td>No</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6.3: Test: Skeleton rotations on the Z axis.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Rotation</th>
<th>Finished?</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>15°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>X</td>
<td>30°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>X</td>
<td>45°</td>
<td>Yes</td>
<td>Correct</td>
</tr>
<tr>
<td>X</td>
<td>90°</td>
<td>Yes</td>
<td>see Fig.6.4</td>
</tr>
</tbody>
</table>

Table 6.4: Test: Arm rotation.

**Skeleton variety**

In order to evaluate how well the algorithm would behave with more complex or more basic skeletons, the algorithm has been tested with a "stick man" skeleton, two normal skeletons and a very complex skeleton (containing more than 50 bones). The algorithm only finished when using the simple skeletons, not when using the complex one. The result of this test is somewhat expected, since the algorithm is not developed for its use in such a high level environments.
Clothing

The clothing system has been tested to analyse the different results obtained when applying a different threshold angle for the cloth rigging process.

Figure 6.1 shows the clothes of a character after applying the rig as described in section 4.3. Notice how the same joint (in this case the left hip) is affecting both the jacket and jeans of the character.

![Image](image1.png)

Figure 6.1: Skeleton weights applied on clothes.

Results from Pinocchio

Baran and Popović [2] present some results of the skeleton embedding process which are interesting. In Figure 6.5 the set of characters that they used for testing are shown, as well as the embedded skeletons for each one of them. Notice how Pinocchio produced a good result for nearly all of them. Only models 7, 10 and 13 have an embedding problem. In these cases a fast user modification should solve the problem.
Figure 6.2: Test: the process took more than usual, and the embedding is incorrect.

Figure 6.3: The skeleton is completely rotated, plus the feet joints are incorrect.
Figure 6.4: The joints of the arms are incorrect.

Figure 6.5: Skeleton embedding results from Pinocchio.
Chapter 7

Conclusions

In this master’s thesis a comprehensive method for the animation of 3D characters was developed. Starting from a static model, this technique is able to obtain a full featured 3D avatar which may be animated using any animation method, including motion capture. Situations where this work may be useful are numerous. A novice modeller may use it to animate his characters with minimal effort, being able to easily review and improve the work done. In real time immersive environments, user experience may be improved, by permitting fast creation of user avatars which directly integrate into the scene.

In chapter 1 an introduction to 3D characters and its animation process was given. The second chapter presented a review of the related work on the subject, as well as a description of the software that had been used. Chapter 3 described the method for the automatic rigging of the avatars, and modified it in order to allow direct user interaction. In chapter 3 a technique for dressing the characters was detailed. Chapter 5 reviewed a motion capture system and explained the integration in the whole process. In chapter 6 some results and tests could be found.

Future lines of work may include the improvement of the different steps of the process. Rigging may be improved by adding hands or facial rig [24]. The whole system may be developed in form of a plug-in for some of the 3D modelling software, in order to allow direct feedback to the user.
Bibliography


on High Performance Computing for Computer Graphics and Visualization, 1995. [cited at p. 3]


