Development of an automatic skinning system

Emeline Got

supervised by

Dr. Antonio SUSIN SANCHEZ
Department of Mathematics

Anthony COLTEL
Quantic Dream

October 2019
Abstract

The creation and animation of characters is an important step in the video game production pipeline. The animation of the characters is often driven by a computer-generated skeleton. To link the character’s bodies to their skeleton—a step called skinning—several automatic techniques exist but often present significant drawbacks that prevent artists from using them rather than doing the work by hand. Therefore, there was a need for developing an automatic skinning tool that would answer the artists’ needs.

The work in this thesis addresses this issue by offering the artists a skinning tool based on geodesic distances inside the characters’ voxelized meshes. A copy skin algorithm was derived from it and offers the artists the possibility to transfer their skinning work from one source character to a target one. With this tool, we manage to answer common problems encountered with the automatic skinning methods implemented in the software Autodesk Maya.
<table>
<thead>
<tr>
<th>A</th>
<th>Shaders</th>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Geodesic voxel copy skin results on open meshes</td>
<td>57</td>
</tr>
<tr>
<td>C</td>
<td>Geodesic voxel copy skin results on naked / clothed models</td>
<td>59</td>
</tr>
</tbody>
</table>
List of Figures

1 Quantic Dream latest games .................................................. 8
2 Video game production pipeline ........................................... 9
3 Concept art for the character of Kara in Detroit: Become Human .................................................. 10
4 Example of the creation of a low-poly model of a woman bust (b) from primitive shapes (a) [1] ....................... 11
5 Example of a hi-poly model of a woman bust [1] ................. 11
6 Rigging process ................................................................. 12
7 Example of a mesh animated by the movement of the skeleton bones ................................................................. 13
8 Extra-deformers ................................................................. 14
9 Skinning artefacts ............................................................... 16
10 Artefacts obtained with joint proximity binding method ....... 18
11 Skinning weights ............................................................... 19
12 Geodesic Voxel Binding process ........................................... 25
13 Geodesic Copy Skin process ................................................ 26
14 Voxelization process .......................................................... 27
15 Slices example for interior voxelization ............................... 29
16 Closest Voxel ..................................................................... 34
17 Closest Point ..................................................................... 34
18 Selection among different closest points ............................ 35
19 General design of the tool UI ............................................... 37
20 Geodesic voxel binding tab .................................................... 38
21 Example of getting a resolution indication for a unit sphere ........................................................................... 39
22 Geodesic voxel binding tab .................................................... 40
23 Examples of displayed errors in the tool UI ....................... 41
24 Success messages displayed in the help bar ........................ 41
25 Popup windows when wrongly selecting objects for copy skin ........................................................................... 42
26 Artefacts from the original Geodesic Voxel Algorithm ......... 42
27 Geodesic copy skin results with a naked body as source (left) and a clothed body as target (right) ....................... 43
28 Maya copy skin results .......................................................... 44
29 Results of the geodesic voxel copy skin on a simple model ............................................................................. 44
30 Importance of the voxelization resolution while copy skinning ......................................................................... 45
31 Vampire mouth .................................................................. 46
32 Copy skin results with the Geodesic Copy Skin method ....... 46
33 Copy skin results obtained with Maya tool .......................... 47
34 The Geodesic Voxel Copy Skin method deals with characters in different locations .................................................. 48
35 The Geodesic Voxel Copy Skin method deals with scaled characters .................................................................. 49
36 Results of the Maya copy skin with several joints at the same position ............................................................... 50
37 Results of the Geodesic voxel copy skin with several joints at the same position .................................................. 51
38 Results of the Geodesic voxel copy skin with several joints at the same position .................................................. 52
39 Results using the Maya copy skin at bind pose ..................... 57
40 Result using the geodesic voxel copy skin at bind pose ........ 57
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>Result using the geodesic voxel copy skin at a random pose</td>
<td>58</td>
</tr>
<tr>
<td>42</td>
<td>Artefacts in the armpit region</td>
<td>59</td>
</tr>
<tr>
<td>43</td>
<td>Artefacts while bending the arm</td>
<td>59</td>
</tr>
<tr>
<td>44</td>
<td>Artefacts in the crotch area</td>
<td>60</td>
</tr>
</tbody>
</table>
Introduction

Driven by my interest in discovering the applications of computer graphics in the entertainment industry, I applied for an internship at Quantic Dream. This French video game company is known for its AAA interactive storytelling realistic games such as Detroit: Become Human (Figure 1a) and Beyond: Two Souls (Figure 1b). In these games, the player has to choose regularly between different possibilities that will affect the actions of the characters and alter the course of the story accordingly. The choices call for the emotions of the player and their sensibility. The games are meant to be realistic so a non-negligible part of the development is spent on rendering characters, props and sets as lifelike as possible. This is expressed at different levels in the production of a video game, in the animation of the characters for instance. Quantic Dream uses its own Motion Capture studio and bases most of its animations on captured data to reinforce the realistic aspect.

![Detroit: Become Human](a) ![Beyond: Two Souls](b)

Figure 1: Quantic Dream latest games Detroit: Become Human (2018) (a) and Beyond: Two Souls (2013) (b)

Developing a game is a long process that can take several years to complete. The timeline that describes how a game is made is called the game production pipeline and is detailed in Figure 2. A game starts from an idea, a concept, that develops to a storyline. It then goes on to the content creation that covers concept arts, character modelling, level designs, AI implementation, etc. At Quantic Dream, this is also the stage at which the shootings are made so it includes motion capture recordings with several actors, treatment and exportation of the data too. Additional steps for the rigging and animation of the characters are performed to start creating the different scenes of the game in the game engine. Regular tests are made during the process to prevent bugs, errors or gameplay issues in the on-going work. At last, the post-production stage happens when all the art has been produced and the code written. The first builds of the games are beta-tested to reach the validated version that will be deployed to the desired platforms and released to the public.
In the video game production pipeline, the steps are not following a linear timeline. The different departments all work in parallel in order to move on progressively. This avoids having to wait for one department to be done working before moving on to the next task. Also, some feedback is done regularly to be able to correct eventual mistakes as the development progresses. It allows games to be developed faster but also means that some work will need several readjustments as other departments use the previously-made work.

In particular, in this thesis, we focus on the skinning process. It represents an essential step in the animation of a character that happens in-between the modelling and the actual animation. Skinning artists work on having character movements that are as plausible and as human-like as possible. However, an important part of their work is manual and unique on every model they work with. There is thus a significant need for automation of the process that would help speed up the production of characters.
1 State of the art

1.1 Character creation pipeline

The creation and animation of characters are key elements in the production pipeline. The process is divided in four main steps:

- character designing,
- modelling and texturing,
- rigging and skinning,
- animating.

The character creation starts with defining the design of the character. This is done by the art direction in collaboration with the game designers who determine how the characters should look like depending on what abilities they possess or what environment they evolve in (Figure 3). Whether they should carry weapons or evolve in a cold/hot country for example will define how they look and how they are dressed.

![Figure 3: Concept art for the character of Kara in Detroit: Become Human](image)

Once the character design has been defined, a first base mesh is created to fit a basic shape we want the character to have with low amount of details. Usually, the modelling artists shape the characters starting from basic geometric
forms called **primitives** such as cubes, cylinders or spheres and use boolean operations and transformations between them to get to the desired shape (Figure 4). Details are then added to the base mesh previously obtained in order to create a high resolution version (Figure 5).

![Example of the creation of a low-poly model of a woman bust (b) from primitive shapes (a)](image)

**Figure 4:** Example of the creation of a low-poly model of a woman bust (b) from primitive shapes (a) [1]

This step also includes the colouring of the model which corresponds to the application of a certain type of surfaces to the 3D model. These surfaces are called **textures** and can be of different kinds depending on the effects desired. For example, there are textures to add shadows, relief, reflection or more simply to apply colors.

The steps presented beforehand suffice to design inert objects of the environment (called **props**) that are not deformed. For characters that we want to animate however, further steps are needed.

To give characters the ability to move and stand in various positions, a **rig** is created. It consists of a certain number of controls that are to be used by the animators to move the character accordingly to their needs. The usual controls
used for rigging are represented by a hierarchy of points called **joints** and segments between them called **bones** in order to mimic human articulations and bones. The hierarchy created this way is called a **skeleton** and it will be placed inside the character mesh similarly to a real skeleton (Figure 6c).

![Figure 6: Rigging process: starting from a mesh designed by the modellers (a), the rigging artists create a hierarchy of bones and joints representing a simplified human skeleton (b) that is embedded inside the character skin (c).](image)

The creation of a pertinent skeleton for animating the characters is a fastidious task. There are several aspects to take into consideration while choosing the joints needed in the skeleton hierarchy. The rigging artist has to place a meaningful equivalent of the human skeleton inside the skin of the character. One might think that the best skeleton design would be the exact reproduction of the human one but we are only interested in having controllers that actually make the body move. Thus typically, bones that are static in the human body, such as the rib cage, would not have any influence on the model. There is also no need to double the bones in the arms and the legs as they are in the human body because both the bones would have the same movement effect on the character. Figure 6b shows an example of one skeleton design for an Android character of the game **Detroit: Become Human**.

Deleting anatomical bones can however be the source of a common problem in character animation which is the loss of volume. Not having a proper rib cage to fill the volume inside the chest cannot prevent the chest to shrink in some bending cases.

Then, a **skinning** (or also called **binding**) step is necessary. It corresponds to attaching the mesh of the character to the skeleton, allowing the skeleton to deform the mesh as the bones are manipulated. The mesh that is deformed by the skeleton is called the **skin**. It is done by attributing a percentage of influence for each vertex of the skin in relation to each bone of the skeleton. As an example, a vertex of the arm could be influenced up to 75% by the bone of the arm and 25% by the wrist. The percentages of influence affected to each vertex of the skin are called **skin weights** (or skinning weights).
With the skin weights attributed, the mesh can be deformed as wanted while manipulating the bones in order to animate the character (Figure 7).

![Example of a mesh animated by the movement of the skeleton bones](image)

**Figure 7:** *Example of a mesh animated by the movement of the skeleton bones*

As mentioned previously, a common artefact encountered while animating a character is volume loss due to the emptiness of the designed mesh. When an artist creates a mesh for a character, it is not filled with organs, muscles or all the bones that our human body has. Indeed, especially in a video game application’s perspective, we want to have models that are as simple as possible to be able to animate them in real-time. Modelling entirely a very human-like character with 3D muscles and organs that alter the deformation of the outside skin would be both very hard to implement and very unlikely to be usable in real-time.

For example, one solution to help preserve volume in character movements can be to consider additional deformers in the tedious areas in order to add more realism such as muscle bulges or clothes rolls (Figure 8).

In this thesis, we are interested in skinning techniques that help preserving the volume during the movements of the characters and that mimic as much as possible the displacement of a human body while preserving a limited computation time.
1.2 Skinning

1.2.1 Skinning methods

The skinning methods in the literature are plentiful. They are split in different categories and we will present some of them. Though what we will focus on are the skinning methods that are particularly efficient for a video game application and the limitations it implies. In particular, our proposal falls in the geometric skinning category and we will present those methods more thoroughly.

**Geometric skinning** deforms the position of the vertices of the model using algorithms that bind the meshes and controllers. The most widely-used skinning model for character animation is skeleton-based skinning. The rig consists of a basic hierarchy of bones that together create a skeleton. Some additional controls can be added on top of the skeleton to help the animators move the characters more intuitively than having to move the bones one by one.

The most popular method is **Linear Blend Skinning** (LBS) also known as vertex blending, Skeletal Subspace Deformation or enveloping. The Linear Blend Skinning method [2] is a low-cost skinning. Skinning transformations are represented by matrices blended linearly. The weighted linear combination of the rigid transformations matrices (rotation and translation) of the joints of influence for a vertex gives us its position as the joints are being moved.

Given a rest-pose mesh $M$ with vertex positions:

$$p_1, p_2, ..., p_n \in \mathbb{R}^3,$$  \hspace{1cm} (1)

a corresponding skeleton made of $m$ joints, represented as a hierarchy of rigid transformations:

$$T_1, T_2, ..., T_m \in \mathbb{R}^{3x4},$$  \hspace{1cm} (2)
and a weight function that specifies the amount of influence of the transformation $T_j$ on a vertex at position $p$:

$$\omega_j(p) = M \rightarrow \mathbb{R}, \quad (3)$$

the deformed vertex position is computed as:

$$p'_i = \sum_{j=1}^{m} \omega_j(p_i) T_j \left( \frac{p_i}{1} \right). \quad (4)$$

The blended matrix obtained by linear combination is no longer a rigid transformation but a general affine one (potentially containing scale and shear factors). This leads to undesirable effects such as skin collapsing, commonly known as the candy-wrapper effect (Figure 9a) when the twisting of a bone leads to skin shrinking.

**Dual Quaternions Skinning (DQS)** method [3] suggests another way of blending skinning weights that would preserve the rigidity of the transformations. Dual quaternions represent rotations with arbitrary axis unlike classical quaternions that can only represent rotations whose axes pass through the origin.

By converting the rigid transformation matrices to dual quaternions:

$$T_1, T_2, ..., T_m \in \mathbb{R}^{3 \times 4} \leftrightarrow \hat{q}_1, \hat{q}_2, ..., \hat{q}_m \in Q, \quad (5)$$

we can apply Dual quaternion Linear Blending:

$$DLB(\omega, \hat{q}_1, \hat{q}_2, ..., \hat{q}_m) = \frac{\omega_1 \hat{q}_1 + ... + \omega_m \hat{q}_m}{\|\omega_1 \hat{q}_1 + ... + \omega_m \hat{q}_m\|}, \quad (6)$$

where $\omega = (\omega_1, ..., \omega_m)$ are the skinning weights associated to the influences of one vertex. The computation of $DLB(\omega, \hat{q}_1, \hat{q}_2, ..., \hat{q}_m)$ gives us a unit dual quaternion that can be converted to a rigid transformation matrix $M$. We then use $M$ to compute the deformed vertex position with:

$$p'_i = Mp. \quad (7)$$

Keeping a rigid transformation allows to reduce the artefacts encountered with Linear Blend Skinning but new ones may appear such as joint bulging (Figure 9b) when the bending of joint creates an excessive bulge in the bending area. Despite the side effects exposed, LBS and DQS are still widely used—in particular in the video games industry—because of their efficiency and simplicity of implementation that are both well-suited characteristics for real-time applications.

Other methods try to blend transformations in the orthogonal space such as log-matrix skinning (LMS) [4] or spherical blend skinning (SBS) [5] but similar artefacts can be observed. Disney proposes a method that addresses the usual artefacts in [6] by computing the optimal center of rotation for each vertex of the mesh that can differ from the center of the joints. In order to get good
results, the method relies on a good set of skinning weights attributed to the
vertices of mesh.

![Image of skinning effects](image_url)

Figure 9: Example of two common skinning artefacts with LBS and Dual
Quatennion methods: (a) Candy wrapper effect, (b) Joint bulging.

In Autodesk Maya, the 3D computer graphics software used at Quantic
Dream for asset creation, the two skinning methods available are Linear Blend
Skinning and Dual Quatennions Skinning [7]. To reduce the presence of arte-
facts, Maya also offers the possibility to have a blend between the LBS and DQS
deformed vertex positions.

**Physical-based skinning** focuses on providing more natural effects to char-
acter animation — based on the observation of the human body — that are miss-
ing in geometric methods such as adding dynamic effects (skin sliding [8], wrin-
kles [9] or jiggling [10]). Simulating the internal structure of the body with
bones, muscles and fat tissues, can also help adding realism to the animations
but often implies a trade-off with computational costs [11], [12]. This means
that this type of skinning is not very well-suited for applications in video games.

**Simulation-based skinning** is based on the observation of real examples to
figure out the deformations of the character models. The examples may have
different forms. The most common one is the use of blendshapes [13], that
compute the deformations as a linear combination of basic poses. This method
is often used in facial animation applications. Other methods require samples of
movements to learn the physical behaviour of the models. The Smooth Skinning
Decomposition [14] with Rigid Bones determines the best skinning weights and
the best rigid bone transformations based on the observation of an animated
model. This method can however suggest to use bones that are not meaningful
in terms of character animation and that can be hard to manipulate in post
animation processes. The Fast and Deep Deformation Approximations method
[15] uses a machine-learning-based algorithm to learn accurate deformations
from training data made of correct animations. This method does no provide
controllers for the animation which would also be an issue for post animation if the deformation results were to be adjusted.

1.2.2 Binding methods

By binding methods, we designate the computation of the skinning weights used in the geometric skinning methods previously presented, that is to find how much each bone transform affects each vertex.

A possibility left for the artists to obtain correct animation movements is to work on the distribution of weights applied to the models. Weights are associated to each vertex of the skin in relation to each bone of the skeleton to define how the geometric surface deforms according to skeleton poses. There are different methods to associate each joint of the skeleton hierarchy to the set of vertices of the mesh. With rigid skinning, only one joint can influence each point. In smooth skinning, one point can be influenced by more than one joint. Each joint has a weighted influence on the given point so that the sum of all weights equals one.

There are few papers in the literature that present automatic methods to compute the weight values but it is established that good skinning weights should have the following properties:

- they should not depend on the mesh resolution,
- they should vary smoothly along the surface,
- they should be roughly linked to the distance from the joint to the surface.

For instance, there are four binding methods implemented in Maya [16]:

- **Joint proximity**: the joint that has the greatest influence on a vertex is the one that is the closest to that vertex in world space.
- **Skeleton hierarchy**: the joint that has the next greatest influence on a vertex is a joint linked to the closest joint in world space.
- **Heat map**: this method uses a heat diffusion technique to determine the distances between joints and vertices based on [17].
- **Geodesic voxel**: uses geodesic distances between the voxels of the model’s voxelization based on [18].

The two first methods assign bone weights purely based on proximity in world space so they ignore the character’s topology. They would thus fail in most situations that are not basic ones (Figure 10).
Figure 10: Artefacts obtained with joint proximity binding method

The Heat map method bases its distance computation on heat equations by considering the character volume as an insulated heat-conducting body and setting the temperature of one bone to be 1 while the temperature of the others is set to 0. The equilibrium temperature at each vertex on the surface gives the weight of the bone at that vertex. However this method is assuming that the character has a well-defined interior as well as the surface of the mesh. It thus does not work on non-manifold geometry which is often a characteristic of production meshes.

The Geodesic voxel method computes a voxelization of the model to be able to conserve the consideration of the character’s topology when attributing skinning weights. The use of voxelization is a pertinent choice and we will use this idea in our own proposal.
Automatic methods are not self-sufficient and still often require a highly skilled skinning artist to correct the skinning weights attributed to the models by painting them by hand on top of the meshes’ surfaces (Figure 11). This is a time-consuming task and difficult as well. Sometimes the artists would rather start their painting of skinning weights from scratch than to spend time finding the specific areas in which they have to adjust the work of an automatic skinning system. Especially if the automatic skinning methods generate unexpected results on some meshes and not on others without any explanation. Another solution to speed the skinning process up is to reuse existing skinning information of a already skinned model onto a new model.

1.3 Copy Skin

Copy skinning corresponds to the action of copying the deformation of a source rigged skin to a target skin.

[19] is one of the first paper to mention retargetting motion between characters of different sizes with similar structures. Most works in the literature about transferring animation from a source to a target address the issue by retargeting the source character deformation [20]. In [21], they compute an automatic registration between the source and the target meshes for facial animation. [22] is based on a database of partial rigs from a set of source characters and then finds correspondences with the target mesh using a semi-automatic method.

However, these methods do not compute skinning weights for the target meshes which greatly limits their applicability in a typical animation pipeline where skinning weights are needed for animating the characters. The copy skin methods implemented in Maya [23] tackle the copy of skinning weights between
a source skin and a target skin providing different possibilities:

- **Closest point on surface**: finds the closest points between the source and target surfaces and smoothly interpolates the skin weights at those points.

- **Ray cast**: uses a ray-casting algorithm to determine sample points between the two surface meshes.

- **Closest component**: finds the closest polygons at each sampling point and uses its skin weights value without interpolation.

- **UV space**: uses the UV texture coordinates for the sampling of skin weights.

These methods are almost all based on finding the closest point between the two meshes.

In [24], the correspondence between the source character rig and the target character mesh is semi-automatic and the retargeting concerns both the skeleton and the skinning weights. [25] presents a method for transferring skinning weights to clothes by projecting the clothing vertices onto a character skin. They use the barycentric coordinates of the projected points except in more complicated areas such as the armpits where continuously moving planes are constructed and used as projection reference planes.
2 Overview

2.1 Motivation

The motivation for the subject was mainly based on the study of the Geodesic Voxel Binding algorithm by Autodesk [26]. This binding method was added to the 2015 version of the software Maya but still not used by the Quantic Dream artists in 2019. The pertinence of the results obtained using this method made no doubt, especially in specific areas where other binding methods would fail, like the fingers or the armpit area. However, it was not uncommon to observe some unexpected behaviours that stayed unexplained in addition to the lack of understanding on how to make the correct choice for the method’s parameters values. The first part of my work was first to go through the Geodesic Voxel Binding paper and implement my own version of their algorithm using the Maya API.

A second part of the work was to study how the Geodesic Voxel Binding method could be used for copy skin applications. The ability to copy skinning weights is often used by the skinning artists. Indeed, during the production pipeline of a video-game, the modellers shape the different characters of the game. They start by working on naked shapes that correspond to the body of the characters without clothes which gives a global idea of the morphology of the different characters. The animators start to work on how to animate the characters and paint skinning weights on the naked shapes. These weights will then be used as reference for the different variations of the characters. For example, a body that is wearing a t-shirt will have a different behaviour in the torso area but the arms and hands will behave the same as the naked body. The weights can be directly copied from one model to another in these specific areas. Also even if the weights are different from one naked body to one wearing clothes, it is easier for animators to start with a base similar to the naked body one and to modify from there rather than from scratch or from too imperfect weights.

There are some copy skin algorithms already implemented in Maya that were used by the artists in some specific cases. However, they often encountered undesirable behaviours in what could seem like basic cases such as copying weights between two identical meshes. Copying weights from a naked mesh to a clothed one could also generate errors similar to the skinning ones in the armpit area and leg gap. It was thus interesting to understand how the Geodesic Binding Algorithm was capable of taking into consideration the topology of the character for the application of the skinning weights and then see if a similar method could be applied for the transfer of skinning weights from one character to another.

What was asked of me at Quantic Dream was to work closely with the skinning artists to get a comprehensive idea of what their tasks consist of and to find out what could be done to simplify them. A lot of time can be lost on what could seem like small details for modellers such as growing a character of some centimeters or giving it a slightly bigger nose or adding a fold of cloth. All these modifications require a new work from the skinning artists and they
even have to start everything from scratch sometimes. My work was focused first on getting familiarized with the software used by the company, Autodesk Maya, studying especially the skinning techniques already implemented in the software. Then I studied more specifically the Geodesic Voxel Binding algorithm that offers a way of considering the topology of the models while binding a skin. Implementing some parts of this algorithm allowed me to adapt it to the Quantic Dream models and to use them for a new application in copy skin.

2.2 Work scope

At Quantic Dream, the software used for character modelling, level designing and animation is Autodesk Maya. A team of 5 people is dedicated to creating tools useful for the Maya users in the studio. It covers multiple types of tools such as importing and exporting characters, tools to alter the viewer and the different tabs of the software accordingly to the most-used buttons, implementation of algorithms to avoid repetitive tasks, etc. I was part of this team during my internship as I was implementing a new tool in Maya. I worked with the Maya API to implement an algorithm that would be efficient on the models created by Quantic Dream. I wrote a Maya plug-in that is presented as a unique command line for the artists to use. Plugins in Maya can be written in MEL, Python or C++. Python is convenient to write quickly prototypes or small plugins but is soon limited by performance issues. The C++ API is more complicated to understand but it allows to do many more computations with a better performance.

2.3 Objectives

The aim of this work was to answer a need the character artists had at Quantic Dream to have an efficient tool for skinning, will it be skinning from scratch or from another model directly. Implementing a custom version of the geodesic voxel binding algorithm was a good opportunity to understand where the power of the algorithm lies and how it can be adapted for the specific needs of the artists of the studio. Some particular cases linked to the skeleton used at Quantic Dream and that could not be planned by Maya, can be treated in the custom design of the tool. It also gives more transparency to the process which leads to a better understanding of the parameters we could tune and the obtained results.

Creating a new tool means to also spend some time listening to the needs of the users. The aim of the tool is to be used by the artists because they understand easily how to use it and find it pertinent in its effects. If a tool is not well-designed and does not answer their needs properly then it will not be used. This involves the creation of a proper User Interface (UI) with the adequate options for the artists to use for their particular practical cases as well as an explanatory document to present the functionalities.

In this report, the automatic skinning system developed will be presented as well as the UI interface that was designed accordingly to the artists’ needs. The method used is to a large extent based on the Geodesic Voxel Binding algorithm and was extended to a custom application in copy skinning. The proposal was
developed as a Maya plugin in C++ using the Maya API. The UI was written in Python using the Qt toolkit for Maya.
3 Development of the proposal

3.1 Proposal Overview

3.1.1 Characteristics of the system

The system we developed presents several features that were not in the current Maya version:

- **Symmetrical**: on symmetric objects, the skinned result obtained is symmetrical as well.

- **Meaningful**: the tool options are customized so that the artists can understand them. For example, a specific computation of the necessary voxelization resolution was added so that the resolution was not set randomly.

- **Works on simple cases**: the tool fixes the issues encountered with Maya tool even for basic cases.

- **Customized for Quantic Dream’s models**: when working on the company models, the nomenclature is the same in all the characters. It is thus easier to customize the algorithm to be more efficient on the company skeletons using the name of the joints for example.

- **Customized tuning**: some options were added to meet the expectations of the artists and the specific cases they usually come across while working on skinning.

3.1.2 Process presentation

The proposal is split in two main processes: one automatic binding algorithm and one copy skin algorithm. We will present the main steps of the two algorithms.
Custom Geodesic Voxel Binding (Figure 12) Starting from a mesh that can be open and a skeleton, we voxelize the mesh using a hardware-accelerated method. We compute geodesic distances with a form of Dijkstra’s algorithm and define the skinning weights with a falloff function applied to the previously computed distances.
Geodesic Copy Skin (Figure 13) Starting from a source mesh and a target mesh, we voxelize the border of the source mesh and the entirety of the target mesh. We compute geodesic distances between the two meshes. For each target mesh vertex, we perform a hit test to find the closest point in the closest voxel. We compute the skinning weight of the vertex using the barycentric coordinates of the closest point.

3.2 Custom Geodesic Voxel Binding

The developed proposal was inspired to a large extent by the Geodesic Voxel Binding algorithm [27]. The purpose of the Geodesic Voxel Binding is to have the skinning weights be attributed taking into consideration the topology of the model. We want to have coherent skinning weights attributed to the different limbs of the character. For example, this means that we do not want the left leg vertices to move with the right leg bones. It seems like a quite obvious principle to think about but if we want to compute it automatically, it gets more complicated. This is why the authors of the Geodesic Voxel Binding algorithm
came up with the idea of using the voxelized version of the model to give the algorithm the sense of the model’s topology. By filling a model with voxels and computing a distance between voxels instead of a world space distance, we would avoid typical automatic skinning techniques problems of ignoring the empty space of the meshes.

3.2.1 Voxelization

![Figure 14:](image)

**Figure 14:** Voxelization process: Starting from an initial mesh and skeleton (a) that can be open or non manifold (b), we voxelize the mesh. The voxelizations of the border (in blue (c)) and the interior (in red (d)) are two separated processes.

The specificity of the voxelization for the algorithm to work on production meshes is that it has to be a solid voxelization. It means that we want to voxelize the border of our model as well as the interior. Solid voxelization on closed meshes is a common problem that is usually treated by counting the number of intersections of a certain number of rays with the surface defined by the quadratic mesh. However, this technique would not be efficient enough or require a too important number of cast rays to have a good solid voxelization result. In video games, most of the character meshes are open. The clothes are not filled nor closed (Figure 14b). The algorithm had thus to take this aspect into consideration and to perform a voxelization that would be coherent even if the mesh is not closed.

To do so, the process was divided in two classification processes:

- Border voxels classification (cf. Algorithm 1)
- Interior voxels classification (cf. Algorithm 2)
The rest of the voxels are tagged as exterior.

**Voxelization of the border (Figure 14c)** As the border of the mesh is defined by triangles, an intersection test between the voxels of the grid and the triangles of the mesh would suffice to define if a voxel lies on the border. To speed the process up, an octree structure was necessary. The octree nodes would store the triangles intersecting inside the volume of the node and would pass them on to their children. We subdivide the octree nodes as long as we do not reach the maximum depth defined by the desired resolution of voxelization. Once at this level of division, we have equivalency between the nodes of the octree and the voxels of our voxelization. We can tag the voxels intersecting with triangles as border ones. The intersection test performed between the triangles of the mesh and the box defined by the octree node is Akenine-Möller’s Fast Triangle-Box Overlap Test [28].

**Algorithm 1: Border voxelization**

1. Create empty octree nodes queue \( Q \);
2. Add \( O \) root to \( Q \);
3. while \( Q \) is not empty do
4. Pop \( q \) from \( Q \);
5. if \( q \) is not at max depth and \( q \) intersects with some triangles then
6. Subdivide \( q \);
7. Add \( q \).children to \( Q \);
8. end
9. else if \( q \) intersects with some triangles then
10. Tag \( q \) as border;
11. end
12. end

**Voxelization of the interior (Figure 14d)** To voxelize the interior of the model, the method used is similar to the one of the original Geodesic Voxel Binding. It is a hardware-accelerated method based on z-buffer slicing. The algorithm performs a slicing of the model by moving the near clip plane along the 3 main directions (\( x, y, z \)) and renders the mesh to an off-screen buffer of size \( dx \times dx \) with \( d \) the resolution of the voxelization. The mesh is rendered with the back faces in white and the front faces on top in black. We superimpose corresponding slices in the positive and negative direction for each of the 3 main directions (i.e. \( +x \) and \( -x \)) with the logic operator OR. We have an equivalence between one pixel of the buffer for a particular slice and a voxel of the voxelization. A voxel is thus classified as interior if the corresponding pixel is white in at least one of the 3 directions (Figure 15).

To limit the number of draw loops as much as possible as it is a rather costly operation, we set the draw actions to 2 per slice in each direction. In comparison, the method presented in the original Geodesic Voxel Binding paper suggested to first render the back faces and then the front faces but this operation was realized simultaneously in our proposal by using a custom fragment shader and
by setting the correct order for the front faces and back faces with the OpenGL function `glFrontFace` (see Appendix A for the shaders’ code). Also, instead of drawing separately the slices in the two opposite directions (i.e. $+x$ and $-x$) and joining the pixels of the two off-screen buffers, using the same off-screen buffer and drawing the two directions with the blending function OR allows us to get the same result by reading only one off-screen buffer.

**Figure 15:** Slices example of a character’s top for interior voxelization in the three directions ($x$, $y$, $z$)
Algorithm 2: Interior voxelization

Input: Voxels $V$, Triangular mesh $M$

1. foreach view direction $x, y, z$ do
2. foreach volume slice do
3. Set View and Projection matrices in the positive direction;
4. Draw $M$ to off-screen buffer;
5. Set View and Projection matrices in the negative direction;
6. Draw $M$ to off-screen buffer;
7. end
8. foreach pixel of the slice corresponding to voxel $v_i$ do
9. if $pixel.color = white$ then
10. $p_i += 1$;
11. end
12. end
13. end
14. foreach voxel $v_i$ do
15. if $p_i > 1$ then
16. Tag $v_i$ as interior;
17. end
18. end

3.2.2 Distance computation

As for the Closest Joint skinning method, the Geodesic Voxel Binding is based on the computation of a closest distance between a vertex and a joint. The closer the joint is to the vertex, the higher the skinning weight associated to this joint for the considered vertex will be. The difference lies in the way we compute the distance. In the case of the Closest Joint skinning, the distance is simply the world distance between two points. For the Geodesic Voxel Binding, we want to have a distance that has a meaning in terms of the model topology. To do so, a pathfinding algorithm was used to compute the shortest path between each bone and each border voxel.

A bone is represented by a segment (between a joint and its child) or several segments (between a joint and each of its children). We select the voxels intersecting with the bone segment(s) as source for our path finding algorithm. Our destinations are all the border voxels. Since we have a multiple source multiple destinations problem and we want distances between all sources and all destinations, we use a Dijkstra’s algorithm. The algorithm used is the same as the one suggested in the Geodesic Voxel Binding paper (cf. Algorithm 3).

As every bone treatment is independent to the others, this process is done in parallel to speed up the computation.
Algorithm 3: Geodesic distance computation

Input: Volumes V, Skeleton S

1. foreach bone b_i of S do
   2. Create empty voxels queue Q;
      // Initialize distances
   3. foreach non-exterior voxel v_i of V do
      4. if v_i intersects with b_i then
         5. d_{v_i} = 0;
         6. Push v_i to Q;
         7. Mark v_i as visited;
      else
         9. d_{v_i} = ∞;
      end
   11. end
   // Compute geodesic distances
   12. while Q is not empty do
      13. Pop v_i from Q;
      14. foreach non-exterior neighbor v_j of v_i do
         15. temp_dist = d_{v_i} + |v_i − v_j|;
         16. if temp_dist < d_{v_j} then
            17. d_{v_j} = temp_dist;
            18. if v_j is not visited then
               19. Mark v_j as visited;
               20. Push v_j to Q;
            end
         end
      23. end
   25. end

3.2.3 Weight computation

Once we have computed the distances between every border voxel and every joint of the skeleton hierarchy, we can use this value to define the skinning weights. Since the skinning weights are applied to the vertices of the mesh, for each vertex, we have to find the voxel to which it belongs and get the corresponding distances. From the distances, we compute the skinning weight associated to each bone with (8).

\[ w_{i,v} = \left( \frac{1}{(1 - \alpha) d_{i,j} + \alpha d_{i,j}^2} \right)^7 \] (8)

Where \( d_{i,j} \) is the distance from vertex \( i \) to joint \( j \) defined by (9), \( d_{i,v} \) is the geodesic distance between the voxel the vertex belongs to and the joint \( j \).

\[ |p_{vertex} - p_{voxel}| \] is the distance between the vertex and the center of the voxel it belongs to. \( D \) is the sum of the bounding box extents.

\[ d_{i,j} = \frac{d_{voxel,j} + |p_i - p_{voxel}|}{D} \] (9)
3.3 Geodesic Voxel copy skin

When it comes to copy skinning weights from one mesh to another using the closest point method, it often occurs that we encounter the same kind of issues as for the binding in areas such as the armpits or between the legs. The reasons of this are also linked to the computation of a world space closest distance that does not take into consideration the empty space of the model. It was thus interesting to find a way of using the power of the geodesic voxel binding algorithm that solves a number of similar problems in skinning and try to apply it for copy skinning.

As the power of the Geodesic Voxel Binding algorithm lies in the use of the voxelization to compute the distances between the vertices and the joints, it seemed interesting to use the voxelization process for copy skinning too. Both source model and target model could be voxelized using the same resolution and we would have an equivalency between the voxel grid of one and the voxel grid of the other. With both models voxelized, we could find an interesting way of computing a distance that would help us find a closest point taking into consideration the models’ topology.

Based on this idea, the developed proposal first computes the voxelization of the source and target meshes (Section 3.3.1). Using geodesic voxel distances, we find the shortest path from target mesh to source mesh (Section 3.3.2). This allows us to restrict the search for closest point to the closest voxel (Section 3.3.3) and helps preserve the sense of topology when copy skinning. A necessary additional step is finding the equivalence between joints of the source and target mesh (Section 3.3.4). An extra option of normalization was also added in the proposal (Section 3.3.5).

3.3.1 Voxelizations

The first step of the copy skin algorithm is to voxelize the source mesh and the target mesh. To be able to make an accurate comparison between the voxelizations, they are done at the same resolution in the same volume — the volume being the smallest bounding box containing the two meshes.

Source mesh voxelization In the case of the source mesh, we are interested only in the border voxels. This is done similarly as for the skin binding by using an octree structure and intersection tests voxel-triangle (cf Algorithm 1).

Target mesh voxelization The voxelization of the target mesh is done entirely, interior and border voxels (cf Algorithm 1 and Algorithm 2).

3.3.2 Distance computation

Once both models voxelized, we compute the shortest path from each border target voxel to the border source voxels. We have a several sources path finding problem. As for the binding, we use a form of Dijkstra’s shortest path finding. The algorithm needs to be processed only once with all source voxels as sources
and target voxels as destinations and a storing of the route taken along the way (cf Algorithm 4). The distance computation is performed only once compared to the distance computation for binding that is performed once per bone of the skeleton hierarchy.

Algorithm 4: Geodesic distance computation for copy skin

Input: Source Voxels $V_S$, Target Voxels $V_T$

1  Create empty voxels queue $Q$;

// Initialize distances
2  foreach non-exterior voxel $v_i$ of $V_T$ do
3      if $v_i$ is non-exterior in $V_S$ then
4          $d_{v_i} = 0$;
5          Push $v_i$ to $Q$;
6          Mark $v_i$ as visited;
7          $parent[v_i] = v_i$;
8      else
9          $d_{v_i} = \infty$;
10     end
11  end

// Compute geodesic distances
12  while $Q$ is not empty do
13      Pop $v_i$ from $Q$;
14      foreach non-exterior neighbor $v_j$ of $v_i$ do
15          temp_dist = $d_{v_i} + |v_i - v_j|$;
16          if temp_dist < $d_{v_j}$ then
17              $d_{v_j} = temp_dist$;
18              $parent[v_j] = v_i$;
19              if $v_j$ is not visited then
20                  Mark $v_j$ as visited;
21                  Push $v_j$ to $Q$;
22          end
23      end
24  end
25 end
3.3.3 Closest point

The purpose of the distance computation is to find, for each border target voxel, the closest border source voxel. As we stored the parent of each voxel along the shortest path finding, we reverse the path to find the closest voxel (cf Algorithm 5).

Algorithm 5: Closest voxel determination

Input: Target Voxels $V_T$

1. **foreach** non-exterior voxel $v_i$ of $V_T$ **do**
2. \[ \text{parent} = \text{parent}[v_i]; \]
3. \[ \text{while} \text{parent is not parent[parent]} \text{ do} \]
4. \[ \text{parent} = \text{parent}[\text{parent}]; \]
5. **end**
6. \[ \text{closestVoxel}[v_i] = \text{parent}; \]
7. **end**

For each point of the target mesh, we want to find the closest point on the source mesh and apply the skinning weights of this point to the corresponding point on the target mesh.

The closest voxel obtained with Algorithm 5 is a border voxel of the source voxelization. Since it is a border voxel, it intersects with a certain number of triangles of the source mesh. We consider that the closest point belongs to one of the triangles intersecting inside the closest voxel previously found (Figure 16). We first find the closest triangle among the triangles intersecting inside the closest voxel. Then, we find the closest point on this triangle by performing a hit test (Figure 17).

**Figure 16:** Closest Voxel: For each border target voxel (red), we find the closest source border voxel (blue)

**Figure 17:** Closest Point: we find the closest point in the closest voxel and get its barycentric coordinates
This gives us the barycentric coordinates of the closest point on the source mesh \((\alpha, \beta, \gamma)\). We use these barycentric coordinates to weigh the skinning weights of the triangle vertices. Equation (10) is used to compute the weight of one target vertex with respect to the joint \(j\) when the closest point belongs to the triangle \((v_{\text{source}1}, v_{\text{source}2}, v_{\text{source}3})\).

\[
w_j(v_{\text{target}}) = \alpha \cdot w_j(v_{\text{source}1}) + \beta \cdot w_j(v_{\text{source}2}) + \gamma \cdot w_j(v_{\text{source}3}) \tag{10}
\]

**Particular case** In some particular cases, the closest point can fall on several triangles. For example, this can be the case when we copy skinning weights from one model to the same one when a point is shared between different triangles. It is not a problem as long as the points in the same location do not belong to two separated parts of the mesh. A recurrent problem at Quantic Dream happened when attempting to copy meshes of the head of the characters. If the points of the two lips were at the same position, we would have artefacts such as the closest point found by the Maya copy skin algorithm would sometimes choose a point on the upper lip for points of the lower lip.

To avoid these problems, an additional test is performed when more than one triangle vertex is found at the position of the closest point. To select the correct closest point among the several possibilities, we consider the different triangles to which the closest point belongs to. The point has a different vertex normal in each of these triangles. We select the point belonging to the triangle in which the vertex normal associated has the closest value to the vertex normal of the point we consider in the target mesh (cf Algorithm 6 and Figure 18).

![Figure 18: Selection among closest point possibilities by checking the vertex normals in the triangles the closest point belongs to. For example here, \(P_3\) would be selected.](image-url)
Algorithm 6: Find best closest point to \( v \)

**Input**: Vertex \( v \) of the target mesh, closest point \( P \) of the source mesh

1. \( n = getVertexNormal(v) \);
2. \( dot = -1 \);
3. \( closestPoint = P_0 \);
4. \textbf{foreach} adjacent polygon \( i \) of \( P \) \textbf{do}
   5. \( n_i = getVertexNormal(P_i) \);
   6. \( tempDot = n_i \cdot n \);
   7. \textbf{if} \( tempDot > dot \) \textbf{then}
      8. \( tempDot = dot \);
      9. \( closestPoint = P_i \);
   \textbf{end}
5. \textbf{end}

3.3.4 Joint equivalence

As mentioned previously, the equation (10) gives us the computation of the skinning weight of one target vertex associated to a joint \( j \) that is an influence of the source mesh. However, target and source can have different skeleton hierarchies and it is needed to find the equivalence of the source influences for the target mesh.

In the proposal, we used the benefit of knowing the nomenclature of the skeleton design at Quantic Dream to create a custom treatment of the joint equivalence since the skeleton creation is normalized for all the characters produced for one game. Finding joint equivalence between two characters of the same production is thus straightforward as all the joints of the source skeleton are present in the target one as well. We thus base our joint equivalence directly on the names of the the joints in the skeleton hierarchy. This method was preferred to a localization check because of errors that can be generated if some joints have the same localization.

For same production meshes, there are two possible cases we can encounter:

- If the target mesh is not yet bound to any skeleton hierarchy: the target mesh is bound to the source skeleton hierarchy with the same influences as the source mesh and the weights can be directly applied.
- If the target mesh is already bound to a skeleton hierarchy: the proposal checks if the influences of the source mesh are already influences of the target mesh. If needed, the missing influences are added since they are present in the skeleton hierarchy. Then the computed weights are applied to the influences one to one.

It gets more challenging when the meshes we consider do not have the same skeleton hierarchy, which can be the case when copying from meshes of different productions. In this situation, the skeleton is usually only slightly modified and the changes correspond to a deletion or addition of bones. The choice we made for the proposal in these cases is to try and find common bones as much as
possible between the two hierarchies. When a specific joint is not found, we attribute its weight to its closest parent in the skeleton hierarchy.

### 3.3.5 Normalization

A last option was added to the copy skin functionality of the proposal that computes normalized versions of the meshes before computing the copy skinning. It was added in order to deal with different scales or different locations of the characters. It corresponds to transforming the vertices of the mesh so that it fits in a unit bounding box using Equation (11).

\[
v = \frac{v - Bbox_{\text{min}}}{Bbox_{\text{max}} - Bbox_{\text{min}}} \tag{11}
\]

where \(v\) is a vertex of the mesh and \(Bbox\) is the bounding box of the mesh.

### 3.4 Tool UI

In this section, we will present the layout of the user interface (UI) that was developed according to the artists’ needs. In Figure 19 is shown the global look of the UI.

![Figure 19: General design of the tool UI](image)

The UI is divided in three parts: the geodesic voxel binding tab, the geodesic copy skin tab and the help bar.

#### 3.4.1 Geodesic Voxel Binding tab

The geodesic voxel binding tab (Figure 20) is composed of the different options useful to perform the geodesic voxel binding computation. There are four of them:
• the resolution of the voxelization,
• the maximum number of influences,
• the possibility to separate mesh treatment,
• the possibility to consider specific joints as influence.

![Figure 20: Geodesic voxel binding tab](image)

The resolution of the voxelization determines how small the voxels of the voxelization are going to be. For instance, a voxelization of resolution 256 means that the bounding box of the model is going to be divided in 256 on each of its dimension (width, height and depth). The bigger the resolution is, the more the bounding box is divided into voxels and the smaller the voxels are going to be. This means that we can get a more detailed voxelization with a bigger resolution but also that the computation will be slower as there will be more voxels to go through.

The maximum number of influences limits how many joints can influence one vertex. This data is interesting in terms of the obtained results that can be improved by a good choice of maximum number of influences but also because of performance limitations from the game engine that often restricts the maximum number of influences.

The two previous options were already available in the Maya version of the algorithm. We added two more options that seemed useful for the artists’ use of the tool.

When selecting several meshes to bind, it seems like the Maya tool treats the meshes as a whole entity and not separately. This is obviously quicker as the voxelization is performed only once for the entirety of the model. However we may desire sometimes to have a separated voxelization for each model as we will consider the bounding boxes of the meshes separately for the resolution of the voxelization. Therefore selecting the chest and the hands together at a certain resolution of voxelization may not result in a voxelization that is precise enough to have separated fingers but treating them separately would provide
the expected result.

The last option available for the weight computation is the possibility to consider the use of the additional joints as influences. At Quantic Dream, these specific joints all start with the prefix "EX_." It can be interesting to sometimes include them as influences as they may add a roll effect on a t-shirt for example but most of the time they are quite particular joints and their influence is very specific on the desired resulting effect from the joint movement. Including them in the computation might produce curious results in most cases.

An additional option available is not one that affects the computation of the geodesic voxel binding weights but one that helps the artists choose an appropriate resolution of voxelization for the mesh they consider (Figure 21). It is an option that we felt was necessary because the only indication of the voxelization resolution number did not feel really meaningful for the artists and for the mesh they would consider. Indeed, one could think that choosing the maximum number of voxels for all the meshes would give the best results in every case. But given the time of computation implied, it would be a great waste of time to choose the highest resolution of voxelization even for the meshes that do not require it. This is why we added this indication of voxelization resolution that relies on giving the biggest resolution at which the size of a voxel is greater than the minimum distance between two vertices of the mesh. This is not necessary a sufficient resolution of voxelization right away but at least it gives the artists a good idea of a resolution at which they can start working with.

![Figure 21: Example of getting a resolution indication for a unit sphere](image)

### 3.4.2 Geodesic Copy Skin tab

The geodesic copy skin tab (Figure 22) contains three slots dedicated to the selection of the source mesh and the source and target root paths. Selecting the root paths in addition compared to the options of the Maya tool that does not require any information on the skeleton hierarchy. We added this selection in order to get a better result on the Quantic Dream meshes, using the nomenclature of the skeletons instead of the position of the joints in world space to find
equivalecencies between the joints of the source and target meshes. It has then two options for the copy skin computation:

- the resolution of voxelization,
- the meshes normalization.

Figure 22: Geodesic voxel binding tab

The resolution of the voxelization, similarly to the geodesic voxel binding, determines the size of the voxels in the bounding box of the two meshes. We have the same case as for the geodesic voxel binding, that is the bigger the resolution, the slower the computation. However the loss of time is more limited in the copy skin case as the geodesic distance computation is performed only once instead of once by joint of the skeleton hierarchy.

The normalization of the mesh option was implemented in order to give the artists the possibility to consider similar models but with different scaling factors or different positions in space. If this option is checked, the models would be normalized in a unit bounding box centered at the origin before computing the voxelization and performing the copy skin. It would thus erase any consideration of the position and scaling if the models are similar.

To execute the algorithm, the user has to fill the slots for the source, the source root path and the target root path by selecting them in the scene and pressing the button next to the corresponding slot to fill the text bar. They select the desired voxelization resolution in the tool tab. Then, the user selects the target of the copy skin that can be one or several meshes or some vertices only and press the Copy skin button to start the computation.

3.4.3 Help bar

A help bar was added at the bottom of the tool to display visible information for the users. A help button gives access to a help page written on the company intranet to explain to a user how to properly use the tool and what the different
In addition, two types of messages are displayed: error messages or information messages. As indicated by their names, error messages are used to display an error in the computation of the algorithms. They are displayed in red so that the user can see right away that there is a problem with the execution of the calculation (Figure 23).

![Figure 23: Examples of displayed errors in the tool UI](image)

Other information messages can be displayed to indicate the success of the computation (Figure 24).

![Figure 24: Success messages displayed in the help bar](image)

Besides these messages, some popups appear when the user selects wrong types of objects for the copy skin slots that indicate them what they need to select instead (Figure 25).

![Figure 25: Popup windows when wrongly selecting objects for copy skin](image)
4 Evaluation of the proposal

4.1 Geodesic voxel binding

The results obtained with our custom version of the geodesic voxel binding are similar to the original geodesic voxel binding algorithm implemented in Maya so we will not present them thoroughly. Still, some artefacts from the original geodesic voxel binding could be avoided in our own version with a modified falloff function and our voxelization process (Figure 26).

(a) Our modified falloff function helps preventing a too important bulge in the armpit area (original GVB on the left, our custom version on the right)

(b) artefacts in the fingers eliminated with our voxelization method (original GVB in the foreground, our custom version in the background)

Figure 26: artefacts from the original Geodesic Voxel Algorithm that are not present in our own version
4.2 Geodesic voxel copy skin

One of the direct applications of the developed proposal is to be able to copy skinning weights from a naked body to a clothed body. This is achieved by our proposal as shown in Figure 27. But more importantly, we wanted to avoid the usual artefacts encountered by the skinning artists while copy skinning with Maya tools. Maya offers several techniques for copy skinning but we will show how the use of our proposal only solves the problems we come across while using the Maya tool.

4.2.1 Contribution of the use of voxelization

Using voxelization was a way to maintain the sense of the model topology while copy skinning. While using Maya copy skin tools, we would get artefacts similar to the geodesic voxel binding ones (Figure 28a) or get a totally unusable result (Figure 28b).
Figure 28: Maya copy skin results: (a) Closest point on surface method, (b) UV space method

Using the same simple example as in Figure 28, the geodesic voxel copy skin is able to copy the skinning weights as expected (Figure 29).

Figure 29: Results of the geodesic voxel copy skin on a simple model: (a) skinning weights on the target mesh, (b) skinning weights on the source mesh

As for the geodesic voxel binding, the resolution of voxelization determines the size of the voxels in the bounding box and sometimes it is necessary to go to a higher resolution in order to have an adequate result in risky areas such as between the legs, the fingers or the armpits (Figure 30).
Additional results on more complex meshes are shown in Appendices B and C.

4.2.2 Contribution of the normal check

When we asked the artists about the typical problems they would encounter while copy skinning, the **vampire mouth** was the first one raised (Figure 31). This type of artifact would show up as soon as the points on separated surfaces overlapped. This would typically happen on the mouth area where the modellers shut the mouth by superimposing the points of the upper lip on the points of the lower one.
Figure 31: Vampire mouth: example of common copy skin artefacts obtained with the Maya algorithm when points on separated surfaces are overlapping.

With the fixes added in our copy skin method by checking the vertices’ normals, we were able to eliminate these artefacts (Figure 32).

Figure 32: Copy skin results with the Geodesic Copy Skin method.
4.2.3 Contribution of the normalization

The following examples are using copy skin on one mesh to show the specific artefacts linked to normalization that are even more obvious if shown on the same mesh as source and target.

Figure 33: Copy skin results obtained with Maya tool on models in different location using (a) the closest point method with the same skeleton and (b) the UV space method with different skeletons
Figure 33 presents common artefacts artists encounter while working on skinning models using the Maya copy skin. It sometimes happen that similar characters are placed in different location in world space. As the Maya tool relies on world space distances, the position of the models in world space have a great influence on the results of the copy skin computation. Whether using the same skeleton for both the models or different skeletons or whether using UV space method or closest point on the surface method, all cases fail to provide satisfactory outcomes.

With the addition of the normalization option, users are able to avoid these artefacts in cases of changes of character’s location (Figure 34) or scale (Figure 35).

Figure 34: The Geodesic Voxel Copy Skin method deals with characters in different locations
Figure 35: The Geodesic Voxel Copy Skin method deals with scaled characters
4.2.4 Contribution of the use of the skeleton nomenclature

Using the skeleton nomenclature correspondence between the source and the target allows us to avoid finding correspondence using the world space position of the joints. Checking the world space position leads to undesired behaviours when several joints are in the same position (Figure 36).

Figure 36: Results of the Maya copy skin with several joints at the same position. The skeleton hierarchy is the same for the source and the target but the algorithm does not attribute any influence to the shoulder bone in the target mesh.

When the skeleton hierarchy is the same for the source and the target, the correspondence between the joints of the source and the ones of the target is
immediate and we can avoid the artefacts previously encountered (Figure 37).

![Image](image_url)

(a) Influence of the arm bone on the target mesh  
(b) Influence of the shoulder bone on the target mesh

**Figure 37:** Results of the Geodesic voxel copy skin with several joints at the same position

It also allows us to better consider different skeletons for the source and the target meshes by assigning the weights of the missing bones in the target mesh to the parent in the hierarchy in case of a bone deletion for example. This attribution of weight is the most coherent one according to the skinning artists for weight attribution instead of finding the closest bone in world space (Figure 38).
Figure 38: Results of the Geodesic voxel copy skin with several joints at the same position. From left to right: source mesh weights, Maya copy skin results, weights attributed by the skinning artists, Geodesic copy skin results.
Conclusion

In this thesis, we developed a tool designed to answer a need of the skinning artists and animators. With the perspective to automate an often tedious and time-consuming manual task of the skinning process, the work was split in two main axes: the binding component and the copy skinning process.

The binding process was addressed based on the study of the Geodesic Voxel Binding algorithm. We comprehended the different steps of the algorithm and implemented them as an Autodesk Maya custom plugin. We also identified some of the weaknesses of the algorithm and added custom options to prevent them on the specific framework of Quantic Dream models and applications.

The design of a copy skinning process emerged from several discussions with the skinning artists that voiced out their issues with the various options of the copy skinning tool available in Maya. The list of their needs gave clear guidelines of what functionalities and results they expected to obtain in a good copy skinning algorithm. With the expertise acquired from the binding algorithm, we were able to extract the meaningful techniques that we could use for copy skinning. Using voxelization of the models, we were able to develop a copy skinning technique that would take into consideration the topology of the models we applied it to.

After a testing phase done by the skinning artists, the designed proposal is ready to be used in the next game productions. It will be added to the version of the in-house Maya software at Quantic Dream that will be released at the end of the year for the artists.

For now, the tool answers the needs that the artists had vocalized. As I was hired by the company as the end of my internship, it will be interesting to see how the use of the tool by the artists might bring other issues that we did not foresee. I am looking forward to working with them again on how to improve the tool as much as possible to make it as robust as it can be on every type of model and skeleton.

An interesting evolution of the work in this thesis will also be to study how the techniques developed can be used in other departments. For instance, voxelization techniques are already implemented for illumination purposes in game levels but this hardware-accelerated method could help speed up some time-consuming processes. Likewise, the animation of clothes is based on a technique similar to skinning. Weights are applied to cloth vertices in order to animate them according to the laws of physics. A manual work of painting weights is necessary just like in the skinning process. There are probably interesting applications of the geodesic voxel binding and copy skinning that could be applied to the animation of clothes as well.
References


[16] Bind methods for smooth skinning — Maya — Autodesk Knowledge Network.


[23] Copy smooth skin weights — Maya LT — Autodesk Knowledge Network.


Appendix

A Shaders

```cpp
# version 450
layout (location = 0) out vec3 vert;
uniform mat4 modelview;
uniform mat4 projection;

void main()
{
    gl_Position = projection * modelview * vec4(vert, 1.0);
}
```

geodesic.vert

```cpp
# version 450
layout (location = 0) out vec3 outColor;

void main()
{
    if (gl_FrontFacing)
    {
        outColor = vec3(0.0);
    }
    else
    {
        outColor = vec3(1.0);
    }
}
```

geodesic.frag
B  Geodesic voxel copy skin results on open meshes

In the following mesh examples, a closed skinned model was used as source (middle one) on a target mesh that is open. The mesh on the left represents the model that was skinned by hand by the artists while the mesh on the right is the result of the copy skin algorithm, either the Maya copy skin or the geodesic voxel copy skin.

Figure 39: Results using the Maya copy skin at bind pose

Figure 40: Result using the geodesic voxel copy skin at bind pose
Figure 41: Result using the geodesic voxel copy skin at a random pose. The resulting mesh is similar to the mesh produced by the artists but we can observe some volume loss in the hip area for example. As the source skeleton does not have all the joints that the target skeleton has, this could be quickly fixed by giving the desired influence to the remaining extra joints that do not have influence yet and that help preserving the volume in the source model.
C  Geodesic voxel copy skin results on naked / clothed models

In the following examples, a "naked" body was used as source (middle one) on a target mesh that is representing clothes of the same proportions of the naked body. The mesh on the left represents the result obtained while copy skinning with Maya’s algorithm and on the right with the Geodesic voxel copy skin method. The red arrows indicate the areas in which the Maya method fails while ours works.

Figure 42: artefacts in the armpit region

Figure 43: artefacts while bending the arm
Figure 44: artefacts in the crotch area