



Expertise
and insight
for the future

Nea Kontio

Creating an in-game character cutscene animation

Case study of the game Skábma – Snowfall

Metropolia University of Applied Sciences

Bachelor of Culture and Arts

Media

Bachelor's Thesis

June 2020

Author(s) Title	Nea Kontio Creating an in-game character cutscene animation Case study of the game <i>Skábma – Snowfall</i>
Number of Pages Date	43 pages + 1 appendix June 2020
Degree	Bachelor of Culture and Arts
Degree Programme	Media
Specialisation option	3D Animation and Visualization
Instructor(s)	Peke Huuhtanen, Senior Lecturer
<p>This thesis covers the different stages of creating in-game animations and studies which aspects build up a good and functional cutscene animation from a technical point of view. The thesis concentrates on character animations and briefly covers other supporting steps related to the cutscene creation process.</p> <p>The thesis is divided into two parts. The theoretical part presents the most common industry standard animation and cutscene creation pipeline. The case study section follows the progress of one cutscene animation from the game <i>Skábma – Snowfall</i> by Red Stage Entertainment Oy. The section also offers different solutions and tactics on how cutscenes can be approached with limited resources and shows how they differ from the industry standard methods. This thesis is accompanied by several illustrative videos, in which the progress of the character animations can be seen and the first and second iteration of the full cutscene.</p> <p>The case study makes it possible to conclude that creating quality cutscenes is a long process that acquires multiple steps. It can, however, be streamlined using various tools, physics simulations and ready-made assets. The key factors in making a good cutscene lie with quality animations and making the characters come to life with movements to their clothes and changes of expressions.</p>	
Keywords	animation, cutscene, game, in-game

Tekijä(t) Otsikko	Nea Kontio Pelinsisäisen hahmovälänimaation luonti Tapaustutkimuksena pelin Skábma – Snowfall välianimaatio
Sivumäärä Aika	43 sivua + 1 liite Kesäkuu 2020
Tutkinto	Viestintä
Tutkinto-ohjelma	Medianomi
Suuntautumisvaihtoehto	3D-animaatio ja -visualisointi
Ohjaaja(t)	Peke Huuhtanen, Lehtori
<p>Tämä opinnäytetyö käsittelee pelien välianimaatioiden luomisen eri vaiheita ja tutkii sitä, mitkä kaikki asiat rakentavat hyvän ja toimivan välianimaation teknistä näkökulmaa painottaen. Opinnäytetyö keskittyy eniten välianimaatioiden hahmojen animaatioihin, mutta sivuaa lyhyesti muitakin välianimaatioihin liittyviä työvaiheita.</p> <p>Opinnäytetyö jakautuu teoriaosuuteen ja tapaustutkimukseen. Teoriaosuudessa käsitellään yleisesti välianimaatioiden teon vaiheita. Tapaustutkimuksessa seurataan suomalaisen Red Stage Entertainment Oy:n peliprojektin Skábma – Snowfall yhden välianimaation edistymistä. Opinnäytetyö tutkii ja käy läpi erilaisia välianimaatioiden luomisen metodeita pyrkien laadukkaaseen jälkeen mahdollisimman tehokkaasti ja vähällä vaivalla. Opinnäytetyössä on liitteenä useita havainnollistavia videoita, joissa tutustutaan mm. hahmojen animaatioiden edistymiseen sekä myös koko välianimaation ensimmäiseen ja toiseen versioon.</p> <p>Tapaustutkimuksesta voidaan päätellä, että laadukkaan välianimaation luominen on pitkä prosessi, johon kuuluu monta vaihetta. Näitä vaiheita voidaan kuitenkin tehostaa ja nopeuttaa mm. erilaisin työkaluin, valmisanimaatioilla sekä fysiikkasimulaatioilla. Keskeisiä tekijöitä visuaalisesti kiinnostavan välianimaation luomisessa ovat laadukkaat hahmoanimaatiot hyvillä kasvoanimaatioilla sekä vaatteiden ja hiuksien elävöittäminen.</p>	
Avainsanat	animaatio, välianimaatio, hahmovälänimaatio, peli

Contents

1	Introduction	1
2	3D software	2
2.1	Maya	2
2.2	Unity	2
3	In-game cutscenes	3
3.1	What are cutscenes?	3
3.2	Creating a cutscene	5
3.2.1	Pre-production	5
3.2.2	Production	7
3.2.3	Post-production	9
4	Project: Skábma – Snowfall cutscenes	10
4.1	Cutscene creation process	10
4.2	Animation	12
4.2.1	Animation process	12
4.2.2	Animation reuse: retargeting and editing	15
4.2.3	Exporting	22
4.3	Unity implementation	25
4.4	Unity timelines	28
4.4.1	Animation masks and target aim	31
4.4.2	Blend shape animation	36
4.5	Unity dynamic bone and cloth physics	37
5	Conclusion	38
	References	41
	Appendices	
	Appendix 1. Links to the animations and Unity examples	

1 Introduction

Cutscenes are well-known and frequently used storytelling elements in digital games. They have been used all the way from the first arcade machine games to the current modern AAA titles. They are usually used to break the player's active gameplay section to showcase important aspects of the game world as well as to create progress in the story.

This thesis aims to cover the creation process of in-game character cutscene animations for the game engine Unity. The thesis will demonstrate the creation process through a case study of the game project *Skábma – Snowfall*. It mainly focuses on the production phase of the game, including the full animation process and Unity implementation as well as briefly covering what comes before and after the production phase. The thesis studies what makes a good cutscene through a technical lens, using the said project as an example.

Red Stage Entertainment Oy is a Finnish video game developer focusing on storytelling and narrative. *Skábma – Snowfall* is their magic filled adventure game based on indigenous Sámi stories and mythology. *Skábma*'s concept was piloted in 2018 in a half an hour long proof-of-concept game and further expanded in *Skábma – Snowfall*, making it the first entry in the series. The assets and animations from *Snowfall* are used as a case study in this thesis.

Good cutscenes which create progress in the story are created using methods that make the process efficient and quick, still maintaining a quality that does not break the immersion. Due to this, the scope and the quality of the project is mostly limited by a set time frame but also by the amount of people working on the game. *Skábma* was worked on by around ten people of which two, myself included, were mainly in charge of the animations. As cutscenes serve a huge role as a narrative device in a story driven game, *Skábma – Snowfall* had many of them. This thesis was written for people interested in working with game animations. It serves as a guide for both professionals and hobbyists, informing about the challenges and possibilities that come with cutscene animations.

2 3D software

2.1 Maya

Autodesk Maya, also often referred to as Maya, is an industry leading 3D computer graphics software developed by Autodesk that enables professionals who work with animated film, television programs, visual effects, and video games to create highly professional three-dimensional (3D) cinematic animations. (Edulearn 2016.)

Creating these animations can include models, character rigs, animating the said rigs as well as dynamics, fluids and other simulated effects, all of which Maya is capable of producing.

Regarding the *Skábma* project, Maya version 2019 was used as the main program for almost all 3D related assets and procedures. The high poly character and asset sculptures were retopologized, uv-mapped, rigged and animated in Maya, after which they were moved into the Unity game engine to create the final look for the game.

The only parts of the 3D workflow that were not covered with Maya were the sculpting of the high poly characters and assets as well as the texturing of all models. Sculpting was done with the program Mudbox, which is also developed by Autodesk, better suited for 3D digital painting and sculpting. Substance Painter was used for the textures.

2.2 Unity

Unity, or Unity 3D, is a cross-platform game engine popular with both small developer teams and AAA studios. It is developed by Unity Technologies and was first released in 2015. It gained popularity through the free version Unity Technologies offers which allows developers to create games as long as they do not exceed \$100, 000 in profit with their Unity game. Due to its massive following of users, the software has an extensive library of resources and documentation as well as a huge assortment of videos and tutorials online. (Petty b.)

Unity comes with a wide variety of tools, including shaders, physics-based materials, post-processing and lighting, which enable game developers and designers to make both

2D and 3D games. Additionally it features tools for character animation and cinematic content, but Unity is great for animating almost anything else too.

3 In-game cutscenes

3.1 What are cutscenes?

The best definition of a cutscene according to Hancock (2002), is “*any non-interactive storytelling or scene-setting element of a game*”. Many cutscenes tend to fall under the “any non-interactive storytelling element” aspect but there are examples of in-game cutscenes in which players have some control over their own actions whether it is in the form of choosing dialog options that change the outcome of the game, moving around the camera or even controlling the character itself. This is why the definition of a cutscene also covers “any scene setting element of a game”.

Cutscenes are useful in many more ways than just for creating progressing in the story of the game. They often break the gameplay to, for example, show a conversation between characters, set the mood of the situation or show the effects of a player’s actions. They can be the indicators of when a new game level starts or ends by setting the stage for the action that follows or signal to the player they have achieved the objectives of the particular level or section. The latter can be used in a way that allows players to take a break from the game's difficult tasks by acting as a reward. Cutscenes can also be used as a way to introduce necessary clues and gameplay elements which are needed to finish a gameplay level. Other uses of cutscenes include their usage in marketing to gain viewer excitement. (Schnitzer 2013.)

Games throughout their existence have had many different types of cutscenes. In their earliest form since the mid-70s, they had nothing more than little pixelated characters moving across the screen with some text displaying the story. Nothing more was expected of them either as games overall were a very new invention. As time passed and technology allowed more impressive cutscenes, they started to evolve to forms we are more familiar with today. (Wyatt 2012.)

Live-action cutscenes are a type of cutscenes which use real actors, sets and props to produce short-form films which are then inserted into the game. These types of cutscenes are generally known for being filmed on a low budget as they can often be the

cheapest and easiest option to liven up the story of the game. They were especially popular during the early-to-mid 1990s. (GiantBomb 2018.)

A *pre-rendered* cutscene is a video that has been created beforehand using 3D assets and that plays between gameplay sections. The cutscene can be done to mimic the look of the game in every way so that it is not apparent that the game is playing a cutscene, or they can be made with much higher graphical quality in mind. (Wyatt 2012.) High-quality, almost cinematic level, cutscenes are often created by different teams than the teams creating the actual games. They are often outsourced to outside animation studios as they require more time and bigger resources to create. (GiantBomb 2018.)

In-engine cutscenes, or real-time cutscenes, are the most common type of cutscenes in modern games which are done inside the game engine. This type of cutscene is the cheapest to produce due to the existing game assets that can be reused. (GiantBomb 2018.) As graphics in video games have improved, so has the quality of in-engine cutscenes. There is less need to cut into pre-rendered scenes to show big events in games as the current in-engine scenes can achieve the same results. In-engine cutscenes allow interactivity, where the player, for example, can change the course of the cutscene through their actions. (Wyatt 2012.)

In-engine animations were used for *Skábma – Snowfall's* gameplay cutscenes, partly due to its small scale and the common use of in-engine cutscenes overall in games.

3.2 Creating a cutscene

Animations generally have a similar pipeline no matter what purpose they have been created for. This is also the case with cutscene animations. Some technical aspects like whether the animation is in 3D or 2D can change the workflow, but the basics are the same. The animation process is split into three parts. These parts are pre-production, production and post-production which will be covered in more detail right after the general descriptions starting from chapter 3.2.1.

Pre-production is the starting point of all animations and it involves everything that needs to be done before the actual animation process can start. With all animation projects, this includes having the story and idea for the scenes. These will then be visualized with the help of scripts, concept designs and storyboards. Sometimes animatics are made to showcase the animation plan in more detail. (Beck 2017.)

Production starts when all the necessary planning has been concluded. With game cutscenes the characters need to be brought from 2D concepts to 3D models that can be animated. After this is done the animators start creating character animations. During a game's production phase the rest of the game is produced alongside. Environment assets, locations, sounds and gameplay mechanics are all being built to give the game and its cutscenes their final look. (Beck 2017.)

Post-production is the part where other final touches such as visual effects are added. These include FX effects and post processing. Depending on the project, sounds will also be implemented at this point since developing suitable sounds for animated characters is difficult without seeing the final product. (Beck 2017.)

3.2.1 Pre-production

“Pre-production is the phase in which the elements that lay down the foundation for the production are assembled” (Dowlatabadi & Winder 2013, chapter 8). It starts with the script, conceptual design and storyboarding. The script is the main story of the game written down in text format. Conceptual designs are drawings, images or descriptions of various gameplay, character and environment designs. Once these two are well underway, storyboarding can begin to visualize and pace each cutscene. (Schnitzer 2013.)

A storyboard is a sequence of images which depict the characters who will be presented in a scene and what happens in it. They are traditionally made by drawing quick thumb-nail-like images which give the basic idea of the progression of the story. More detailed passes can be made afterwards but they are not always necessary especially with game cutscenes. It is more important to see a crudely animated layout of the 3D characters. (Dowlatabadi & Winder 2013, chapter 8.)

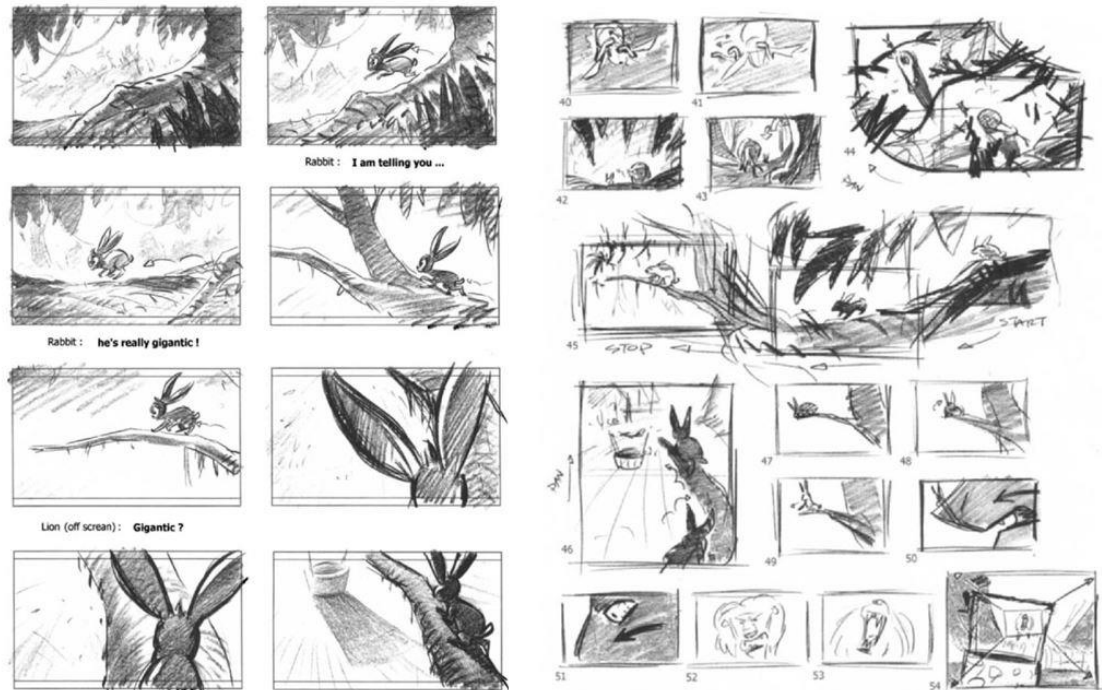


Figure 1. Examples of storyboard thumbnails (Dowlatabadi & Winder 2013.)

In the animation pre-production phase animatics come before layouts. An animatic is an animated storyboard where the drawn images are put on a timeline to animate these story beats, giving the viewer a better understanding of how the scene is supposed to play out. The layout on the other hand brings the animatic to another level by taking the 3D characters and animating them to move, often sliding in their original T-poses to show the duration and movement of the basic action. Creating a layout of each cutscene can clarify how the story progresses and if anything about it needs to be changed. (Adib 2019a; Adib 2019b.)

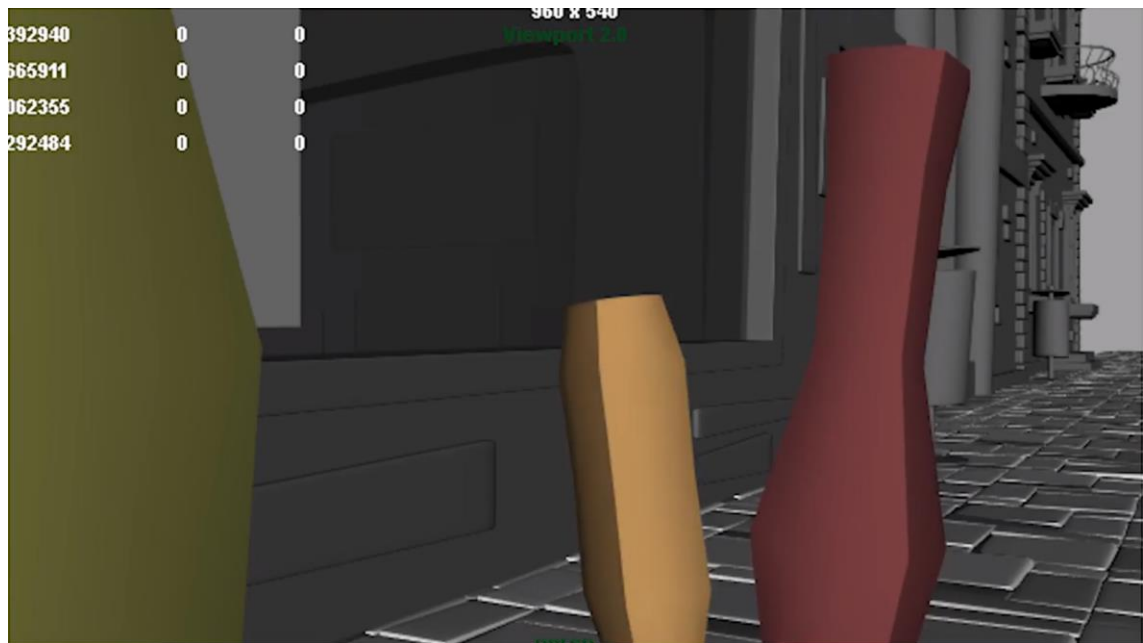


Figure 2. One frame snippet of a 3D Animation Layout video (Dream Farm Animation Studios 2019.)

Scripting, conceptual design and storyboarding often overlap each other, unlike in live-action film where a script is needed before the storyboarding can start. Planning animations is more effective with all three underway simultaneously as they affect each other more closely during game development. (Schnitzer 2013.)

During game development it is crucial to keep the script and plans as open and flexible as possible since during production the situation can change easily. In large projects which use actor performances for the animations, once a cutscene has been built, changes to them can range from expensive to impossible. (Nelson 2019.) Small projects' challenges on the other hand lie mostly on the amount of recourses available and funding. Christoph Schnackenberg and Marcel Hampel explain in Dicky Phillips' interview (Phillips) how the scope of the game is difficult to estimate and how everything needs more time than initially envisioned. Not knowing which kind of features to include early on in the development phase of the game can throw off the game's time schedule and cause integrity issues if not handled carefully.

3.2.2 Production

The production phase of the game's cutscenes is heavily tied into the production of the entire game. Like in any other animation workflow, the production phase is when the

actual work for the cutscenes is done. With the 3D animation workflow, it starts with modeling the characters and environment assets which have been planned in the pre-production phase.

“Modeling is the process of creating a three dimensional representation of any object, from humans and animals to machines and natural environments” (Ovrick 2017). These 3D objects are constructed from a collection of vertices, edges and faces, creating a polygon mesh. A polygon mesh is the surface of an object that defines the shape of every 3D object. In simpler terms, the 3D meshes are formed by polygons, which are usually triangles or rectangles that fit together to build the whole object. (Petty a.) Figure 3 shows an example of a 3D object which is formed by triangle shaped polygons.

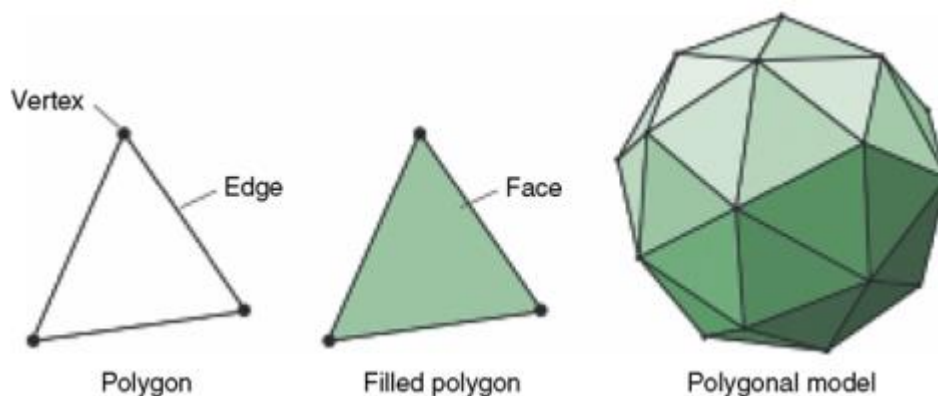


Figure 3. Differences between a vertex, edge, face and a polygon (Khan 2016.)

Ovrick (2017) in his article about 3D modeling states that 3D modeling is usually split into two groups; organic and hard surface modeling. Organic modeling is used, as the name suggests, in organic forms such as characters, plants and other natural objects in the environment. Hard surface modeling comprises of objects such as vehicles, buildings and anything that is constructed.

Digital sculpting is especially useful with organic models where the 3D meshes are created using methods that are more similar to working with traditional clay. The modeler in this situation does not worry about the amount of polygons in the 3D meshes, but after the digital sculpt is done, he/she moves on to a process called retopology. Retopology is used to create an organized polygon wireframe over the dense sculpture. With animated characters this retopology needs to be arranged so that the character’s joints have

certain polygon loops so that the character can bend optimally and more realistically. (Ovrick 2017.)

Before the character models can be animated, they require rigging. Rigging is a “...*process of creating the bone structure of a 3D model. This bone structure is used to manipulate the 3D model like a puppet for animation*” (Petty c). The animation of the characters is a big part of the cutscene creation process during production, since the characters are often the main focus of the cutscenes.

Other important aspects to develop during the production phase of a cutscene are building of the surrounding environments using the 3D assets, creation of character effects, FX and lighting. Character effect artists create everything that is moving on a character including clothing and hair. FX artists, on the other hand, are responsible for all the details such as footprints being left on the ground or the aftermath of an explosion. Lighting the cutscenes establishes the look and mood of each scene. (Beck 2017.)

3.2.3 Post-production

Depending on an animation project, post-production is the part where compositing, color grading as well as music and sound design comes into play. Compositing is where the different elements of an animation are brought together and it creates the final look of the animation product. These elements can be final rendered images of an animated character and other atmospheric elements such as depth of field and color correction. (Beck 2017.)

When it comes to in-game cutscene animations, compositing is not exactly the same as it is in the most usual way of doing 3D animation. Since every frame is calculated in real time inside the game engine, cutting and pasting a character’s rendered images would not work. Compositing in these types of cutscenes is the action of telling Unity to add effects (depth of field and color corrections as an example) on top of the cutscenes by adjusting camera focuses and other effect related settings.

Music and sound design, on the other hand, can be started immediately when the environment locations are done and the layout of the animation has been made. This can already start in the production phase. Some sounds, such as characters interacting with

the environment in the cutscenes, can only be worked on after the final versions of the character animations have been created.

4 Project: *Skábma* – Snowfall cutscenes

Skábma – Snowfall is a story driven first episode of what is planned to become a longer game. It focuses on showcasing the Sámi culture through its world, lore and characters. The story is about a young Sámi child named Áilu, who finds a magical shaman drum which Áilu can use to get past obstacles and hardships on their way. The core gameplay mechanics revolve heavily around Áilu using these magical abilities by drumming the drum. Initially the *Skábma – Snowfall* project started in August 2019.

The following chapters in this thesis go through the creating process of cutscenes in a chronological order. The thesis only briefly covers the parts that are unrelated to character animations. Before any animation could start to be planned, many assets and other pre-production pipelines had to be finished first. These included crafting the main story and script, from which character concepts and models were made. Control rigs for the models with proper skin weights were also necessary before animation. Around the time of these being underway, the team planned and built the map of the game and environment assets and programmed the core gameplay mechanics for *Skábma* to be a playable game.

When we planned the cutscene animations, the main focus was to first finish the base movement system of the main character. These movements included running, walking and jumping animations all the way to Áilu's magic powers with the drum. After the base movements were finished, many of them could be reused in the cutscenes, reducing the amount of unique animations for each cutscene.

4.1 Cutscene creation process

Once the script for each cutscene had been written, the animators and writers went through it together to see which animations were required. As with a small game development team, our game Director and Chief Executive Officer, Sahin Cengiz, was in charge of the final look and pacing of the cutscenes. The animators worked closely with him to create fitting scenarios for the game. While going through the scripts of each cutscene, discussions were held about which animations needed to be created from

scratch, what could be reused and which ones could be made by editing already existing animations.

While starting this project, there were many options on how to go about the cutscenes. One option was to animate a full scene in Maya with all the characters present from one cutscene. This would give the most control over how each animation looked and how the characters interacted with each other. However, this option was discarded as building up scenes in Maya using already existing animations is somewhat slow. Our team and Cengiz came to the conclusion that the process of combining cutscenes in timelines in Unity was faster and easier. For example, Maya's Time Editor (end of chapter 4.2.2) does mostly what Unity's Timeline Editor (chapter 4.4) can achieve, but Time Editor requires many settings to be tweaked and the process was not very intuitive. Using Unity to build the final cutscenes gives possibilities in taking animations from other characters (as all the game characters use the humanoid system) and there is no need to go through retargeting as it would have to be done in Maya.

When it comes to cutscenes, we thought about which unique separate animations we want for the character for each scene. This led to our team skipping the storyboarding phase altogether from the animation process. What we did instead was create first iteration timelines for the cutscenes in Unity. These basically served as layouts, where the characters would crudely do the basic actions from sliding positions to using ready animations. Here the planning of the angles and lengths of the animations as well as getting the general idea of the overall look could be achieved.

The plan was to focus on making the animations work and then build the cutscenes around them, rather than planning strict shot lengths for the animator to fill out perfectly. Strict shot lengths are often done in animated movies where every shot has to be planned beforehand not to waste any resources. Our team's animation focused approach caused the problem of the animation having to look good from all angles. Often the full body needs to be moved instead of, for example, just Áilu's hand in a close up shot. However, due to the very lax nature of the cutscenes being built around the animations, this is not a problem because we can focus on the good parts. Any weird movements or actions can be removed with clever cuts and camera angles that do not show the problem areas.

This thesis will be focusing on the second cutscene of the game named *The Fall* to showcase each step of the cutscene creation process.

4.2 Animation

The animation of the characters is often the main focus of a character driven story and its cutscenes. This is the part where people focus the most, and odd glitches or bad animation can ruin the viewing experience. With the game project *Skábma – Snowfall*, two different methods for making animations were used; animations completely done from scratch and animations using already existing animations and retargeting them to the characters. Both of these methods were also used together. For example, the base for the animation came from a ready-made asset and it was then modified to fit the game's needs.

In the cutscene *The Fall*, Áilu finds a reindeer lost from the rest of the herd. Áilu approaches it and pets it before looking far off to the distance. After a few moments Áilu turns to head back to the village with the reindeer, but a sudden explosion in the distance shakes the ground and startles them both. The shaking does not stop and Áilu and the reindeer decide to make a run for it before suddenly the ground splits open beneath them. They fall down into a dark cave. The necessary animations required for this cutscene to work were the following: Áilu seeing the reindeer, approaching the reindeer and petting the reindeer. In addition, it was important to animate Áilu turning to look into the distance and turning back to head home, Áilu's reaction to the explosion and Áilu backing away from the explosion, running away and falling down to the cave. The reindeer also needed to have animations such as being idle, being petted, reacting to the explosion, running and falling.

From these requirements, we can already see that Áilu's existing running and walking animations can be used to move Áilu around when they approach the reindeer and run away from the explosion. An idle animation is also useful when Áilu looks into the distance. The rest of the animations for Áilu need to be done from scratch.

4.2.1 Animation process

The basic process of animating will be gone through with the animations where Áilu reacts to the explosion and pets the reindeer. The names CINÁiluExplosionReaction and CINÁiluPetsReindeer were used while referring to these animations. The CIN prefix stands for cinematic, which is used to distinguish animations that belong to cutscenes.

The animation process starts with having an idea of the action the character in the cutscene is doing. This can be put down as a description or even as a simple image with lines and stick figures such as in Figure 4 below.

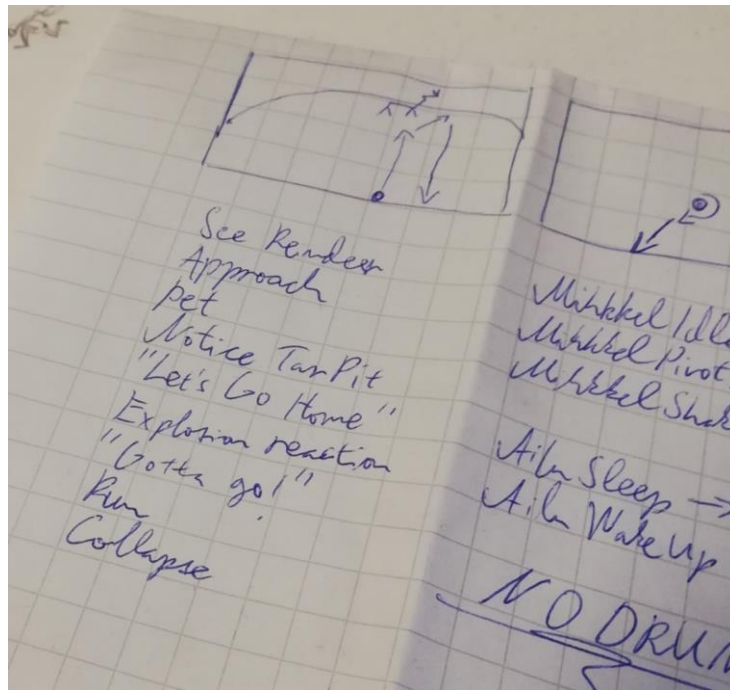


Figure 4. First plans of *The Fall* cutscene.

After a plan is done, testing the movement starts with posing the characters on a Maya scene to their key positions. These are called keyframes, which show the main movements of a character as still “images”. At this point, the animation can be edited very easily, as keyframes can be moved along the timeline to adjust the timing of the animation and the character positions can be made more suitable to showcase the action.

Becker (2017) tells about key, extreme, breakdown and in-between poses in his 12 Principles of animations video series which sums up many important things about animation both in 3D and 2D. He describes key poses as being the main poses that show the start and end of the action. Extreme poses show the farthest positions the animated character will go in each direction. Breakdown poses are for deciding how the key and extreme poses connect to each other and in-betweens are all the rest of the frames filling out the empty spaces. For the sake of simplifying all the animation terms, the key and extreme poses will be referred to as simply key poses since they both showcase the most important movements of the animation.

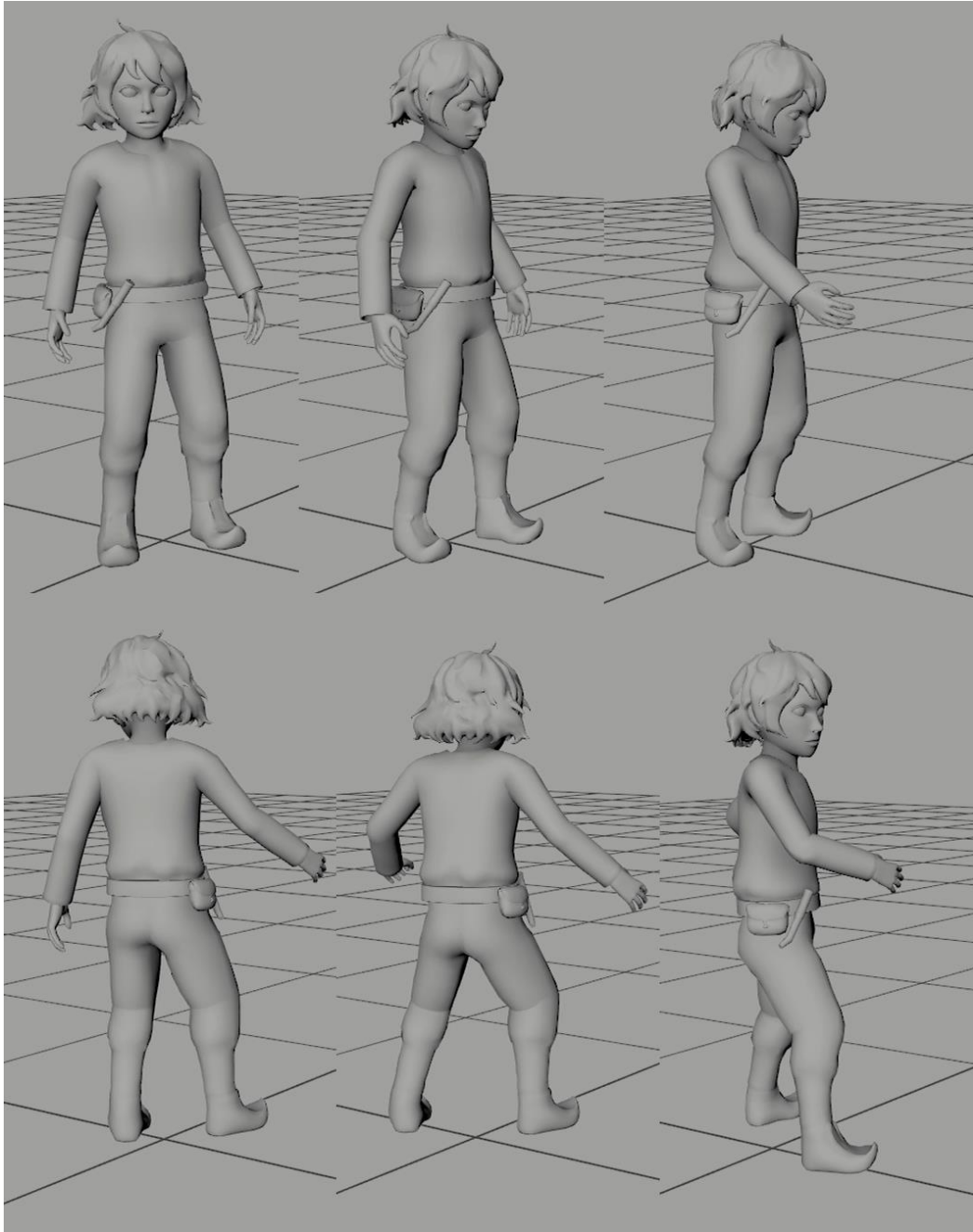


Figure 5. Key poses of the CINÁiluExplosionReaction first iteration animation.

In the blocking phase of an animation, things can look very crude and this works as long as the basic idea of the animation comes across. In the CINÁiluExplosionReaction animation, we are not even thinking about Áilu's finger or wrist movement. Even bigger aspects such as the legs staying planted on the ground are not yet needed. In CINÁiluExplosionReaction, Áilu starts with looking into the distance before turning around and nudging the reindeer to follow. Suddenly, there is a loud explosion behind Áilu who gets startled and quickly turns back to glance at what happened.

Our game director, Cengiz, preferred to get the full range of animation showing even in the blocking phase rather than seeing the character's movement as still images. This was easy to provide by letting Maya blend between each key pose. The first iteration of CINAILuExplosionReaction can be seen in video 1 (cf. appendix 1) of this thesis. After getting feedback on whether the animation is what was first envisioned, it gets approved and the creation of the next iteration can start.

Becker (2017) also goes through other useful animation tips which are essential for finishing up the crude first iteration of the CINAILuExplosionReaction animation. A few of the principles that were useful with this particular animation were follow-through and overlapping action. Becker (2017) tells how this is “*a technique of having body parts and appendages dragged behind the rest of the body and continue to move when the body stops*”. When starting to fill out CINAILuExplosionReaction's key poses with breakdown poses, it is important to give Áilu's upper body, arms and head time to get dragged behind and to keep moving them when the rest of the body has already stopped moving. This brings a great deal of realism to the animation and makes it look smoother. Appendix 1 (cf. link 1) includes a link to both to the first iteration and the second more polished animation iteration of CINAILuExplosionReaction. The video also shows the progress of the CINAILuPetsReindeer animation, including the finished animations for the reindeer.

Follow-through and overlapping action apply also to things that are attached to Áilu. Áilu's hair and loose clothes should move behind the main action. These, however, do not need to be animated since the movement of the clothes will be generated in Unity with cloth physics and the hair movement will be done with Unity's dynamic bones. Chapter 4.5 gives more in depth information about these methods.

4.2.2 Animation reuse: retargeting and editing

The animation process regarding *Skábma – Snowfall* started by reusing animations from the previous pilot game, as well as stock animation packages. Many of these animations were reused by retargeting them to the new characters and then edited to fit them better. We'll be following Áilu's running animation throughout the explanation of how the retargeting system is used.

Animation retargeting is a procedure where the animation data from the joints of the character skeleton are moved to another (target) skeleton. These two skeletons can be

vastly different to each other in their proportions, as long as their skeletal structure abides by the Autodesk HumanIK (Inverse Kinematics) character structure. This structure requires at least the hips, upper legs, legs, feet, spine, arms, forearms, hands and head to be present in the character skeleton. (Autodesk Knowledge Network 2018a.)

Extra joints such as shoulders and neck can also be applied. Despite there being slots for possible character fingers and toes, Maya's HumanIK system does not recognize them when retargeting bone animations to a character rig's control nodes, which is also why in *Skábma's* case they did not need to be put into the character definition. Below is a picture of all the needed joints for HumanIK.

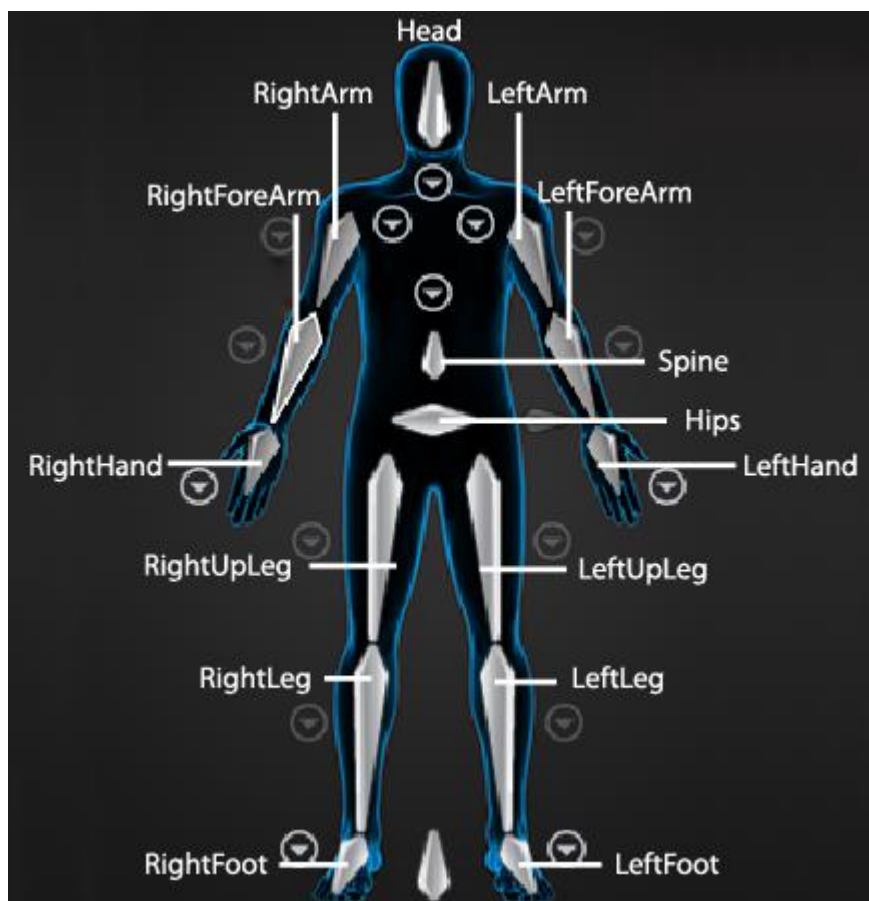


Figure 6. Required joints named for the HumanIK. (Autodesk Knowledge Network 2018a)

The running animation was previously made for Áilu in *Skábma's* pilot game. However, it was animated with a different character model which is no longer in use. To move the running animation from old Áilu to new Áilu, both the skeletons' character definitions need to be created.

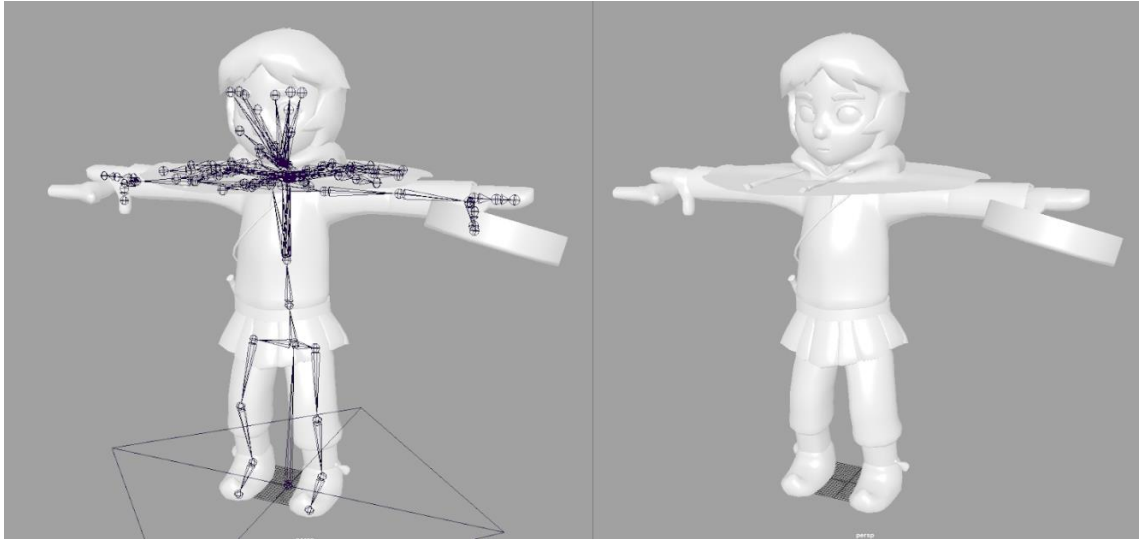


Figure 7. The old Áilu from the previous Skábma pilot game in a T-pose.

All the required procedures regarding retargeting can be found in Maya's HumanIK window shown in Figure 8 on the left. "Create Character Definition" opens the main definition window. Maya needs information on which joint corresponds to its HumanIK character structure. Maya is given this information by assigning the character skeleton's joints to each of the corresponding empty greyed out joint slots in Maya's Definition window seen in the middle of Figure 8. This needs to be done for every joint that was listed in Figure 6. (Autodesk Knowledge Network 2016c.)

After each joint has been transferred, all the joints in the definition window should be glowing green or yellow as seen on the right in Figure 8. The yellow in the case of Áilu's rig indicates that the character's arms are not modeled completely straight. This is not an issue as they are still very close to being such.

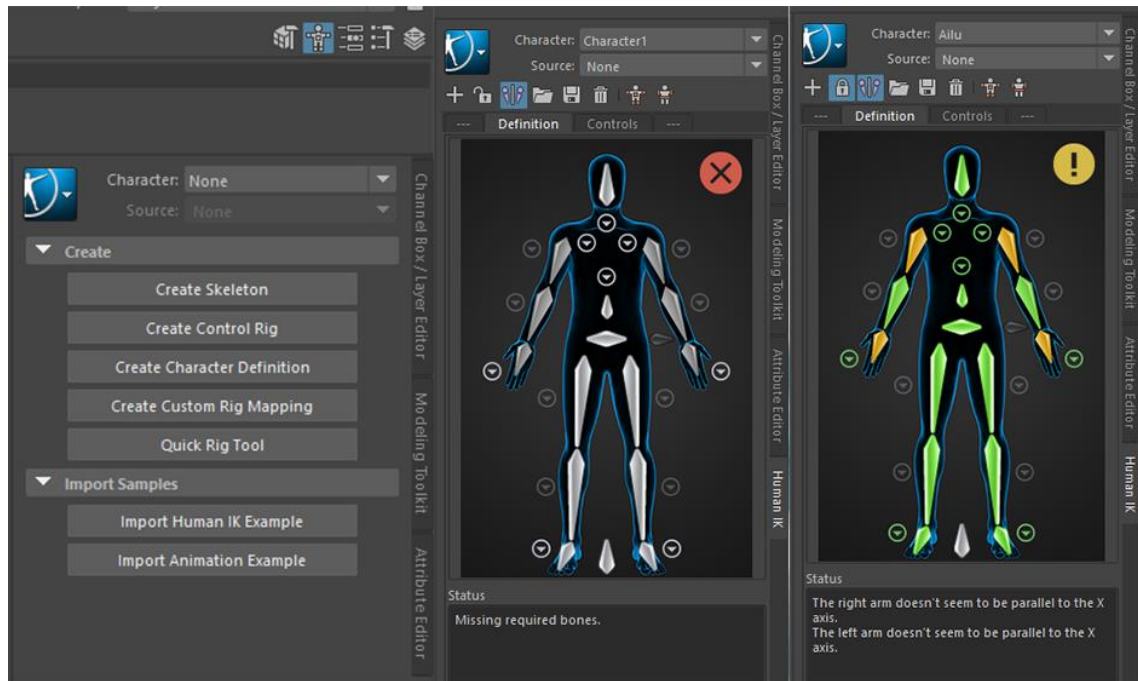


Figure 8. Pictures of the HumanIK window with various stages.

When the process of creating character definitions for both skeletons is finished, it is time to create a custom rig mapping for the target skeleton's control rig. Instead of the retargeted animation being baked to the bones, it will be baked to the control rig which can then be normally animated using its animation controllers.

The procedure is very similar to creating a character definition. This time, however, the correct control rig controllers (effectors) are being assigned to the slots in the human IK window. Depending on what type of control rig is being used, not all of these slots necessarily need to be filled. In Áilu's case, choosing Áilu to have FK arms and IK legs worked the best. The FK arms need all their slots filled, as seen in Figure 9, while the IK leg controls only need the ankle slots. (Autodesk Knowledge Network 2019d.)

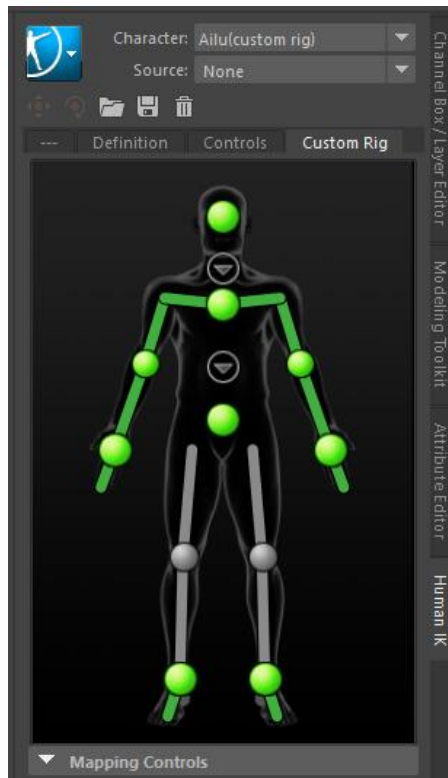


Figure 9. Custom rig window in Maya.

Problems may arise with Áilu's knee aims (i.e. controls that make Áilu's knees always aim forward) which are not regarded in any way but these can be separately animated or parented to the leg controls after the retargeting is done. Custom rig mapping does not include finger or toe controls and the number of spine and neck slots is also limited which is unfortunate as some of our ready animations had animated toe and finger curls.

To make the new Áilu run similarly to the old Áilu, both characters need to be in a T-pose. Then it is only a matter of putting the "Character" tab on the HumanIK window to receive the new Áilu and the "Source" to have the old Áilu. Setting these two correctly instantly makes the target skeleton inherit the movement from the source skeleton. Nothing, however, is keyframed or baked into place yet. The animation's parameters can still be tweaked and it will update in real-time to the target skeleton. Figure 10 already shows how the new Áilu inherits the running animation from the old Áilu skeleton.

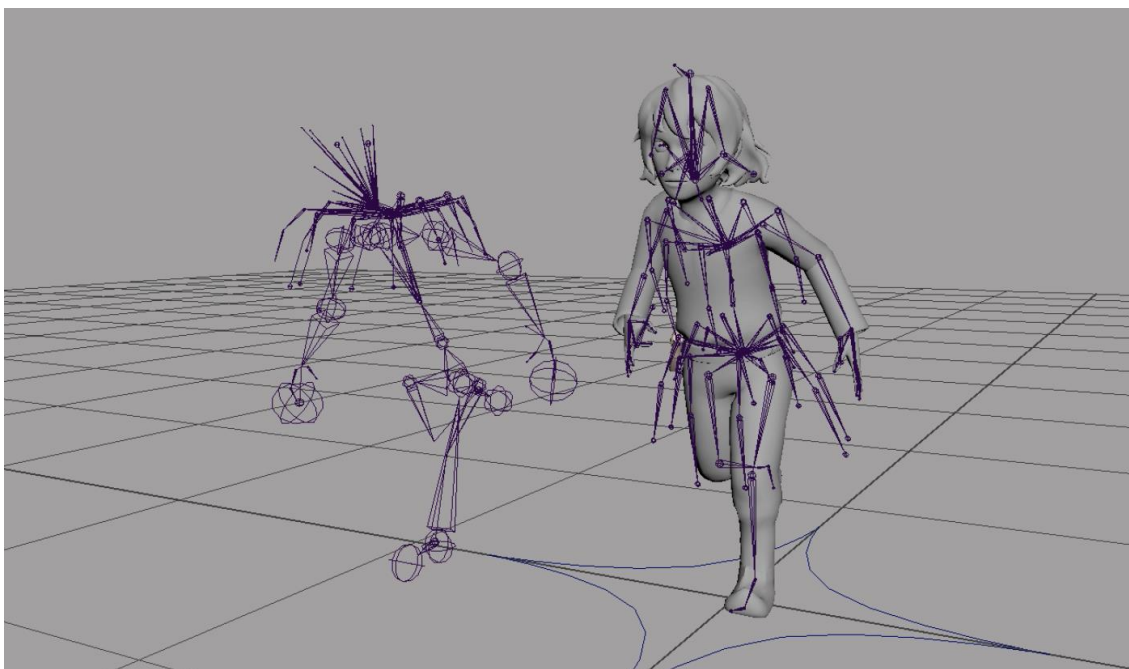


Figure 10. Picture of the old Áilu skeleton's running animation baked into the new Áilu.

Sometimes the retargeting does not come out entirely the way it is intended. The hips or the feet of the character may be placed in incorrect positions, causing the character's limbs to stretch out too far or be too close to each other. Many of these problems can be fixed with the HumanIK system's own editing tools. These tools can be found from the "Edit HIK Properties" section. A couple of quite important settings that can help with problems can be seen on the Retarget Specific tab. Here especially Hips Level Mode, Feet Spacing and the Ankle Height Comp mode can be adjusted to match the original source better. (Autodesk Knowledge Network 2017e.)

Creating character definitions, for each animation we want to retarget, is slow. It is also time consuming to go through all the same retargeting procedures listed above for every single animation individually. This is where Maya's Time Editor can streamline the process.

Multiple animations of the same character can be dropped into Time Editor and edited. (Autodesk Knowledge Network 2018b.) With retargeting, it is easy to drop a big amount of animations into one clip, then do the target skeleton's and the source skeleton's character definitions only once. All of these animations can then be retargeted and later separated into their own files when the need arises. The separation of long animation files

into smaller clips can also be done from Unity's side as suggested in chapter 4.3. Figure 11 below is a picture of the Time Editor window.

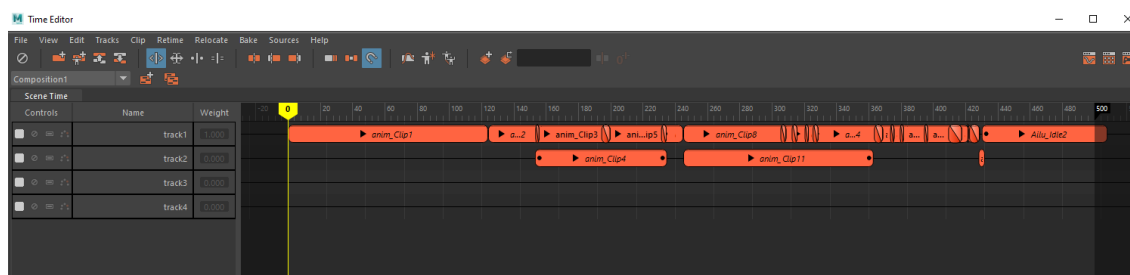


Figure 11. *Time Editor* window with multiple animation clips edited to form one big animation.

Character rigs with the exact same naming conventions for the joints can even bypass retargeting. Due to Maya recognizing the same names and applying the animation to the desired character is possible simply by dragging the source character with animation on top of the target character. This was useful when our team wanted to move the same animations between characters with only small edits without having to retarget everything. Unity accepts animations from different characters through the humanoid system, but the animations do not always look perfect as the proportions between characters can be vastly different. Bringing the same animations to Maya quickly and editing them to fit the target character is an easy way to make a unique and fitting animation for a specific character. Problems may arise from IK controls as Maya does not convert their values in relation to a different sized character. This means a smaller character's legs may take wider steps if the animation was first taken from a bigger character with a longer stride.

After the retargeting of Áilu's running animation is done, it can be edited. Áilu's old running animation does not look fully natural on the new Áilu despite it having been edited through HIKproperties. This is partly due to the new Áilu having proportions closer to a real human in comparison to the old one. Animations that looked passable with a very big-headed and cartoon-like old Áilu do not work well anymore. This means the running animation needs to be adjusted by editing keyframes and animation curves. Appendix 1 (cf. link 2), shows the progress of the edits made to the running animation.

4.2.3 Exporting

When an animation has been finished, it needs to be turned into the FBX file format which Unity can then read and process correctly. It is important to include only the necessary information in this file as anything extra creates bigger file sizes, thus possibly slowing down the game's performance. For animation exports, only the character's joints need to be exported as they hold the rotation and movement information of the character mesh.

The exporting starts with selecting all the character joints. Due to the nature of Áilu's rig's naming convention, this includes all the joints which have the prefix JNT. Only the base skeleton is needed as IK and FK bones do not give the character's movement any additional information; thus, they can be excluded from the selection.

After the correct joints have been selected, Maya's "export selection" settings should be opened. Here the location to save the export file can be set as well as all the other needed information for what should be exported. Red Stage Entertainment's guideline for exporting includes Maya's standard scene scale being set to meters instead of its normal centimeters before exporting. Unity recognizes meters as its standard unit and to have Maya's exports to look exactly the same in Unity, this setting is important to change to avoid any possible scaling issues in Unity. In the exporting settings, the "animation" tab needs to have the correct frame range set. "Bake animation" is also good to have enabled to ensure the animation's movement will stay the same when later viewing it in Unity. Baking animations sets keyframes for each individual frame of the animation.

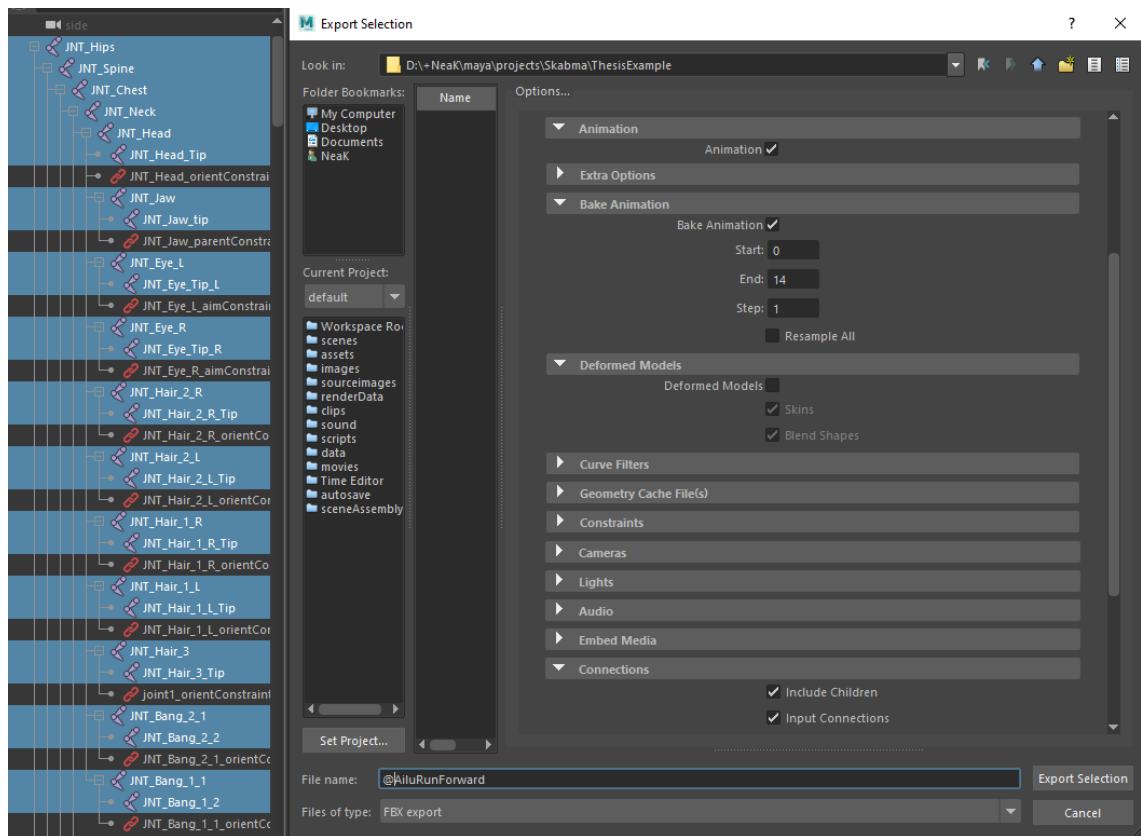


Figure 12. Maya Export Settings window for AiluRunForward animation. On the left, only Áilu's joints have been selected.

Crucial for a successful export is to check the exported file by dropping it back into Maya to see any possible issues which can emerge during the export. In the case of the *Skábma* project, the new FBX file shows that the export has taken the whole control rig with it despite us only selecting the joints for the initial export. These controls need to be removed and the joints exported again. This process can technically be avoided by switching off the tabs for "Include Children" and "Input Connection" in the export settings. The reason Red Stage Entertainment's standard pipeline has not adopted this method is due to unknown issues when bringing the exported animation file to Unity. These issues include problems with the configuration not matching up to the exported bones, creating incorrect animation movements and the mesh acting unpredictably as seen in the figure below.

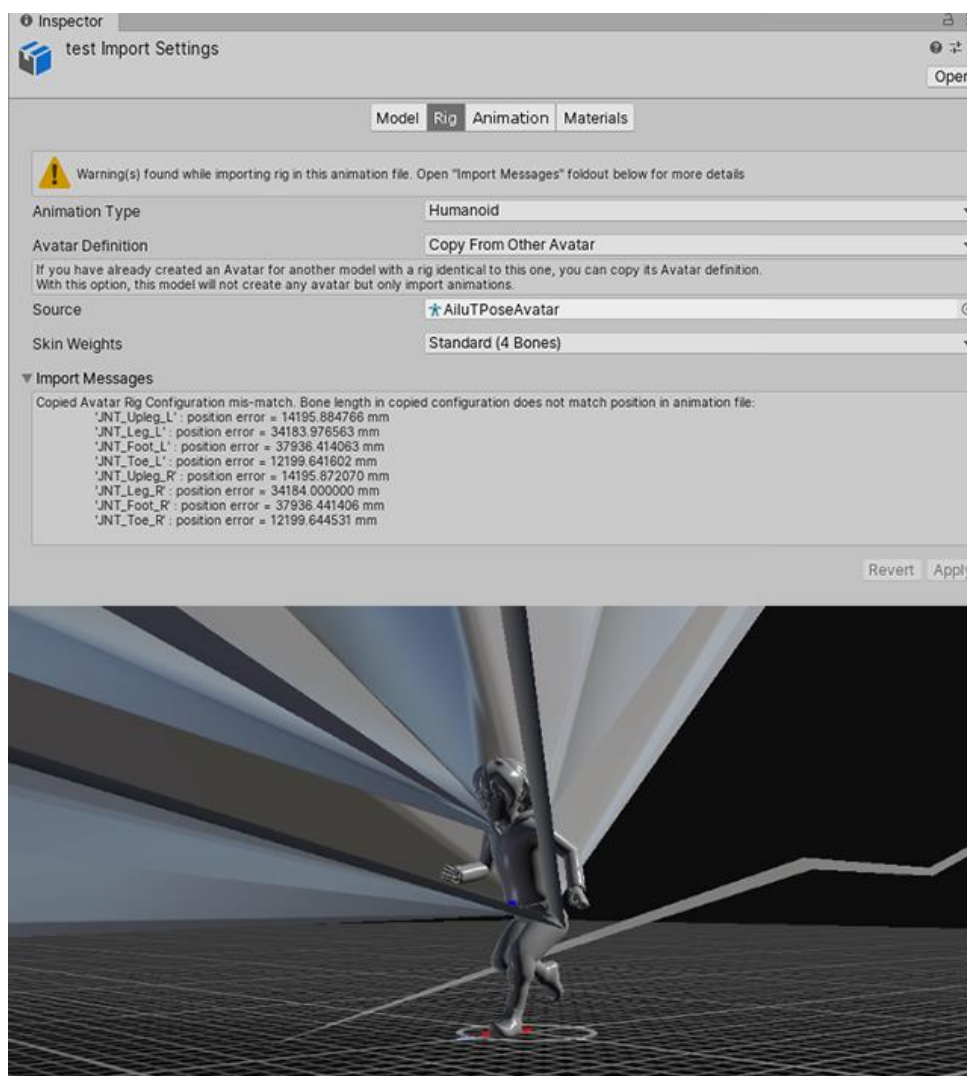


Figure 13. Picture of what the wrongly exported FBX causes.

Another noteworthy issue that occurred during *Skábma's* animation exports was that some of the joints got translation changes keyed into them because of issues with the character rig. In particular, Áilu's spine joint caused most of the troubles and this showed up as a warning when Unity read the FBX file. This was easily corrected by fixing the rig to only inherit rotation values. It is very important to have a working rig when animating and exporting. Tests to ensure everything works correctly in Unity are highly encouraged to further avoid small mistakes like this.

Before the exported FBX can be moved and made to work in Unity, one more export needs to be done. This export, named AiluTPose, is the default export of the animated character, Áilu, and it includes the mesh and skeleton, but no animation. With it, a base for the character Avatar can be made in Unity, and every single future animation will be

referencing it. This way the character mesh is only used once in the Unity game project, saving the game from unnecessarily big files as the animation files only need to have the joints to tell how this base model mesh should move.

4.3 Unity implementation

Next the AiluTPose FBX file needs to be set correctly in Unity. All the other FBX files, which have only the bones and animation data in them, will reference this AiluTPose file in which Áilu's skin weights and mesh have been set correctly. This reduces the amount of clutter and unnecessary duplicates of the Áilu mesh in the Unity project. For the referencing to work correctly, an Avatar for Áilu has to be made.

Creating an Avatar for AiluTPose is as follows: After dropping the AiluTPose FBX into Unity, the import settings should be set up as Figure 14 shows.

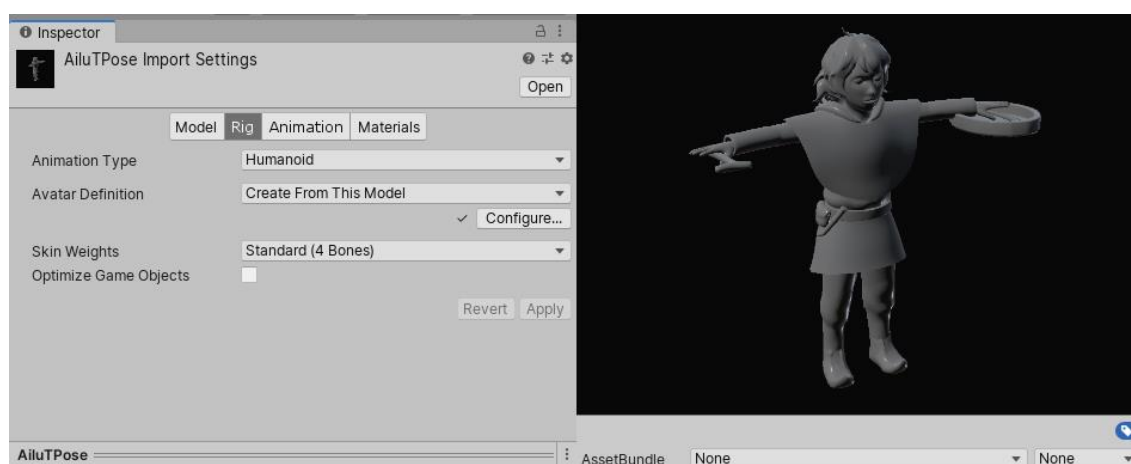


Figure 14. AiluTPose import settings in Unity

The Humanoid system makes sure animations can be used between other characters who also have the humanoid system's bone structure. This is one of the biggest reasons our team decided to go with Unity's humanoid system for our animations. Opening up the small button with the name "Configuration" enables seeing the humanoid character's expected slots for each bone and the bones assigned to them from the Áilu character as seen in Figure 15.

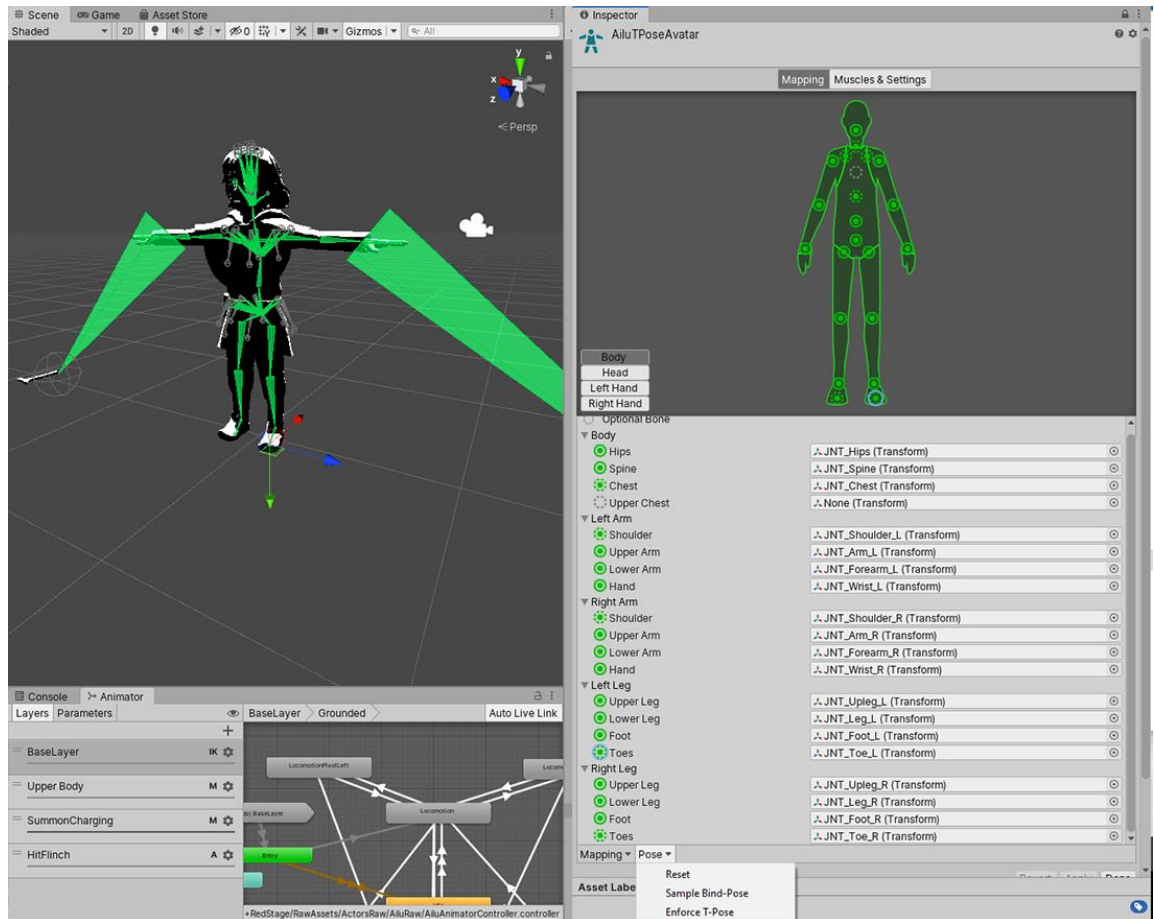


Figure 15. Áilu configuration window in Unity.

Depending on the rig of the character, the amount of bones can slightly vary. For example, Áilu Rig does not have an extra bone for the slot Upper Chest. Other slots marked with a dotted line are bones which are also not necessarily needed for the Unity humanoid system to work. While viewing the configuration of the rig, it is important to note that Unity has assigned each bone to its correct counterpart. If not, the correct bones need to be assigned manually. In Áilu's case, Unity did not recognize Áilu's chest bone which had to be assigned manually.

The Unity manual (Unity Manual 2019a) says that the T-pose is the default pose required by Unity animation and it is the recommended pose to model in a 3D modeling application. If the model of the character and the animation do not work as expected, the Enforce T-Pose option seen at the bottom of Figure 15 can be useful if the character in question has not been perfectly modeled in a T-Pose.

Now that Áilu Avatar is in place, AiluRunForward animation can be brought into Unity. The FBX file can be dropped into the game project's animation folder. In the project, this folder is named AiluAnimations to specifically differentiate Áilu's animations from other characters' animations. Just like bringing AiluTPose into Unity, the AiluRunForward animation type needs to be set to Humanoid from the AiluRunForward Import Settings. This time, however, the Avatar Definition is set to "Copy from other Avatar" which will be set to the said AiluTPoseAvatar. Other import settings such as removing Unity's Material Creation Mode from the "materials" tab can be changed since it is unnecessary.

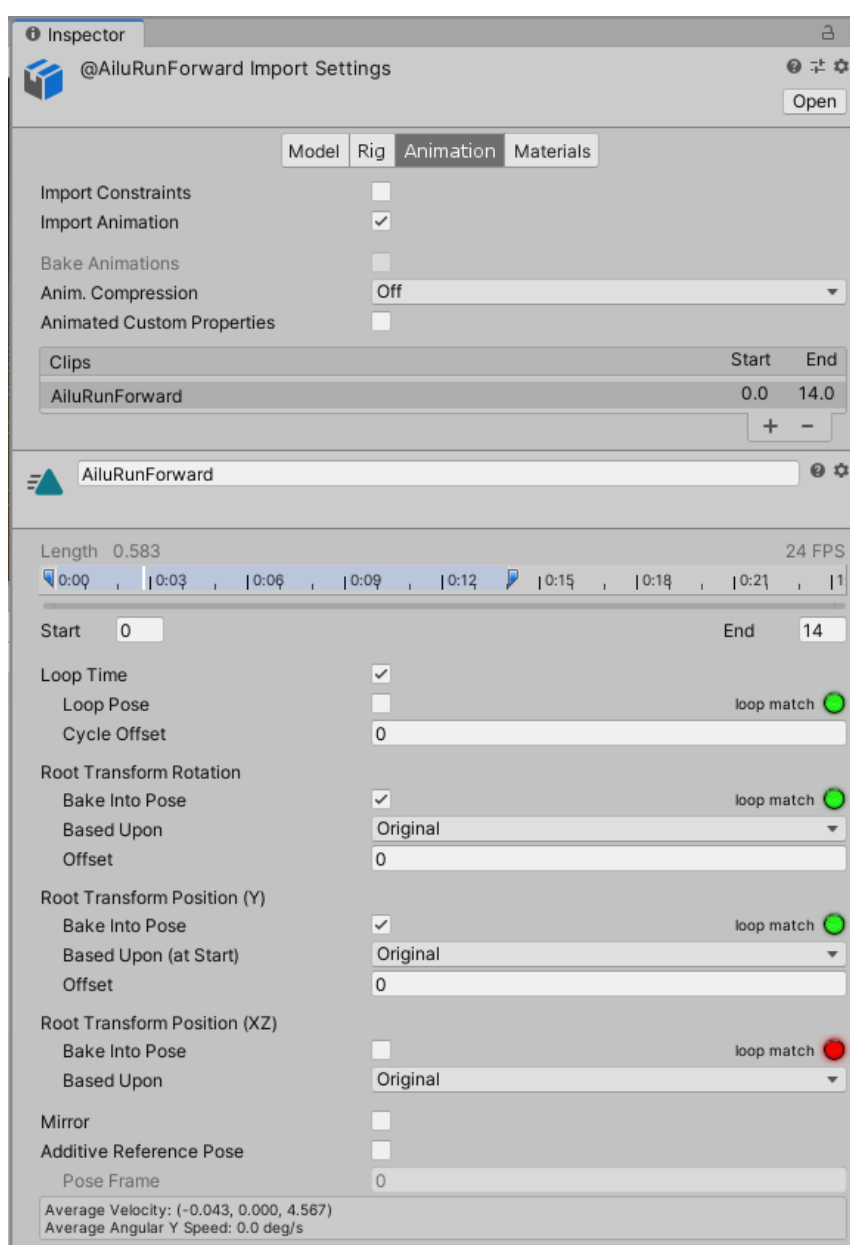


Figure 16. Image of the AiluRunForward animation Import Settings.

At this point, Áilu's running animation is almost fully implemented into Unity. What it still needs is a couple of more setting tweaks as seen in Figure 16. The first crucial part is to name each animation clip accordingly. Unity sets clip names on default to "Take 001". By naming the running animation FBX file as @AiluRunForward, Unity knows to immediately name the clip as AiluRunForward. If the AiluRunForward animation was longer and had other kinds of running such as jogging and sprinting cycles in it, it could be split into multiple clips. Clicking on the plus icon in the clips section in Figure 16, new clips can be created, renamed and new frame ranges set to start and end at different times. However, because the AiluRunForward FBX file only has one animation in it, the frame range is set to start at frame 0 and end at frame 14, just like they were created in Maya.

Since AiluRunForwards is supposed to be a looping animation, the logical conclusion is to also check the "Loop Time" tab. Now Áilu's animation does not stop at its last frame when used in the game. The other baking options in Figure 16 determine how Áilu's Root node behaves. To explain what each baking option does briefly it is important to know about the two types of animations Unity can take. These are animations with and without root motion. Animations without root motion stay at the origin and code is used to move the animation around inside a game. Animation with root motion, on the other hand, has the movement information built into itself.

AiluRunForward is an animation with root motion. "Root Transform Rotation" seen in Figure 16, when baked, keeps Áilu's root rotation exactly the same as it is in the original FBX file, making sure Áilu keeps running forward. Without it being baked, Áilu starts to turn slightly, the longer the animation loops. "Root Transform Position (Y)", when baked, keeps Áilu's root node at 0, not giving the animation any height. "Root Transform Position (XZ)" locks Áilu's animation root node in the origin again, letting the character move away from the root. (Unity Manual 2019b.) This option is usually good to bake when multiple characters interact with each other and they all need to have the same origin point so that the characters do not clip through each other. More about this particular case can be found at the end of chapter 4.4.

4.4 Unity timelines

"A timeline is a literal GameObject with a timeline component, which is editable in the Unity Timeline window that controls the animation keyframes and object life cycles" (Uc-

cello 2018). In *Skábma – Snowfall* the cutscenes were being put together in Unity's timeline tool. As previously mentioned, our team also used this tool to create rough layouts of our cutscenes with the animations and the assets we had ready at the time. Due to this, our first layout was quite detailed as all of Áilu's first iteration animations for *The Fall* cutscene were ready to be used.

Setting up a new timeline starts with creating an empty game object in the wanted Unity scene. Opening the timeline tab shows the button "create timeline". This opens up the timeline window where the "actors" of each cutscene can be dropped into. When bringing the characters in, Unity creates unique tracks for each one. Every animation clip that is added to a character's track will only move that specific character. (Uccello 2018.) In *The Fall* cutscene, the actors are Áilu and the reindeer. Since the *Skábma* project uses the humanoid system for all the human characters, dropping another humanoid system's animation clip into Áilu's track will also make Áilu move. The animation transfer is not always perfect due to the different proportions the characters may have.

Working on cutscenes in Unity's timeline window is very similar when working with any other video editing program. Character animation clips can be dropped into the timeline, and they can be blended with each other, cut to pieces, duplicated, slowed down, sped up and made to loop among many other options. The characters' animation clips can also be easily moved around into different positions in the scene by changing their translation values.

The reason for our team preferring Unity's timeline tool for building the cutscenes instead of using Maya's Time Editor tool mostly came from Unity's timeline handling animation clip matching very easily. As an example, in the beginning of *The Fall* Áilu walks towards the reindeer. The animation can be made to loop to keep Áilu walking forward instead of stopping after the unlooped animation is over. However, when we want to blend the walking animation into an idle animation as Áilu stops in front of the reindeer, Unity suddenly moves Áilu back to the position where the character was at the start of the walking animation. By using match offset on the idle clip, Unity automatically checks Áilu's latest position from the walking loop and moves the idle animation to start right where the walk ended. (Unity Manual c.)

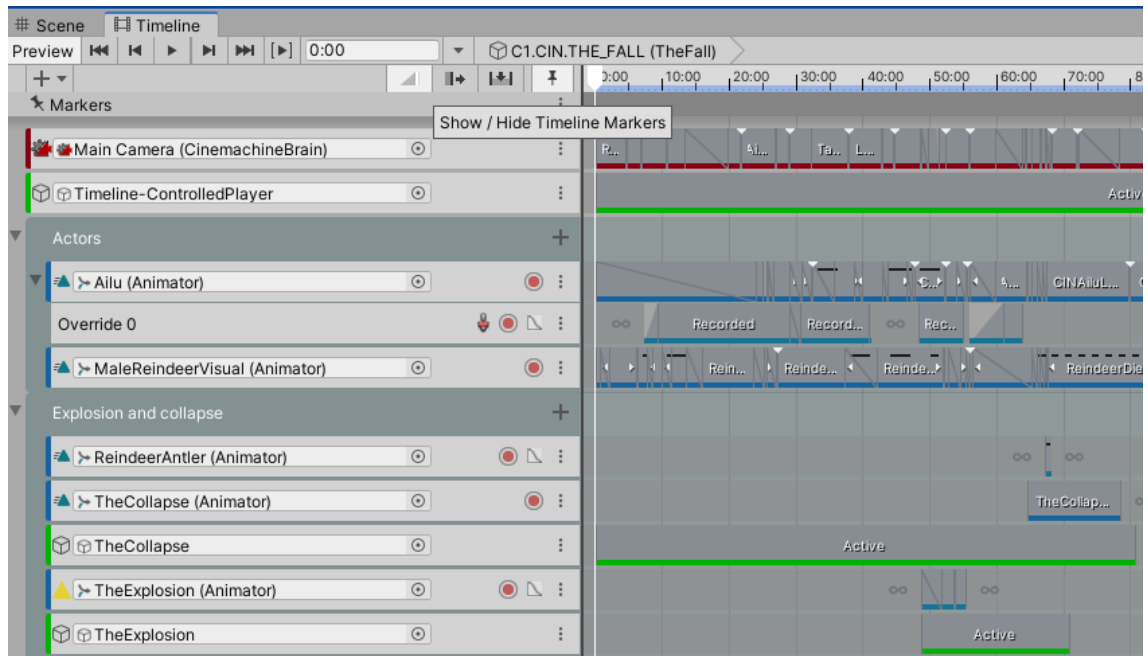


Figure 17. Image of The Fall cutscene's first iteration timeline

Other things such as cameras and lights can also be animated through the timeline. Figure 17 shows *The Fall* cutscene's first iteration timeline where the camera and other rocks and rubble elements from the explosion are shown being animated. A link to the first iteration of the cutscene can be seen in appendix 1 (cf. link 6). This is the version where only crude animations and unfinished assets were used to quickly build up an idea of the scene.

When Áilu pets the reindeer in *The Fall* cutscene, the movements of these two characters need to be matched without one of them clipping through the other as Unity does not automatically know where each character is supposed to be situated. This kind of problem can be avoided by going to the CINailuPetsReindeer and CINReindeerFemaleBeingPet animation settings and baking each animation to their original position. This sets the zero point of the animation to where the origin of the animation in the original Maya file is. As seen in the figure below, this offsets the reindeer slightly from the center.

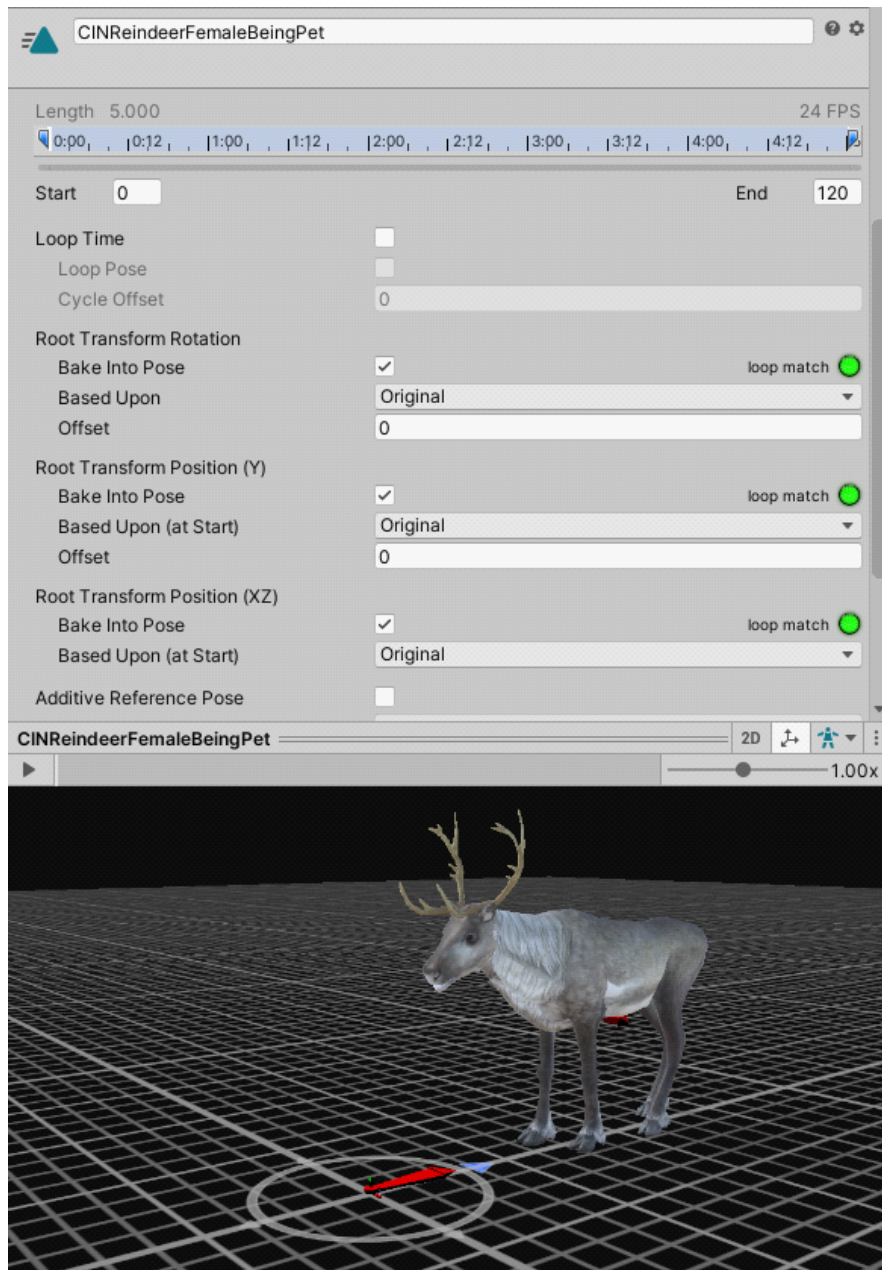


Figure 18. Image of the reindeer's animation settings.

With Áilu and the reindeer being offset, their zero points are now in the exact same position in relation to each other. Now when the animations are dropped into Unity's timeline, they will show correctly when both have the same translation values.

4.4.1 Animation masks and target aim

During most of the gameplay in *Skábmá*, Áilu holds the drum and the drumming stick in their hands. However, at the beginning of the game, Áilu is not supposed to have anything in their grip. We needed to alter the character's animations so Áilu is not holding

the drum. As an example, in *The Fall* cutscene we do not want Áilu to walk closer to the reindeer with their hands around a nonexistent drum and drumstick. This can be seen in a snippet of the first iteration of *The Fall* cutscene in Figure 19.



Figure 19. Áilu's hands in a grip around the invisible drum and drumstick. Snippet from the first iteration of *The Fall* cutscene (cf. appendix 1).

Our team had a couple of options on how to combat this issue. The most straightforward option was to have two versions of each needed animation. For example, in *The Fall* cutscene Áilu walks towards the reindeer. We would then need one version of the walk where Áilu has the drum for normal gameplay, and another where Áilu does not have the drum for the cutscene. This approach is reliable to get the exact animation we want but takes time especially if there are several animations that need duplicates.

Another way to handle this is to use animation masks in Unity. What masking does is “it allows you to discard some of the animation data within a clip, allowing the clip to animate only parts of the object or character rather than the entire thing” (Unity User Manual 2017). In Áilu's case, we can create a mask that overrides the hand grip on the drum and opens the hands. The mask can also take into consideration more than just the fingers and hand. When Áilu is holding the drum, their arms are also further away from their

body to compensate the big drum not clipping through Áilu's body. With masks, this animation can be taken away, but it can also be replaced with another character's walk animation where the arms are in the correct place thanks to Unity's humanoid system recognizing the same bones.

Right clicking inside a project folder, selecting create > avatar mask, creates an Avatar Mask. Unity offers the humanoid system's Avatar Mask which is easy to edit to have only the arms and hands selected as seen in the figure below.

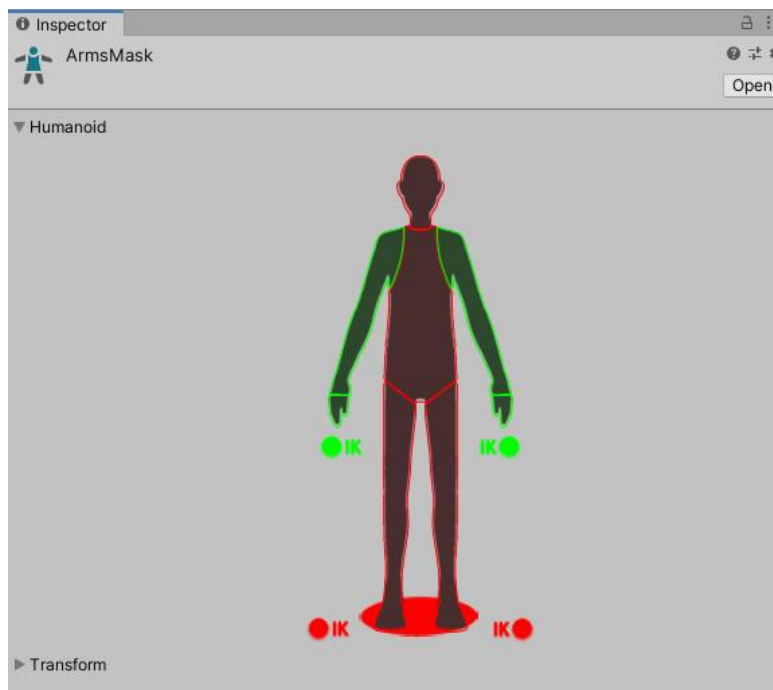


Figure 20. Unity's humanoid Avatar Mask named ArmsMask.

This mask, named ArmsMask, can now be used as an override track in the timeline. Creating an override track happens through right clicking over the Áilu character track and selecting it from the opened pop up menu. The new override track has a setting option where the newly created ArmsMask can now be dropped into. The only thing missing is to select an animation where Áilu's (or another humanoid character's) arms and hands are in a relaxed pose, which can be placed on the override track.



Figure 21. Override track in Unity using ArmsMask.

As seen in Figure 21, when viewing a section where the override track is not active (the line pointed by the red arrow is not over the SuonjarIdle track), Áilu's hands stay the way they are in the initial animation. When the override is active (line is over SuonjarIdle track) Áilu's hands open. In this particular case, the animations where Áilu has open hands came from another character's, SuonjarIdle's, idle animation.

A problem our method of working with cutscenes created was how our characters never looked at each other while interacting during a cutscene. Since we made each individually needed animation for each cutscene separately from the Unity scene, it was impossible to predict in what direction our characters would need to face in the actual cutscene. Animating characters to be looking into a certain direction would have been pointless since we wanted to reuse as many animations as possible. If the character animations had had set directions which they would be facing, our team would have had to make many unique animations for every single situation our cutscenes required. For this problem we had a target aim system at our disposal.

This target system, created by the *Skábma* team's programmers, adds rotations to Áilu's upper body, head and eyes. It always makes Áilu face an invisible target object without making their full body rotate.

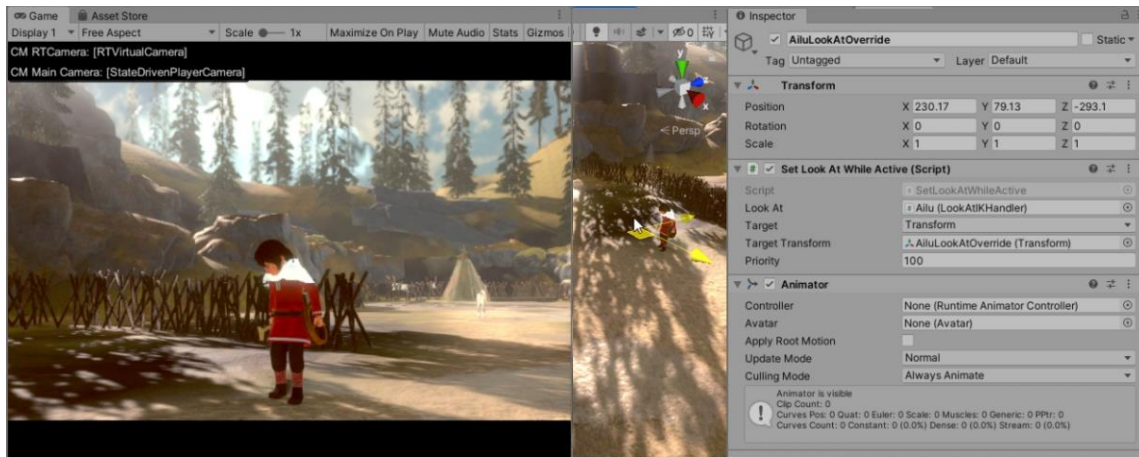


Figure 22. View in Unity where Áilu can be seen looking down at the invisible target object shown as two yellow arrows in the middle image when moved.

This is useful for tweaking each cutscene in which characters have to turn to look at each other or other targets. It saves animators a considerable amount of work. This target system is also very easy to use since it relies on the same timeline override track just like animation masks. A moving example of the aim system can be seen in appendix 1 (cf. link 5).

4.4.2 Blend shape animation

With cutscene animations, everything done in Maya focused on full body animations, and facial animation was done with blend shapes. Blend shapes are copies of a character's face which have been edited. These edits are key expressions such as extreme sadness, happiness and anger.

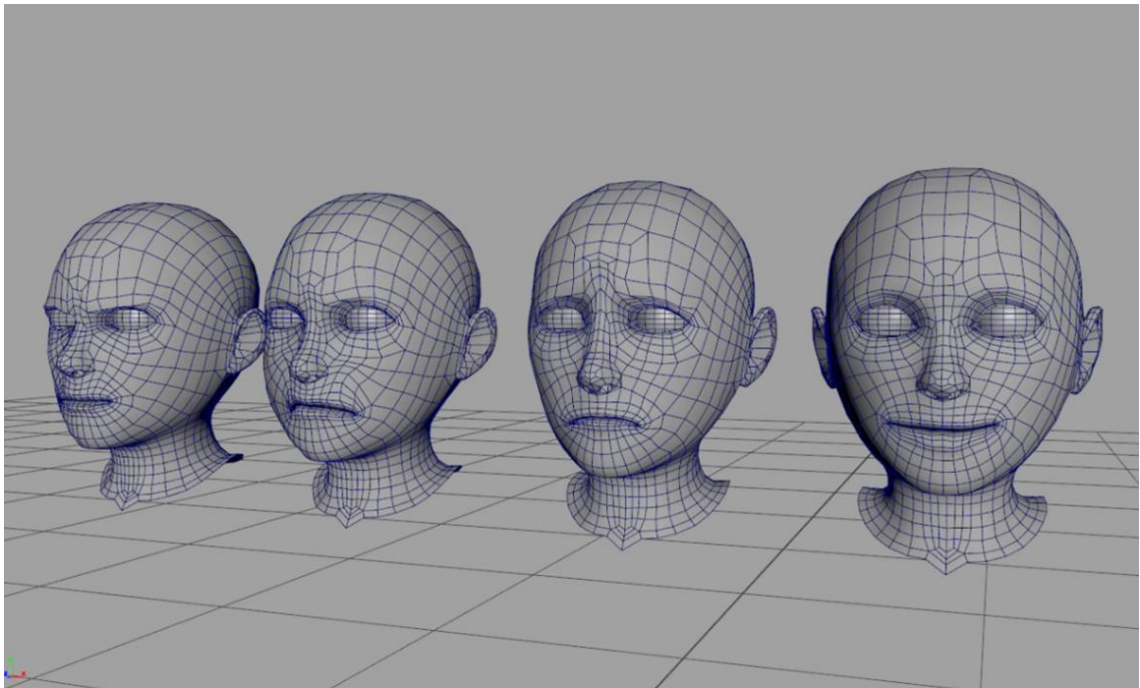


Figure 23. Various blend shapes of Áilu presenting anger, disgust, sadness and happiness from left to right.

Basic lip-syncing can also be animated with blend shapes as well as closing the character's eyes. Due to our small game development team with a very limited time frame, we opted to do all our facial animations with blend shapes. Our team came to the conclusion that setting up a full facial rig for each character and then animating them separately was too time consuming. This certainly would have made quality animations but would have also been costly for our budget. With blend shapes, the expressions can be blended together much more quickly with sliders that determine how many of the moved vertices affect the character mesh.

The facial character animations can even be reused for other characters. Our team's programmers made sure animating one character's blend shape sliders (as seen in Figure 24) can be referenced by another character's blend shapes with similarly named expressions.



Figure 24. Picture of the Unity blend shape setup and sliders.

In the figure above, we can see how it is possible to mix different facial expressions with each other creating a dozen more of unique expressions. The same can be seen in real time in video 3 (cf. appendix 1). The animation of each slider can be animated through Unity's timeline.

4.5 Unity dynamic bone and cloth physics

To liven up the final look of the animations, our team decided to go with Unity's Dynamic Bone plugin. The Indie game developer LeeT (LeeT Game Development 2019) describes in his video about Dynamic Bones how the Dynamic Bone plugin applies physics to a character's joints and bones, making parts such as hair, cloth and breasts move more realistically. This is the reason our team added extra bones to our game character's hair and clothes, to bring in extra movement without having to animate it by hand. A demonstration of how the Dynamic Bone works can be seen in video 4 (cf. appendix 1).

Our original goal was to have Áilu's luhkka and skirt move with the Dynamic Bone's physics. Luhkka is the poncho-like loose fabric Áilu has over their shoulders. However, this did not work as planned as the dynamic bones of the skirt and luhkka did not collide

correctly with the rest of Áilu's mesh which caused a great amount of clipping. Our problem was that we set up too few bones for the colliding clothes and this meant that the bones could slip through, for example, Áilu's legs, making the skirt mesh clip through the leg mesh. The Dynamic Bones on Áilu's hair were more successful as they can get away with slight clipping.

As the Dynamic Bone did not work well with the cloth in our case, we opted to switch to actual Unity cloth physics. Cloth physics treat the mesh of the luhkka and skirt as something that should collide with the rest of Áilu's body while with Dynamic Bones, only the joints collided. This can be viewed in video 4 (cf. appendix 1). This method is definitely more work-intensive than simple Dynamic Bones as the position of each vertex needs to be calculated. The fewer polygons the cloth objects have, the easier the process is to calculate, which is why our team also made sure to make Áilu's luhkka and skirt as low poly as possible.

5 Conclusion

The appendix of this thesis includes links to videos made in the project. The last two videos (cf. links 6 and 7) showcase the first and second iteration of *The Fall* cutscene. The second iteration has been improved with the various additional features that have been covered throughout this thesis, including the usage of masks, blend shapes, cloth physics and Dynamic Bones, a target aim and the latest versions of the character animations. Throughout this project, the surrounding environment has also been worked on. A new reindeer model was implemented and new sound and FX effects were added.

Despite all these new features being included in the newest version of *The Fall* cutscene, it is not yet finished. To finish the cutscene various glitches with the cloth physics need to be fixed. Animation-wise the facial animations of the character still need refining and the reindeer requires more animations for turning around. These fixes will be worked on for the final version of the game *Skábma – Snowfall*, but for this thesis the second iteration of *The Fall* serves the purpose of showcasing the progress of the cutscene.

What we can already deduce from the current state of the *The Fall* cutscene is that creating cutscene animations, such as creating any animations, is a long task that takes careful planning and testing to go smoothly. With the help of various tools, this task can

be made easier and faster. At no point during the case study discussed in this thesis did we face issues that we would not have been able to overcome.

From a visual standpoint, an in-game cutscene is good when it creates the right feeling for the viewer without breaking the immersion. A bad in-game cutscene confuses the viewer and distracts them. To achieve this level of immersion a cutscene needs to be lively and visually appealing to the human eye. For example, without small movements of the hair and fabrics, animations can look very artificial and awkward which distracts the viewer. As video 6 (cf. appendix 1) shows, the first iteration of *The Fall* animation would not work as it is, as the unfinished stages of the character animations make the viewer laugh at the awkwardness rather than follow the story. Big mistakes such as characters clipping through objects also jar the viewing experience even further. For *The Fall* cutscene, the FX effects such as the explosion in the distance, smoke rising and the ground splitting under Áilu are good examples of what sets the appropriate mood for a cutscene. Sound-wise all kinds of ambience noise, footstep sounds, reindeer noises and an explosion boom are just some of the many audio components that play a big role in bringing the cutscene to life and making the character performances believable to the audience.

From a technical standpoint, a good in-game cutscene works visually but has also been made efficiently by cutting corners where it has been possible. Using old animations from the previous *Skábma* pilot game and other available animation packages saves the animators' time, letting them to focus on animations that need to be made from scratch. I would argue that the *Skábma – Snowfall's* pipeline to focus only on creating individual animations, which then were combined and built in Unity's timeline, saved us a lot of trouble. Had we started creating the full cutscenes in Maya, we would have spent time making unnecessary small movements for the characters when they, for example, turn to look at each other. Now the head and body rotations of the characters and other small movements can be made using Unity's timeline override tracks with the target system described in chapter 4.4.1 and mixing different animations together. The animations can even come from other characters as our project used Unity's humanoid system.

Building the cutscenes in Unity also gave us the possibility of adjusting the camera angles and cuts to fit the animations rather than the animations having to fit the fixed camera views. This way a lot of problem areas in the animations, such as character skin weights not showing correctly in the character's extreme poses, can be hidden by simply

not showing them. Adjusting the character skin weights to fit every single animation can sometimes even be impossible and creating blend shapes to fix the faulty areas takes more time which a small game development team does not always have.

One of the biggest challenges throughout this project was coming up with working pipelines and determining which way of working was the easiest and least time consuming. Testing early on many different aspects of the pipeline proved to be useful for making the conclusions presented in this thesis. Other noteworthy challenges came with learning the details of Maya's retargeting system and time editor and how to use them both to make the whole retargeting process faster. Maya's own manuals had little information for streamlining this process and many aspects of the procedure had to be determined by digging through online threads and coming up with one's own solutions. Through many of these threads and articles, I came to the conclusion that there is no one set way of creating cutscenes. Different companies and hobbyists use different methods and there are no clear right or wrong answers on how cutscene creation should be approached.

By showcasing the pipelines and procedures carried out in the case study discussed in this thesis, I hope to have shed light onto things one should take into consideration when starting off with their own cutscene project. I hope this thesis has been able to inform about the common problems as well as give useful tips for making the cutscene creation process more efficient.

As final words, I would like to acknowledge the valuable assistance and support from the Red Stage Entertainment team, who has made carrying out this project possible. Being granted the permission to use *The Fall* cutscene animations has improved the quality of this paper. A great deal of technical knowledge was gained throughout this project from my colleagues regarding the usage of Maya and Unity.

References

Adib, Payam 2019a. What is a 3D Animation Layout and Why Does it Matter? Dream Farm Studios <<https://dreamfarmstudios.com/blog/what-is-a-3d-animation-layout-and-why-does-it-matter/>> (17 April 2020)

Adib, Payam 2019b. Animatic in a Nutshell; The Storyboard Made Animated. Dream Farm Studios <<https://dreamfarmstudios.com/blog/animatic-in-a-nutshell-the-storyboard-made-animated/>> (17 April 2020)

Autodesk Knowledge Network 2018a. HumanIK Character Structure <<https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Maya-CharacterAnimation/files/GUID-5DEFC6E5-033C-45D5-9A0E-224E7A35131B-htm.html>> (30 April 2020)

Autodesk Knowledge Network 2018b. Time Editor <<https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Maya-Animation/files/GUID-E4B5DB7D-7351-4561-BD8B-60AC9D48DDF6-htm.html>> (10 May 2020)

Autodesk Knowledge Network 2016c. Define an Existing Skeleton for HumanIK <<https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-33859AA1-887B-4F8A-8360-F881C28EF351-htm.html>> (13 May 2020)

Autodesk Knowledge Network 2019d. Create a Custom Rig Mapping <<https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/Maya-CharacterAnimation/files/GUID-D8B0FEDE-C651-4B88-8DAA-C1EA73F7D538-htm.html>> (13 May 2020)

Autodesk Knowledge Network 2017e. Retarget Specific Attributes <<https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2017/ENU/Maya/files/GUID-EB2A8BB0-801E-4251-8C4C-09986049CCBB-htm.html>> (13 May 2020)

Beck, Bobby 2017. Animation Production: A Step-By-Step Guide to Making a 3D Animated Film. Artella <<https://www.artella.com/index.php/2017/09/21/animation-production-step-step-guide-making-3d-animated-movie/>> (3 May 2020)

Becker, Alan 2017. 12 Principles of Animation (Official Full Series). AlanBeckerTutorials YouTube channel <<https://www.youtube.com/watch?v=uDqjld14bF4&t=3s>> (25 May 2020)

Dowlatabadi & Winder, 2013. Producing Animation. USA: Focal Press.

Dream Farm Animation Studios 2019. 3D Animation Layout <<https://www.youtube.com/watch?v=nEX1XvJdvXs>> (17 April 2020)

Edulearn 2016. What Is Autodesk Maya? <https://www.edulearn.com/article/what_is_autodesk_maya.html> (22 November 2019)

GiantBomb 2018. Cutscene <<https://www.giantbomb.com/cutscene/3015-22/>> (7 December 2019)

Hancock, Hugh 2002. Better Game Design Through Cutscenes. Gamasutra <https://www.gamasutra.com/view/feature/131410/better_game_design_through_.php> (5 December 2019)

Khan, Rayyan 2016. 3D Modeling in 3D Production Pipeline <<https://animationjedi.wordpress.com/2016/10/16/3d-modeling-and-its-stages-in-3d-production-pipeline/>> (3 May 2020)

LeeT Game Development 2019. Humble Bundle, Dynamic Bone, Unity 2019 <https://www.youtube.com/watch?v=mdK30_UEbHA> (23 April 2019)

Nelson, Xavalier Jr. 2019. How Developers Create Cinematics. PCGamer <<https://www.pcgamer.com/how-developers-create-cinematics/>> (28 February 2020)

Ovrick, Jacob 2017. Pro Tips: 3D Modeling Best Practices. Artella <<https://blog.artella.com/index.php/2017/10/18/pro-tips-3d-modeling-best-practices/>> (3 May 2020)

Petty, Josh a. What is a Polygon Mesh? Concept Art Empire <<https://conceptartempire.com/polygon-mesh/>> (3 May 2020)

Petty, Josh b. What is Unity 3D and What Is It Used For? Concept Art Empire <<https://conceptartempire.com/what-is-unity/>> (28 November 2019)

Petty, Josh c. What is 3D Rigging for Animation & Character Design? Concept Art Empire <<https://conceptartempire.com/what-is-rigging/>> (3 May 2020)

Phillips, Dicky. Top 3 Challenges to Developing an Indie Game: As Voted By 9 Indie Devs. RenGen <<https://www.rengenmarketing.com/indie-devs-share-indie-game-development-challenges/>> (10 May 2020)

Schnitzer, Adam 2013. How to Build a Better Cutscene, Gamasutra <https://www.gamasutra.com/view/feature/131298/how_to_build_a_better_cutscene.php> (5 December 2019)

Uccello, Anthony 2018. Introduction to Unity Timeline. Raywenderlich <<https://www.raywenderlich.com/5315-introduction-to-unity-timeline>> (25 April 2020)

Unity Manual 2019a. Importing Humanoid Animations <<https://docs.unity3d.com/Manual/ConfiguringtheAvatar.html>> (24 April 2020)

Unity Manual 2019b. Root Motion - How It Works <<https://docs.unity3d.com/Manual/ConfiguringtheAvatar.html>> (13 May 2020)

Unity Manual c. Animating a Humanoid. About Timeline <https://docs.unity3d.com/Packages/com.unity.timeline@1.2/manual/wf_char_anim.html#matching-clips> (13 May 2020)

Unity User Manual 2017. Avatar Mask <<https://docs.unity3d.com/560/Documentation/Manual/class-AvatarMask.html>> (19 April 2020)

Wyatt, David 2012. The Art of Cutscenes. InMotion Gaming <<http://www.inmotiongaming.com/the-art-of-cutscenes/>> (5 December 2019)

Links to the animations and Unity examples

1. Animation iterations (of AiluExplosionReaction and AiluPetsReindeer)
<https://youtu.be/voOCqV9MTI0>
2. Iterations of Áilu's running cycle
<https://youtu.be/p3QJz1WBKqk>
3. Facial Expression Blend shapes in Unity
<https://youtu.be/TNRY8QnMPkw>
4. Dynamic Bones and Cloth Physics on Áilu
<https://youtu.be/9K6ROOt54wI>
5. Target Aim
<https://youtu.be/-aZtnkebNAc>
6. First iteration of The Fall
<https://youtu.be/L2JhSmtv4NA>
7. Second iteration of The Fall
<https://youtu.be/2r67f-FMT58>