jamk

Dive into 2D Technical Animation

Jerry Kovanen

Bachelor's thesis
May 2024
Business Information Technology

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Kovanen Jerry

Dive into 2D Technical Animation

Jyväskylä: Jamk University of Applied Sciences, September 2024, 41 pages

Degree Programme in Business Information Technology. Bachelor's thesis.

Permission for open access publication: Yes

Language of publication: English

Abstract

The 2D technical animation lacks in documentation and the job position seems to be overshadowed by 3D technical animation jobs. The job itself has people divided on what kind of things it could be about, making it hard to define. The path to the job position itself is not clear.

Purpose of this study was to gather information from different viewpoints about technical animation and understand common subjects that the job would entail. The importance was put on the tasks and techniques that 2D technical animator would need to know and learn about.

The research was done with mixed methods. The research contained qualitative research in form of literature review of various texts and videos and comparing them to find definite answers and information. The information learned aided in the practical implementation of this research, where the job of technical animator from popular standpoint on it was tested on small project on game engine.

The conclusion of the research led to that the 2D technical animation can't be defined in one type of job. The job title has adapted multiple meanings. The commons views to see the job was either as person who does rigging or the who develops animation tools and occasionally both.

Keywords/tags (subjects)

2D animation, computer animation, technical animation, game development

Miscellaneous (Confidential information)

Contents

Figures

1 Introduction

This thesis focuses more on the animation to the final implementation process but does not go through using animation tools outside of game engines.

The things that the research was to answer

1. What does it entail to do 2D technical animation?
2. Is there one direct path or pipeline to 2D technical animation?
3. Why is the topic of 2D technical animation so poorly documented online?
4. Are 3D technical animation and 2D technical animation basically the same field?

The research was done due to the lack of straight up documentation on the topic. Since the technical animation jobs are vastly different and mostly focus on 3D side of it, the documentation of 2D technical animation is barely anywhere. While researching the topic initially, finding anything comprehensive or detailed about the topic was very hard. The initial research did not give any idea on what the job would be like.

The author is aspiring to become technical animator on the 2D side due to their interest in the animation and technical side of game development acquired from studying at JAMK. It has been hard to go towards though since there is not much specific courses or tutorials for it. My first time hearing this term technical animator was in job listings and list of game development works.

After getting into Live2D model making, which is program made for animation and 2D models where you assign every movement to parameters and build your physics in the model there, the author got really into the idea of being technical animator themselves. The idea of doing complex rigging, getting it working on game properly and even getting the physics working in there, was amazing.

For this thesis, there will be simple project on Unity, where to try out setting up working animations and physics. The project will be done alongside with writing the experience along with the information from the sources. The reason for this type of research was to better comprehend the

information that has been researched and to give example on what 2D technical animator could be doing in actual job setting.

The thesis was meant to serve as more specific reference to this specific topic to go off of. For someone else interested in this field to understand what sort of topics they should familiarize themselves with to achieve their goals.

2 Information gathering

The research has been case study of the technical animator job title. The research process been completed through multiple methods of qualitative research. Start of the research done with literature review via Google Scholar where long time has been used looking for articles and publications of 2D animation and technical animation. Finding information on this specific subject can be hard in Google Scholar and you need to focus on game engine specific material like “Unity Animation” “game animation” and “Unity 2D”. There you need to go through the material to find the specific things form the technical animation and 2D graphics. If you search things like “animation mesh” or “2D rigging” you don’t get material related to it almost at all.

To get better luck on the material the research focus turned to Google search engine. In Google search engine results, there was articles and tutorials that explained the specific things like mesh and weights. Searches like “animation weights”, “2D rigging” and “technical animator” gave information that related to the topic way more than in Google Scholar on the topic. Most of the things you will find are the tutorials. While they are mostly there to tell you how to do one thing in specific program, they often explain the concept more in their own words and give you visuals for better understanding which was valuable for the research.

Even if most information could be found online, library was also one of the research places. Finding information on this topic was really hard in library and there could only found one valuable book in the games section in education in the library that was visited. You will definitely have better luck using material from the internet as reference. The virtual library Janet Finna was also used to find the books and essays. With Janet Finna, searching the obvious things often lead you to the right materials.

Information seeking for job position would highly value from information from someone in this specific position. While it was the intention to interview professionals in the field, it didn't work out in the end.

3 Technical animator

Technical animator is job title most used in Game Industry of position with tasks in between of programmers and animators. The job of technical animators varies differently on the studio. While in many studios they write animation tools in others they do rigging or even both (Reames, 2023) One thing Reames agrees every technical animator do is implement the assets into the engine.

At this point the title technical animator might be little confusing. There are no clear answers, and every studio sees it differently. What really seems to stay the same is working between all parts of the game development (Reames, 2023) You should be able to communicate and understand the work that comes to each part of the team. Reames even mentions that studios looking for technical animators are often listing all kinds of different jobs nobody else in the team is doing.

Some people use the term in more animation rigging standpoint. When looking at it that way, it has all kinds of details and nitpicks tied to it from the technical side like mesh, rigging, skinning and implementation (Cooper, 2019). When thinking of working as a technical animator with this viewpoint, you need to have knowledge of animation, game development, engine and even coding in certain cases. This is why it is not documented much and studying to become technical animator is not direct at all. In smaller studios the job of technical animation falls to the animator themselves according to Cooper. This explanation of the job title is more of what I am exploring in this thesis.

3.1 Technical animator or technical artist?

You might be thinking at this point what is the difference of technical animator to technical artist. They sound very similar, and things overlap on the description. While trying to find answers to this question, I discovered multiple answers to this question that contradict each other. Reames states that the difference would be technical animators' participation to all the game development departments, while technical artist is mostly tied to the graphic team.

4 Film animation

Animation as word has taken many new meanings under it as the techniques have been developed over time. It can be summarized as a sequence of images that create illusion of movement to unanimated objects. This can be done in traditional ways like drawing frame by frame every movement for the character and combining them to video or with computer-generated imaginary ways like making 3D models move to create animation and rendering it as film.

To create eye catching animations with dynamic movement and high-quality results, you can't just follow the real life. After the publication of The Illusion of Life: Disney Animation (Thomas & Johnston, 1981) we were introduced to the concept that Disney used in their animations; the 12 principles of animation. To this day these are the animation 'rules' that are considered the fundamentals of animations.

4.1 History

4.1.1 Start of it all

Computer animation has been around for tens of years now, but film animation itself has been around much longer. You could trace the animation back to year 1900 when J. Stuart Blackton started making stop motion animations and capturing them to film ("Encyclopaedia Britannica", n.d.) His first animation, which was called Humorous Phases of Funny Faces, was made with camera and blackboard that he drew into (Frank, 2020) This animation inspired many others to give the new medium a change and we got animations like Little Nemo by McCay that was based on their comic Little Nemo in Slumberland and Émile Cohl with his animation Fantasmagorie.

Figure 1 *Humorous Phases of Funny Faces*

Note. from *Humorous Phases of Funny Faces* [Video], by J. Stuart Blackton, 1906 (https://youtu.be/wGh6maN4l2I?si=Bkv1QU73VkMGzhUL).

In 1919, the first animation star was born which defined the animation scene going forward; Felix the Cat. Felix the Cat was released by Paramount Studios with their first animation Feline Follies (Bondfield, 2023) Felix the Cat has been credited for being the inspiration going forward in animations with his looks with his large eyes and circular head (Najeeb, 2020)

Figure 2 *Felix the Cat - Feline Follies*

Note. From *Felix the Cat - Feline Follies* [Video], by Paramount Studios, 1919 (https://youtu.be/7HskWL82GeQ?si=LbQzUuBl-KA3sNeJ).

4.1.2 Rise of animation from 1928

While the animations before were impressive, they really blew off when Steamboat Willie by Disney came out in 1928. It was short animation with music, sound effects and dialogue; something that the animations never had before as whole (“tvtropes”, n.d.) Even if Steamboat Willie is often credited as being the first cartoon with sound, there were several animations before hand that had sound in them. First one of them were animation series Song Car-Tunes by Max Fleicher and Dave Fleischer in 1924. This series was animated in style of karaoke with bouncing ball going through the song (“Zedem Media”, n.d.)

The Golden Age of animation started with the Steamboat Willie. The Golden Age of American animation was time period that lasted about 30-40 years from when Steamboat Willie came out (“tbtropes”, n.d.) In The Golden Age there was significant rise of animation on popularity and the

techniques. The animations of that time were done mostly by traditional hand-inking The animations started being longer film length pieces or even full series aired on TV. They also started to have colour. This period of time was start of many iconic animated characters that we still see today.

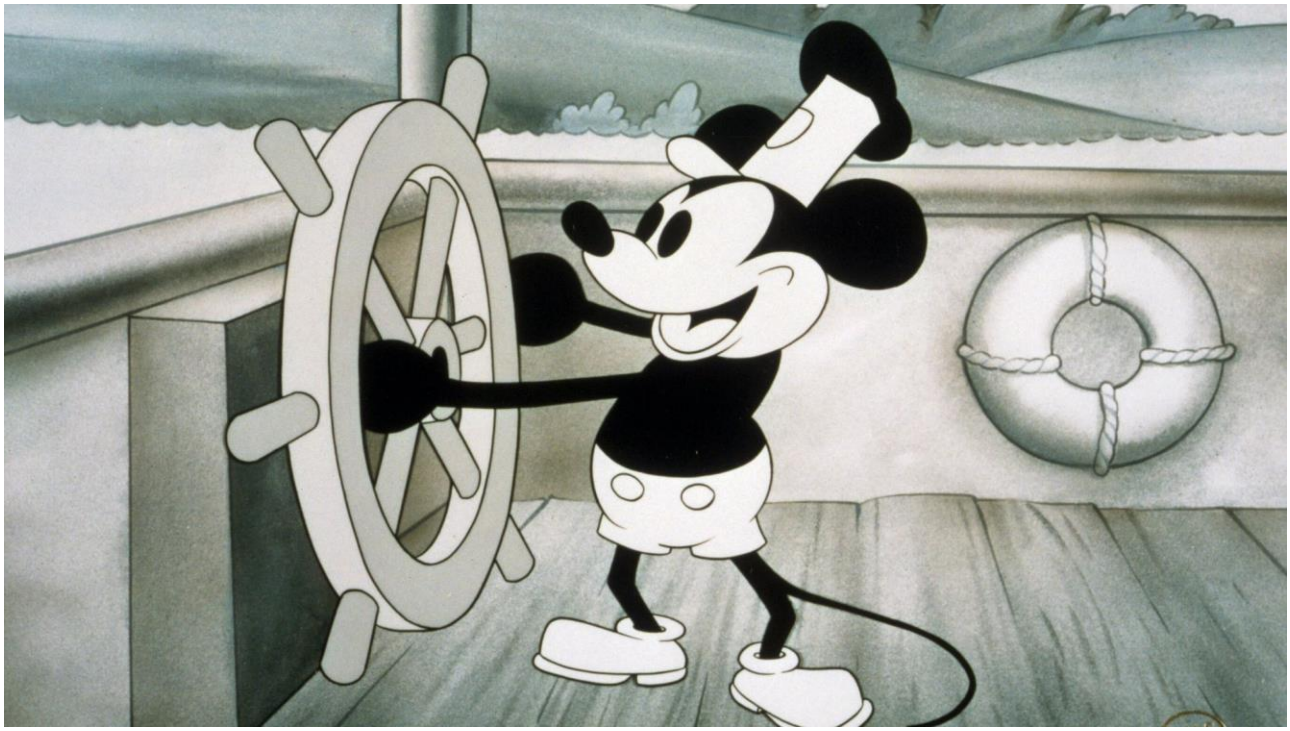

Figure 3 *Steamboat Willie Mickey Mouse*

Note. From *Steamboat Willie Mickey Mouse* [Video], by Walt Disney Animation Studios', 1928 (https://youtu.be/BBgghnQF6E4?si=o5e1nvZ1lMjGbqEs)

4.1.3 The rise of computer animation

When the computers started grow in popularity in 1960s, so started computer animations (Barnhart, 2024) Computer animations were step up from the traditional inking, due to the computer being able to produce much faster and less costly animations. The first animated films were small scale animations like Hummingbird by Charles Csuri (1967) and Metadata directed by Peter Foldes (1971) that used vector animation.

The quality of the computer animations kept rising when the computer power increased and the software were developed in 1980s (Barnhart, 2024) Animation studios started to use 3D CGI, computer generated imaginary, that was used in movies. The morphing and tweening in animation were developed in this time, allowing whole new level of animation production with photorealistic animations.

The industry keeps growing along with the technology and things like 2D computer animated movies and motion capture became big things. The computer animation has improved a lot since then from the first fully computer-generated animation Toy Story by Disney to our modern-day games with realistic graphics like Tekken 8 by Bandai Namco Studios Inc. It is norm for animated media and is used for things other than films like games, applications, and education.

4.2 Game Animation

When we think of game animations, we think of tens to thousands of individual animations for different characters and background elements and cinematic cutscenes. They can be basically separated into gameplay animation and cinematic animations (Cooper, 2019). For example, when the game starts, it is very common that you see animation that sets up something about the story and the world where you do not have interactions. This animation would be called cinematic animation or more commonly referred as cutscene. When you move your character around to find guest, that is the gameplay animation.

4.3 3D animation

3D animation is, like the name says, animation in three-dimensional form with 3D models. It is commonly used in games, films, and education, especially since the possibilities with it are growing rabidly with new tools and technological advancement. In 3D environment rather than drawing your animation frames yourself like traditional animation, you use models that you move around frame by frame or less to generate movement. This can be through moving the model around or using rigged model you can manipulate different parts with or even motion capturing, which is method of animation where you record people and objects to use the movement in 3D models (Rokoko, 2021)

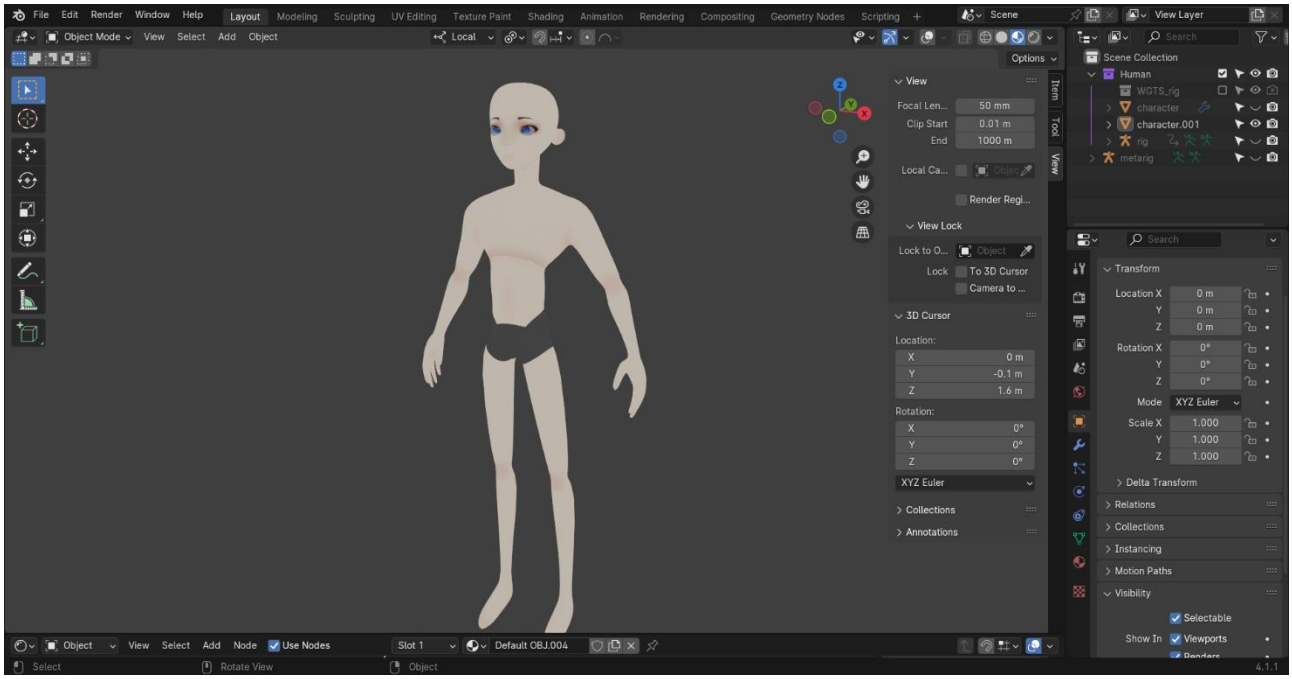

Figure 4 *3D model you can use for animation*

5 Programs used

5.1 Unity

Unity is popular game engine made by Unity Technologies. It was released 2005 as Mac OS X game engine but have since been implemented to support multiple platforms. Unity is completely free to use as long as the profits using it do not exceed over 100 000$, allowing smaller indie developers to make their games without engine costs. Unity supports both 3D and 2D development and offers ready made tools for both.

Unity offers great tools to develop your games. It uses the ever-popular C# as its coding language, it supports drag and drop for visual elements. (Jared Halpern, 2019, p.?) It has implemented features like mapping, render pipelines, timelines, physics engine etc. The Unity asset store even allows the users to buy or get for free user made functions and assets, making your time making a game easier.

5.2 Clip Studio Paint PRO

Clip Studio Paint is multi platform drawing, animation and comic program developed by Celsys. It has been developed since 2010 (Clip Studio, 2022) and has since been developing visual tools meeting industry standards for illustration and comic making. It has tools like 3D models, comic panels, animation timeline and onionskin, vector layers and variety ways to export your visuals to other platforms.

It comes in 2 different versions; Clip Studio Paint PRO and Clip Studio Paint EX that are both available for one time purchase for Windows and MacOS and as Annual and Monthly for other platforms like iOS and android. PRO is mostly focused on illustration and small animations, while EX adds on all the animation and comic tools.

6 Game Engine and animations

6.1 Rigging tools

While 2D cartoon animations are mostly focused on doing the characters frame by frame, in games it is more efficient to use rigging method instead. Rigging, also known as skeletal animation, is technique of creating the skeleton with bones to the object to give you control of the rotation and movement of certain assets. You can change the structure or move specific assets attached to the bone to make animations. It makes it easier to make quick animations without having to draw more and it allows more control of the character like manipulation of the physics to create procedular animation.

When you hear the word mesh you probably think of 3D models with hundreds of lines going over grey model. These are called meshes, bunch of geometrical shapes that form the character together (Deepmotion, 2018). In 3D, these shapes are flat and different sizes that form characters and objects in three-dimensional space. The more there are, the more detailed the character is in terms of shape. According to Deepmotion, meshes are not common in 2D animation software's.

The difference between 3D and 2D meshes is the dimension its in, but also the amount of meshes does not directly affect the quality of the character itself, only the animation quality made with

them. Depending on the part of the asset, sometimes simpler is better (Figure 5) and benefits you more than over complicated mesh.

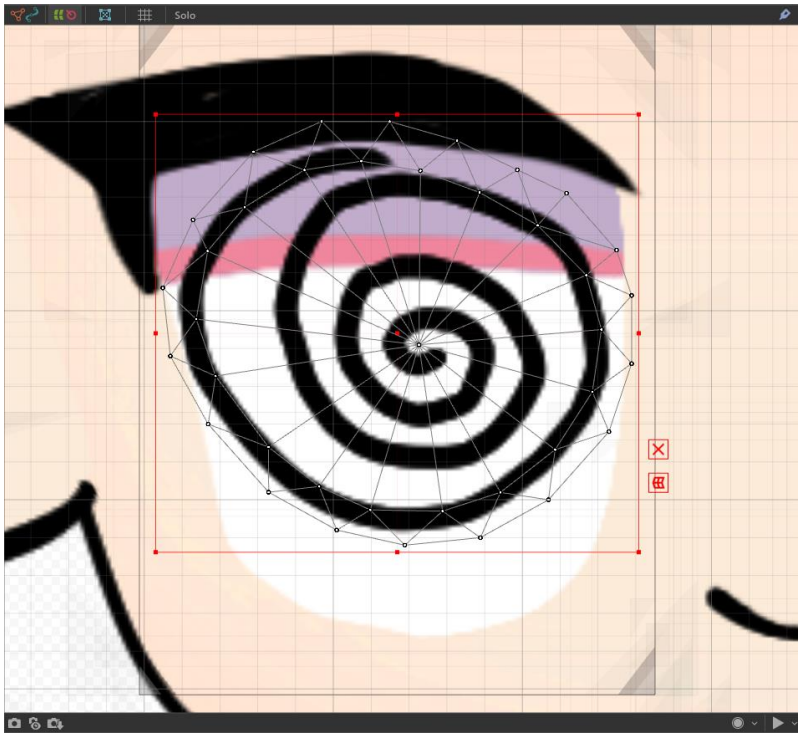

Figure 5 *Example of simple, but working mesh*

Addition to the bones, some animation programs and game engines also have weights. Weights and the process of weighting is where you define which parts of the mesh specific bone affects (DeepMotion, 2018) This makes you have more control of your rig, resulting in better results. It is often visualised as colours you can paint onto the mesh that are based on the specific bones' colours. When the mesh are is more of the bones colour, the more it should be affected by the bone. You might not need to cut some parts of the character if you use weights well.

6.2 Physics Engine

You can approach physics in games in a way of making the individual physics for the game, usually better for smaller games. Physics engine is another and more common way to approach physics in games according to Ian Millington (2010). Since making physics was always tedious and hard to do which led to the development of physics engine, which reuses the physics and calculates all the mathematics for the physics for you. Ian Millington even compares it to a big calculator.

6.3 Animation State Machine

Animation state machine is animation controller that is responsible for connecting all the animations and set rules in visual way. Halpern Jared (2019) states that the state machine used to determine the animation clip to play for the object it is attached to. You can change details like the transitions in there. Game engines like Unity, Unreal Engine and Cobalt have state machine system built in them.

Animation state machine is especially useful as you can reuse the state machine and the behaviours inside it multiple times. If implemented individually, you run into the problem of programming the same things multiple times, making yourself work more than necessary (Tsung, 2016)

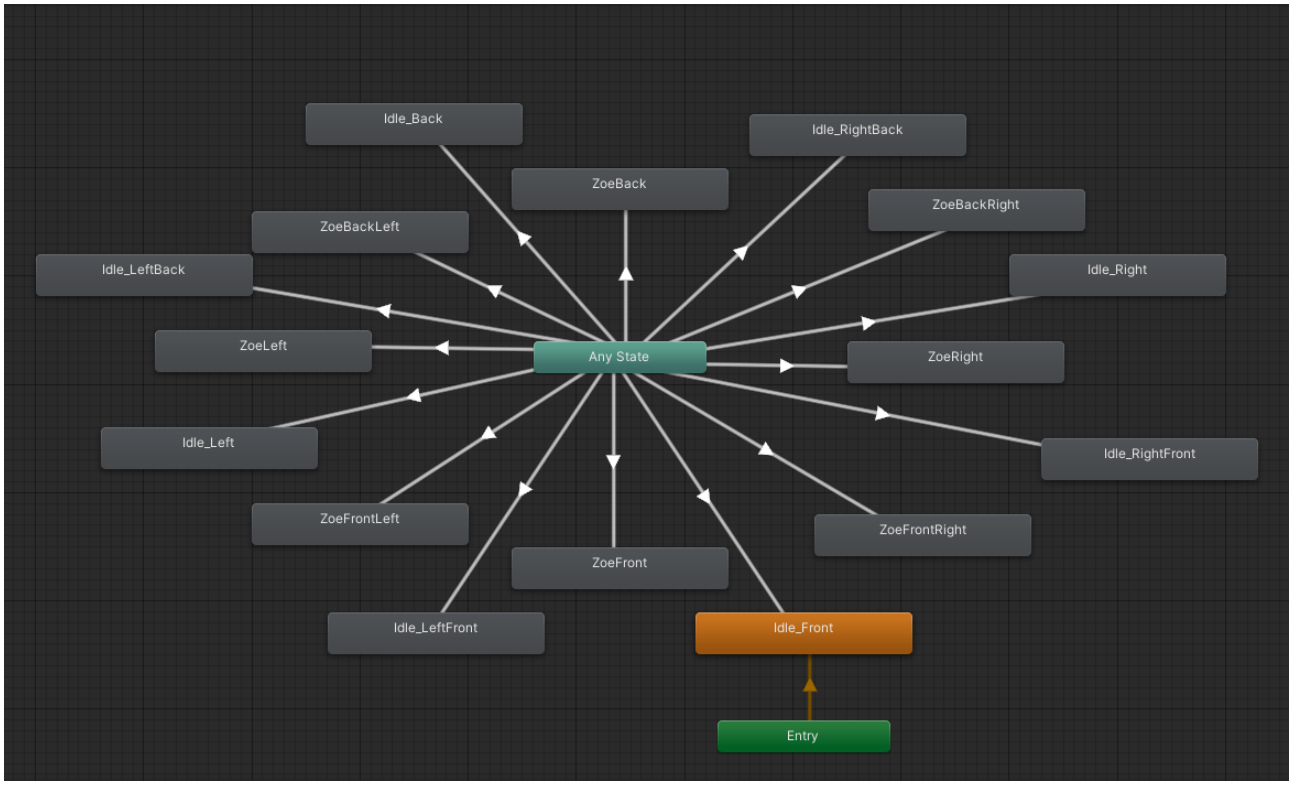

Figure 6 *Animation State Machine in Unity Engine for character movement by Kovanen*

6.4 Sprites

In 2D games, other popular method of 2D animation are sprites. While sprite itself is just 2D image, but many of them put together in game engine is sprite animation, very similar way to traditional animation for films (Halpern, 2019). Sprites are great for games pixel art and simple art styles. To easily get the sprite animations to the game engine, you export them in the same canvas from equal distance to each other.

Figure 7 *Example of walking sprite sheet*

6.5 Procedural animation

Procedural animation is in game animation that is generated by the game engine by calculating the forces that affect it defined by you (Kennedy, 2012). The procedural animations are randomized and can get force from things like wind and character movement. For example, you can set up the player characters hair to be affected by both wind and character movement. That way when you run, the hair flows backwards like it would in real life. While you could bake the hair animation in for the movement, you limit its movement a lot by eliminating other forces that could be affecting it.

6.6 Separating parts

Separating parts is when character or object is cut into several pieces that are easier to manipulate in more complex motions. This process is often referred as cutting in Live2D rigging scene (Omori, 2022) due to the process of 'cutting' can involve cutting artworks that have not been prepared for animation originally. To give you example on parts to separate, you should cut the arms to at least 3 pieces, the hand, the forearm, and the arm as separate parts, cut and little overlapped on parts where your joints are. This way when you do your rig, the hand naturally has the joint points in

place. The usage of the rigged model is important when choosing the things to separate in your model.

When talking about complex cutting, the asset can go to hundreds of different layers. You can get more control over the animation and more complex movement with hundreds of layers. You see these types of separated characters often in rigs that are using face tracking like Live2D models.

Figure 8 *Simple separation of model*

In 3D graphics, there are hardly cutting involved in the same extent as 2D cutting. You are able to manipulate the models' lot easier with 3D mesh making cutting all parts unnecessary. It is however necessary part in modelling to achieve specific goals (Zikei, 2023) Zikei lists out in the *Step-by-Step Guide to Cutting Objects in 3D Modeling* plenty of reasons to use cutting in 3D models such as more precise modelling, articulation of model parts and for 3D printing.

6.7 Coding

Coding is essential part of game development. Coding is translator between the human and computer like Vanesha McGee (2023) summarizes it. There are different coding languages that translate our inputs into series of numbers that the computer understands. Some of the popular programming languages are Python, C, HMTL, Java and C++.

Coding is used to develop programs, applications, websites, and games. There are multiple different programming languages that work for developing games that depend on your preferences and the engine you are using. C# and C++ are the most popular ones (Johns & Semah, 2024) due to their performance and the popular engines they are used in; Unreal Game Engine and Unity Engine.

6.7.1 Web animation

While you can control your ready-made animations or assets rig with your code, there is also ways to do animations only with coding. These animations are called web animations and are often used in web pages to bring more appealing visuals, making the user experience better or interactivity (Ayebola, 2023) Popular coding languages for it are CSS and JavaScript due to their capabilities on it.

The reason why you would want to code in the animation instead of adding gif or individual png raster layers to your site is the huge size they have (Junferno, 2021) In web browser this could result in easy crash or long loading times and you having to lower the quality of the animation itself. This is certainly why making and replicating your animations in the code is crucial for smooth user experience.

Most of the time these animations are in smaller scale and interactive in the websites like scrolling down and showing the new information in cool fade, but also simple vector and pixel animations. There are also ready-made tools that let you make the animations without code first and convert them into code language for example svgator.

7 Project: Functioning Stickman

7.1 Introduction

In this project, we will be trying out creating experiment on Unity by doing a 2D project with working animation and physics. It will be more focused on the technicalities of it that using time to make create complex visuals for it. It will be using stick figure drawing cut in few pieces, and the animations and the rigging is made inside the Unity engine. While using other program for it could give better idea on how the process would go more commonly, working in Unity as the only program will save time and better keep focus on the parts other than creating visuals and animating the character. Clip Studio Paint will only be used to make the stick figure and implement it to work on Unity Engine.

Physics engine will be used due to it being more common than doing the physics individually. There will be character animations, but not cinematic in the process itself. The part of making cinematic animations in Unity will be discussed for differentiate the process from just character animations. The character will have working rigging and animation made with it. The attempt was to make procedural animation that works with the animation for the character. The implementation of the animations to the code would have been done with C#, code language used by Unity engine. To tie the animation to code, Unity's animator would have been used to create working state machine. In the end the result should be application/game that just shows the animations working, with at least one control for you to control the animation in real time.

When it comes to technical animation, the common way to get to this position is to work first in either animation or technical field (ScreenSkills, n.d.) before it to learn the larger picture. Learning both before hand in school is not too common and takes so much more time to reach. Since the studies of the author have been focusing on the animation side more, this method was used as a way to give opportunity to expose the author to parts of the process they are not used to.

This project servers to answer questions about what the 2D technical animation would be. It will also include analysis on the difference on 2D- and 3D technical animation.

7.1.1 Case Study

This project has been done with case study methods of research. The pipeline of 2D technical animator was constructed through this research. In this case study the research done till this project was considered and qualitative research methods were used as more information seeking. The case study looked at the concrete details of the technical animation and challenge some of the steps in it.

In this case study, the pipeline of this topic was inspected and based on the findings, the research was used into the project. The qualitative research was conducted by using interviews and analysis of sources.

7.2 Character preparation

To prepare the stick figure for the project, Clip Studio Paint PRO was used to create the character. To do this in the program, you can use the basic shape tools provided by Clip Studio Paint and symmetry tool. For this project, the character was separated into 10 parts of the character with each limb having 2 parts and the body and head as individual. This simple character is enough for project when the purpose is testing the animations and physics.

When cutting character in Clip Studio Paint, you need to separate all the parts to different layers and name them accordingly. Good naming conventions are necessary for smooth game development process; you find the parts you need easier and implementing the specific parts to different functions and parameters is much easier with “forearm 1” than “img_3248”. When naming it is recommended to name the left and right parts from the perspective of the character themselves.

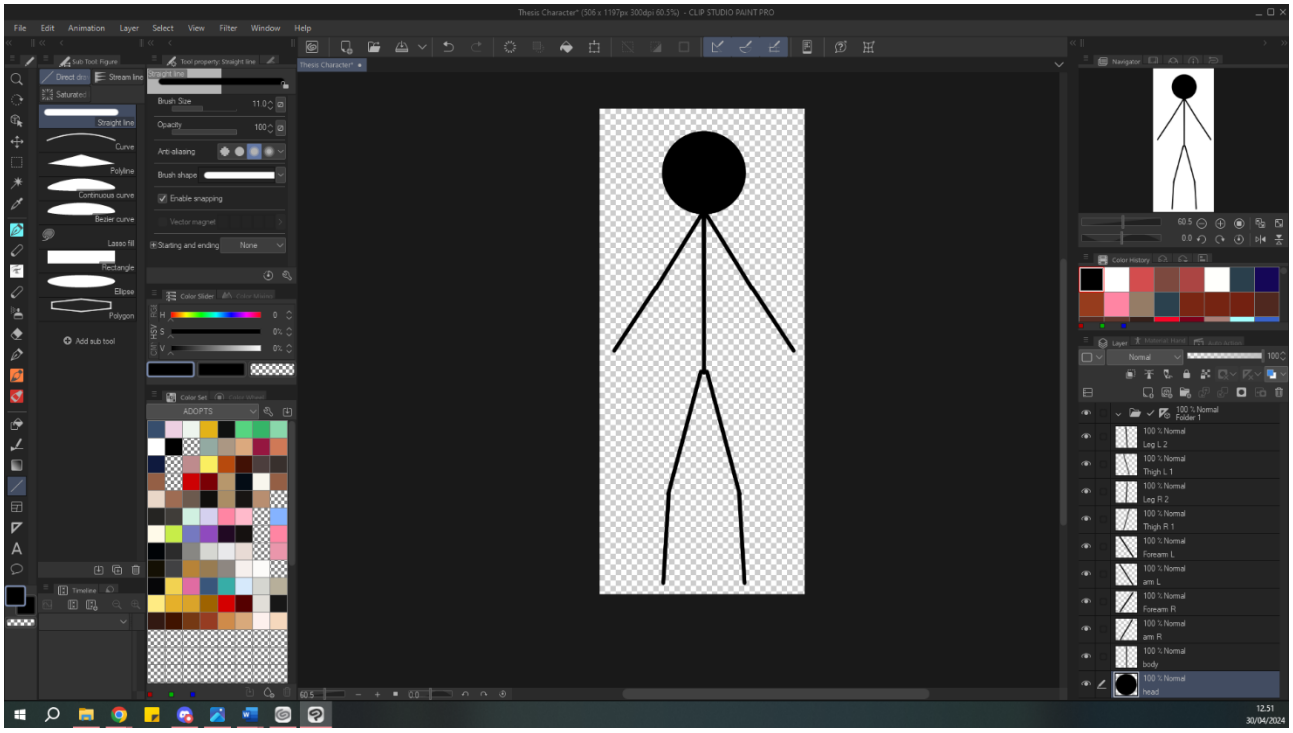

Figure 9 *Stick figure made in Clip Studio Paint PRO. All the parts are named clearly.*

To get the character from Clip Studio Paint to Unity Engine, export the character as PSB or PSD File, that are supported by Unity engines PSD importer. The reason of using PSD Importer rather than to make sprite sheet of the character is that the PSD Importer keeps the layers into the correct places and does not separate them from each other. This makes the model more accurate and the workflow shorter.

7.3 Unity Project staging

This project will be using Unity version 2021.3.37F1 as it is up to date version of Unity engine. To start, make new project on Universal 2D template. The author used this template over 2D (Built-In Render Pipeline) to have the Universal Render Pipeline already installed and configured to the project. It makes it easier to add shaders and effects on the project later if you decide to study other things than technical animation in this same project.

To start drag the PSB file to Unity. It automatically is parted into the layers you made in Clip Studio Paint and keeps the original placements of the layers, which makes the whole process easier. They are all tied to the character itself in Unity Engine. You can add the character to the Unity project

scene to examine it in the Hierarchy panel better and preview your changes and animation when we do them.

7.4 Sprite Editor

To go forward with making the rigging you open the Sprite Editor. The first thing you see is the Sprite Sheet where all your parts are separated from each other. If the parts overlap each other, you can manually move them to their own spaces. The rigging menu itself can be opened from the top left options from ‘Skinning Editor’.

In the Skinning Editor, you can find tools to make mesh, control weights and add the bones to your asset. You can test your rigging with the Preview Pose feature, to make sure your weights and rig are assigned properly.

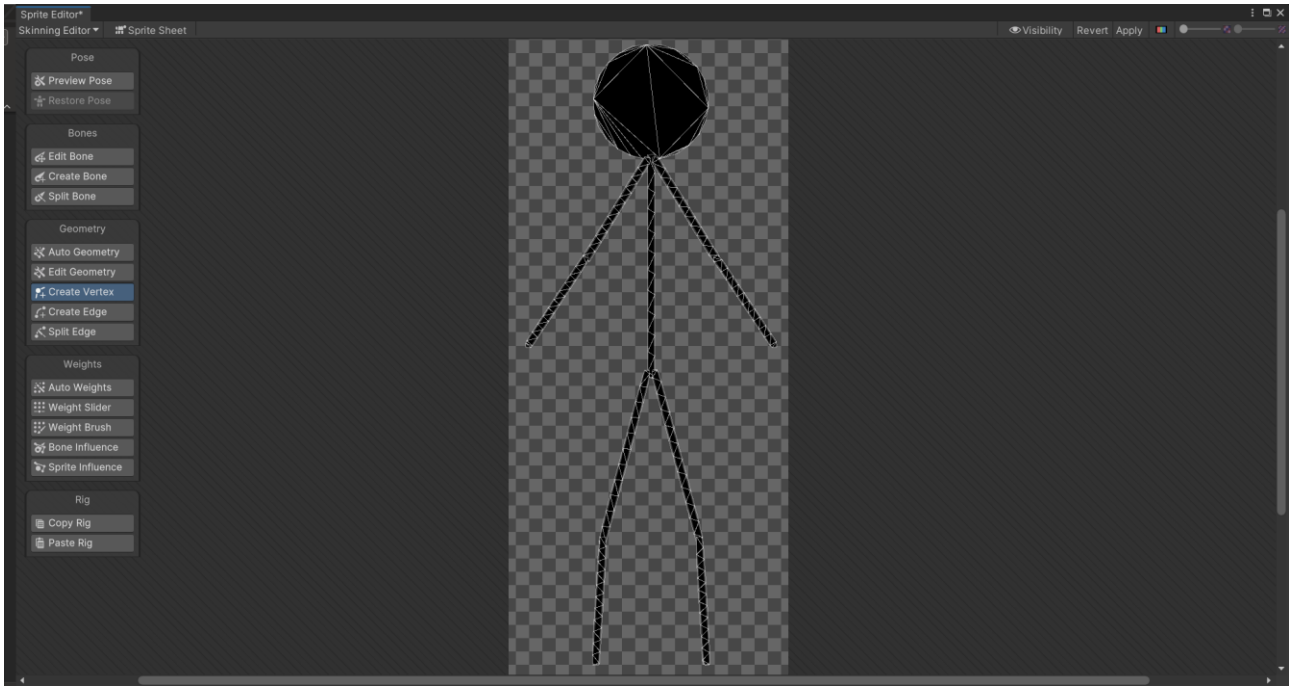

Figure 10 *Skinning Editor in Unity Engine*

7.4.1 Mesh

The first thing you should start with is making the mesh better. You can see the tools for it in the geometry section. Unity automatically makes meshes for all the parts, but they are not always the

most optimal for your character. Especially for characters with complex parts and detailed animations, clean mesh is very beneficial. In case of the stickman in this project, the only part that needs mesh fixing is the head. To create and move the vertexes, the points that form the mesh, use the 'Create Vertex' tool and double click the part you want to change. With this tool you can change the mesh to your benefit. To circular shape with not much details like the stickman head, vertex to the middle makes the circle more clean and easier to reshape, now that the geometry is about symmetrical.

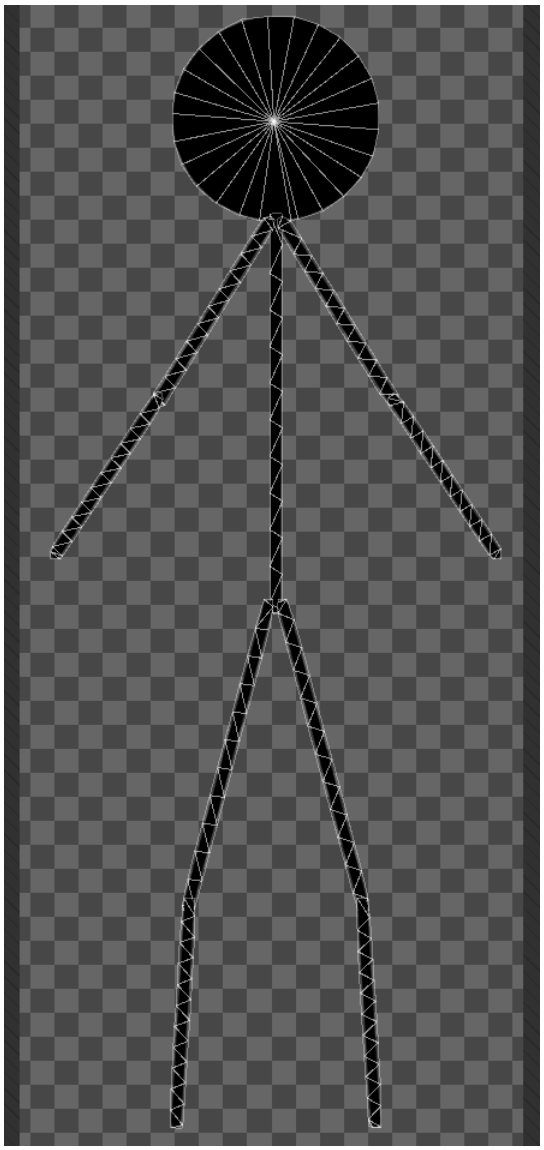

Figure 11 *Stickman with mesh*

7.4.2 Bones

Now that the mesh is okay, bones should be done next. To do this you need to create bones with the ‘Create Bone’ tool. Every bone tool works very differently depending on the program. It is however almost always made of tool of with one bigger point with long length that ends with small point (figure 12) The big point is the pivot point that act as the point that the bone rotates around.

When making bones on Unity, you need to make sure the joint hierarchy is okay. The joint hierarchy is the way the bones are connected to each other and in result affect each other. The first bone in the character is the root joint that every other is connected to (Lee, 2023) The root joint is usually placed in characters torso that ties all the other bones together. For human characters, figuring the hierarchy can be done by absorbing where the joints are. Like for example, when thinking of rigging arms, you can divide it into 3 main parts, the arm, forearm, and the hand. When you move your arm, forearm and hand move with it, while moving hand, doesn’t necessary affect the arm. So, with this logic you make the bones connect from arm to forearm and then hand. You can see the hierarchy when you press the visibility menu open and from there you can assign the hierarchy if it wasn’t done already in the bone creation phase.

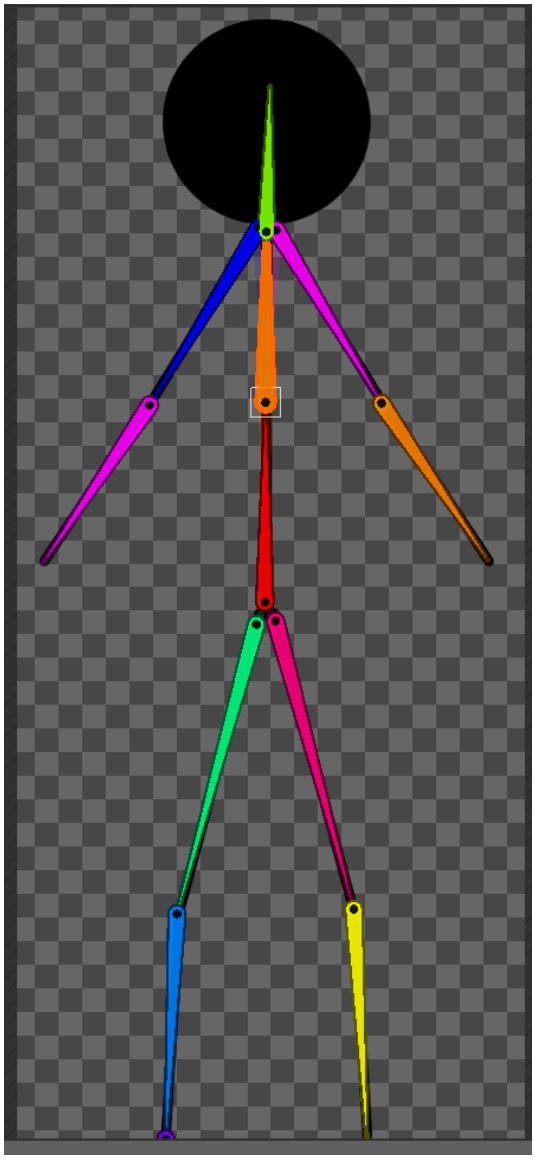

Figure 12 *Model with bones created, the stickman model only needs 11 bones*

Now that the bones are created, you need to name them and assign what the bones control. In Unity Engine you need to manually assign for each mesh which bones can influence them or assign each bone which mesh they can influence. You can do that with the Sprite- and Bone Influence tools. whichever tool you use is based on preference or convenience. The bones should be controlling the nearest mesh and, in some cases, the one besides it to give the mesh more realistic movement.

7.4.3 Weights

To even get your character moving you need to add the weights to the mesh. There are few different ways to add weights to your character; you can use the auto weights to make easy basic weights based on the position of the bones or use the brush and slider to make your own. The easy way is to first make automatic weights on the mesh and after that correct it with the weight brush, this way you have already foundation ready and possibly even have good weights on the mesh already.

To properly test the weights, you can use the posing function to try the model in action. This reveals your flaws and let you adjust the flaws to perfection like elbows. Elbows and other similar joints usually have the problem of having too much influence on the arm or forearm. While the weights could be only affected with the bone associated with it, you can achieve more natural movement with having the other bones near it affect it in the near places. When doing weights, you can notice the importance of good mesh of the parts, and this is a good time to fix them. Make sure to apply all the changes you made in Sprite Editor.

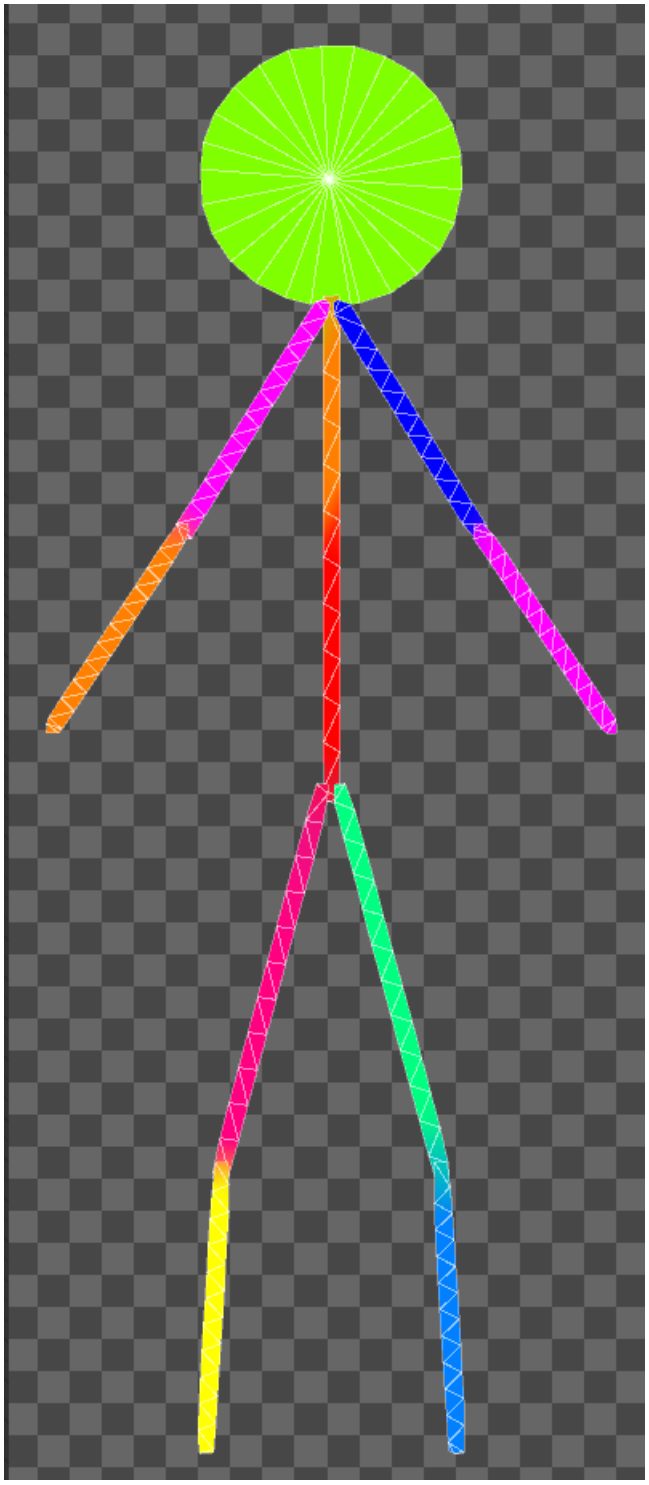

Figure 13 *The stickman with the adjusted weights*

7.5 Animating in Unity Engine

Now that the model is properly prepared, you can use it for animations. The main way in Unity Engine to make animations is the Animation tool. There you have timeline you can make keyframes

in. Timeline is the animations ‘canvas’ that you put all the movement in, and you can decide which frame each thing happens.

To use your character in the animation, choose your character in the project, go to the animation window, and create new animation. If you do not have the animation window on your Unity view, you can find it in Windows menu under animation. That opens the timeline, and you are able to start animating your character. By default, the keyframe editing mode is disabled, so make sure it is on to safe all your movement with the red button.

In the timeline itself there are keyframes that are in this context basically the point where you have made the movement change. This is usually very important movement or end of movement that you need. Since the Unity animator has technology to automatically animate the in-betweens, you do not need necessarily keyframe every single frame for the movement to happen.

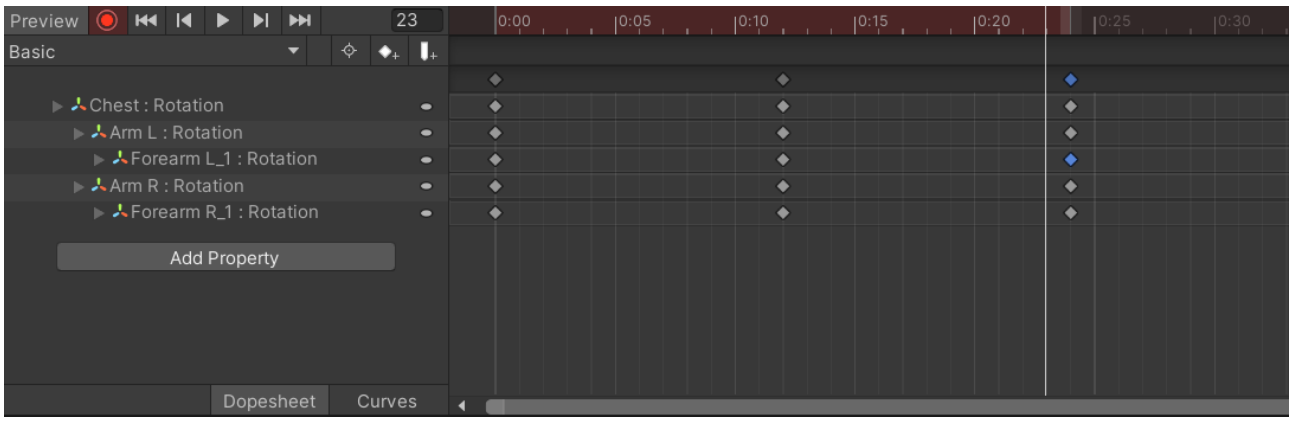

Figure 14 *Timeline in Unity Engine*

The in-betweens and their movement can be further edited with the curves. Curves are in different editing mode of the timeline that shows the keyframes and the in-betweens as points and curves to indicate the movement. These curves define how the engine makes the in-betweens move in the animation. You can change the curves in Unity Engine by moving the control points in the keyframes or by choosing one of the Unity’s own presets for them when right clicking the keyframe.

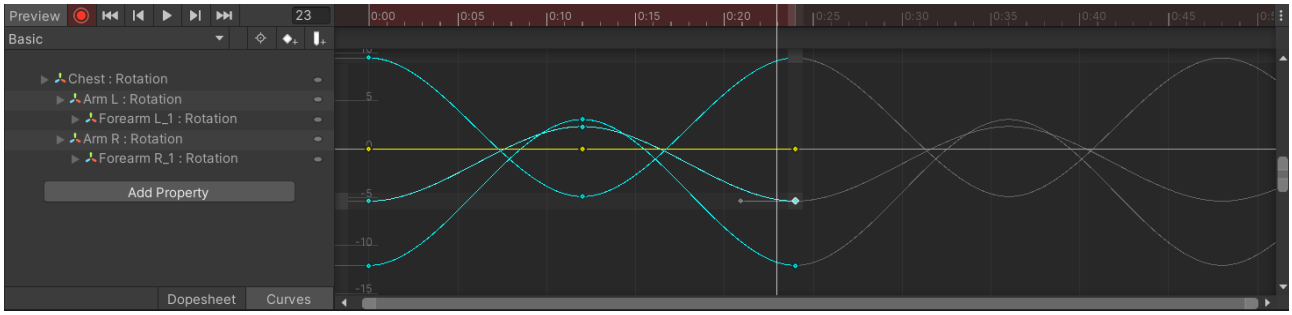

Figure 15 *Curves in the Unity Engine animation tool*

For simplicity's sake the animations for the character that was animated was one where they lean forward and another of them kicking forward. They have the same starting and ending frame, so they loop around themselves, making it look like someone jamming to a music and that that they are kicking. For animation that you want to have playing for longer while they are best to make looping to avoid it looking weird with sudden change in the pose when the animation reaches the end and starts over.

7.5.1 Cinematic animation

In Unity there is different tool for the cinematic animations inside the game called Timeline. While talking about the timeline in the animator as the term used for keyframes, this Timeline tool in Unity Engine is tool that you can construct animations together. While in this project it is not going into the cinematic animation or other various things you can use Timeline for, for better understanding of Unity animations and differentiating from the animation tool, it is important to understand.

The Timeline tool is much more different than the animation tool. You do not have any control of your character rigging in it, but you can put the character animations in there you made in animation tool. You can also set up various other things like audio, signals, or activation requirements. With all the tools in it, you can create things like cinematic animations, particle effects and gameplay- and audio sequences (Unity, 2018)

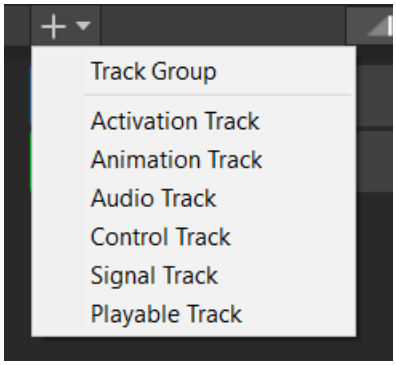

Figure 16 *Timeline track types*

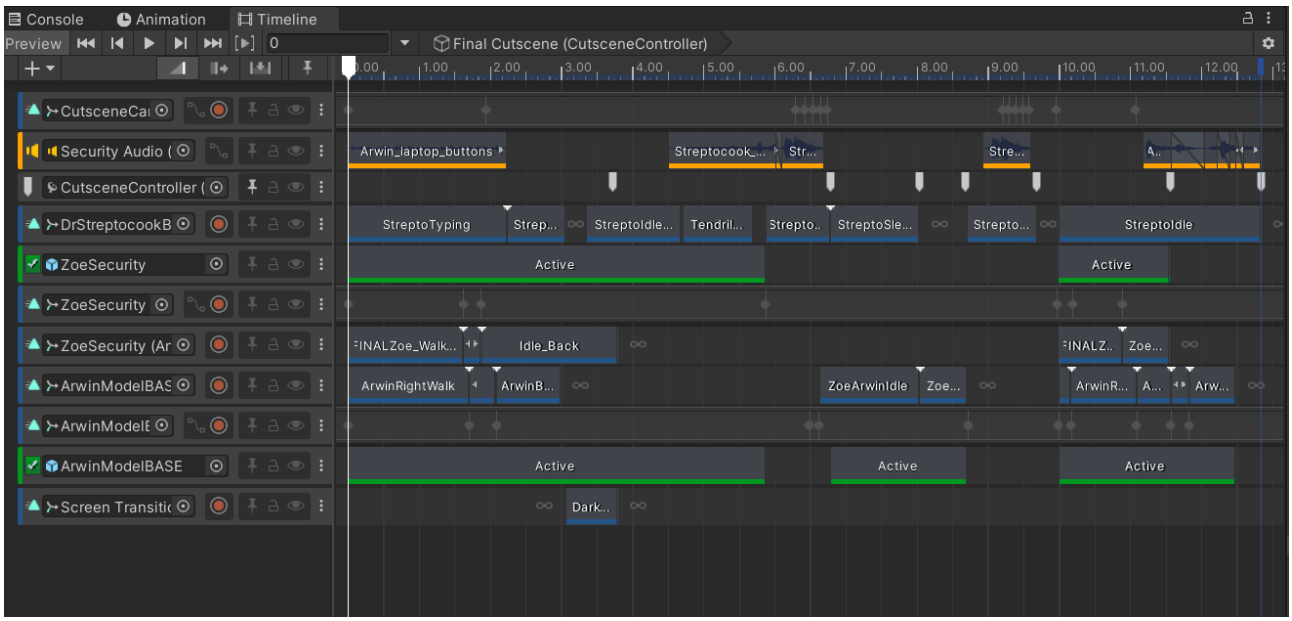

Figure 17 *Timeline made for Rootless: Memories of Green*

7.6 Animator

Now that the character has animations we can move on to the more technical part of this project. If the character doesn't already have one, you need to add Animator component to your character and assign it controller. This makes it so that the animations of your character are all put into the animator where you can control how they interact with each other like transitions and the blending.

Animator is the Unity's animation state machine (Unity, n.d. State Machine Basics.) It shows all the states of the character animation and the transitions in visual way and let you control them. In the animator tool you can see the layout area with all the animations tied to the character and the Entry, Exit and Any State Nodes and the parameter pane (Unity, n.d. The Animator Window.)

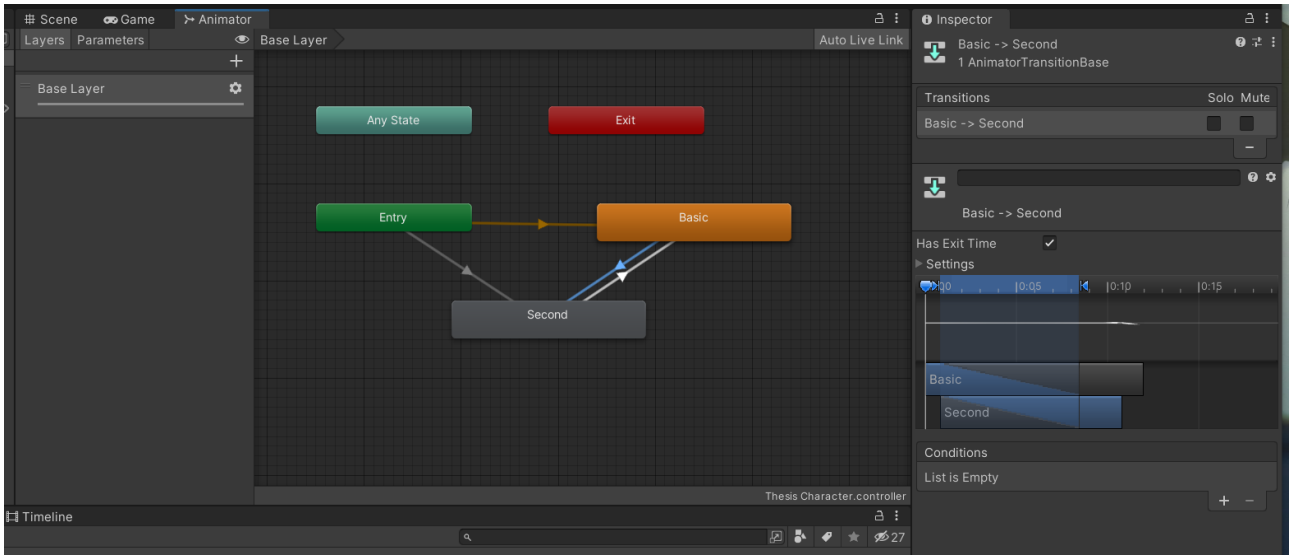

Figure 18 *Animation State Machine with 2 animations*

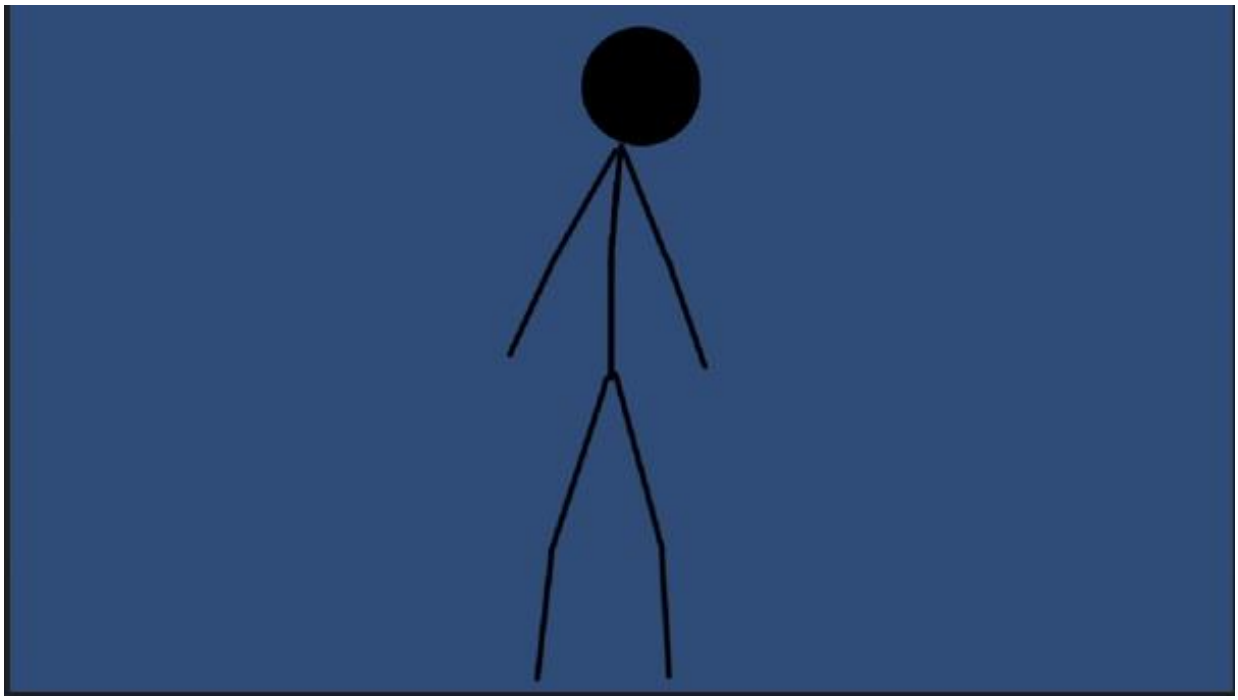

Figure 19 *Character animations with working transitions*

Note. (https://youtu.be/3Ujdg8eYL8A).

7.7 Physics

The physics of the character ended up causing more problems that expected in this project. The documentation of 2D Unity physics is not as easy to find and has several different ways to approach it. The biggest problem of the information gathering and moving forward with this part was the different ways for people to rig in Unity Engine and the documentation on the topic being too old to have the correct up to date tools considered that Unity has to offer.

The first attempt with the physics was with the rigidbody2D. Rigidbody 2D component makes it so that your game object can be affected by physics and control how the physics work (Unity, n.d.) I attached the component to my characters bones. At first this triggered the reaction of falling down out of the screen. To prevent this you put the constraints for the bones with Freeze position function, where you tick out X and Y, this however lead the bones to not have any reaction and the simulation of the physics does not work. To go around this problem the Body Type that is by default dynamic needs to be changed top Kinematic. The best results of this method after playing around with the settings in Rigidbody 2D in my model was setting them Kinematic body type and Freeze rotation Z. The results were weird monster like movement (Figure 18)

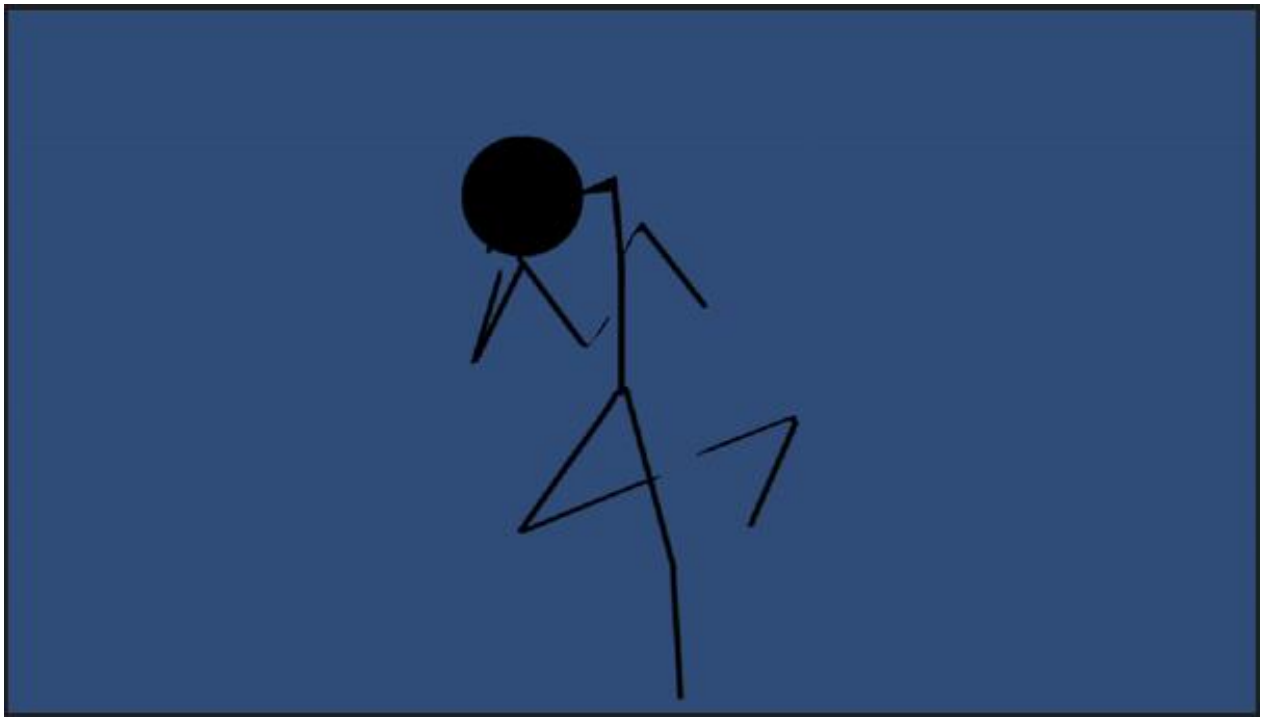

Figure 20 *The result of first physics test*

Note. (https://youtu.be/3Ujdg8eYL8A?si=P1G3cdskJRIa8qhf&t=2)

The second way to try out the physics was trying a technique for softbody physics with Sprite skin script (PyroPhysics, 2021) It needed a few more packages before it could work. The Sprite Skin script, provided by 2D animation package, needs packages Burst and Collections for you to use deformation batching. This will allow us to create bones to the Sprite skin script from the ones we already created. When ready with the download, the sprite skin was added to the Character prefab and automatically got put into the each individual sprite of the character. In each sprite the bones attached to them were the bones that the weights were assigned to. Without changing any settings on the sprites the character stopped moving, not performing the animations in the Animator anymore. After messing around with the settings like assigning correct Root bones to each sprite and disabling everything except the head, the character still didn't move.

It was clear this was not going to work and the problem that was run after this was the character not moving after the removal of the Sprite skin script on each property of the character. It was time to restart the project in case it would help, but the character still wasn't budging. Everything looked like it was in the rigidbody2D test, but the character wasn't functioning anymore. To try to

find this problem the next thing to eliminate was the packages installed. The Burst and Collections package removed and project restart the character still didn't work. The Animator was showing the character to be going through both of the animations which eliminated the problem being there.

7.8 End of the project

This Unity project could not move forward after this error. The project before trying out the soft-body technique could not be recovered due the backup being from the instance where the project had the packages installed. In a hindsight, it would have been ideal to backup the project or make GitHub for the project to better restore the project in case of errors like this.

There would most likely be solution to fix this issue in the state of the project it is in, but due to time constraints this kind of bug finding could not be possible to achieve in the time frame available efficiently. The coding part of this project could not be implemented at all in the project for the same reason.

This sort of conclusion to the project was disappointing, but it also gave very realistic perspective on what technical animator could run into in the game development process. It is realistic to not always take the right path in the development. Melker Samuel Berg, technical animator for Studio Gobo, brings this part to the light "Almost every problem in Technical Animation has multiple potential solutions" (Melker, 2020, What's the hardest thing about your role?) Melker further states that the decision you make as Technical Animator can seem like the best idea at first, but later on these decisions will cause problems to the project. It is however better to try things out right away than leave hanging on to find the most optimal solution.

7.9 Results (actual conclusions)

The pipeline of the 2D technical animator job from the skeletal animation perspective is full of different types of tasks. From more graphical side to the implementation to the code itself, based on this research it includes notes from multiple areas of knowledge. This makes the job itself harder since the knowledge base the worker needs to have should be from multiple fields.

The pipeline consists of multiple rigging based tasks like mesh, weights and bones. Some of these parts are application specific tasks and differ on the tools you use. It also consists of lots of problem solving and necessary decision on the path you take. Nothing is straight forward and the decisions you make might be crucial to the process of the whole animation implementation process to the game. Like in this case study, the decisions mattered and it ended up dooming the project. Bug fixing was big part of the project pipeline.

The technical animation pipeline in Unity with skeletal animation

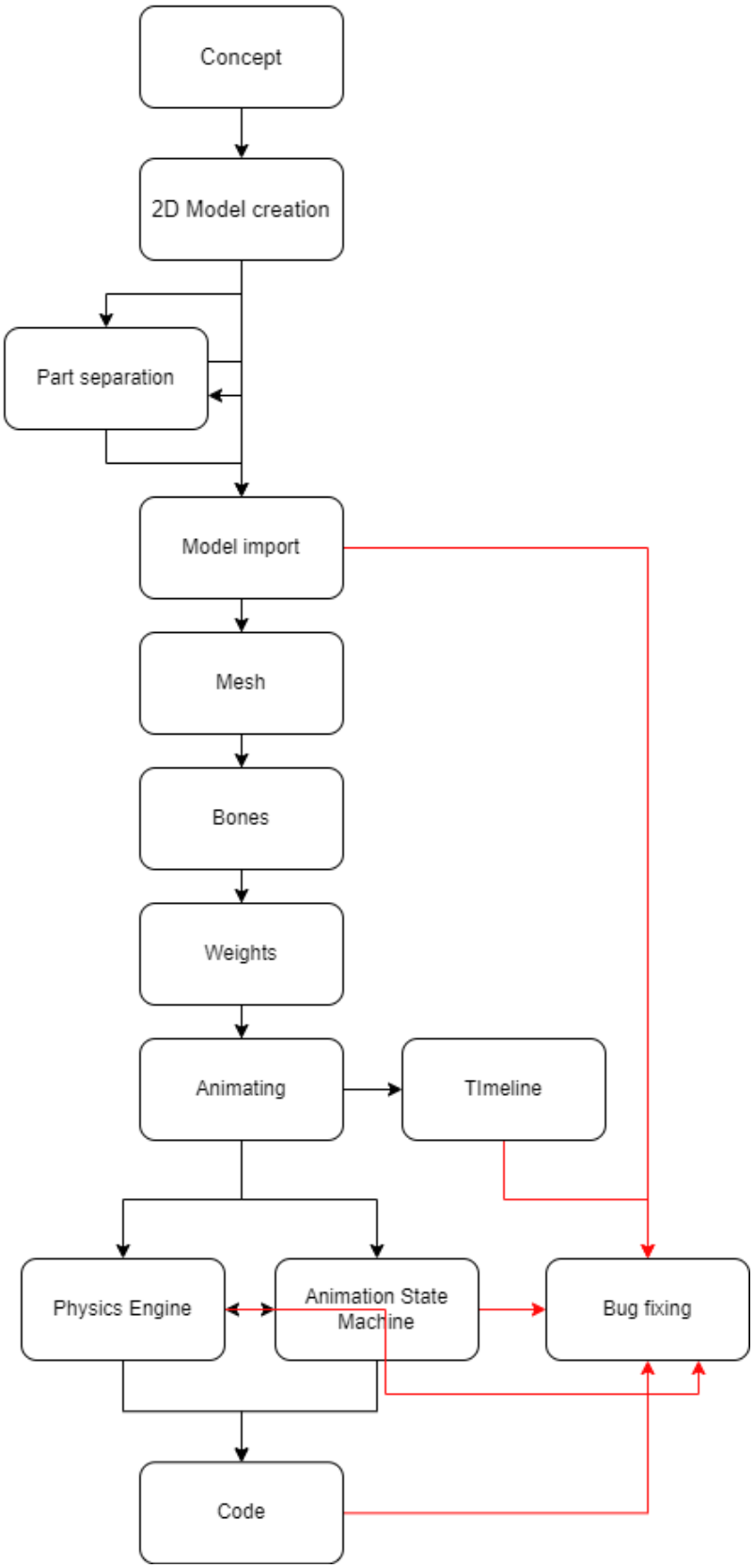

Figure 21 *Pipeline made based on the project*

The 3D side of the technical animation is much more similar in terms of pipeline than expected, but with differences in how to approach these steps in the process. The 3D and 2D graphics for games both can be rigged, have mesh, have weights and uses state machines to control the animations and their relationship with each other. The tools are the same kind but require different process to make and work with. In Unity Engine, which was examined more in the project, there are same tools made for the 3D and 2D development like Rigidbody and Rigidbody 2D that differ with their script and functions a bit but serve the same purpose.

8 Conclusion

Technical animator as job is more than just one position type that is cut in stone. It is more like title you put in the papers since the studio does not have other ideas for your title. Often they are found with some similarities with the titles tasks, and they do have jobs in the studio related to animation too, from rigging to programming animation tools themselves. Based on the contradicting opinions on the subject, it is hard to determine which one would be the right answer.

There is not direct path to get to this position which explains the lack of documentation and information online. The documentation that is around is hard to find and need to be found in smaller sizes. The most recent research is not 2D development oriented. The focus on the 3D side of the game development has taken bigger focus on game development going forward. This can be seen on the lack of documentation on the 2D, but the increasing amount of documentation on the 3D game development.

While this thesis was made to act as one step closer to find specific information on the 2D technical animation, the author has not had the experience on working on the topics with team and in bigger scale than professional themselves. Further documentation from someone with long experience on the topic, like person in 2D technical animation position, would be valuable to the future of this job. While there were some interviews found on the internet of people with Technical Animation title, more comprehensive research and documentation would be needed. This research would have greatly benefitted from interview from professional on the field or more realistic setting of the technical animation project, that was made.

8.1 Ethics and research

The research on the 2D technical animation from the perspective of programming animations tools in the engine and general programming related to the field lacked in this research and limited the view of it. While the topics were brought up in this research, the information on the side of it would be needed to further understand technical animation.

The research was done mostly using non academic sources due to the lack of proper documentation on the research subjects. The researcher chose each source used to gather information with critical thinking and academical assessment with ethical practices. The hard time to find specific or up to date information should be taken into consideration when looking at the reliability of this research.

It is also good to note that this research only has one case study of the topic. Only one project made in this research does not give justice to the difference of the pipelines this process could have. Multiple case studies from different environments should be conducted for the research of the subject.

References

Barnhart B. (2024, February 18). *The History of Computer Animation.* Linearity. https://www.linearity.io/blog/computer-animation/

Bondfield M. (2023). *Felix the Cat: 1920s cartoons*. National Film and Sound Archive of Australia. https://www.nfsa.gov.au/latest/100-years-felix-cat

Clip Studio. (2022), *Clip Studio Paint History*. Clip Studio. https://www.clipstudio.net/promotion/10th_anniversary/en/history

Cooper J. (2019). *Game Anim: Video Game Animation Explained*. CRC Press/Taylor & Francis Group.

Deepmotion. (2018, August 17). *Character Animation 101: Weighting Your Rig.* Deepmotion. https://blog.deepmotion.com/2018/08/17/character-animation-101-weighting-your-rig/

Deepmotion. (2018, February 28). *Character Animation 101: Creating a 2D Mesh*. Deepmotion. https://blog.deepmotion.com/2018/02/28/lesson-1-turning-your-character-into-a-2d-mesh/

Encyclopaedia Britannica. (n.d.). *J. Stuart Blackton.* Britannica. https://www.britannica.com/biography/J-Stuart-Blackton

Halpern J. (2019). *Developing 2D Games with Unity: Independent Game Programming with C#.* Apress.

intogames. (2020, November 4). *What does a Technical Animator in games do?.* intogames. https://intogames.org/news/how-do-you-become-a-technical-animator-for-videogames/

Johns R. and Semah B. (2024, January 18). *7 Best Programming Languages for Game Development in 2024*. Hackr.io. https://hackr.io/blog/best-programming-language-for-games

Johnston O. Thomas F. (1981). *The Illusion of Life; Disney Animation*. Abbeville Press.

Junferno. (2021, July 4). *The Art of Pure CSS* [Video]. YouTube. https://www.youtube.com/watch?v=wUQbchYY80U&ab_channel=Junferno

Kennedy, S. R. (2013). *How to Become a Video Game Artist : The Insider's Guide to Landing a Job in the Gaming World.* Watson-Guptill Publications.

Kira Omori Live2D. (2022, October 1). *how to draw & cut/separate a VTuber for Live2D - UTTER BEGINNERS [Clip Studio Paint*] [Video]. YouTube. https://youtu.be/NXx8xbBYFh4?si=9H1beWTq9r9MFQGR

Lee T. (2023, March 21). *Master These Important 3D Rigging Terms*. Academy of Animated Art. https://academyofanimatedart.com/master-these-important-3d-rigging-terms/

McGee V. (2023, November 8). *What Is Coding and What Is It Used For?.* Computer Science. https://www.computerscience.org/resources/what-is-coding-used-for/

Millington I. (2010). *Game Physics Engine Development: How to Build a Robust Commercial-Grade Physics Engine for your Game, Second Edition*. Taylor and Francis.

PyroPhysics. (2021, February 4). *How to Make Softbody Physics in Unity!* [Video]. YouTube. https://youtu.be/3avaX00MhYc?si=VPMovUqtGWQoiXX8

ScreenSkills. (n.d.). *Technical Animator*. ScreenSkills. https://www.screenskills.com/job-profiles/browse/games/animation/technical-animator/

Sergeev A. (2023, July 10) The Artistic Journey of a Technical Animator https://80.lv/articles/the-artistic-journey-of-a-technical-animator/

Tsung D. (2016, August 31). *Don't Re-invent Finite State Machines: How to Repurpose Unity's Animator.* Medium. https://medium.com/the-unity-developers-handbook/dont-re-invent-finite-state-machines-how-to-repurpose-unity-s-animator-7c6c421e5785

Unity Technologies. (2024, May 30). *Rigidbody 2D*. [Unity Manual page in Unity Documentation]. https://docs.unity3d.com/Manual/rigidbody2D.html

Unity Technologies. (2024, May 24). *The Animator Window* [Unity Manual page in Unity Documentation]. https://docs.unity3d.com/Manual/AnimatorWindow.html

Unity. (2018, October 10). *About Timeline.* Unity. https://docs.unity3d.com/Packages/com.unity.timeline@1.2/manual/index.html

Zikei M. (2023, October 7). *Step-by-Step Guide to Cutting Objects in 3D Modeling*. SelfCAD. https://www.selfcad.com/blog/cutting-objects-in-3d-modeling

Appendices

Appendix 1. Stickman Product

The project where the animation process was tested out.

Project: https://github.com/jerrykovanen0/Thesis

Figure 19: https://youtu.be/3Ujdg8eYL8A

Figure 20: https://youtu.be/3Ujdg8eYL8A?si=1IqZXLMvFXaguP21&t=2