CREATING A GAME SCENE IN UNITY

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Hämeenlinna University Centre
Degree Programme in Business Information Technology

Autumn 2020

Saku Eräniitty

Tietojenkäsittelyn koulutusohjelma
Hämeenlinnan korkeakoulukeskus

Tekijä	Saku Eräniitty	**Vuosi** 2020
Työn nimi	Creating a game scene in Unity	
Työn ohjaaja	Lasse Seppänen	

TIIVISTELMÄ

Tämän opinnäytetyön tarkoituksena oli luoda yksinkertainen peliskene ilman pelattavuutta. Peliskenellä oli tarkoitus esitellä lukijalle peliskenen luonnin eri vaiheita ja aiheita, sekä auttaa lukijaa valitsemaan mitkä aihealueet pelinkehityksessä ovat hänelle mieluisimpia. Tarkoituksena oli myös auttaa lukiaa pääsemään hyvään alkuun oman peliskenen luonnissa.

Pelinkehitys kattaa valtavan määrän aihealueita. Tässä opinnäytetyössä kerrotaan miten Unityssä luodaan uusi projekti ja asennetaan siihen tarvittavat liitännäiset parhaan grafiikan mahdollistamiseksi. Sen lisäksi selviää miten hahmon veistäminen, animointi ja luurakenteen luominen Blenderissä toimii. Lopuksi vielä näytetään, kuinka partikkelisysteemiä voi käyttää vaikkapa tulen tekemiseen ja kuinka taivas tehdään peli skeneen.

Työ ei ollu minkään yrityksen tilaama, vaan kirjoittajan täysin itse valitsema aihe ja se soveltuu parhaiten alotteleville pelinkehittäjille.

Avainsanat 3D Modeli, Animaatio, Rigging, Partikkeli Systeemi, Skybox

Sivut 41 sivua, joista liitteitä 2 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna, University Centre

Author	Saku Eräniitty	**Year** 2020
Subject	Creating a game in Unity	
Supervisor(s)	Lasse Seppänen	

ABSTRACT

The purpose of this thesis was to create a simple game scene without playability. The game scene was meant to introduce the reader to the different stages and topics of creating a game scene and to help the reader choose which topics in game development are the most interesting. The purpose was also to help the reader get off to a good start in creating their own game scene.

Game development covers a huge number of subjects. This thesis explains how to create a new project in Unity and install the necessary plugins to enable the best graphics. It also explains how character sculpting, animation, and rigging work in Blender. Finally, it shows how a particle system can be used to create for example fire, and how the sky is made for a game scene.

The work was not commissioned by any company, but a topic entirely chosen by the author himself and is best suited for beginning game developers.

Keywords 3D Model, Animation, Rigging, Particle system, Skybox

Pages 41 pages including appendices 2 pages

CONTENTS

1 INTRODUCTION

The game industry is a huge and constantly growing industry. The competition is hard on all of the numerous fields of game development. However, sometimes it can be hard for a game developer to identify which field of game development is the most tempting. That is what is the core purpose of this thesis, to help out the reader by briefly going over most of the fields of game development.

The author has been fascinated by video games his whole life and got recently introduced to game development in an exciting study of half a year abroad. In the study, he created an open-world game together with 23 other students. He took part in the programming department but got to closely follow the work of the other departments as well.

Other departments in the game development project were game design, art, and sound. The game design team designed the levels and added some effects and functionalities that did not require programming. The art team created all the art for the game except for the music. This art included, for example, 3D characters, other 3D models, textures, concept art, and logos. The sound team created all of the sounds and music for the game.

These fields and departments of game development may vary based on the game and especially based on the size of the development team.

Despite being part of the programming department, the author shortly after returning home from this journey installed the tools required and began practicing 3D modeling.

The thesis was not ordered by a client but was created for the author himself. The subject of the thesis was selected solely due to the interest and passion to learn more about game development.

Research questions:

- How are 3D models created for games?
- How is it possible to create a simple animation for a game with Blender?
- How is it possible to use C# code in a game scene?
- What is it possible to use particle systems for a game scene?

2 UNITY

A game engine is software created for building video games. It usually offers the basic features used in most games such as physics, rendering, and handling inputs. All this allows the developers to focus on creating their own unique features for their game instead.

Unity is the most popular game engine and game development platform excelling more specifically at mobile, low-end PC, augmented reality, and virtual reality (VR) games while its biggest rival, Unreal Engine is leading the market in more high-end games. Unreal Engine is also known for making their own games, such as Fortnite, which Unity does not do at all. Unity is a cross-platform game engine, which means that it can be used to create games for multiple platforms. In Unity's case, this covers 25 different platforms including all the most popular ones in the market. It is also used to create both, 2D and 3D games. (Takahashi, 2018)

In 2005 Unity was released by a company called Unity Technologies in Denmark. By 2006 Unity already received an award at Apple's 2006 Worldwide Developer's Conference as it was presented as the first-ever fully powered game engine for iPhone. (Smykil, 2006)

Past few years Unity has also been used increasingly in film production for making movies and cut scenes. 2017 added the Cinemachine feature gave the developers a nice variety of different camera types to choose from. These cameras were also great for creating films and require no code whatsoever.

In 2019 Unity did $542M in revenue up 42% from the year before. (Motschwiller, 2020) The same year more than half of the top 1 000 mobile games were powered by Unity and on average all mobile games powered by Unity combined were being downloaded 3 billion times every month. (Unity Technologies, 2020e)

2.1 Creating a new scene

To create a new project in Unity, first, the Unity Hub must be opened. In Unity Hub at the Projects tab, “New” must be selected to start a new project. Next to the “New” button is an arrow, where the version of Unity can be chosen for the project. For this project version, 2019.3.2f1 was used. When working in groups, it is good not to keep upgrading the Unity version constantly to avoid merge conflicts.

In the next window Unity asks for the name- and the location of the project. Here the template for the project is also defined. This project is going to be a 3D scene with High Definition Render Pipeline (HDRP), so the High Definition RP template could be chosen, which has the HDRP and Post Processing stack pre-installed, but for educational purposes, the 3D template was chosen for this thesis and then the HDRP and Post Processing stack was installed from Unity’s Package Manager.

HDRP is a scriptable render pipeline for more compatible platforms such as PC. It enhances the lights and helps the user create a project matching to a high graphical standard. (Unity Technologies, no date a)

Post Processing stack improves the visuals and helps in achieving the desired look for the game. It affects the camera’s image buffer before the image is shown on the screen. It is good to think of Post Processing as a special lens added to the virtual camera that offers different visual tuning to the footage. (Unity Technologies, no date b)

Package Manager is a window inside Unity where the user can manage the installed packages and install new ones. It is located in the top menu bar under Window and Package Manager.

2.2 Setting up High Definition Render Pipeline and Post Processing

A new project should now be created. The next step is to click on the Window tab at the top left corner and then click on the Package Manager. From Package Manager should be searched and installed the High Definition Render Pipeline (HDRP). When the installation is complete, next an HDRP Asset should be created to the Assets folder by right-clicking the Asset folder, then choosing Create, Rendering, and finally High Definition Render Pipeline Asset. Now the asset must be assigned to the project. To do this, the Edit menu must be opened from the top left corner of the screen. From Edit Project Settings should be chosen. In the Graphics tab of the Project Settings the HDRP asset can be dragged to the Rendering Pipeline Asset slot. It is also recommended to go to the Player tab and to Other Settings to switch Color Space to Linear. (Brackeys, 2019)

Color Space stands for the mathematical method used by Unity to calculate the lighting. The Linear option should give more fidelity to the lighting in most cases, but once there are some lights and textures in the scene, it is good to switch this around to see the difference, but at this point, it makes next to no difference to our scene. (Unity Technologies, 2020a)

HDRP uses its own purpose-built implementation for the Post Processing tool so it should not be installed separately when using HDRP since it would not be compatible with HDRP either. If however HDRP is not used, Post Processing can be installed from Package Manage just like HDRP was just installed above. This installation would install Post Processing Version 2. (Unity Technologies, no date b) (Unity Technologies, 2020b)

The game is now using the High Definition Render Pipeline and Post Processing tool. Since there are no visuals whatsoever in the scene right now, there is no point in tweaking any settings yet.

2.3 **Terrain**

In this section, a terrain is going to be created for the scene. The shaping of the terrain happens completely in Unity, but for creating the texture Krita and Blender will be used.

To shape a terrain, first, a terrain is needed. The terrain can be found by clicking the plus icon in the hierarchy. From this menu, any game object can be added to the game. The option to create a terrain is under the 3D Object selection.

When working with Unity it’s good to keep in mind that one unit in Unity indicates one meter. By default, Unity creates a 1 000m x 1000m terrain. One of the four corners of the terrain located in the zero position of the global axis. For some people, it might feel nicer to work on the terrain when the middle of the terrain is in the zero point instead. For this reason, in the Transform options in the Inspector of the new terrain game object, the X and Z positions can be set to -500. However, this is totally optional and comes down to personal preference.

One default sized terrain is usually plenty of room to work with, but if another terrain is wanted next to the first one, it is better to do it from the Inspector of the first terrain. In the Terrain tab, a Create Neighbor Terrains option can be found. In that option, the side where the new terrain is wanted just must be selected to create a new terrain. This option is great and easy to use even later in the game developing process since it takes into account the shapes and the textures of the original terrain. It smooths out the shapes and copies the base texture unlike a manually added whole new terrain would.

2.4 **Creating Fire**

There are several ways to create fire in either Blender or Unity. This chapter is going to explain how to do it in Unity by using the particle system. Creating fire with particles is one of the lighter options when it comes to performance. The particle system is also very easy to set up by just a few clicks. All that is needed really is just some kind of an image of fire to show as the particles.

To begin with, a Particle System object should be created by clicking on GameObject, Effects, and then selecting the Particle System. Next, a material for the particles is needed. Firstly, a location where the material is wanted to be saved must be defined anywhere inside the Assets folder. Once at the desired folder location, new material is created by right-clicking anywhere on the empty space in the folder, then selecting Create, and then Material. At this point, a fire asset is needed. Any picture of fire with a black background will do for now since it can easily be changed later. The fire asset must be saved somewhere in the Assets folder again. In general, it is good to create a separate folder for the materials, textures, etc. to keep everything nice and organized. Once the fire asset is in Unity, it must be dragged to the Base Map slot in the new material. The Base Map slot is located in the Inspector of a material under the Surface Inputs tab. When the fire asset is placed in the material, can the material simply be dragged to the particle system object. The particles are going to look awful, but no reason to worry since that will be fixed soon. (Vegas, 2015)

At this point, the particles should just look like the image of the fire added earlier. This problem is fixed by opening the Inspector window of the material and from the top, the Shader must be changed by selecting Legacy Shaders, Particles, and either Additive or Additive (soft). As the name suggests, Additive (soft) should give a softer result. (Vegas, 2015)

There are almost countless options when it comes to the particle system. The following is a list of important options for creating a fire. In the list can also be found suggestions for the values. It’s good to keep in mind that in this project these values were used to create a fire for a fireplace, so for different purposes, the values will surely vary and it’s good to play around with the values to find the perfect fit. These following options can be found in the Inspector of the object under the Particle System tab:

Table 1. Settings of a particle system in Unity

Setting	Description
Looping	Looping should always be turned on. This option keeps the particles coming indefinitely.
Start Lifetime	Start Lifetime defines how long each particle is going to show before disappearing. This is one of the options that can be used to adjust how long the flames grow. Around 5 should be a good value for this setting.
Start Speed	Start Speed defines the speed the particles travel in. Leaving this option low but Start Lifetime high can result in richer and fuller flames. 1.5 is a fine value for this.
Start Size	Start Size defines the size of the particles. Can be left at 1.
Start Rotation	Start Rotation defines the rotation of the particle when it first appears.
Flip Rotation	Flip Rotation makes some particles to appear with a different rotation. Setting Start Rotation to for example 60 (depends on what looks best with the fire asset being used) and Flip Rotation at around 0.5 should create nice randomness in the flames making the flames more realistic looking.
Start Color	Start Color basically defines the color of the flame. It is good to set this to a nice orange color again based on the flame texture. A good example is around hex: F38500. This setting can also be used to create different colored flames such as blue or green if needed.

Play On Awake	Play On Awake should be checked to automatically start the particle system when the scene is played.
Gravity Modifier	Gravity Modifier is a good option if the particles are wanted to slow down towards the end of their lifespan. It is good to be careful with this option since just 0.01 or 0.02 should be enough. This can also be used to speed up the particles but for this reason, it is usually better to use the Start Speed option.
Rate over Time	The emission tab must be checked to edit the value of Rate over Time, which defines the number of particles appearing each second. Around 20 is usually good for this option.
Shape	The shape tab also must be enabled to define the shape where the particles move in. For creating a fire usually to box or a cone is used. For creating this fireplace, the box shape was used. From the Scale settings of the shape, the size can be scaled to match exactly the size of the object that is on fire.
Size over Lifetime	Size over Lifetime lets the user assign the size progression of the particles over the lifespan of the particles. Assuming the particles are wanted to start big and then slowly fade away as they fly higher, the settings for this are easy. In figure 1. the left indicator (which indicates the start of the lifespan) should be dragged to the top left corner of the chart. The right indicator (which indicates the end of the lifespan) on the other hand should be dragged to the bottom right corner of the chart.
Rotation over Lifetime	Rotation over Lifetime simply rotates the particles over the lifespan. This should always be checked to create a more randomized and realistic look. Around 60 should be a suitable value for most fires.

Figure 1. A graph that shows the size over a lifetime of a particle

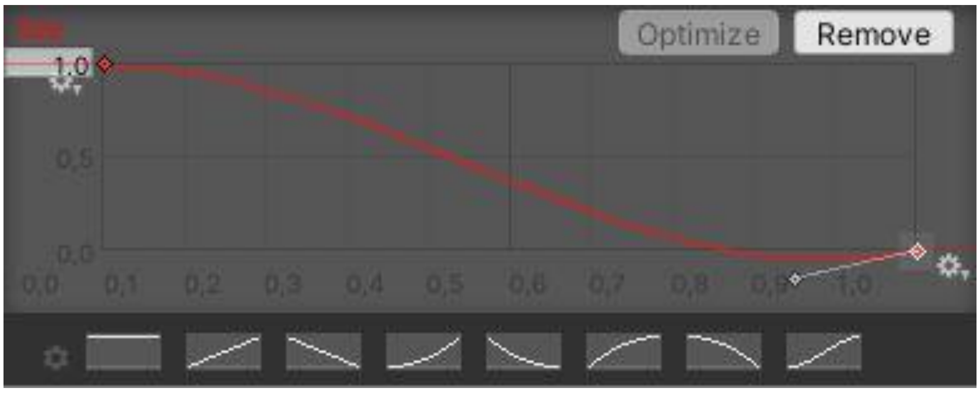


Table 1 covers the most important options for the particle system while creating a fire. With these options, one should be able to create a great looking fire, but all the options are surely worth checking out.

To finish the fire, one last thing should be added. A point light adds a nice final touch and makes it looks like the fire is lighting up the area around it with warm orange light. A point light is added as a child object to the particle system by right-clicking on the particle system and then selecting Point Light under the Light tab.

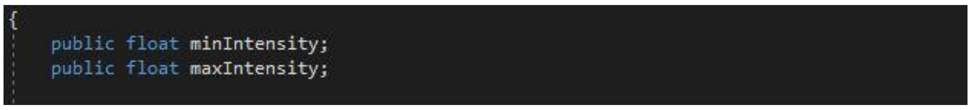
To set up the light correctly for this purpose, it is best to make sure that the X and Z positions are at 0 and then to lift the light on the Y-axis just slightly. Next under the Emission tab, it is important to change the color of the light to warm orange to match the fire. The range is also an important option to adjust to fit the use.

Since fire never remains exactly the same and flames keep bursting out in different sizes, it is only logical to have the point light also pulse in intensity. This can be created in a few simple lines of code.

Adding code to an object is done by clicking on the object (point light) and by scrolling down to the bottom in the Inspector of the object. At the bottom by clicking Add Component and New Script, the user gets to name the new script and then to add it by clicking Create and Add. The new script should now be found in the assets folder.

Next, the script must be opened in Visual Studio or any other integrated development environment (IDE) and the coding is ready to begin.

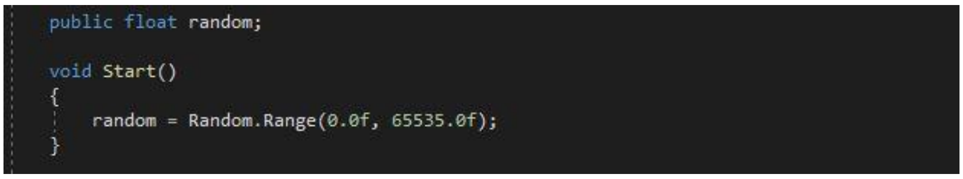
Figure 2. Maximum and minimum intensity values for the point light (Statement, 2011)



```
{
    public float minIntensity;
    public float maxIntensity;
```

The First two float values in figure 2 are needed to define the maximum and minimum intensity of the light. (Statement, 2011)

Figure 3. Random modifier for the point light (Statement, 2011)

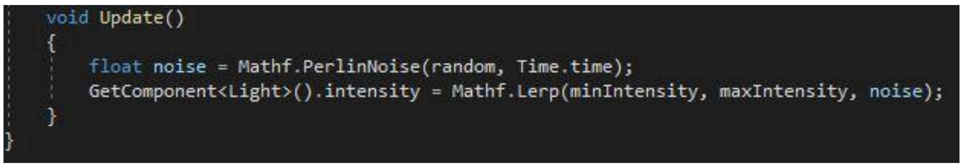


```
    public float random;

    void Start()
    {
        random = Random.Range(0.0f, 65535.0f);
    }
```

Next, it is good to create a random float value that is being calculated inside the void Start. This random value is used to calculate each light its slightly unique pulse so if there is more than one light with the same script in the scene, they do not pulse in sync. When the random value is calculated inside the void Start, as shown in figure 3, it means that the value is only calculated once at the start of running the script and so the random value remains the same. (Statement, 2011)

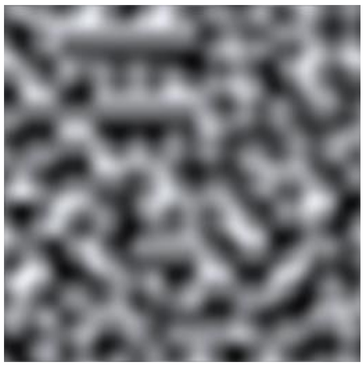
Figure 4. Void Update of the pulsing light function using PerlinNoise (Statement, 2011)



```
    void Update()
    {
        float noise = Mathf.PerlinNoise(random, Time.time);
        GetComponent<Light>().intensity = Mathf.Lerp(minIntensity, maxIntensity, noise);
    }
}
```

One more float value, float noise is needed to define the final intensity. Figure 4 shows how the value for noise is calculated by using Mathf.PerlinNoise, the random value calculated earlier, and Time.time value, which is basically just a value that comes from a running real-time timer that starts running when starting the scene. (Statement, 2011)

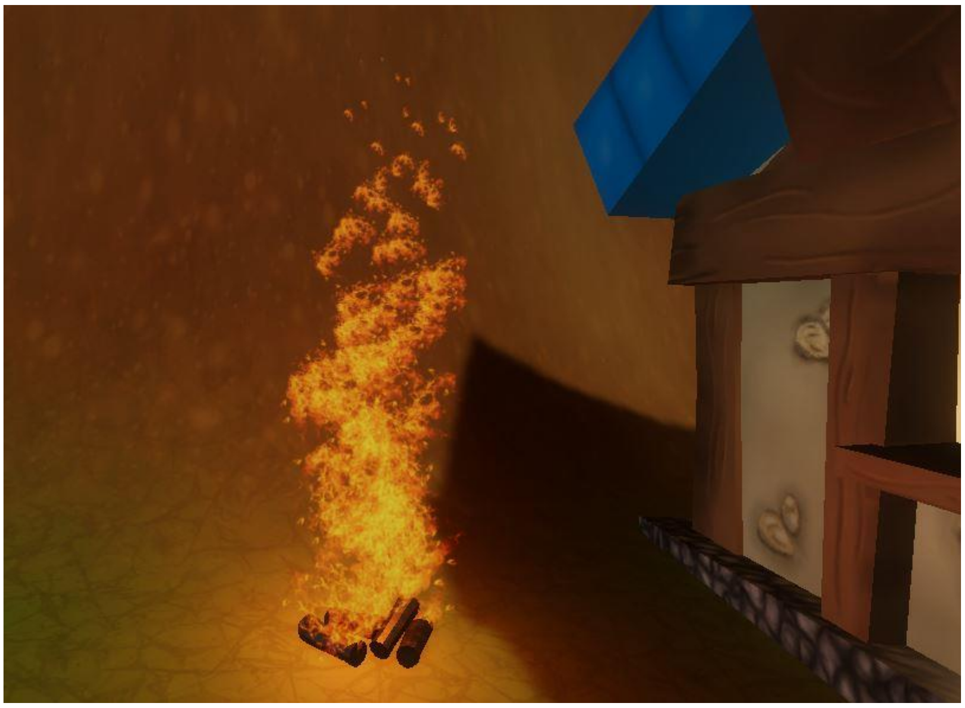
Figure 5. 2D noise map used in PerlinNoise function.



PerlinNoise returns a value based on a 2D plane such as figure 5 and the X and Y values (in this case the random and Time.time values) given to it. The white areas on the plane represent 0 value and the darker areas return a value all the way up to 1. The X and Y values define where on the 2D plane the value for the PerlinNoise is picked from. (Unity Technologies, 2020c)

Finally, the last line gets the light component’s intensity value and gives it the Lerp value calculated from the minimum intensity, maximum intensity, and noise values. How the Lerp works are that it calculates a value using the third value (in this case noise) to interpolate a value between the first and the second values (in this case the minimum intensity and the maximum intensity). (Unity Technologies, 2020d)

Figure 6. Result of the particle system created above.



The final fire should at this point be ready. With these options, it should look something like in figure 6 and have a pulsing light to it. The same technique can be used to create fire for a lot of different purposes by just adjusting the values.

3 BLENDER

Blender is one of the leading 3D modeling software. It is a community-driven tool where everyone can develop it or even help by writing documentation. Also, it is completely free. Blender supports 3D pipeline-modeling, rigging, animations, simulation, rendering, compositing, motion tracking video editing, game creation and it works on Linux, Windows, and Macintosh platforms. (Blender, no date)

In October 2018, Blender just released a major update, Blender 2.8. This update really took Blender to a whole new level with numerous new features and a whole new user interface. For this project, Blender 2.8 is used for making 3D models, sculpting, and animations.

3.1 **Sculpting**

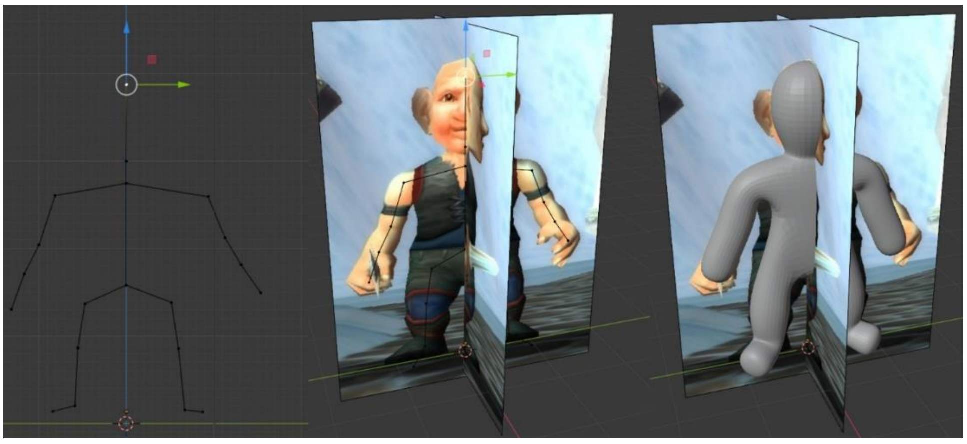
Modeling is perhaps what Blender is the most known for. Sculpting is also an essential part of modeling. Sculpting is mainly used for modeling characters like humans, animals, or other living creatures. This section is going to explain how to create models and sculpt characters. Later in the project animations will be created for the sculpted models.

3.1.1 Preparing a sculpt

To start on a new sculpt, once Blender is open, a new project is created by pressing File, then New, and then General in the top left corner of the screen. This opens an empty screen with just a cube in the middle. Now in the modeling tab, if the cube’s all vertices are selected and the project is in Edit Mode, by pressing key combination Alt and M and then from the pop-up menu selecting At Center, all the vertices should be merged to the middle. Now from the merged vertex in the middle can a base shape for the character be extruded by using the hotkey E. (Abbitt, 2018)

A good thing to do at this point is to add a reference picture to the background so it will be easier to make the base shape and later to sculpt up against the reference picture. To do this, while in the Layout tab, by pressing the key combination Shift and A, a pop-up menu will open again with the option to add an image. Under the Image option, it is better to choose a reference and then pick the wanted image from the PC. (Rios, 2020)

Figure 7. What first steps of preparing a sculpt looks like



Once the base shape of the character is ready, some volume can be added again from Modifier Properties. From Add Modifier -menu choosing Skin is going to give some volume to the shape and Subdivision Surface is going to make the shapes nice and round so they are going to be easier to work on right from the start. In the Subdivision Surface modifier, the settings can be adjusted if more or less detail is required. (Abbitt, 2018)

Now the thickness of the mesh can be scaled by selecting a vertex in Layout -tab’s Edit Mode and then pressing key combination Ctrl and A. For this make sure Wireframe from Viewport Shading is activated. The Viewport Shadings can be accessed by pressing Z -key or from the top right corner of the screen. Scaling the mesh can help you get closer to the final shape quicker. More vertices can also be added by selecting two vertices and then right-clicking on them and then selecting subdivide from the pop-up menu. When the scaling is done, the modifiers can be applied by pressing Apply on each modifier while in Object Mode. It is also good to apply the modifiers from top-down to avoid any errors. (Abbitt, 2018)

Lastly, if both sides of the mesh need to be identical, a new handy tool called Auto Mirror can be used for that. Auto Mirror can be activated from the Add-ons tab in Preferences, which is located in the Edit tab in the top left corner of the screen. Once it is activated, the Auto Mirror tool can be found in the side bar's edit tab. When mirroring the mesh, the mesh needs to be chosen and the settings must be set depending on which side of the mesh is wanted to be copied and on what axes. When all this is done, the mirroring happens by pressing AutoMirror –button and then by applying the modifier from Modifier Properties. Figure 7 describes these three steps of preparing a sculpt. (Lapineige, no date)

After these steps, the model should look like the one on the right-hand side.

3.1.2 Tools

Blender has a lot of tools for sculpting, but for this project, only the most important ones will be used. Those tools include the following sculpting brushes in table 2:

Table 2. Most important sculpting brushes in Blender

Grab	By holding and dragging from anywhere in the mesh pulls out or pushes in a part of the mesh. Depending on the Radius -setting can pull anything from a very sharp spike (low Radius -setting) to a wide area of the mesh (high Radius -setting).
Pose	A great brush for adjusting the posture of the model. For example, to move a whole arm or just to bend a little finger.

Flatten	Rubbing this brush to a surface quickly begins to flatten the surface.
Smooth	Rubbing this brush to a surface smooths out the surface. This brush is extremely handy and one of the most important brushes.
Crease	This brush creates a slit, which is great for the final touches on the model. Used for example to create bends on limbs, outline muscles, or add depth to the fine details of a face.
Blob	Inflates a round shape out of the mesh.
Inflate	Inflates any area of the mesh to create basic or more detailed shapes. This is hands down the most important and used brush in sculpting.

Most of the brushes can be adjusted in radius and strength. The opposite effect of a brush can be created by switching around the Direction -setting.

Outside the brushes, one of the most important tools for sculpting in Blender is Dyantopo. Dyantopo defines the amount of detail printed on the mesh when sculpting. In Detailing -setting the most popular options are Relative Detail, which changes the detail size based on brush size. Relative Detail is a great setting if the model needs more detail on some parts such as the face, and less on the others, like a leg. Constant detail is another popular option. With Constant detail, the user knows exactly how much detail is being added to the surface at all times.

3.1.3 Starting the Sculpting

When the preparation is done, begins the hardest and most time-consuming part of making a character model, sculpting. When going into sculpting, it is good to keep in mind what the model will be for and so to a suitable amount of detail in the mesh. If for example, the model will be for a game, too detailed of a mesh will punish the performance of the model and the game.

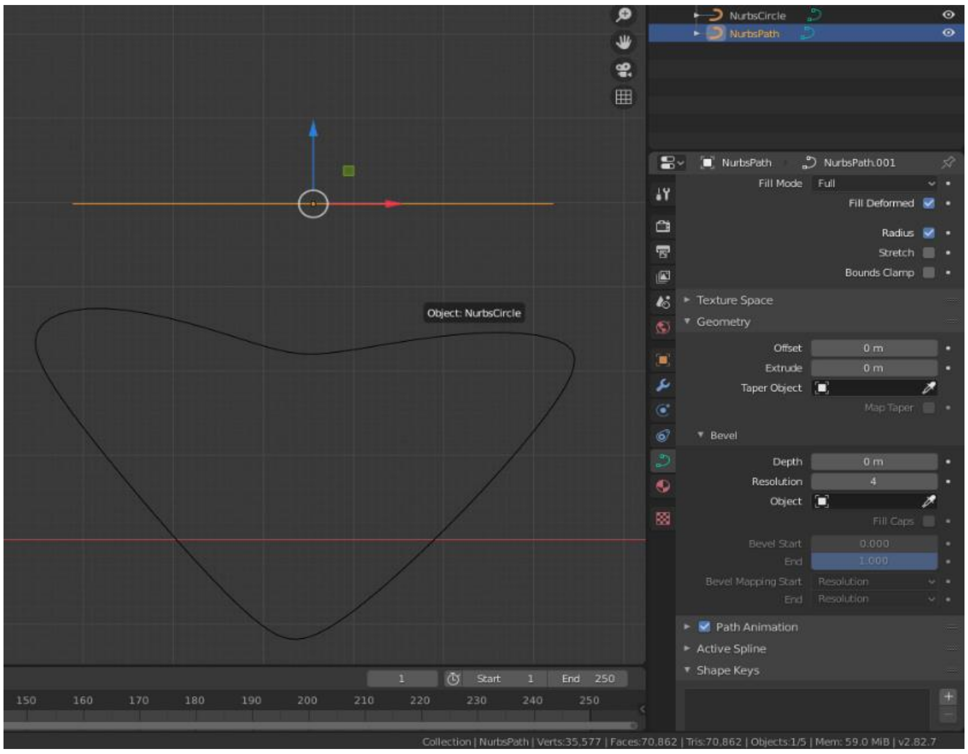
Possibly the best way to begin sculpting is to start dragging parts of the mesh in place with the Grab tool. Especially at the beginning of the sculpting, it is good to keep a reference image as a background to get the basic shapes of your character done. If the sculpture is getting too angular, the Smooth or Inflate tool with Dyntopo turned on smooths out the edges nicely and adds more resolution.

When basic shapes have been dragged in place, it is good to start using other tools. Inflate tool is often the next logical step. With inflate tools, the biggest muscles in the body can be created. After inflating the muscles, the Crease tool is great for outlining the muscles and other shapes in the body.

3.2 Fangs and Hair

There are many ways to create hair in Blender, but for this more cartoonish than realistic style in this project, the following method was used. This method is great and simple. It is easy to rotate and move separate batches of hair when using this method. This method is also very flexible and so can be used to create for example fangs.

Figure 8. Path Curve and a Circle Curve



NurbsPath › NurbsPath.001

Fill Mode: Full
Fill Deformed: ✓
Radius: ✓
Stretch
Bounds Clamp

Texture Space

Geometry
Offset: 0 m
Extrude: 0 m
Taper Object
Map Taper

Bevel
Depth: 0 m
Resolution: 4
Object
Fill Caps
Bevel Start: 0.000
End: 1.000
Bevel Mapping Start: Resolution
End: Resolution

✓ Path Animation
Active Spline
Shape Keys

Collection | NurbsPath | Verts:35.577 | Faces:70.862 | Tris:70.862 | Objects:1/5 | Mem: 59.0 MiB | v2.82.7

Firstly, two different curves are needed, a Path Curve and a Circle Curve. In figure 8 the curve on the top is a Path Curve and the one on the bottom is a Circle Curve. These can be created by pressing key combination Shit + A and selecting the wanted curves under the Curve option. The Path Curve's geometry must be linked to the Circle Curve by going to the Object Data Properties tab of the Path Curve. This can be done in Object Data Properties under Geometry and Bevel tabs by using the color picker tool. (YanSculpts, 2020)

The Path Curve should now be mimicking the shape from the Circle Curve. The shape can be changed by selecting the Circle Curve and by going to Edit Mode. In edit mode, the vertices of the curve can be moved or more vertices can be added by selecting multiple vertices, right-clicking on them, and then clicking Subdivide. (YanSculpts, 2020)

Next, it is good to close the ends of the hair to give it a more natural look. This can be done by now selecting the Path Curve instead and again going to the Edit Mode. In Edit Mode select each of the ends of the curve and scale them down with Alt + S key combination. When creating for example a fang, the other end of the curve might be better to leave open. Again, all the vertices on the curve can be scaled or moved and more vertices can be added by subdividing. (YanSculpts, 2020)

When the curve begins to look like an acceptable shape for an average hair, can the curve be moved in place and then be copied to make more hair with the key combination Shift + D. Since no hair is exactly like another nor positioned the same as another, it is good to remember to edit separate hairs to break any form of continuation in order to add a more natural look. (YanSculpts, 2020)

3.3 **Rigging and Animating**

To make a character animate, it must be rigged. Rigging is the action of creating a bone structure for a character. Blender has ready bone structures for the most popular models like a human and for a punch of animals too. These bone structures can be found under Armature in the Add menu. However, this segment will go over how to rig a character from the start.

3.3.1 Rigging

Rigging is started by first creating the bone structure. A bone can be created by my pressing the key combination Shift + A and then choosing Armature and Single Bone. Generally, it’s a good idea to place the fist bone as the spine and begin extruding the bones from there. To extrude a bone, the bone needs to be selected and Edit Mode must be activated. Once in Edit Mode, the end that is wanted to be extruded must be selected and then the bone can finally be extruded by pressing the E key. Bones can also be subdivided by right-clicking and selecting subdivide. (Gu, 2019)

When the Bone Structure is ready, it is time to parent the structure to the character by selecting the whole character and the bone structure and right-clicking to open the Object Context Menu. From Object Context Menu must the Parent and With Automatic Weights be selected. The bone structure should now be linked to the character. This can be tested by selecting the bone structure and then by going to Pose Mode from the top left corner of the screen. While in pose mode, if the bones are moved or rotated, the character should move and rotate with the bones. If the character or the clothes are not moving the desired way, that means Blender did not automatically set the weights right, so those must be edited manually.

Weight Paint defines how much of an impact a bone has on the painted mesh when the bone is being moved. Weight Painting can be done in the Weight Pain mode when first selecting the wanted mesh. Once in Weight Paint mode, in the Object Data Properties tab of the Properties window can be found the bones. Selecting each bone by just clicking on one will then display the weight painting of that particular bone on the mesh. In Weight Painting, a dark blue color stands for no effect at all from the selected bone and red color stands for the maximum amount of effect.

Weight Painting does not have nor need a lot of different fancy tools. The painting is done just by selecting the right bone and picking the suitable Brush Setting for the job. The rest is just painting as one would paint with any brush. One thing to note is that it is always good to fade out the weights and not just go straight from red to dark blue. If the weights are not faded out smoothly, the movement of the mesh is not going to be smooth and realistic either.

3.3.2 Animation

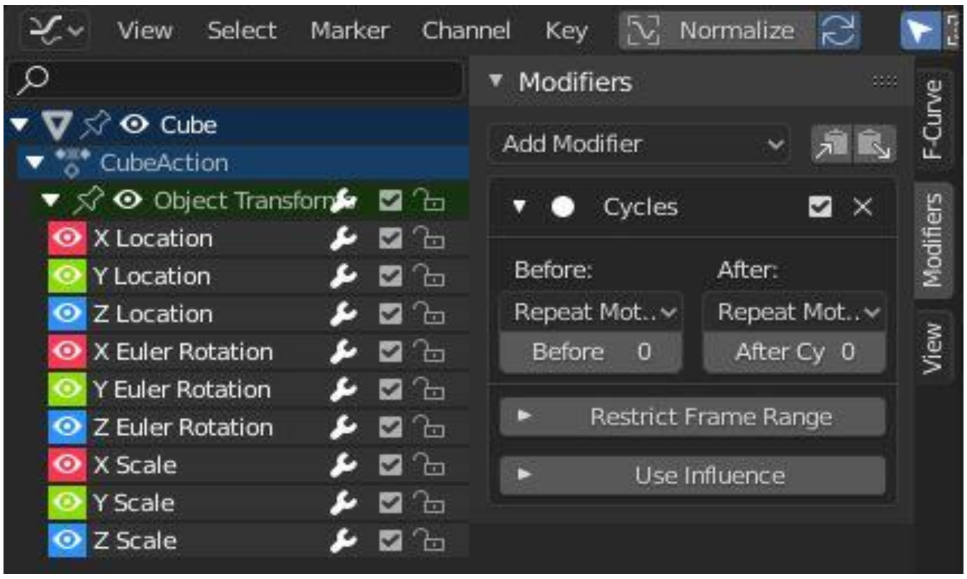
Once rigging is complete and the weights are right, can the animating be started. To start creating an animation, the bone structure must be selected, and the Animation tab has to be opened. In the Animation tab, an Auto Keying is good to be turned on in the Timeline window. Auto Keying automatically saves the movement done to the bones at a given frame. Once Auto Keying is on, can the bones be moved to create a pose for the chosen frame. The most logical and so the easiest way to start creating an animation is usually to start from the first frame. To start from the first frame, the timeline must be dragged to zero before making any changes.

Once the position for the first frame is finished, it will be automatically saved to that frame and the indicator on the timeline can be moved to the next frame where the next position is wanted to be defined. Blender will automatically move the pose from the first pose to the next. This means that not every frame has to be animated individually.

When looping an animation, the end of the animation just must be cropped to the final pose of the animation. If making for example a walking animation, the first and last poses should be the same and the end of the animation should be cropped to one frame before the last pose. So, if the last pose is at frame 30, the animation should be cropped to frame 29. This way the animation does not play the same pose for two frames in a row (the very first and the very last).

When creating an animation loop for just one object, an even easier way of doing so is as follows: first, the animation is created similarly as mentioned above. When the animation is ready to be turned into a loop, a Graph Editor window must be opened. In the Graph Editor window all the locations, rotations, and scales must be selected that are meant to be looped. At this point, by pressing N on the keyboard a small window for additional tools will open at the edge of the Graph Editor window. From the new window at the Modifiers tab, new modifiers can be added. To loop an animation, the Cycles modifier, such as the one shown in figure 9, must be selected. With Cycles modifier added, the animation loop should be complete.(rely, 2019)

Figure 9. Cycles modifier to loop the animation



Before: Repeat Mot..
After: Repeat Mot..
Before 0
After Cy 0

4 SKYBOX

This segment is going to explain in detail how to make a skybox for a game scene. In this process, three different programs are going to be used. First a program for painting the texture, then a program to make the texture sit nicely on a sphere shape as the skybox is. Finally, the texture will be taken to Unity and put to use.

A lot of time can be saved by downloading a ready sky texture not to mention a ready skybox material from for example unity Asset Store. However, for education purposes in this project, the skybox was created from scratch. To paint the texture, a painting program, such as Krita or Photoshop, is needed. For this project, Krita was the program of choice.

4.1 Krita

Krita is a free drawing and painting program. It is much like Photoshop and has all the basic image processing tool, but Krita is more painting oriented. Much like Blender, it is open-source and claims to be created by the artists for the artists.

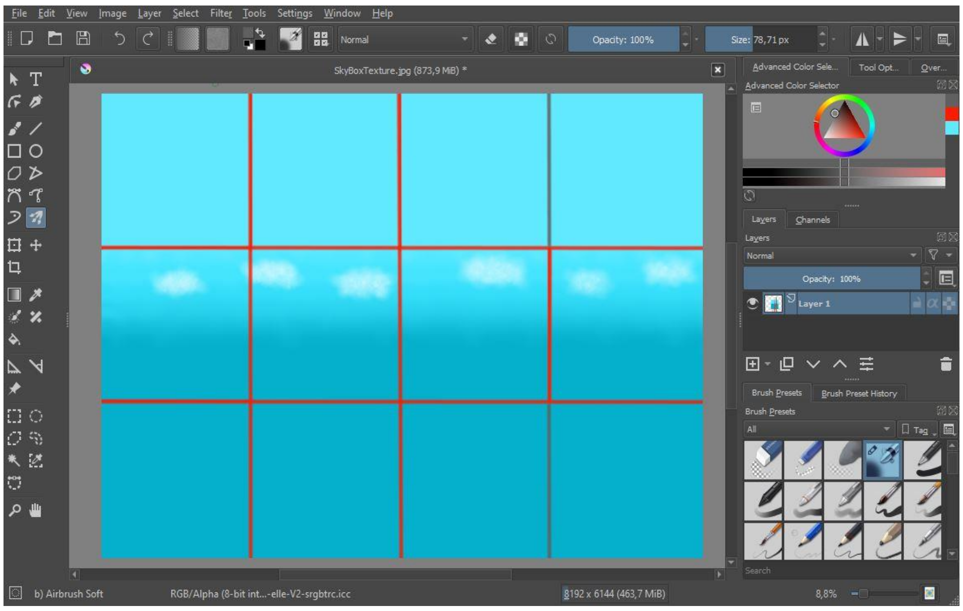
The first version of Krita was released in 2004. At first, it tried to be more of a generic image manipulation application like Photoshop or GIMP, but in 2009 the focus shifted more into making Krita the best painting application. More specifically for concept art, cartoonists and illustrators. (Krita, no date)

4.2 Painting the texture in Krita

When creating a skybox texture, it is good to create a big image due to the stretching the image is going to go true. For this project, the texture was 8 192 x 6 144 px. If an image of this size is too big for the use for any reason, can it be divided by two to keep the suitable measurements. This leaves the image with a size of 4 096 x 3 072 px. (Andrews, 2017)

When the picture is created, can any background layers be deleted, since only one layer is needed to create this texture. However, more layers can of course be used if needed for the wanted result.

Figure 10. Demonstration of which parts of the skybox texture will be used in Unity


File Edit View Image Layer Select Filter Tools Settings Window Help
Normal
Opacity: 100%
Size: 78,71 px
SkyBoxTexture.jpg (873,9 MiB) *
Advanced Color Sele... Tool Opt... Over...
Advanced Color Selector
Layers Channels
Layers
Normal
Opacity: 100%
Layer 1
Brush Presets Brush Preset History
Brush Presets
All
Tag
Search
b) Airbrush Soft
RGB/Alpha (8-bit int...-elle-V2-srgbtrc.icc
8192 x 6144 (463,7 MiB)
8,8%

The goal is to paint a sky much like in the picture. It is also good to keep in mind that when the image is being used in Unity, it will be split into 12 parts (4x3) and only 6 of it will be used. In figure 10 the 4 squares in the middle row will be the sides of the skybox, the second from left on the top row will be the top of the sky and the second from left on the bottom row will be the very bottom of the skybox. It is still good to paint a little over the lines or even the whole picture to avoid any seams when stretching the picture.(Andrews, 2017)

The most simple and easy way is just to make an ombre of blue colors progressing from dark blue to lighter. It is good to tap some of the blue colors in between with a brush to break the continual. Lastly, add some clouds by setting white color and low opacity to a brush and tapping that the desired way.

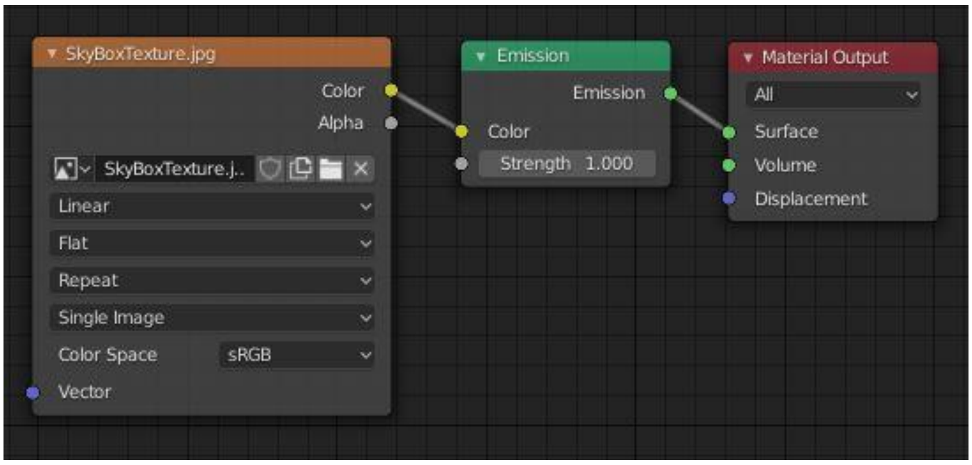
4.3 Matching the texture to a sphere in Blender

The texture is good to be previewed on a sphere skybox like shape. This can be done in Blender by following a few easy steps. As previewing the texture on the sphere, it can also be edited to add final details and fix any seams that might be wrong.

To do this, a new blender project must be created. In the project, there should be a cube by default, but if not, a new cube must be created.

On the new cube, the sky texture can be added right away. This can be done in the Shading tab by pressing the key combination shift + A to add a component. From the add component menu by selecting the first Texture and then Image Texture, a texture for the cube can be added. Next by clicking Open, can the right texture be searched for on the computer. Once the texture is in place, only one more component must be added. The texture must be shadeless and for that, an Emission component is needed. Emission component can be found again by clicking shift + A key combination and then by choosing Shader and then Emission. Once Emission is added, all components needed are in place and just need to be connected right. The connections go as follows: Color from the Image texture goes to Color of Emission. Emission from Emission goes to Surface of Material Output. If the Principled BSDF component was ever added by default, it can be deleted. The result should look like the one in figure 11.

Figure 11. Shading nodes of the skybox texture in Blender



At this point, it is good to make sure the cube is unwrapped right. UV Unwrapping happens at the UV Editing tab. The texture and the unwrapped cube can be viewed in the UV Editor window that should be opened by default on the right side of the screen when entering the UV Editing tab. However, to see the unwrapping over the texture, the whole cube must be selected in edit mode. Now the goal is to place the unwrap as planned in the earlier chapter ”Painting the Texture in Krita”. This can be done simply by manually selecting and then rotating and moving the unwrap on the UV Editor window. It is good to remember that if in Blender anything is wanted to be scaled one axis at a time, that can be done by first selecting the object, then pressing the S key for the scale tool, and then pressing X, Z, or Y depending on the axis wanted to be scaled.

For the next step, it is good to return to the Layout tab and make sure the Viewport Shading from the top right corner of the screen is set to Rendered to see the result closest to what it is going to look like in Unity. Next, while the cube selected in edit mode, the Normals of the faces must be turned inside out from Mesh, Normals, and then selecting Flip. To check if the Normals are facing inside, the Display Normals setting must be turned on from the Viewport Overlays menu at the top right corner of the screen. By default, the Display Normals setting shows the direction the Normals are facing with a short blue line. This line is easier to see in the Viewport Shading: Wireframe mode.

Now for the last step, a modifier must be added to the cube in the Modifier Properties tab called Subdivision Surface. On Subdivision Surface, the Viewport Setting should be increased to 6 to give it a nice and round shape. Now the texture is all set for final fixing and preview. This can be done by opening the Texture Paint tab and scrolling the view to inside the object. Painting tools and the smear tool can be used to make final touches. When the texture is ready, it can be saved normally and taken to Unity for use.

4.4 **Adding a skybox in Unity**

Adding a skybox to a scene in Unity is easy. Possibly even a little easier when not using HDRP but in this project, HDRP is used and that also makes adding a skybox a little different. However, rendering pipelines allow some additional settings for a skybox.

Adding the skybox can be started by first adding the texture to the wanted location in the Assets folder. In the textures Inspector window, the setting Texture Shape should be set to Cube. Always when adjusting texture settings, the Apply must be clicked to execute the changes. (Singh, 2018)

Next, a new empty object is needed. This object can be named SkyBox for clarity. In the object, a component and some overrides are required. First, add a Volume component by just clicking Add Component in the object's Inspector and by writing Volume in the search. Once the volume is added, can a new profile also be added by pressing New at the Profile part of the Volume. Unity will automatically name that profile SkyBox Profile if the object was named SkyBox.(Singh, 2018)

Next, some Overrides are needed to achieve the wanted result for the skybox. At this point, it is good to add the most important override, the HDRI Sky. On the override, all the settings can be enabled by checking all the boxes. At this point, the texture is finally required. The texture can be added simply by dragging and dropping it to the HDRI Sky slot. From this Override, the Rotation and Exposure of the skybox can also be adjusted, but those are better to fix soon when the skybox is visible. Usually, around 0.5 Exposure is good to start with. (Singh, 2018)

Now finally to make the skybox visible, a Visual Environment Override must be added. In Visual Environment Override, check the Type box and select the HDRI Sky option. Now the volume should finally use the HDRI Sky with the sky texture as a skybox.(Singh, 2018)

Additional nice overrides to add are for example the Fog Override. As the name suggests, fog can be created with this override. Fog can also be used to fade out the sharp edges of the terrain if that is needed.

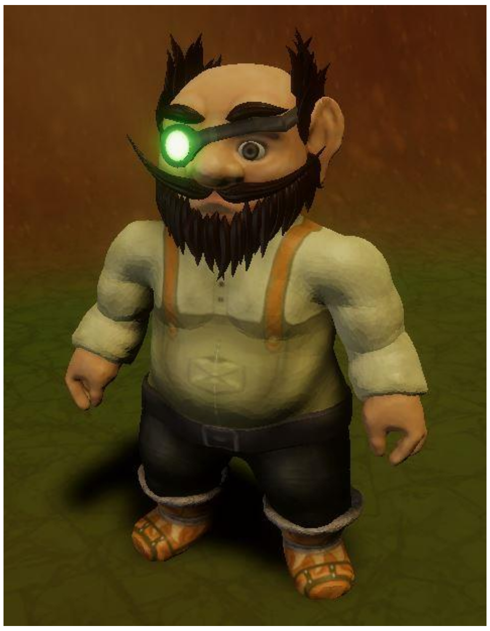
5 FINAL RESULT

This chapter showcases the outcome of the methods explained in this thesis. Some of the methods were used in multiple locations to create different results.

5.1 Characters

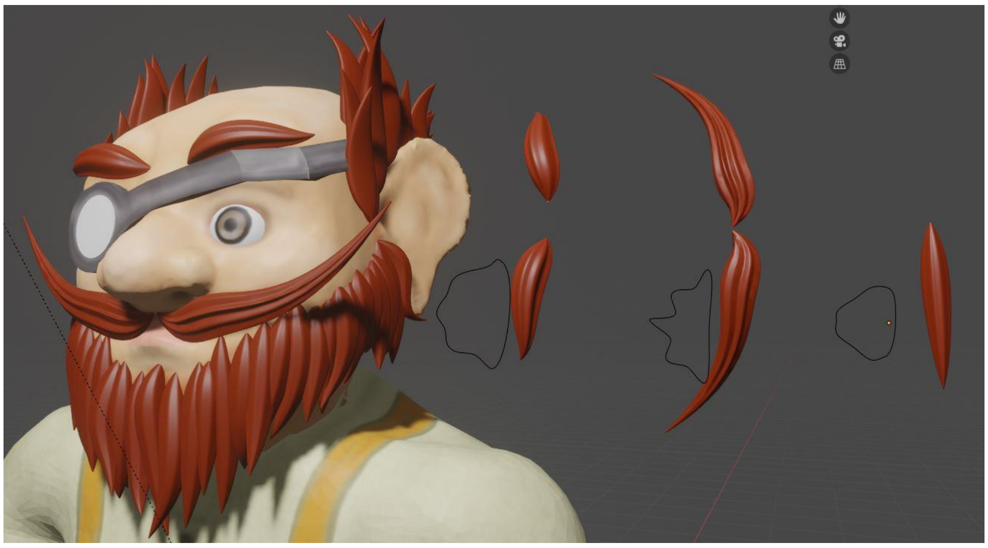
With the methods described in the sculpting chapter, two characters were sculpted for the scene. First, sculpt was the main character, the gnome shown in figure 12. When the character was brought to Unity, a green point light was added to its goggle to add a nice glow effect.

Figure 12. The final gnome character model in Unity with an added point light added to the goggle.



This character’s hair and beard turned out great as well using the path and circle curve method. Figure 13 shows all the hairs used on the character. The singular hair on the right-hand side was used to create the whole hair for the character by just copying and scaling it to different shapes and sizes over and over again. The clothes were sculpted as separate objects by copying the part of the model where the cloth is going to be, scaling it slightly bigger, and then sculpting it to give it the final shape.

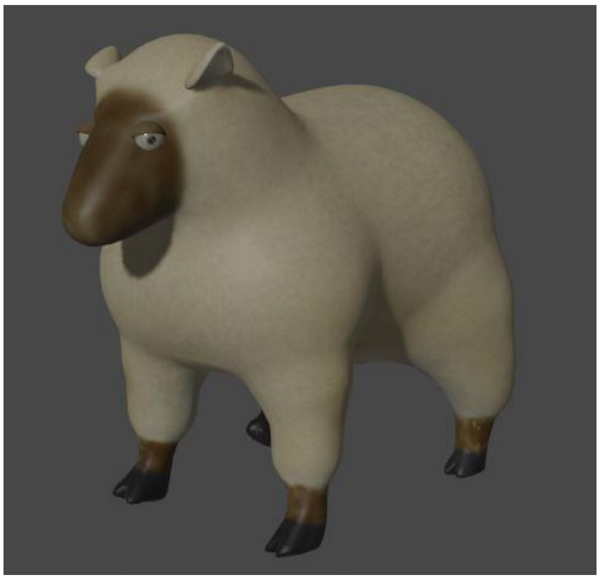
Figure 13. Gnome model’s path and circle curves.



Overall, the character turned out great. The head ended up with far more detail and for that reason does not really match the rest of the body as well as it could.

The sheep in figure 14 was the second sculpt for the scene. Both the characters were created with the same sculpting methods despite how different the models are.

Figure 14. The sculpted sheep model.



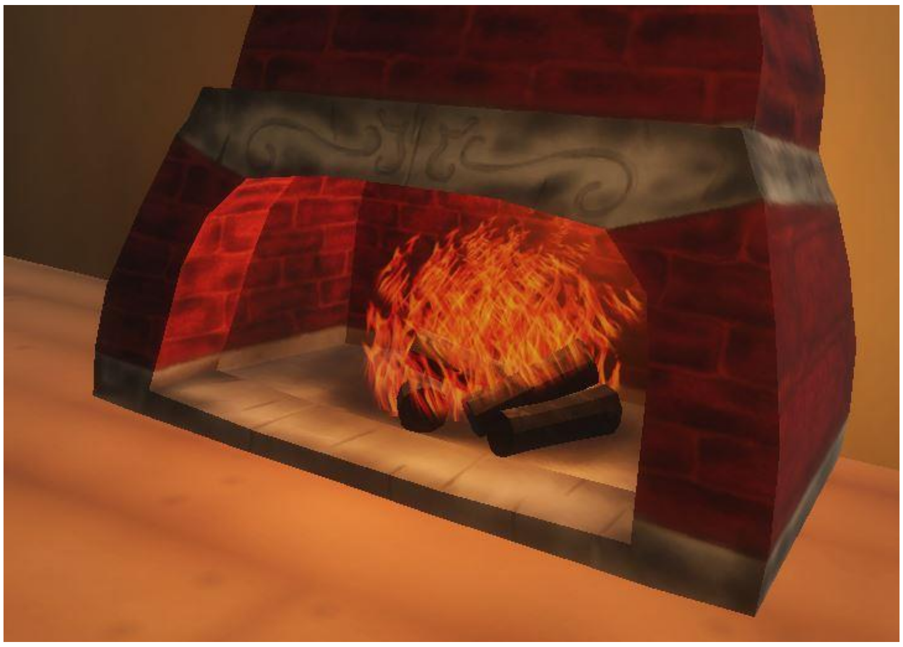
The sheep only had the wool texture painted to it, so no special methods were used for that. However, the wool part of the sheep could look better if it were not so smooth. Blender has a couple of ways to make a surface rugged which could be used for this purpose.

Both of the characters received a bone structure and were animated a walking animation. The great thing about rigging in Blender is that there are ready bone structures for a human and a few different kinds of animals. For the gnome, the bone structure was created manually by just extruding bone after bone. For the sheep actually, a bone structure of a horse was used. The ready bone structures in Blender are fairly complicated and have a lot of bones, so for this simple sheep, all the excessive bones were deleted to simplify the structure.

5.2 **Fires and Waterfalls**

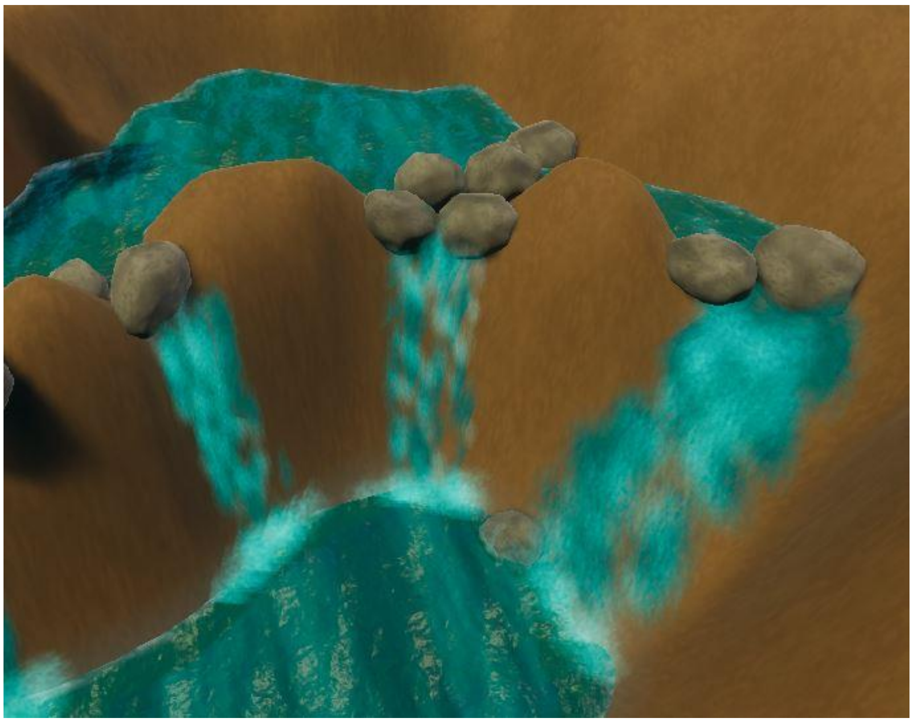
The particle system was greatly utilized in the project. As in the chapter before the particle system was used for creating a fire for multiple locations such as the one in figure 15.

Figure 15. Fire in a fireplace created with the Particle System.



Besides the fires, the Particle System was also created to create some waterfalls. The method between the fire and waterfalls is very similar. Just using a different texture for the particles and adjusting some settings such as the direction the particles move, made all the difference.

Figure 16. Waterfall made with the Particle System.

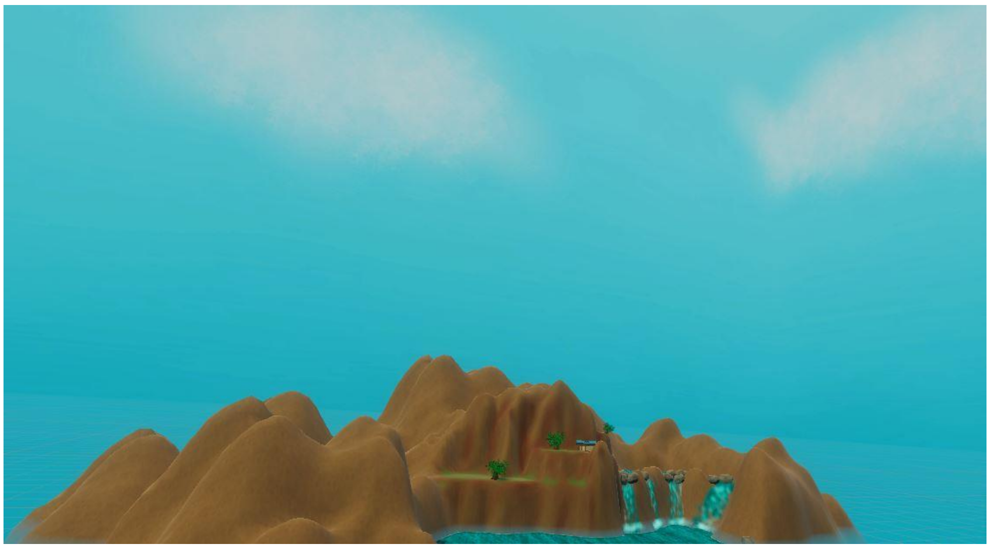


For creating the waterfalls, three different particle systems were used for each waterfall. The part where the water is falling has a darker layer of blue particles but also an almost white one to represent foam forming from the water. Additionally, at the very bottom, there is another near white particle system to blend the waterfall to the water more seamlessly with some more foam or fog looking particles. This particle system also rotates differently from the others. It only rotates the particles around and moves them slightly outwards from the middle of the particle system. Figure 16 shows the final result of these waterfalls.

5.3 Terrain and Skybox

As shown in figure 17, the result of the skybox and the terrain also turned out great and was a great base to build a game scene on. The skybox was created exactly as described in the thesis. Adding some fog from skybox settings definitely blended the skybox and the terrain together nicely.

Figure 17. Skybox and the terrain of the scene.



In this case for the terrain, it was found best to first go over the terrain with a large brush to shape out where the mountains were going to stand and roughly what size they were going to be. After that, some smaller hills were added with a smaller brush, and finally, the platforms were evened out with a smooth brush for the cabin house and water areas.

A total of three textures were created in Krita and then were moved over to Unity for painting the terrain.

6 SUMMARY

The goal of the thesis was to have a simple Unity 3D game scene with some effects and functions but no playability. The main purpose was to show the reader how different fields of game development work to help the reader identify the field closest to their own interest.

The final scene has a shaped and textured terrain and a skybox. Plenty of 3D models were made including two character models with some animations. Due to lack of time, the character models never received an AI for the movement to showcase the walking animations, but that is surely a subject for later development. Additionally, fire and waterfalls were created with particle systems. The scene is now also using High Definition Render Pipeline, which did not really have much use past just installing it in this thesis, but it can be later used for some high visual fidelity.

Overall, the thesis turned out great even though a lot of functionalities did not quite make it to the final scene. At the end of the day there are an endless number of fascinating functionalities so even if the author had twice the time to work on the thesis, there would still be plenty of more left to show. This thesis was but a touch on the surface of game development. However, hopefully, it gave the reader an idea of what game development can be like and most of all encouraged the reader to find out more and move forward in the arts of game development.

REFERENCES

Abbitt, G. (2018) *Quickly Create Base Meshes for Sculpting | Skin modifier | Blender 2.8 - YouTube*, *Abbitt, Grant*. Available at: https://www.youtube.com/watch?v=wCI8ZbTBP1w&t=263s (Accessed: 9 August 2020).

Andrews, K. (2017) *How To Make a Unity Skybox with Photoshop - YouTube*, *Kyle Andrews*. Available at: https://www.youtube.com/watch?v=-ZutidNYVRE (Accessed: 9 August 2020).

Blender (no date) *About — blender.org*, *Blender*. Available at: https://www.blender.org/about/ (Accessed: 9 August 2020).

Brackeys (2019) *How to get GOOD GRAPHICS - Upgrading to HDRP - YouTube*, *Brackeys*. Available at: https://www.youtube.com/watch?v=12gkcdLc77s&t=715s (Accessed: 9 August 2020).

Gu, D. (2019) *(1) Armatures - Blender 2.80 Fundamentals - YouTube*, *Blender*. Available at: https://www.youtube.com/watch?v=cZ3o5tjO51s (Accessed: 9 August 2020).

Krita (no date) *History | Krita*, *krita.org*. Available at: https://krita.org/en/about/history/ (Accessed: 15 October 2020).

Lapineige (no date) *Auto Mirror — Blender Manual*, *docs.blender.org*. Available at: https://docs.blender.org/manual/en/latest/addons/mesh/auto_mirror.html (Accessed: 9 August 2020).

Motschwiller, M. (2020) *Unity Software IPO | S-1 Breakdown*, *Meritech Capital*. Available at: https://www.meritechcapital.com/blog/unity-software-ipo-s-1-breakdown (Accessed: 13 October 2020).

rely (2019) *(1) HOW TO EASILY LOOP ANIMATIONS BASIC TUTORIAL | BLENDER 2.8 - YouTube*, *rely*. Available at: https://www.youtube.com/watch?v=QMrqtgZRWco (Accessed: 9 August 2020).

Rios, J. (2020) *Blender 2.8: Background Image – Simply Explained | All3DP*, *all3dp.com*. Available at: https://all3dp.com/2/blender-background-image-simply-explained/ (Accessed: 9 August 2020).

Singh, R. (2018) *How To Apply A Skybox In Unity 2018 (HDRP & LWRP) - YouTube*, *Rohit Singh*. Available at: https://www.youtube.com/watch?v=nlIIVPWD13I (Accessed: 9 August 2020).

Smykil, J. (2006) *Apple Design Award winners announced*, *Ars Technica*. Available at: https://arstechnica.com/gadgets/2006/08/4937/ (Accessed: 13 October 2020).

Statement (2011) *How to Randomly Change the Intensity of a Point Light with a Script - Unity Answers*, *answers.unity.com*. Available at: https://answers.unity.com/questions/41931/how-to-randomly-change-the-intensity-of-a-point-li.html (Accessed: 9 August 2020).

Takahashi, D. (2018) *John Riccitiello Q&A: How Unity CEO views Epic's Fortnite success | VentureBeat*, *Venturebeat*. Available at: https://venturebeat.com/2018/09/15/john-riccitiello-interview-how-unity-ceo-views-epics-fortnite-success/ (Accessed: 20 October 2020).

Unity Technologies (2020a) *Unity - Manual: Linear or gamma workflow*, *docs.unity3d.com*. Available at: https://docs.unity3d.com/Manual/LinearRendering-LinearOrGammaWorkflow.html (Accessed: 9 August 2020).

Unity Technologies (2020b) *Unity - Manual: Post-processing*, *https://docs.unity3d.com/*. Available at: https://docs.unity3d.com/Manual/PostProcessingOverview.html (Accessed: 9 August 2020).

Unity Technologies (2020c) *Unity - Scripting API: Mathf.PerlinNoise*, *docs.unity3d.com*. Available at: https://docs.unity3d.com/ScriptReference/Mathf.PerlinNoise.html (Accessed: 9 August 2020).

Unity Technologies (2020d) *Unity - Scripting API: Vector3.Lerp*, *docs.unity3d.com*. Available at: https://docs.unity3d.com/ScriptReference/Vector3.Lerp.html (Accessed: 9 August 2020).

Unity Technologies (2020e) *Welcome to Unity | Who are we*, *Unity.com*. Available at: https://unity.com/our-company (Accessed: 13 October 2020).

Unity Technologies (no date a) *High Definition Render Pipeline overview | High Definition RP | 7.1.8*, *docs.unity3d.com*. Available at: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@7.1/manual/index.html (Accessed: 9 August 2020).

Unity Technologies (no date b) *Post-processing in the High Definition Render Pipeline | High Definition RP | 7.2.1*, *https://docs.unity3d.com/*. Available at: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@7.2/manual/Post-Processing-Main.html (Accessed: 9 August 2020).

Vegas, J. (2015) *Mini Unity Tutorial - How To Create Fire - Beginner - YouTube*, *Jimmy Vegas*. Available at: https://www.youtube.com/watch?v=qShjsxopbfQ (Accessed: 9 August 2020).

YanSculpts (2020) *Easiest Way To Create Hair in Blender - 5 Minute Tutorial - YouTube*, *YanSculpts*. Available at: https://www.youtube.com/watch?v=BqWYgrXw7Jk (Accessed: 9 August 2020).