Tampereen ammattikorkeakoulu

Creating Okami Inspired Shader in Unity's Shader Graph

Aino Kuismin

BACHELOR'S THESIS
November 2020

Business Information Systems
Game Development

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Pelituotanto

KUISMIN, AINO:
Creating Okami Inspired Shader in Unity's Shader Graph

Opinnäytetyö 33 sivua
Marraskuu 2020

Opinnäytetyössä tutkitaan, kuinka Okami -pelin tunnusomainen grafiikka on tehty sekä mistä se on saanut inspiraationsa. Saman tyylistä grafiikkaa pyritään luomaan työkaluilla, joihin Tampereen ammattikorkeakoulun tietojenkäsittelyn tutkinto-ohjelmassa ja eritoten pelituotannon puolella tutustutaan opintojen aikana. Nämä työkalut ovat sellaisia, joihin jokaisella opiskelijalla on mahdollisuus päästä käsiksi vaivattomasti ja edullisesti.

Tavoitteena työssä oli luoda Okami-pelin visuaalista tyyliä jäljittelevä varjostin Shader Graphilla, Unity-pelimoottorin sisäänrakennetulla työkalulla, jolla voi luoda varjostimia ilman, että tarvitsee kirjoittaa riviäkään koodia. Okamille tyypillistä ovat mustat paksut eläväiset ääriviivat ja maalauksellinen ulkonäkö. Okamin visuaalinen ilme onkin ottanut vahvasti inspiraatiota japanilaisesta maalaustekniikasta nimeltä Sumi-e.

Opinnäytetyön teoriaosuus koostuu Okami-pelin esittelystä, sekä sen visuaalisen tyylin ja grafiikan tutkimisesta, hieman syvemmästä katsauksesta japanilaisen Sumi-e maalaustaiteen maailmaan ja Unity-pelimoottorista löytyvien varjostinmahdollisuuksien tutkimisesta. Työn käytännön osassa käydään läpi työvaiheet yksinkertaisen 3D-mallin luomisesta varjostimen tekemiseen ja lopuksi kaiken yhdistämiseen Unity-pelimoottorissa.

Tutkimuksessa selvisi, että ainakaan vielä tällä hetkellä Unityn Shader Graph -työkalulla ei ole mahdollista saada täysin haluttua lopputulosta. Vaikka lopputuloksessa ääriviivat olivat tyylitellyn näköiset, niissä on muutamia visuaalisia sekä suorituskykyyn vaikuttavia ongelmia. Visuaaliset ongelmat voitaisiin luultavasti helpoiten ratkaista käyttämällä 3D-mallissa yksinkertaisempaa geometriaa, mutta suorituskykyyn vaikuttavat seikat täytyisi selvittää muuttamalla varjostin prosessia kokonaan.

Asiasanat: visuaalinen tyyli, shader graph, sumi-e, peligrafiikka

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Game Development

KUISMIN, AINO
Creating Okami Inspired Shader in Unity’s Shader Graph

Bachelor's thesis 33 pages
November 2020

This thesis is a study of how the distinctive graphics of the video game Okami were created and where it gained its inspiration. The aim was to create graphics of the same style with tools that are introduced during the studies in the Degree Programme in Business Information Systems at Tampere University of Applied Sciences, and especially in the field of Game Development. Every student has an opportunity to use these tools effortlessly and inexpensively.

The objective was to create a shader that mimics the visual style of Okami with Shader Graph, a built-in tool in the Unity game engine that allows creating shaders without having to write a single line of code. Okami has characteristic black thick varying outlines and a painting-like appearance. Okami's graphics have indeed taken a strong inspiration from Japanese painting technique called Sumi-e.

The theorical part of this thesis consists of a presentation of Okami, as well as a study of its visual style and graphics, a little deeper survey into the world of Japanese Sumi-e painting and exploring the shading possibilities found in the Unity game engine. The practical part of this thesis goes through the workflow from creating a simple 3D model to making a shader in Shader Graph and finally combining everything in Unity.

The study revealed that, at least for the time being, it is not possible to create a completely desired end-result in Unity’s Shader Graph. Although the outlines looked stylized in the end, they have a few visual as well as performance related issues. Visual problems could probably be easily solved by using simpler geometry in the 3D model, but the performance issues should be tackled by completely changing the shader creation process.

Key words: visual style, shader Graph, sumi-e, game graphics

CONTENTS

TERMINOLOGY

CG	C for Graphics
HLSL	High Level Shading Language

1 INTRODUCTION

When the video game Okami was released in Finland, I was eleven years old. I do not clearly remember when I started to play the game, but what I do remember is how mesmerizing the story, the gameplay and especially the art were. The story and the characters captivated me playing for hours and hours, and because of the remastered version of the game they still do. I found myself really loving the art style of the game and constantly trying to mimic it in my own creations when I was younger. As to this day I am still wondering how it was possible to create a game so good from all aspects of a video game.

As a game artist student now in Tampere University of Applied Sciences I am obviously mostly interested in the graphics and the visuals of Okami. How were those developed? Which tools were used that time? And most importantly how could I achieve that same unique visual style in my 3D game art now with the equipment that I have easily on hand.

This thesis aims to be a study on how to achieve same looking 3D graphics as in the game Okami with the tools that any student can afford. Chapter 2 of this thesis explains what Okami is about. Chapter 3 studies the traditional Japanese Sumi-e painting style and its practices. Chapter 4 introduces shaders and tools for creating them in Unity game engine. Chapter 5 explains the workflow of creating an outline shader, similar as in Okami, for a simple low poly 3D model and implementing it all to Unity.

2 OKAMI AS A GAME

Okami is an action/adventure game set in ancient Japan. The player takes on the role of the sun goddess Amaterasu and sets on a journey. The purpose of that journey is to save the land of Nippon, which has been under great demonic powers for years. (Cowan 2006.) Amaterasu has taken a form of a white wolf whose fur is like clouds and covered in red celestial markings. On her back, she carries weapons that are inspired by traditional Japanese myths. The weapons can be equipped as a main or sub-weapon, and used in combat with melee attacks, which Amaterasu can learn as the player advances in the game. (Capcom-Clover Studio 2012.)

In addition to weapons, Amaterasu also has a special technique, a unique gesture-system called the Celestial Brush. The use of the Celestial Brush mimics the way of painting with an actual brush. The player can use the Celestial Brush to pause the game, bring up a canvas on the screen, and then draw different patterns on it. This feature is used to advance in the game, for example in combat, as Amaterasu is fighting with enemies or in puzzles that require for instance building a bridge over a river that otherwise blocks the way. (Roper 2012.)

Okami has often been compared to The Legend of Zelda games because of all the similarities that the two games have in common (Shepard 2018). There are many dungeons to explore, abilities to learn, and puzzles to solve, just like in the Legend of Zelda games. Like Okami, The Legend of Zelda: Wind Waker ignored the existing trend towards graphical realism and like Okami, The Legend of Zelda: Twilight Princess features a wolf as a playable character. (Totilo 2006). Although Okami is essentially very similar to The Legend of Zelda games, it has its very own unique approach to the genre.

Okami combines Japanese and other Eastern Asian mythologies, folklores, and art brilliantly. There are brisk believable characters who fit in the game world of Okami extremely well. All enemies are demon-like characters or creatures from different Japanese or other Eastern Asian folklores and beliefs. The interesting and vivid environments show aspects of ancient Japanese culture and history.

2.1 Different Okami versions

Okami was first released only for Sony's PlayStation 2 game console in 2006 in Japan and North America, and later in 2007 in Europe and Australia. The game was developed by Japanese independent game studio, Clover Studio, and published by Capcom. (Okami Wiki 2020). Okami suffered from poor sales when it was published, but despite that many critics and gamers still love the game. Okami won the #1 Best of 2006 award from Associated Press and IGN's Overall Game of the Year 2006 award, among other over 30 awards (Capcom 2017).

The same Okami version was released a few years later for the Nintendo Wii console in 2008 with motion controls of the Wii remote. An HD version of the game was released for PlayStation 3 in 2012 with PlayStation Move support. Even better version of the game with 4K support was released in 2017 for PlayStation 4, Xbox One and on the Steam platform for Windows PC. The most recent version of the game was released for the Nintendo Switch console in 2018. (Okami Wiki 2020).

2.2 Okami's game graphics

Okami has its own very distinct non-photorealistic rendering style. Graphics are cel shaded and heavily inspired by Japanese Sumi-e paintings. The game has a parchment like screen overlay and all the materials have hand-painted looking textures. Most of the assets in the game, both 2D and 3D, have a thick, very much varying, black outline, which imitates the brush strokes in the real-life Sumi-e paintings (Picture 1).

PICTURE 1: Thick varying black outlines and hand-painted textures make Okami appear like a painting. (Okami 2006)

At the time when Okami was developed the hardware was nothing like nowadays. Regular pixel shaders, that dictate the characteristics of a single pixel, were not used in the development. For example, the black outlines that give the most distinct look for the game were most likely done by reversing the winding direction of the polygons and drawing the object twice. The reversed side of the polygon stays visible because of the back-face culling, a method in computer graphics programming, which determines whether a polygon of a graphical object is visible. (Kaldaien 2017).

Because the outlines are not parallel to the object's geometry, AuraTummyache (2017) believes that the developers adjusted the normals a bit to achieve the hand-painted look for the outlines. In 3D modelling, a normal is a direction or a line that is orthogonal to the objects surface. The developers of Okami most likely had a script that was running when exporting the models. The script possibly added a random value to each normal between for example +0.05 and -0.05. This resulted in that varying effect that the outlines seem to have. (AuraTummyache 2017.)

It is important to note that nowadays making the outlines with the method described before would affect the performance of a game negatively, especially when having multiple high poly objects visible on the screen at the same time. Currently one of the better and common ways is to, for example, use screen space shader in post-processing. The shader detects the edges of an image by first drawing the outlines with the depth and normals. After that the shader calculates the viewing direction of the object. (Ross 2019). This way the shader draws nice and clean looking outlines as a result.

3 WHAT IS SUMI-E

The roots of the Japanese Sumi-e art style go way back in history and the technique holds many different aspects and practices. The history and manners behind the technique were found very interesting and are studied in this part of the thesis.

Sumi-e is a Japanese word for Black Ink Painting. Sumi-e is originally a painting technique in which the painter uses only black ink or some different shades of grey by varying the brush pressure and toning down the ink density. (Picture 2). Sumi means black and the -e refers to a painting. Sumi-e paintings history and roots go back to China all the way to the T'ang dynasty (618-907). Sumi-e technique arrived in Japan among the Zen-monks and in Japan the technique developed and gained new traits. (Karma 2013.)

PICTURE 2: A good example of a Sumi-e painting with only black ink and variety of different brush strokes. (Sato 2010)

3.1 How Sumi-e paintings are done

Sumi-e artists use tools that are called “the Four Treasures”. These tools include an ink stick and stone, a brush, and a paper. (Picture 3). Painting in Sumi-e style is not just about tracing the appearance of the subject, it is about capturing its deepest essence. The artists express their mind and spirit through “the Four Treasures”. (Shannon 2013).

PICTURE 3: The tools that are called “The Four Treasures”. (xb100 2016)

Ink sticks and ink stones were invented over two thousand years ago. Carbon soot was mixed with glue and it was pressed into solid sticks that were left to dry. The ink stick was grinded on the ink stone with water. The stones had a carved area in which the grinded ink gathered. The ink is permanent and still looks fresh on paintings from over 1000 years ago. (Jaranson n.b.). Nowadays many artists still use this ink grinding method for warming up the hands and preparing their minds for painting. Preprepared liquid ink, which is cheaper, is also available.

Brushes are handmade using natural materials like various animal hairs. The brushes can be divided into three basic types by their colour and usage. Brown

haired brushes are stiffer and have resilient bristles that retain a sharp pointed tip when painting. White haired brushes are soft and pliable. Mixed-hair brushes are a combination of the two. (Jaranson n.b.)

Paper, typically called ‘rice-paper’ in the western countries, is a generic term for many forms of handmade Asian paper. It is usually thin paper made from mulberry bark, Xuan or Pi paper in China and Washi paper in Japan. (Shannon 2013). Finished painting is usually pasted onto another piece of paper to support and protect it. The other piece of paper smoothens and flattens the actual painting removing wrinkles and fixing accidental rips. (Jaranson n.b.)

The Sumi-e artists must learn to use brushes and ink with carefully controlled strokes. The paper used in painting is highly absorbent and that for very unforgiving. The brushstrokes should be very fluid and fast because mistakes can hardly be corrected. (Shannon 2013).

3.2 Painting Themes

A great Sumi-e painting is not ever evaluated by the photographic likeness, but by the brush strokes and their ability to capture the essence of the subject. The painting themes were often related to nature, but nowadays the artists portray other subjects as well. (Jaranson n.d.).

Because nature is the most used theme among the Sumi-e artist, beginners often start practicing by painting “The Four Gentlemen”. “The Four Gentlemen” refer to subjects of nature that include all the basic brush strokes. These subjects are usually bamboo, chrysanthemum, plum blossom and orchid that all represent different seasons as well. Other flowers and plants are popular subject to paint as well, usually with birds. Landscapes and animals are also painted often as well as people’s figures. These can all be presented in just a few descriptive strokes or be painted in more precise detail. (Jaranson n.d.).

4 UNITY'S SHADER TOOLS

Unity is a 3D and 2D game engine created by Unity Technologies. Unity has two different approaches to shading, by writing shader code or by using visual shader editing tools, like Unity's built-in Shader Graph tool. Unity provides some standard shaders that can be modified and used in all of Unity's rendering pipelines. (Hasu 2018). There are also third-party tools available for shader creation, but the focus in this thesis is on the Shader Graph tool.

4.1 Different shader types and writing the shader code

Shading is one of the most important aspects in game development process. Shaders define the final look of the game by modifying how the texture and lightning are displayed. In Unity most of the shaders are written in descriptive ShaderLab language. The actual shader code is however written in CG or HLSL shading languages. (Hasu 2018).

Unity supports three different types of shaders: surface shaders, vertex and fragment shaders and fixed function shaders (Unity User Manual 2020). The surface shader helps to generates code, which makes it easier to write shaders that interact with lighting and shadow. Fragment shaders output a colour of a single pixel by calculating its position on a polygon. Vertex shaders on the other hand compute the vertices position in the image. (Hocking 2020).

There are also more complex shaders that can change the base geometry of a mesh. These are called geometry and tessellation shaders. Both shaders can be used to adapt the visual quality to the required level of detail by optimizing the mesh in real-time depending on its distance to the camera. (Hasu 2018).

4.2 Shader Graph

Unity's node-based Shader Graph tool has been available to use since the Unity version of 2018.1. Shader Graph tool lets the user build their own shaders visually instead of writing the code. Nodes can be created and connected inside

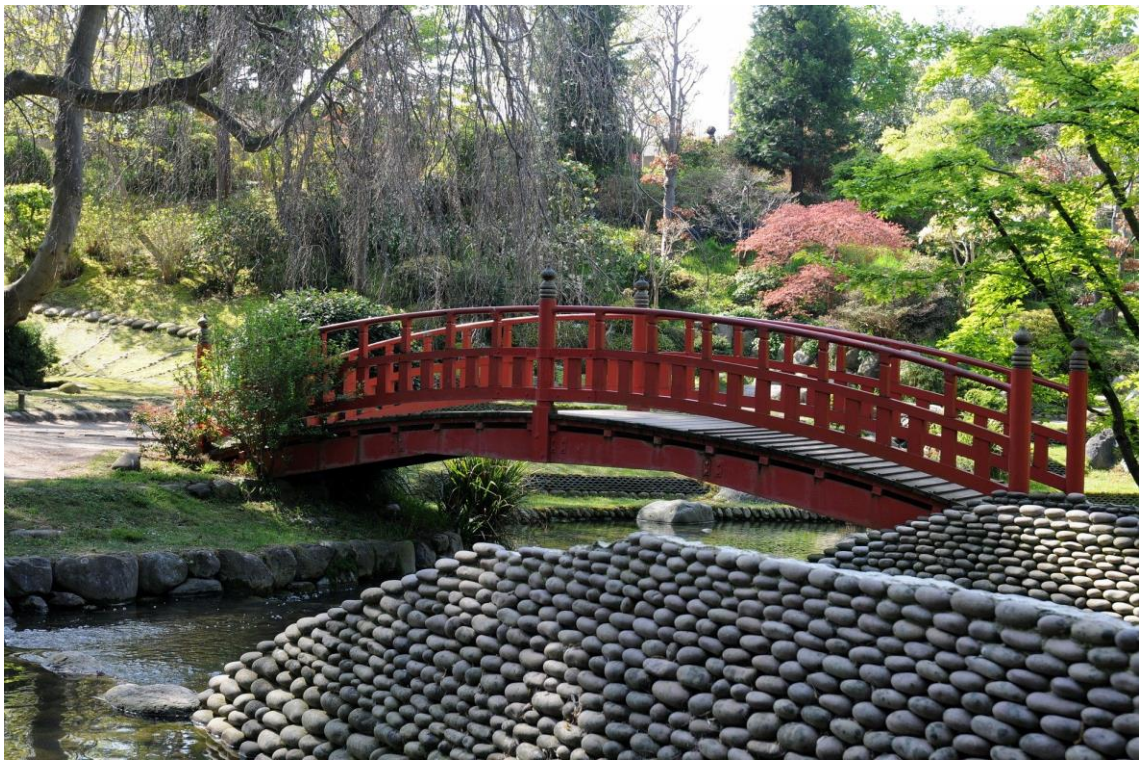
a graph network. The graph network gives instant visual feedback of the changes, and it is simple enough that even developers new to shader creation can learn it quite quickly. (Cooper 2018).

Nodes in Shader Graph are basic building blocks that receive inputs and outputs. Nodes are connected to each other and then joined in Master Node. Currently there are two different master node options available. Unlit Master Node that is not affected by lights and PBR master node that has full lightning support. The nodes have ports that have a data type, which defines the edge type that can be connected to the node. Edges are the connections between two ports. Only one edge connection can be in the input port, but output port can receive multiple edge connections. (Hasu 2018).

5 CREATION PROCESS

This part of the thesis includes a briefly described workflow of making a low poly 3D model, creating textures, implementation to Unity Project, and the shader creation process. The software used for 3D modelling in this project was Blender. Textures were created with Adobe Photoshop. The Shader was done with Unity's Shader Graph tool.

Before starting the process, even when modelling just a simple low poly model, it is useful to search and look at reference pictures. Sketching concepts based on the references, on paper or in digital painting software, helps at the start of the process. Sketching makes it easier to understand right proportions of the object and notice important details that should be included in the model to make it more believable. (Karon 2019) A Traditional Japanese bridge was chosen for this purpose because it goes with the theme of this thesis and although it is quite simple object, it is interesting enough to look at. (Picture 4)

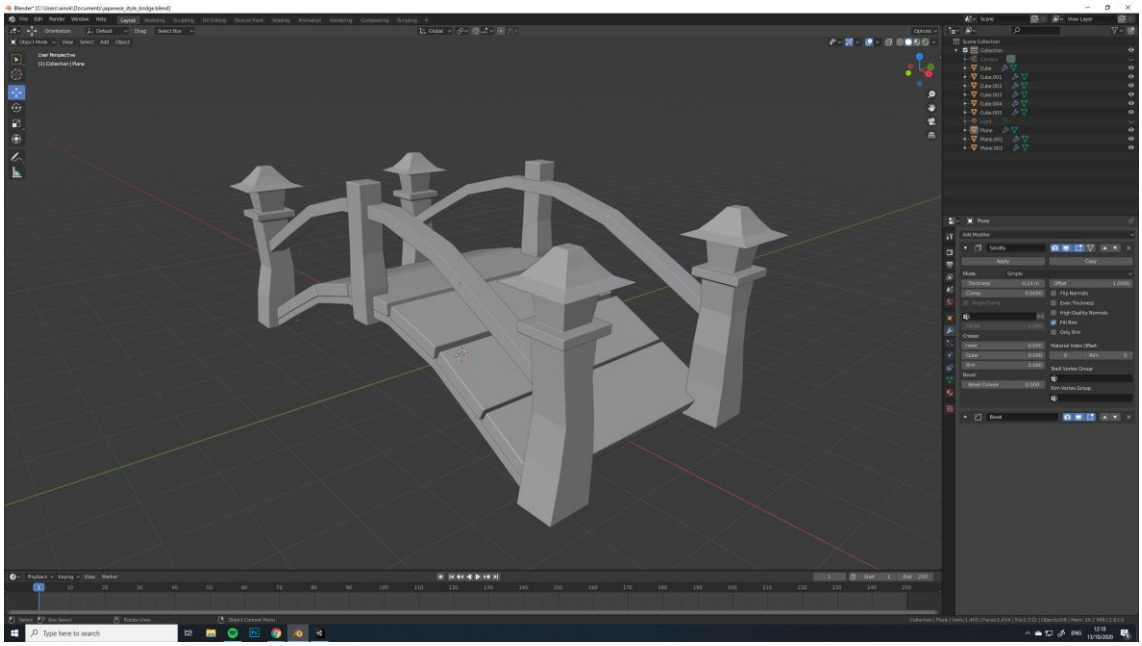


PICTURE 4: Japanese bridge chosen as a reference picture for the 3D model. (DenisDoukhan, Pixabay 2010)

5.1 Creating the 3D mesh

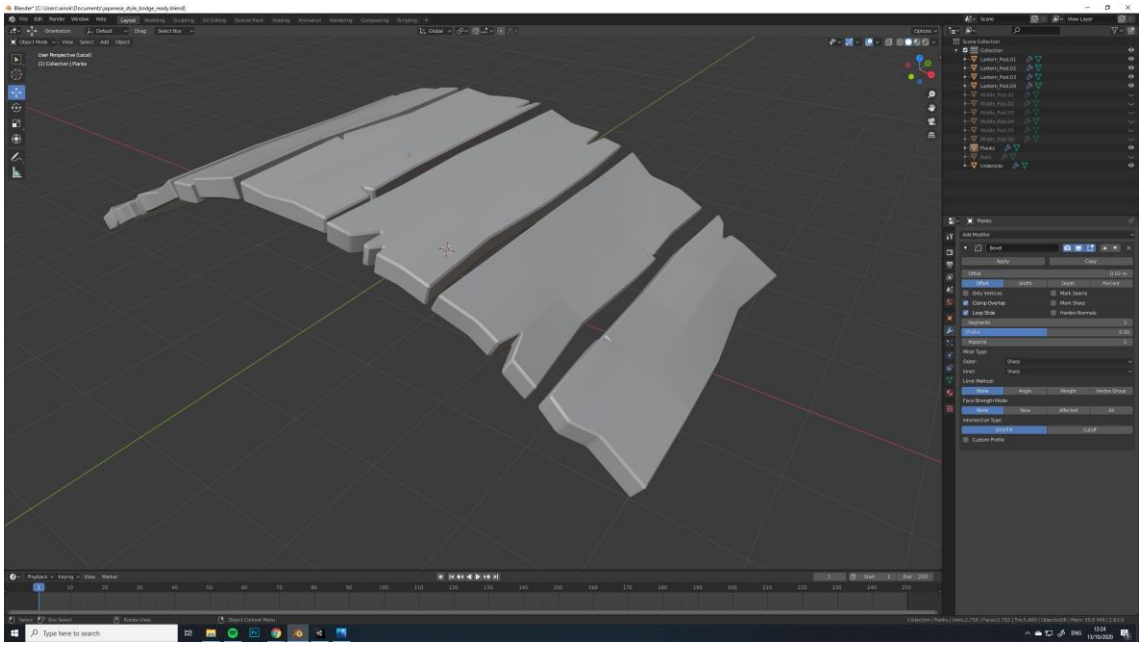
After the core idea of the 3D object was clear, it was time to move on to the actual modelling of the 3D mesh. 3D mesh consists of vertices, edges, and faces that together define the shape of the 3D object. (Petty n.b.)

Modelling process started with blocking out just the general big shapes of the object, keeping the mesh simple and the polygonal count as low as possible. (Picture 5). Extruding method and couple of different mesh modifiers were mostly used in this part of the process. Duplicates of the same 3D meshes were used with a little bit of variation in the scales and rotations. Creating duplicates of a one object mesh is much faster than modelling every single similar object one by one.



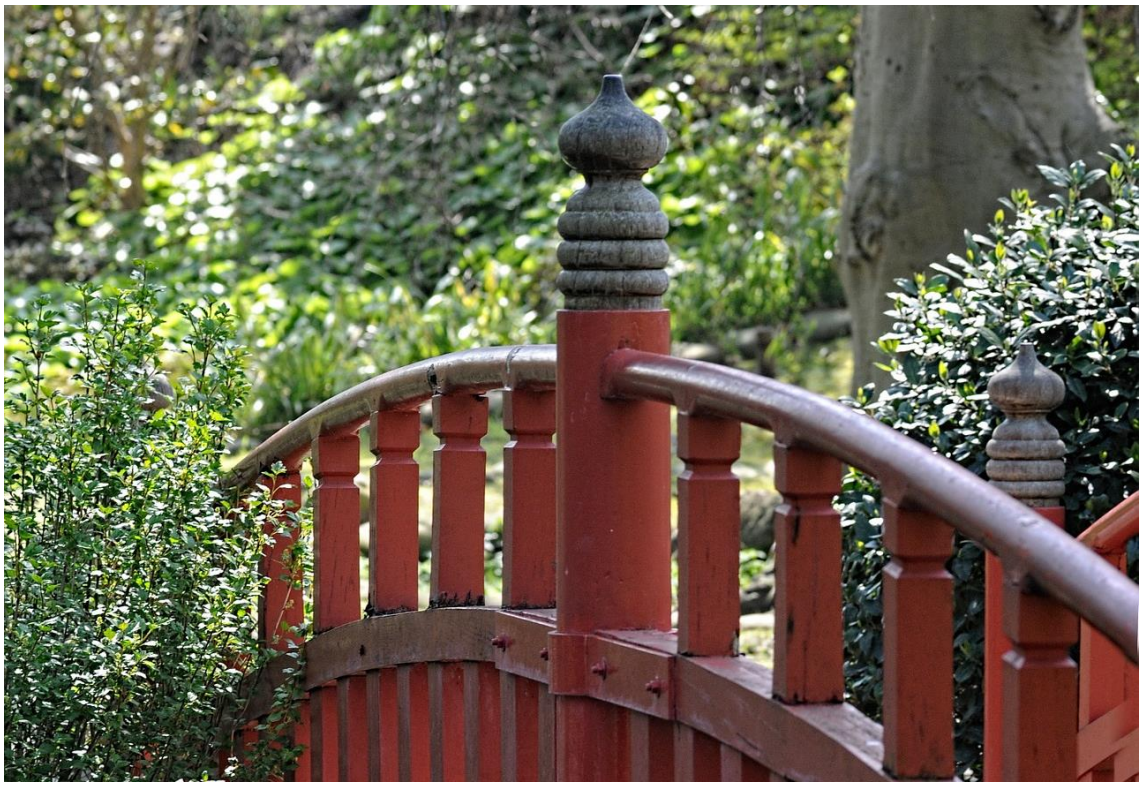
PICTURE 5: Simple low poly 3D mesh of the bridge with only the base shapes of the object modelled.

After the base shapes of the object were modelled, the focus was tilted more towards the details that would make the object interesting and believable. The details were done by using the same modelling methods as before. The boards of the bridge needed a lot more character; the old and used look was achieved by bevelling and tweaking the edges and the vertices. That resulted in distortions and cuts on the boards. (Picture 6).

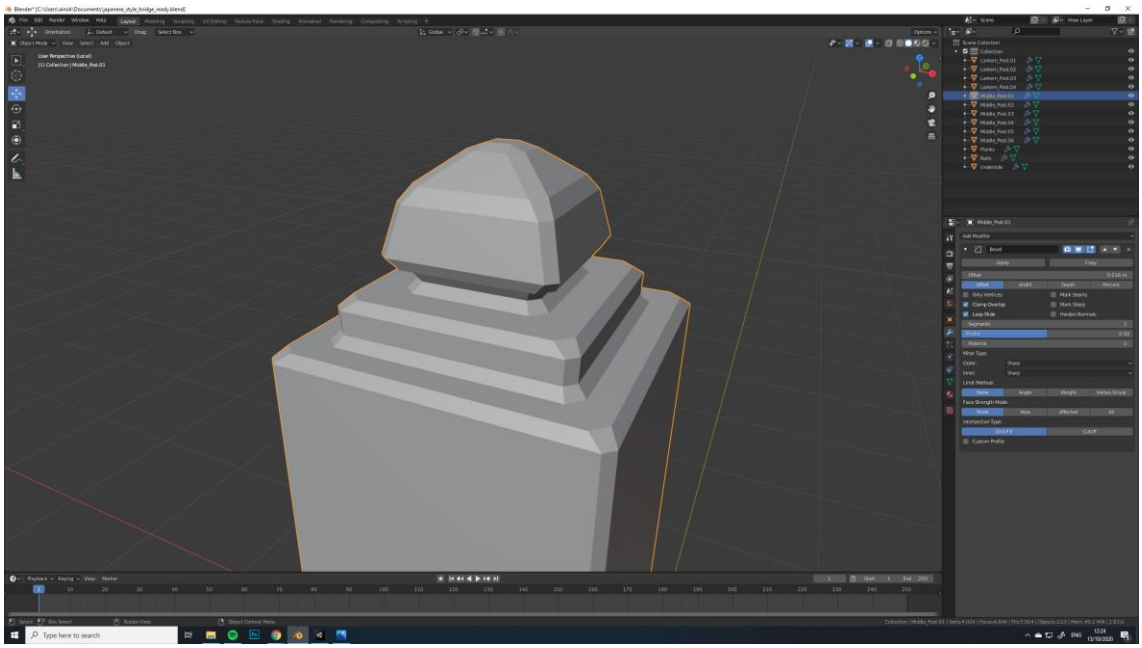


PICTURE 6: Worn and old look of the bridge was achieved by making cuts and some distortions to the meshes of the boards.

When looking at the reference pictures of Japanese bridges it was noted that often the poles of the bridges had some simple decorations on their tips. (Picture 7). Based on that some details were also added on top of the poles using extruding method and scaling. (Picture 8).

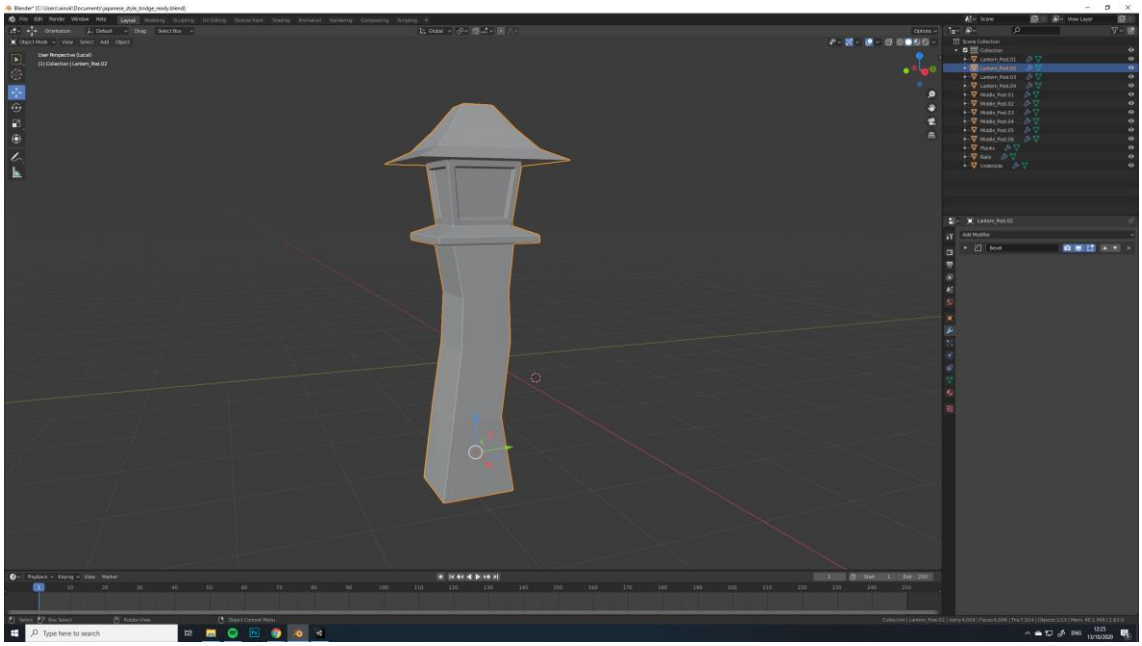


PICTURE 7: Reference for the decorations on top of the railing's poles. (Doukhan 2010)

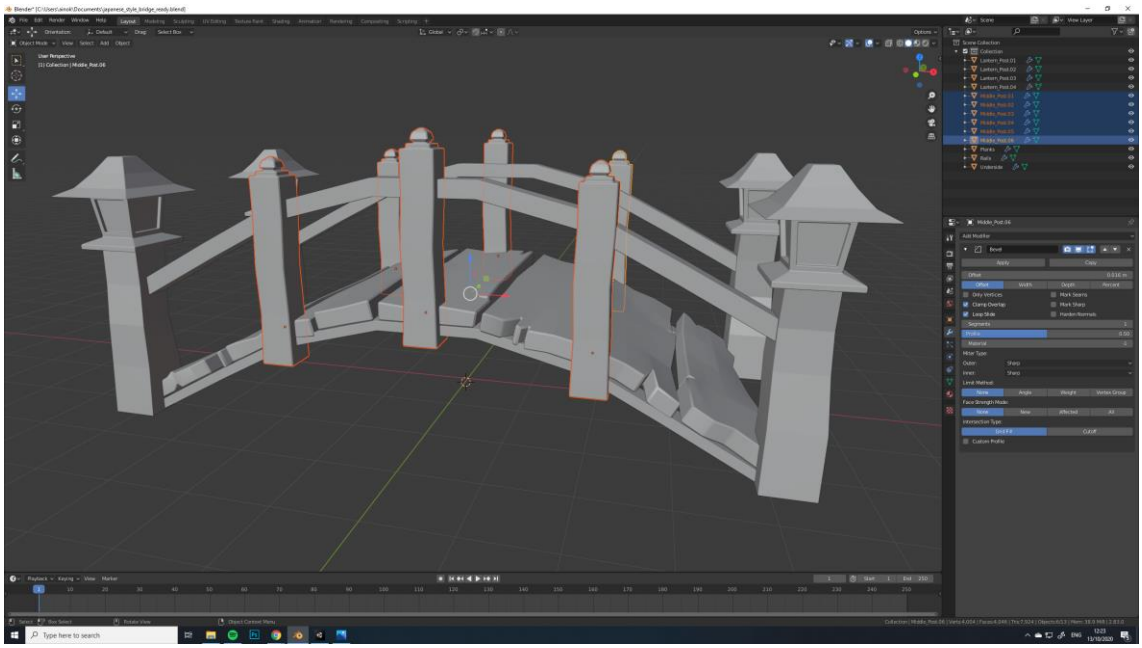


PICTURE 8: Simple details extruded on top of the poles.

The lamp posts looked dull compared to the other parts of the object. By first inverting and then extruding the middle parts of the lamp inwards, the more of a lantern-like look was achieved. (Picture 9). The middle pole of the bridge was duplicated a few times and the duplicates were placed evenly on both sides of the bridge to make the overall look of the bridge appear fuller. (Picture 10).



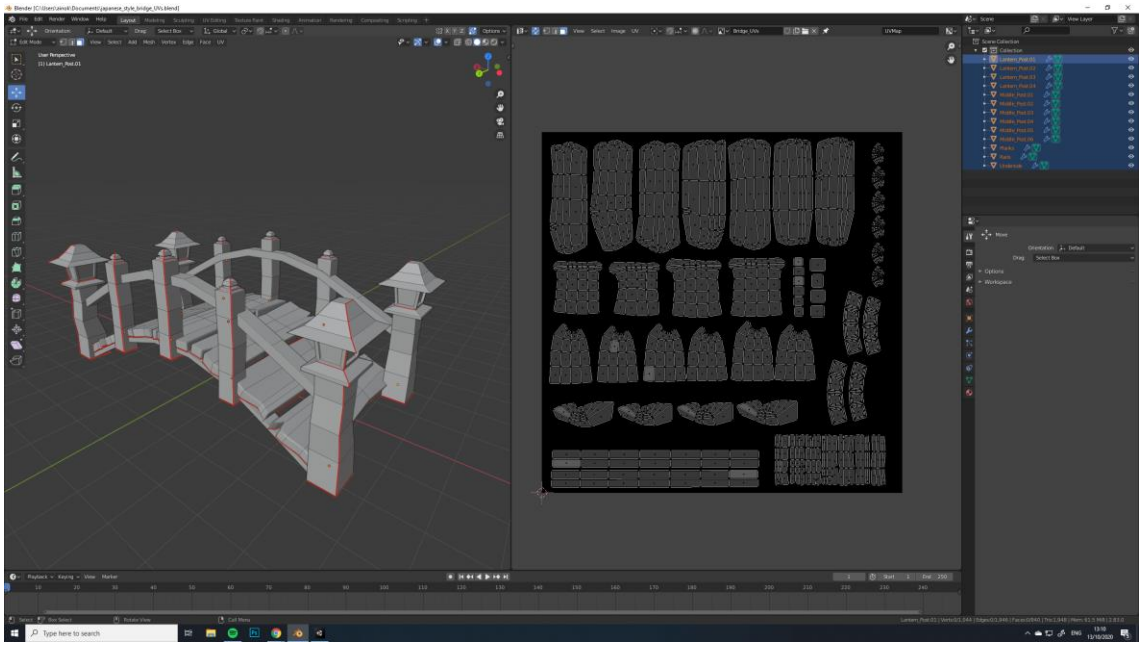
PICTURE 9: The lamp post got a more distinct lantern look by inverting and extruding the middle parts.



PICTURE 10: Poles were duplicated and placed evenly on both sides of the bridge.

After the base modelling was done, it was time to go through the UV unwrapping. A UV map is a flat presentation of the surface of a 3D model and the creation process of a UV map is called UV unwrapping. UV unwrapping is often needed to perform even if textures are not added to the model. That is because real time engines such as Unity need assets to be unwrapped to be able to bake some of its lights. (Denham n.d.).

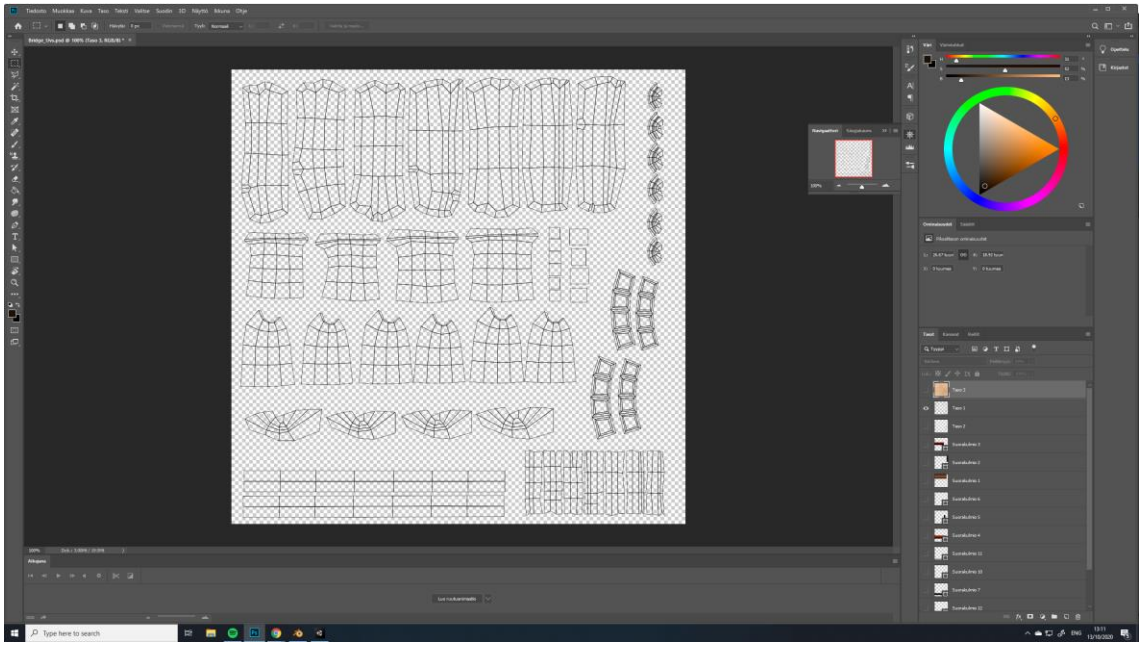
UV unwrapping is started by marking seams to the model by hand. A seam is a part of the mesh that determines where the 3D mesh will be split. (Denham n.d.). Automatic unwrapping is also available in Blender, but it has turned out to be not as accurate. The first method was used in this process because of its accuracy and easier handling, especially when moving and scaling different parts around the UV map. (Picture 11). After the UV unwrapping was complete, the UV map was exported from Blender as a UV layout and .png file. The UV layout includes the UV map on a transparent background. This UV layout worked as a guide in texture painting.



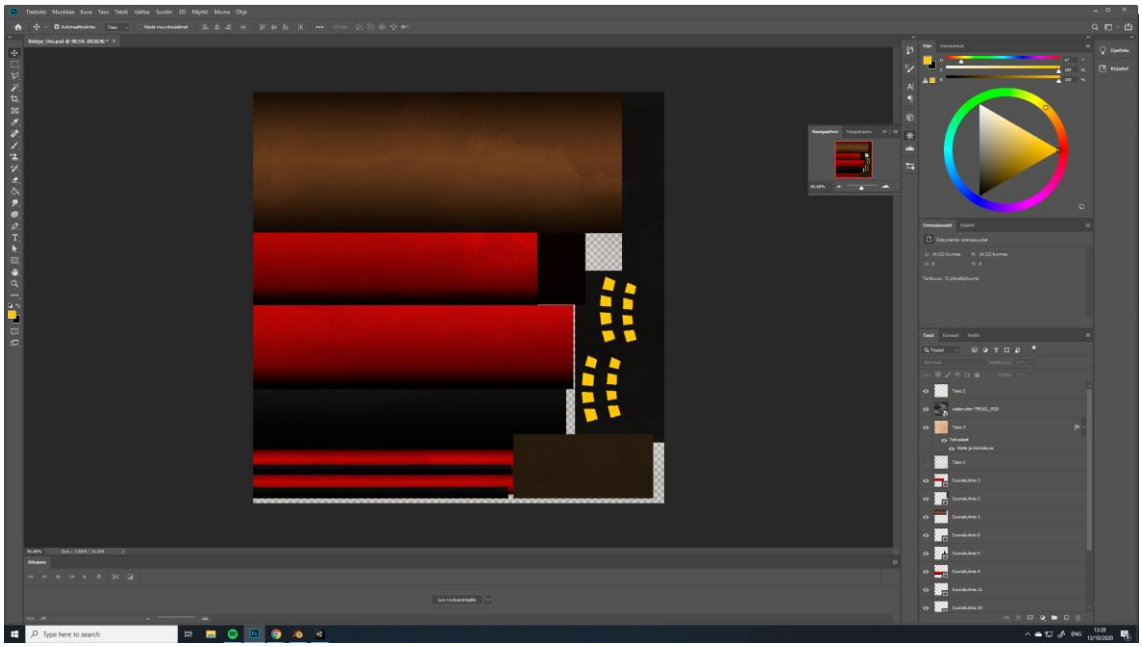
PICTURE 11: Bridge with marked seams on the left side of Blender workspace and the unwrapped UV map on the right.

5.2 Texture painting

The exported UV layout was opened in a digital painting software Adobe Photoshop (Picture 12). Every part of the UV layout was painted on different layers, which made it easier to manage the colours and effects for different parts of the mesh individually. Gradient effects and selection tool were the mostly used in this texture painting part. The actual painting was left out this time because the painting process can be very time consuming and tedious. Lastly two different images of a paper were imported on top of the UV layout's colours. By changing the opacity of these two images, a simple paper like texture was achieved and added on top of the colours. (Picture 13).



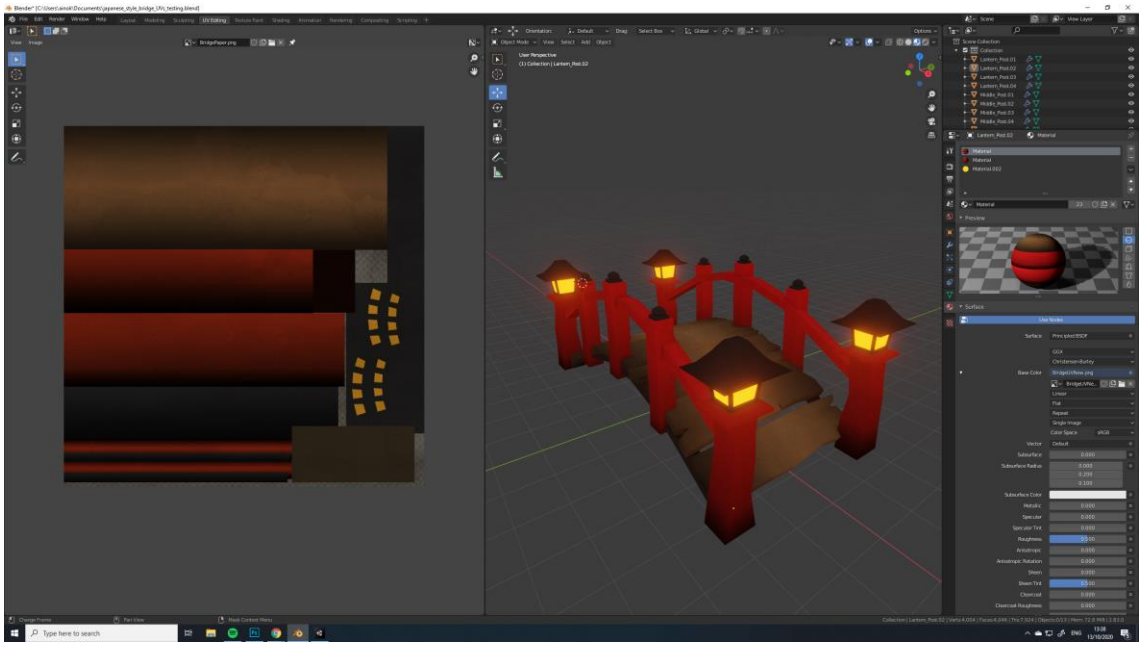
PICTURE 12: UV map imported to Adobe Photoshop.



PICTURE 13: The colours and paper effect on the texture map.

When happy with the look of the textures, they were imported back to Blender as a texture map. To see how the texture looks on the model, a material had to be created from the texture map. Material Base Colour was changed to Image Texture and the texture file was chosen as the source image. The created material can then be assigned to all parts of the mesh. (Picture 14).

If the texture is not quite right, for example, the colours are not vibrant enough, the texture file can be tweaked in Photoshop and imported easily back to Blender. In this project, some parts of the UV map were a bit off and did not land in correctly. The UV map had to be corrected in the UV editor by moving and scaling the parts of the railings. When pleased with the look of the 3D model it was exported from Blender as .fbx file.

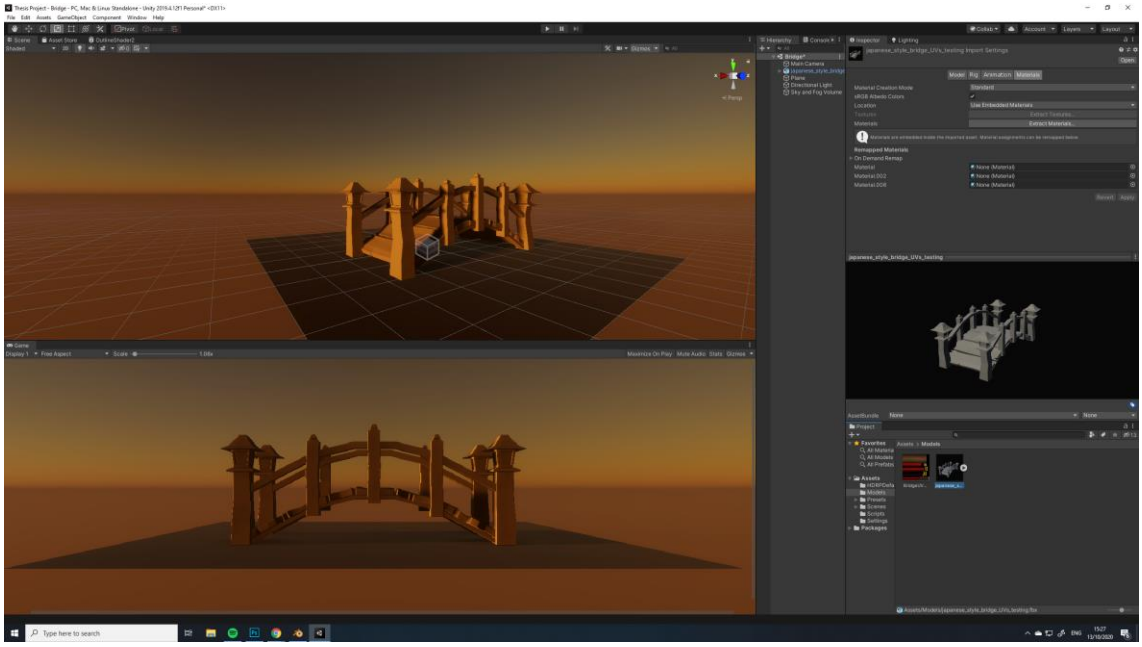


PICTURE 14: The material created from the texture map in Blender.

5.3 Implementation to Unity Project

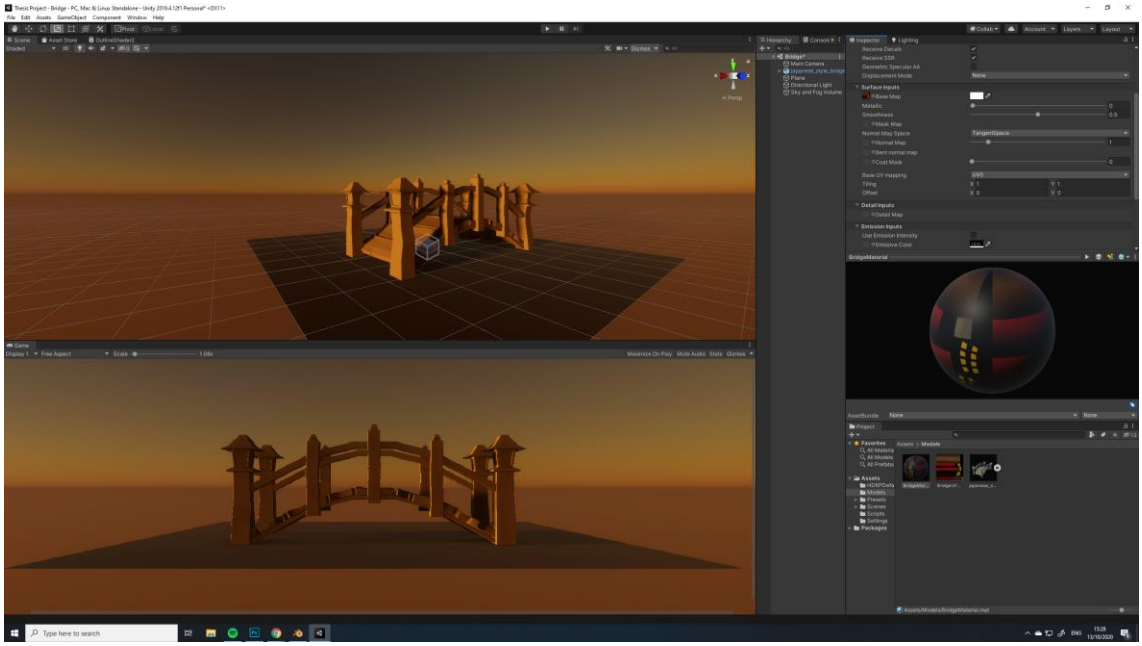
A new Unity 3D Project was created. The High-Definition Render Pipeline was used in this project with Unity's version 2018.0.0.

First the base scene was prepared for the model. The lights and camera were set to the scene and a plane was added to serve as a ground for the 3D model. Then the .fbx file of the 3D model and texture .png file were imported to the Unity project. After that, the model could be placed on its place in the scene. (Picture 15).

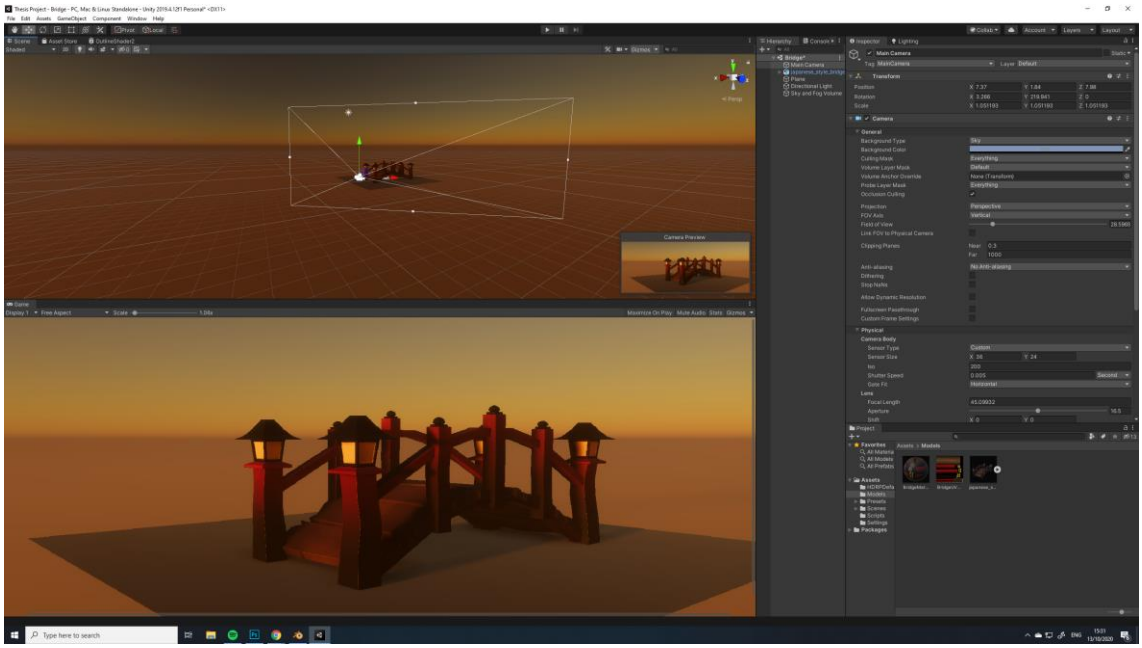


PICTURE 15: Unity scene with the bridge model dragged to the scene.

For the textures to appear in the scene, a material had to be created. After the base material was created, the texture could be dragged to the material's base map slot. The created material had a couple of settings that needed to be changed. Metallic input was set to zero because no glossiness or glow was wanted to be visible on the material. Smoothness was set about in the middle of the slider. (Picture 16). After the material was created it could be assigned to all the material slots in the model asset. The bridge texture was now in place and the shader could be implemented. (Picture 17).



PICTURE 16: Material is created, and the textures assigned to the base map slot. Some changes also to the material properties.

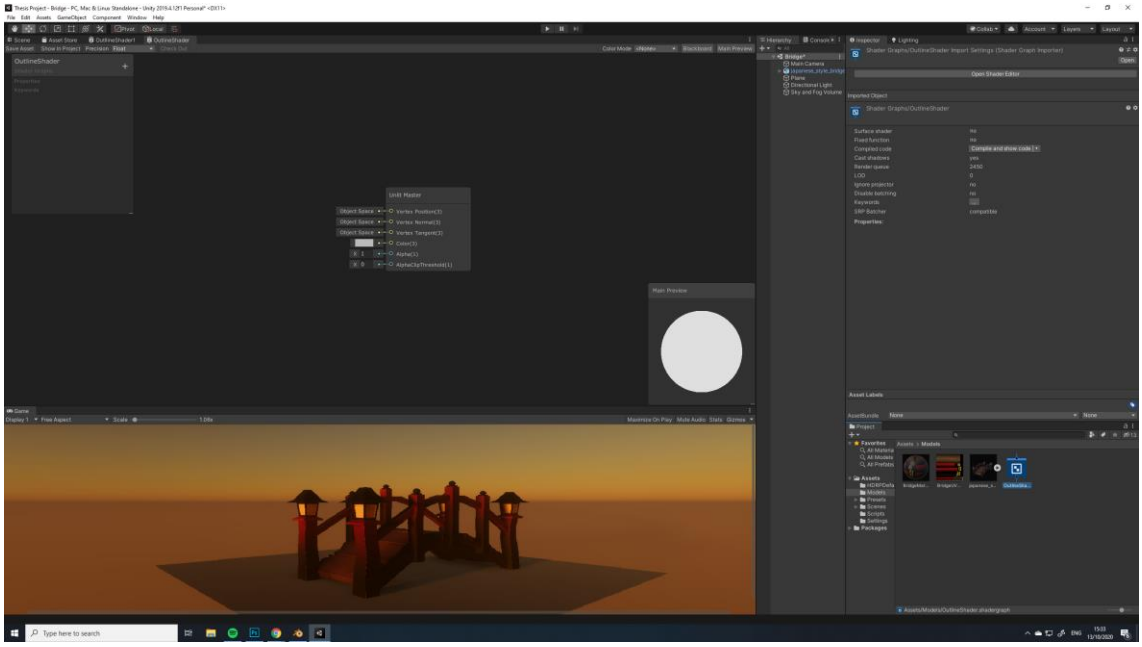


PICTURE 17: The bridge with visible materials in the scene.

5.4 Creating the Outline shader

The shader creation part was the most interesting and surprising part of the process. The shader should be able to draw a black outline for the model, which thickness could also be altered. The shader creation was done with Unity's Shader Graph tool and it consisted of modifying the geometry, applying back-face culling, and assigning a colour for the outlines. This Shader creation process was based on two different outline shader tutorials by AE tuts from 2019.

The process began with creating an unlit graph (Picture 18). By creating the shader with unlit graph as a master node, it was ensured that the black outlines were going to actually be black and could not receive or react to any light that is in the scene.

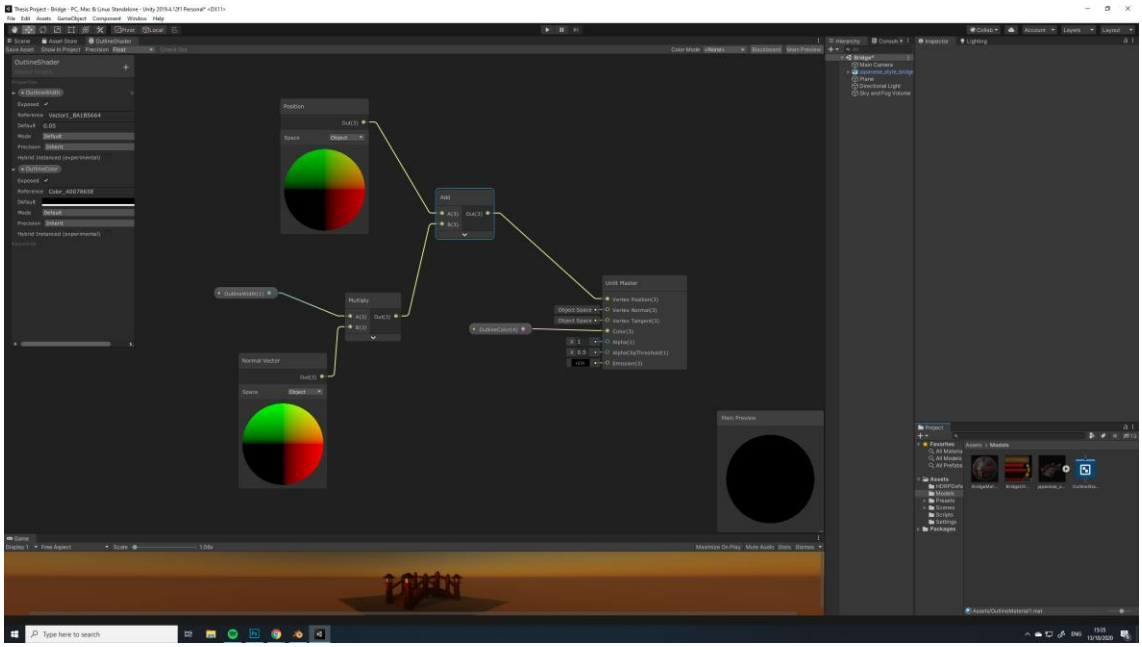


PICTURE 18: Shader Graph in Unity with unlit graph created and ready to be edited.

Normal vector node was added first. The Normal vector was multiplied with a Multiply node. A vector property, in this case named as OutlineWidth, which controls the width of the outline, was added and then given to the multiply node. This extruded the geometry along its normals and made everything appear thicker. The OutlineWidth property was set to 0.05 because that makes it easier to control the property if having to modify the width later.

Position node was created and connected to an Add node. The Add node receives the just created Multiply node the Position node. The Add node gives out a Vertex position to the Unlit Master node. After that, the normals were reversed using back-face culling to make the backside of the geometry visible instead of the front.

The Outlines needed to be a solid black colour as well, for that purpose a colour property was created, which in this case is called OutlineColour. Default colour was set to pitch black and assigned to Unlit Master node as a Colour. (Picture 19). The actual shader creation with Shader Graph was complete and it was ready to be set for the bridge model. This should have given the model a thicker silhouette that looks like an outline when the object was going to be drawn in front of it.

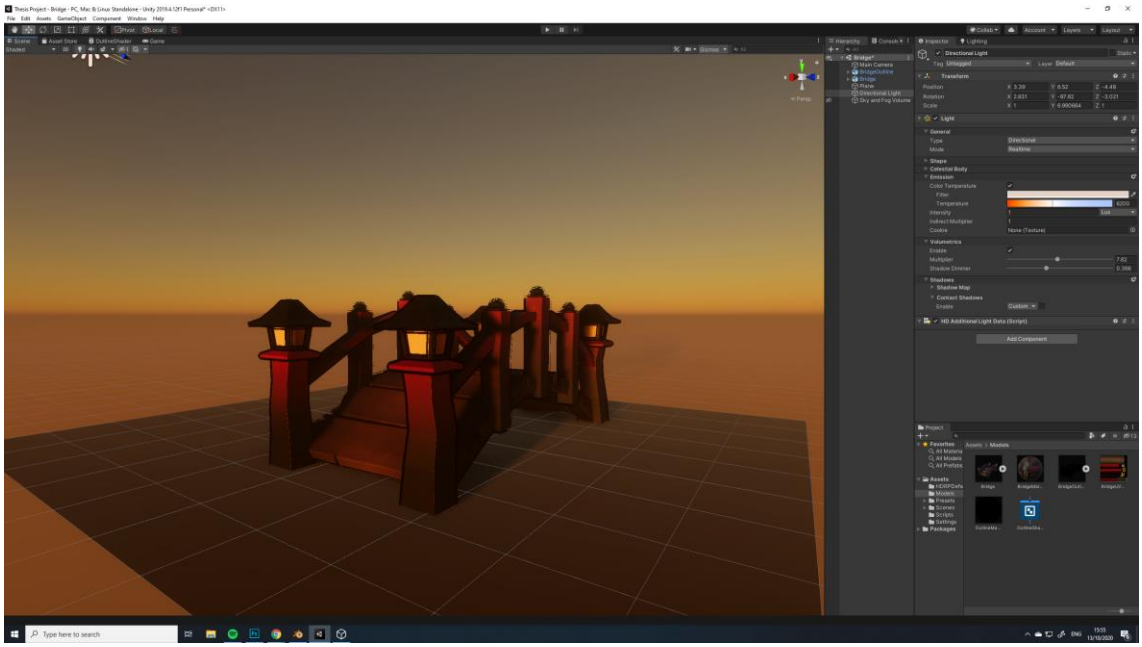


PICTURE 19: The outline shader in Shader Graph with all nodes and properties.

5.5 Final results

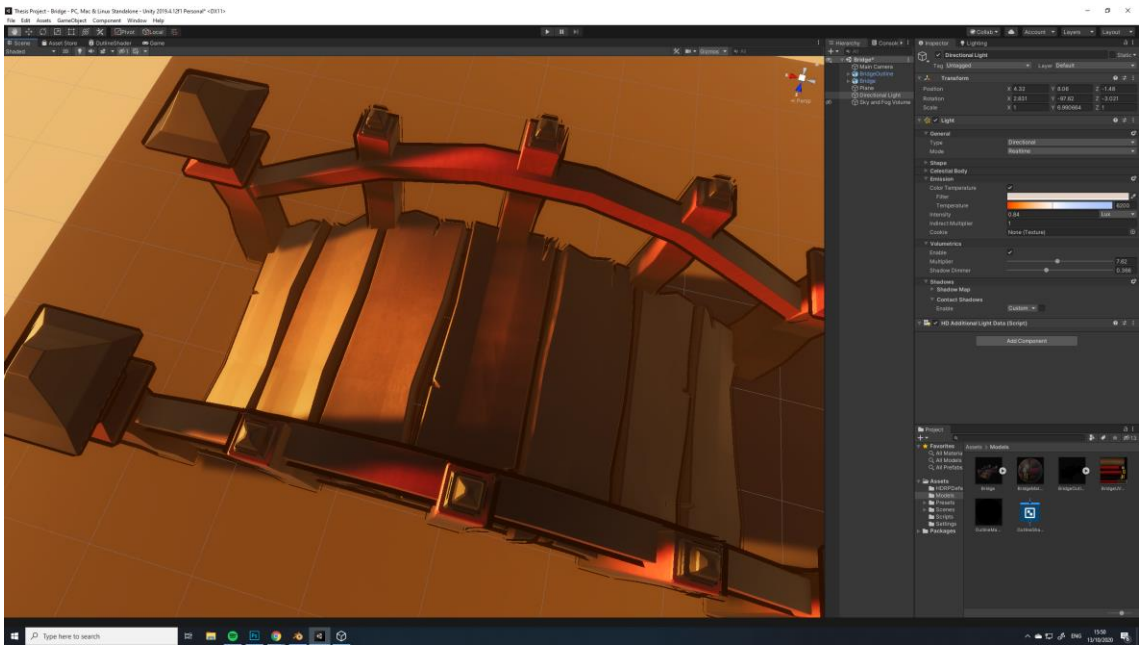
During the shader creation process it turned out that Shader Graph has some limitations that have an effect to the end product of this process. It appeared that it was not possible to create an outline shader without another model underneath the actual model in Shader Graph. That is because the Render Pipeline together with Shader Graph do not support multiple shader passes. When trying to create a shader that draws the outlines of an object multiple shader passes are usually required.

Acknowledging the fact that the outline shader was not possible to create with Shader Graph. It was decided to use the old and not as efficient way. A duplicate of the bridge was added to the scene. New material with the outline shader signed as base map was created. The outline material was then added as a material for the duplicated model. The second model was then set on its place right underneath the first bridge model. (Picture 20). Because of the specified scale of the shader and back-face culling, only the black edges were showing from behind the first model and that created the illusion of black outlines.

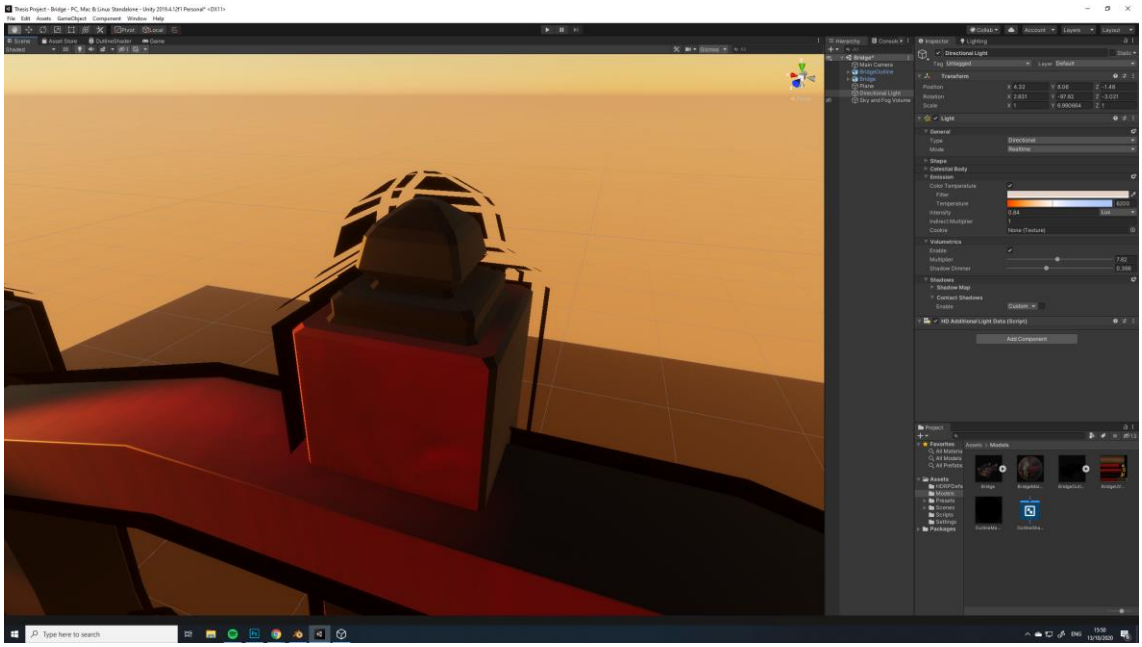


PICTURE 20: Bridge model in the scene, under that is another bridge model with the outline shader assigned as a material.

The way this effect was executed has however big limitations and efficiency problems. When edges in the 3D model are not smooth enough or the model has very sharp angles the outlines will have gaps between them. The outlines may also clip into the geometry in an unwanted way. As shown in the picture 21, the outlines look pretty good and seem to work nicely. In the picture 22 that has a close-up picture of the decorations on top of the poles, the lines have huge gaps between them, and it looks messy. The conclusion based on that result was that with, for example, a round object the outlines would be connected to each other and more solid looking.



PICTURE 21: The outline effect looks nice from certain angles.



PICTURE 22: There is more complex geometry in the details on top of the poles. The outline shader behaves in undesired way.

6 CONCLUSION

Examining Okami and its inspiration gained from Sumi-e art style and East Asian culture and mythologies are very present throughout the game. The aspects taken from Sumi-e painting technique are visible in the graphics but in the game-play as well.

It turned out that Okami's graphics are apparently much simpler than they look. but the methods used to achieve that distinct look would not be efficient from the game developer's point of view. Placing a second model underneath the first one, using back-face culling and randomized scaling had maybe been the only choice to achieve the desired effect back in the days, but nowadays developers would possibly use some other method, for example, post processing shaders.

The outline shader creation process was all in all very instructive, but the outcome was not expected at all. Thinking that creating a simple outline effect in Unity's Shader Graph tool would have been common knowledge and indeed possible, but it clearly turned out to be quite the opposite. The Shader Graph tool in Unity has been around just a couple of years and that for lacks some features. It was however chosen for this project because of its easiness and accessibility. The Graph network is rather quickly manageable, and even beginners can get their head around the most basic nodes.

When making a video game and deciding on its graphical style to have bold outlines or the game having any outline effects at all, using Shader Graph to that purpose should not be considered at the time of writing. There are too many limitations to the geometry of the 3D objects, lack of efficiency when having two models on top of each other and that for, a possibility to have grand issues in the performance of a game.

In game development the basic outline effect is quite common, and Unity holds other ways for creating shaders. The better solution for this outline shader case would be to write the shader with shading code in Unity. The suggestion for fur-

ther examination would be to find out more about the shader coding and implement the shader in some other way. There were many shader scripting tutorials found when searching about outline shaders, so the information is definitely not limited.

REFERENCES

AEtuts. 2019. Outline Shader using Shader graph (2.5 techniques). Youtube video. Watched 17.8.2020. https://www.youtube.com/watch?v=_zl-CyYeM6IU&t=574s&ab_channel=AETuts

AEtuts. 2019. Outline Shader using Shader graph (2.5 techniques). Youtube video. Watched 17.8.2020. https://www.youtube.com/watch?v=_zl-CyYeM6IU&t=200s&ab_channel=AETuts

AuraTummyache. 2017. Art styles like Okami? Read 17.5.2020. https://www.reddit.com/r/gamedev/comments/69lpo5/art_styles_like_okami/

Capcom. 2017. Read 12.4.2020. http://www.okami-game.com/index.php

Capcom – Clover Studio. 2012. The Weapons of Okami. Read 16.10.2020. https://www.ign.com/articles/2006/08/11/the-weapons-of-okami

Cooper, T. 2018. Introduction to Shader Graph: Build your shaders with a visual editor. Read 12.8.2020. https://blogs.unity3d.com/2018/02/27/introduction-to-shader-graph-build-your-shaders-with-a-visual-editor/#:~:text=A%20Shader%20Graph%20enables%20you,You%20can%20do%20things%20like%3A&text=Share%20node%20networks%20between%20multiple,nodes%20through%20C%23%20and%20HLSL

Cowan, D. 2006. Critical Reception: Capcom's/Clover Studio's Okami. Read 16.4.2020. https://www.gamasutra.com/view/news/101890/Critical_Reception_CapcomsClover_Studios_Okami.php

DenisDoukhan. 2010. Albert Kahn Garden. Picture. Accessed on 11.9.2020. https://pixabay.com/photos/albert-kahn-garden-bridge-656482/

DenisDoukhan. 2010. Albert Kahn Garden. Picture. Accessed on 11.9.2020. https://pixabay.com/photos/albert-kahn-garden-japanese-garden-562113/

Hasu, J. 2018. Fundamentals of shaders with modern game engines. Tietotekniikan diplomityö.

Hocking, J. 2020. Introduction to Shaders in Unity. Read 28.10.2020. https://www.raywenderlich.com/5671826-introduction-to-shaders-in-unity#toc-anchor-001

Jaranson, C. N.d. What is Sumi-e? Traditional East Asian Brush Painting. Read 13.6.2020 http://sumiesociety.org/whatissumie.php

Kaldaien. 2017. Okami HD. Read 17.5.2020. https://steamcommunity.com/app/587620/discussions/0/14990005474866316 84/?ctp=2

Karma, K. 2013. Sumi-e-tekniikan tutkiminen oman teoksen kautta. Kuvataiteen koulutusohjelma. Kuvataiteilija (AMK). Kulttuurialan opinnäytetyö.

Karon, P. 2019. Why Digital Artists Need to Use Reference Images. Read 27.8.2020. https://cgcookie.com/articles/the-importance-of-reference

Petty, J. N.d. What is Polygon Mesh. Read 28.10.2020. https://conceptartempire.com/polygon-mesh/

Roper, C. 2012. Okami Review. Read 20.9.2020. https://www.ign.com/articles/2006/09/15/okami-review-2

Ross, E. 2019. Outline Shader. Read 20.5.2020. https://roystan.net/articles/outline-shader.html

Sato, S. 2010. Japan Objects. Sumi-e painting of a bamboo. Picture. Accessed on 15.10.2020. https://japanobjects.com/features/sumie

Shannon, C. 2013. The Four Treasures. Read 19.10.2020. http://caseyshannonstudio.blogspot.com/2013/11/the-four-treasures.html

Shepard, I. 2018. More Than Breath of the Wild, Okami HD is the Zelda Game I've Been Wanting for the Switch. Read 27.10.2020. https://adventurerules.blog/2018/10/12/more-than-breath-of-the-wild-okami-hd-is-the-zelda-game-ive-been-wanting-for-the-switch/

Totilo, S. 2006. GAMEFILE: 'OKAMI' GOES GREEN; OFFICIAL WII WORD; 'IDOL' LAUNCH AND MORE. Read 14.10.2020. http://www.mtv.com/news/1542741/gamefile-okami-goes-green-official-wii-word-idol-launch-and-more/

Unity Technologies. 2020. Shader Reference. Read 28.10.2020. https://docs.unity3d.com/Manual/SL-Reference.html

xb100. 2016. Chinese calligraphy scene. Picture. Accessed on 26.11.2020. https://www.freepik.com/free-photo/chinese-calligraphy-scene-text-chinese-ancient-prose_1167970.htm