Texturing Process in Hand Painted Low Poly Game Art

Inka Kaasinen

BACHELOR'S THESIS
June 2023

Degree Programme in Media and Arts
Interactive Media

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Media and Arts
Interactive Media

Inka Kaasinen
Texturing Process in Hand Painted Low Poly Game Art

Bachelor's thesis 56 pages, appendices 1 page
June 2023

The low poly modeling approach has been a commonly used and vital technique for video games since the earliest 3D games were made. What started as a necessity caused by system restrictions has now become more of a stylistic choice. Whilst 3D game models still aim for lower polycounts than models used in animation, today’s hyper-realistic in-game models can still consist of tens of thousands of polygons. What is nowadays coined as a low poly model is typically a very stylized and blocky 3D model with sharp edges, often accompanied by simplistic, cartoony, and flat textures.

The objective of this thesis was to research low poly modeling and the implementation of complex and stylized hand-painted textures onto a low poly 3D game model. The study also approached what kind of production methods could be suitable for independent game developers, hobbyists, or other people with limited resources and budgets.

The theory part of the thesis studied different approaches for modeling a 3D character. It also examined texturizing and producing hand-painted textures in a game development environment. The conclusions of those studies were put into practice in the form of a project. In the project a low poly 3D character was modeled in Blender without utilizing a high poly base. Two hand-painted textures were produced for the model by using 2D and 3D based painting programs. The results indicated that both programs could be used to create professional textures and they could benefit from being used together.

Key words: 3d modeling, hand painted textures, low poly, texturing, game character

CONTENTS

ABBREVIATIONS AND TERMS

2D	Two-dimensional
3D	Three-dimensional
Baking	The process of saving and transferring information related to a 3D model into a texture file
Digital Painting	Using the paint tools of a photo-manipulation program or a digital painting program to create imagery
FBX	Filmbox, a 3D model file type typically used in game development or animation
FPS	Frames Per Second, how many frames a game can process into a single second of gameplay
High Poly	A 3D model with a high amount of polygons
Low Poly	A 3D model with a low amount of polygons
Map	A 2D image that is wrapped around a 3D model
Mesh	A 3D structure consisting of polygons
Model	A 3D structure
N-gon	A polygon with more than 4 vertices and edges
PBR	Physically based rendering, an approach to render realistic textures
Polycount	The number of polygons in a model
Polygon	A flat shape in a 3D space, defined by three or more vertices and edges
Primitive	A default 3D object or shape provided by the 3D modeling program
Render	A 2D image put together by a program’s render engine which contains all the geometric data, materials and world information of a 3D model or a scene
Texture	A 2D image that is wrapped around a 3D model
UV mapping	A form of texture mapping
Workflow	A series of tasks needed to finish a broader task

1 INTRODUCTION

Low poly modeling and video games have been steadily gaining popularity since the early 2010s. Whereas low poly used to be a restriction caused by lack of power in hardware, it is nowadays a conscious stylistic choice. The modern low poly style highlights the cartoony and boxy nature of 3D models and embraces the low-resolution feel it brings. Simplistic and often very flat textures are also an integral part of the low poly style we see today.

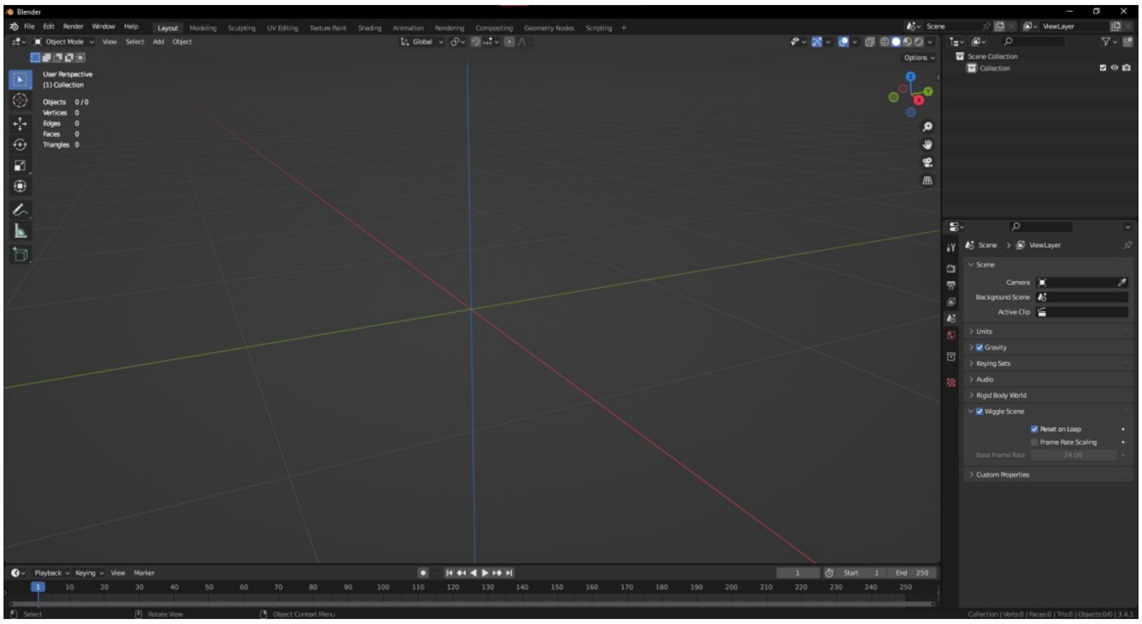
How would one step away from this simplistic and flat style whilst keeping the structural and performative advances of low poly models? Combining low poly-count models with highly detailed textures is a common workflow in 3D game production. However, these textures are often generated by utilizing a high poly base and procedural texturing. Even hand-painted textures like those seen in games such as World of Warcraft often utilize high poly sculpted bases to generate baked texture maps for the in-game models. Baked base maps are often the preferred method, as efficiency and speed are highly valued in the game development field and map generation can help reduce the time spent on texture painting. However, baking and procedural texturing bring with them their own sets of limitations to the creative freedoms the texture artist can take with the model.

With efficiency being a must in game development, is it even valuable to approach texture painting from a more manual painting direction? This thesis aims to explore low poly modeling as well as methods for producing hand-painted textures for a low poly 3D model without utilizing a high poly base mesh.

2 3D MODELING

3D modeling is a computer graphics technique to produce digital three-dimensional structures or surfaces, typically by using a 3D modeling program. These 3D structures, referred to as 3D models, can be utilized in many fields from engineering to architecture and from animation to 3D printing. From these fields, the ones focusing more on the artistic sides of 3D modeling may be more noticeable to the average person. 3D art is something a person might encounter numerous times in a day, as it exists in advertisements, TV, movies, games, and various other forms.

3D modeling requires an understanding of how a 3D space works and, according to Zeman (2014), how objects and structures flow in said space. To help keep track of the 3D space, 3D modeling programs utilize a universal 3D grid with three separate axes, x, y, and z (Picture 1). In addition to these, 3D grids utilize a world origin, which is a singular point in the 3D space from which all other things are measured (Zeman 2014, 2).



PICTURE 1. Colored lines indicating the 3D space’s x-, y-, and z-axes in a 3D modeling program.

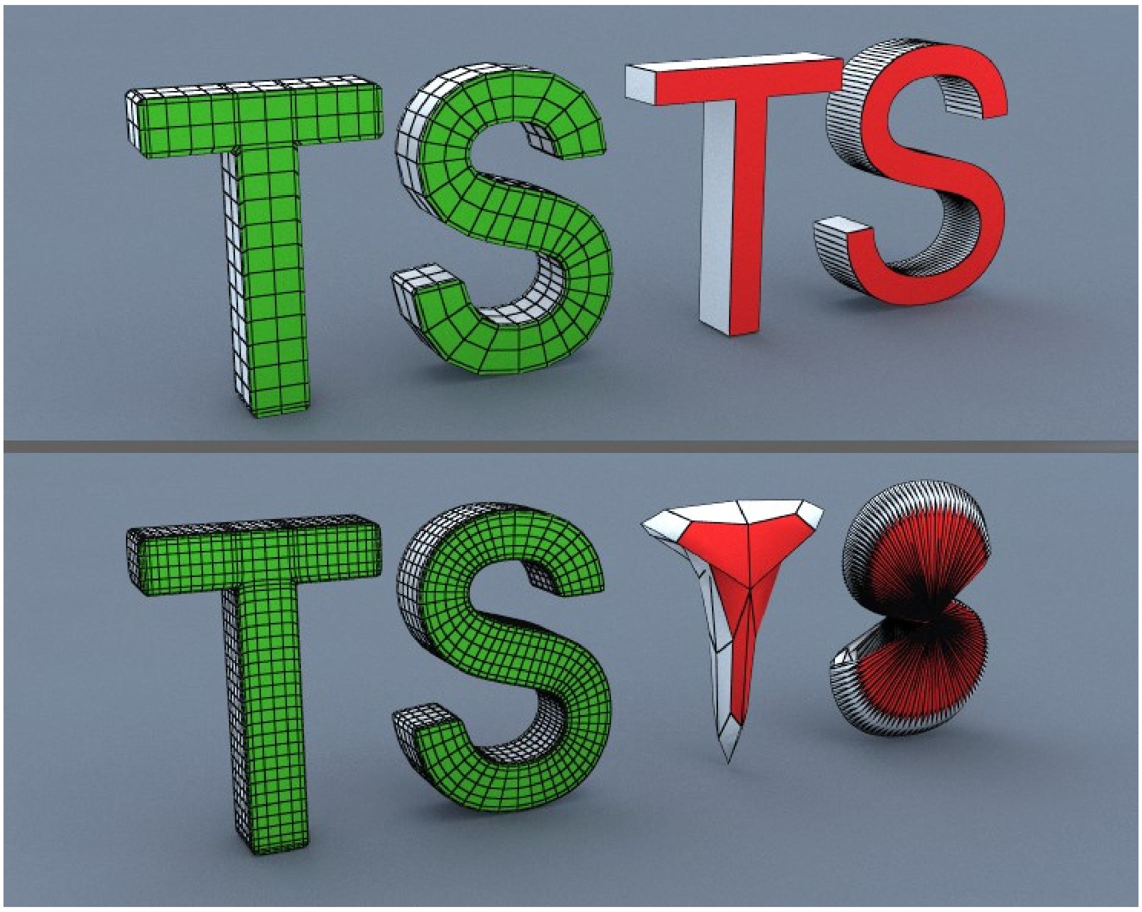
2.1 Creating 3D objects

3D models can be constructed utilizing various methods and approaches. The produced structures thus vary in their properties, the way they look, and even how they fundamentally function. Some of these 3D structures are such as curves, NURBS, and text objects. Curves are Bezier curve 3D elements, which can be controlled by adjusting their control points, also known as handles. For example, in Blender, default curves have two of these handles. Multiple curves can also be used to define a three-dimensional shape, which is called a surface. NURBS stands for Non-Uniform Relational B-Spline, and they are curves that do not have handles, unlike Bezier curves.

Digital 3D objects like NURBS or curves are more commonly used in fields like architecture, as they can produce very realistic 3D models and renders. However, their features are not very suitable for 3D animation or game development, as especially game engines favor models consisting of vertices and polygons.

Vertices are singular points in a 3D space. They have coordinates in the space, but they do not have width, height, or depth by themselves. When three or more vertices are connected, they form a polygon. A polygon with three edges is called a tri or triangle, one with four edges is a quad or quadrilateral polygon, and a polygon with more than four vertices and edges is called an N-Gon (Turbosquid n.d.). When polygons are combined to form a surface or structure, it’s called a mesh. Vertices, polygons, and meshes can all be manipulated with the transformation tools in a 3D program.

When creating assets for a video game, it is good practice to try and avoid using N-Gons in your models. As mentioned on Turbosquid (n.d.), N-Gon meshes are more difficult to subdivide and they might behave in unpredictable ways when rendering the models or importing and exporting them to and from the modeling software (Picture 2).

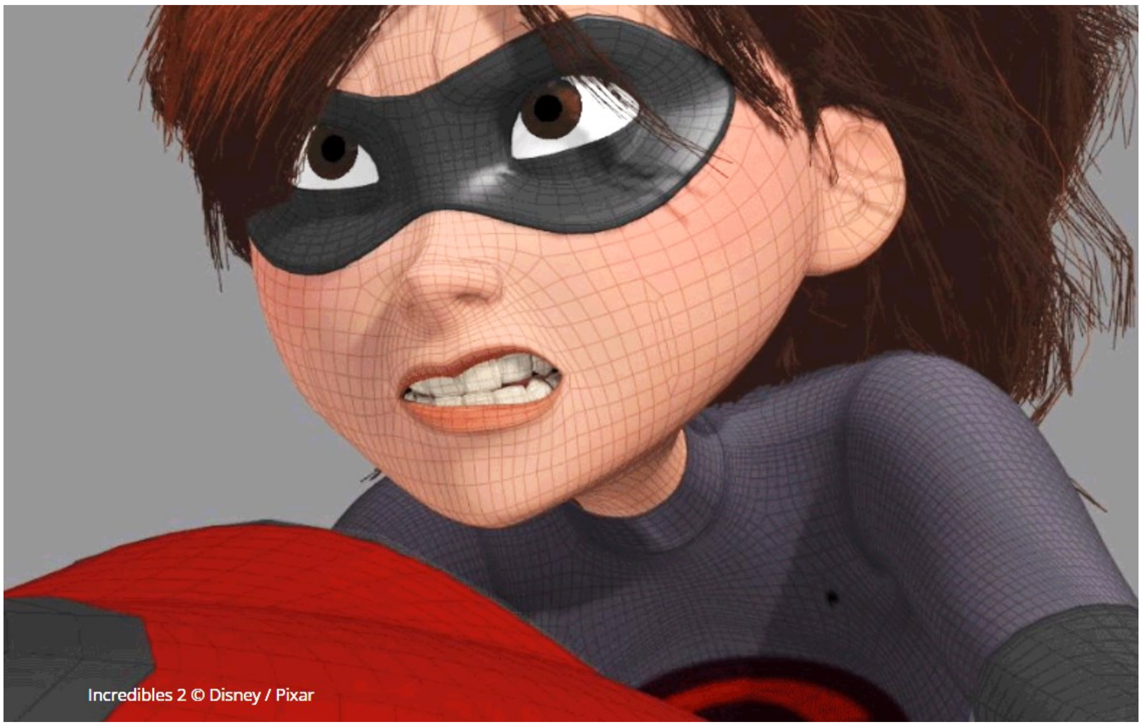


PICTURE 2. Models containing N-Gons subdivide unevenly and unpredictably (Turbosquid, n.d.).

2.2 3D modeling for games and other forms of media

The production methods and resulting 3D models can vary greatly depending on what type of media or usage the 3D model is produced for. Architectural 3D renders need to be able to convey the space realistically, and as such they need to have 3D models with realistic shapes, textures, and materials. An engineer working on a 3D drawing of a motor part needs to have accurate mathematical measurements for all the different gears, screws, and other parts but does not necessarily need to add any materials to the model. Neither the architect nor the engineer needs to worry about their models' polycounts either. This is something a game artist must keep in mind when working on their 3D models.

The workflows for creating 3D character models for animation and games share similarities with one another, as they both aim to create movable models that can be rigged. The models still differ from one another, however, with the most noticeable difference being their polycount. This difference is caused by the way the models are rendered. 3D animation is prerendered and as such, is a compiled collection of 2D images. Thus, no render engine is working as a person is watching the finalized animation. Thanks to this, 3D animation models can have a high polycount (PICTURE 3). Materials, eyes, hair, and many other parts can be portrayed and animated more accurately with a higher polycount. These accuracies help bring a level of authenticity and lifelikeness even to stylized or cartoony animation models.



PICTURE 3. The wireframe of an animation character from Disney Pixar's Incredibles 2 (Leif Pedersen 2019, edited).

Unlike animation, video game engines use real-time rendering. For every frame in the game, the game engine must process the models, materials, lighting, and all other information that is presented on the screen while the game is running (Totten 2012). To ensure a good playing experience, the game engine must be able to do this processing as smoothly as possible. A good way to ensure smooth playback and a good FPS rate for the game is to optimize the game's assets. A game artist can make sure this happens by controlling the assets and models' polygon count (Totten 2012). The lower the polycount the models have, the less processing the game engine must do and the smoother it runs.

2.2.1 Restrictions on polycounts

There are no universal guidelines for what a 3D game model's polycount should be, as it can depend on various factors. Game engines and consoles can have polygon budgets which can limit the overall number of polygons that can be on the screen at one time (Totten 2012). It is also essential to consider how important the model is. Typically, an important model has more leniency in how many polygons it can have, and the polycount tends to go up alongside the importance of the model. For example, a box on the background does not require a higher polycount than a player character.

The restrictions and limitations set for 3D game models have also changed over the years. As game engines and hardware develops, game models can utilize higher polycounts, bigger texture sizes, multiple texture maps, and many other improvements. For example, in God of War II, a game released for PlayStation 2 in 2007, the main character Kratos' model consisted of around 5700 polygons (Picture 4). Twelve years later, in God of War released for PlayStation 4 in 2019, Kratos' model has about 80000 polygons in total – 32000 of those just in his face (Picture 5). (Shuman 2019.)

PICTURE 4. Kratos from God of War II (Santa Monica Studio 2007).

PICTURE 5. Kratos from God of War 2019 (Raf Grassetti 2018).

2.3 Modeling techniques

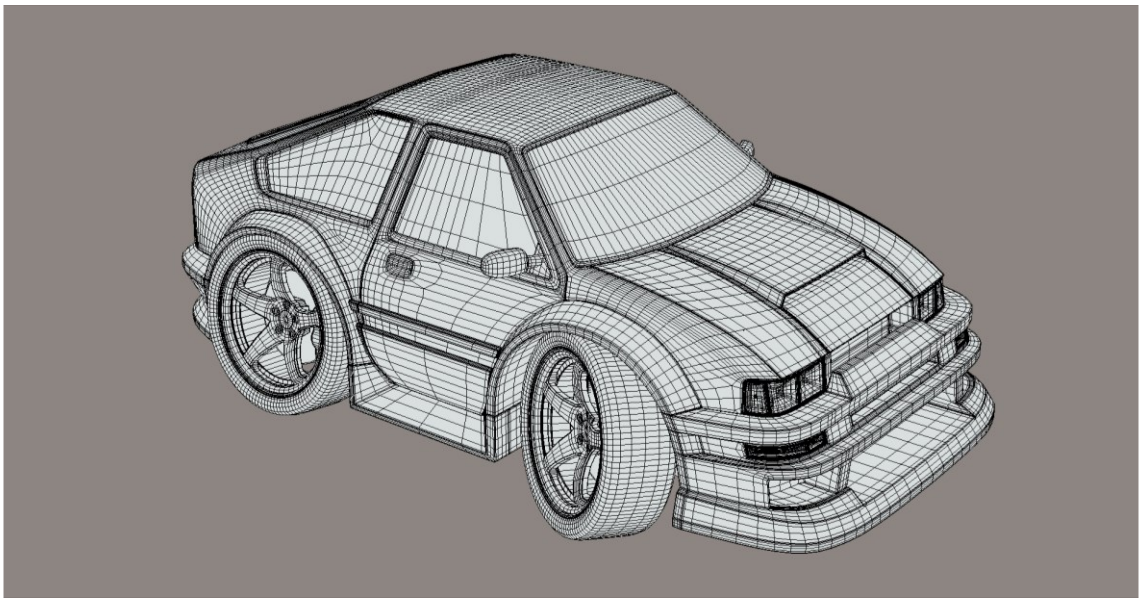
As stated previously, 3D models can be produced with different techniques and pipelines. In video game development, commonly used approaches are polygonal modeling and 3D sculpting. Polygonal modeling includes such techniques as

- high poly modeling
- low poly modeling
- box modeling
- point-to-point modeling.

Outside of polygonal modeling there are techniques such as curve modeling, photogrammetry, and simulations. These can also be utilized in game development when needed, but they are rarer as they may require extra work and adjustments to ensure the models work inside game engines as well. For example, in photogrammetry, where a model is produced by generating it from multiple pictures of an object, the result is a very high-resolution model. This model often has a messy topology, which would then need to be cleaned and turned into a usable low poly model.

2.3.1 Polygonal modeling

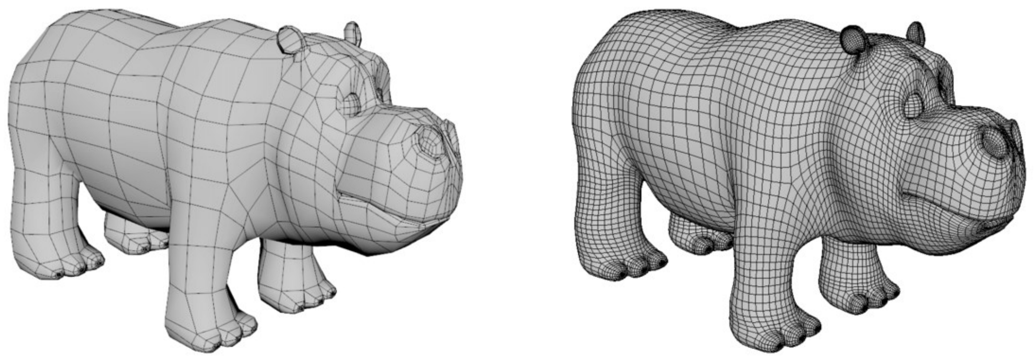
As the name suggests, 3D models produced with the polygonal approach consist of polygons (Picture 6). The number of polygons in the model dictates the division of the model into either high or low poly. Both high and low poly models are utilized in video games, but as mentioned previously, it is preferable to keep the game models' polycount on the lower side to help with the render times and overall game performance. Video games use real-time rendering. As such, the more polygons the game engine must go through, the longer the render times will be. In turn, longer render times translate into worse framerate in the game.


PICTURE 6. Polygonal 3D model of a car (Wallon 2022, edited).

When using polygonal modeling for video games, it's important to keep in mind what kind of polygons the model contains. In game development, it is preferable to use models containing either tris or quads, as game engines break models down to tris when rendering them. If a model was made with n-gons, polygons consisting of more than four sides, the game engine might have trouble breaking it down to tris. As such, as mentioned before, the resulting in-game model might render in unexpected ways (Turbosquid n.d.).

Polygonal modeling is especially useful in hard surface modeling. Hard surface modeling is a modeling technique used for generally inanimate objects, such as cars, armor, or machines (Wingfox 2022). Being able to accurately control and manipulate a model’s polygons is an advantage when working with these types of objects. Polygonal modeling can also be used to model characters or creatures, however as stated by FlippedNormals (2020), it is typically easier to utilize 3D sculpting with these kinds of organic models due to their more complex nature. When producing low poly models, the benefit of being able to accurately control the polygons outweighs the difficulties of modeling organic shapes with them. As such, polygonal modeling is the ideal approach to use with low poly models, no matter what the model is of.

Polygonal models can be divided into high and low poly models (Picture 7). While there are no strict rules or guidelines regarding where the division between these two subclasses happens, low poly models generally stay within or under thousands of polygons, whereas high poly models can contain even millions of polygons.

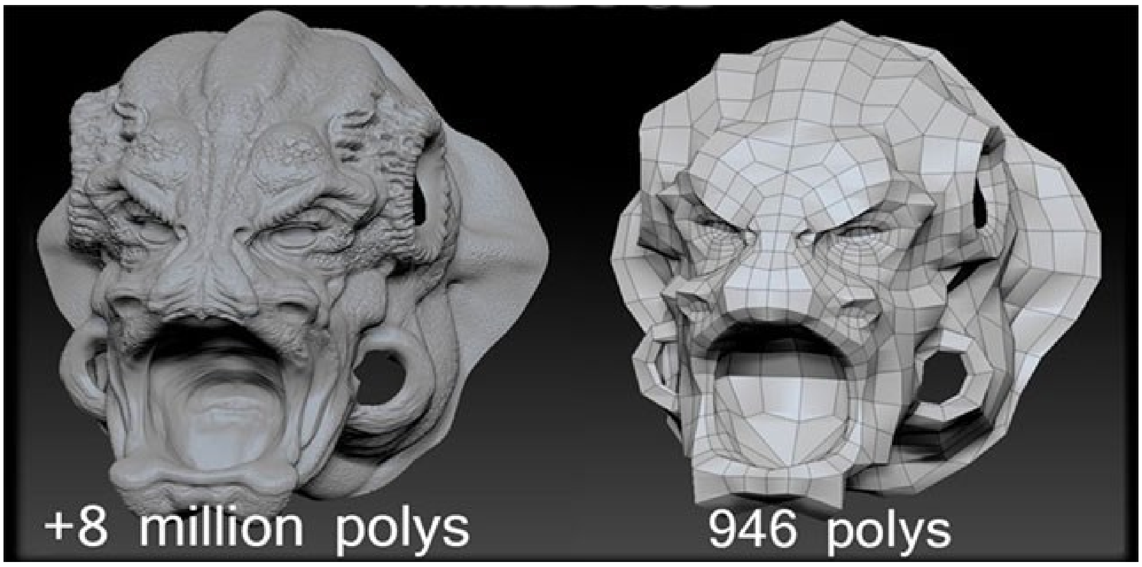


PICTURE 7. Low poly and high poly versions of a 3D model (3DCoat n.d., edited)

The division between a low poly and high poly model can also depend on things like what the model is of, the importance of the model, and the overall style of the game, as well as which game engine and which platform it is being developed in and for. High poly models tend to be used more in animation. In game development, they mostly exist as a base for the lower resolution in-game 3D model, as well as for baking detailed normal maps onto the low poly model.

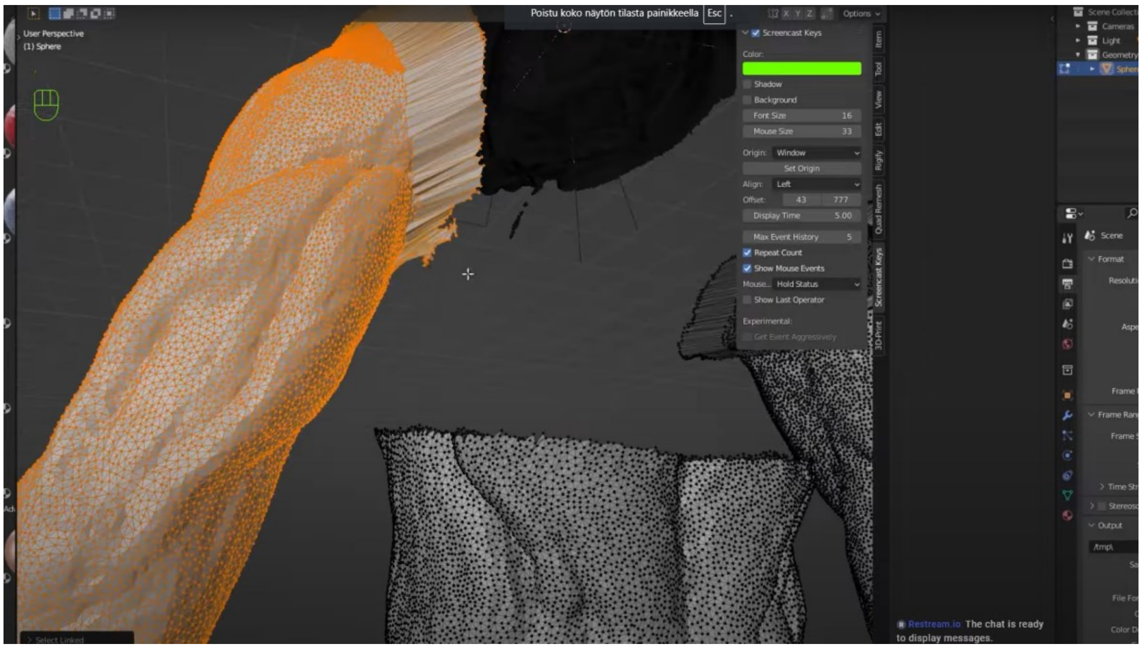
2.3.2 3D sculpting

3D sculpting can also be referred to as digital sculpting. When modeling with 3D sculpting, the artist sculpts the model out of 3D material similarly digitized clay (Heginbotham n.d.). The model resulting from this approach can often consist of even millions of polygons (Picture 8). These models are not usable in video games as raw sculpts, as the number of polygons would be too high and too heavy for game engines to process.



PICTURE 8. The polycount of a 3D sculpt compared to its low poly equivalent (WKU n.d.).

On top of being too heavy for the game engines, raw sculpts are prone to bad topology (Picture 9). Topology is not something that is considered in the sculpting phase, as the mesh is constantly being reorganized and subdivided. As such, in game development 3D sculpts are typically used to create baked mesh maps, which allow the high poly sculpture’s details to be projected onto a low poly model. The sculpt may also be used to help create said game-ready low poly model.

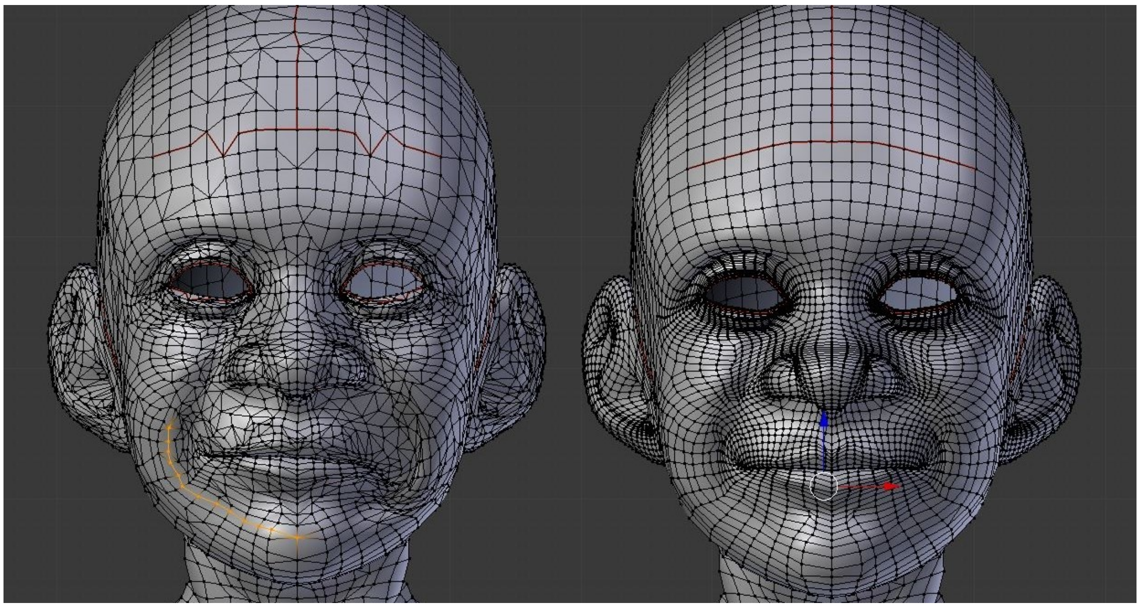


PICTURE 9. Topology on a sculpted 3D model (Noggi 2022, edited).

As 3D sculpts can consist of millions of polys, they typically make for very heavy files which can be difficult to process even for a 3D modeling program running on a powerful computer. This is why there are programs that have been specifically designed for 3D sculpting. These programs include such as ZBrush and Mudbox among others. However, with technology and software advancing, even Blender, which is an open-source 3D modeling program, is nowadays capable of advanced and professional 3D sculpting.

2.3.3 Topology

A term that is associated especially with polygonal modeling is topology. A 3D artist must pay good attention to this, and it is something new artists might struggle with. Topology is meant to describe the way the polygons are arranged on the 3D model's surface. Good topology typically has the polygons organized in a neat, orderly fashion, whereas a model with bad topology might have its polygons placed haphazardly and some of them might even overlap with one another (Picture 10).


PICTURE 10. Two head models, left displaying bad, haphazard topology and right displaying good, more organized topology (Danan 2016, edited).

It is important to ensure that your models have a good topology, as it can most notably affect the functionality of the model and the ability to animate it. Good topology ensures smooth and professional animations.

Totten (2012) states that there are three most important rules that one should follow to ensure good topology: making sure all polygons in the model are quads, joints on the model's extremities should have at least three edge loops and the model's facial geometry should have loops around openings such as eyes or mouth. Quad polygons help the modeling and game development software to figure out how to deform the model properly. Edge loops around joints and facial openings on the other hand help these parts bend and animate more naturally.

2.4 3D modeling programs

There are multiple programs for 3D modeling. Some commonly used programs in game development include such as Blender, Maya, and ZBrush. Blender is a free, open-source program developed by Blender Foundation. It was initially re-leased in 1994 and is regularly updated. Blender comes with a strong array of different tools capable of producing high-quality and professional 3D models, sculpts, and animation (Gupta 2020). As the program is completely free and funded by community contributions, it has grown in popularity. It is a popular choice for independent creators and students, as other 3D modeling programs tend to be costly.

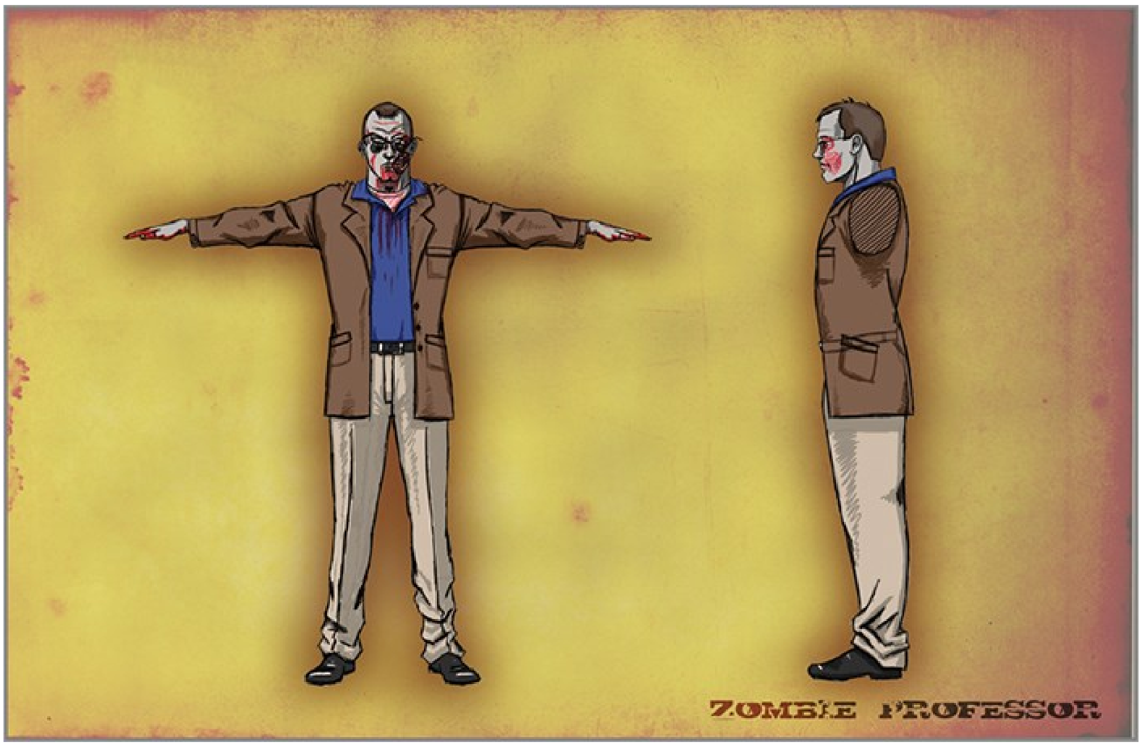
One example of a costlier 3D modeling program is Autodesk Maya, which is considered to be somewhat of an industry standard in game development. It does happen to be free for students who have an Autodesk student account, but for independent creators and developers, it can cost upwards of 280€ per month at the time of this thesis being written (Autodesk n.d.). For a creator with a limited budget, this price point can very well be too high.

ZBrush is another popular program for video game development, as it specializes in both 3D sculpting and painting. It is cheaper than Maya, with its monthly cost being about 33€ (Maxon n.d.), but it has the drawback of being mainly aimed at modeling and sculpting organic models. As such, ZBrush is not ideal for hard surface modeling or producing the low poly models needed in a video game.

2.5 Character modeling

3D character modeling has its own pipelines and workflows. These may vary depending on what type of character you're modeling, what medium the character is for, and if the character is meant to be in the background or one of the main characters. As mentioned previously, an animation character will typically have a higher polygon count than a game character, and the main player character will be more detailed than a box in the background. This reflects in their polycount.

Characters can be complex in structure, and difficult to model completely free-handed, regardless of if it is a humanoid, animal, robot, or anything in between. A good starting point for modeling a character is to have a reference for the model. These references are typically called model sheets or character sheets. They are used to help visualize the character from multiple directions and are often used in comics, animation, and video games. A model sheet used for animation or comics might contain multiple drawings of the character, typically in a pose with their arms on their sides (Clinton 2008, 53). Utilizing these kinds of sheets as references for game characters is however difficult, as some areas of the model may be hard to see or completely invisible. As such, model sheets in game development typically feature characters with their arms stretched out to their sides. The resulting character model will also share the same arm position, which, due to how it looks, is called a T-pose (Picture 11). The T-pose can enable modeling parts of the geometry at 90-degree angles, which can make the character modeling process easier (Totten 2012).

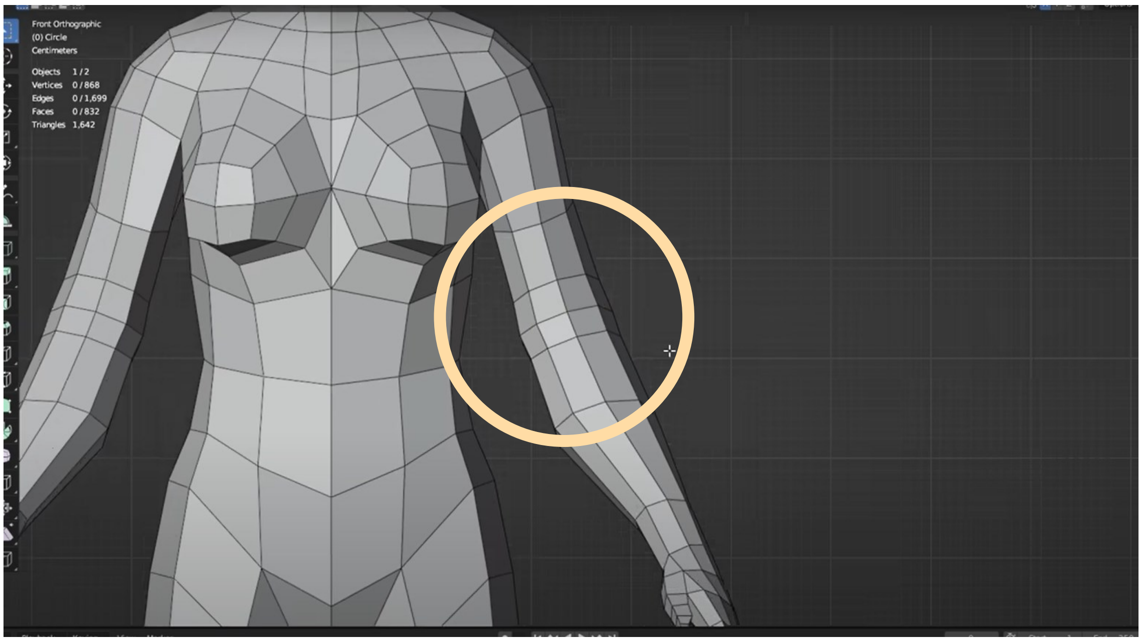


PICTURE 11. T-pose model sheet for a character (Chris Totten 2012).

Today, 3D character models for games are often produced by first sculpting a high-resolution model, especially if the game's style is aiming for realism. A low poly character model on the other hand can be modeled directly with the polygonal approach. Approaches and pipelines for polygonal character modeling vary between artists. A common approach is to start the modeling process from the character's center point, this typically being the torso. When modeling a human, a beginner 3D artist might start their model from a shape that resembles the human torso, like a sphere or a cylinder (Totten 2012). A more optimal starting point for a low poly game model however is a cube. Despite being a very inorganic shape in the beginning, a cube offers more flexibility and control over the modeling process and the model's polycount (Totten 2012). The cube primitive is also easier to subdivide than for example a cylinder, and thus extruding elements such as limbs from the cube is more straightforward.

As even low poly characters can be complex to model, it is generally recommended to utilize mirroring while modeling. In Blender specifically, this happens with a mirror modifier. The modifier selects an axis along which it will mirror the model, and all the changes made to one side of the model will appear on the other. This essentially cuts the modeling workload in half and makes the process much more efficient.

When modeling a character that is going to be animated, there are certain aspects in the process that the 3D artist must keep in mind. One such feature is the character's joints. To ensure that the joints bend properly and do not distort the model in unwanted ways, additional loop cuts need to be added to the joints (Picture 12). It is generally recommended to add three loop cuts per joint on the model's extremities. Loops are also utilized when modeling facial geometry. As mentioned previously, loops around the facial openings help the model's face distort better for animation and as mentioned by Totten (2012), enable the "mouth and eye movements to look more natural".

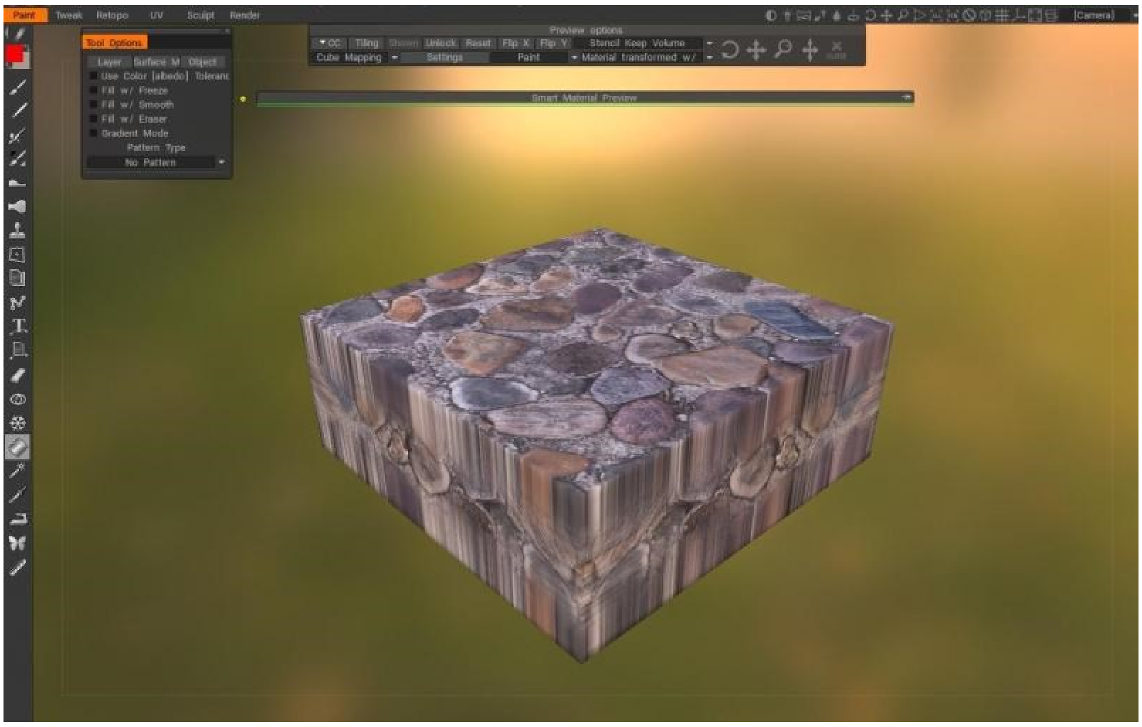


```
Objects    1 / 2
Vertices   0 / 868
Edges      0 / 1,699
Faces      0 / 832
Triangles  1,642
```

PICTURE 12. Three loop cuts on a 3D character's elbow joint (Thomas Potter 2022, edited).

3 TEXTURING

Once a 3D game model is produced, the next phase is to texturize it. Textures are 2D images placed on a 3D model, and they are an important part of 3D models because, as stated by Clinton (2008, 8), they "will define what the rendered surface will look like". A texture artist must keep in mind the fact that the flat shapes they are creating will be placed on 3D forms (Ahearn 2016). Adding textures to a 3D model is not necessarily straightforward, however, as a 2D image will not automatically follow a 3D shape without guidance. This can cause unwanted texture stretching to appear on the model (Picture 13). Thus, textures need to be mapped onto the 3D shape. This is often done by UV unwrapping the model, a mapping process in which the 3D model is taken apart and flattened out onto a 2D surface. This method will be further elaborated upon in chapter 3.2.



PICTURE 13. Texture stretching on the edges of a 3D model (Grzegorz Więcek 2019).

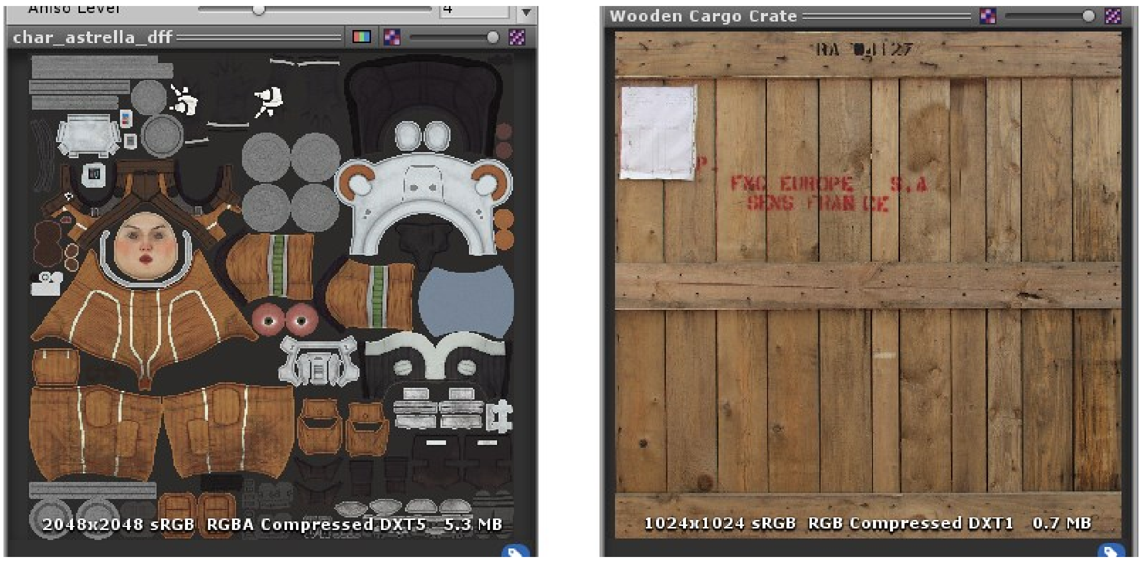
3.1 Texture maps and PBR

A 3D model can be textured in many ways. The chosen approaches for the texturing process may depend on such things as the style of the game, what type of model the texture is made for, production costs, and efficiency. A common approach for video games, especially those aiming for hyper-realism, is to utilize PBR in their texturing. PBR stands for Physically Based Rendering, and it is a shader system where the way light behaves on a surface is simulated in the way it does in the physical world (GarageFarm 2021). How light behaves and interacts with its environment can make or break a scene when it comes to photorealism. This is why PBR is a popular technique for handling textures and shading. By utilizing PBR, a texture artist can, according to Adobe (n.d.), achieve realistic-looking assets, environments with cohesiveness no matter the lighting conditions, and a sustainable and faster workflow.

PBR utilizes different image versions of the base 2D texture image to calculate things like emission, light reflection, depth, and metallic values for the final, compiled version of the texture. These images are called maps, and they can be applied to 3D models even outside of the PBR workflow. Totten (2012) introduces seven different texture map types, which are

- color maps
- bump maps
- normal maps
- alpha maps
- specularity maps
- reflection maps
- illumination maps.

All these maps serve a separate purpose when texturing the model, and they can be used independently from one another. For example, in Metallic Roughness Workflow, PBR texturing utilizes color, normal, metallic, roughness, height, and ambient occlusion maps (GarageFarm 2021). The color map (Picture 14), which is also known as the albedo or diffuse map, is the most basic type of texture map (Totten 2012), and a very commonly used one in game development. As the name suggests, it is the map that gives the 3D model its colors. These colors can be anything between a single color, an elaborate hand-painted texture, or an image texture made from a photograph.


Aniso Level
char_astrella_dff
2048x2048 sRGB RGBA Compressed DXT5 5.3 MB
Wooden Cargo Crate
1024x1024 sRGB RGB Compressed DXT1 0.7 MB

PICTURE 14. Two different color maps (Unity Documentation n.d., edited).

Another important map often used in 3D games is the normal map. As stated by CG Masters (2020), normals are “the direction of a face or a vertex” (Figure 1). A normal map is a 2D image map with various RGB values, with each color value representing a normal facing a specific direction in a 3D space. A 3D program or game engine can project the data presented by these RGB values onto a 3D model’s surface. This then causes the light hitting the model to bounce back according to the direction of the normals on the normal map.

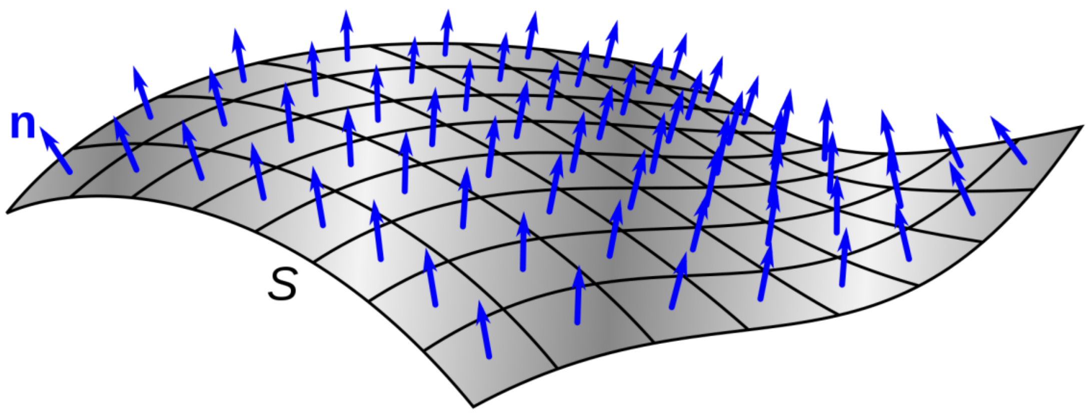
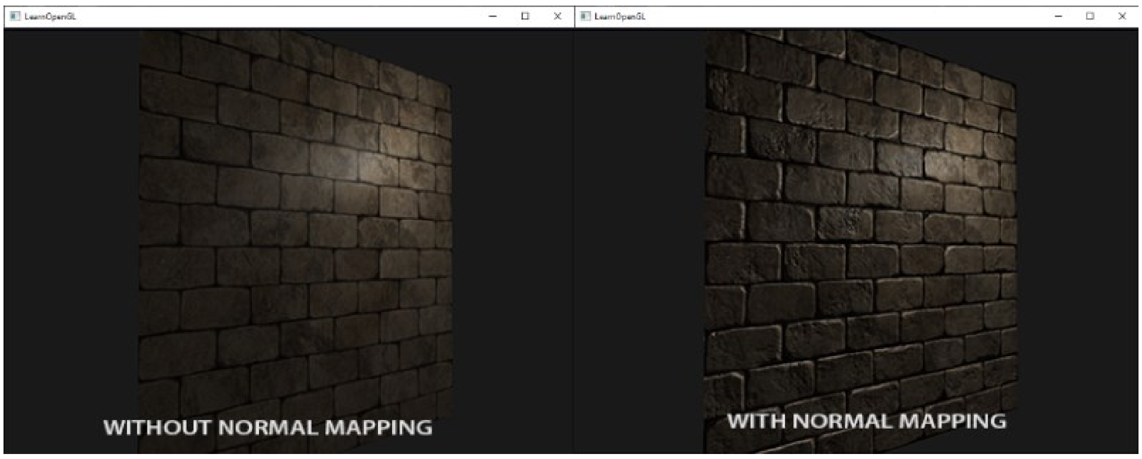
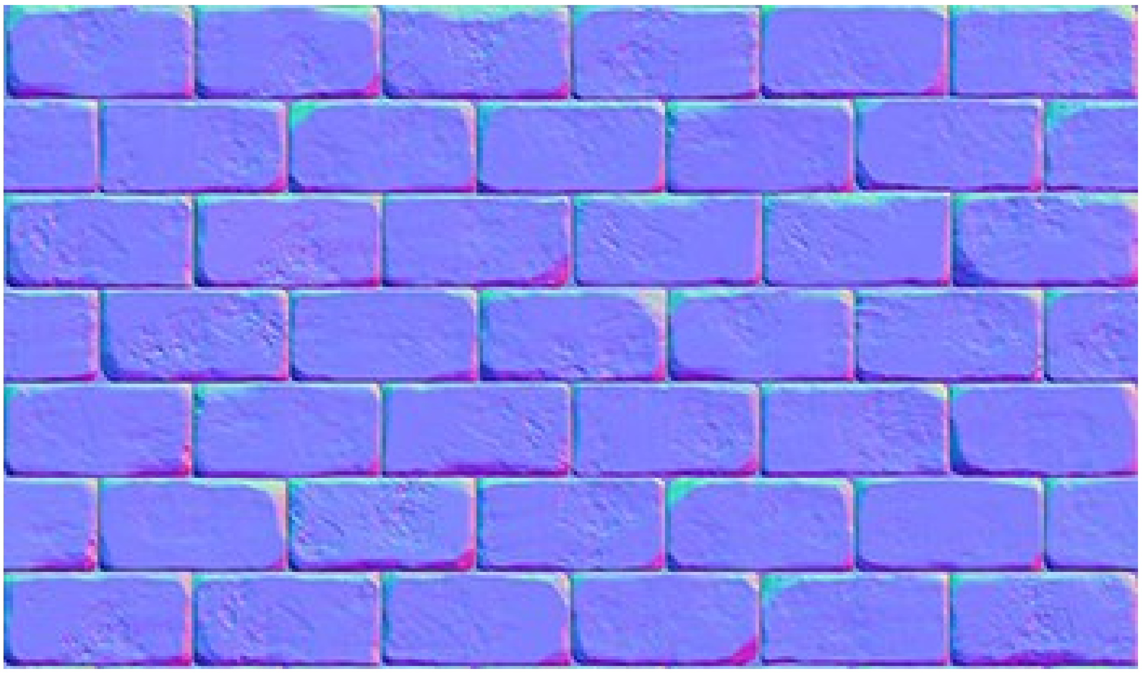


FIGURE 1. A curved surface with blue arrows indicating the direction of the surface sections’ normal (Chetvorno 2019).

As can be seen in Picture 15, normal maps are used in game design to make the low poly in-game models appear more detailed and higher in resolution than they actually are (Totten 2012). Normal maps are typically created by utilizing a more detailed high poly mesh, analyzing it, and translating its surface properties onto a 2D image (Picture 16) through a process called texture baking.



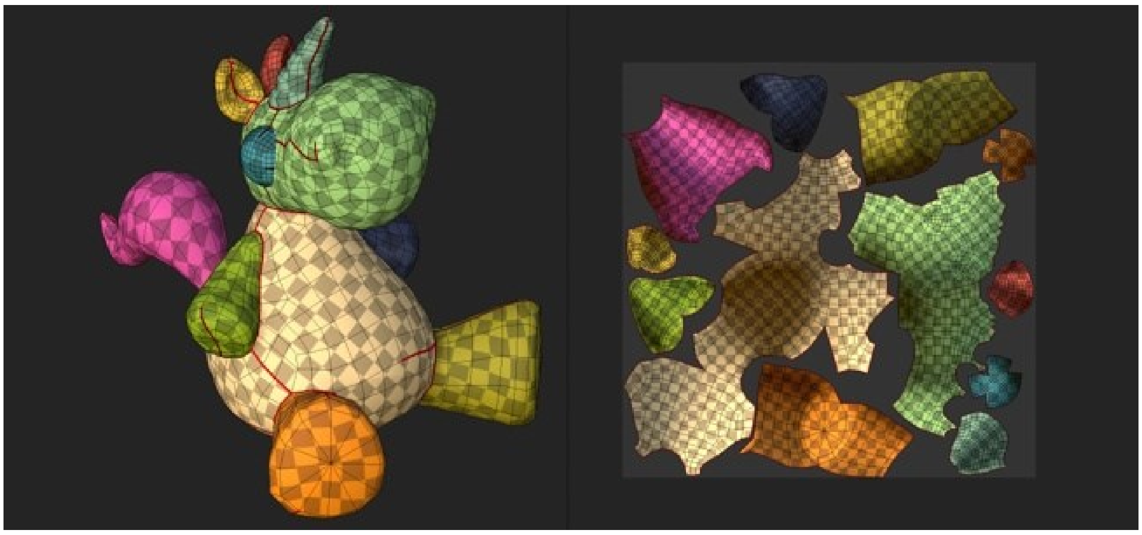
PICTURE 15. The same wall texture without and with an applied normal map (Joey de Vries, n.d.).



PICTURE 16. Normal map for a brick wall (Joey de Vries, n.d., edited).

3.2 UV mapping and unwrapping

UV mapping and unwrapping are an integral part of the texturizing process. There are different ways to map textures onto a 3D model, but UV mapping is probably the most accurate of them. UV mapping is also popularly used in game development when adding textures to assets and models, as it is a way to custom-fit a texture onto a model. A UV map is essentially a 2D projection of a 3D model, with the model being opened and flattened out (Picture 17). This flattening process is called UV unwrapping.


PICTURE 17. 3D model and its UV map (Adobe 2022).

When unwrapping a model for UVs, it is basically taken apart. According to Totten (2012) the UV unwrapping process “is analogous to cutting a paper doll apart at certain seams and then laying the pieces flat for coloring”. When unwrapping a model seams are placed down, which then determine along which edges the model gets cut open. While there are no universal rules as to how you should place these seams and unwrap your model, there are some established best practices. For example, it is good to try and hide the model’s seams in places where they are the least visible, for example alongside natural clothing seams or other edges, or underneath the model. This ensures that if the texture does not tile perfectly along the seam, it will not be excessively visible.

It is also important to make sure none of the separated UV sections, called UV islands, stretch too much after the model has been unwrapped. This might happen for example when trying to unwrap complex geometry in one large piece. A stretched UV will make the applied texture map also appear stretched and thus, poor quality. The size of the UV islands can also be adjusted to be bigger or smaller. This can be done to give more pixels for important or detailed parts of the model, such as a face or a decorated piece of armor.

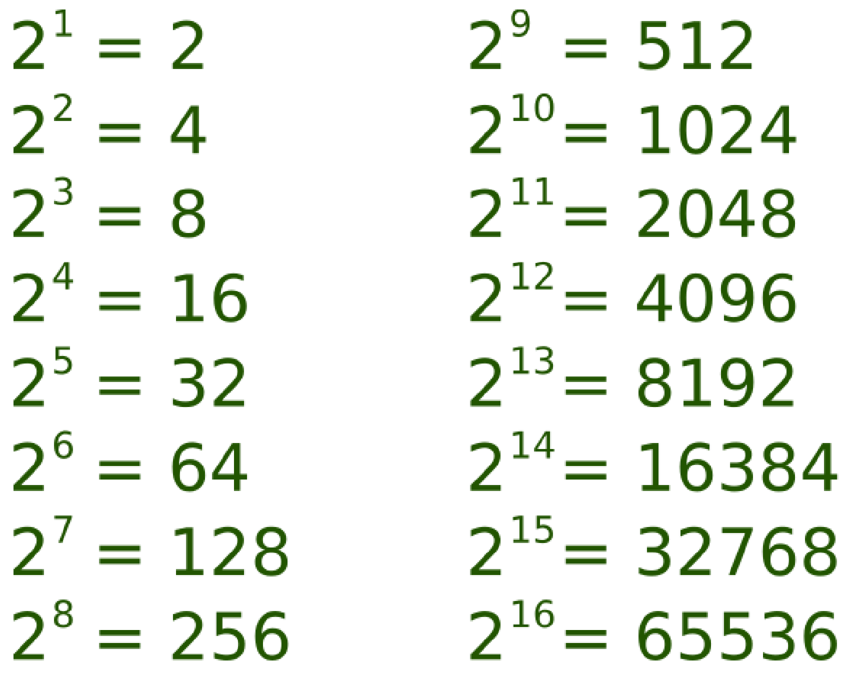
3.3 Optimizing textures in game development

When producing textures for a video game, an artist must again follow certain rules and limitations the medium brings with it. As in the case of the 3D model, an important part of producing efficient and game-ready textures is optimization. Optimizing texture assets can come at a cost, with the common trade-off being between what looks good and what runs well (Ahearn 2016). With game textures having limitations in their size and resolution, they inevitably suffer from some loss in fidelity and details. However, as mentioned by McBride (McBride, Clarke, Gonzalez 2018), this can also give a unique beauty to game art.

In game development, optimization is about saving space and computing power. A good way to do this when working with textures is by keeping the texture sizes small. Ideally, a texture file’s size is kept approximately between two and four thousand pixels on the longest side. The larger, four-thousand-pixel-wide size is generally reserved for the most important and visible textures, such as the one on the game’s main character. Less important models’ textures, such as ones for crates or other background items, can easily be kept as two thousand pixels wide or smaller, as the player will not be looking at them closely for long. It is however rare to see textures under 1024 pixels wide nowadays. These kinds of small textures are used typically for particles and small details (Ahearn 2016).

3.3.1 Power of two

A general practice in producing textures and other image assets for games, on top of keeping their sizes as small as possible, is to make their sizes follow the power of two (Table 1). According to Ahearn (2016), having your textures follow this sizing is important because “most computer game engines and 3D cards require a texture to fit certain criteria because they can process and display them faster”. Power of two also makes it easier to keep assets uniform and to place them accurately without having to eyeball them, as it is often used as a standard sizing method in game development.



$2^{1} = 2$	$2^{9} = 512$
$2^{2} = 4$	$2^{10} = 1024$
$2^{3} = 8$	$2^{11} = 2048$
$2^{4} = 16$	$2^{12} = 4096$
$2^{5} = 32$	$2^{13} = 8192$
$2^{6} = 64$	$2^{14} = 16384$
$2^{7} = 128$	$2^{15} = 32768$
$2^{8} = 256$	$2^{16} = 65536$

TABLE 1. Powers of two up to 16 (Martin McBride 2016, edited).

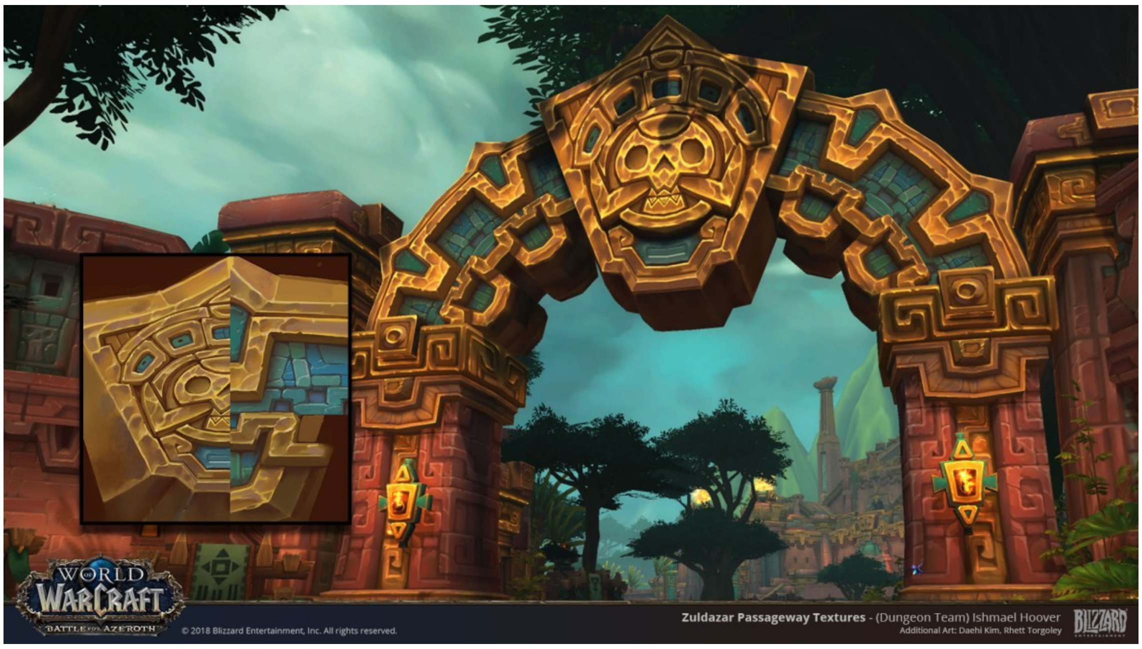
It is also recommended for your texture files to have square proportions, especially when working with the power of two sizing. This is helpful when creating grids and texture atlases. Texture atlases are singular images with multiple different textures in them. They are used to help organize the game project’s files and to minimize file space usage within the project.

3.3.2 Work big, save small

In game art, especially with textures, the original source files are also almost always bigger than the eventual game-ready files. This is because it is good practice to essentially work big and save small. The source file may contain lots of details, it can be flexible with adjustments, and overall, it saves a large amount of information. This is good while still working on the source file as adjusting the file is easier but can be detrimental to optimization with the finalized file in the game project itself. As such, as mentioned by Ahearn (2016), game-ready files are typically resized, optimized, and converted. The specific settings for the file conversion may vary. Some basic criteria for file conversion are such as what looks the best, what the technology or system requires and supports, and what your superiors may tell you to use (Ahearn 2016).

3.4 Hand-painted textures

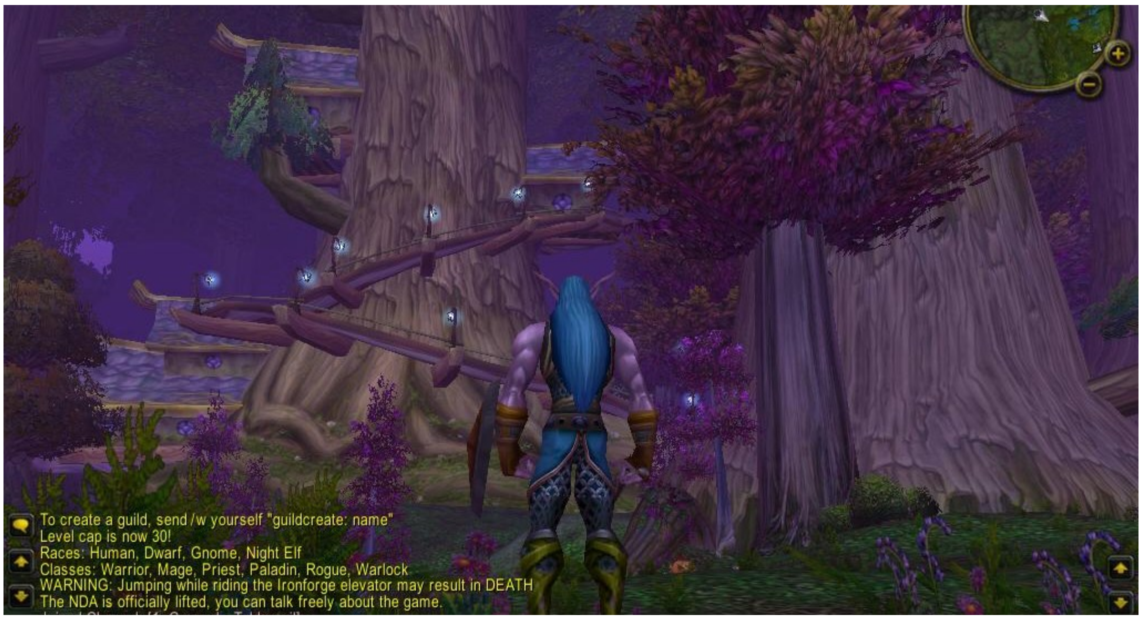
Game textures are often created by utilizing procedural texturing or normal and bump maps from a high poly base. This has not always been the case. Prior to normal maps, 3D sculpting, and texture baking, textures were mainly hand-painted or put together in programs like Photoshop (The Scout 2019). As technology has developed, hand-painted textures have decreased in popularity. Unlike before, some surface properties like reflections, specular highlights, and bump mapping are nowadays processed in real-time and no longer painted on the model. Having these be processed in real-time enables games to achieve levels of realism that were previously not possible. (Ahearn 2016). Despite this, hand-painted textures have not disappeared completely. While the demand for hyper-realistic games with extremely detailed normal mapping and PBR textures is rising as technology advances, some games choose to go with a more stylized look. For some of them, like Activision Blizzard’s World of Warcraft (Picture 18), the hand-painted texture style is the way to go.


WORLD WARCRAFT BATTLE FOR AZEROTH
© 2018 Blizzard Entertainment, Inc. All rights reserved.
Zuldazar Passageway Textures - (Dungeon Team) Ishmael Hoover
Additional Art: Daehi Kim, Rhett Torgoley
BLIZZARD

PICTURE 18. Hand-painted textures from World of Warcraft's "Battle for Azeroth" expansion (Ishmael Hoover 2018).

The benefit of hand-painted textures lies in the ability to highly stylize the texture's final output. The textures do not need to be realistic or lifelike, which in turn can give them more personality and visual interest compared to for example a hyper-realistic PBR texture. Hand-painting textures is also a way to give them an authentic painterly look. This is something that can be somewhat imitated by using nodes and other procedural generation tools. However, as mentioned by Faustine Dumontier, the Art Supervisor at Fortiche Production SAS, the best way to texturize a model and have it look like a painting is to paint (Arcane 2022a).

Even as technology develops and new and improved production methods emerge, hand-painted and stylized textures remain relatively ageless. The game World of Warcraft was first released in 2004, and whilst Activision Blizzard has developed and improved upon their texturing approaches, the original textures from the game work well to this day (Picture 19).

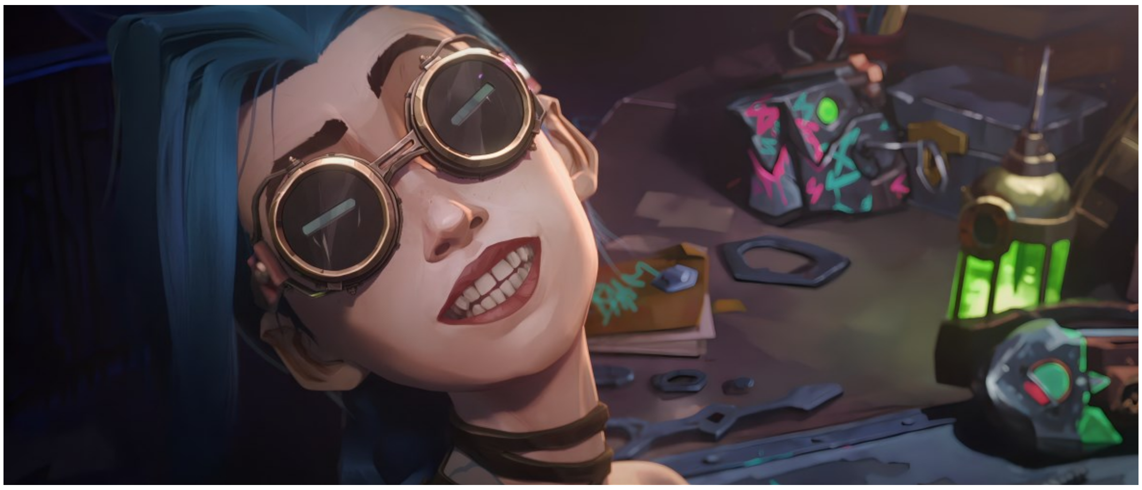

To create a guild, send /w yourself "guildcreate: name"
Level cap is now 30!
Races: Human, Dwarf, Gnome, Night Elf
Classes: Warrior, Mage, Priest, Paladin, Rogue, Warlock
WARNING: Jumping while riding the Ironforge elevator may result in DEATH
The NDA is officially lifted, you can talk freely about the game.

PICTURE 19. World of Warcraft in 2004 (Greg Kasavin 2004, edited).

Thomas Vu, a producer on the award-winning animated television series Arcane, describes his thoughts on hand-painted textures,

> “Because of their hand-craftedness and the painter-y look, it gives this timeless look that I think even when the technology evolves, that this will still feel lasting. You know in the same way I think I can go back and watch like the Disney animations like Bambi, or you know, like Cinderella, and it’s timeless. It doesn’t age because it’s hand-crafted.” (Arcane 2022b).

Arcane is an animated television series released in 2021, created by Christian Linke and Alex Yee for Netflix. The series was animated by Fortiche Production SAS, a French animation and production company known for its stylized and experimental combination of hand-painted textures and 3D. Arcane has received a great deal of praise for its distinctive painterly visuals (Picture 20). According to Pascal Charrue, co-founder of Fortiche, with Arcane Fortiche wanted to exceed the quality that could be achieved with 2D animation, and this was made possible by combining the stylized, hand-painted look with 3D (Arcane 2022b). This kind of combination is also possible in video games, as games like World of Warcraft, Hades, and Torchlight have shown, albeit not necessarily to the same level of intensity and detail. This is due to some of the previously mentioned differences and limitations between pre-rendered animation and real-time rendered video games.



PICTURE 20. Arcane (Netflix 2021).

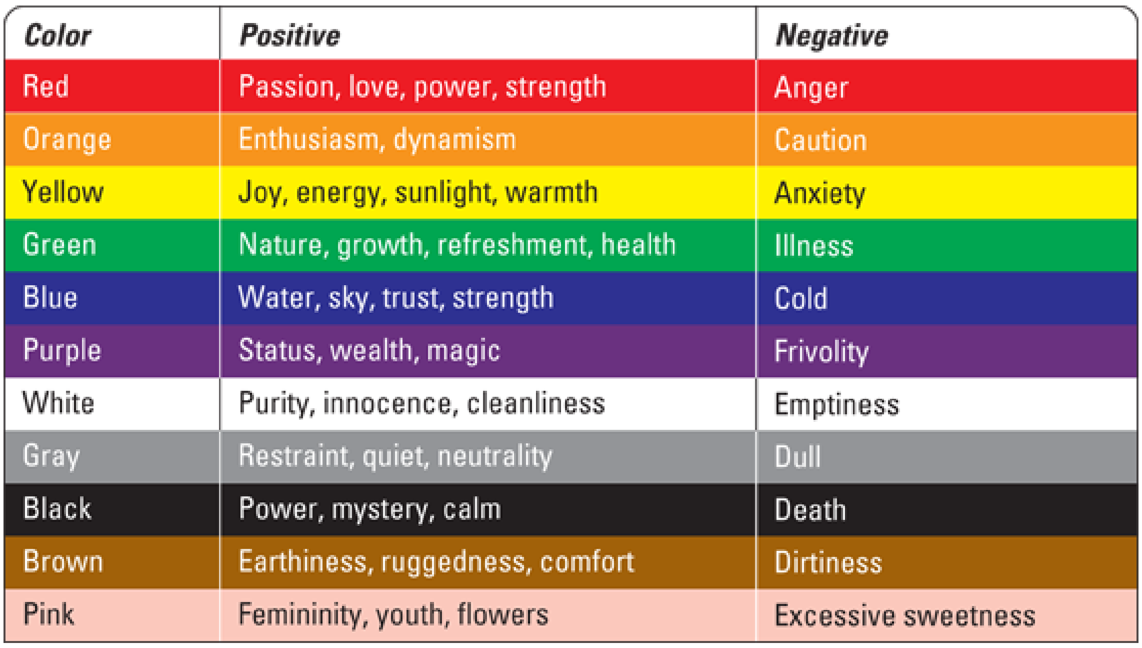
3.4.1 Real-life principles in texture painting

Like any artist, a texture artist needs to understand how different objects, materials, and surfaces function in real life. Even highly stylized textures need to have a basis in reality. The human eye can spot unrealistic inconsistencies, even if the viewer does not fully understand exactly what looks wrong in what they are looking at (Ahearn 2016).

One important aspect of hand-painting textures is light and shadow. Whilst game engines are capable of rendering lighting in real-time, some of the shadows and highlights still need to be painted onto the texture itself to give it believable depth and character. In all of this, the artist also must make sure the lights and shadows are not too strong in the texture. Textures need to be versatile and able to fit in multiple different lighting conditions (Ahearn 2016), and intense shadows and lights can inhibit this. Harsh shadows and highlights can also make the texture look undesirably flat as the player moves around it in-game (Ahearn 2016). The direction of the light on the texture is also important in order to achieve believable results. Conflicting light sources produce previously mentioned inconsistencies that the viewer will be able to see.

References are an important tool for a 3D texture artist. Like with lighting, in order to stylize different materials and surfaces, an artist needs to know first how they look and function in real life. Utilizing reference images can be a great aid in this endeavor. Being able to see and study what a rusted piece of metal or a rotten wood plank looks like in real life helps reproduce the material in a more stylized manner.

Humans rely on their vision and are capable of making decisions based on what they see, even subconsciously. Thus, visuals have the potential to influence the viewer. One aspect of such visuals is colors. As seen in Table 2, colors can be associated with things, ideas, and emotions (Hibit 2022), and this is often utilized in game development.



Color	Positive	Negative
Red	Passion, love, power, strength	Anger
Orange	Enthusiasm, dynamism	Caution
Yellow	Joy, energy, sunlight, warmth	Anxiety
Green	Nature, growth, refreshment, health	Illness
Blue	Water, sky, trust, strength	Cold
Purple	Status, wealth, magic	Frivolity
White	Purity, innocence, cleanliness	Emptiness
Gray	Restraint, quiet, neutrality	Dull
Black	Power, mystery, calm	Death
Brown	Earthiness, ruggedness, comfort	Dirtiness
Pink	Femininity, youth, flowers	Excessive sweetness

TABLE 2. Positive and negative associations with colors (Eric Hibit 2022).

Color associations can be used in many ways in games. A common way to utilize them is to affect the player's decision-making with areas or objects. If a chest in the middle of a room is surrounded by a red glow, the player might steer away from it, expecting it to be dangerous, whereas if the glow is green or yellow, the player will most likely approach the chest and anticipate a positive outcome. This color association can also be utilized in textures. An antagonist character may have black and red hues in the color palette of their texture and a healer could have browns, greens, and whites in theirs.

3.4.2 Digital painting

Painting textures by hand is very akin, if not almost identical to overall digital painting. This is especially highlighted if the texture painting happens in a digital painting program as opposed to a 3D texture painting program like Substance Painter. Thus, it is advantageous for a 3D texture artist to have a strong base in digital painting, its programs and tools, and overall work methods.

Digital painting is typically done by utilizing drawing tablets, which help the artist draw and paint on the screen like one would on paper. The painting itself happens inside a painting program, such as Photoshop, Clip Studio Paint, or Procreate. These programs are built for digital art and painting, and provide a wide array of tools, brushes, adjustments, filters, and other helpful gadgets for the painting process. The previously mentioned programs are all behind a paywall, with Photoshop and Clip Studio requiring a monthly or annual fee. Nowadays there are also very capable free programs for digital painting. These programs include such titles as GIMP and Krita.

Achieving an organic and hand-made feeling with digital painting can be difficult and generally requires experience with the available tools. Adding filters like graining or scratches on top of an illustration can help, but the most important tools a digital artist can use to make their piece feel hand-painted are the brushes. In painting programs such as Photoshop or Clip Studio Paint, the brush tools' features like opacity, angle, and shape can be adjusted freely. A brush can even be made to look like an image, which will then repeat in the brush strokes (Figure 2). Features like visible brush strokes, imperfections, and uneven surfaces can make the digital illustration feel more organic and painterly (Picture 21) compared to a highly polished and smooth surface (Jones 2019).

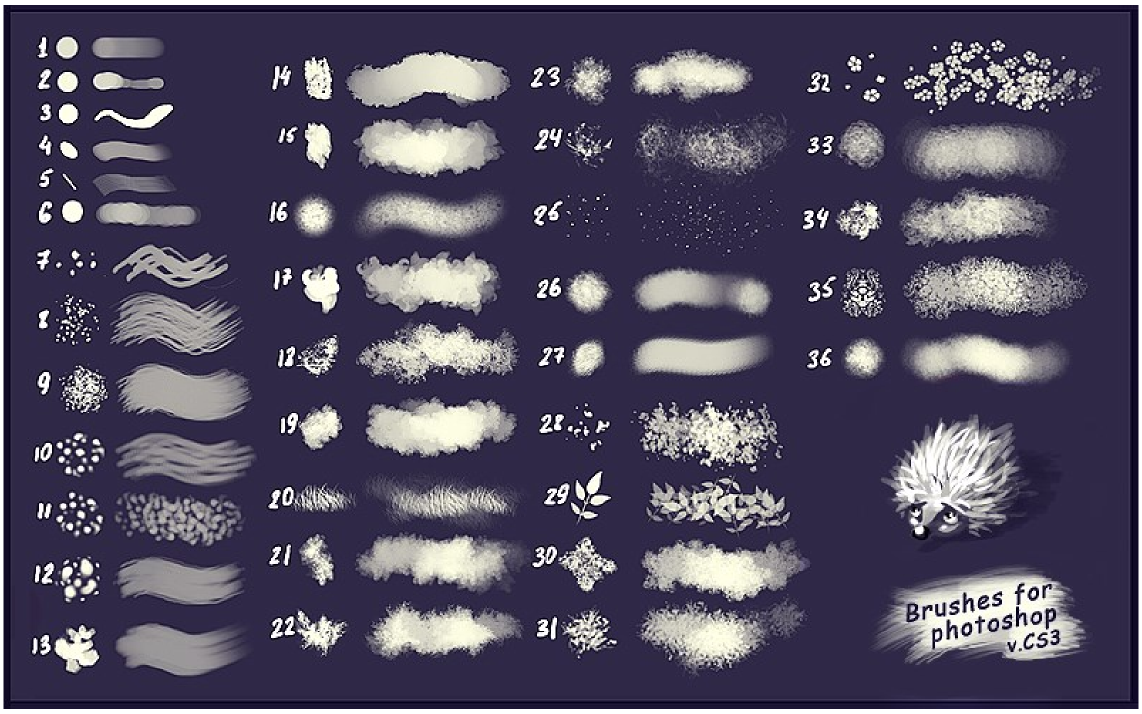


FIGURE 2. Photoshop brush shapes and their respective strokes (Mar-ka 2010).

PICTURE 21. Digital paintings produced with paintbrush-like brushes (Aaron Griffin 2016).

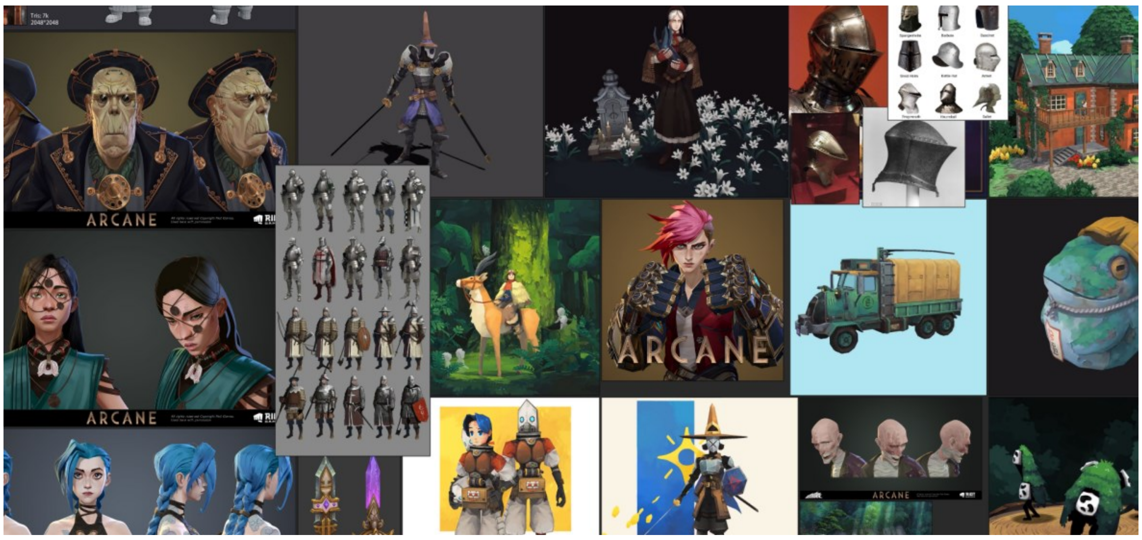
4 PRACTICAL PROJECT

The main objective of the practical project was to study the creation of hand-painted textures and how they would work together with a low poly 3D model. To achieve this, a low poly character was designed and implemented into a 3D model. The model was then exported to a UV map and an FBX model for texture painting purposes.

4.1 Character design and model sheet

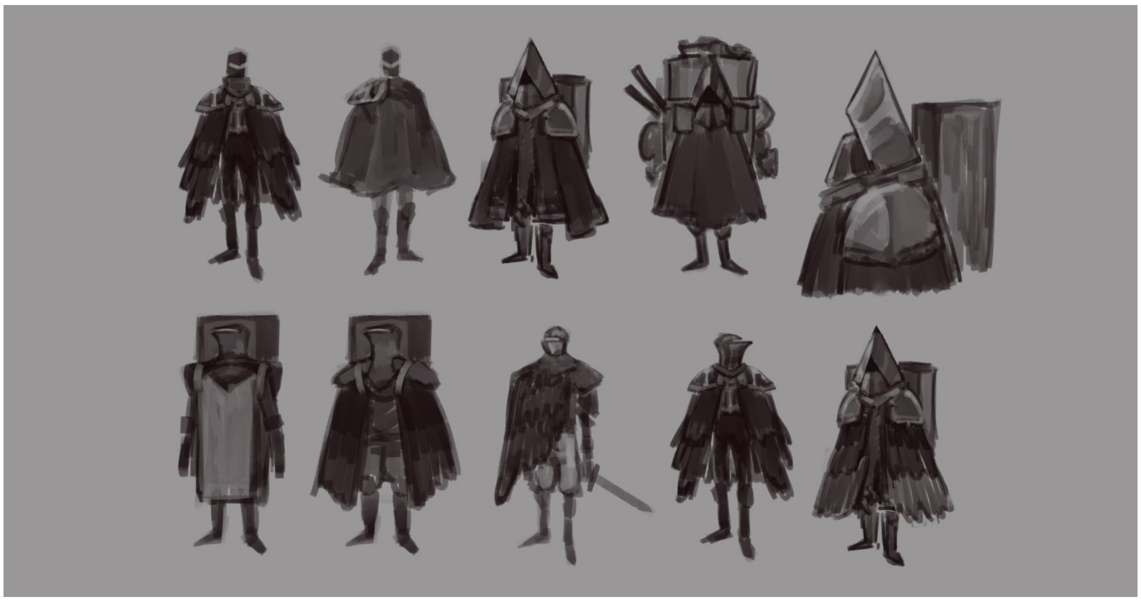
The first step of the project was to come up with a character design that would fit within the parameters and prerequisites set by the project's main objective. With the 3D model needing to have a low polycount, the character design needed to have a simple-enough shape language to facilitate this.

The first step in the design process was to gather references (Picture 22). This helped guide the design into a more uniform direction both in terms of the shape language and what would eventually be required in the texturing process. The gathered references consisted of various 3D models with stylized or hand-painted textures, as well as photographs or paintings of different inspiring subjects.

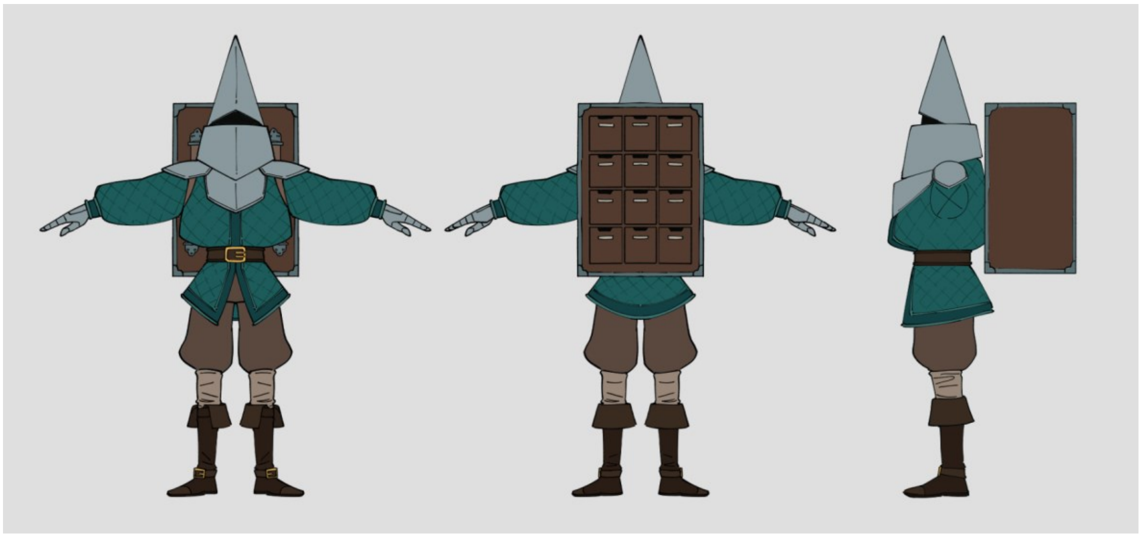


PICTURE 22. Reference board for the character designing process.

The next step after gathering references and figuring out which direction the design would be taken, was to start sketching the character. The approach chosen for this was to sketch various character silhouettes (Picture 23). This helped in making sure the design's shape language would stay simplistic but still have readability in it. The idea was for the character design to also feature multiple different materials in it. This was to make the eventually painted textures both pop out and be more interesting to create.


PICTURE 23. Character silhouettes with some detail refining.

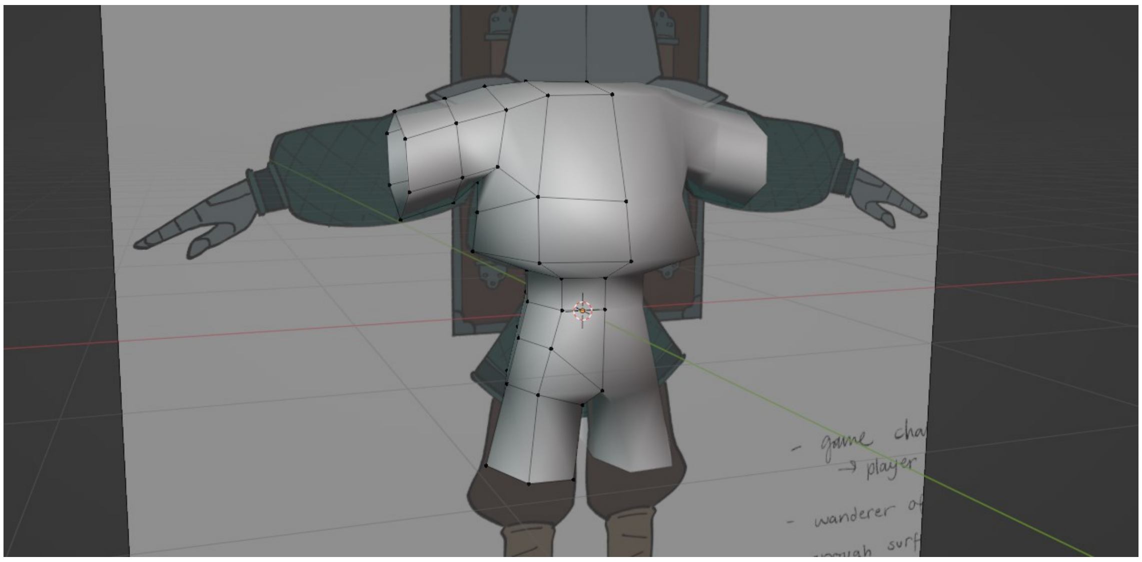
The final chosen design was a character with a cone-like helmet, shoulder armor, and a large, wooden box on their back. Originally the character was also meant to have a cape akin to the silhouette sketches, but the idea was scrapped after it proved to interfere too much with the wooden box. The final design thus lost the original A-shaped silhouette it had, but this gave room to focus on parts of the model that the cape would have covered. The character was meant to feel like a non-threatening traveler a player might encounter in a video game. As such, the color scheme of the design featured various neutral shades of muted teals and browns, complimented by the armor's silvery metal and a few golden details. These ideas were put together in the form of a model sheet (Picture 24), which would then be used as a reference for the character's 3D model.


PICTURE 24. Model sheet of the character.

4.2 Producing the 3D character model

The 3D model of the character was produced in Blender. The model sheet was exported into Blender as a reference object to provide guidelines for the modeling process. As the aim of the project was to create a low poly model directly without utilizing a sculpted base or any other kind of high poly mesh, the modeling process started from a single cube primitive. A mirror modifier was applied to the cube along the X-axis to ensure the character model would be perfectly symmetrical along its center line.

As a starting point, the cube primitive was manipulated to resemble the shape of the character’s upper torso. From there it was subdivided as needed and additional portions of the body, such as the limbs or more of the torso, were extruded from it (Picture 25). At first, the geometry was kept as simple as possible to ensure a low polycount without paying attention to details such as joint geometry. The aim was to first get the main portions of the geometry in place, and only after that were additional details like the hems of the jacket extruded.

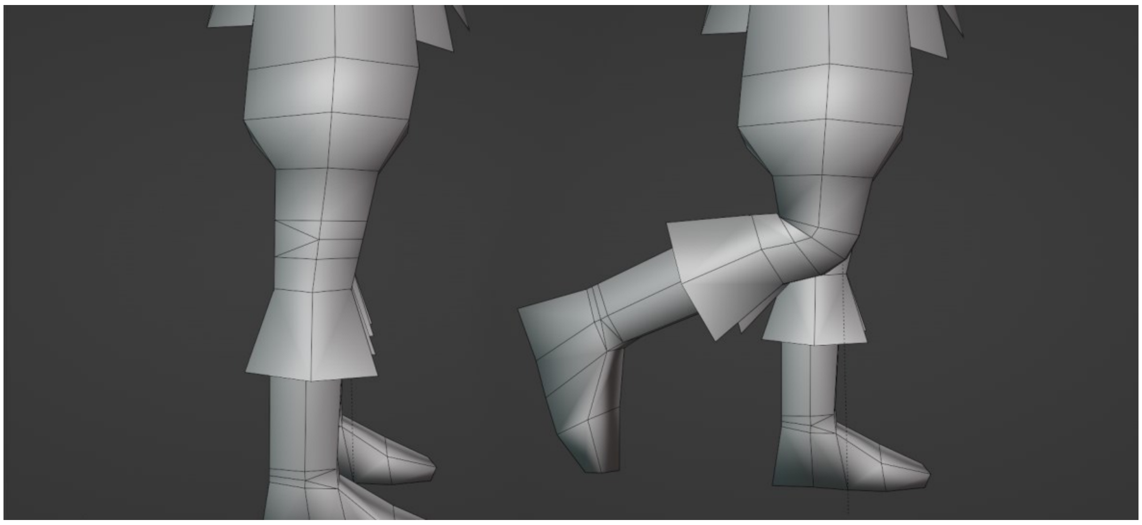


PICTURE 25. Start of the modeling process of the character’s body with the mirror modifier enabled.

All vertices in a 3D model do not necessarily need to be connected, and as such, modeling some parts separately is often easier. As such, some portions of the model were modeled completely separately from the main body instead of being extruded from it. These included the armor pieces, the character’s hands, and the wooden box. For example, the cone-shaped helmet started as a separate cylinder object, which was then merged from one end to achieve the desired shape. The hands were modeled from a cube primitive and duplicated identically onto the opposite side of the model with the mirror modifier. The armor pieces were achieved similarly, with their original shape being a single plane. These objects were then merged together with the main character model object.

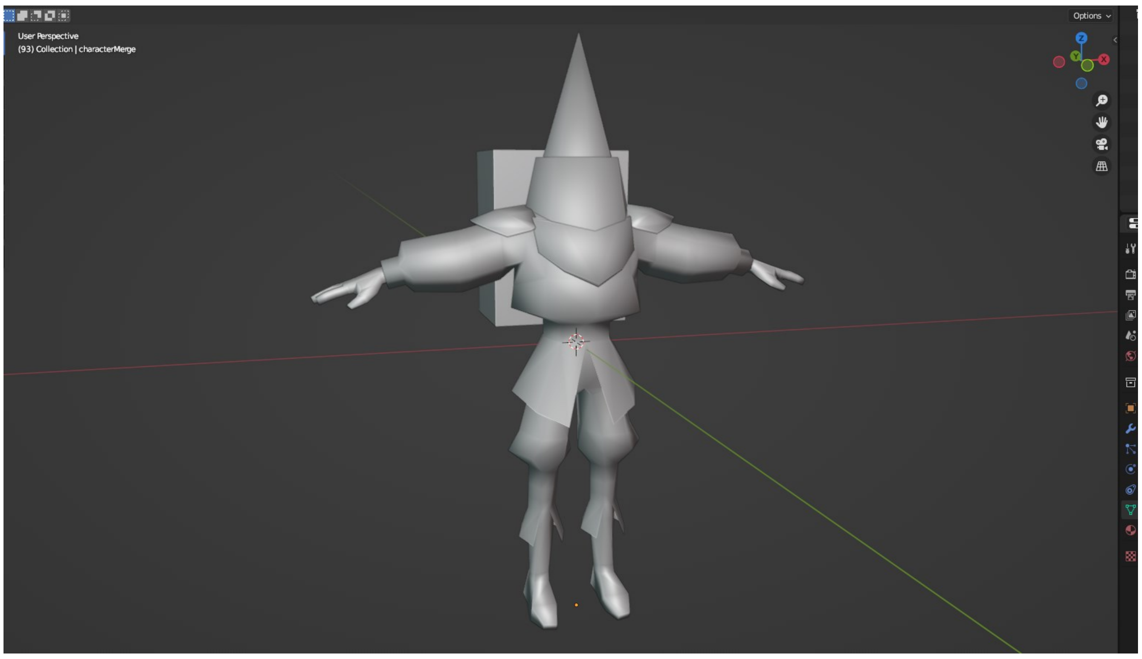
After the 3D model’s main geometry was finished and all of its pieces were in the right place, some final detail adjustments were required. Firstly, it was ensured that all the normals on the model were facing the right direction, that being towards the outside of the model. Any normals facing inwards would be invisible inside a game engine, and as such those that were needed to be flipped. These included the hem on the character’s jacket and the toplines of the boots. This was most likely due to how the geometry in these areas was extruded from the main body.

As the model needed to potentially work as a 3D game model and be animated, additional loop cuts were added to its joints to facilitate better deformation. These loop cuts were then merged on the inside of the joint. This would allow for the mesh to collapse in on itself when bent, providing a more realistic deformation on the joint (Picture 26).



PICTURE 26. A collapsible joint mesh on the 3D model's knee.

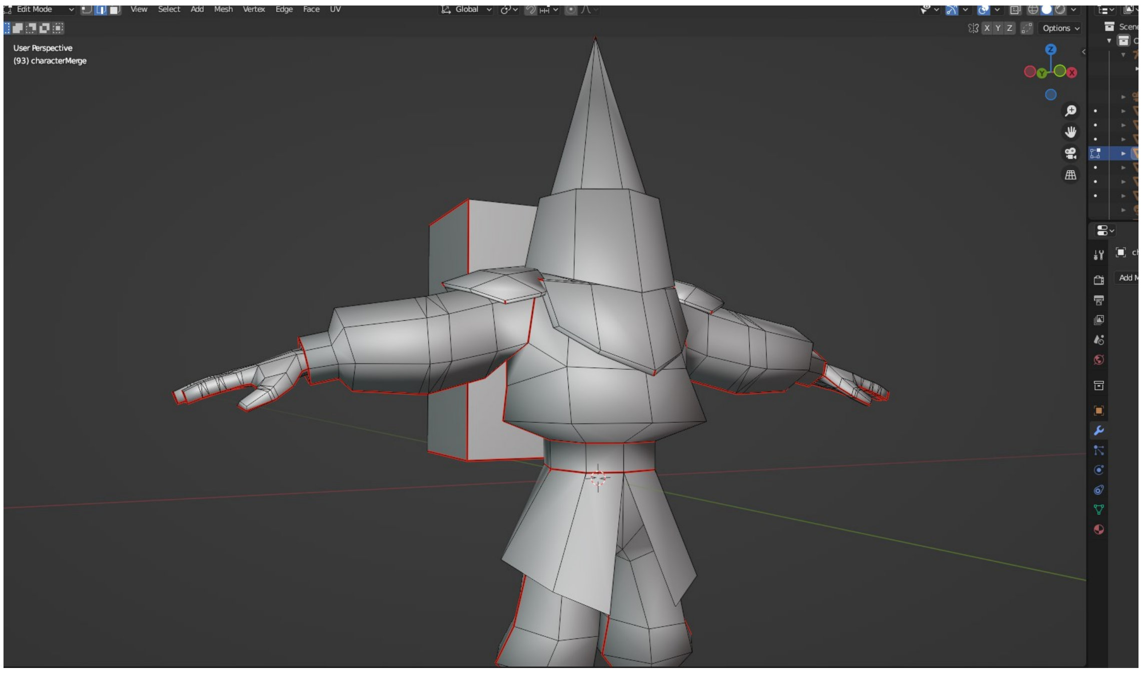
Once the final loop cut adjustments and normal redirections were finished, the mirror modifier was applied to the model. Any unnecessary vertices still lingering in the center of the model were cleaned, resulting in a final 3D model (Picture 27) that was ready for UV unwrapping. The model's polycount was 1496 polys, which is very low by today's technological standards.



PICTURE 27. Finalized 3D character model.

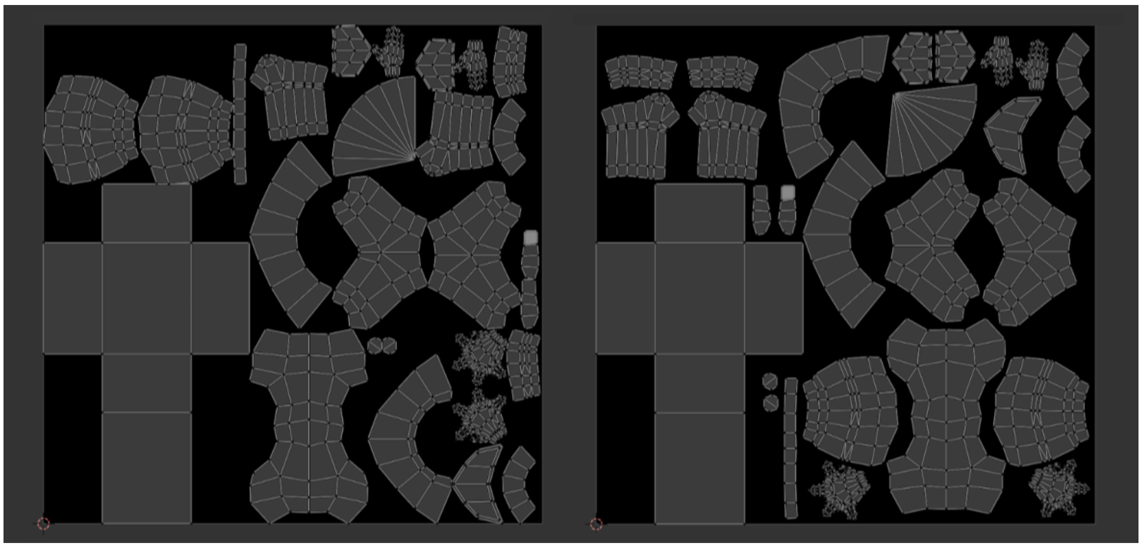
4.3 Preparing the model for texturing

To prepare the 3D model for texturing, it needed to be UV unwrapped. The first step in this process was to place down the seams where the mesh would then be opened (Picture 28). To keep the UV map as uniform as possible, these seams were placed symmetrically on the character along the X-axis. On top of this, they were also placed in spots where possible inconsistencies along the seams would not be as noticeable. These included such areas as the underside of the arms, along the inside of the model's legs, and the bottom of its feet. Seams were also placed where one clothing item would switch to another, which can also help in hiding the harsh seam edges.



PICTURE 28. Seams on the 3D model's mesh are colored red.

Once the seams were placed on the model, it was unwrapped by utilizing Blender UV Mapping's "Unwrap" -command. The UV map resulting from this was satisfactory from the get-go, but it was adjusted slightly to better fit the hand-painting workflow (Picture 29). To do this, the UV islands were grouped together by material type, where they were located on the model, or where they fit the best. For example, the islands from the jacket were grouped next to one another. The islands were also moved away from one another to give more leeway for the painting process and to ensure the textures would not accidentally overlap.



PICTURE 29. Unedited UV map from Blender on the left, edited and organized UV map on the right.

4.4 Texturing the character

Two textures were produced for the 3D model in two different programs while utilizing a drawing tablet for the painting process. The programs used for the texturing were Clip Studio Paint EX and Substance Painter. These programs represent different ends of the texture painting process. Clip Studio highlights the more traditional method of producing textures as flat 2D images from the get-go. Substance Painter on the other hand presents the more modern approach of painting the texture directly on the 3D model itself.

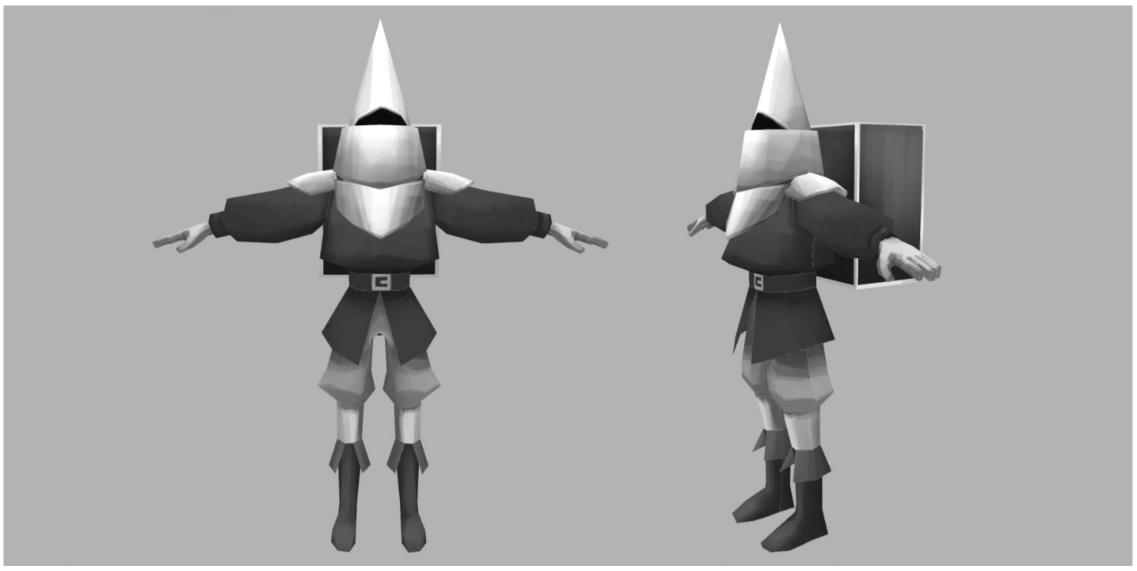
These specific programs were chosen for this project to analyze the differences in their overall workflow. They were also studied based on how they would fit the hand-painting workflow, and if they would be capable of producing the desired texturing outcome. The aim of the textures themselves was to highlight a painterly style and feature different materials and surface textures in said style.

4.4.1 Clip Studio Paint EX

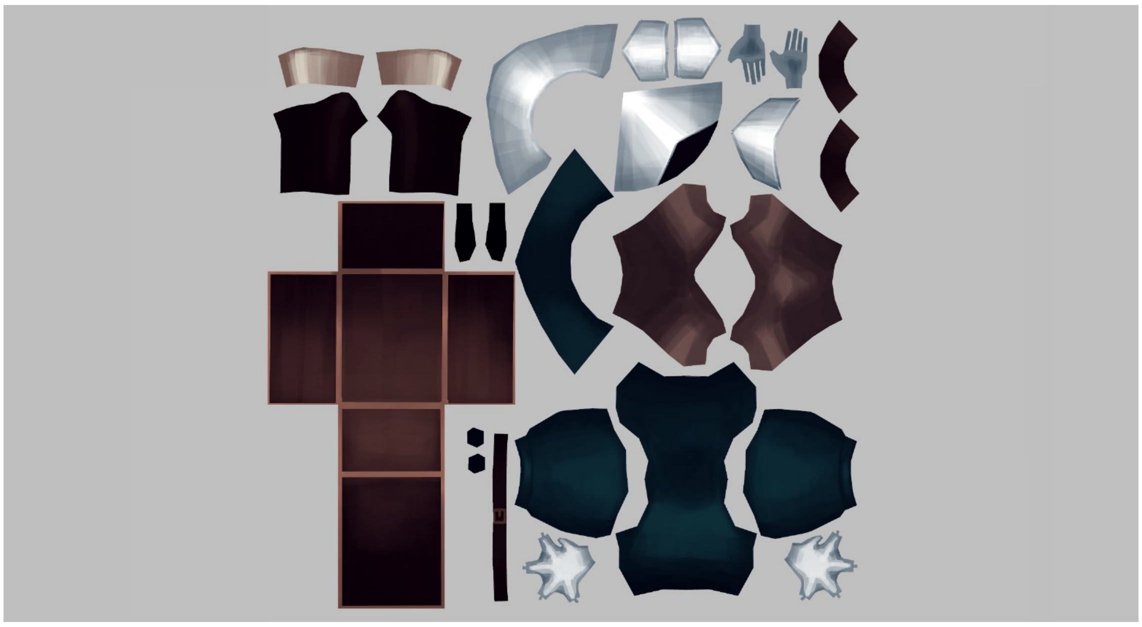
Before 3D painting programs like Substance Painter were developed, texture painting was done by hand in digital painting programs like Photoshop. This was the case even when the textures were not stylized, as realistic textures could be achieved through photobashing. To study this more traditional workflow, the first texture for the 3D character model was created by using a digital painting program called Clip Studio Paint EX. Clip Studio Paint is a versatile painting program comparable to Photoshop in how it functions.

To prepare the model for texturing in Clip Studio, it first needed to have a material set to it. After this material was set in Blender, it was baked onto a 2D image by using the “Bake” -option in Blender’s Render Properties. This produced a 4096-pixels-by-4096-pixels basic color map, which could then be exported to Clip Studio and would be used as the template for the texture painting.

Once in Clip Studio, the first step was to establish the base colors on the map. The colors were picked by using the model sheet that was prepared earlier. These colors were then separated into different layers and folders in the program to keep them organized and easy to browse through. Utilizing layers when painting digitally is also a way to keep parts of the painting easily editable or removable when necessary. After the base colors for the model were in place, a black and white layer was added on top of them. These grayscale layers were then used to paint and establish the rough highlights and shadows on the pieces. This phase required exporting the grayscale color map out of Clip Studio and testing it in Blender on the 3D model (Picture 30). This was to ensure the light direction and strength of the shadows stayed consistent throughout the model.

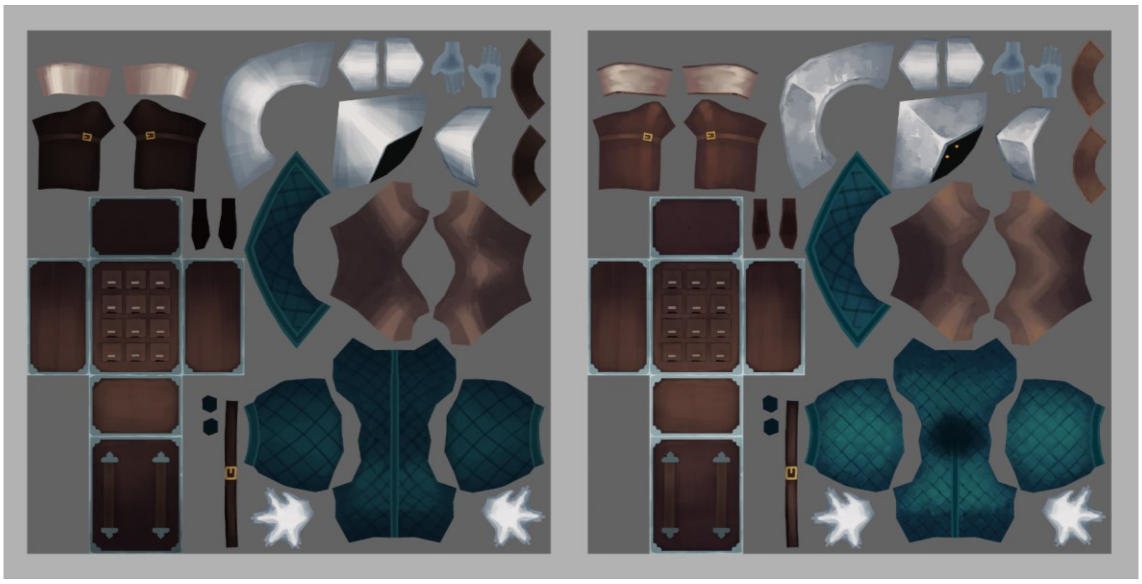

PICTURE 30. Rough black and white color block out test of the lighting on the character model.

After the rough shading was set in place and no glaring continuity issues were to be seen along the UV seams, a gradient map was added to the shading layer. Gradient map is a type of adjustment layer featured in many painting programs, and it is commonly used to give color to a black-and-white base shading layer, thus making it more optimal for further painting. The colored shading layer was then set to “Overlay” blending mode so the original base color could be seen through it and the shadows and highlights would mix into it (Picture 31).


PICTURE 31. Texture with shading layer applied on top as overlay.

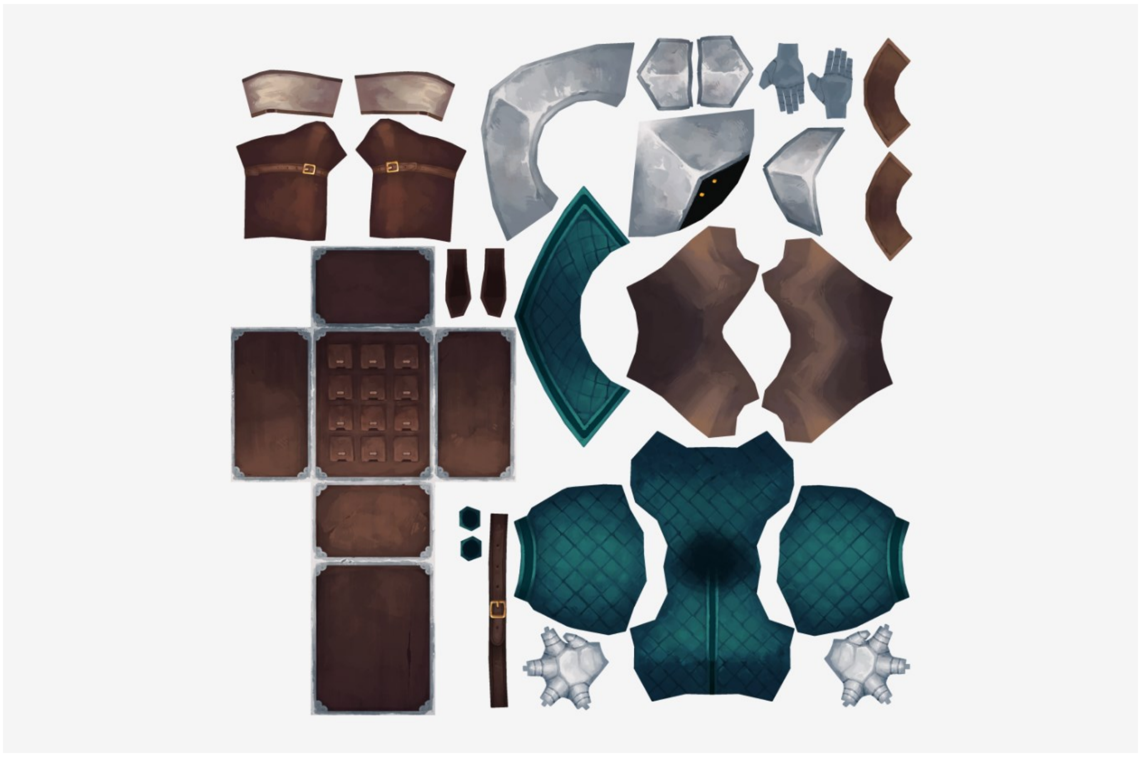
Once the lighting baseline was established, the next step was to block out patterns and other important details on the texture (Picture 32). This included such things as belt buckles, metal edges on the wooden box, fabric trims, and the quilting pattern on the character’s jacket. As it was not possible to see how the patterns connect on the 3D model within Clip Studio, this phase required frequent exporting of the color map and testing it on the 3D model in Blender. Especially the quilting pattern required multiple tests and iterations before the result was visually pleasing. For this to happen, the quilting pattern needed to be symmetrical with the light hitting the squares from a uniform direction. The UV seams also needed to be hidden, as the pattern did not line up perfectly along the edges. The quilt pattern on the skirt of the jacket needed to be bent manually with Clip

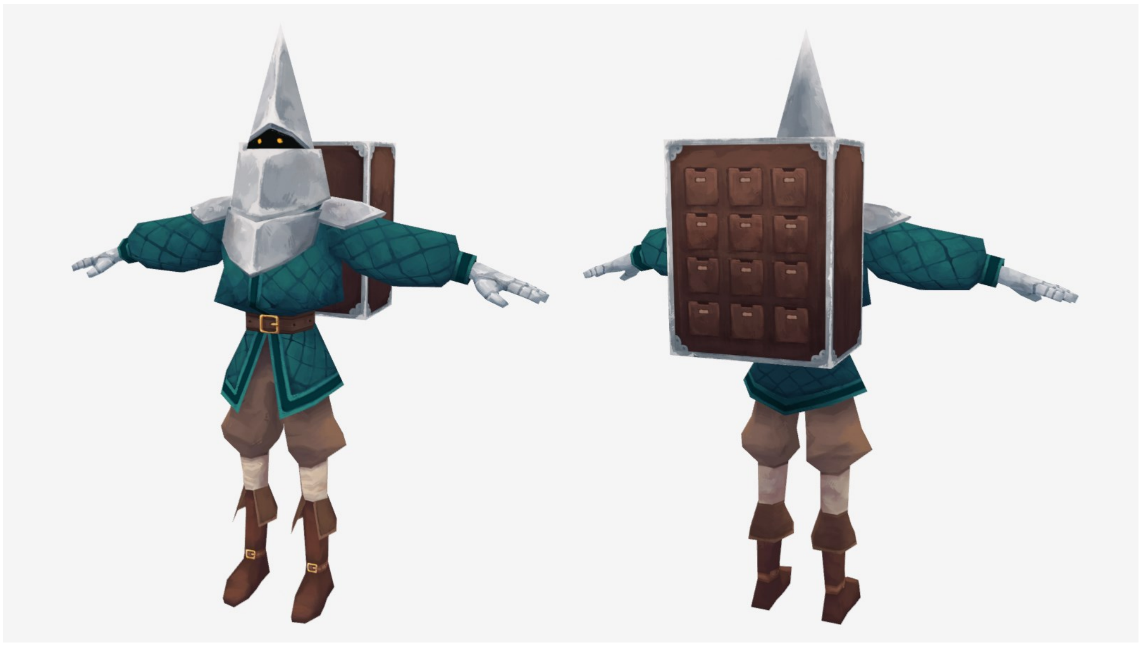
Studio's liquify tool to follow the skirt's shape. The seams and unnoticed deformation of the UVs on the model made the pattern tile and stretch in unexpected ways, notably on the front of the model.


PICTURE 32. Two versions of the texture. Early color and detail block out on the left, further details and the start of final texture painting on the right.

Starting the more creative and free part of the texture painting process was possible only after establishing the lighting and texture placement, as well as performing multiple tests of the work-in-progress textures on the 3D model. After the base was achieved and properly secured the painting process became very intuitive and akin to regular digital painting. The stylization aimed for visible brushstrokes and imperfections overall on the texture, but especially in the metallic portions to give the surface a used and roughed up look. Details and material texturing were painted on top of the base on separate layers to ensure editability. Clip Studio's transformation and liquify tools were also indispensable in the process, as they allowed for quick and effortless editing if a portion of the painted texture was deformed or otherwise laid slightly off from where it needed to be.

Once the painting process was finished and the texture featured the desired amount of painterly feel and details to it, the size of the UV islands was slightly increased by widening the base layers. This overflow on the texture helped to ensure that once it was applied to the 3D model, there would be no visible empty pixels along the seams where the texture would not have originally reached. After this, the texture was exported to a transparent PNG file and underwent final color and brightness adjustments in Clip Studio (Picture 33). After these adjustments, it was ready to be added to the 3D model (Picture 34).


PICTURE 33. Final version of the texture made in Clip Studio Paint EX.

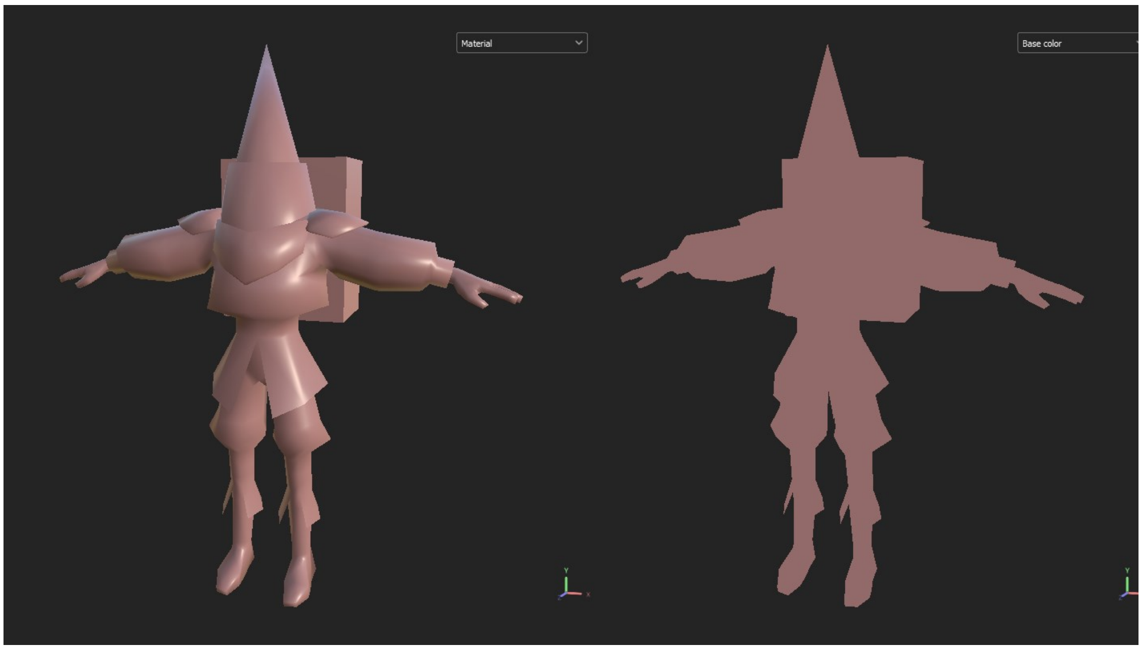

PICTURE 34. Final Clip Studio texture version on the 3D model.

4.4.2 Substance Painter

The second version of the hand-painted texture was produced in a 3D painting program called Substance Painter. Substance Painter is specifically designed for texturing 3D models, and it is capable of displaying the results of the painting process in real time on the model. It is also possible to paint directly on the model itself. Whilst it is a very useful tool for texture painting, Substance Painter is also a very heavy program to run. This project was produced on older hardware which occasionally struggled to keep up with the resource-heavy software.

As Substance Painter utilizes the 3D model itself, the model needed to be exported from Blender to an FBX file. Once exported, it could be set up as the file base for the Substance Painter file. In this step, it is important to ensure that the 3D model has only one material applied to the areas that need to be exported. In the case of this 3D model, it had multiple materials applied and thus the import to Substance Painter was faulty at first, resulting in only parts of the model appearing in the program.

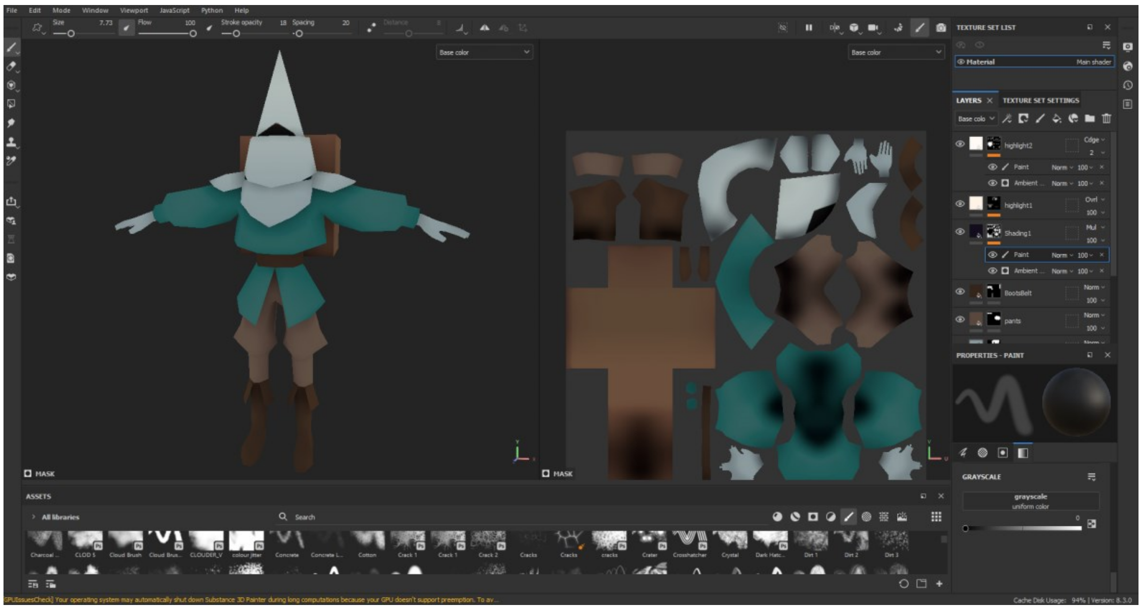
Substance Painter is capable of showcasing lighting conditions on the 3D model (Picture 35), which can be especially useful for example when texturing realistic assets. This feature is on by default, but it was turned off for this project, as the shading and shiny highlights resulting from it were distracting during the painting process.



PICTURE 35. 3D model in material view (left) and base color view (right).

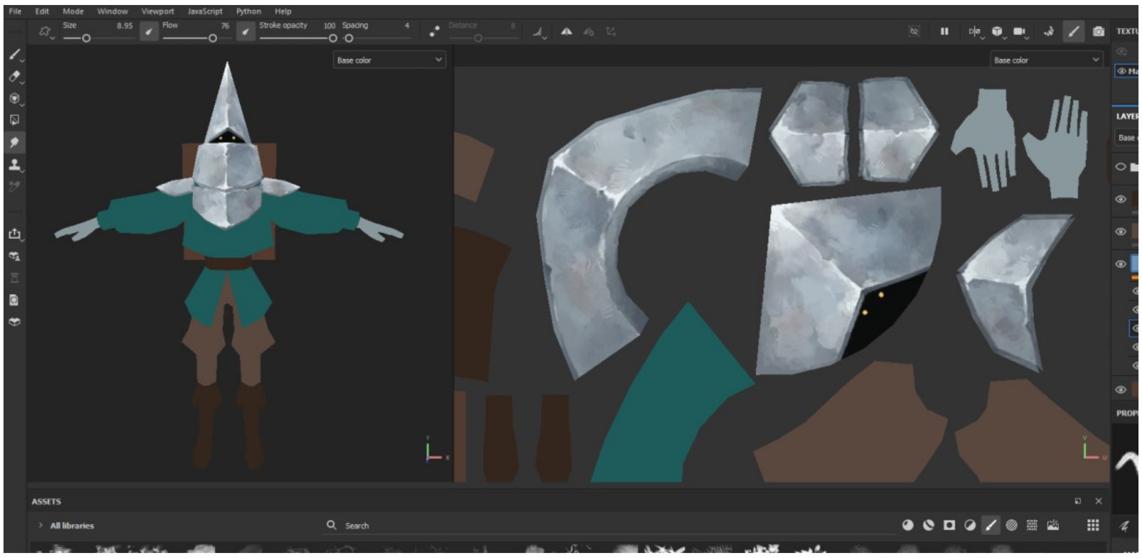
Like in Clip Studio, the chosen size for the texture was 4096 by 4096 pixels. The different portions of the texture were also divided into various colors and layers. This was done by adding fill layers with the model's base colors. Their area of effect was then limited by attaching masks to the layers. Thus, the base color layer and any painting added on top of it would only appear within the masked area. The base color could also be changed at any time if necessary thanks to the fill layers' properties.

Substance Painter has multiple tools to make the texture painting process easier. One such tool which is especially useful for hand-painted textures is the ability to bake mesh maps inside the program itself. For the texture in question, such maps as ambient occlusion, world space normal, and position were baked with the low poly model being used as the base mesh. These maps were then used to create generator layers on the model, which were a quick, effective, and consistent way to establish primitive lights and shadows on the texture (Picture 36). Thus, a 3D shape was already established for the model early without any hand painting being needed. This helped shave time off from the painting process, as it was not necessary to figure out the lighting base and direction out of nothing.



PICTURE 36. The 3D model and texture with only generator layers applied over the base colors.

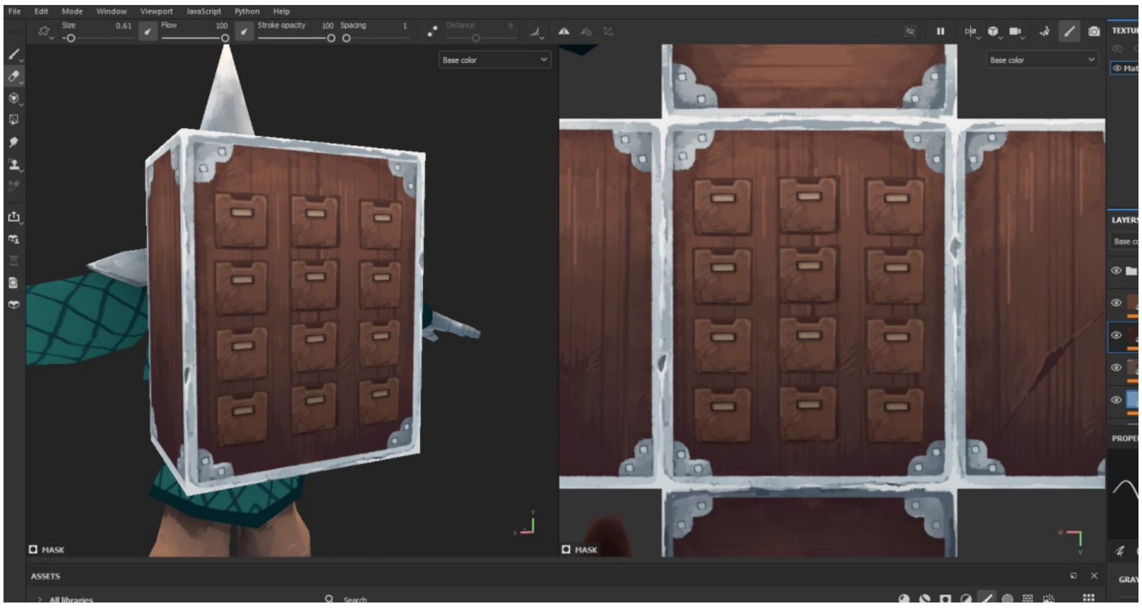
The texture painting was started after the base setup was finished. Additional ABR brush sets were imported to Substance Painter to aid in the painting process. These brush sets were the same as those used in Clip Studio when creating the first texture version. In order to paint on the fill layers, paint effects needed to be added to them. These effects function similarly to layers in any digital painting program. As such, the hand-painting process itself was very similar to how it was in Clip Studio. For an artist with a background in 2D digital painting, this made the program very comfortable and intuitive to use. Painting the texture on Substance Painter also required no testing, as it could constantly be seen on the 3D model itself. This made it possible to include more stylized brush strokes as well as to paint the texture portions in more detail from the start (Picture 37).



PICTURE 37. Portions of the texture being finalized whilst some only have base colors in place.

Being able to see the painted textures directly on the model while working also gave room for more experimentation with the painting style. As there was no need to second-guess how different parts of the model would fit together or if a certain surface texture would jump out too much, the colors and stylization were easier to adjust and intensify compared to when painting in Clip Studio.

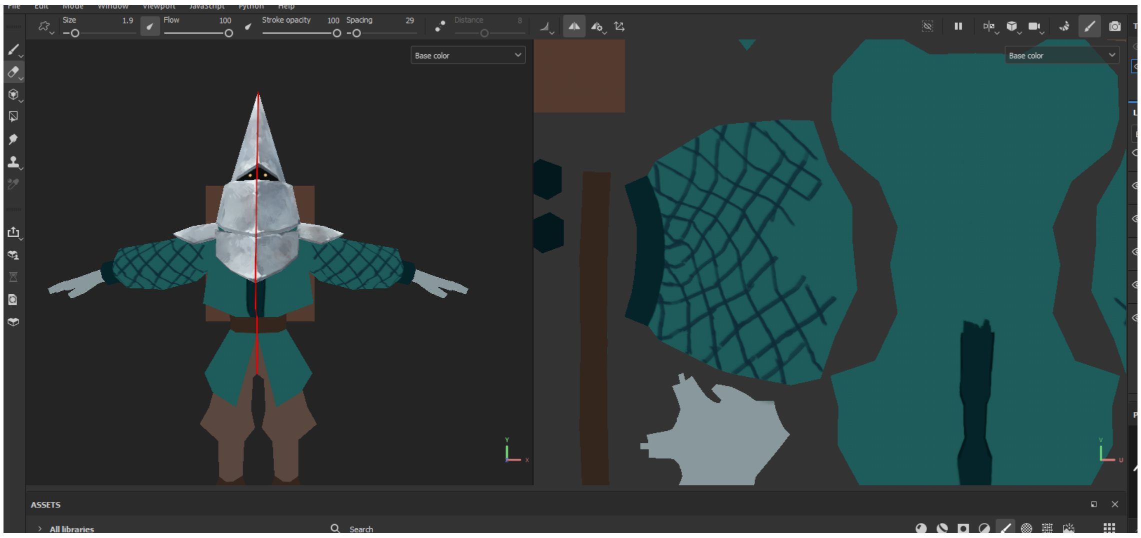
Whilst painting directly onto the 3D model was useful in being able to play around with the brush strokes and see the results simultaneously, Substance Painter limited some aspects of the process with its occasional rigidity. Unlike traditional 2D painting programs, Substance Painter does not feature selection or movement tools that could be used to manipulate the texture easily and quickly. As such, if a portion of the texture was painted in the wrong spot, the only way to move it around was by using a layer transform filter. The transform filter would move the layer contents on the X- and Y-axes. This would, however, move everything on the selected layer, and thus was less versatile and flexible than the liquify tool in Clip Studio. When painting the texture for the project, the transform filter was utilized when centering the small drawers on the larger wooden box (Picture 38). To save time in the painting process, the drawers were also multiplied from a single source drawing by using Substance Painter's cloning tool.



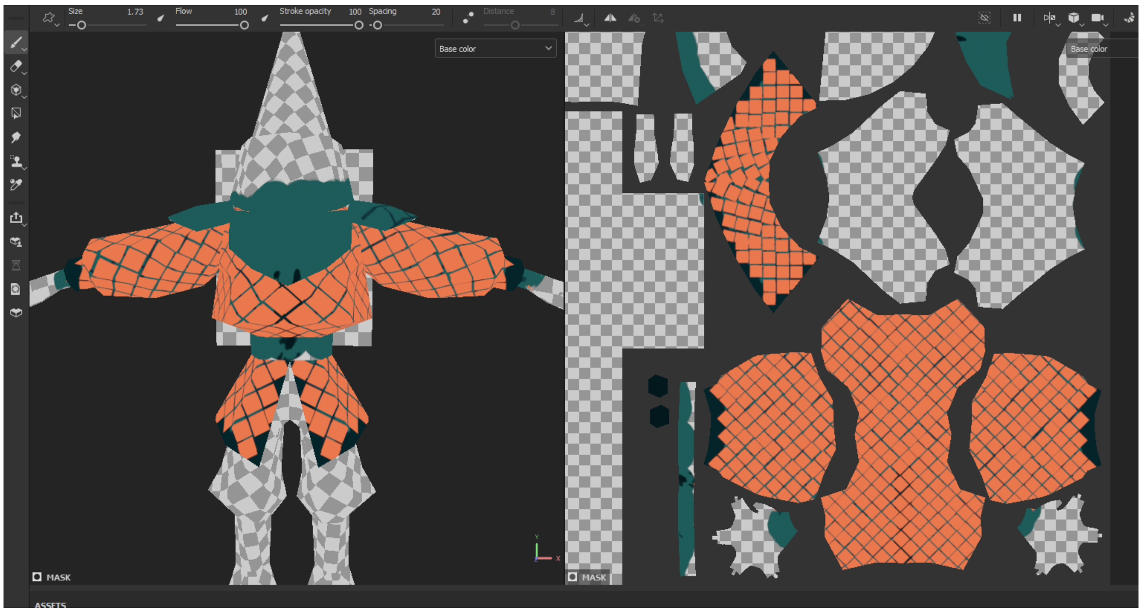
PICTURE 38. Small wooden drawers multiplied and centered onto the larger box.

The majority of the texture painting process in Substance Painter was very straightforward and akin to more traditional digital painting. The large, simple surface areas of the low poly model made it easy to paint different materials and surface textures on them. However, more complicated patterns, especially highly geometrical ones like the quilting pattern on the character's jacket, proved to be difficult to paint directly on the model.

The curving surface of the jacket made the pattern stretch and bend in unwanted ways when they were blocked out on the 3D model (Picture 39). As such, instead of painting it on the 3D surface, the quilting pattern was masked by hand on the 2D texture by using a square brush. This yielded an even geometric pattern for the jacket (Picture 40). A drawback of this was that the pattern did not line up seamlessly along the UVs' edges. This however could be masked with shading. Once the pattern was established, a "Blur Slope" -filter was added to the masking layer to give the pattern some organic randomness and to slightly blur its harsh edges. After this, the colors and final details such as the jacket's fabric trims were painted on the model, at which point it was ready for final checks and adjustments.



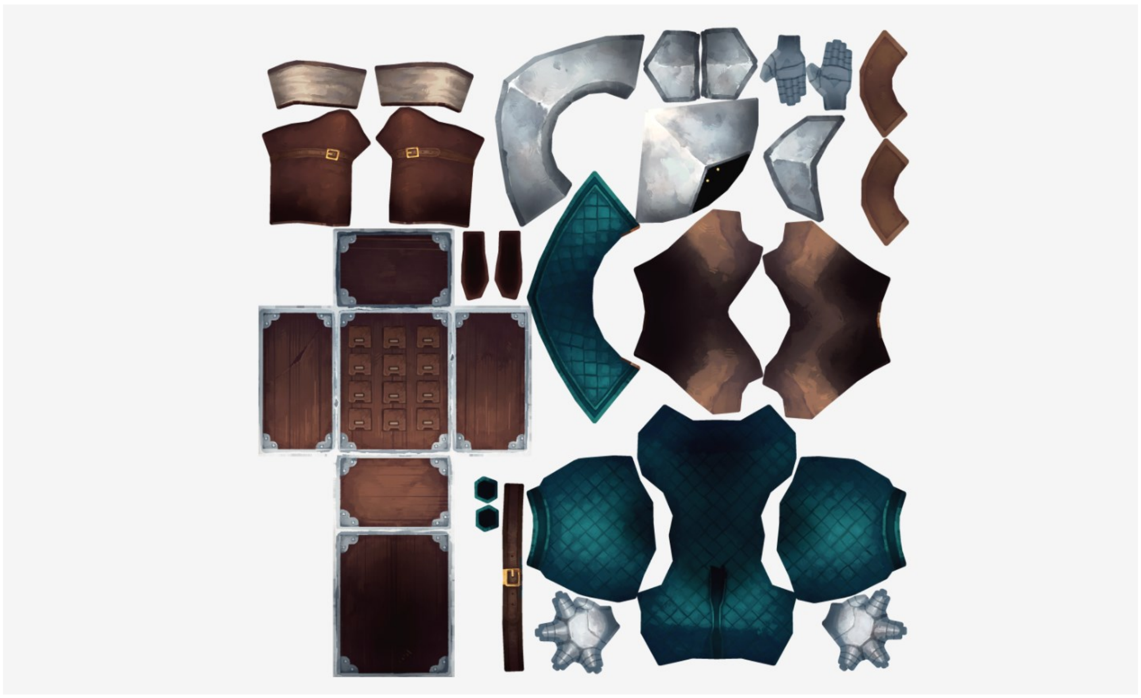
PICTURE 39. Block out sketch of the hand-drawn quilting pattern showing unwanted deformation on the 2D texture.



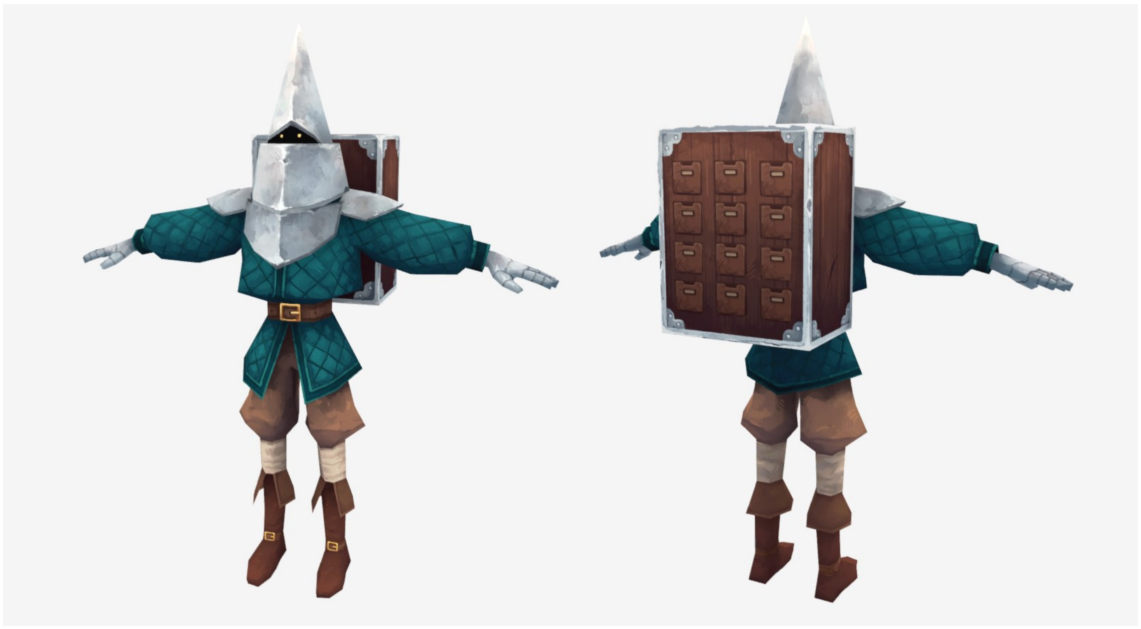
PICTURE 40. Quilting pattern masked with a square brush.

Once the painting process was finished and the texture did not require any more editing from that standpoint, it was exported to a color map. Substance Painter features multiple different exporting templates, which can be useful when exporting textures to certain 3D programs or game engines. The texture in this project, however, did not need to use any premade template. The exported texture contained only the basic color map that was needed. The 4K texture was exported with a transparent background and some added padding to prevent possible empty spaces between the UV seams.

In post-processing, the finished texture underwent slight adjustments to its contrast and color values. This was done within Substance Painter by adding the exported texture on the 3D model as an image layer. Various filter layers could then be added on top of the image layer, similar to how adjustment layers would be used in digital painting programs like Clip Studio or Photoshop. After post-processing, the adjusted texture was re-exported (Picture 41) and added to the 3D character model in Blender (Picture 42).



PICTURE 41. Final version of the texture made in Substance Painter.



PICTURE 42. Final Substance Painter texture version on the 3D model.

5 DISCUSSION

Low poly models can be less tedious and faster to model compared to 3D sculpts, and this is one of their biggest advantages. Considering the time and budget constraints the game development environment often brings, it is beneficial to be able to produce 3D models at a swift pace. As low poly models are also typically modeled with the polygonal approach, they do not require the extra workload of going through the retopology workflow either. These factors make low poly models ideal to be combined with the more labor-intensive hand-painted textures.

Besides time management, low poly models and hand-painted textures benefit from one another in other ways as well. Low poly 3D models can be very blocky and simplistic in their shape language, which is something that highly detailed textures can help mask. With carefully painted textures, the viewer can be fooled to think that a flat surface could contain more geometry and details than the model actually has. On the other hand, low poly models' simplistic geometry can also work as a better painting canvas, as covering larger areas is easier than having to paint multiple separate small details. Whilst UV stretching can be a possible drawback in low poly models, this can often be masked and made undetectable during the painting process.

The findings from the thesis indicate that painting 3D textures by hand is a time-consuming process, but one that can be easily approached especially when having previous experience in digital painting. Texture painting workflows and traditional digital painting workflows are very similar to one another. Texture painting can also be done by utilizing the same or similar programs as digital painting does. As such, hand-painting textures can be a good choice for an independent creator with a limited budget but possible access to either free or already previously owned painting programs.

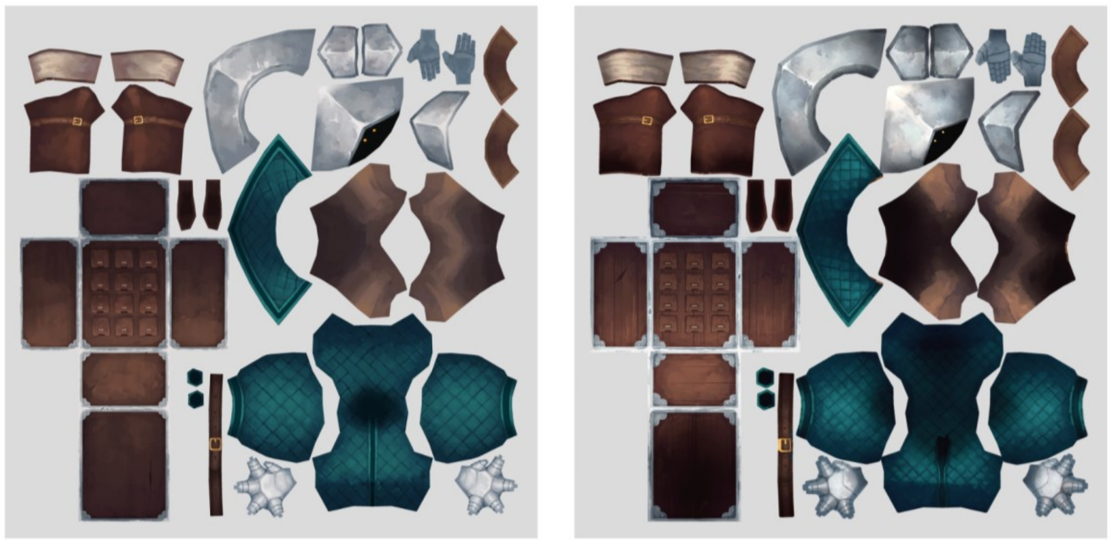
The study compared two different programs for hand-painting textures: the digital painting program Clip Studio Paint EX, developed by Celsys, and the 3D texturing program Substance Painter, which was originally developed by Allegorithmic and acquired by Adobe in 2019. These programs were chosen because they represent contrasting workflows for texture painting. Clip Studio functions in a 2D space similarly to Photoshop, and thus approaches how textures used to be produced prior to the conception of 3D painting programs. Substance Painter represent the more modern approach of painting directly onto a 3D surface.

Clip Studio Paint EX is the more affordable option of these programs, as its monthly subscription is 5,58€ a month compared to Substance Painter's 18,52€. Clip Studio Paint EX can also be replaced with the cheaper version of the program, Clip Studio Paint Pro, as both versions share equivalent options and tools when it comes to digital painting. Clip Studio Paint Pro's subscription price is 1,94€ per month.

When working on the texture the main difference between the programs was the ability, or lack thereof, to see the texture painting result in real time. The texture produced in Clip Studio required multiple tests and iterations to ensure it fit well on the 3D model. As such, the overall production time for this texture was higher than its Substance Painter counterpart, with the Clip Studio texture taking approximately 20 hours to create and the Substance Painter texture taking an estimated 15 hours. It is important to consider, however, that the textures produced for this study were treated as important main character textures. Thus, they contained a high level of detailing and were overall more time-consuming to produce.

As Substance Painter is designed to be a 3D painting program, it also benefitted greatly from its built-in functions such as generators, seamless painting over UV borders, and different exporting options. It is, however, a very resource-heavy program and requires up-to-date hardware to function without any hindrances. Clip Studio on the other hand makes up for what it lacked in real-time viewing in flexibility and adjustment possibilities. Both programs were able to handle the painting process itself very well with accurate brush control and intuitive color selection and mixing.

Appendix 1 of the thesis contains a link to a showcase video of the finalized textures on the low poly 3D model. The results of the produced textures were comparable with one another in terms of production quality. The differences between them were trivial and mainly had to do with minor things like color balance or contrast (Picture 43). The Substance Paint texture features slightly stronger color contrasts and shading as the live view of the 3D model made them easier to adjust. Likewise, the painting stylization on the Substance Painter version has also been pulled further for the same reason.


PICTURE 43. Clip Studio texture on the left, Substance Painter texture on the right.

The study found that, whilst different in their approaches and workflows, both Clip Studio Paint EX and Substance Painter could be used very fluently to produce professional and game-ready hand-painted textures. As such, both have the potential to be used in video game development and could be useful tools for game artists. For an indie developer or a hobbyist, however, Substance Painter's monthly subscription price of over 18€ could be less appealing than Clip Studio's 5,6€ or even 2€. The findings also suggest that, despite the price point, the ideal way to paint textures by hand could be to combine these two programs. What one program may have lacked, the other typically made up for.

REFERENCES

Adobe. N.d. Everything you need to know about physically based rendering. Read on 16.5.2023. https://www.adobe.com/products/substance3d/discover/pbr.html

Ahearn, L. 2016. 3D Game Textures, 4th Edition. Massachusetts: A K Peters/CRC Press.

Arcane: Bridging the Rift. 2022a. I Only Dream in Risky, episode 1. Direction: Chad Terpstra. Production: Mirror Darkly, Toboggan. 4.8.2022. YouTube. https://youtu.be/Mz4-38d3-AE

Arcane: Bridging the Rift. 2022b. Killstreaks Meet Keyframes, episode 3. Direction: Chad Terpstra. Production: Mirror Darkly, Toboggan. 18.8.2022. YouTube. https://youtu.be/Qd1rOtu4SMI

Autodesk. N.d. Autodesk Maya: Luo laajoja maailmoja, monimutkaisia hahmoja ja upeita tehosteita. [Autodesk Maya’s purchase page]. Read on 10.3.2023. https://www.autodesk.fi/products/maya/overview?term=1-YEAR&tab=subscription&plc=MAYA

CG Masters. 2020. 3D Basics – What are Normals? Watched on 12.6.2023. https://youtu.be/hkTjreiookM

Clinton, Y. 2008. Game character modeling and animation with 3ds Max. Amsterdam; Boston: Elsevier/Focal Press.

de Vries, J. N.d. Normal Mapping. Read on 7.4.2023. https://learnopengl.com/Advanced-Lighting/Normal-Mapping

FlippedNormals. 2020. Poly Modeling vs Sculpting – Which is Better? Watched on 15.5.2023. https://youtu.be/EvzQYzczUH8

GarageFarm Academy. 2021. PBR Explained in 3 Minutes – Physically Based Rendering. Watched on 20.5.2023. https://youtu.be/_ZbkOZNgwNk

Gupta, A. 2020. Blender 3D: A brief introduction to the most popular open-source 3D software. Read on 21.3.2023. https://uxdesign.cc/blender-3d-model-animate-create-70c774806691

Heginbotham, C. N.d. What is 3D Digital Sculpting? Read on 1.4.2023. https://conceptartempire.com/what-is-3d-sculpting/

Hibit, E. 2022. Color Theory for Dummies. New Jersey: John Wiley & Sons, Incorporated.

Jones, C. 2019. How to Make Digital Paintings Look Traditional. Watched on 12.6.2023. https://youtu.be/0Dbf63YnkeA

Maxon. N.d. Plans and Pricing. Read on 10.3.2023.
https://www.maxon.net/en/buy

McBride, K., Clarke, C. & Gonzales, G. 2018. Artists at Work: Creating World of Warcraft Art. Blizzcon 2.-3.11.2018. https://youtu.be/UYElhgeoOnU

Shuman, S., Playstation Blog. 2019. From Kratos to the fearsome Rathalos, see the polygonal evolution of 5 iconic PlayStation characters. Read on 10.4.2023. https://blog.playstation.com/archive/2019/12/16/from-kratos-to-the-fearsome-rathalos-see-the-polygonal-evolution-of-5-iconic-playstation-characters/

The Scout. 2019. A Brief History of 3D Texturing in Video Games. Read on 21.4.2023. https://discover.therookies.co/2019/05/09/a-brief-history-of-3d-texturing-in-video-games/

Totten, C. 2012. Game Character Creation with Blender and Unity. Indianapolis: Sybex.

Turbosquid. N.d. Tris, Quads & N-Gons. Read on 5.3.2023. https://resources.turbosquid.com/training/modeling/tris-quads-n-gons/

Wingfox. 2022. What is Hard Surface Modeling? Read on 11.4.2023.
https://blog.wingfox.com/what-is-hard-surface-modeling/

Zeman, N. 2014. Essential Skills for 3D Modeling, Rendering, and Animation. Natick: CRC Press LLC.

APPENDICES

Appendix 1. Link to 3D character turnaround with textures

https://youtu.be/PLdCQbzeqOs