Achieving Comics Inspired 2D-look for a 3D Game Character with Cel-Shading Aesthetics

Roni Kähäri

BACHELOR'S THESIS
December 2022

Degree Programme in Media and Arts
Interactive Media

ABSTRACT

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

KÄHÄRI, RONI:
Achieving Comics Inspired 2D-look for a 3D Game Character with Cel-Shading Aesthetics

Bachelor's thesis 50 pages
December 2022

The goal of this thesis was to discover and examine the techniques used to make real-time 3D game characters resemble 2D art inspired by comics and cartoons. The research was conducted by studying the history of games with 2D-stylized 3D art, as well as by reviewing literature, articles, and videos regarding the subject matter. The techniques found during the research were then used to create a character model in 2D-stylized 3D art to be used in a real-time game engine, providing a further practical look into the difficulties that may arise in the creation of a 2D-stylized 3D character model.

The key result of the research was the existence of various common techniques used in the creation of 2D-stylized 3D art, such as cel-shading, manual normal editing, and different texturing styles. For some aspects of development, multiple methods and solutions were found.

The results indicate there exist various contradictory ways to achieve the goal of 2D-stylized 3D game art, and which methods should be used depends on the artist's resources and the desired art style. Further research on more techniques, as well as the effect of animations, would help provide a more comprehensive look at the possibilities of 2D-stylized 3D game art.

Key words: 3D modeling, cel-shading, game art, non-photorealistic rendering

CONTENTS

ABBREVIATIONS AND TERMS

Back-face culling	a method that determines whether a polygon of an object is visible in computer graphics
Mixamo	a 3D computer graphics technology company
Sprite	a two-dimensional image used in a game engine
Unity	a game engine by Unity Technologies
Z-buffering	a depth buffer, used to detemine the depth information of a scene in computer graphics

1 INTRODUCTION

From the advent of 3D graphics in video games, the field has pushed for the advancement of highly realistic characters and environments. 3D modeling softwares like Blender and game engines like Unity and Unreal Engine default to realistic lighting and shading setups, while games such as God of War Ragnarök (2022) and The Last of Us Part II (2020) continue to break the limits of photorealism. However, instead of striving for realism, some games strive for the opposite. While physically based rendering (PBR) imitates photorealism, non-photorealistic rendering (NPR) is a field of rendering that specializes in stylized rendering techniques. Although NPR can be used for many different purposes and stylizations, one specific subset is 2D-stylized 3D graphics: the imitation of 2D visuals, such as cartoons, animation, and comic books, with 3D models and environments.

My own fascination with this specific art style dates back to the release of The Simpsons Game (2007). Although the animated series had inspired multiple video games before, to me this was the first game to truly capture the visuals of the original animations and make me feel like I was playing the cartoon itself. To me, this is the essence of the art style: the fantasy of being able to go inside an animation or a comic book, while still retaining the benefits of 3D gameplay.

What 2D-stylized 3D graphics look like depends entirely on the inspirations and intentions of the artist. One game may imitate the visuals of a cartoon, while another may replicate the style of a painting. Despite the varying end results, many of them still share similar modeling, texturing, and rendering techniques. The main goal of this thesis is to discover and showcase these methods. All the techniques will be specifically applicable to game development, to be used for real-time rendered characters, and as the focus will primarily be on modeling and texturing, no animation-related techniques will be discussed. Due to the variety of possibilities within the art style, the thesis will not showcase a single strict pipeline, but rather discuss different options and methods that can be used to create many different stylizations. Additionally, a practical project that shows the full process of creating a 2D-stylized 3D character from the initial concepts to the final rendering in the Unity-game engine, is included at the end. This provides a further examination of the techniques and shows a practical example of both the disadvantages and benefits they may offer.

2 UNDERSTANDING 2D-STYLIZED 3D GAME ART

Before discussing the specifics of creating 2D-stylized 3D game art, it is beneficial to develop an understanding of the limits and possibilities of the art style. The purpose of this chapter is to provide a look at the history of the art style, and a summary of its advantages and challenges.

2.1 What is 2D-stylized 3D?

2D-stylized 3D game art refers to 3D game art that resembles or imitates two-dimensional drawings and graphics. Although the style is often referred to as toon-shading or cel-shading, for this thesis a wider term “2D-stylized 3D art” will be used instead, so that games that utilize PBR techniques may be included in the discussion as well. Furthermore, cel-shading generally refers only to the shading itself and does not include other aspects of development used to make the models resemble 2D art, such as outlines, normal editing or texturing.

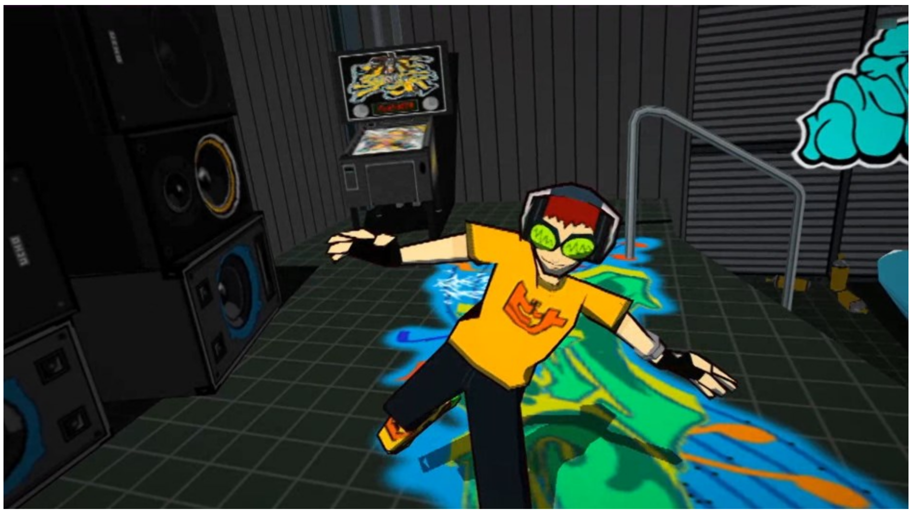
The field of 2D art itself is expansive and has a vast range of different styles and mediums. Sketching, painting, printing, digital art, and many more techniques can create very different visual outcomes. There are no limitations to what type of 2D art can be used as the source of inspiration when imitating 2D art in 3D, though most games with this style seem to pull inspiration from animation and comic books.

Features such as black outlines, dot rasters and the limited use of colors derive specifically from the early days of comic book printing. Comic books used to be printed on a four or five-color newspaper press with the spot color-method, where each color layer was printed individually (Wiley 2019). This was eventually replaced by the Ben Day process, where the colors were printed in small dots that could overlap, allowing for more shading, depth and colors (Wiley 2019). Although these processes are no longer used in modern day comic book printing, their visual impact remains and many cel-shaded games continue to imitate their effects to capture the feeling of comic book art.

2.2 Brief history of 2D-stylized 3D game art

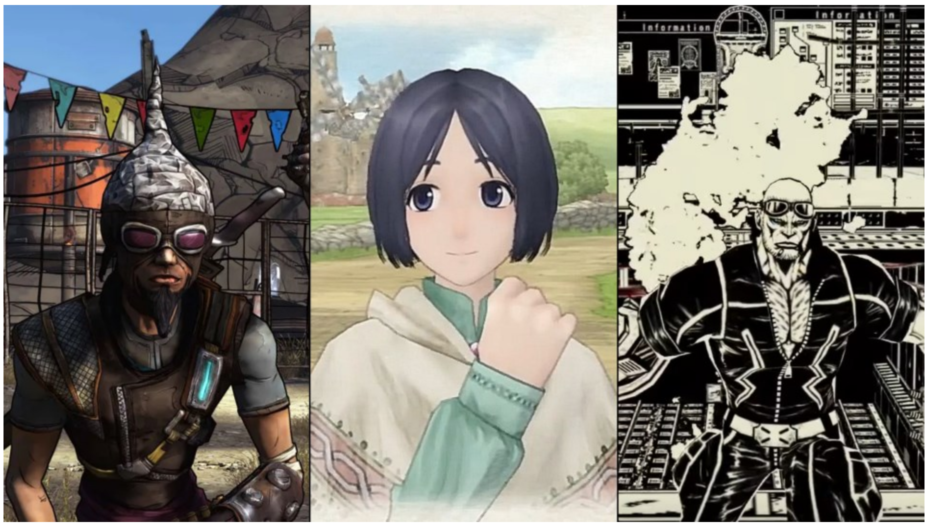
During the first decades of the history of video games, 2D games dominated the industry. While some games in the 1970s and 1980s utilized pseudo-3D techniques, such as scaling sprites up and down to create the illusion of moving in a three-dimensional environment, it was not until the 1990s that full 3D games became commonplace due to the improvements made in video card technology. (Laud 2017; PC Plus 2010).

One of the first attempts at imitating 2D art in a 3D game came in the form of static cel-shading. In this art style, used in games such as Mega Man Legends (1997) and Fear Effect (2000), the shadows and highlights were drawn to the textures of the 3D models in a way that gives the illusion of cel-shading, even though no actual dynamic lighting was in place. Similarly, PaRappa the Rapper (1996) combined 2D characters with 3D environments and camera. It was not long until true cel-shading made its first appearance with the release of Sega's Jet Set Radio (2000). The game featured clean and colorful cel-shading with thick black outlines and cartoony proportions, which made it resemble a comic book or a cartoon (Picture 1). The game was a critical success and won multiple awards, some of which were for its innovative art style (ScouseGamer88 2014).


PICTURE 1. Jet Set Radio was the first game to use cel-shading (Jet Set Radio 2000)

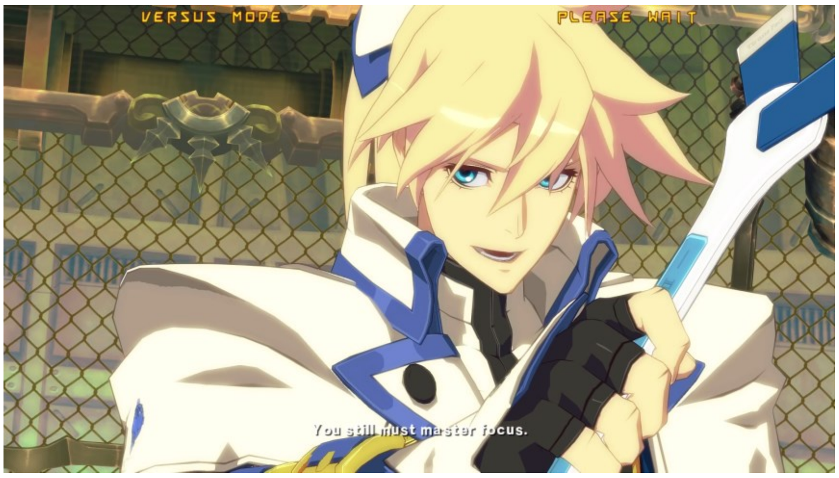
In the following years, 2D-stylized 3D graphics found a footing within the industry with the release of various popular titles: Sly Cooper (2002) used cel-shading to imitate the visuals of a children's comic book, while Ōkami (2006) replicated the look of the Japanese ink wash painting-style, sumi-e (Jones 2012). Cel-shading became especially commonplace in anime and cartoon-inspired games, as well as licensed video game spinoffs of existing 2D animated franchises, like Naruto: Clash of Ninja (2003) and The Simpsons Game (2007). Not all reception to the art style was positive. The Legend of Zelda: Wind Waker (2003) was released to a divisive reception due to its cute and cartoony cel-shaded graphics, which many saw as inferior to the more realistic graphics of the game's predecessors (ScouseGamer88 2014).

In the late 2000s and early 2010s, 2D-stylized 3D game art began to grow beyond simple and colorful cel-shading and games with a variety of different styles released over the following years (Picture 2). Borderlands (2009) combined the otherwise photorealistic lighting with black inner lines and thick, strong outlines; Valkyria Chronicles (2008) experimented with gentle pastels and soft hatching in the shadows to give the impression of sketchbooks and watercolors; and Mad World (2009) impressed with its highly stylized black, white and red, comic book -like graphics.



PICTURE 2. Different implementations of 2D-stylized 3D graphics in the late 2000s (Borderlands 2009; Valkyria Chronicles 2008; Mad World 2009)

In the mid-2010s, Guilty Gear Xrd Sign (2014) was released featuring extremely polished models and shading that appears nearly identical to 2D animation (Picture 3). The game's graphics gained praise from critics and fans alike, and Metro GameCentral's review of the game even called the graphics "easily the most successful [attempt at recreating anime style visuals] so far" (Metro GameCentral 2015). Other cel-shaded titles, such as Persona 5 (2016), The Legend of Zelda: Breath of the Wild (2017), and Genshin Impact (2020) released in the following years to wide critical acclaim and popularity. In addition to the games created by well-established studios, many indie developers have found success with the style as well, as evidenced by games such as A Short Hike (2019) and Haven (2021).



PICTURE 3. Guilty Gear Xrd Sign's 3D graphics imitate anime style extremely closely (Guilty Gear Xrd Sign 2014)

When comparing the graphics of Jet Set Radio (2000) and Guilty Gear Xrd Sign (2014), it can be seen how fast 2D-stylized 3D game art evolved in the 14 years that passed between the releases of the two games. As more game developers and artists continue to expand the style, discover new techniques, and gain inspiration from one another, the possibilities of 2D-stylized 3D graphics keep only growing further. Although the style has already developed fast in both technology and popularity, there still remains room for improvement and change.

2.3 Using 3D to replicate 2D art

Before getting into the specifics of modeling and development, it is important to address why 3D is used to imitate 2D art. This chapter explores the benefits that 2D-stylized 3D game art provides, and introduces some of the challenges that make translating 2D art into 3D graphics difficult.

2.3.1 Advantages

A video game's art style can be vital to the way the game is perceived. It can help strengthen the game's mood, enhance the story and message the game is trying to convey, and attract a specific audience (RocketBrush n.d.). A game with bright and cheerful graphics is more likely to target a younger audience, while grim and realistic graphics may indicate an adult target audience. 2D-stylized 3D games can be specifically targeted to the fans of the original art forms. If a video game is intended for an audience that loves anime, making the game look like playable anime can help catch their attention and attract them to it.

There still remains the question as to why 3D graphics should be used instead of 2D art. After all, if a game is supposed to look like a comic book or a cartoon, would it not be easier to use 2D sprites instead of utilizing tricks and unconventional techniques to imitate the same style in 3D? The reason for this is the various advantages that 3D graphics provide over 2D sprites, which can help make the game more visually impressive and the development smoother.

An advantage that 3D models have over 2D sprites is their easy adaptability. Although the initial workload of a 3D model may be more time-consuming than drawing a single sprite, the 3D model can be edited or transformed into a separate model quickly at any point in the production (Walla Walla Studio 2022.) This is especially crucial when considering animation: if an animated object's design is changed late in production, or another object needs to reuse the same animation, every single frame needs to be corrected to match that new design. For a 3D model, this can be as simple as applying the old animation data on the new model, but for a 2D sprite, the frames will have to be redrawn manually.

3D models allow for free and dynamic movement of the camera due to their ability to be viewed from any side and distance. This enables various gameplay mechanics that would be difficult to execute in a 2D environment and allows for more innovative storytelling and cutscene possibilities. For example, when the fighting game Guilty Gear Xrd Sign (2014) moved from 2D sprites to 3D models, the added dimension allowed the camera to move around during special attacks and finishers, making the battles feel more dramatic (Motomura 2015). While it is possible to use 2D sprites in a 3D environment to gain a similar effect, having to animate the sprites in multiple viewing directions increases the amount of animating work required.

Furthermore, 3D models have better compatibility with dynamic lighting. Although dynamic lights and shadows can be applied on a 2D sprite with normal maps, the creation of one adds another layer of work to the animating process. For 3D models, the lighting is calculated based on the model's topology, normal data, and -textures. As a result, 3D models are more convenient to use in games that require changing light directions or multiple light sources.

2.3.2 Challenges

For a 3D character to look convincingly 2D, everything on screen must be an intentional choice (Motomura 2015). In a 2D illustration, every line, shadow, and highlight is deliberately chosen by the artist. Lines and shapes do not have to follow the rules of physics or mathematics; everything is the result of an artistic decision. On the other hand, 3D models and their lighting follow calculations, and while they can be manipulated through various methods, this can be a difficult and time-consuming task. Even the slightest imperfection can break the illusion and remind the player that they are looking at a 3D model rather than a 2D character. As such, a great amount of care, intentionality, and unconventional 3D modeling and rendering methods are often needed to create 2D-stylized 3D models. What challenges an artist may face are also dependent on the desired art style: it might be easier to imitate simple and plain colored art styles, rather than those based on complicated 2D art, such as paintings and near-photorealistic drawings.

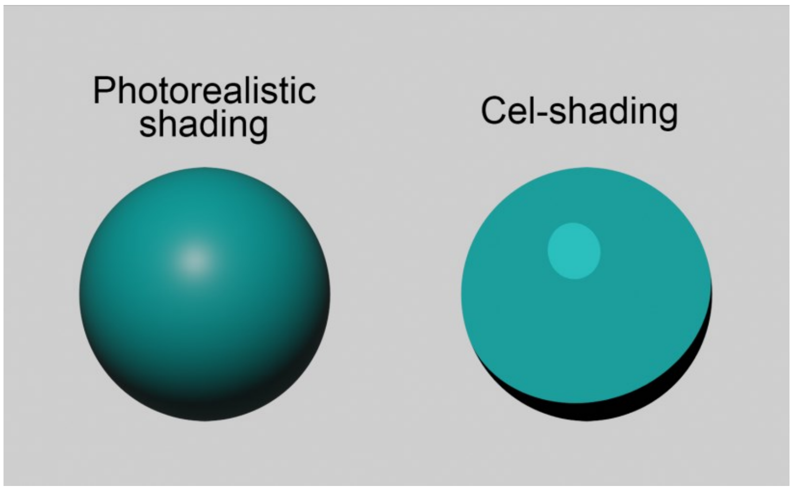
3 TECHNIQUES FOR REPLICATING 2D LOOK IN 3D

To imitate the characteristics of 2D art with 3D graphics, various specialized techniques can be utilized. This chapter will focus on showcasing and discussing some of these techniques. The purpose of this chapter is not to show a single specific pipeline, but to consider and compare many different methods. As a result, some of them may not be compatible with one another, or may even contradict each other. This will allow artists to pick and choose what works best for the art style that they wish to create.

3.1 Cel-Shading

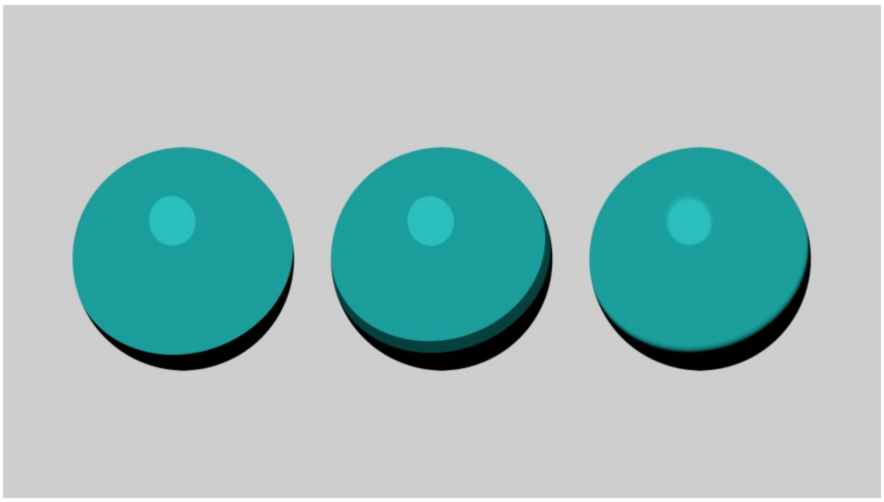
The basis for most 2D-stylized 3D models is cel-shading, also known as toon shading. The term cel-shading derives from celluloids—clear sheets of acetate that were used in traditional 2D animation. By using chunky, limited colors to represent the lights and shadows of an object, cel-shading gives the model a flat look that resembles comic books and 2D animation. (Jones 2021.)

Picture 4 showcases the differences between photorealistic shading and cel-shading. The photorealistic object's shape and lighting are represented by soft gradient shadows and spotlights, while the cel-shaded object's lighting is depicted with only three tones, giving the impression of a two-dimensional surface.

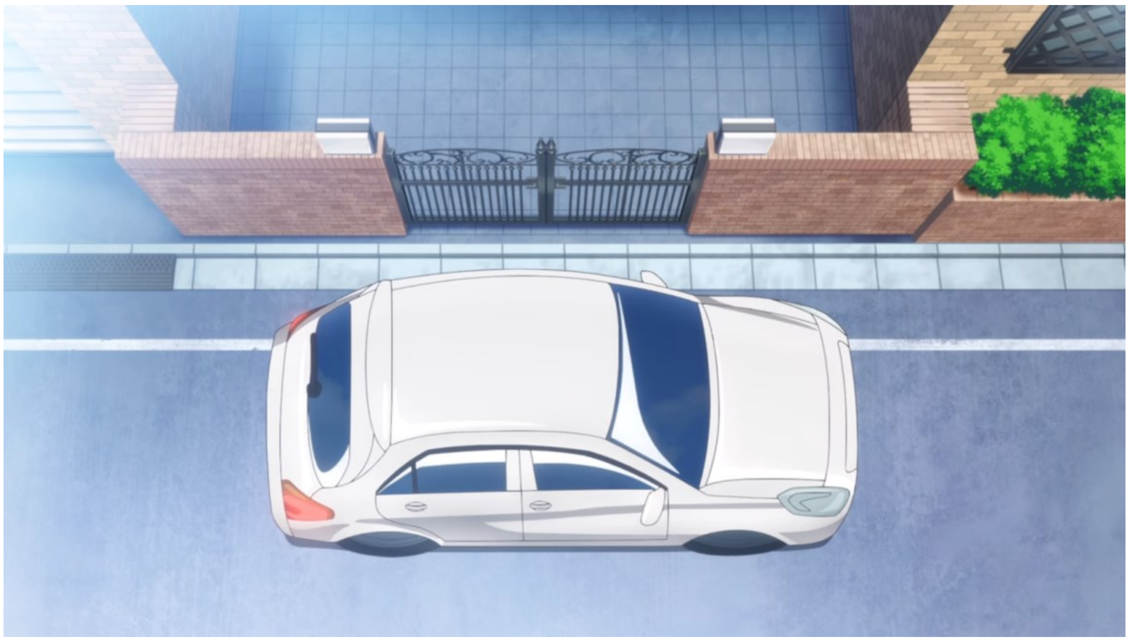


PICTURE 4. Comparison between photorealistic shading and cel-shading

The base of the shader is either a texture or a color, while the lighting is calculated from the object's normals and the angle of the light (Hamlaoui 2001). The shadows can be either sharp or have a ramp, or feathering, that softens the edge, as showcased in Picture 5. The leftmost sphere has no feathering at all, while the middle sphere has its shadows split into multiple different tones, and the rightmost sphere has a slight soft feathering to its shadows. The variation in cel-shading allows for imitating different types of 2D art styles: comic-like graphics might require sharp edges, while a game that tries to replicate the shading of a watercolor painting might use a softer edge.


PICTURE 5. Different levels of shadow edge feathering

In addition to the basic lighting, highlights and specular lighting—the reflection of a light source on the surface—can be used to further determine the dimensions and material of an object (Wu 2019). In 2D animation reflections and highlights are often portrayed in a way that follows artistic intention rather than photorealistic physics (Anjyo & Hiramitsu 2003). For example, in Picture 6 we can see that the reflection on the 2D-animated car's window does not reflect the surrounding environment, but rather appears as stylized white beams. To replicate this effect in 3D, specular lighting can be calculated with the view direction and the direction of the light source, while the glossiness of the object and the color of the highlight can be controlled either with a numerical value or a separate map (Zucconi 2015; Torchinsky 2020). Detailed areas such as faces will often require manually edited normals for the highlights to appear correctly, which will be discussed in more detail later in Chapter 3.4.


PICTURE 6. A car in a 2D-animated style with non-photorealistic window highlights (Komi Can't Communicate 2021)

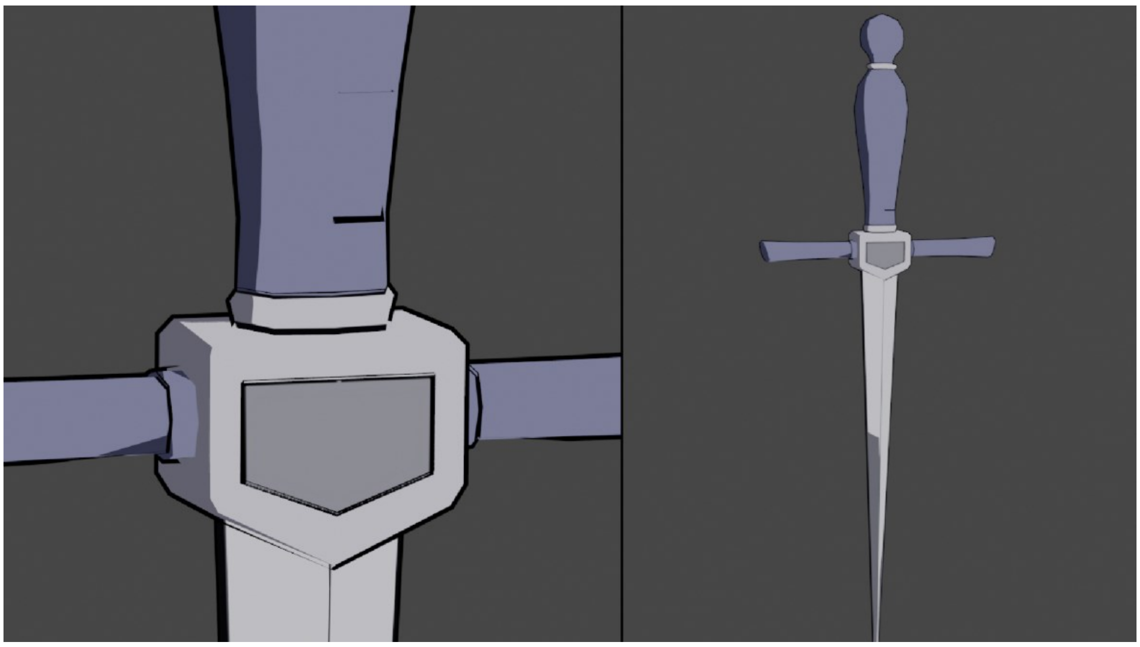
3.2 Outlines

A common companion to cel-shading is outlines—lines that follow the edges of the object, much like the lineart of a 2D illustration. Outlines can vary in size, shape, and color, as well as in the technique used to render them. In this chapter, two different rendering methods will be explored: the wireframe detection method, and the edge detection method.

3.2.1 Wireframe detection method

In the wireframe detection method, the object is rendered twice. First, it is drawn as a slightly larger version of itself in flat color, after which the back-face culling is inverted, and the back-facing triangles are rendered in the color of the outline. Once the outline is drawn, the back-face culling is returned to normal and the object is rendered as usual. The image is composited through Z-buffering, leading to a final object with black lines on the outside, as well as inner contour lines. (Jones 2021.)

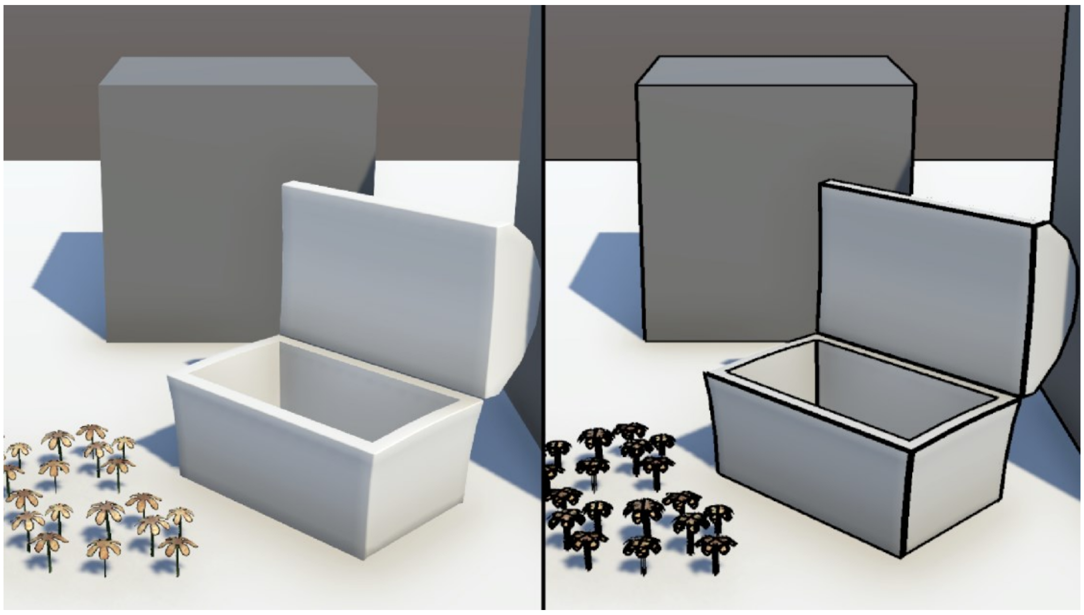
Although this method is performant and versatile to use, it requires a specific modeling setup. If the mesh is not made with the method in mind, the end result may have unintended artifacts and holes in the outline (Golus 2020). In Picture 7, an unintended line appears in the middle of the dagger's grip—this is due to the model having internal geometry that connects to the outer shell of the model and thus breaks the outline.


PICTURE 7. An unintended artifact in the outline of a 3D model

3.2.2 Edge detection method

Another way to render outlines is with the edge detection method. In this technique, the outlines are created based on a depth texture and the world space surface normals of the screen scene after the initial 3D scene is rendered. An edge detection filter is applied to create an edge surface, and finally, the two images are composited together for the final render. (Jones 2021.)

The downside to this method is that it allows for less control of the outlines on a per-object basis. While it is useful for outlines that apply to the entire screen, it can cause lines to show up in unintended places, unless the shader is finetuned to eliminate these artifacts (Ameye 2021.) On the right side of Picture 8, we can see various of these artifacts and mistakes: the treasure chest's outline is missing from several parts of the model, and the outlines are too thick for the flowers on the ground, making their color, shape, and texture difficult to see.


PICTURE 8. A scene with and without edge-detected outlines

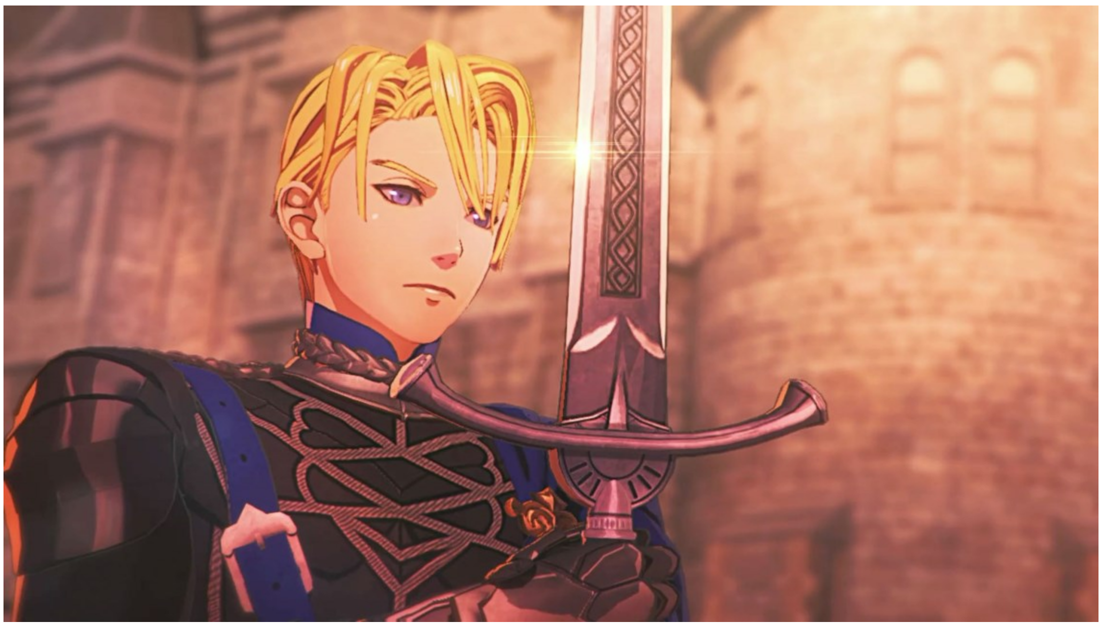
3.3 Texturing

Textures play a critical part in the overall look of any 3D model. They are used to apply color to a 3D object, whether in the form of a bitmap image, a procedurally generated image, or a combination of the two (Chopine 2011, 151). Texturing tends to fall into two main categories: hand-painted texturing and PBR texturing. As PBR textures are primarily designed to replicate photorealistic quality, they are less commonly used in 2D-stylized 3D game art. As such this chapter will primarily focus on hand-painted texturing, and another less commonly known texturing technique, limited UV texturing.

3.3.1 Hand-painted textures

For 2D-stylized 3D graphics, the most common way to texture a model is by manually painting them. This method, known as hand-painting or diffuse painting, was born from technical restrictions due to low polygon count demands and primitive shaders, lighting, and rendering techniques (Phan 2012, 118). Painted lines and shadows can help establish details and effects that would otherwise be too difficult or performance-demanding to replicate in a 3D engine, such as specific types of shading or non-photorealistic lighting.

An example of hand-painted textures can be seen in Fire Emblem Warriors: Three Hopes (2022) (Picture 9). As shown in the picture, the sword and shoulder pads have been painted to resemble metal, while the character's hair has stripes of dark and white to create the illusion of individual strands and highlights. The character's neck, inner ear, and the underside of the nose have hand-painted shadows, which ensure the areas will always appear with correct shadowing, no matter the lighting or the environment.

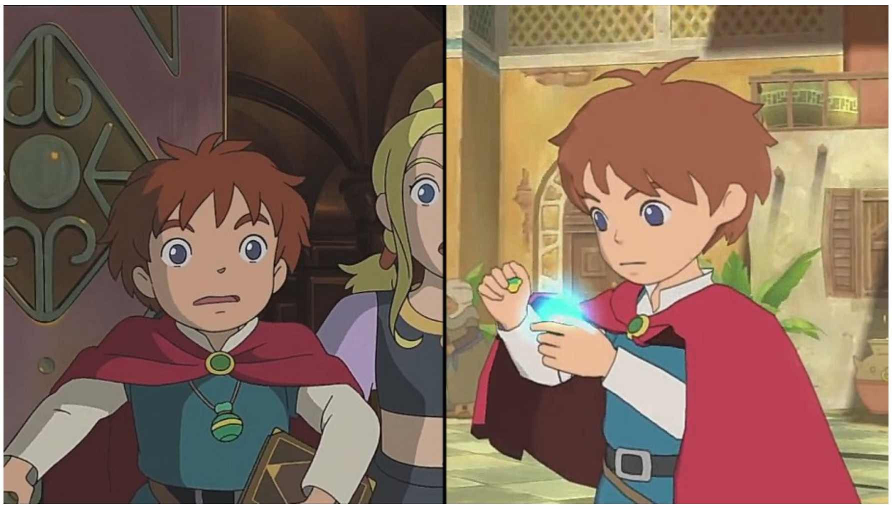

PICTURE 9. A character and sword with detailed hand-painted textures (Fire Emblem Warriors: Three Hopes 2022)

The painting of a texture can be done in drawing or image manipulation software such as Photoshop or Krita, in 3D modeling or sculpting software such as Blender or Zbrush, or in a 3D-texturing software such as Substance Painter. Although the name of the method might imply painting everything purely by hand, it is also possible to bake an ambient occlusion map of the model, which can be used as a guideline or a base for the final painting layer (Phan 2012, 123).

While painting, it is important to consider the level of detail and keep the visual quality consistent. The UV map of the model should be scaled so that even small details have enough space to match the overall quality of the whole texture. Even then, hand-painted textures can run into the issue of appearing blurry if a low-resolution texture is shown in the foreground. This can be mitigated with the use of high-resolution texture files, though that may mean sacrificing the size and performance of the game. (Chopine 2011, 163-164.)

3.3.2 Limited UV textures

In some cases, the desired art style might not require complex painting and shading, but rather a focus on colors and clarity. Shown in Picture 10 is the video game Ni No Kuni: Wrath of the White Witch (2011), which replicates its 2D animated cutscene style in 3D by using textures that have no added painting or details. The details are modeled directly into the mesh and the shadows are derived from the cel-shading, while the textures themselves remain simple plain colors.

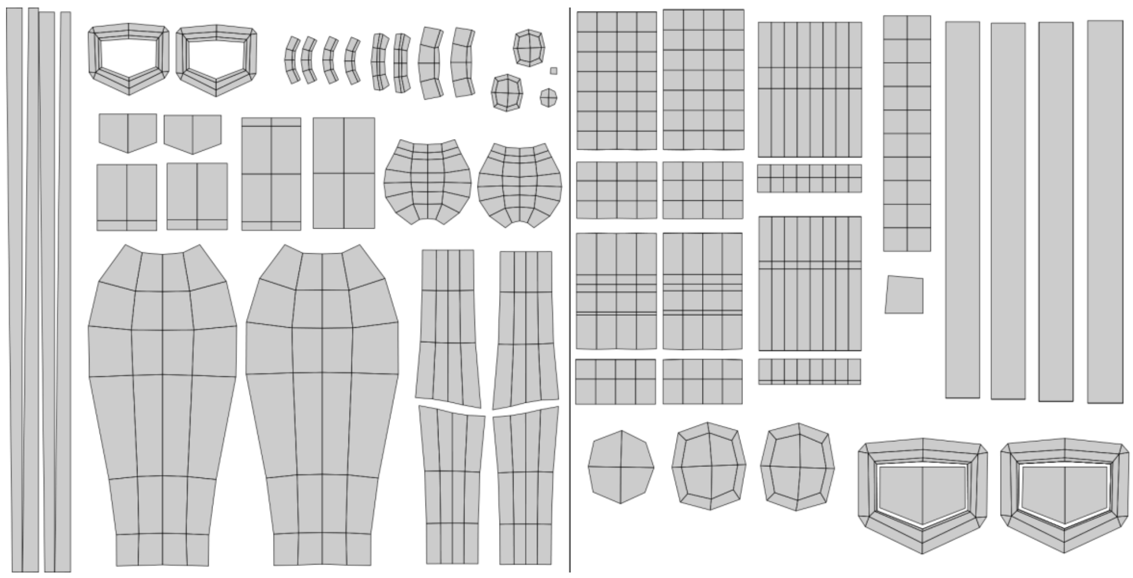


PICTURE 10. Comparison of a 2D animated scene on the left, and 3D graphics on the right (Ni No Kuni: Wrath of the White Witch 2011)

Although a similar outcome could be achieved with traditional UV maps and textures, there is an alternative method that utilizes specific topology and unconventional UV mapping to create high-quality texturing for a cartoon 3D look.

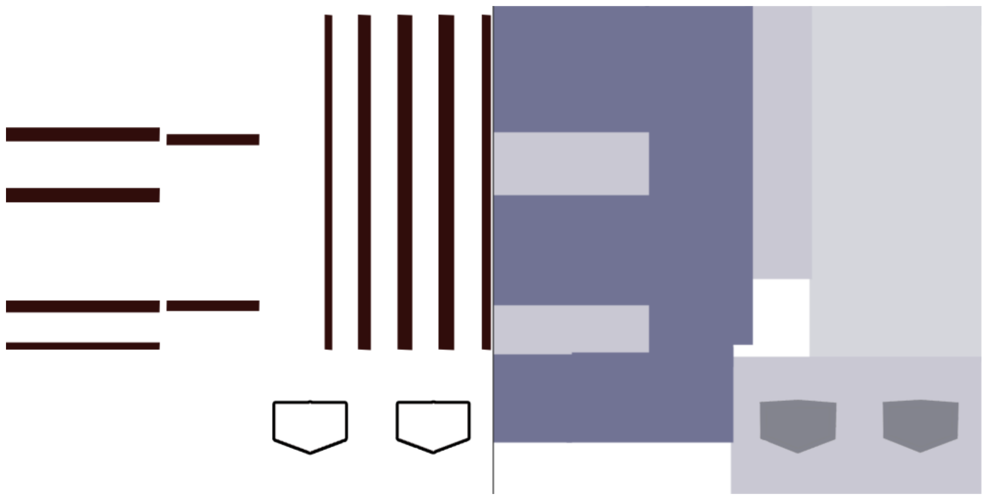
There is no commonly used name for this technique, but for this thesis, it will be referred to as Limited UV texturing.

Limited UV texturing was developed by the game studio Arc System Works for their game Guilty Gear Xrd (2014) (Motomura 2015). In this method, most details exist in the mesh itself, while the limited UV map drives the colors and the inner outlines of the model (Motomura 2015). The UV maps do not follow the usual conventions but are collections of the colors used for the model, as seen on the right side of Picture 11. In this specific case, most of the separate UV islands have been straightened to squares to make them fit better—as the texture will not include any potentially stretching details, it does not matter that the UVs do not follow the shape of the model.

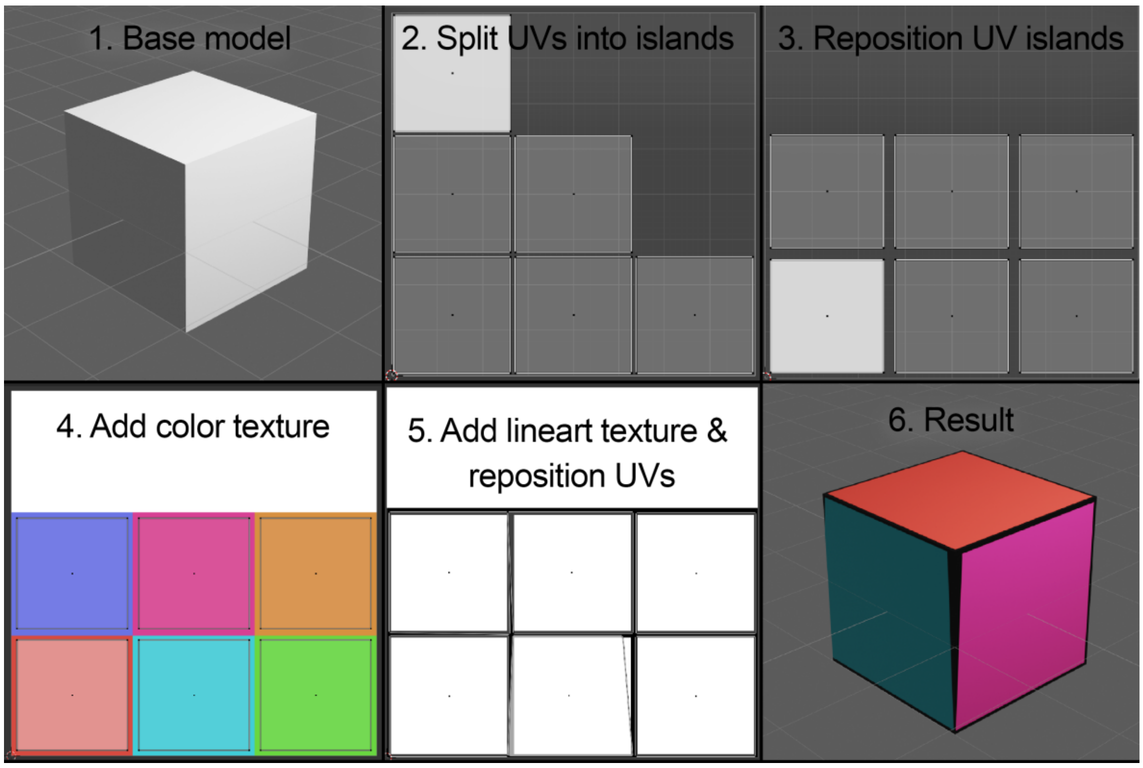


PICTURE 11. A typical UV map (left) and a limited UV map (right), both of which were created for the same base model

For the inner lines of the model, another black-and-white image is multiplied with the base color texture. The lineart texture is otherwise white but features black color near the areas where the inner lines will be shown (Picture 12). The final weight of the line is controlled through the position of the vertex UVs.

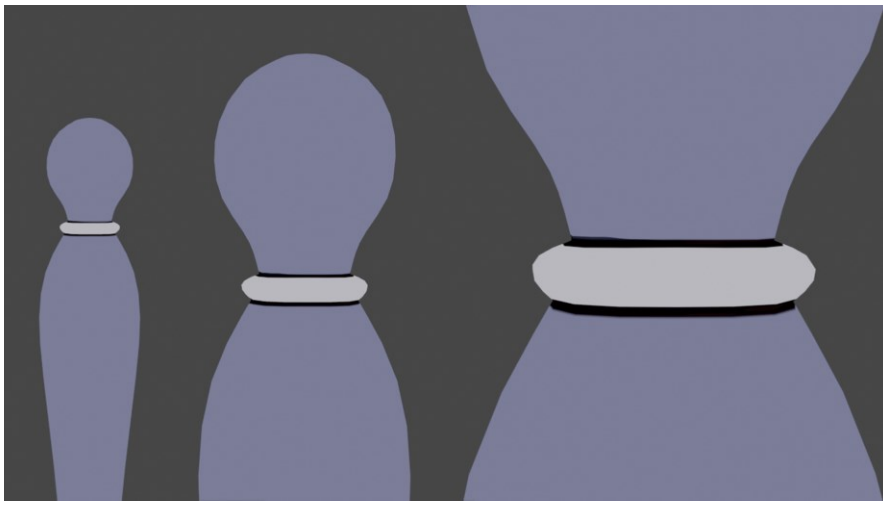

PICTURE 12. Lineart texture (left) and the color texture (right) for the same model

Picture 13 showcases the full process from start to finish. The base model's UV's are split at every edge where a color changes or an inner line appears, and the resulting UV islands are positioned so that the edges have room for the lineart texture. After a color texture has been added, a lineart texture is multiplied on top. Finally, the UVs around the inner lines are repositioned to overlap the black parts of the lineart texture, changing the width of the lines.



PICTURE 13. The full process of creating limited UV textures for a cube 3D model

The outcome of this method is a line that appears sharp even when the camera zooms close in, or when the texture resolution is low (Picture 14). For regular textures, the only way to avoid blurry and pixelated lines is by using high-resolution image files, which take a space and can slow down performance. In this method, however, the image size of the texture can be as small as 2048x2048 pixels, or even 1024x1024 pixels, without it affecting the overall quality (Motomura 2015).

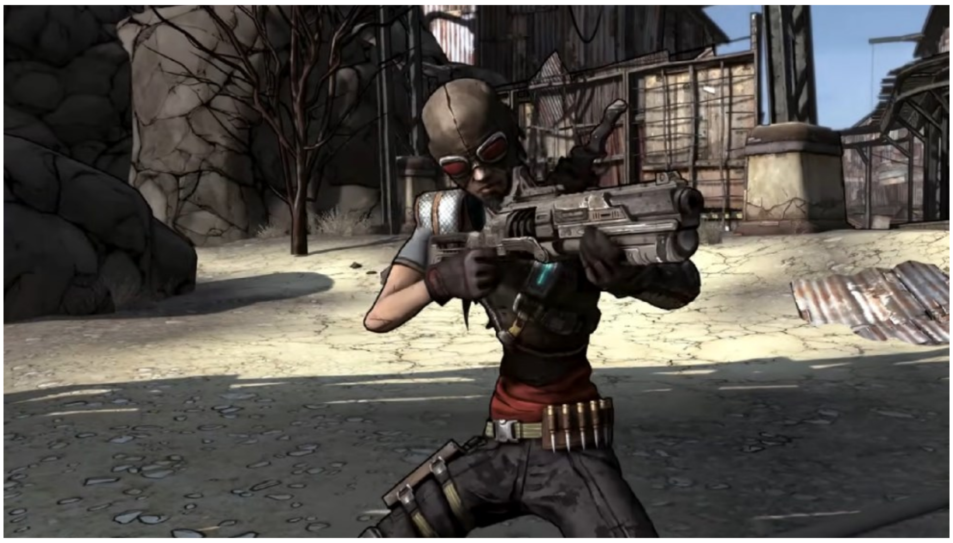

PICTURE 14. An inner line created with a 512x512 pixel lineart texture, shown from multiple distances

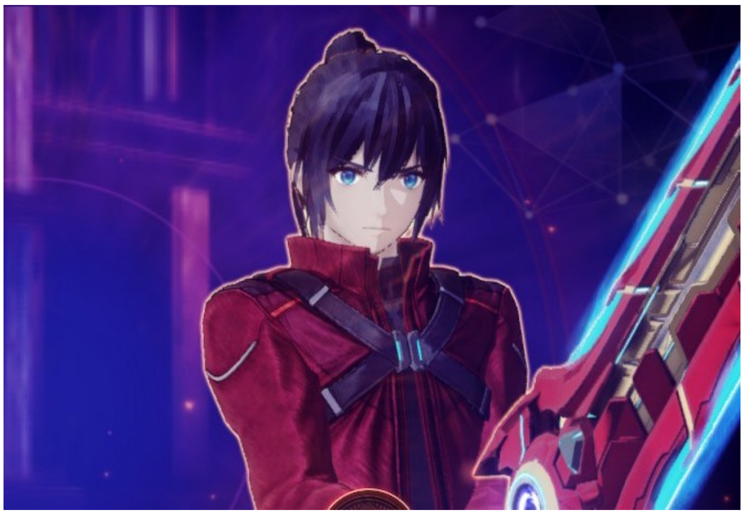
The downside to Limited UV texturing is that it requires a specific and carefully constructed topology, and if the model has a large number of details, the model's polygon count can grow into high numbers. In addition, the specificity and careful UV mapping that is required for every model make this method labor-intensive and slow (Motomura 2015).

3.3.3 PBR textures

PBR textures refer to the types of textures commonly used along with physically based rendering. They use multiple texture maps to to control different surface attributes of the object's material, such as reflectivity and micro surface details (Mesquita 2021).

Although PBR textures are rarely used in 2D-stylized 3D game art, some games have utilized them to enhance the 2D-stylized look. An example of this can be seen in the game Borderlands (2009), where the character models have thick comic book-like outlines, but the textures use realistic shading (Picture 15). Additionally, aspects of PBR textures, such as normal or metallic maps may be used to enhance the art style even when other textures are hand-painted. Xenoblade Chronicles 3 (2022) uses cel-shading and hand-painted textures for the character faces, but PBR shading and texturing for the clothing (Picture 16). This allows for the clothing to have more realistic details and reflections while maintaining the overall anime look for the characters.

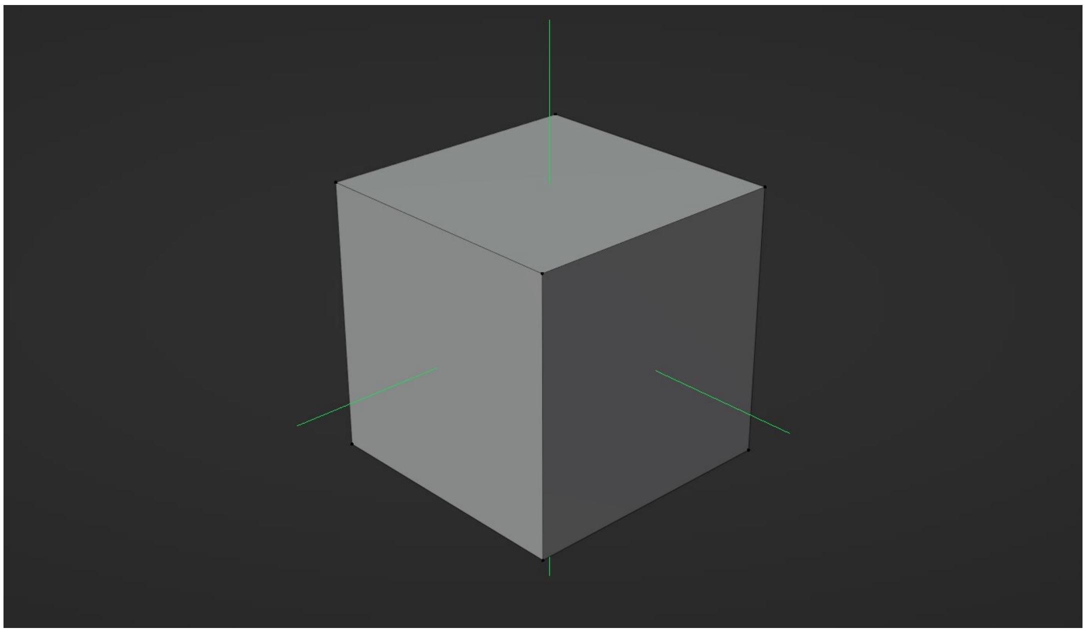

PICTURE 15. PBR textures can be mixed with outlines and other techniques to create a 2D-stylized look (Borderlands 2009)


PICTURE 16. Character with a mix of cel-shaded hand-painted textures, and PBR textures (Xenoblade Chronicles 3 2022)

Mixing 2D and PBR aspects may break the strict illusion of two-dimensionality, but it can also enhance the mood and create a unique art style. PBR textures can be created in similar ways as painted textures, either with 3D modeling or painting software, or with the use of photographs or traditional painting software.

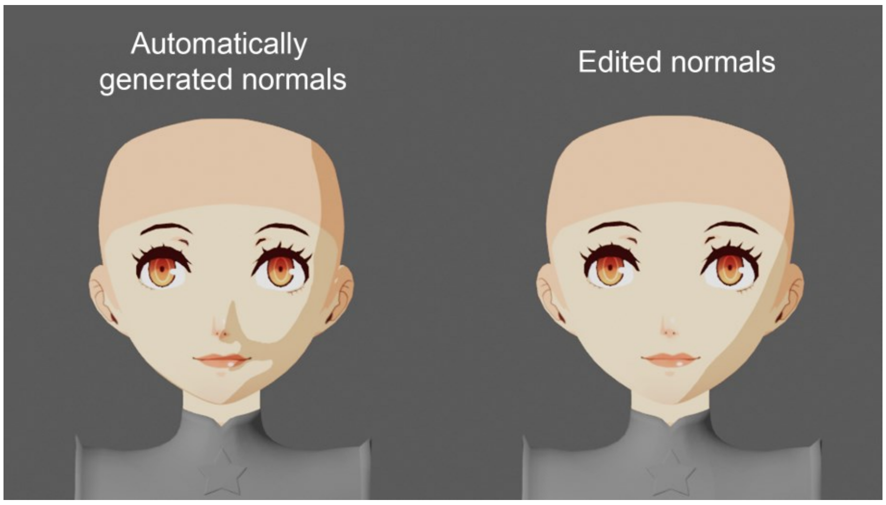
3.4 Normal editing

The mathematically calculated shadows of a 3D character rarely look immediately the same as a 2D character's would. In a 2D illustration, every line and shadow is intentionally decided by the artist, and to imitate the look in 3D, the same amount of intentionality is required. One way to do this is with the modification of the model's normal data. Normals are the direction that a polygon is facing at, and they are used to calculate how light interacts with the model (Chopine 2011, 34). Editing normals changes the way that light appears on the model, making them a useful way to manipulate an individual object's shading. Inside a 3D modeling software, normal directions are often illustrated with lines (Picture 17).



PICTURE 17. The normal directions of a 3D model depicted as green lines

As a 2D-stylized character's model might follow odd or unnatural shapes, the automated normals often end up causing shadows to be cast in areas where they should not appear, breaking the illusion of a hand-drawn character or object. This issue commonly appears in human faces, where the character's mouth, nose, and eyes can cause undesired artifacts and wrinkles. As seen in Picture 18, the automatically generated normals cast unintended shadows near the character's mouth, while the edited normals maintain the illusion of a 2D character better.



PICTURE 18. Automatically generated normals (left) versus edited normals (right) of a 3D anime character model

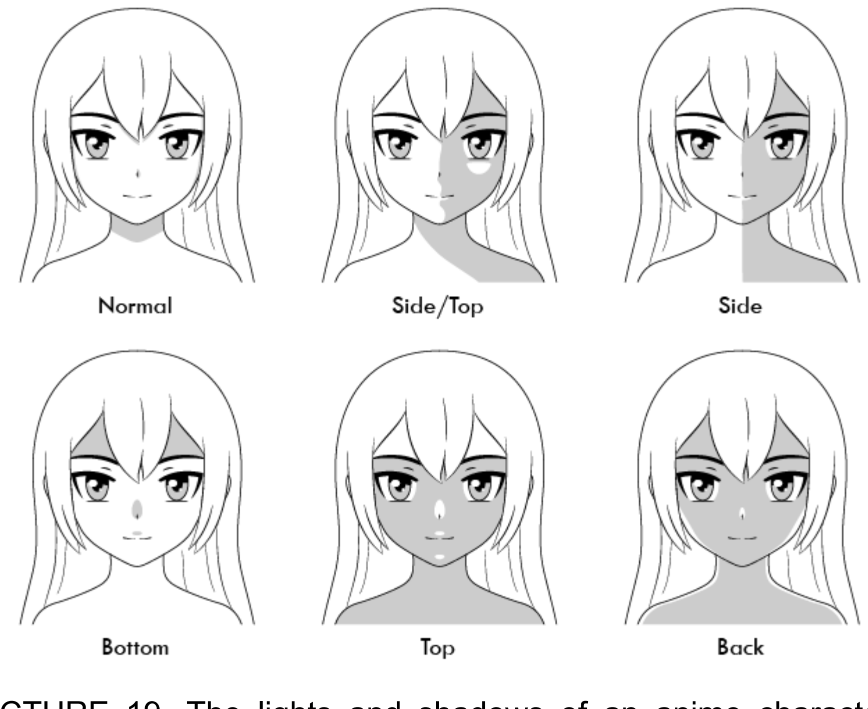
There are various ways to manipulate normals, and this chapter will focus on two of them: transferring normals and manual editing. Which technique should be used depends on the intended art style, as well as the time and resources available to the artist. Editing normals for multiple characters can be time-intensive, and therefore it may sometimes be better to utilize a simple technique, even if it leads to less detailed results. In addition, some art styles may not require complicated shadows; for those situations, a quick normal transfer might be preferable, if any editing is needed at all.

3.4.1 Normal transfer

One way to manipulate normals is by transferring them from one object to another. The transferrable object can either be a simplified version of the initial model or a completely separate mesh, such as a sphere. For example, when normals are transferred from a sphere to a human character's face, the facial details, such as nose or mouth, will no longer cast any shadows, similar to how a drawn character's facial features might not. The normals in Picture 18 were edited this way, and it can be seen how the shadow no longer reacts to the shape of the mouth. This is a quick and simple way to give the characters a more cartoon-like look and to avoid most unwanted artifacts.

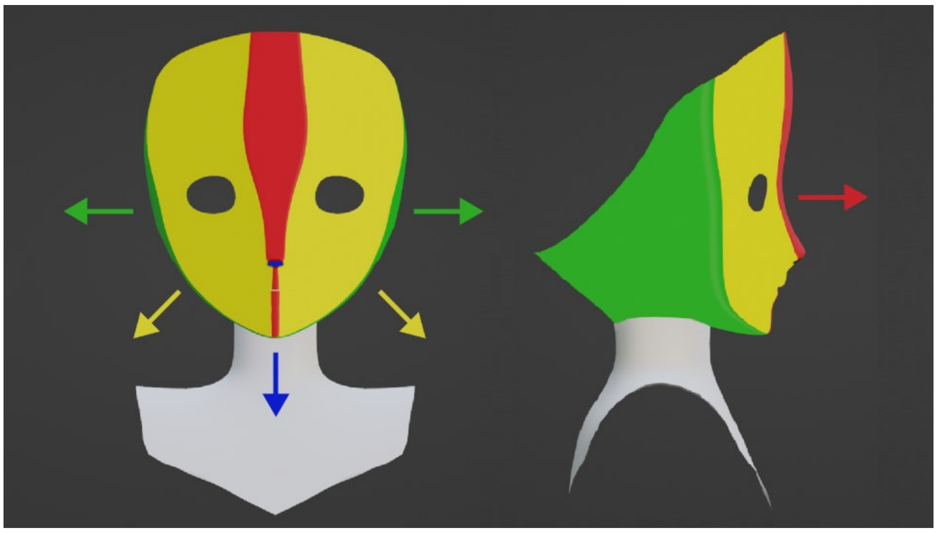
3.4.2 Manual editing

Normals can be manually edited either with a normal map or by hand so that the shadows behave in a way that is even more reminiscent of 2D lighting. This is especially useful when lighting faces: a 2D character's facial shadows are often simplified, or heavily stylized for artistic or dramatic effect. The shadows may snap from one position to another, unlike photorealistic shadows that slowly shift along with the light. For an anime character, the facial shadows might look something like the shadows in Picture 19. The edges of the shadows are sharp and distinct and do not use gradients or other interpolation.

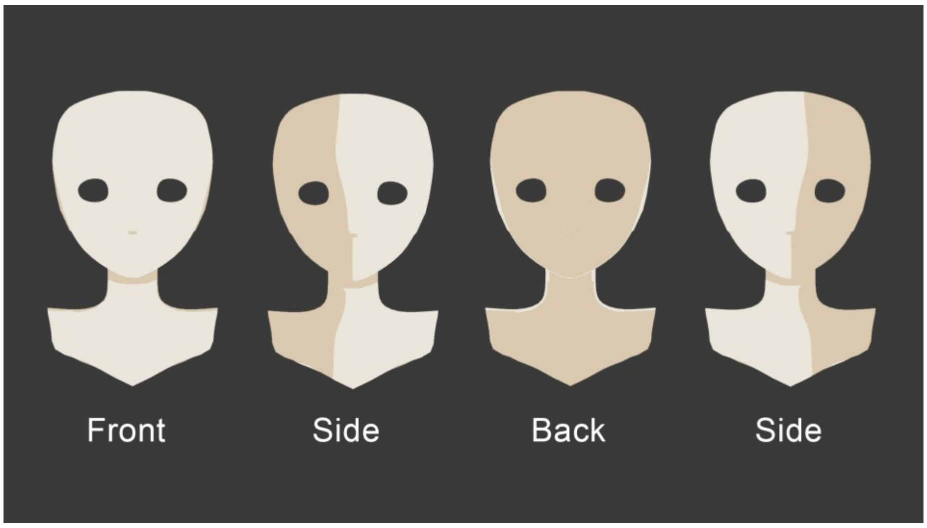


PICTURE 19. The lights and shadows of an anime character (Animeoutline n.d.)

To translate this type of snapping anime lighting to 3D, the face’s normals need to be manually edited. The head is divided into facial zones based on the direction that the normals should face when the model is lit from the front (Activemotionpictures 2022). Picture 20 shows an example of different facial zones on a 3D anime character: the red zone’s normals face forward; the green zone’s normals face to the either left or right; the yellow zone’s normals face a 45-degree angle, either left or right, and the blue zone’s normals face downwards. When the light is rotated, the shadows change position, as seen in Picture 21. The vertices in each zone share the same exact normal direction, making the lighting snap from one shadow position to another without interpolation.



PICTURE 20. An example of facial zones for a 3D anime face. The arrows show the area’s normal direction



PICTURE 21. The shadow rotations for a 3D anime face with manually edited normals

3.5 Shadow coloring

Colors in anime are specific and often intentionally picked to represent the character and atmosphere (Motomura 2015). To properly replicate the feeling of anime, it can be beneficial to use a separate shading map to change the shadowed colors, instead of having a single ambient shadow color that is multiplied over the base colors.

3.6 Rembrandt triangle

A common feature in anime shading is a triangle that appears on the character’s cheek in certain positions, as seen in the side/top view of Picture 19. This triangle shape is known as the Rembrandt triangle; it is a lighting setup made famous by the artist Rembrandt Harmenszoon van Rijn and is commonly used in painting and photography to create a dramatic yet natural lighting effect (Lagregor 2017). To replicate the Rembrandt triangle in a 3D model, a specifically constructed topology is required: the vertices under the eye need to be shaped in a triangular shape so that the moving shadows form the shape in the desired spot (Activemotionpictures 2022). This can be done either with the use of manual normal editing or by using a separate shading map where the triangle is left lit even when the rest of the model is under a shadow.

4 CREATING A 2D-STYLIZED 3D MODEL

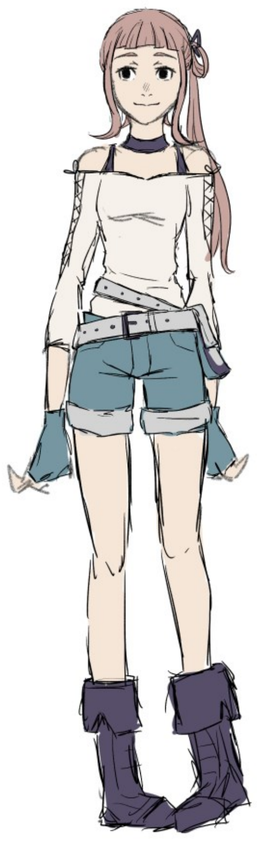
To understand the techniques discussed earlier better, I set out to design, model, texture, and render a 2D-stylized 3D character from scratch. The goal was to test the different methods, face the challenges that might arise, and show the results. The final render will be shown as an image, but it will be captured in real-time in Unity.

4.1 Concept art

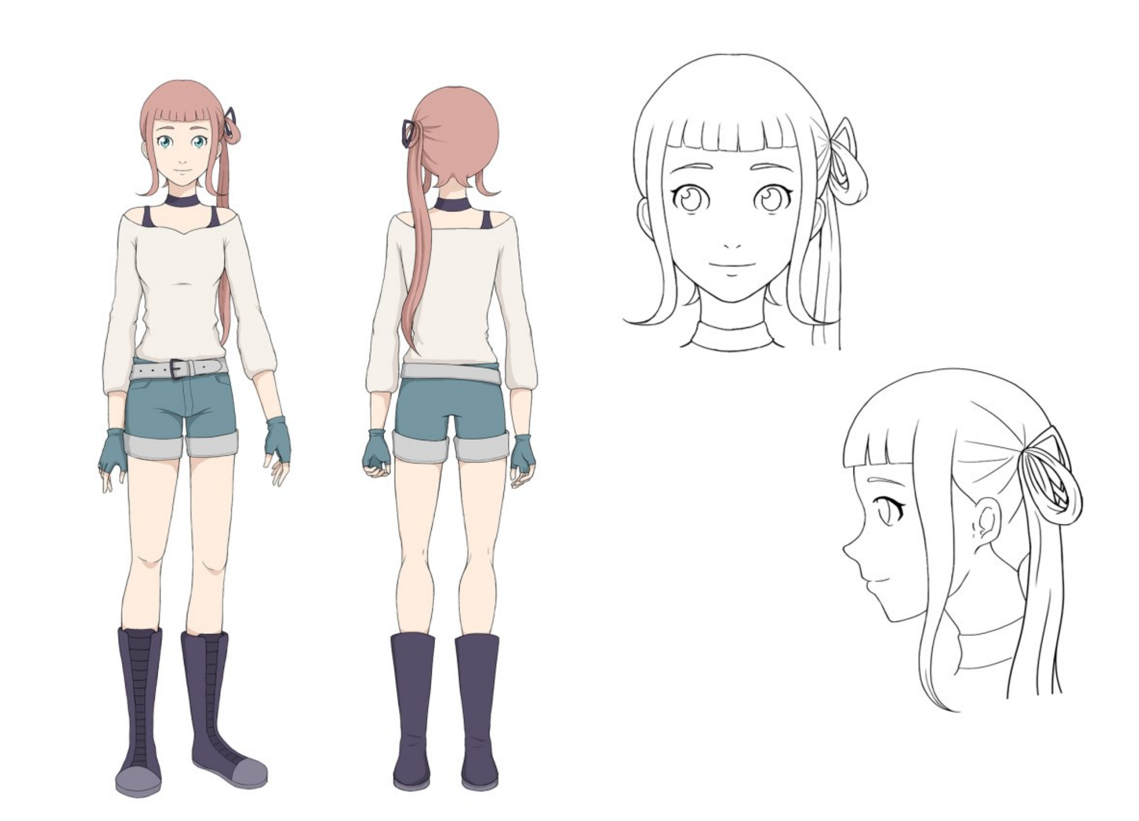
In order to have a clear idea of what the 3D model should look like, I first created concept art for the character. The base idea for the character was an anime girl from an urban fantasy setting. The art style was inspired by anime and cartoons such as Yuri on Ice (2015) and Avatar: The Last Airbender (2005), both of which have round and expressive eyes and mostly realistic body proportions. I gathered various screenshots from the shows to form an inspiration board to use as a reference for the character concept (Picture 22). The purpose of this board was to inform the general art style and facial shape, even if the color scheme and clothes of the character were to aesthetically differ.

PICTURE 22. Inspiration and style reference board for the concept art (Yuri on Ice 2015; Avatar: The Last Airbender 2005; Howl’s Moving Castle 2004; modified.)

I sketched out different character designs and eventually settled on the design seen in Picture 23. The character has light pink hair with a side ponytail, tall boots with crisscrossed lacing, blue shorts, a thick belt, and a plain cream-colored shirt. Due to time constraints, I decided to go with a simple design that would not have too many complicated details or accessories while still allowing me to test the techniques discussed earlier in the thesis.


PICTURE 23. Initial character design concept

The final concept art, seen in Picture 24, was drawn in Clip Studio Paint and shows the whole character from the front and the backside. The character design itself was slightly simplified with the removal of the second belt and the simplification of the shoes. The character's face is shown from the front and from the side to give a clear reference for the proportions of the 3D model. The concept art resembles anime in style with shadows and highlights so that the colors and shadow placements can be used as a rough basis for the model later on.

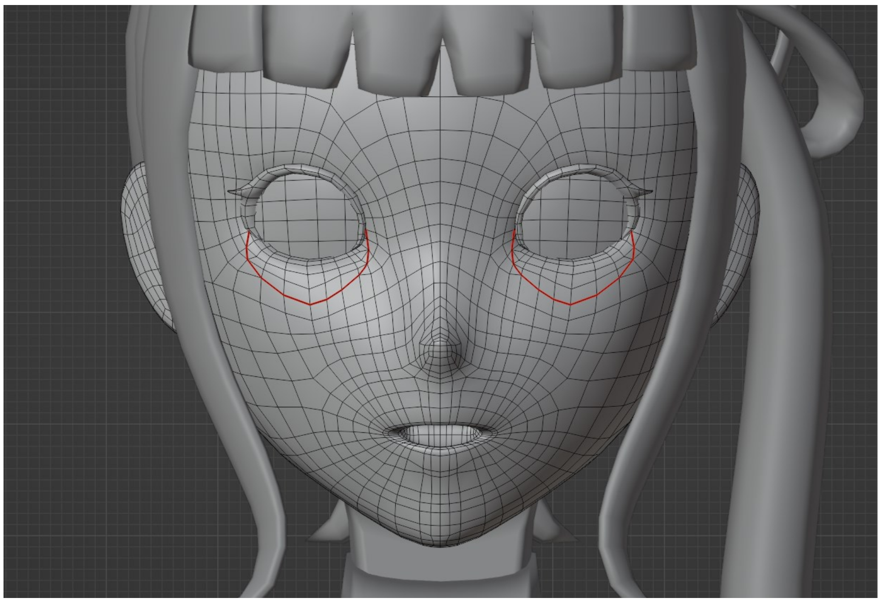


PICTURE 24. Concept art of the character

4.2 Character modeling

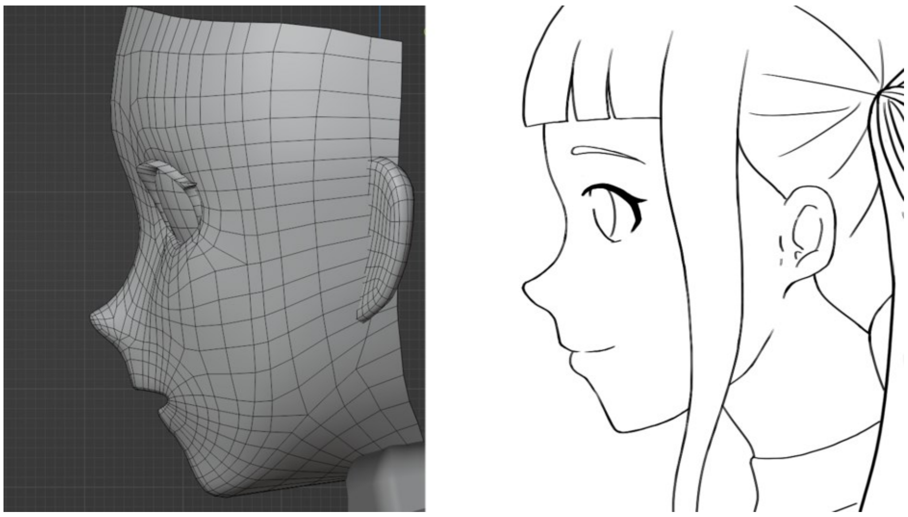
The character was modeled in Blender, utilizing general box and polygon modeling methods. Using the concept art as a guideline, I first created an overall body shape and then modified the details and added clothing and hair. The hair was initially modeled with the use of curves, which were then converted into meshes and manually cleaned up and joined together.

Highly detailed areas, such as the character’s face, provided difficulty due to having to make important decisions regarding the topology. As the character’s normals would be manually edited later, it was important to already consider where the shadows should snap and which areas should be highlighted. For example, the areas underneath the eyes were modeled in a V-shape so that it would later be possible to create Rembrandt triangles during normal editing (Picture 25).



PICTURE 25. V-shapes under the character's eyes allow for the use of Rembrandt triangle later

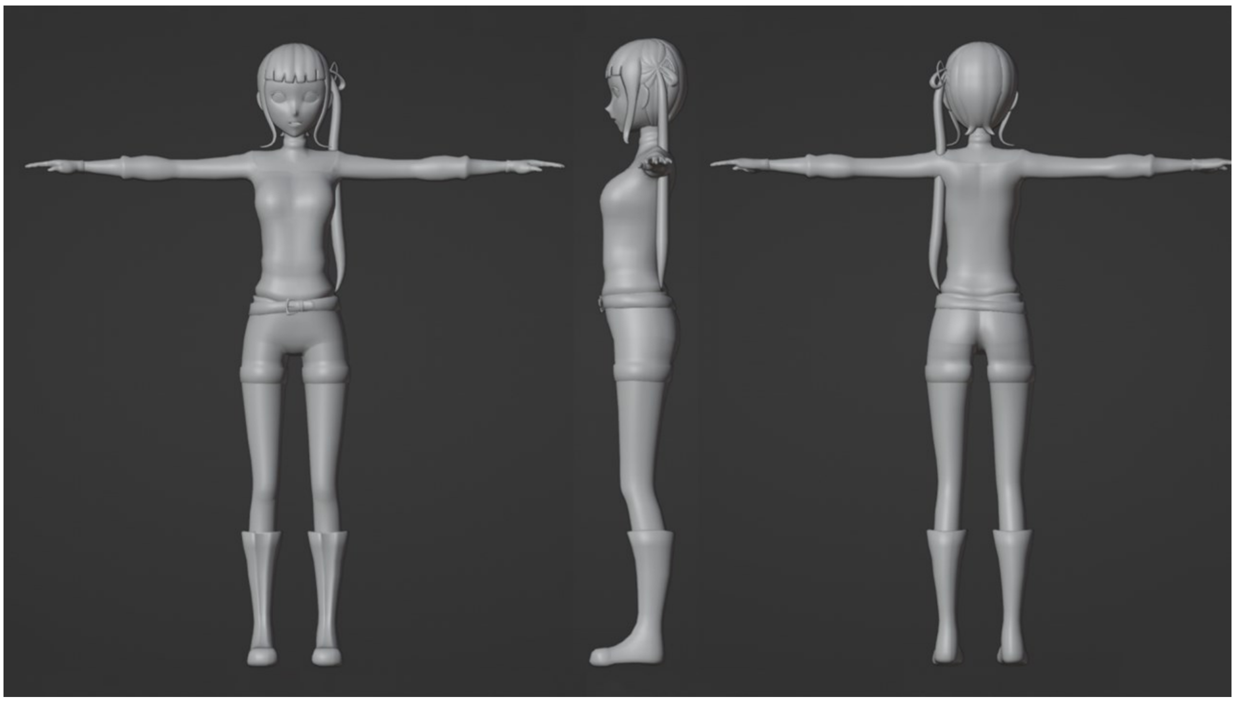
Head, cheeks, and eye shape also created issues. In a 2D drawing, the eye or face shape does not have to remain completely consistent when the character is drawn from different angles, but with a 3D model, the shape needs to work and look good from any angle. For the eyes to appear the correct shape from the front, they ended up looking slightly different from the concept art when viewed from the side (Picture 26). In these situations, I decided to follow what looked the most pleasant overall, rather than trying to force the model to strictly follow the proportions of the concept art.



PICTURE 26. Comparison between the 3D model and the concept art head from the side

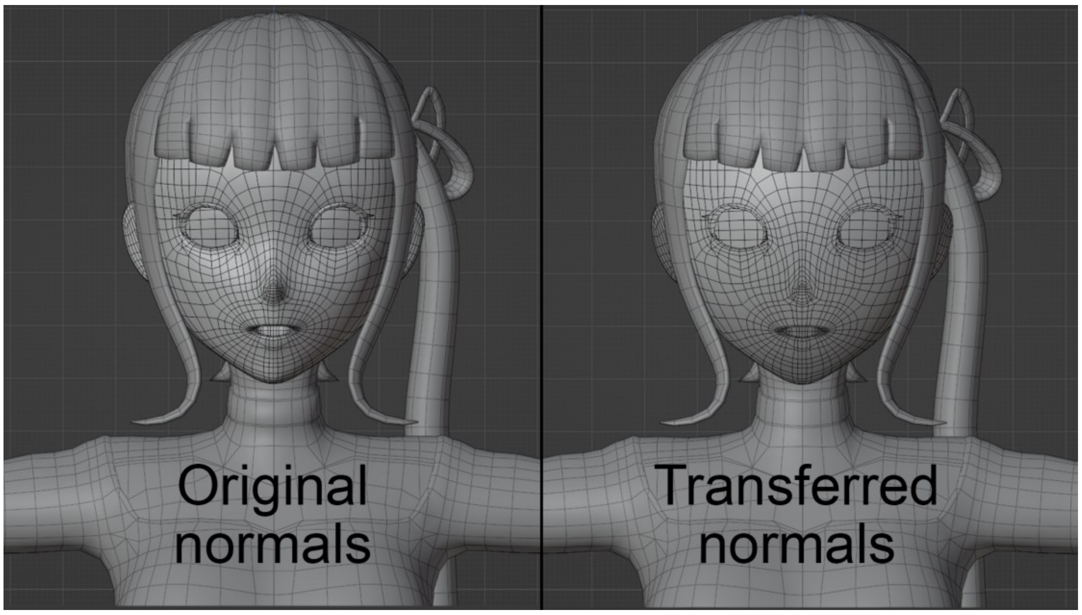
The character’s eyes were created by using a hand-painted eye texture on a slightly curved plane. The texture can be scrolled in the eye’s shader in Unity to simulate eye movement. This way I could avoid any possible complications that the round shape of a more realistic eyeball might cause.

The fully modeled character model can be seen below in Picture 27. In total, the complete model’s polygon count is 22,276 triangles.


PICTURE 27. The finished character model

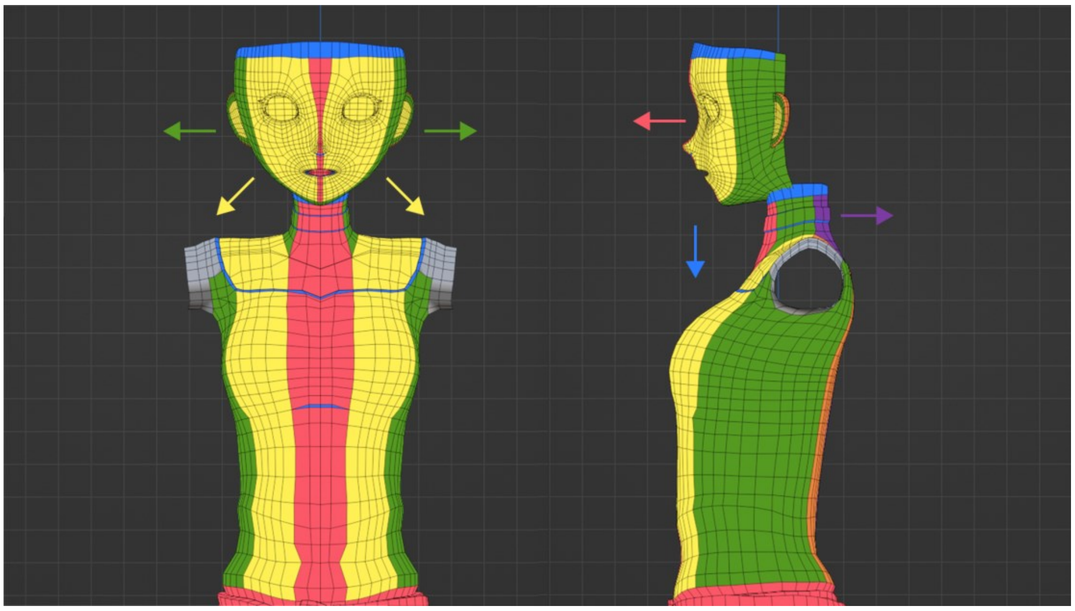
4.3 Normals & shadows

The normal editing process started with identifying sharp edges, applying automatic smoothing, and transferring the normals of a sphere to the character’s face. This allowed for a fast initial look at how the model might look with shadows and provided a good starting point for the proper editing process (Picture 28).



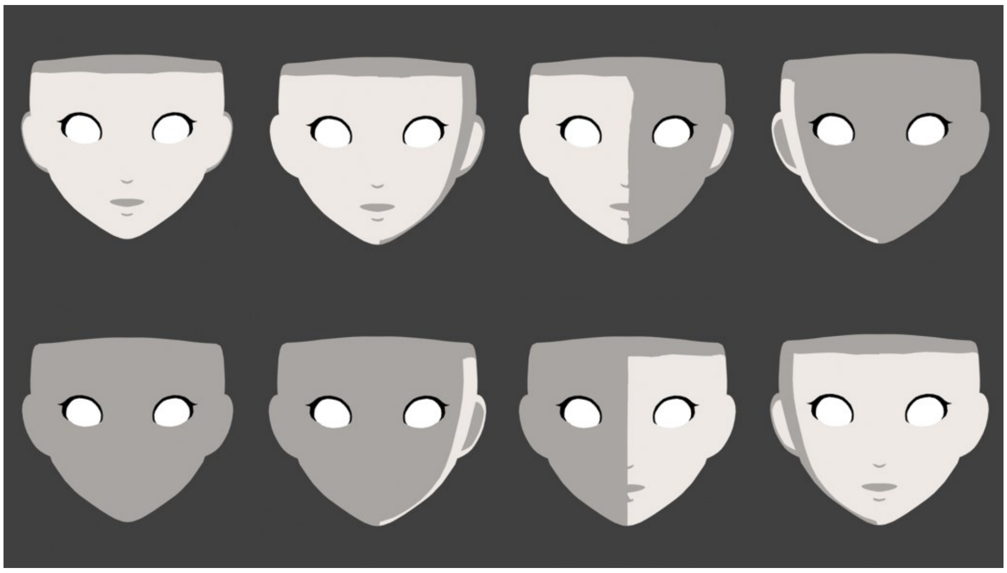
PICTURE 28. The character model’s face with transferred normals

Although the quickly edited normals may be satisfactory for other projects, for this project manual editing was needed to create the snappy anime lighting discussed earlier. The face and body were divided into zones that would determine the normal direction (Picture 29). After the zones were decided, I edited the normals one zone at a time. For each zone, I first selected the vertices and then pasted the respective normal direction to the whole area, so that all vertices in a zone would share the same exact normals.

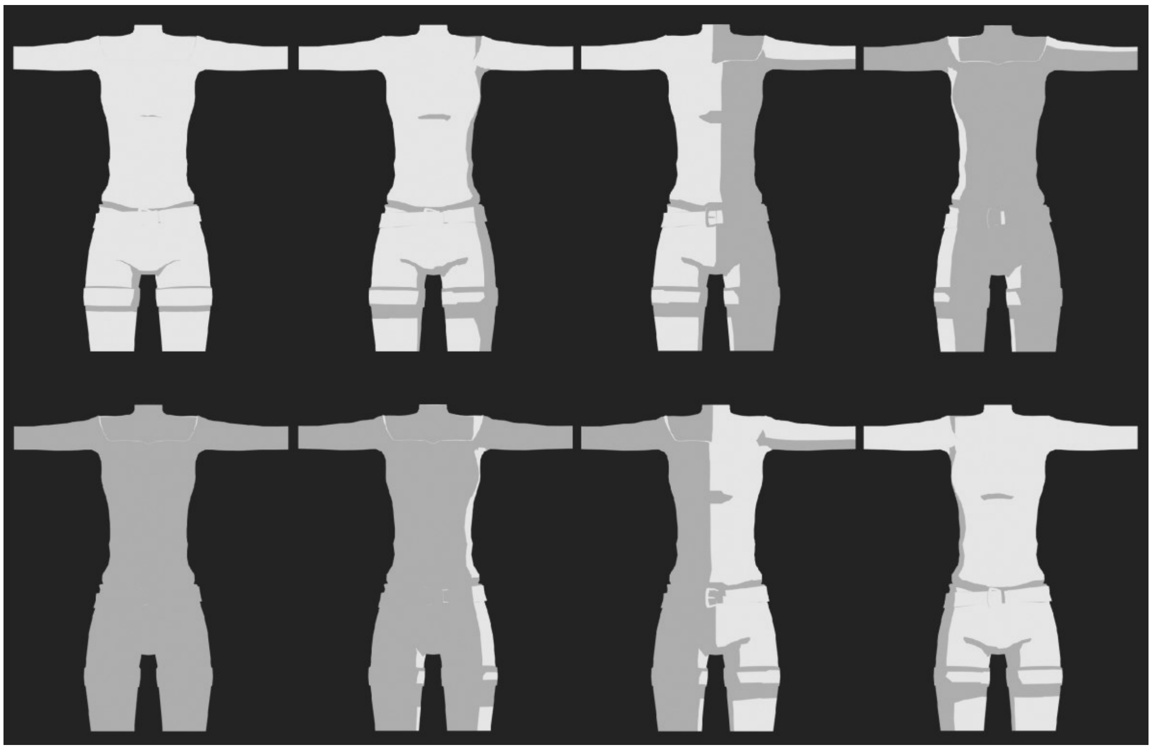


PICTURE 29. Shadow zones for the model’s head and upper torso

For areas that are permanently shadowed, such as the underside of the character’s nose, the normal direction was set as either facing downwards or upwards, depending on the location of the vertices. This way the areas remain shadowed in most lighting directions (Picture 30). The overall result of the normal editing is shadows that snap from one position to another without interpolation and with minimal artifacts (Picture 31).



PICTURE 30. The character’s mouth, underlip and bottom of the nose remain shadowed regardless of the light direction



PICTURE 31. Shadow rotations for the character’s torso

4.4 Character texturing

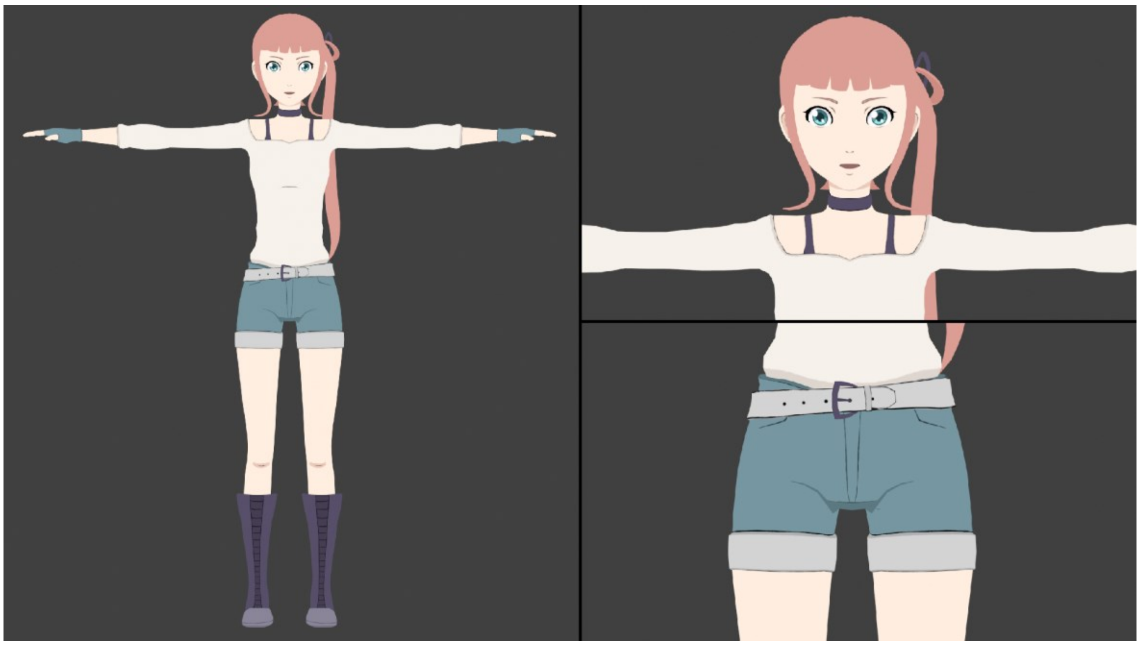
For the character’s textures, I tested the two main methods discussed in Chapter 3.3: hand-painted texturing and limited UV texturing. After textures for both styles were finished, one was chosen for the final render.

4.4.1 Hand-painting

For the hand-painted version of the character, I unwrapped the model using general UV-unwrapping methods and then manually painted it in Substance Painter. I used simple colors for the base and then painted the permanent shadow details on a separate layer. For this model, the permanent shadows were quick to identify due to the manual normal editing that was done beforehand. As such, I already had a good idea of what areas should be painted, and many of the areas were clearly visible within the topology itself.

Inner lines were painted in any area that has a lineart in the original design but does not count as an edge of the model, such as the shoe’s lacing and the character’s nostrils. The limitations of hand-painting became especially apparent during this process, as I had to return to rethink and change the size of the UV planes to keep the drawn lines consistent in resolution.

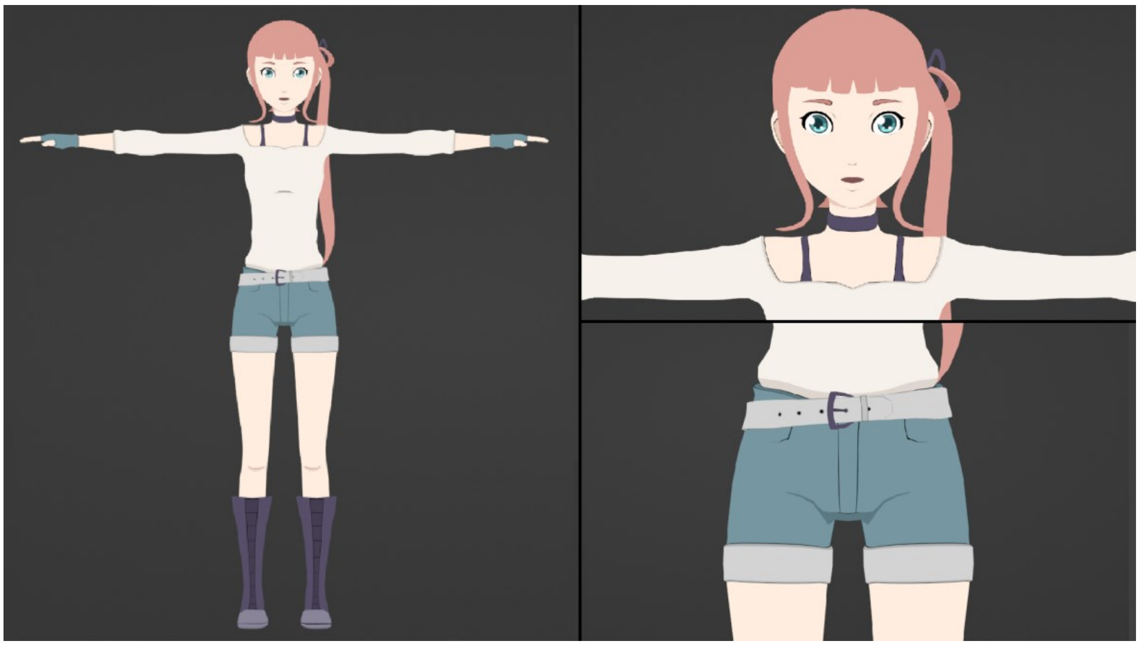
The final result of hand-painting can be seen in Picture 32. Hand-painting made it easy to create permanent shadows and details even in areas where the topology is simple. On the other hand, the inner lines proved to be difficult to paint, as they easily turned out too uneven and rough.


PICTURE 32. The character model with hand-painted textures

4.4.2 Limited UV texturing

For limited UV texturing, I first figured out the edges that would have inner lines. I then unwrapped each area, straightened the UV islands into squares, and positioned them within the UV map so that there is enough space to add in the lineart texture later.

I created a texture for the model's base colors and permanently shadowed areas in Substance Painter. For the inner lines of the model, I created the lineart texture map in Photoshop by manually creating black blocks around the edges that are supposed to have an inner line on them. Finally, I repositioned the UV vertices to change the weight of the lines to make them consistent and more natural. The outcome can be seen in Picture 33.


PICTURE 33. The character model with limited UV textures

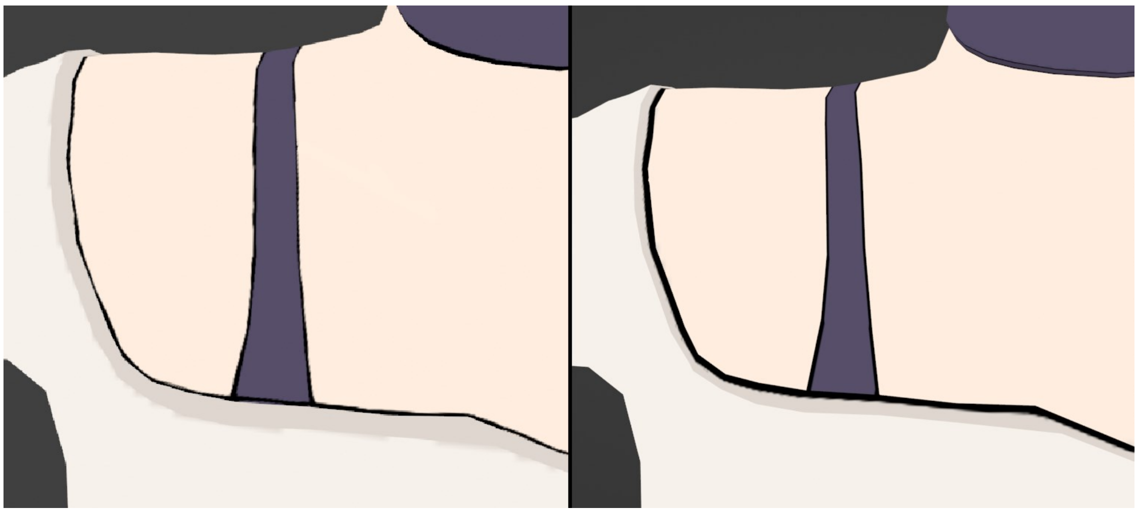
4.4.3 Comparisons

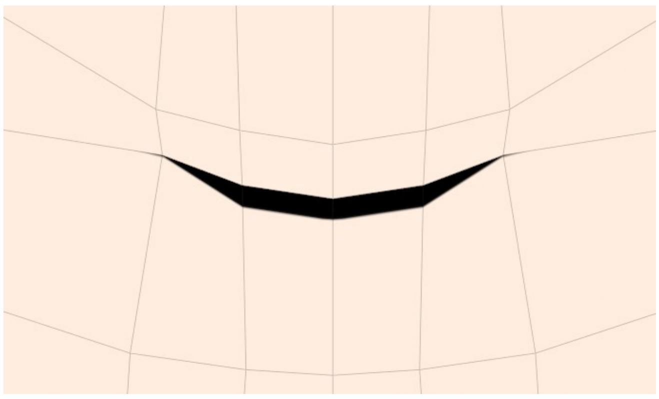
The limited UV texturing required a more demanding and time-consuming setup than the hand-painted textures. For the lines to appear clean, the UV islands had to be straightened and carefully positioned. Every line had to be manually added to the line texture and then fine-tuned in the UV map so that the line width stayed consistent and faded out as needed. In addition, this texturing method required careful attention during the modeling process, as lines can only be added to areas where vertices appear.

Another downside to the limited UV texture is that it only works with plain colors. For example, if I wished to add a tiling paper texture with the base colors to give the model a paper-like look, the texture would stretch and have a non-uniform shape. However, if an image or some other kind of drawn feature is needed in an otherwise limited UV textured model, it is still possible to use hand-painting for that specific part. This was how I handled the holes in the belt: even in the limited UV texture, the holes are hand-painted, as the only other way would have been to model the circles in the mesh itself, which seemed inefficient for a tiny detail.

As mentioned before, the hand-painted textures are limited by the resolution of the image file. This meant that fine-tuning and cautious UV unwrapping was required to keep the lines consistent in quality. Additionally, hand-painting requires stable drawing skills in order to create smooth and clean lines, unlike in limited UV texturing, where the lines' width can be easily controlled by changing the UV vertice positions.

When comparing the line quality between the hand-painted and limited textures, the limited textures are sharper and clearer (Picture 34). On the other hand, whereas a hand-painted line can be drawn anywhere, the limited textures strictly follow the vertices which sometimes leads to overly angular lines even when the line is intended to be curved (Picture 35). This can be mitigated by using higher polygon models but can be difficult to fix if the need is noticed only when creating the lineart texture.

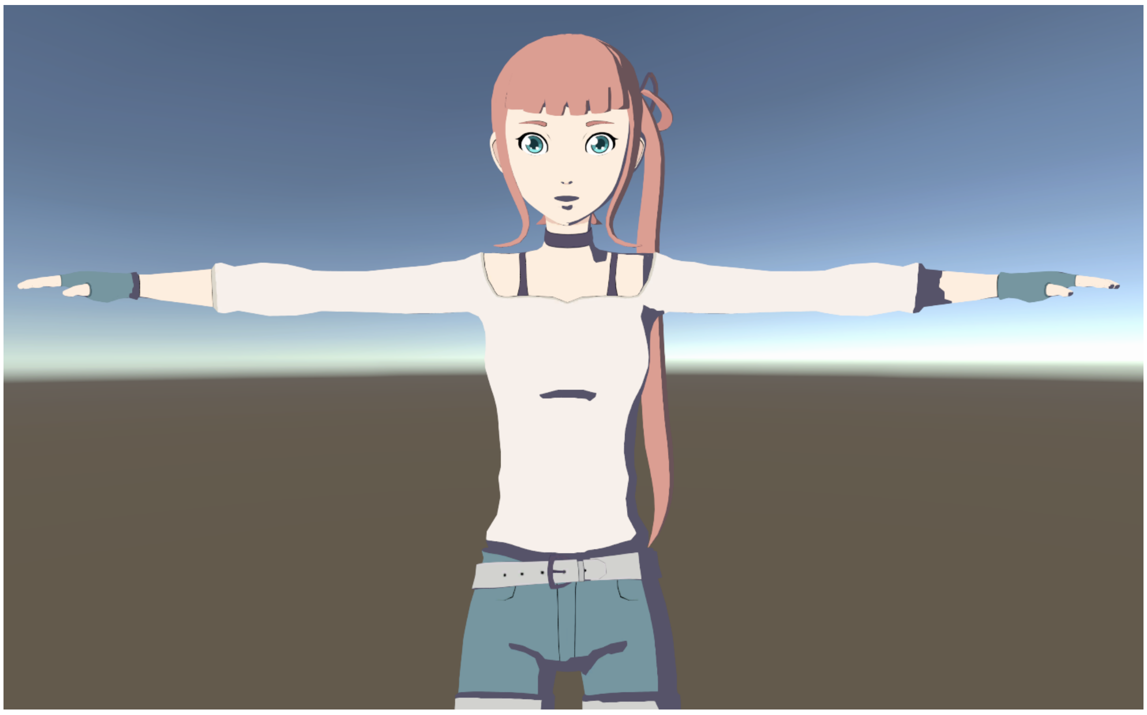

PICTURE 34. Hand-painted lines (left) and limited UV lines (right)


PICTURE 35. Lines in limited UV textures may appear angular

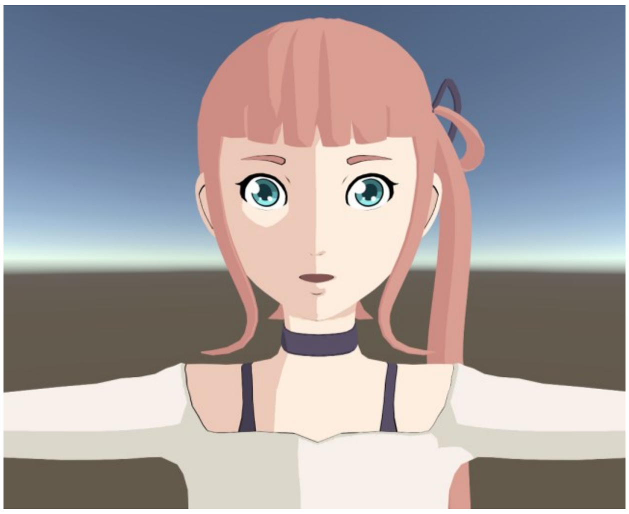
Ultimately, I was more satisfied with the results of the Limited UV texturing: although the curved lines were more angular in shape, the overall quality of the lines and color edges was higher. In the end I decided to continue with the limited UV textured version of the model.

4.5 Shader

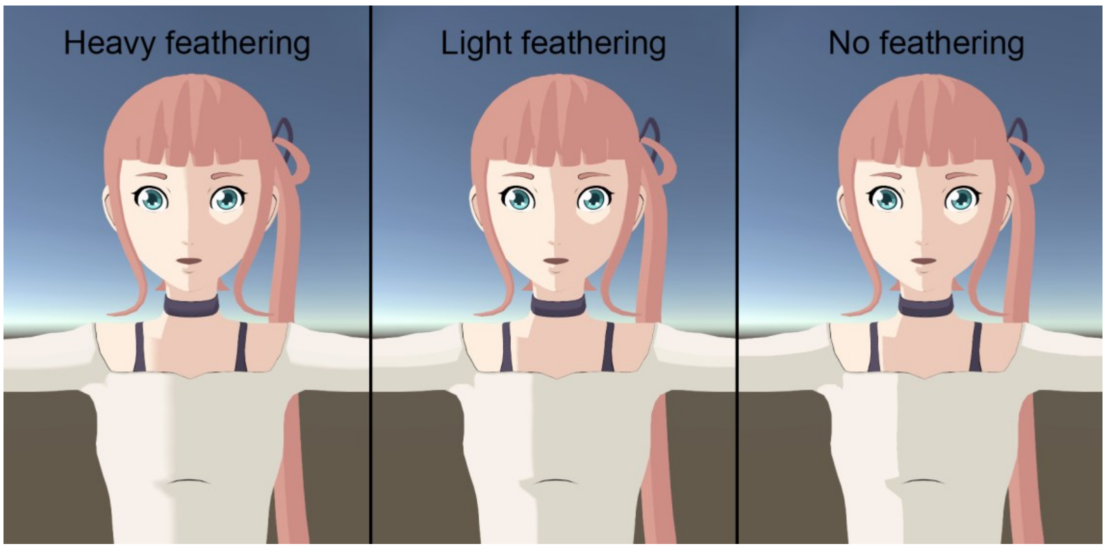
I initially considered writing my own shader in Unity but decided against it due to time constraints. Instead, I used the Unity Toon Shader—Unity's own publically available shader designed to be used with cel-shaded 3D characters—for the project, as the shader has built-in features for edge feathering, separate shading color maps, and outlines. Applying the Unity Toon Shader on the model resulted in the character looking like in Picture 36.


PICTURE 36. The character model in Unity with toon shader

As the shader allows for the use of separate shading maps, I created an alternate version of the character's base textures that matched the shadowed colors of the concept art. The triangles under the eyes were left lit in the shading map, allowing for Rembrandt triangles to form when the character’s face is in shadow (Picture 37).

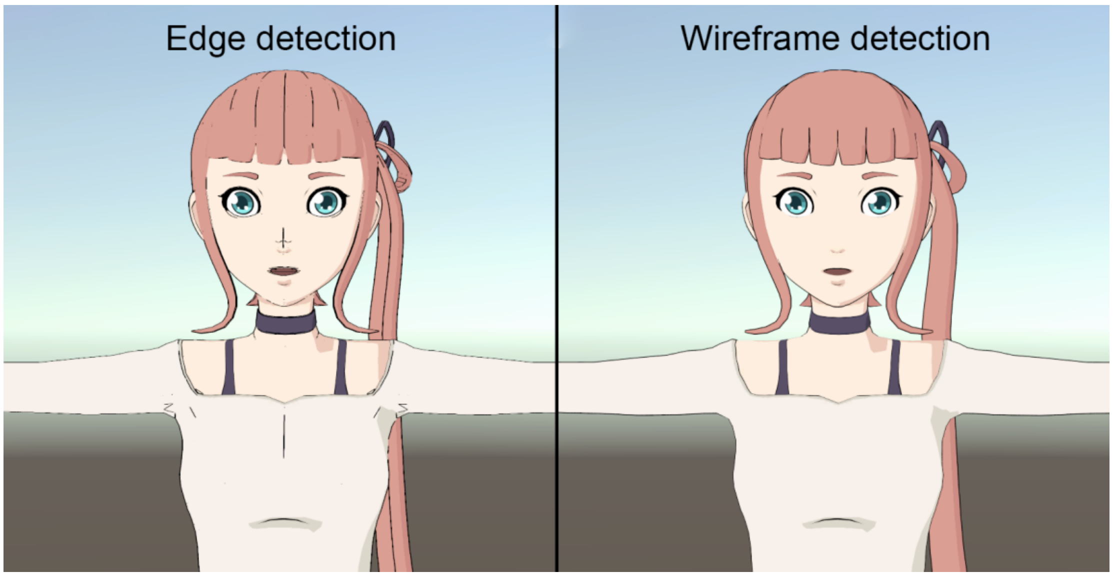

PICTURE 37. Rembrandt triangle forms under the character’s eye when the face in the shadow

Next, I tested different edge feathering for the model. Heavy amounts of edge feathering caused the shading to look softer (Picture 38). For this case, having no feathering at all was ultimately the better choice, as it resembled the anime-like art style of the concept art more.



PICTURE 38. Comparisons between different amounts of edge feathering

For the outlines, I tested both edge detection and wireframe detection-based methods (Picture 39). Edge detection was difficult to finetune for the model and caused lines to appear in places where they were not meant to appear, like the character’s nose and chest. Although the wireframe detected outlines left occasional holes and gaps depending on the model’s pose and view angle, they were still a better option in this case.



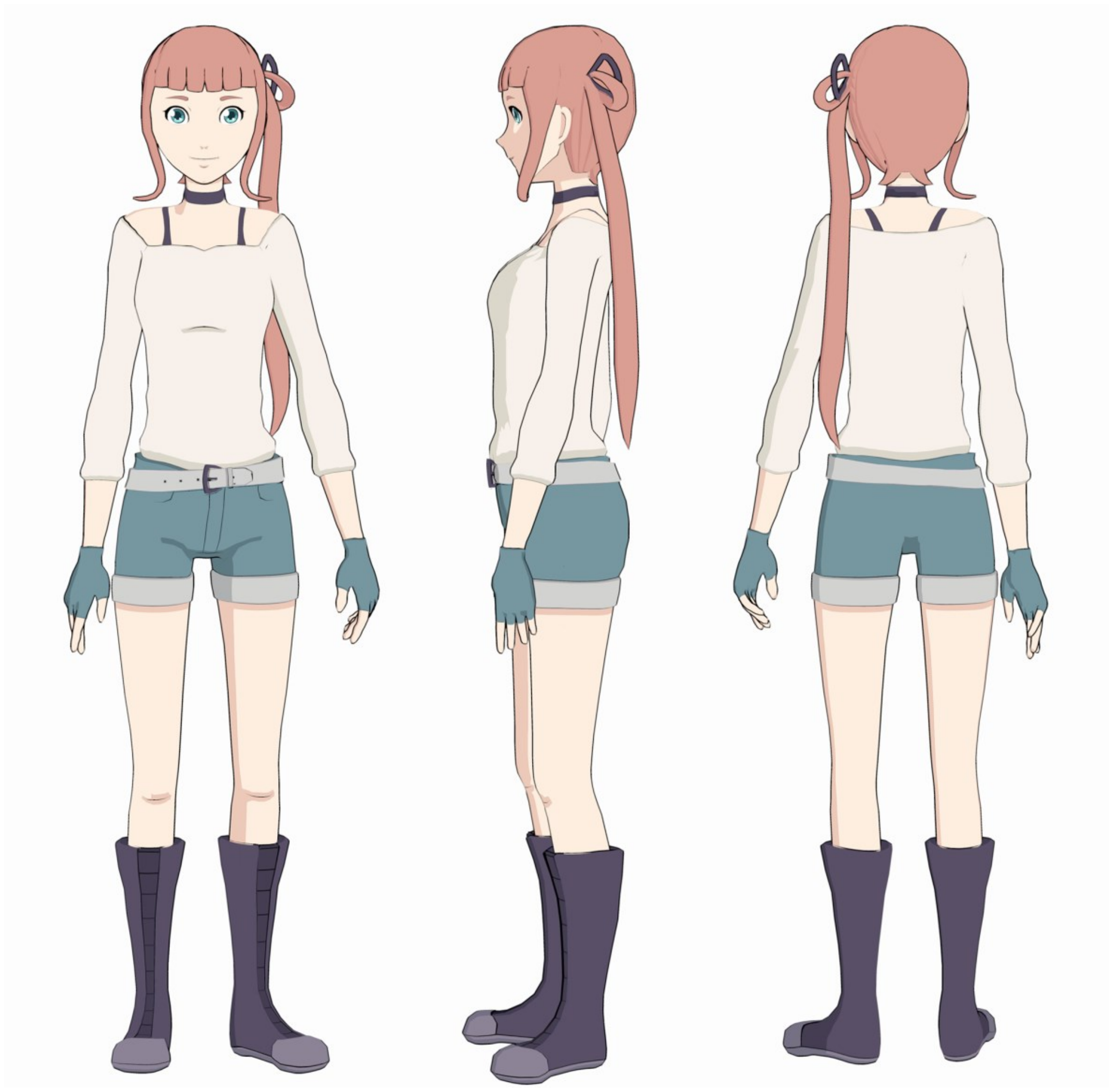
PICTURE 39. Outlines created using the edge detection and wireframe detection -methods for the character models

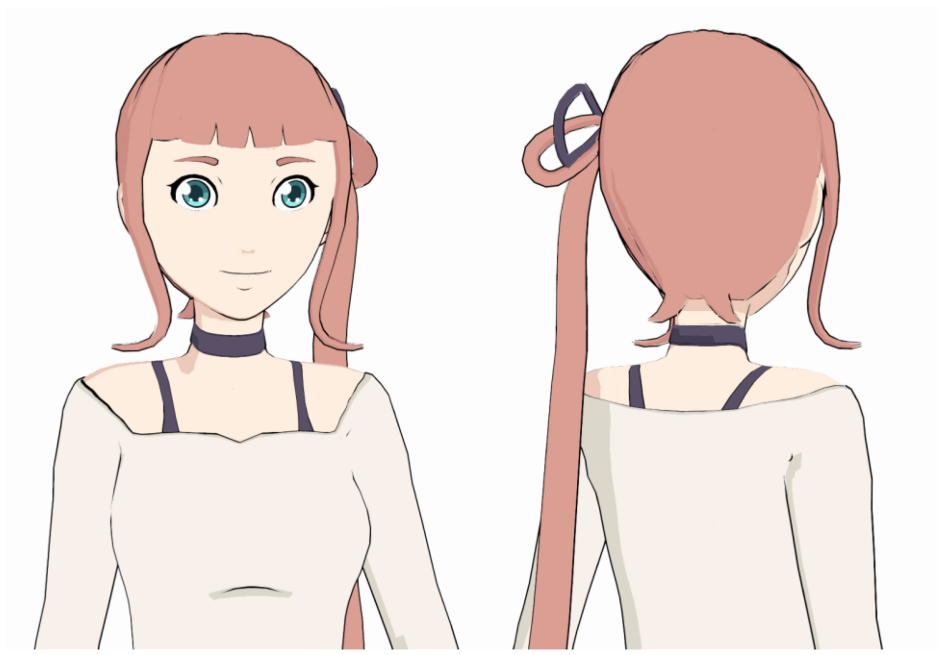
4.6 Rigging & posing

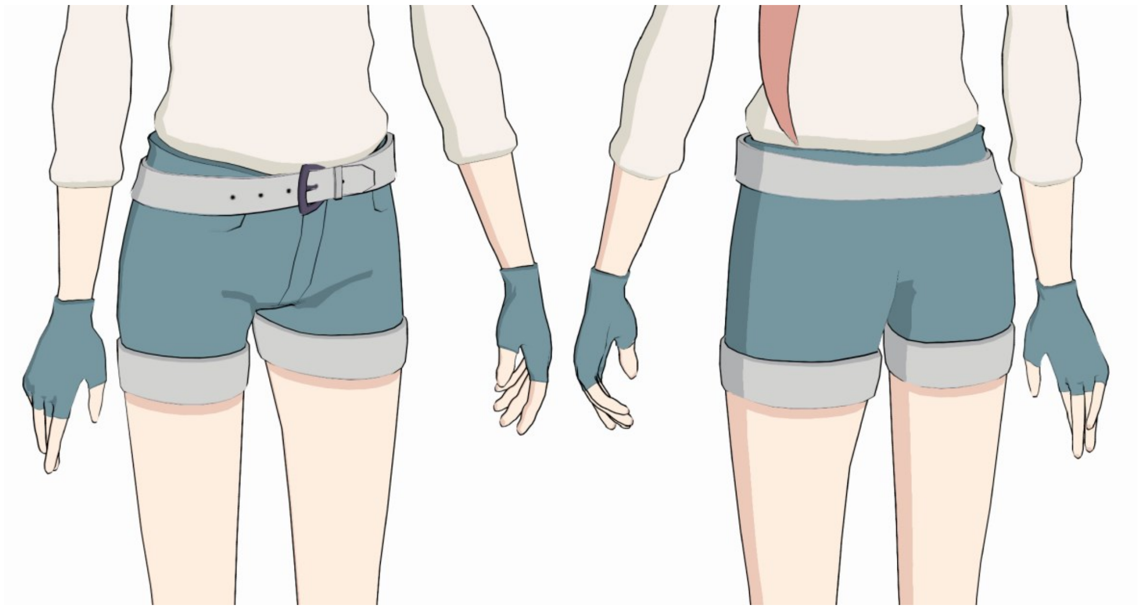
A rig is the skeleton of joints that moves a 3D model (Chopine 2011, 85). As the process of rigging had little to do with the topic of this thesis, I used Mixamo’s online auto-rigging tool and then modified the rig in Blender to add in bones for the unique details of the model, such as the side ponytail and the front hair. After the model was fully rigged, I positioned the character in the same pose as in the concept art.

4.7 Result

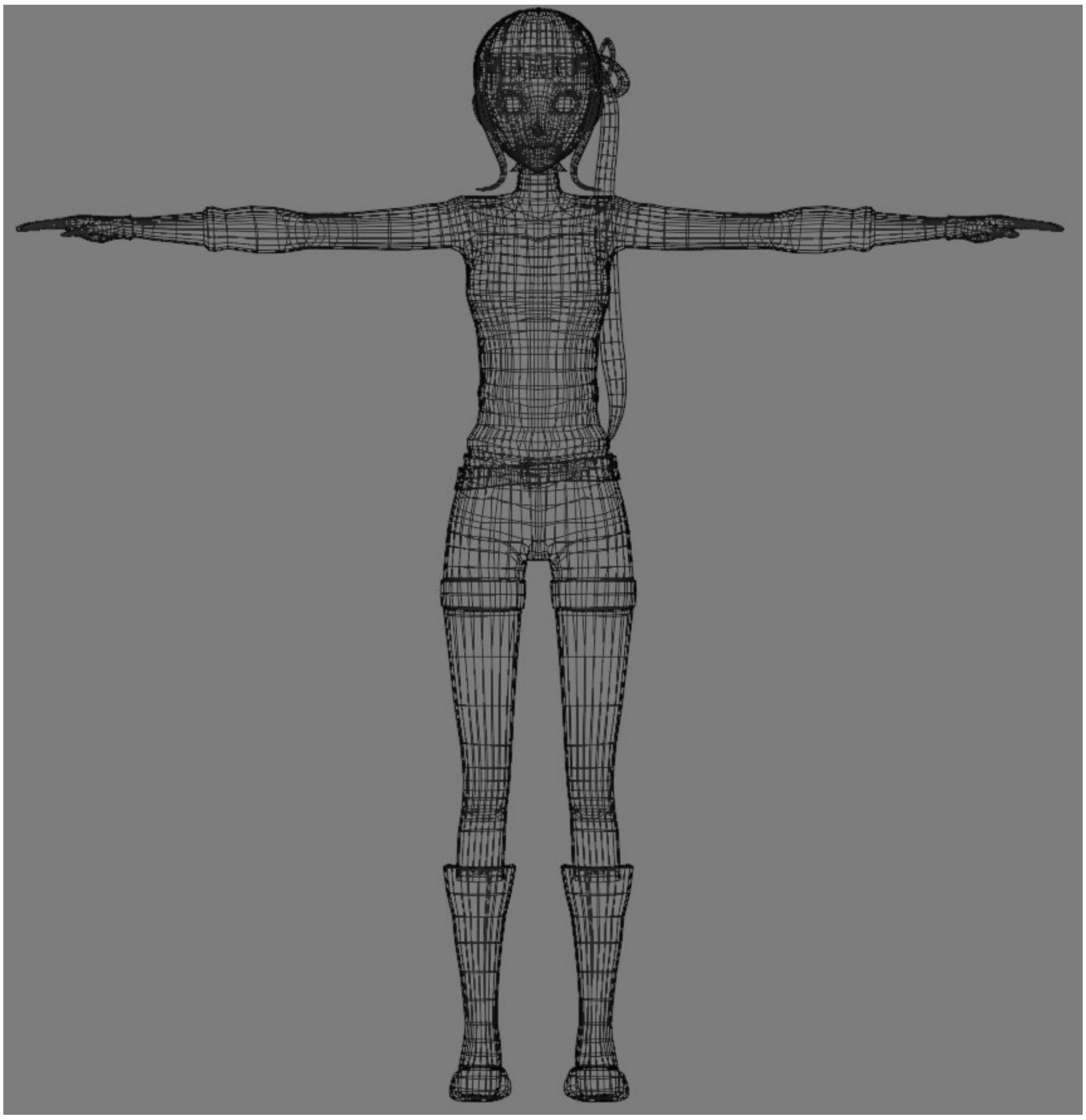
The final renders of the character were captured in real-time in Unity. The first three renders show the full character in the same pose as the initial concept art (Picture 40; Picture 41; Picture 42; Picture 43). A wireframe render was captured in Blender (Picture 44).


PICTURE 40. Final render of the character


PICTURE 41. Final render of the character


PICTURE 42. Final render of the character

PICTURE 43. Final render of the character


PICTURE 44. Wireframe of the character model

5 DISCUSSION

The end result of the thesis is the various methods that can be used to create 2D-stylized 3D game art. These different techniques can lead to varying outcomes, and which method should be chosen for a project depends on the intentions and resources of the artist.

The practical project worked as an example of the techniques. By creating a model in this style, I gained a personal insight into the difficulties one might face when imitating 2D art with 3D graphics. Even when the character design is intentionally simple, using techniques like manual normal editing and hand-painted or limited UV texturing can be a slow process. Each step requires careful planning, as making an error early on can cause trouble later: for example, one needs to keep in mind the placement of the shadows when modeling the base model, as the position of the vertices is important when creating specifically shaped shadows. My original plan had been to program my own shader for the project, but time constraints lead to me using Unity's toon shader instead, though this did not affect the visual end result. I am overall satisfied with the final result of the project—although there are still imperfections in the model, it fulfills its purpose of imitating a 2D character.

When considering whether 2D-stylized 3D art should be used for a project, it is important to consider the demands of the style. Manual normal editing and limited UV texturing, for example, require models with high polygon counts for the shadows and lines to appear smooth. The more complicated the character's design is, the more time will go into careful modeling, UV unwrapping, and inner line editing. As such the style might not necessarily be a good fit for teams with highly limited modeling resources. However, due to the nature of 2D-stylized 3D art, there is room to customize and choose the methods to fit the needs of each particular project—not every art style requires complicated snapping shadows or highly detailed textures.

Due to the constantly evolving landscape of 3D modeling, as well as this thesis' focus on humanoid characters and anime and cartoon-style inspirations, there are surely techniques that were missed during the research. In addition, as this thesis did not discuss anything related to animation, research into animating methods that can enhance the style might also expand the knowledge on the subject. Further research into these matters would provide a wider view of the style and its future possibilities.

REFERENCES

Activemotionpictures. Normal Editing for anime faces 101. Published on 14.6.2022. Watched 17.10.2022. https://www.youtube.com/watch?v=79XKQVAS9dI

Ameye, A. 2021. 5 ways to draw an outline. Published on 16.8.2022. Read on 13.9.2022. https://alexanderameye.github.io/notes/rendering-outlines/

Animeoutline.com. How to Shade an Anime Face in Different Lighting. N.d. Read on 7.11.2022. https://www.animeoutline.com/how-to-shade-an-anime-face-in-different-lighting/

Anjyo, K. & Hiramitsu, K. 2003. Stylized Highlights for Cartoon Rendering and Animation. IEEE Computer Graphics and Applications 23 (4), 54–61.

A Short Hike. 2019. Adamgryu.

Avatar the Last Airbender. 2005. Direction: Michael Dante DiMartino & Bryan Konietzko. Production: Nickelodeon Animation Studios. Original release: 21.2.2005 – 19.7.2008.

Borderlands. 2009. 2K Games.

Fear Effect. 2000. Eidos Interactive.

Fire Emblem Warriors: Three Hopes. 2022. Nintendo.

Genshin Impact. 2020. miHoYo.

God of War Ragnarök. 2022. Sony Interactive Entertainment.

Golus, B. The Quest For Very Wide Outlines. Blogus. Published on 19.7.2020. Read on 22.8.2022. https://bgolus.medium.com/the-quest-for-very-wide-outlines-ba82ed442cd9

Guilty Gear Xrd Sign. 2014. Arc System Works.

Chopine, A. 2011. 3D Art Essentials: The Fundamentals of 3D Modeling, Texturing, and Animation. 1st Edition. Oxford: Focal Press.

Hamlaoui, S. 2001. Cel-shading. Gamedev. Published on 13.7.2001. Read on 12.9.2022. https://www.gamedev.net/reference/articles/article1438.asp

Haven. 2021. The Game Bakers.

Howl's Moving Castle. 2004. Director: Hayao Miyazaki. Production: Studio Ghibli.

Jet Set Radio. 2000. Sega.

Jones, L. 2021. Cel-shading in Art and How to Use It. The Fields of Green. Published on 27.6.2021. Read on 21.8.2022. https://thefieldsofgreen.com/cel-shading/

Jones, S. 2012. Okami HD - Traditional art graphics in a modern game. Fanatical. Published on 17.12.2012. Read on 22.9.2022. https://www.fanatical.com/en/blog/okami-hd-traditional-art-graphics-in-a-modern-game

Komi Can’t Communicate. 2021. Direction: Ayumu Watanabe. Production: OLM. Original release: 7.10.2021 – 22.6.2022.

Lagregor, P. The Rembrandt Triangle In Photography: Beginner’s Guide. Improve Photography. N.d. Read on 18.9.2022. https://improvephotography.com/50146/rembrandt-triangle-beginners-guide/

Laud, J. 2017. History of the First 3D Video Games. It Still Works. Published on 22.9.2017. Read on 27.9.2022. https://itstillworks.com/12314899/history-of-the-first-3d-video-games

Mad World. 2009. Sega.

Mega Man Legends. 1997. Capcom.

Mesquita, L. 2021. Everything About PBR Textures And A Little More. Published on 4.1.2021. Read on 15.9.2022. https://www.artstation.com/blogs/luismesquita/PwEm/everything-about-pbr-textures-and-a-little-more-part-1

Metro GameCentral. 2015. Guilty Gear Xrd -SIGN- review – unreal fighter. Published on 6.1.2015. Read on 27.9.2022. https://metro.co.uk/2015/01/06/guilty-gear-xrd-sign-review-unreal-fighter-5010522/

Motomura, J. 2015. GuiltyGearXrd's Art Style : The X Factor Between 2D and 3D. Talk. Game Developers Conference on 2.–6.3.2015. San Fransisco. Watched 17.9.2022. https://www.youtube.com/watch?v=yhGjCzxJV3E

Naruto: Clash of Ninja. 2003. D3 Publisher, Tomy.

Ni No Kuni: Wrath of the White Witch. 2011. Namco Bandai Games.

Ōkami. 2006. Capcom.

PaRappa the Rapper. 1996. Sony Computer Entertainment.

PC Plus. 2010. The evolution of 3D games. TechRadar. Published on 11.7.2010. Read on 27.9.2022. https://www.techradar.com/news/gaming/the-evolution-of-3d-games-700995/2

Persona 5. 2016. Sega.

Phan, H. 2012. Avian Defender: Diffuse Only Model Workflow Analysis. Ryanhawkins’s VERTEX1. 118–127.

RocketBrush. 2022. How to Choose The Art Style That Fits Your Game. Published on 8.4.2022. Read on 2.10.2022. https://rocketbrush.com/blog/how-to-choose-art-style-for-your-game

ScouseGamer88. 2014. Pretty as a Pixel: The History of Cel-Shading in Video Games. Read on 22.9.2022. https://www.scousegamer88.com/2016/07/10/pretty-as-a-pixel-the-history-of-cel-shading-in-video-games/

Sly Cooper. 2002. Sony Computer Entertainment.

The Last of Us Part II. 2020. Sony Computer Entertainment.

The Legend of Zelda: Breath of the Wild. 2017. Nintendo.

The Legend of Zelda: Wind Waker. 2003. Nintendo.

The Simpsons Game. 2007. Electronic Arts.

Torchinsky, A. 2020. Cel-shading: some tricks that you might not know about. Published on 14.1.2020. Read on 12.9.2022. https://torchinsky.me/cel-shading/

Valkyria Chronicles. 2008. Sega.

Walla Walla Studio. 2022. 2D vs. 3D Animation: The Difference Between 2D and 3D Animation. Published on 8.2.2022. Read on 2.10.2022. https://wallawallastudio.com/article/2d-vs-3d-animation-the-difference-between-2d-and-3d-animation/

Wiley, N. 2019. The History of Comic Book Printing Dot By Dot. Printivity. Published on 8.8.2019. Read on 29.11.2022. https://www.printivity.com/insights/2019/08/08/the-history-of-comic-book-printing-dot-by-dot/

Wu, R. 2019. The Importance of Capturing and Controlling Specularity. Premium Beat. Published on 18.12.2019. Read on 12.9.2022. https://www.premiumbeat.com/blog/specularity-explained/

Xenoblade Chronicles 3. 2022. Nintendo.

Yuri on Ice. 2015. Director: Sayo Yamamoto. Production: MAPPA. Original release: 6.10.2016 - 21.12.2016.

Zucconi, A. 2015. Physically Based Rendering and lighting models in Unity3D. Published on 24.6.2015. Read on 12.9.2022. https://www.alanzucconi.com/2015/06/24/physically-based-rendering/#more-1964