Tampere University of Applied Sciences

Creating Realistic Clothing and Armor for 3D Game Characters

Riina Koivisto

BACHELOR'S THESIS
December 2025

Degree Programme in Business Information Systems
Games Production

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Games Production

KOIVISTO, RIINA:
Creating Realistic Clothing and Armor for 3D Game Characters

Bachelor's thesis 58 pages, appendices 1 page
December 2025

This thesis combines theoretical knowledge with a practical production workflow in order to provide a clear and applicable guide on creating realistic 3D character clothing. The aim of this thesis was to examine and explain the techniques used in the game industry for creating realistic and game-ready 3D character clothing and to explore a potential workflow for creating the clothing assets. The goal was to apply the concepts discussed in the theory chapters into a practical project where a game-ready medieval fantasy armor set is created for a 3D character.

The thesis is divided into a theoretical section where industry standard workflows and methods are discussed, and a practical section where a 3D medieval fantasy armor set is created. The theoretical portion examines the typical stages of 3D clothing creation, including project preparation and concept gathering, modelling methods and pipelines, topology, UV mapping, texturing and deformations through rigging and simulation. In the practical section these methods are applied to create a medieval inspired armor set. The armor set is created from start to finish using Blender, Substance 3D Painter and Unity, incorporating different modelling techniques, realistic PBR materials and real-time cloth simulations.

The outcome of this thesis is a broad and practical theory section on the key principles of creating realistic 3D character clothing, along with an example production pipeline that resulted in a game-ready and visually realistic clothing and armor set. This thesis may be useful for indie developers, hobbyists and game and art students, and supports beginner to intermediate 3D artists in creating clothing assets.

Key words: 3d, clothing, armor, game character, workflow

USE OF AI IN THESIS

I have used AI tools in my thesis:

☐ No

☒ Yes

The AI tools used in my thesis and the purpose of their use has been described below:

Names of the tools and versions: ChatGPT (OpenAI GPT-5.1 model)

Purpose of the use: AI was used to correct spelling and grammar mistakes as well as to suggest alternative wording when combining text from different sources. The AI generated text was never copied straight to the thesis text but used as a guide to give alteration ideas.

Parts in which AI was used: AI was used in the abstract, introduction, theory of game clothing and conclusions chapters.

I am aware that I am totally responsible for the entire content of the thesis, including the parts generated by AI, and accept the responsibility for any violations of the ethical standards of publications.

CONTENTS

1 INTRODUCTION

The creation of 3D clothes is an important part of game development that involves various techniques, tools and workflows. Clothing can enhance gameplay, be an integral part of a character's story or influence and even create the atmosphere. This is why it's important to create high quality and game-ready clothing that enhance the gameplay.

This thesis combines essential theoretical knowledge with a practical production workflow in order to provide a clear and applicable guide on creating realistic 3D character clothing. The aim of this thesis is to examine and explain the techniques used in the game industry for creating realistic 3D character clothing and to explore a potential workflow for creating realistic clothing assets. The purpose of this thesis is to apply the concepts discussed in the theory chapters into a practical project where a game-ready medieval fantasy armor set is created for a 3D character. The thesis is divided into a theory portion where industry standard theory, workflow and methods are discussed as well as a project portion where a 3D medieval fantasy armor set is created utilizing the theory discussed while also sharing practical methods used in the project.

The theory portion examines the typical stages of 3D clothing creation, including project preparation and concept art gathering, modelling methods and pipeline, topology requirements, UV mapping, texturing methods and PBR texturing as well as deformations using both rigging and simulations. For the practical project portion, a medieval fantasy armor set is created from start to finish following the industry standard pipeline and methods that are discussed in the theory portion. Additionally, practical advice is shared throughout the project stages by explaining all the methods and tools that were used. A 3D medieval inspired armor set is created incorporating different modelling techniques, real time cloth simulations and realistic PBR materials.

The project portion of the thesis is mainly created with Blender 4.0, Substance 3D Painter version 11.0.2 and Unity 6. Blender is used for modelling, preparing for texturing and rigging, while Substance Painter is used for creating realistic

PBR textures and Unity 6 for the final rendering and cloth simulations. The project does not go into character clothing design in detail or cover creating a rig or animations, though all the design choices are explained and justified, skinning and weight painting the armor set to the rig is discussed, and animations are imported from Mixamo to test the model in Unity.

This thesis may be useful for indie developers, hobbyists and students in the game and art fields as well as for others interested in the 3D clothing creation process. The aim is to combine essential theory and practical methods into a single document that supports the creation of clothing assets for beginner to intermediate 3D artists.

2 THEORY OF GAME CLOTHING

2.1 Planning and Preparation

The creation of 3D clothes is an important part of game development that involves various techniques, tools and workflows. It's important to take time to plan the project carefully to smooth the workflow and avoid reworking things later in the project. Before modelling anything, it's important to understand and set requirements for the asset. Questions such as how the mesh is going to be deformed in the game, is the asset unique or modular, what are the hardware constraints, and many more need to be answered before starting asset creation. Depending on the requirements, the asset details and creation methods could be quite different even though the pipeline stays the same. Knowing as much as possible about the asset's needs before starting the creation process helps in creating optimized and high quality results.

The technical needs of the project will define the budget for the asset. Budget in this context means how much memory or processing power the asset can take up in the final game and affects many aspects of the model such as the final polygon count, the size of the textures and the number of texture sets that can be used. These aspects are determined by many factors such as the importance of the asset in-game, the size of the game and which engines or platforms the game will be on, for example. (King 2023, 7.5.)

For clothing, it's important to know how it's going to be deformed in the game. It's good to know what kind of movements the character is doing in the game and whether the game uses cloth simulations or not. These two things will help determine where to put the details of the clothing folds and how to create the actual asset. The character movements determine where the clothing folds might naturally form and knowing this helps in creating a more realistic asset. For example, the asset prioritizes different things if the character is presented from the first person view versus the third. Likewise, a character holding a gun with their hands up most of the game versus one walking around with their hands down might require different things from their clothing. (Karimfazli & Tokarev

2019.) Different modelling techniques can also be used depending on how the clothing mesh is deformed in the game. Creating clothes that use simulations versus ones that use animations might require different meshes to get the best result. For example, when simulating cloth, the mesh deforms best if there's no thickness to it and the mesh consists of triangles (Thunder Cloud Studio 2024c). And on the other hand, it's typical to use quads when animating organic assets like clothing (Steppig 2022).

Keeping in mind the requirements of the project and the character, reference gathering can begin. When creating realistic assets, the focus is to accurately represent lifelike features from anatomy to materials. That's why it's especially important to spend time researching and gathering references of real-life instances for projects like these. For clothing this would mean examples of the materials needed, the structure of the clothing such as seam placements, how gravity affects the cloth and other important aspects that affect how the clothing looks and behaves. It's important to get an idea of the asset from every angle to make the modelling and texturing process easier. (Thunder Cloud Studio 2024a.) Larger projects typically have concept artists that provide project specific concept art but gathering additional references of the needed materials and details can enhance the results. References can be found nearly anywhere from different games, social media and dedicated tools or databases. Once collected, they are gathered and organized into a single reference board to support the design and production process.

Once the requirements are clear, general concept idea is ready and references are gathered, the artist can move on to planning the project in more technical detail. According to Thunder Cloud Studio (2024a) there are three key aspects to assess the workload and prepare for production: identifying distinct and symmetrical objects for modelling, determining the number of UV sets required and finding reference images. Identifying which objects can be mirrored and which are unique streamlines the modelling process and avoids overlap in the workflow. The UV usage directly impacts the quality of the textures, so to ensure good optimization, the model and UV layout should be planned out from the start. This could be done simply by drawing over the concept art picture like in Picture 1. (Thunder Cloud Studio 2024a.)

PICTURE 1. Example of mesh and UV plan (Thunder Cloud Studio 2024a).

Once the asset requirements are clearly set and the concept art is gathered, it's time to move into the modelling phase. These plans and references are general guidelines that can be added to or altered later on, because changes throughout the project are expected. Even if changes occur during development, establishing the core requirements and workflow early on helps ensure a smoother process.

2.2 Modelling

Modelling is the stage where 3D assets start to take shape in 3D forms. While the general workflow for clothing follows the same principles as any other 3D asset, clothing assets present additional challenges like realistic deformation and the balance between detail and performance. In this chapter, different modelling methods and the typical modelling pipeline are explained.

Polygonal modelling, referring to the meshes being composed of polygons, is the most common form of 3D modelling for games (Shahbazi 2025). A polygon is a plane formed by at least three vertices and edges that create a closed shape. Polygons in games are typically triangular (tris) or quadrilateral (quads). A vertex is the point where two or more edges meet and join. Moving the vertices around

in the 3D space is the most common method for shaping the mesh. An edge is a two-dimensional line that has vertices at either end. (Kevuru Games 2023.) And lastly, a face is the flat 2D shape that is created when a polygon is formed (Adobe n.d.b).

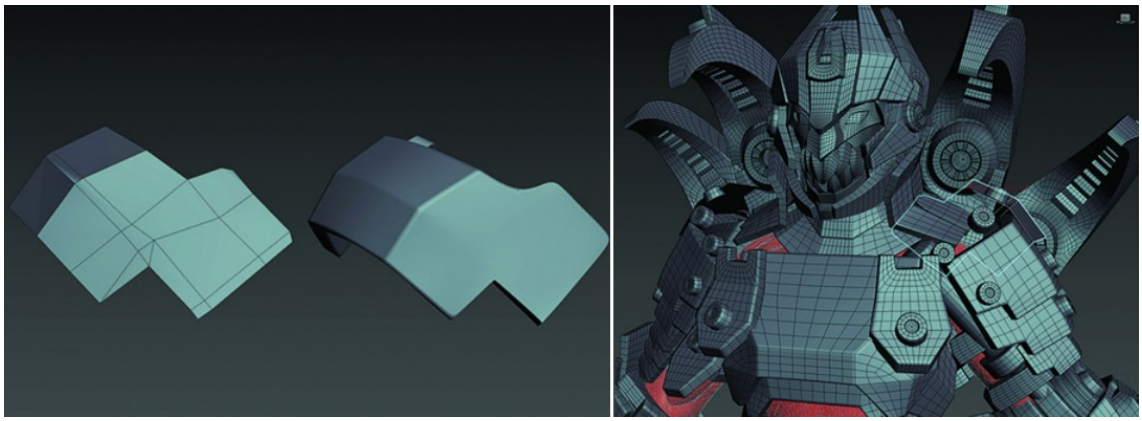
Polygon modelling is often preferred in the game asset pipeline for the precise control it gives artists over topology and detail. Developers must be mindful of polygon counts and optimization in relation to different hardware constraints, since many assets need to be rendered in real-time at once. (Adobe n.d.b.) The topic of optimization through topology will be covered in more detail in chapter 2.3.

Polygonal modelling is a typical starting point for any 3D modelling process with simple shapes like spheres or cubes as the base. When continuing, there are multiple different modelling techniques that are suitable for different circumstances. Even when modelling a single asset, like a piece of clothing, different methods can be used and combined depending on the needs of the asset. The next sections will focus on some of the most common modelling methods that are used in clothing asset creation.

Hard surface modelling is a 3D modelling technique that is used to create rigid, non-deforming objects. It’s typically used to create sharp edges, clean geometry and detailed structures. (Shahbazi 2025.) This method works well when creating smaller details, accessories or more rigid clothing like armor (King 2023, 7.3).

Manipulating the mesh precisely and utilizing different modifiers, booleans and a non-destructive workflow can be an effective modelling method for certain assets (Shahbazi 2025). Starting with a simple base shape and utilizing different modifiers that affect the symmetry, polycount or other aspects of the model is a typical way to create hard surface assets. Saving time by using a non-destructive workflow that is easily manipulated and using techniques like kitbashing helps to ease the workflow. Non-destructive workflow refers to a modelling approach where changes can be easily edited or reverted without altering the underlying geometry. Kitbashing refers to creating new models by modifying and combining pre-existing mesh components. Picture 2 shows how hard surface modelling can

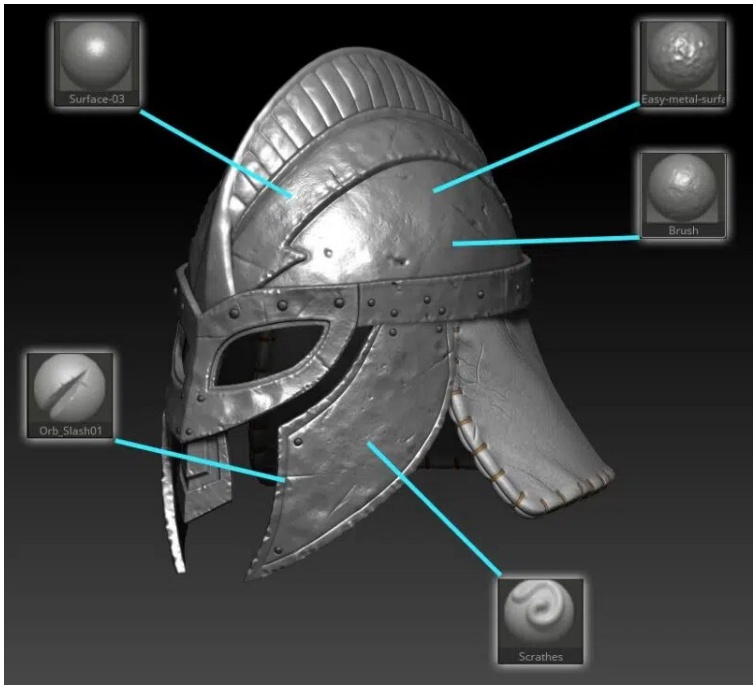
be utilized for creating armor pieces by using simple shapes that are further defined by mirror and subdivision modifiers. (King 2023, 7.3.) Some typically used modelling software includes Autodesk Maya, Blender, 3Ds Max, SketchUp and SolidWorks (Shahbazi 2025).



PICTURE 2. Example of hard surface armor pieces that use modifiers (King 2023, combined from chapter 7.3).

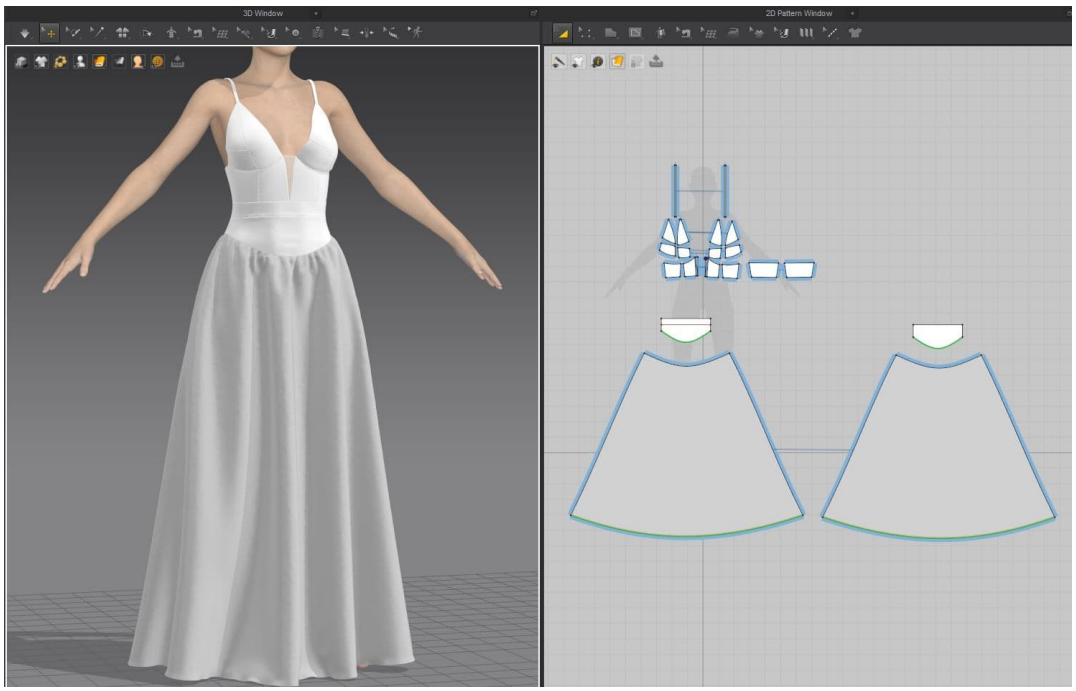
Another common technique used to create clothing is sculpting. Sculpting mimics the process of clay sculpting, allowing artists to push, pull and mold the mesh. Soft and detailed shapes can be created freely which makes it a good technique for modelling organic materials like cloth and creating details like wrinkles or surface textures. Sculpting requires a high polycount base mesh especially when making a highly detailed asset. Tools such as ZBrush, Mudbox, Sculptris, 3D-Coat and Blender can be used for 3D sculpting. (Shahbazi 2025.)

Sculpting is usually done with different kinds of brushes that are designed to help with different aspects of sculpting. Brushes can range from generic that form larger shapes to very specific for creating details like creases or different material textures, for example. Picture 3 shows how multiple different brushes can be utilized for detail texturing in a single asset. Specific designs can be made by using projected alphas on the brushes or utilizing masks while sculpting (King 2023, 7.3). Masks can be utilized in both the larger blockout stage of clothing and creating the finer details. For example, the shape of pants can be masked out of the characters legs and sculpted outward to take shape, or details like pockets or buttons can be masked and sculpted out of the pants mesh. Sculpting encompasses many versatile methods for creating organic assets.



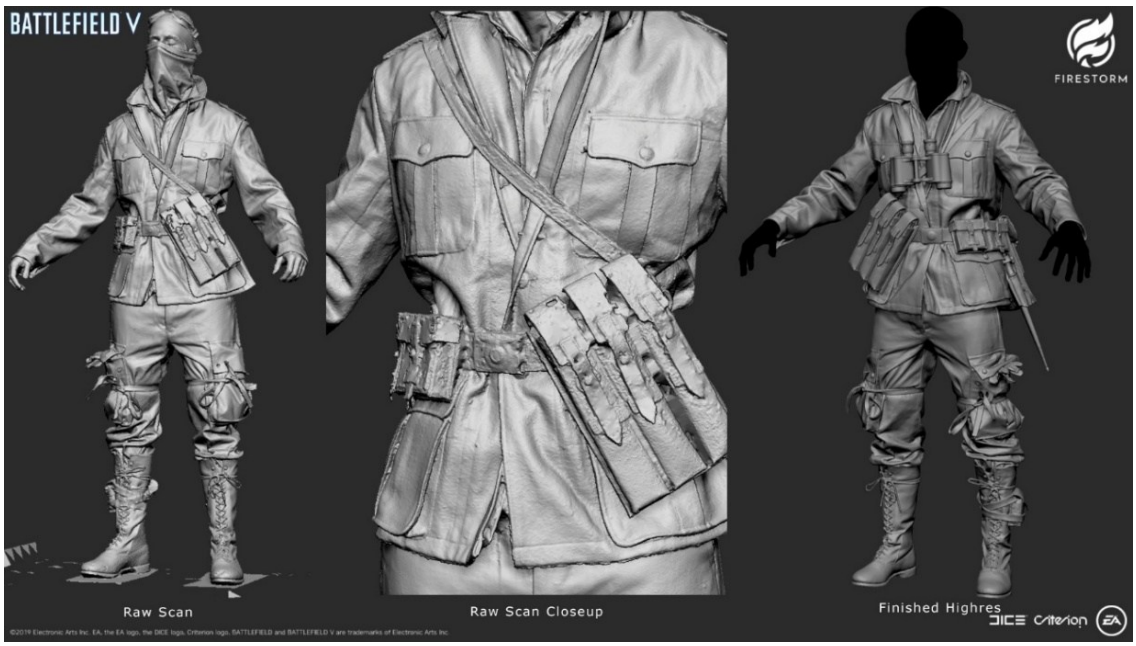
PICTURE 3. Viking helmet that utilizes different brushes for details in Zbrush (Thunder Cloud Studio 2024a).

Simulation is a very popular and often used clothing creation method. Pattern-based digital tailoring combines real-life garment making and computer simulations, where clothing can be created by using patterns and altered while being simulated on a character. Using simulations is a popular method because it's a quick way to prototype assets that can be easily altered and manipulated. The simulations provide realistic movement and physics to the clothing, creating natural looking clothes with relative ease. Tools like Marvelous Designer and CLO 3D are the most widely used software for simulated garment creation. (Radivojevic 2024.) An example of simulated fabric and the pattern making up its parts can be seen in Picture 4, showcasing how simulation with simple patterns can create a detailed looking mesh with realistic creasing. Simulation is a great way to get the base mesh of an asset quickly blocked out but often requires further refinement and cleanup.



PICTURE 4. Example of pattern-based tailoring in Marvelous Designer (Chernyshenko 2022).

Body scanning is also a method used by some studios, especially when creating specific clothes from the real world. Body scanning technology captures a highly detailed mesh of clothing or accessories by scanning them from a real life model or mannequin. (Radivojevic 2024.) Scanning creates realistic weight and details to the clothing since it's copied from a real-life counterpart and is an efficient method for capturing the look of features like multiple layers. The scan is treated as a high-poly base, so after scanning, the clothing is separated into individual objects and cleaned up and refined as shown in Picture 5. Lower layers and the insides of the clothing are also modelled to support deformation when animating or simulating. Additional details or accessories can be created or refined further to get the desired look. Scanning is typically limited to the studio's in-house capabilities. (Karimfazli & Tokarev 2019).



PICTURE 5. Example of a raw 3D scan and cleaned up version (Karimfazli & Tokarev 2019).

Even though these methods seem quite different from each other, they are typically used together and artists go back and forth between different methods during modelling. For example, a workflow for a shirt could start with it being simulated to create the base, followed by sculpting to refine details and end with adding hard surface details like buttons. Before starting to model, it's good to assess which modelling methods are best for building the asset. An iterative approach can combine the strengths of each technique and ensure an efficient workflow and a realistic result.

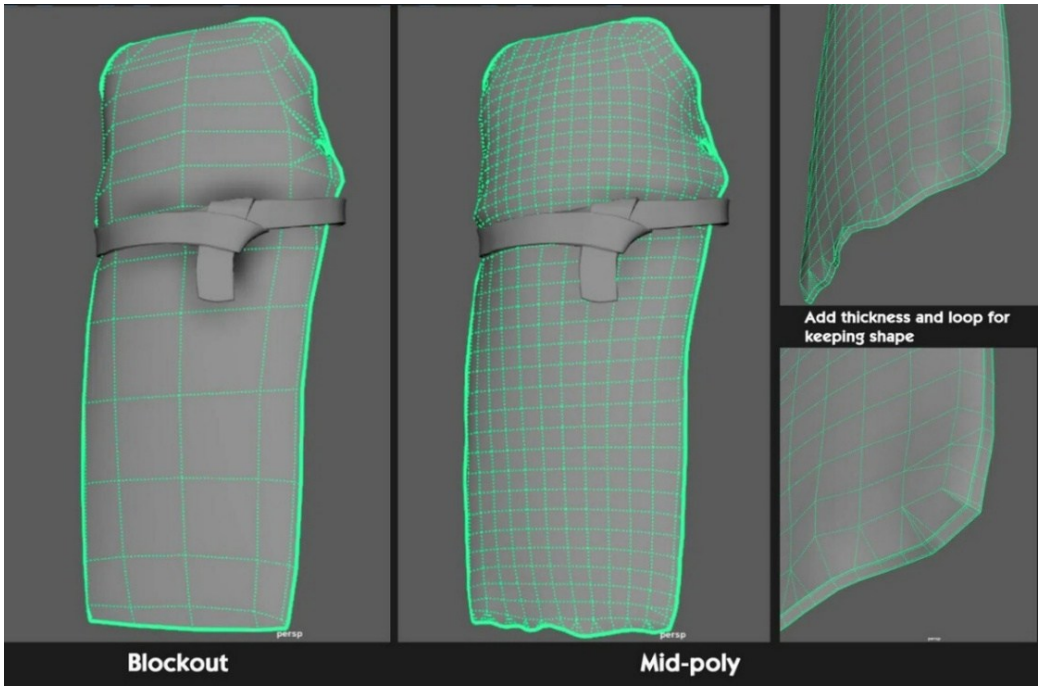
The following sections outline a typical asset creation pipeline. The pipeline typically consists of three phases: a blockout phase where the primary shape is created, a high-poly modelling phase where details are added and a low-poly or retopology phase where an optimized model is created.

The first stage of modelling is blocking out the main shapes. The purpose of a blockout is to create the most important aspects of the model first, and focus on the primary forms, proportions and silhouette of the model. It's important to create a strong base with the correct scale before starting on any details. (King 2023, 7.2.) Blockouts are typically made from primitive geometric shapes like cubes, spheres or cylinders and then shaped into more desired forms. Different assets benefit from different blocking methods depending on how they're modelled. For

example, when blocking out for sculpting, different high-poly shapes can be combined and used to create the base, and on the contrary, when blocking out for hard surface models, modelling is started with flat planes or simple shapes and further defined with modifiers or specifically placed details.

For loose clothing like shirts, dresses or pants, simulation is a quick way to blockout the largest folds and wrinkles. It's generally faster and easier to simulate the cloth than to manually sculpt them. Simulation also typically gives more realistic and easily modifiable results. At this stage assets like clothing, belts and armor should have no thickness so they remain easy to adjust. Creating them as flat planes with low polycounts makes them easier to manipulate and alter during the blockout stage. (Thunder Cloud Studio 2024a.)

After finalizing the main shapes and proportions it's time to refine each part. The generally blocked out shapes are maintained and built upon to create a good foundation for the high poly modelling phase. Once the overall shape is as desired, the assets can be given thickness and well-placed edge loops, like shown in Picture 6, to make them ready for high-poly modelling. At this stage it is good to evenly distribute topology density to ensure that increasing subdivision levels works well. (Thunder Cloud Studio 2024a.)



PICTURE 6. Clothing asset apron from flat blockout to thicker and denser mid-poly asset (Thunder Cloud Studio 2024a).

High-poly model refers to a mesh that has an extremely dense polycount and that is typically very detailed. The point of this stage is to add as much detail as possible without worrying about the final polycount. (Adobe n.d.b.) These details will be baked into the low-poly model to combine the highly detailed mesh with an optimized mesh.

The level of detail that's added during this stage depends on the project and asset requirements as well as personal workflow preferences. In Thunder Cloud Studio's workflow (2024a) smaller clothing folds and even microdetails such as the fabric surface texture are added to clothing during this stage. If the concept idea is clear and the asset requires a specific surface texture, it can be achieved by sculpting or by using different procedural noise textures that tile over an asset. Picture 7 shows how detailed the finished high-poly model can look when this method is used. On the contrary, Karimfazli and Tokarev (2019) state that depending on the model, the smallest details are added in the texturing phase. This can be done for asset modularity purposes or to more easily change them later on (Karimfazli & Tokarev 2019). For example, different fabric details like stitching or the pattern of the fabric can be added during the texturing phase so they're easier to change later in the production, or so that multiple different assets can be created from the same base mesh.

PICTURE 7. High-poly sculpt of a viking with microdetails on the clothing (Thunder Cloud Studio 2024a).

The deformation of the mesh also impacts the level of detail that looks good on an asset. Static folds in areas that are going to be simulated or deformed significantly can look unnatural, so it’s good practice to reduce them or remove them entirely. Any folds that don’t align with the cloth movement should be cleaned up. (Karimfazli & Tokarev 2019.)

After the high-poly model is complete and all the desired details are added, the process moves on to creating a low-poly version. Low-poly model refers to a mesh that has a small number of polygons and that has been recreated or optimised to lower its polycount (Adobe n.d.b). The high-poly mesh contains all the fine details but is too dense to be used for real time rendering or animation so it must be optimized with a low-poly version. In this stage the focus is to create a model with a lower, optimal number of polygons while keeping the silhouette clear and topology suited for animation. Creating the low-poly model is a highly technical stage that’s tightly intertwined with topology and optimization, and affects later stages such as texturing and animation (Thunder Cloud Studios 2024a). The technical details of topology are discussed in chapter 2.3.

The key aspect of a low-poly model is the general shape and silhouette of the model. It’s important to ensure that each model still has a distinct and recognizable outline even from far away and that it matches up with the high-poly model as closely as possible. Polygon density should be allotted according to the model needs, meaning more detailed places such as the hands and face have more polygons, and less detailed places have fewer polygons. To achieve this, investing polygons into strong shapes and smoothly curving edges ensures that the form of the low-poly model matches up with the high-poly model. (Kevuru Games 2023; Thunder Cloud Studio 2024a.)

A low-poly model can be created from scratch by using retopology techniques or by optimizing the high-poly or blockout mesh. What is considered a low-poly asset depends on the project requirements such as the target engine. Regardless of these specifications, an optimized low-poly model is essential for the quality and performance of the game asset.

2.3 Topology

Topology is an important aspect to keep in mind when developing assets that are rendered in real time, like they are in games. Topology refers to how the vertices, edges and faces are structured to create the mesh. The topology of an asset determines how efficiently a model performs, how it deforms during animations and the quality of its visual appearance. (Steppig 2022.) This chapter covers typical topology rules that are followed when creating game-ready assets.

Topology is often discussed through retopology, which is a crucial step in the game asset pipeline. Retopology involves creating a low-poly mesh following the shape of the high-poly mesh, while optimizing its topology for texturing, animation and rendering (King 2023, 7.5). Like discussed briefly in the previous chapter, the highly detailed mesh is typically too heavy to be used in engine so it must be optimized with retopology. The details of the high-poly mesh are baked into texture maps and used in the texturing phase with the low-poly mesh. The texturing process is discussed further in chapter 2.4.

Clean or good topology refers to topology that minimizes issues for later stages such as texturing or rigging, and supports deformation in a way that is believable for the asset (Steppig 2022). There are some industry standard practices which are followed when creating any asset to ensure it has good topology.

It is typical to only use tris and quads in the mesh to ensure that shading looks correct, the mesh is easy to manipulate while modelling, deforms well and modifiers work as intended. According to Steppig (2022), quads are the most optimal for topology because they work predictably with different types of modifiers and enable loop cuts to flow through them. In contrast, both King (2023) and Thunder Cloud Studio (2024b) state that tris can also be used while keeping in mind good topology flow, since the final mesh is triangulated in-engine. N-gons on the other hand, referring to faces that have more than 4 verts, should be avoided because of the shading and subdivision issues they typically cause (King 2023, 7.5; Thunder Cloud Studio 2024b).

Good topology is not just about ensuring deformations work correctly but also about proper shading. Shading defines how the model appears based on the direction of individual face normals and how they interact with the light. As a result, different topology decisions affect how the shading will look. Normals indicate the direction that each face is pointed towards in space. With flat shading, each face displays its normals individually, creating a flat and accurate look of the mesh. Smooth shading, on the other hand, calculates the average between connected edges, creating softer transitions between faces. (Steppig 2022, 7.) Vertices along a soft edge share similar normals and create smoother transitions compared to further away or sharper edges that create distinct and harsh transitions. Managing these transitions with good topology and creating good shading is an essential part of the final look of the asset. Incorrect geometry, poorly defined edges or missing UV seams along hard edges can result in shading artifacts that affect texture baking later on in the process. (Thunder Cloud Studio 2024b.)

It’s essential to avoid non-manifold geometry as it can cause complications with subdivisions, shading and texturing (Thunder Cloud Studio 2024b). Non-manifold geometry occurs when a mesh can’t be flattened into a continuous surface with consistent normal directions and thus cannot exist in the real world. The most common non-manifold mesh problems are illustrated in Figure 1, including T-type intersections, open geometry, internal faces, opposite normals and surfaces connected by only one vertex. (CGTyphoon n.d.) Another common issue besides the ones depicted in the image is having a concave face, which happens when the deformation of a quad is too great and the face folds through the middle (Thunder Cloud Studio 2024b). Non-manifold issues can cause problems with mesh subdivisions, textures and baking because they break the surface normals and thus affect the shading of the mesh.

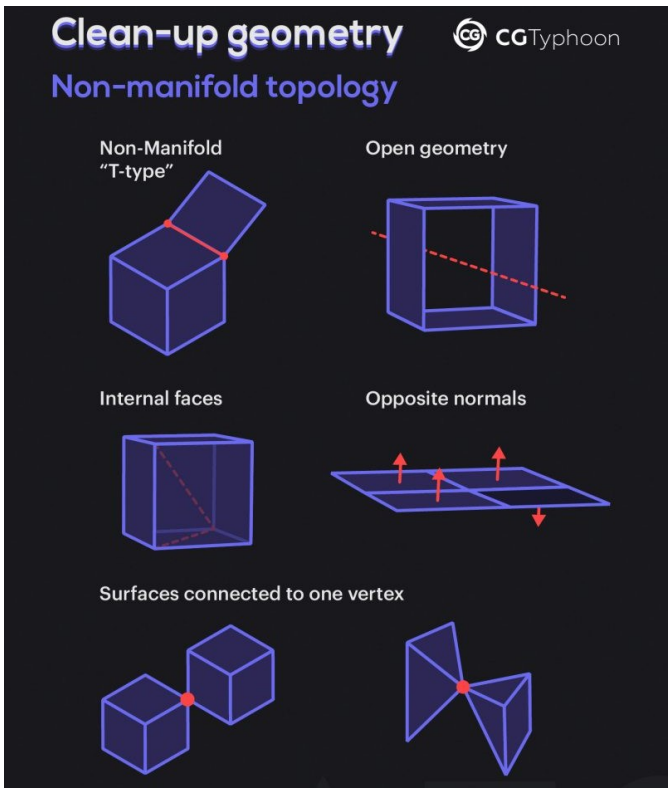
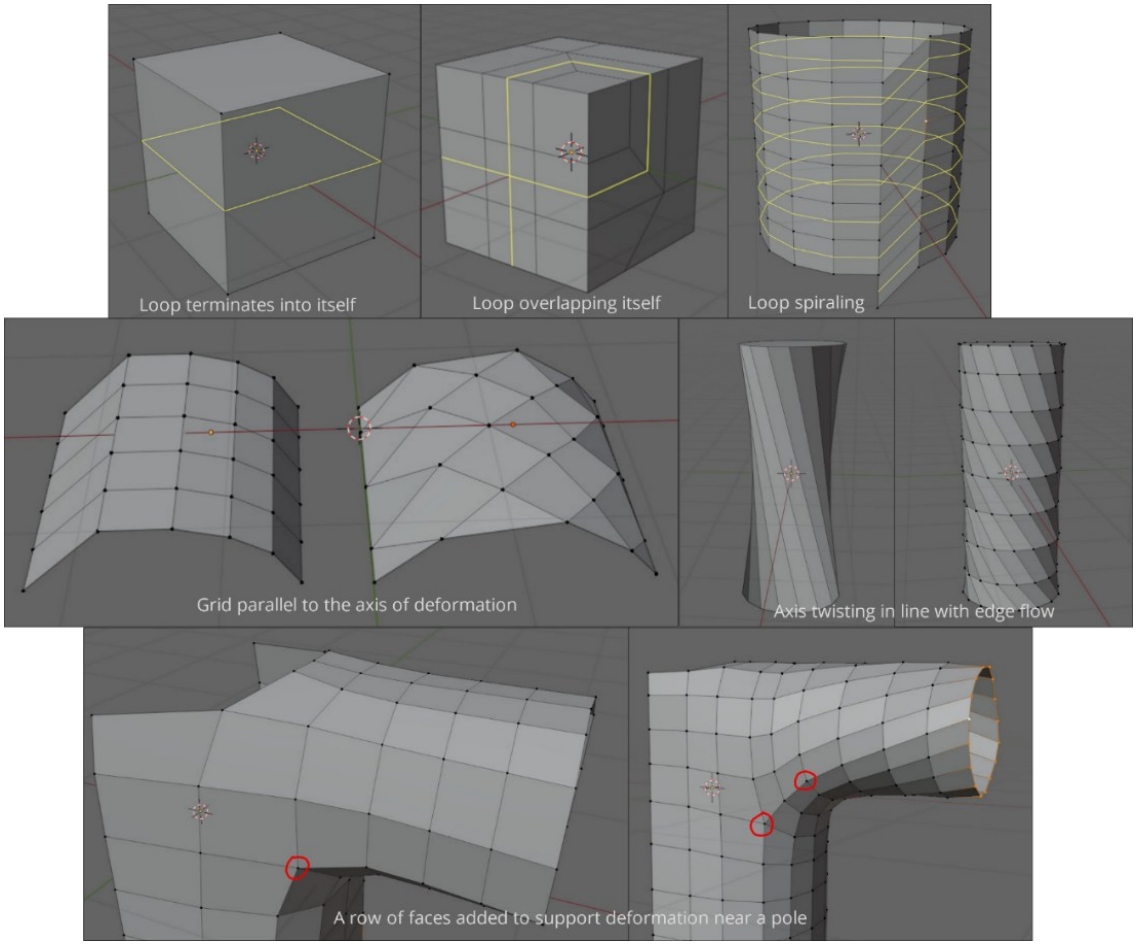


FIGURE 1. Examples of non-manifold geometry (CGTyphoon n.d.).

It's good practice to keep the mesh density consistent throughout the entire model and the faces as close to squares as possible. This eases laying out the UVs and prevents the textures from stretching or distorting in unwanted ways during texturing. Additionally, the density of the mesh also affects it's shading. This can sometimes make sharp edges appear softer even with edge loops placed near the edge. Keeping the mesh density consistent helps prevent these kinds of distortions. (Steppig 2022, 7; Thunder Cloud Studio 2024b.)

Steppig (2022) proposes that there are a series of geometrical and deformation rules that when followed ensure a good topology for a mesh. His geometrical rules affect the shape of the mesh and state that all loops need to terminate into themselves or into the void and none of the loops can overlap themselves or spiral around the mesh. His deformation rules state that the edges of a grid need to be parallel to the axis of deformation, the axis of twisting should be in line with the edge flow, and one row of faces should be put between a pole and the specific area of deformation to support good deformations. A pole is a vertex where more than four edges meet. (Steppig 2022, 2-3.) These rules are depicted in Picture 8

in the order they’re mentioned. Together these rules give a simple list of principles to follow when creating good topology.

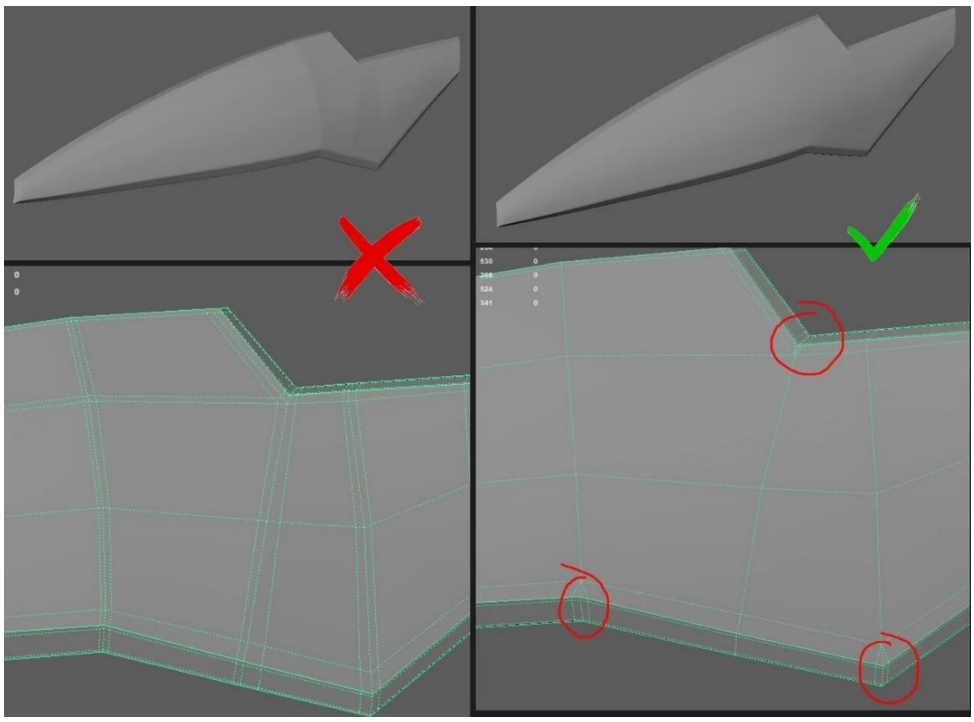


PICTURE 8. Geometrical and deformation rules for mesh topology (Steppig 2022, combined from chapters 2-3).

These rules can be further built upon when taking edge flow into consideration. King (2023) emphasizes the importance of clean and logical edge flow throughout the model to support realistic deformation during animation. Edge flow refers to how topology follows the natural forms and directions of the model. For example, with a shirt the edge flow should follow the curves of the body both vertically and horizontally to move naturally with it. Clothing typically moves with the body, so similar topology rules apply to both. In organic surfaces like clothing, edge loops are typically aligned with the underlying muscular and skeletal features of the body to ensure natural flow and deformations during motion. Additionally, more edge loops are added in areas that move, bend or twist so the mesh doesn’t become too low resolution or stretched out in extreme poses. (King 2023, 7.5.)

In addition to following the anatomical or organic forms, edge loops can also be used to preserve the overall silhouette of an object. Especially for hard surface objects, having the edges gradually flow inwards toward the centre of the model maintains the silhouette and prevents distortion. The same logic of maintaining continuous loops can be applied to clothing assets as well. Ensuring the thickness of the garment is a continuous loop, whether it's an organic or a hard surface object, simplifies modelling adjustments and improves UV layouts. (Thunder Cloud Studio 2024b.)

Sometimes a mesh can benefit from using tris instead of quads when managing edge loops. For models with sharp angles or surfaces that curve consistently, typical edge loop placement may waste vertices and create uneven mesh density. Thunder Cloud Studio (2024b) suggests that adding tris in these places will still shape the mesh as intended while reducing unnecessary topology, like shown in Picture 9. This can be quite useful when modelling clothing since they often utilize curved surfaces and corners.



PICTURE 9. Example of using tris to manage round corners (Thunder Cloud Studio 2024b).

These topology guidelines can be altered to fit the asset needs, for example when simulating cloth. Steppig (2022) suggests that his topology rules apply to cloth

simulations as well, while also highlighting the importance of evenly sized quads to provide a consistent simulation and utilization of modifiers. In contrast, Thunder Cloud Studio (2024c) states that when simulating clothing the result is more accurate when the mesh consists of tris instead of quads, since the mesh can bend and stretch more evenly while under calculations. In practice, the choice between quads and tris depends on the intended workflow and engine requirements. Modern tools and game engines typically convert the mesh to tris automatically and many modelling softwares include tools for converting between tris and quads when needed (Thunder Cloud Studio 2024c). Therefore, following the standard quad topology remains a reliable approach during modelling, though adjustments can be made in the pipeline if necessary.

The quality of the cloth simulations is determined by the number of vertices the mesh has but adding too many can greatly slow the calculations (Steppig 2022, 3). The number of verts also affect the overall look and feel of the fabric when simulated. For example, a more detailed mesh creates the look of lighter fabric because of the fine details it can create. Cloth thickness can be tricky to calculate accurately while simulating because it can easily intersect with the inner and outer surfaces (Thunder Cloud Studio 2024c). Due to this, it’s best to add thickness after the simulations if needed.

As mentioned before, it’s good to consider how the mesh is going to be deformed in the game to maintain predictable deformation while optimizing its topology. The technical needs of the project will define the final polygon budget for the asset. Many aspects such as the importance of the asset, the size of the game and the game engine or platform used in the rendering will determine how detailed the base mesh can be. Balancing between a detailed mesh and optimization is crucial when creating clean topology.

2.4 Texturing

The texturing phase encompasses many different aspects and stages. During this stage the mesh is given properties of materials like color, fine details and smoothness. This chapter covers the basic texturing workflow from UV mapping,

baking high-poly details onto low-poly models, to different texturing methods as well as a more detailed explanation of PBR texturing.

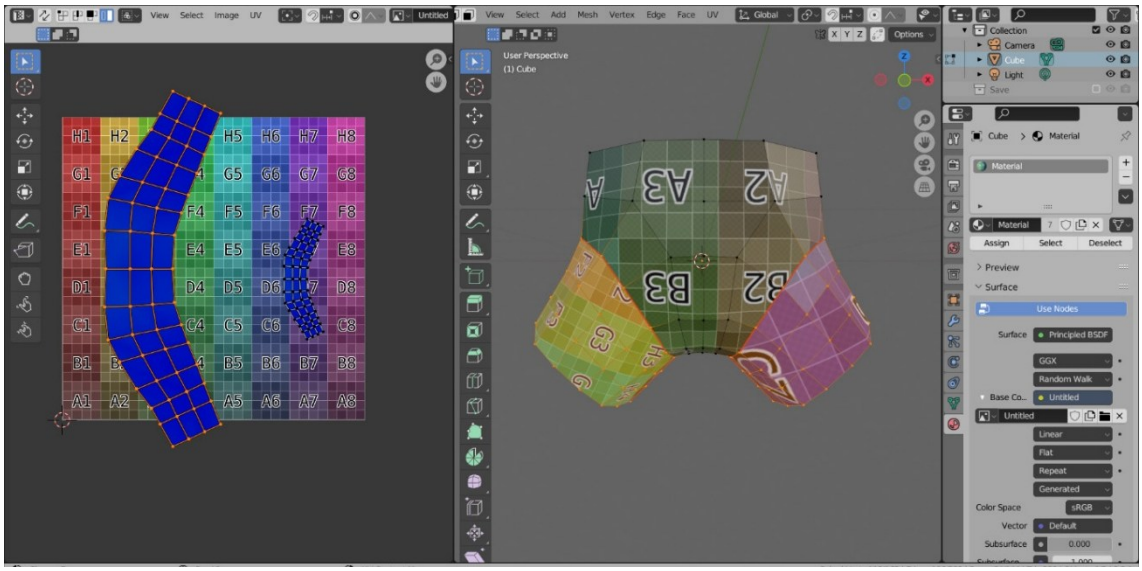
The first stage of texturing is creating the UVs of the optimized, low-poly model. UV mapping is the process of flattening out the surface of the 3D model to 2D forms so textures can be projected onto the model. For game assets, this is done by placing seams and unfolding the model strategically so the UV map is as optimal as possible and doesn't cause distortions on the textures. To keep in mind optimization requirements, it's important to balance high-quality textures with efficient resource usage. Some good UV practices include strategic seam placement, utilizing UV space efficiently and maintaining consistent texel density. (Radivojevic 2023.)

Placing seams on the model strategically helps to minimize texture errors and can help to create realistic and optimized textures. For clothing, seam placement can be copied from real life clothing to create realistic texture details. Similarly to other assets, seams can also be hidden to places where they're less visible to create even textures across larger areas. (Radivojevic 2023.)

Utilizing UV space efficiently means using all the available space of the texture by packing the UV islands close together and using methods like UV stacking when it's possible. It's important to leave some padding space between the islands to take into account texture blurring in engine that occurs during mipmapping, the process in which texture resolution is reduced as objects move farther away from the camera. UV stacking refers to stacking similar UVs on top of each other to save space. For example, mirrored objects like shoes or armor pieces that use similar textures can be stacked to share the same texture. (King 2023, 7.6.) Another important aspect when creating UVs is to rotate the maps that share materials to face the same direction in order to simplify the texturing process. This is especially important when working with patterned or tileable materials to make sure the pattern continues in the correct way across the asset.

Texel density refers to the size of faces in relation to how much space they take up on the texture. Having consistent texel density means that different parts of the asset as well as different assets in the game should have the same quality

and sharpness across the project. (Steppig 2022, 4; Radivojevic 2023.) UVs should be checked using a grid or checker pattern to more easily point out issues with resolution, unwrapping and stretching (King 2023, 7.6). This is demonstrated in Picture 10 where different texel densities are shown on the same asset.

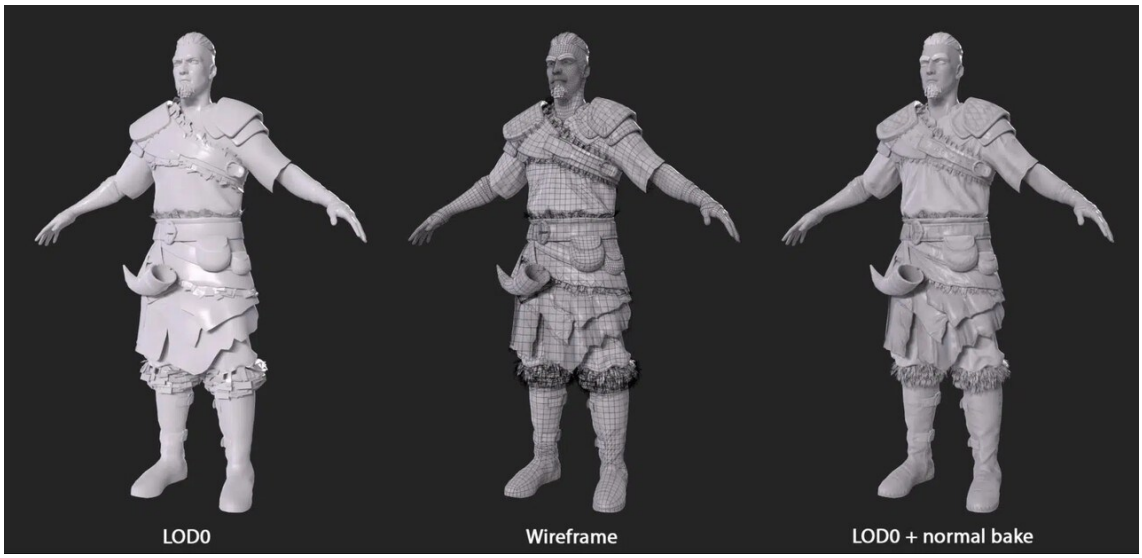


PICTURE 10. Different texel densities on the same asset (Steppig 2022, 4).

Assets can be organized to UV maps based on their material needs to optimize the texturing process. For example, metallic assets like armor and clothing details can have their own separate material while organic materials like fabrics have their own. If an asset has complex properties like translucency in some places, it's optimal to group all the meshes that need transparency on the same material so a single texture can be used to define all of them. (Adobe 2025; King 2023, 7.6.) Since having separate texture sets for every individual material is rarely optimal, materials are often grouped into a shared texture set. In these cases, it's good practice to keep UV islands that are near each other in 3D space or share similar materials positioned close together in the UV layout (Adobe 2025).

When the UV map is completed the texturing process can begin by baking the details of the high-poly mesh. Baking is the stage where the very detailed high-poly mesh is projected onto a series of texture maps that can be applied to the low-poly model. These textures utilize multiple maps, especially the normal map, to copy the details of the high-poly model while working with the optimized low-poly mesh. (King 2023, 7.7.) Picture 11 showcases how detailed the low-poly

model can look when the high-poly details are baked and applied to it. Baking can be done in a modelling software or in a texturing software such as Substance 3D Painter or Marmoset Toolbag. Baking is an important step to creating realistic models while keeping optimization in mind.



PICTURE 11. A low-poly model, its wireframe and the model with high-poly normal map details (Thunder Cloud Studio 2024a).

With the UV maps and baking completed texturing can begin. There are three major texturing techniques: hand-painted textures, scanning real life textures and procedural generation. Often, these techniques are combined and used together. (Adobe n.d.a.)

Hand-painted textures are created by hand, typically by using different digital painting applications or inside the modelling software. This method gives total control to the artist and the result is often dependent on the artist's skills. Hand-painted textures are mainly used for creating more stylized assets because of the time and skill it would take to achieve realistic results. (Adobe n.d.a.)

Scanning real life materials and turning them into texture images can be an efficient way to texture assets. This method is great for copying existing materials from real life objects. This scanning could be done with simply a camera, or with high-resolution machines that measure the surface. (Adobe n.d.a.) The scan gives a good base for the texture but is often further developed by an artist to achieve the wanted result (Karimfazli & Tokarev 2019).

Procedural texturing relies on algorithms to create varied patterns and textures. This method can be very useful for creating detailed and realistic assets quite quickly. Textures can be applied by following different requirements like geometry or rotation, for example by scattering a pattern only on the plains of the model or on harsh edges, making it great for creating certain types of textures. Procedural textures also have the advantage of easy iteration because they can be changed by numbers or code. Procedural texture generation can be very useful for texturing large amount of assets because it can create many unique textures with relative ease. Procedural textures can be created with different node-based systems or shaders in modelling and rendering engines or in softwares such as Substance 3D Designer or Substance 3D Painter that utilizes procedural textures when creating patterns. (Adobe n.d.a; Radivojevic 2024.)

Additionally, different kinds of shaders can be used to enhance the textures or create even more immersion. Multi-layered shading systems can be utilized to create realistic material changes under different environmental conditions. With this method different conditions such as dirt or rain can affect how the material looks and change it dynamically. This method enhances the realism when conditions change but can be performance heavy due to the complexity of the shaders. Some studios even create dynamic wrinkle maps that respond to character movements and poses. These maps are generated through simulations that are then baked into normal maps to create even more detailed models. (Radivojevic 2024.)

Texture types can be split into tileable textures and unique textures. A tileable texture can be tiled in every direction and the texture would continue seamlessly. Contrarily, a unique texture is specifically created for one model and doesn’t tile. (Adobe n.d.a.)

Physically Based Rendering (PBR) is widely used for material creation in game development due to its balance between visual realism and performance efficiency. PBR is a method of shading where physical behaviours of light with different materials are replicated, producing visually realistic results. PBR materials are used because it is a consistent, realistic and efficient way of

mimicking the way light behaves when interacting with real surfaces. PBR materials are based on physical laws, such as the law of energy by not reflecting more light than it receives, and the Fresnel effect that controls how materials reflect to light from different angles, to ensure that materials behave predictably under any lighting conditions. PBR materials enhance realism while streamlining workflows and allowing artists to produce assets that are visually coherent and standardized. These material lighting effects are created with the combination of algorithms, shaders and different texture maps. (Iontcheva 2025.) Table 1 lists several types of maps that are used when creating realistic PBR textures.

TABLE 1. Types of texture maps in PBR texturing (Iontcheva 2025).

Albedo	The base color of the material without any lighting or shadows.
Normal Map	Simulates surface details and irregularities without adding real geometry.
Roughness	Controls how shiny or matte the surface appears.
Metalness	Determines the metallic properties of a material, influencing its reflectivity.
Specular	Defines how strong and what color the highlights on the surface are.
Height	Adds depth information to make surfaces look raised or lowered and deforms the actual mesh.
Opacity	Controls how transparent the material is.
Ambient Occlusion	Adds shading to small creases and corners to create depth.
Refraction	Determines how light bends when it passes through the material.
Emissive	Makes parts of the material appear to glow or emit light.

For the PBR workflow the most used texture maps are albedo, metalness, roughness and normal map. These high-quality maps convey different surface details and properties so the asset can achieve realistic and detailed lighting results. These four maps alone can achieve great results but more maps can be added according to the asset and artistic needs of the project. (Iontcheva 2025.)

Additionally, more maps can be generated to support the texturing process, such as world space normal or ID maps. However, the use of these maps depends on the texturing software because they are primarily workflow aids rather than texture maps used in the final asset.

The texture sizes depend on the asset needs such as the size and importance of the asset as well as optimization and engine needs. Assets that are close to the screen or otherwise prominent can be more detailed and use larger textures compared to less important assets. Typically, character texture sizes range from 1024x1024 to 4096x4096 pixels depending on the asset needs. Realistic textures tend to require larger files to capture all the detail. Sometimes multiple textures can be combined into a single map for optimization. A method called PBR-ORM can be used, where the ORM stands for ambient occlusion, roughness and metallic maps that are combined into a single map (Thunder Cloud Studio 2024a). This method might not be supported in all engines, including Unity. Textures can initially be created with very high resolutions and be downscaled later on to fit the target resolution and optimization needs.

While texturing, the importance of good references comes into play. Like stated earlier in chapter 2.1, it’s good to have a wide range of material references from multiple angles to produce a realistic result. Keeping in mind how the materials look and behave both in real life and in the game scene will help to produce a more realistic outcome. While making the materials, it is best to continuously export and test the textures in the final game engine. This gives an idea on what the final textures are going to look like and helps to improve the textures to get the intended result. (King 2023, 7.9.)

2.5 Mesh Deformation

With the model and textures complete, the next stage in the production pipeline is rigging. This chapter covers rigging and skinning the clothing to the rig as well as different methods of deforming the character clothing to create movement.

Rigging is the process of creating a bone system and controls that are used to move the character during animations. Both the character mesh and its clothing

need to be skinned, or bound, to the rig so they deform correctly when the bones are moved. Proper weight painting of the clothing is important so the clothing moves correctly with the underlying body mesh and keeps its intended shape. Manually creating weights for clothing elements often requires extensive tweaking to ensure the weights follow the body mesh accurately. A common use case for weight painting character clothing is to copy the weights from the character body. This way the clothing follows the same principles as the character mesh when deforming with the rig and prevents the meshes from clipping into each other. (Halač 2023, 19.)

In cases where the low-poly mesh still remains relatively high in polycount, using proxy meshes for weight painting can be an efficient alternative. Proxy meshes are simplified versions of the low-poly model that still maintain the shape and structure of the original mesh while having less polygons. Skinning the proxy mesh allows for faster calculation and easier mirroring of the weights. Once the proxy mesh is skinned, the weights can be transferred to the low-poly mesh to maintain correct deformation. (Thunder Cloud Studio 2024a.)

A similar method can be applied when the clothing consist of individual meshes or doesn’t have a complete body mesh to copy weights from. Weight painting separate or overlapping surfaces can be difficult so combining the mesh into one seamless surface eases the weight painting process. Most of the weight painting can be done with this seamless mesh, after which the weights are transferred to the individual meshes and further refined if needed. (Halač 2023, 19.)

Copying weights from the body mesh gives a great starting place for skintight and organic clothing but might need additional fixes when working with longer cloths or more rigid pieces. Deforming rigid or semi-rigid clothing pieces like buttons, belts or armor needs to balance between transforming with the character and still retaining their rigid shape. If the same weight transferring trick is used, the pieces would lose their shape and bend organically with the body mesh. Instead, transferring weights from single verts or well-placed edge loops will give the intended look. (Halač 2023, 19.)

For completely rigid pieces like buttons, transferring weights from single verts will make the asset deform as a solid piece of geometry while still following the body mesh without clipping into the mesh below. This method works well with smaller details but can also be adapted for larger pieces. Large rigid assets like armor plates can be assigned to follow certain bones or be divided into smaller pieces to maintain proper deformation. Semi-rigid objects like belts can use a similar technique but instead using edge loops. Semi-rigid objects need to preserve their structure while still deforming, so using either an edge loop that spans along the target mesh or cutting and creating a new one for it will allow it to deform correctly. (Halač 2023, 19.)

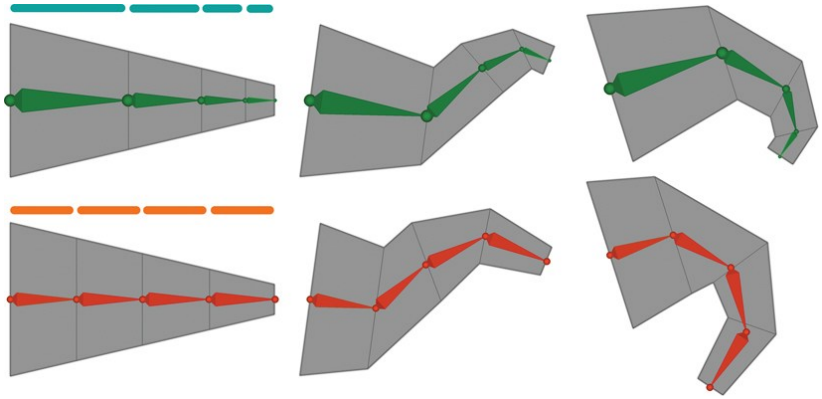
Weight painting the clothing to have similar weights to the characters body, like explained earlier, is the most common way to make clothing move. With this method the clothing follows the characters movements exactly. It’s a very inexpensive method in terms of performance and easy to set up but might give a somewhat rigid feel since there’s no free or secondary movement in the clothing. (Hypesio 2023.) There are multiple ways to deform clothing and specifically cloth to make it more dynamic in games.

Additional detail can be added by deforming the clothing separately from the character. This may require different solutions depending on how the mesh will be deformed in the engine, what kind of look is being striven for and the performance needs of the game. Simply put, the cloth mesh could be either simulated, animated with their own bones or be some combination of the two. If the cloth is fully simulated, additional bones might not be needed at all. Some combinations could include rigging the cloth and simulating the bones instead of the mesh or blending between animations and simulations. (Halač 2023, 29.)

Adding more detail and motion can be achieved by animating the cloth. The cloth could be manually animated either by using additional cloth bones or moving the mesh itself to give it movement. This method is typically used to create a more stylized look since it takes a lot of time and work to create realistic animations. Applying simulations to the bones or mesh while animating and baking the movement into the animation keyframes can be an effective way to create realistic movement. Simulations can be easily edited with different parameters

like gravity, weight, wind or other physical factors, and can be iterated rapidly until the desired result is obtained. (Hypesio 2023.)

Additionally, altering the cloth bone lengths can create a more organic movement to the mesh. Moving evenly spaced bones might create a stiff and mechanical look when animating or simulating. Instead, using bone lengths that get smaller toward the tip of the cloth element, like shown in Picture 12, creates a more dynamic look when animating or simulating. (Halač 2023, 29.)



PICTURE 12. Example of bone length affecting the mesh deformation. (Halač 2023, 29.)

Some studios also integrate motion capture data with the rigged clothing to copy real life movement to animations (Radivojevic 2024). Animating the mesh is a somewhat simple way to add realistic movement and it's less expensive to use animated cloth movement compared to using real time simulations in the engine. The limitation with manual or simulated animation is that they can't dynamically adapt to unpredictable player movement and might look unnatural if the animation doesn't match the situation or movement. (Hypesio 2023.)

The most dynamic way to move cloth is to use real-time simulations in the engine. It is the most expensive in terms of performance because the position of the vertices of the mesh is calculated each frame of the game, taking into account various physical parameters like the weight and plasticity of the fabric, wind and character movement. Simulations react to change dynamically so the cloth looks realistic with different character movements. The downsides of this method are that real-time simulation can be quite expensive and the visual fidelity can be affected by the lack of quality either in the cloth mesh or simulations. Collisions

and self-collisions can also be tricky to calculate and thus affect the realism. (Hypesio 2023.)

Whether achieved through manual animation, simulation or some sort of combination, the goal is to achieve believable motion and deformation. Avoiding visual errors such as clipping, unnatural bending or stiffness is important. Selecting the deformation method depends on the style, performance requirements and the level of realism desired.

3 PROJECT: MEDIEVAL FANTASY ARMOR

3.1 Project Requirements

For the practical project portion of this thesis, a medieval fantasy armor set is created from start to finish following the industry standard pipeline and methods that are discussed in the theory portion. The process includes the same steps as discussed in the theory portion including designing, modelling, texturing, rigging and applying final touches in engine. Additionally, some more practical advice and methods used to create the asset are shared throughout the project stages. The main focus is to create a game-ready model that utilizes multiple modelling methods, realistic PBR materials and cloth simulations.

The project was mainly created with Blender 4.0, Substance 3D Painter version 11.0.2 and Unity 6. Blender is used for modelling, preparing for texturing and rigging while Substance Painter is used for creating realistic textures and Unity 6 for the final rendering and cloth simulations. The project does not cover creating a rig or animations, and it does not go into character clothing design in detail, though skinning and weight painting the armor set to the rig is discussed, and all the design choices are explained and justified. The animations are imported from Mixamo and used in Unity to test the final model.

The aim for this project is to create a game-ready feminine armor set with realistic PBR materials. A medieval inspired armor set that incorporates both metals and cloths is created, featuring realistic and detailed textures. The project also experiments with real time cloth simulations within the engine to create dynamic movement. The mesh is designed to be quite simple and most of the detail is added through the PBR textures.

The armor set is a standalone asset but the requirements are set and treated as a modern PC or console game asset. The armor set is created as a detailed main character or an important side character armor, so the polycount budget is around 30-60k tris with 4096x4096 textures. The project has a time budget of about three weeks for the whole asset. To make production as swift as possible, a readymade

character base mesh is used while modelling and multiple material presets are utilized in Substance Painter. The design is kept quite simple to fit in the scope of this project.

3.2 Ideation and References

Once the requirements are set the ideation and reference gathering can begin. The base idea for the project was clear: a medieval fantasy armor set that combines metals and cloths. The idea was to create a somewhat feminine armor set that utilized real time cloth simulations and combines knight armor with ornate metal engravings and fabric patterns. The main inspiration came from games such as Baldur's Gate 3, The Witcher 3: Wild Hunt, Monster Hunter Wilds, Kingdom Come Deliverance II and other various open world fantasy games.

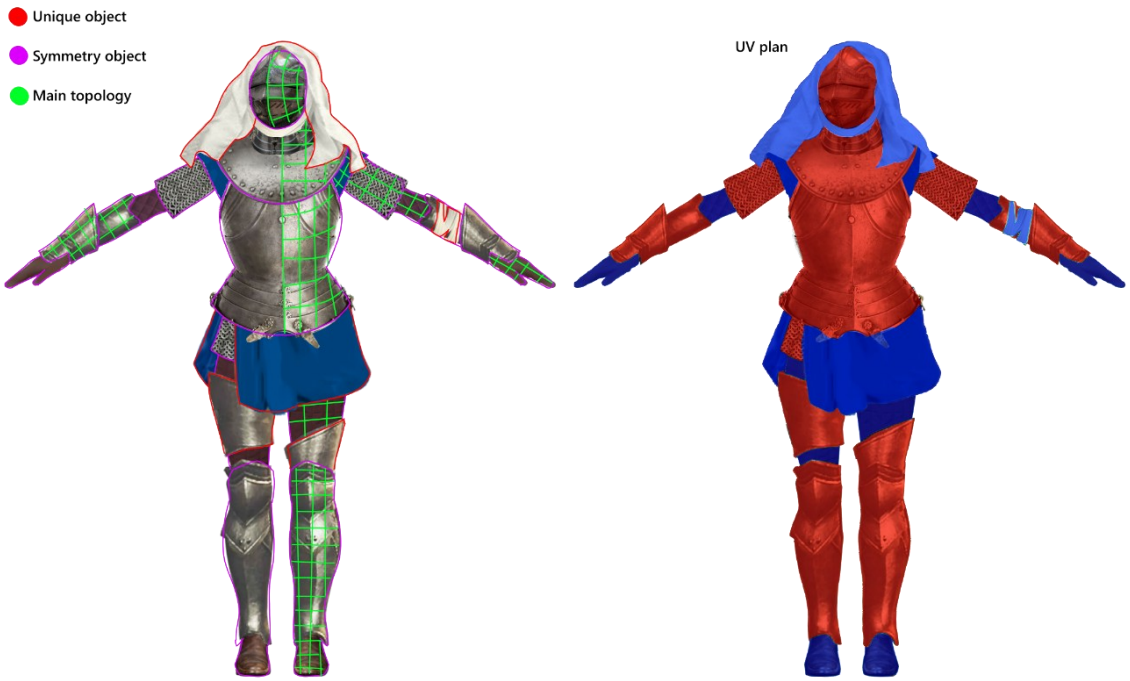
The project was started by gathering a wide amount of references that were eventually narrowed down to the core idea. Most of the references were found through Pinterest which were then gathered and sorted in PureRef. Pinterest offers an endless supply of suggested content which is helpful when searching for ideas. PureRef is great for its simple UI elements and organization options, making it easy to focus on the actual images and categorize them as needed. The initial references were divided into two main categories, with armor on one side of the board and clothing on the other. They were further categorized by key design elements such as armor segments, fabrics, materials and colors to streamline the creation workflow. More references were added throughout the project as needed, especially more angles of different objects for modelling and materials for texturing.

Once the main references were gathered a concept art image was created to streamline the ideas into a more coherent concept. Picture 13 depicts the base mesh concept for the project. The concept image was combined from multiple reference images in Krita and aims to help with the general mesh and material plan. The clothing materials and details were refined further throughout the project.

PICTURE 13. Concept art combined from reference images.

The scope, time limit and requirements of the project were taken into consideration while creating the concept art. The mesh was designed to be quite simple and to be able to utilize the mirror modifier when modelling, while still making some asymmetrical designs to keep the model interesting. Since the project aims to experiment with cloth simulations, the clothing was designed with that and the theory discussed throughout chapter 2 in mind. A simple design was created to avoid pain points especially with deformations. Both the dress and veil were designed to be quite short to simplify the weight painting process and to make sure less issues would rise with the deformations.

A simple modelling and UV plan was made to streamline the work process. This was done by drawing over the concept art as shown in Picture 14. Even though the mesh is mostly mirrored, some assets like the breastplate and collar plate have non-mirrored UVs so there’s no visible seams in the middle of the mesh.



PICTURE 14. Mesh and UV plan.

3.3 Modelling

The plan for the modelling phase was to model the base using mostly hard surface techniques and to use clothing simulations for the veil. The model base was kept quite simple with most of the detail added later in the texturing phase.

A “Neutral female” mesh was used from Reallusion as the base character mesh for the project. All the objects were combined into a single character mesh and the character was scaled and sculpted a bit larger. Since the armor set covers the whole character, the mesh was mostly used to create objects in the correct scale and to get the base mesh for the skintight layers.

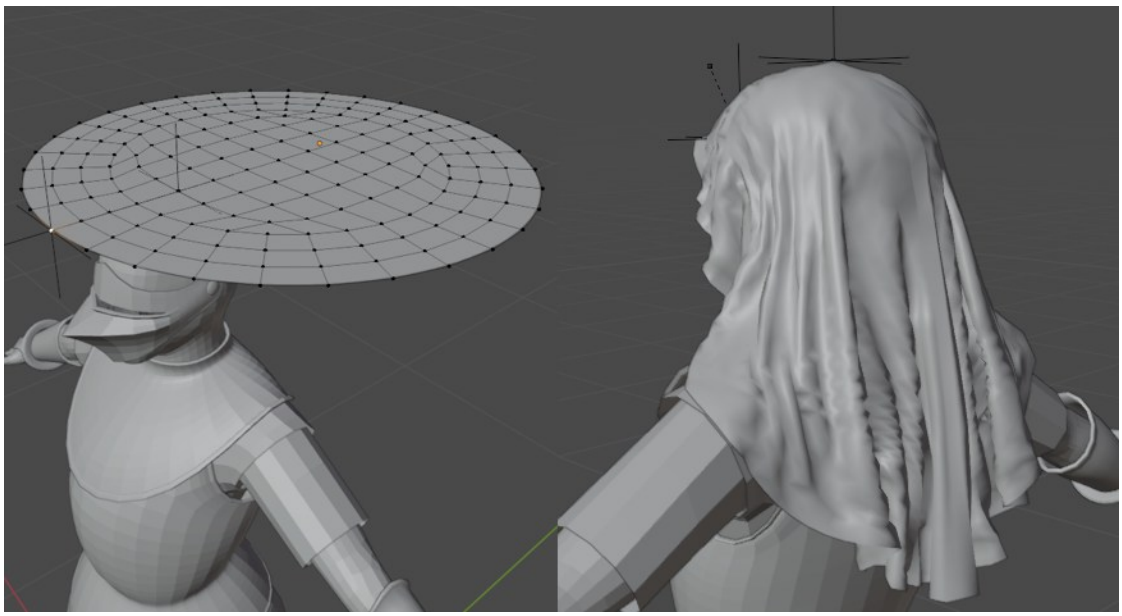
The blockout started with planes and cubes that were further shaped with hard surface methods and modifiers such as subdivisions and thickness, since most of the armor set was hard surface metal. Like planned in chapter 3.2, the mirror modifier was utilized for most of the assets. The skintight clothing assets, such as the gloves, shirt and pants, were duplicated from the base mesh and further altered to save time in the modelling and retopology phase. The blocked out armor set can be seen in Picture 15.

PICTURE 15. Blockout of armor set.

Two approaches were explored for creating the meshes for the real time simulations: one having a very simple base mesh and another with a more detailed mesh. The dress mesh is very simple and it has no ready-made creasing details while the veil has a complex shape and detailing in the mesh.

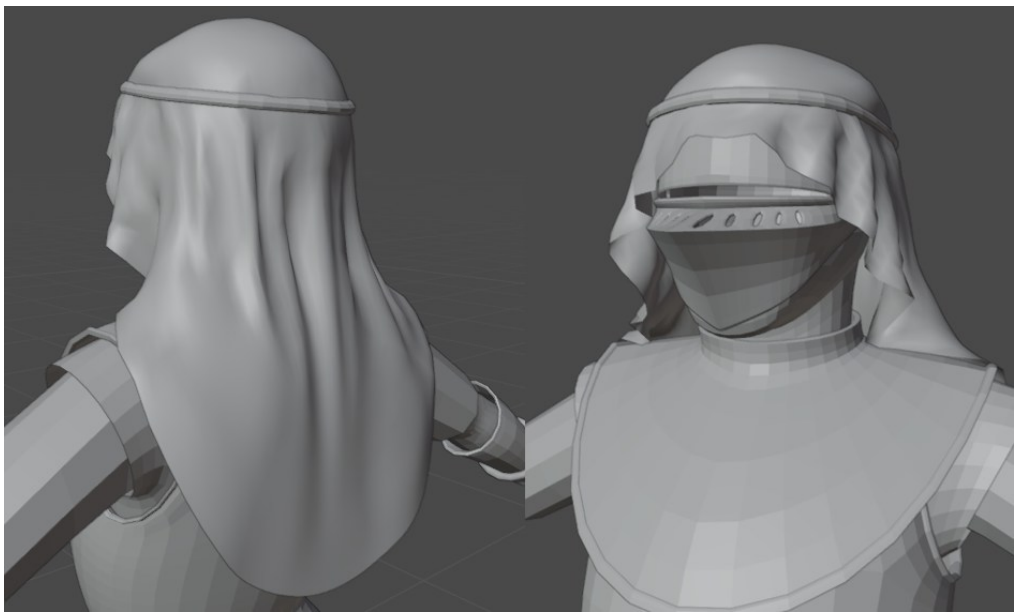
A combination of simulations was used to create the veil. The aim was to create more detailed folds in the front around the face and neck area, and less detailed at the back which would be later simulated in engine. Simulations were an easy way to create the realistic draping. A Blender add-on called “Bystedts cloth builder” was used, which made the simulation process much easier by allowing the addition of pins that could move the cloth during simulation.

A female collider that was provided with the add-on was posed to match with the armor mesh to get the correct shape and collisions when simulating. The shape of the flat veil mesh was copied from a real life veil to get similar draping. The simulation process included a great deal of trial and error to get a satisfactory result. Mainly the size of the mesh, pin placements, shrink factor and subdivision amounts were altered to get the desired look. The final pin placements and simulated mesh shape can be seen in Picture 16.



PICTURE 16. The shape of the veil simulation base mesh and the simulated result.

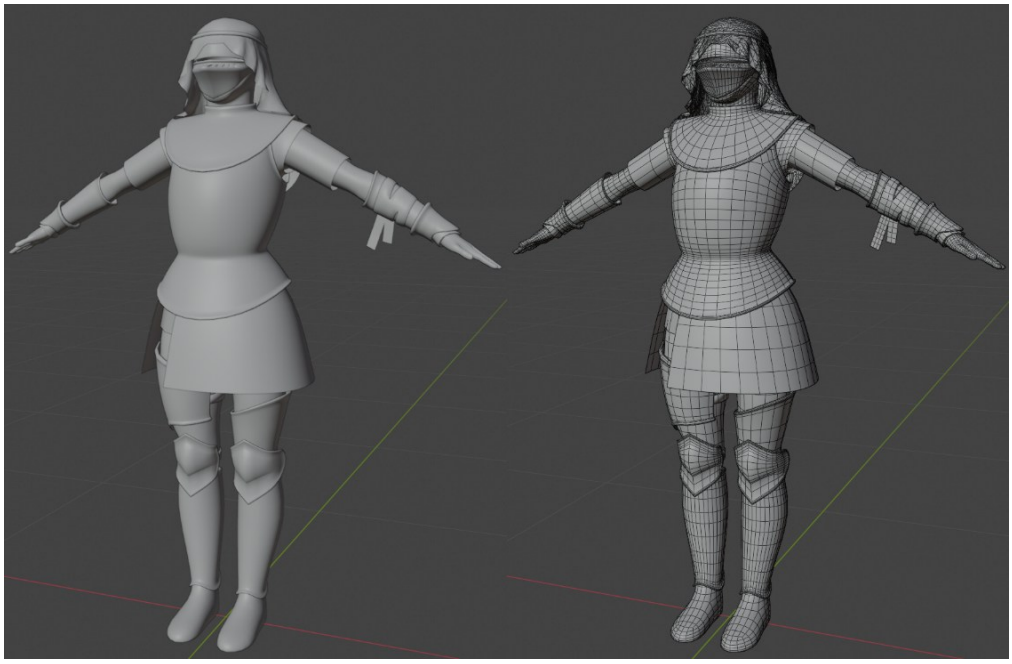
Instead of converting the finished simulation to a mesh, the base was applied from the Bystedts cloth builder shape settings and the cloth component then removed. This way the topology stayed as quads and could be further worked by sculpting it. Since the mesh was going to be simulated again in the engine, most of the folds in the back were smoothed out to get a better simulation result. The details around the face were left since the area was not going to be simulated as heavily. The cleaned up veil can be seen in Picture 17.



PICTURE 17. The shape of the sculpted and cleaned up version of the veil.

The blockout was refined further and more details were added such as thickness and rounded edges for the armor pieces. Multiple versions of meshes with different amounts of detail were saved throughout the project. These versions, which included unapplied subdivision modifiers, allowed for easy alterations. Objects that were going to be simulated in engine like the dress and the veil were kept as flat planes. Additionally, the chainmail shirt was also kept as a plane because of the see-through material it would utilize.

To optimize the production workflow, the need to retopologize was intentionally minimized. Good topology techniques such as using quads and even topology were followed throughout the modelling phase. Because the armor set was mostly made of hard surface objects, the need for a complete retopology was reduced compared to sculpted assets. The refined blockout was used for optimization and was refined and cleaned up as needed. The meshes that were not visible were removed as well as any excess edge loops that did not contribute to the silhouette. The final optimized model had around 38k tris which was within the budget that was set in chapter 3.1. Most of the tris were concentrated in high detail areas such as the fingers, helmet visor or the veil, or in areas that require round shapes such as the breastplate. The final low-poly model and its wireframe can be seen in Picture 18.



PICTURE 18. Low-poly mesh and its wireframe view.

The high-poly meshes were created by utilizing the refined blockout meshes and cleaned up base meshes. For this project, not many high-poly details were created to the meshes so they could be created in the texturing phase instead. This was a matter of personal preference, time limits and project needs as it suited the project to have more flexibility in the texturing phase. The armor set consists of mostly hard surface objects so the detailing focused largely on the edge details of armor trims or the breastplate and collar areas as shown in Picture 19. Since the armor meshes were hard surface models, the details were created by utilizing modelling techniques such as adding and altering seams and beveling edges rather than sculpting, to maintain the rigid shapes. Organic clothing assets such as the shoes and the top of the veil were further detailed by sculpting. Different sculpt brushes such as the clay, crease, mask and cloth brushes were used to create more details. Picture 19 shows the final high-poly model before additional texturing details.

PICTURE 19. High-poly mesh.

3.4 Texturing

This project uses PBR materials to get the most realistic texturing results. The armor set uses the basic maps of albedo, metalness, roughness and normal map

while additionally utilizing opacity maps to create more details around the cloth edges. Substance 3D Painter was used for the baking and texturing process because of its powerful and versatile toolset, popularity in the industry and prior experience with the software. Most of the details for the armor set are created in the texturing phase.

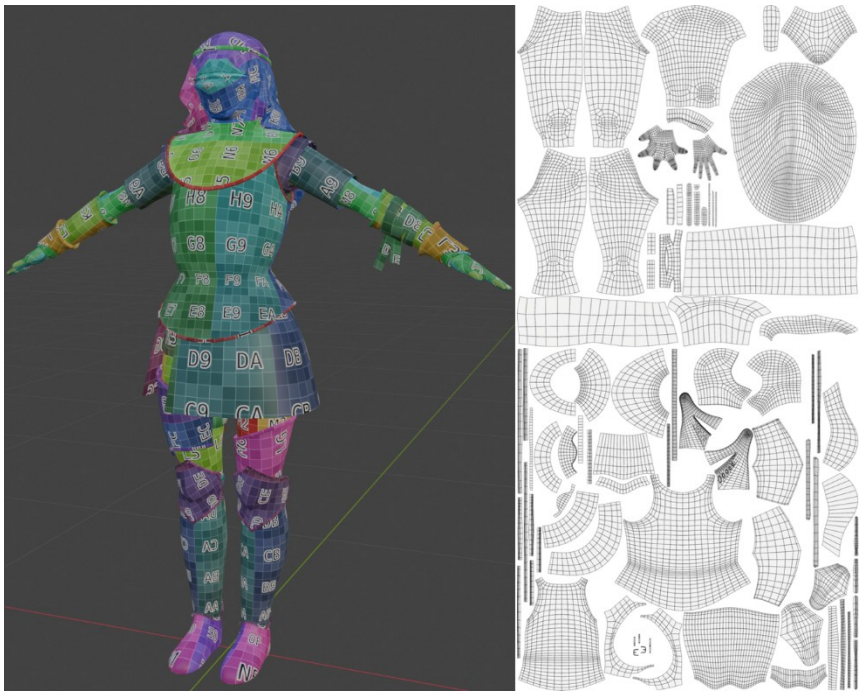
Before the texturing phase can begin, the model must be UV unwrapped correctly. Like planned in chapter 3.2, the armor set used two texture sets that were divided into metallics and cloths, although they overlapped slightly. Armor pieces have their own UV map while the cloths, leathers and the chainmail have their own. The chainmail was mapped with the cloths because they both utilize opacity to enhance realism and create additional details.

The UV mapping is started by placing seams as discussed in chapter 2.4, to create optimized textures. The seams were hidden in less visible places as well as placed to cut out the armor trims. Blender's "Display stretch" was used to test and create optimized UVs. With both materials assigned to their own texture sets, all meshes sharing the same material were selected and unwrapped together. This granted a consistent texel density for the assets and divided the UV space for objects based on their relative sizes.

After unwrapping everything, the organization could start by using Blender's own UV tools as well as an add-on called "Mio3 UV". A few specific aspects were kept in mind when unwrapping and organizing the UVs: collecting UVs by the object, UV island rotations and asset texturing needs. Placing an objects UVs closely together on the UV map optimizes the texturing calculations in Substance Painter. Additionally, UVs of objects close together in 3D space were placed near each other. In practice this meant using Blender's "pack islands" feature on each object separately to automatically organize them close to each other. The scale and rotation options were not used, so the originally set scales stayed the same. The rotation of UV islands is especially important when texturing with repeating patterns. To ensure the pattern looks realistic, for example the dress's skirt and sleeves were rotated to the correct direction, mimicking how real life clothing would be constructed. Additionally, armor trim edges were rotated to face the same way as well as straightened with Mio3 UV settings to make them more

optimized and to ease the texturing process. Some less prominent UVs like the underside of the armor were made a bit smaller to grant more space for the more prominent assets.

After these alterations the UVs were manually organized into the UV map. The UVs were optimized as efficiently as the workflow allowed with tight packing, appropriate padding for texture bleeding and consistent scaling and rotations. A UV grid texture was used to ensure as little distortion as possible throughout the UV mapping. The UV grid test and final UV layouts can be seen in Picture 20, showcasing the consistent texel density and distortion-free texture achieved with the UVs.



PICTURE 20. UV grid texture on the left and final cloth and metallic UV maps on the right.

Substance 3D Painter uses additional maps that are created to make the texturing process easier. Maps such as the world space normal map, position map and ID or vertex map are great tools for the texturing process. These maps are not utilized as final textures but instead as masks or generator inputs for various layers, for example. These maps are created in Substance Painter while baking but some require additional preparations. A typical step is to create ID or vertex maps that ease the texturing process by making it fast to mask out certain

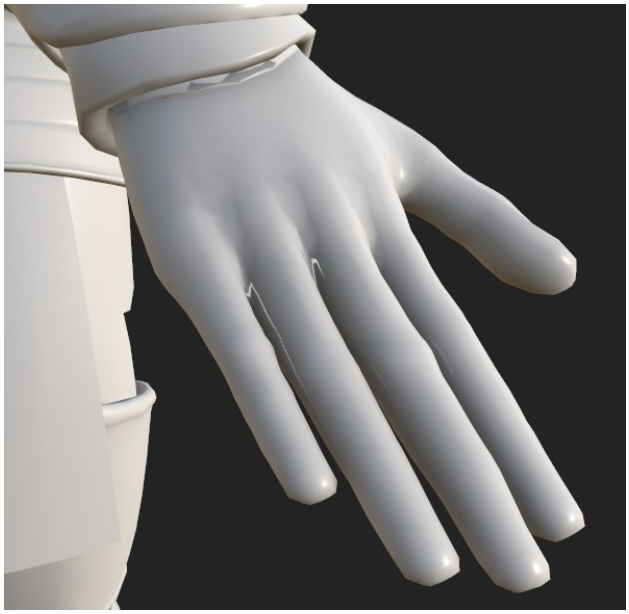
areas. Since the armor set consisted of separate objects, for this project it was simple to use ID maps. The ID map can be generated from separate meshes while baking in Substance Painter. One alternative would be to assign vertex colours in Blender to different parts of the assets. Lastly, with Substance Painter it's important to name all low-poly meshes with _low and corresponding high-poly meshes with _high at the end of their name for the baking process to work correctly. With these preparations done, the optimized low-poly mesh and detailed high-poly mesh can be exported to texturing.

The low-poly asset was imported to Substance Painter and the project utilized the Blender (starter_assets) template with 4096x4096 textures. The texturing process began in Substance Painter by baking the mesh maps. The baking output size was set to be the same as the texture sizes and the high-poly mesh was imported and set to "match by name". As discussed before, the ID map color source was set to "Mesh ID/polygroup" to separate each mesh into their own ID group for easier masking. Additionally, ambient occlusion, curvature & thickness "self occlusion" setting was set to "only same mesh name" to prevent separate meshes influencing each other while baking. This was set because the meshes move around and away from each other while animating and the static baked shadows would look incorrect.

While these settings affect the result, the most prominent setting for baking in Substance Painter is the cage settings. The cage is a shell that is created around the low-poly object and determines how far the baking rays are casted from the low-poly mesh. Essentially, the size of the cage affects the outcome of the bakes by either including or excluding details. Artifacts can occur if the cage overlaps itself while baking.

The armor set mesh had some details and tight corners that caused problems in the baking process. When first baking with the "distance-based" cage, the bakes had some baking artifacts. The issues arose due to assets needing different cage sizes, as a smaller cage didn't capture all the protruding details and a larger cage overlapped itself in tight areas. Picture 21 showcases how the cage overlapped itself between the fingers and around the wrist and caused artifacts. After some testing, the baking had better results with the "automatic (experimental)" cage

which calculated the cage distance dynamically instead of using a set distance for all meshes. This method worked better but some artifacts remained around tight areas.



PICTURE 21. Baking artifacts in tight mesh area.

As a solution meshes were altered in Blender. Artifacts appeared because the high-poly mesh was too dissimilar to the low-poly mesh of an armor piece and because an object was too detailed as a single mesh. The smallest details that had overlapping geometry were separated and the high-poly mesh was altered to resemble its low-poly counterpart better. After fixing the mesh issues the assets were reimported to Substance Painter and baked again using the “automatic (experimental)” cage with good results.

With the baking completed the texturing process could start. Substance Painter’s highlight is the ability to create materials utilizing multiple layers and different types of generators and outputs. These different layers are translated into the texture maps and allow for easy alterations throughout the texturing process. This project utilized multiple readymade materials from Substance Painter’s asset library and Adobe Substance 3D community assets that were further refined with details made with path tools, masks and different alphas and textures.

The metallic material for the armor utilized Substance Painter’s “steel medieval stylized” smart material as the base. Additional details such as ornate carvings

and edge trims were created by altering the height and color output while utilizing masks that had tileable patterns. For the shirt and pants, “Plastic Fabric Pyramid” and “Plastic Fabric Bands” materials were used from the Adobe community assets to create padded fabrics. The roughness and metal parameters were altered and additional mask layers added to create more interesting and medieval inspired coloring. Similarly, the chainmail shirt was created with an “Iron Chainmail” material from the community assets. To enhance realism, the material shader needed to be changed to “pbr-metal-rough-with-alpha-test” to include opacity. The opacity was also used to clean up and decorate the hem lines of all the flat clothing to create more realistic and uneven edges. The clothing such as the veil, dress and arm wrap were textured using a linen material with additional detail layers utilizing similar techniques as stated before. Many height altering details were painted on with paths and brushes to create more creasing details for the arm wrap. The cloths that were going to be simulated were kept flat from additional creasing details to avoid static shadows while simulating.

More details such as patterns and colors were ideated and tested throughout texturing, which turned out to hinder the texturing phase. Additionally, deciding on the final look of the veil and dress patterns was quite difficult and took up much more time than anticipated. A more detailed texturing plan would have made the workflow more efficient.

All the materials were created by first creating simple base materials and adding more detail layers on top. The materials were created to look realistic and medieval inspired, by adding bright colors and various patterns. The metals were kept quite simple with additional floral engravings to create an expensive look while the clothing materials utilized more vibrant colors and simple patterns. The clothing was kept quite matte to contrast the shiny armor. Lastly, color variation and final details like dirt and dust were added on top of all the layers to enhance the realism and create changes in the texture channels. The final materials can be seen exported to Blender in Picture 22.

PICTURE 22. Final materials in Blender.

The materials were tested throughout the texturing process in Blender and in Unity. The materials need to be exported differently depending on the rendering software, so the Blender (Principled BSDF) and Unity Universal Render Pipeline (Metallic Standard) -templates were used respectively to export the textures in correct formats. The final materials utilized albedo, metalness, roughness, normal and opacity maps to create detailed and realistic PBR materials.

3.5 Rigging

The project used a basic humanoid rig for rigging since additional cloth bones were not needed for the simulations. The rigging process faced many issues and required a great deal of testing to work in Unity with the Mixamo animations.

Multiple rigs were first tested on the character base mesh before proceeding to the armor set skinning. This was done to evaluate and test the compatibility of different rigs with Unity's animation system and the Mixamo animations. Both Blender's rigify rig and an imported Mixamo rig were tested in Blender but both had deformation and animation issues with Unity's humanoid avatar setup. Through testing, the rigify meta rig was found to work best and transferred correct weights and bone hierarchy for the humanoid avatar setup. Therefore, the armor

set was skinned to the rigify meta rig to ensure proper deformations and compatibility with the animations in Unity.

Tool settings like “mirroring” and “auto normalization” were used throughout the weight painting process to ease the workflow and divide weights correctly. Additionally, “front faces only” setting was turned on and off to easily paint through meshes when needed. The underlying body mesh edge loops were occasionally used to transfer the weights between objects, though it worked with varying success. Even though only the wanted edge loops were selected on the meshes, the weights of the entire object were transferred. Due to this, mostly manual weight painting was used as well as Blender’s weight smoothing options.

The character mesh was first skinned by parenting it with automatic weights to the rig and used as a guideline on how the clothing and armor meshes should deform. Similarly, all the armor set meshes were parented to the rig with automatic weights and then further weight painted manually. Extra attention was paid to the cloth elements to avoid any unnatural bending. The veil was kept free from hand bone influence to avoid unnatural movements and the weights of the dress were painted so they would follow the hip and leg movements naturally.

Completely rigid meshes like the arm and leg armor were simply weighted to a single bone, while the breastplate, collar and helmet proved a more difficult weight painting task. Since these areas required both rigidness and deformation, a balance was pursued by manually weight painting the meshes to the bones while continuously testing them by moving the rig. Some deformation needed to be added to make the assets work with the imported animations.

The collar was the most difficult asset to weight paint since it had to be rigid while also being affected by the arms, shoulders and neck. The initial weights looked correct in Blender by being mostly weighted to the shoulder bones, but the Mixamo animations made the mesh unnaturally bent in Unity. This could be due to the differences between the source rigs, which changed the shoulder bone rotation radically. As a solution, the collar was weight painted to only be affected by the neck bones which gave it a more rigid look. Though this solution created

additional problems since animations that had more wide arm movements would clip through the collar.

The final rigging results were not as realistic as hoped because some deformations had to be created to the armor around the torso and neck area. Additionally, the rigid sections also caused issues by clipping through meshes when animated. Even though the rigging was mostly successful and realistic, some parts of the mesh caused issues that resulted in a less refined outcome. Overall, further knowledge and alterations of weights or changes in the animations would be required to get a more realistic result.

3.6 In Engine Cloth Simulation

Unity 6.0 was used to render the final armor set and the real-time clothing simulations. Additionally, animations from Mixamo were used to test the armor set and cloth movement. It's important for all the meshes to have scale parameter of 1 to ensure consistent simulation results, so all transforms in Blender were applied and "FBX units scale" was used when exporting the armor set. Due to the time limitations, the Unity scene was kept empty even though the model would have benefitted from more complex lighting to showcase the materials better. These additional enhancements were left out to focus on setting up the cloth simulations.

Like stated in chapter 3.4, textures were exported from Substance Painter in the correct form and applied to basic URP Lit materials. Additionally, a second cloth material was duplicated and altered for the simulated cloths. This material was set to render both sides of the mesh and used alpha clipping to show the transparency map details. The double sided material was only used on the cloth meshes that required it to optimize performance.

The cloth simulations were created with Unity's cloth component. The cloth components were added to the veil, skirt and arm cloth meshes. Capsule colliders were added to the head, chest, arms, pelvis and legs and each parented to their respective bones to follow its movement. The colliders were scaled and positioned to make sure the cloth simulations didn't clip through nearby meshes

and created realistic movements. To get a desired simulation result, mostly the cloth constraints, stretching and bending stiffness, damping as well as cloth self and inter collision settings were altered.

While testing the cloth meshes, the cloth constraints were created with the gradient tool. Since the gradient only works from left to right, the object had to be rotated sideways, selected and painted with the gradient, and then rotated back to get a smooth transition between weights from top to bottom. When creating constraints for the final rigged and weight painted model, this method did not work for some reason. So, the cloth constraints were selected and painted manually while still creating some gradient to ensure a smoother transition between simulated and non-simulated mesh parts.

Like stated before, the animations were imported from Mixamo and copied using humanoid avatars. The animations brought out multiple issues including the mesh clipping and weight issues that were discussed before as well as some cloth simulation issues with the veil and the dress. With certain animations like walking, the veil started to jitter at the top of the characters head. To solve this, the mesh polycount was lowered and cleaned up in Blender, and the root bone was changed from the default one at bottom of the character to the head bone. The jittering could have somehow been caused by the animations since it only appeared on the cloth that was near the top of the head. Additionally, the front half of the dress got stuck on itself and didn’t flow as evenly when animations were playing and changing. A reason for this was not found nor a solution to fix it, though it was only a minor issue. It could have been due to rapid changes in the character’s movement that somehow broke the dress simulation. Perhaps further alterations to the cloth settings could have fixed the issue.

The final armor set asset with the simulations added in Unity can be seen in Picture 23. Additionally, a video demonstrating the armor set and the cloth simulations with several animations is linked in Appendix 1.

PICTURE 23. Final armor set with cloth simulations in Unity.

4 DISCUSSION

The project portion of this thesis covered an example of a clothing and armor creation pipeline for a realistic 3D game asset that was based on the theory learned in chapter 2. The project covered setting project requirements and utilizing them when collecting and creating reference art. The mesh creation was gone over from blockout to high-poly detailing and included hard surface modelling techniques for most of the model while cloth simulation was used to create the veil. The mesh creation was optimized so the retopology phase would mainly consist of cleaning up and optimizing the existing meshes rather than needing to do complete retopology. Optimized UVs were created in Blender while baking and texturing was done in Substance Painter. Typical PBR texture maps of albedo, metalness, roughness, normal and additionally opacity maps were used to create industry standard and realistic materials. Blender's rigify meta rig was used for the rigging process. The meshes were skinned to the rig and imported to Unity where Mixamo animations were used to test and showcase the model and simulations. Double sided materials and cloth simulations were added to the cloth elements to showcase the final results.

Overall, the project turned out quite well when comparing it to the industry standard theory as well as the requirements set at the start of the project. The initially set project requirements and concept art worked great to guide the project along. The requirements helped with decision making and allowed the project to move along steadily while the simple concept art created and the references gathered help with mesh and texture creation. The armor set mesh creation went over smoothly without any major issues and the cloth simulations in Blender turned out to be quite easy to use while providing realistic results. Additionally, the optimized mesh creation left retopology unneeded and allowed for a faster workflow.

Despite some flaws in the workflow, the final PBR materials created in Substance Painter turned out great and created a high quality and realistic looking model. The materials enhanced the simple meshes and created a more interesting model

like planned. The mesh and materials were the highlights of the project, creating the initially planned high quality and realistic looking armor set asset.

As discussed before, the final rigging result was satisfactory but did not reach the expectations of a realistic model due to clipping and deformation issues. The rigid armor meshes created issues with the animations in several places like the torso and the shoulder area and needed to be altered several times. Additionally, getting the animations to work in Unity was challenging and caused some issues but were solved after some testing. The cloth simulations in Unity worked better than expected and created quite realistic results. Both the less detailed flat dress mesh and the more sculpted veil mesh created visually good results and worked well in engine. As planned, the final result of the project was a game-ready, highly detailed and realistic armor set that utilized PBR materials and in-engine cloth simulations.

Even though the armor set asset turned out relatively well there are some aspects that could have improved both the asset and asset creation workflow. Firstly, the concept plan should have been more precise to limit indecisiveness and the time spent testing different patterns and materials in Substance Painter. Even though the textures of the asset came out quite good, the lack of a strict concept hindered the workflow since a great amount of time was spent testing different patterns and workflows for the textures. Deciding on the final textures also proved more difficult than initially thought. Additionally, the techniques used for detailing were quite labour intensive and not very optimized. Since the model used quite a lot of patterns and details, most of them could have been created into a single texture that would be imported and utilized in the texturing phase, rather than creating the details in Substance Painter separately. This method would require a precise plan but would ease the workload in Substance Painter and could be easily utilized by different generators and masks, for example.

As stated before, the rigging and animations turned out to be the major pain points of this project. The rigging and animations took out more time than was planned due to different issues faced throughout the phases. The final rigging result was satisfactory but did not reach the expectations of a realistic model since the meshes created issues with the animations. More time should have

been allocated to the weight painting process or the animations altered to fit the rigid movements of the model better. The results would have been more realistic if the animations were created specifically for an armored character, limiting the movement in places where there's rigid armor like the breastplate and collar area. Perhaps modelling everything to be one single mesh instead of multiple separate ones could have helped with the clipping or on the contrary, making the mesh more separate around joints so the objects could remain rigid would have helped in the rigging phase. More research should have been done about the practical aspects of weight painting rigid objects.

The asset would have benefitted from more interesting lighting in Unity to better showcase the PBR materials. The simulations in Unity would probably need further tweaking when character movement is added to avoid issues with the mesh getting stuck, clipping or creating erratic movements when the character suddenly moves. Altering the cloth constrains and only limiting the freest movements to the ends of the garments like sleeves or hems could create more seamless and predictable results. Obviously, the results of the clothing simulations are tied to the game engine capabilities and thus results could differ when using a different engine.

In conclusion, this thesis provides a broad and practical guide to creating realistic 3D character clothing. It examined and explained commonly used industry techniques and theory in a concise way. In addition to the theoretical framework, the project section demonstrated an example of a production pipeline. Despite some challenges, the workflow ultimately produced game-ready and visually realistic clothing and armor assets.

REFERENCES

Adobe. 2025. Mesh and UV setup | Substance 3D Painter. Adobe 22.7.2025. Read on 25.10.2025. https://helpx.adobe.com/substance-3d-painter/technical-support/performances-guidelines/mesh-and-uv-setup.html

Adobe. n.d.a. 3D texturing solution with Adobe Substance 3D. Adobe. Read on 26.10.2025. https://www.adobe.com/products/substance3d/discover/3d-texturing.html

Adobe. n.d.b. What is a polygon in 3D modeling?. Adobe. Read on 14.10.2025. https://www.adobe.com/products/substance3d/discover/3d-polygon-modeling.html

CGTyphoon. n.d. Types of non-manifold geometry. Read on 21.10.2025. https://cgtyphoon.com/fundamentals/types-of-non-manifold-geometry/

Chernyshenko, N. 2022. Design, Detail and Animation: Creating Custom Clothing with Marvelous Designer. The Rookies 19.6.2022. Read on 20.10.2025. https://discover.therookies.co/2022/06/19/design-detail-and-animation-creating-custom-clothing-with-marvelous-designer/

Halač, A. 2023. A Complete Guide to Character Rigging for Games Using Blender. E-book. CRC Press. Available through O'Reilly for Higher Education (access requires institutional subscription). Read on 10.10.2025. https://www.oreilly.com/library/view/a-complete-guide/9781000934786/

Hypesio. 2023. Cloth simulation – Real-Time Techs #1. Hypesio 28.9.2023. Read on 12.10.2025. https://hypesio.fr/en/cloth-simulation-real-time-techs-1/

Iontcheva, I. 2025. What Is PBR (Physically Based Rendering)? A complete guide. Chaos 13.8.2025. Read on 25.10.2025. https://blog.chaos.com/what-is-pbr-physically-based-rendering-a-complete-guide

Karimfazli, P. & Tokarev, K. 2019. Game Character Production: Industry Standards. 80 Level 14.8.2019. Read on 15.10.2025. https://80.lv/articles/game-character-production-industry-standards

Kevuru Games. 2023. The Complete Guide to 3D Character Modeling for the Uninitiated. Kevuru Games 16.3.2023. Read on 14.10.2025. https://kevurugames.com/blog/the-complete-guide-to-3d-character-modeling-for-the-uninitiated/

King, S. 2023. Digital Character Creation for Video Games and Collectibles. 7.0 Technical Example 01: High-detailed PBR game character. E-book. CRC Press. Available through O'Reilly for Higher Education (access requires institutional subscription). Read on 10.10.2025. https://learning.oreilly.com/library/view/digital-character-creation/9781000883466/

Radivojevic, F. 2024. From Concept to Reality: The Secrets Behind Stunning 3D Game Clothing. RenderHub 19.6.2024. Read on 20.10.2025. https://www.renderhub.com/blog/from-concept-to-reality-the-secrets-behind-stunning-3d-game-clothing

Radivojevic, F. 2023. UV Mapping in 3D: Exploring Techniques, Essential Tools, and Industry Applications. RenderHub 27.12.2023. Read on 25.10.2025. https://www.renderhub.com/blog/uv-mapping-in-3d-exploring-techniques-essential-tools-and-industry-applications

Shahbazi, N. 2025. What Are the Differences Between 3D Modeling and Sculpting? Pixune 4.9.2025. Read on 17.10.2025. https://pixune.com/blog/3d-modeling-vs-sculpting/

Steppig, M. 2022. Squeaky Clean Topology in Blender. E-book. Packt Publishing. Available through O'Reilly for Higher Education (access requires institutional subscription). Read on 23.10.2025. https://learning.oreilly.com/library/view/squeaky-clean-topology/9781803244082/

Thunder Cloud Studio. 2024a. Character Modeling Workflow: Creating Realistic 3D Viking Characters. ArtStation 21.10.2024. Read on 13.10.2025. https://www.artstation.com/blogs/thundercloudstudio/mpQLn/character-modeling-workflow-creating-realistic-3d-viking-characters

Thunder Cloud Studio. 2024b. Topology for Low Poly Game Characters : Essential Tips and Tricks (Part 1). ArtStation 17.10.2024. Read on 16.10.2025. https://www.artstation.com/blogs/thundercloudstudio/y1Nyv/topology-for-low-poly-game-characters-essential-tips-and-tricks-part-1

Thunder Cloud Studio. 2024c. Topology for Low Poly Game Characters : Essential Tips and Tricks (Part 2). ArtStation 18.10.2024. Read on 16.10.2025. https://www.artstation.com/blogs/thundercloudstudio/G980G/topology-for-low-poly-game-characters-essential-tips-and-tricks-part-2

APPENDICES

Appendix 1. Video of the armor set asset in Unity

Link to YouTube video: https://youtu.be/0_o9gKRqGlg