Tampereen ammattikorkeakoulu

Realistic 3D Interior Environment Creation in Unreal Engine 5

Tomas Wass

Bachelor's Thesis
December 2024

Business Information Systems
Games Production

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Games Production

WASS, TOMAS:
Realistisen 3D-sisäympäristön luominen Unreal Engine 5 -pelimoottorissa

Opinnäytetyö 23 sivua
Joulukuu 2024

Unreal Engine 5 -pelimoottori on viime vuosina saanut suosiota erityisesti realisisten peliympäristöjen luomisessa. Opinnäytetyön tavoitteena oli luoda realistinen 3D-sisäympäristö Unreal Engine 5 -pelimoottorilla. Tarkoituksena oli tutkia erilaisia 3D-mallinnustekniikoita ja -menetelmiä realistisuuden saavuttamiseksi pelimoottorissa.

Opinnäytetyössä yhdistettiin teoreettinen tausta ja käytännön kehitystyö. Työssä tarkastellaan Unreal Enginen uusia teknologioita ja niiden hyödyntämistä realistisuuden saavuttamiseksi sekä mallinnuksen eri vaiheita. Projektin 3D-mallinnus ja teksturointi toteutettiin Blender- ja Substance 3D Painter -ohjelmilla, ja lopullinen ympäristö rakennettiin Unreal Engine 5 -pelimoottorissa.

Tulokset osoittavat, että Unreal Engine 5 soveltuu hyvin realististen ympäristöjen luomiseen ja tarjoaa tarvittavat työkalut myös riittävän hyvän suorituskyvyn saavuttamiseen. Pelimoottorin uudet teknologiat mahdollistavat realistisen ympäristön luonnin ja sen reaaliaikaisen renderöinnin.

Opinnäytetyön tuloksena syntyi interaktiivinen ja visuaalisesti realistinen 3D-sisäympäristö, jonka kehitysprosessi tarjoaa hyödyllistä tietoa realististen peliympäristöjen rakentamisesta Unreal Engine 5 -pelimoottorissa. Lisäksi projekti osoittaa, kuinka tietyt mallinnustekniikat ja uudet työkalut voivat helpottaa työprosesseja ja lyhentää kehitysaikaa. Jatkotutkimus voisi tarkastella, miten Nanite- ja Lumen -teknologioiden käyttö vaikuttaa pelimoottorin suorituskykyyn erilaisissa ympäristöissä.

Asiasanat: 3D-mallinnus, Blender, realismi, Unreal Engine

ABSTRACT

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

WASS, TOMAS:
Realistic 3D Interior Environment Creation in Unreal Engine 5

Bachelor's thesis 23 pages
December 2024

This thesis focuses on the use of Unreal Engine 5 for creating realistic 3D interior environments. The objective was to design a highly realistic 3D interior environment using the engine, while the aim was to explore various 3D modeling techniques and methods for achieving realism within the engine. The project demonstrates how modern game engine technologies can enhance both visual fidelity and performance.

The environment was developed using Blender for modeling, Substance 3D Painter for texturing, and finalized in Unreal Engine 5. The theoretical section explores Unreal Engine 5's new technologies and their application to realism, as well as the different stages of 3D modeling. The practical section focuses on applying these concepts to create the final environment.

As a result, an interactive and visually realistic 3D interior environment was created. The development process offers useful insights into building realistic game environments. The findings demonstrate that Unreal Engine 5 is well-suited for creating realistic environments and provides powerful real-time rendering capabilities. Future research could further explore the impact of Nanite and Lumen on different kinds of game environments.

Key words: 3D modeling, Blender, realism, Unreal Engine 5

SISÄLLYS

ABBREVIATIONS AND TERMS

Blender	A free and open-source 3D modeling program
Environment	Digitally created space consisting of objects in a three-dimensional space
Nanite	Unreal Engine 5's virtualized geometry system
Normal	Vector perpendicular to a surface or polygon
PBR	Physically Based Rendering, a realistic rendering approach
Rendering	Generating digital images of a 3D scene
Substance 3D Painter	Texturing software made by Adobe
Texture	2D image that can be applied to the surface of a 3D model
Topology	The arrangement and flow of vertices, edges and faces on a 3D model
Unreal Engine 5	Game engine made by Epic Games
UV	Coordinate system used to map a 2D texture onto a 3D model
Workflow	A step-by-step process that can be repeated

1 INTRODUCTION

The creation of realistic 3D environments in game engines is now much more accessible than ever before. Unreal Engine 5 introduced two powerful technologies, called Nanite and Lumen. Nanite offers optimized geometry rendering, and Lumen provides real-time global illumination. These technologies improve the environment creation workflow and allow realistic environments to be rendered in real-time.

This thesis goes through the process of creating a realistic 3D environment using Unreal Engine 5, with the help of Blender and Adobe's Substance 3D Painter. Blender is a free 3D modeling program and Substance 3D Painter is a texturing program. The project aims to take advantage of Unreal Engine's new technologies to achieve realism while ensuring optimal performance within the game engine.

The creation of realistic environments requires a diverse set of skills, including 3D modeling, texturing and understanding of lighting. Bad geometry, noticeable issues in textures, or bad lighting can break the illusion of realism, so the mastery of all these steps is required when trying to create a realistic environment. Though Unreal Engine 5 has many options one might never need to use, it is still important to know its most important tools to be able to use it to its fullest potential. The same logic applies to Blender and Substance 3D Painter.

In this thesis a practical project is done to help showcase the environment creation workflow and the integration of these tools and technologies.

2 REALISM IN GAMES

2.1 Realism in Game Development

The advancement of video game graphics has had a big impact on the video game industry. Game graphics have evolved from pixelated to photorealistic and screen resolutions have drastically increased. New rendering technologies such as global illumination and ray tracing have become increasingly popular in video games and increase the realism of shadows and lighting (Atreya 2022, 5353). These graphical improvements do also raise some challenges for developers, as highly detailed games need to also run somewhat smoothly on lower-end hardware. The pursuit of realism is still beneficial for driving innovation in both software and hardware.

The improvement of video game graphics has resulted in increasingly immersive game experiences for players and increased players' expectations, as they demand more realistic environments in games they play (Atreya 2022, 5354). Modern hardware and rendering technologies mostly eliminate constraints on how realistic games can look. Developers no longer need to design around hardware limitations and can instead focus on their own creative visions.

The introduction of NVIDIA's RTX graphics cards in 2018 was a big leap for realism in video games. It made it possible to achieve real-time performance of ray-traced global illumination, thanks to dedicated cores speeding up the ray-tracing process. Global illumination (GI) has been a fundamental aspect of achieving realistic lighting in rendered scenes and game developers are constantly seeking to improve it (Toth 2021). It simulates how lighting interacts with geometry and material surfaces by taking absorption and reflectiveness into account. It still presents some challenges in terms of computational power and efficiency. Unreal Engine 5 currently incorporates this technology, making it accessible for game developers.

2.2 Unreal Engine 5: New Technologies

Unreal Engine 5 is Epic Games' successor to Unreal Engine 4, first released in early access on May 26, 2021. Epic impressed game developers with a new tech demo showcasing the new engine and its features (Makuch 2021). Projects created in the engine tend to look very realistic right out of the box, making the creation of photorealistic environments more accessible to developers of all levels.

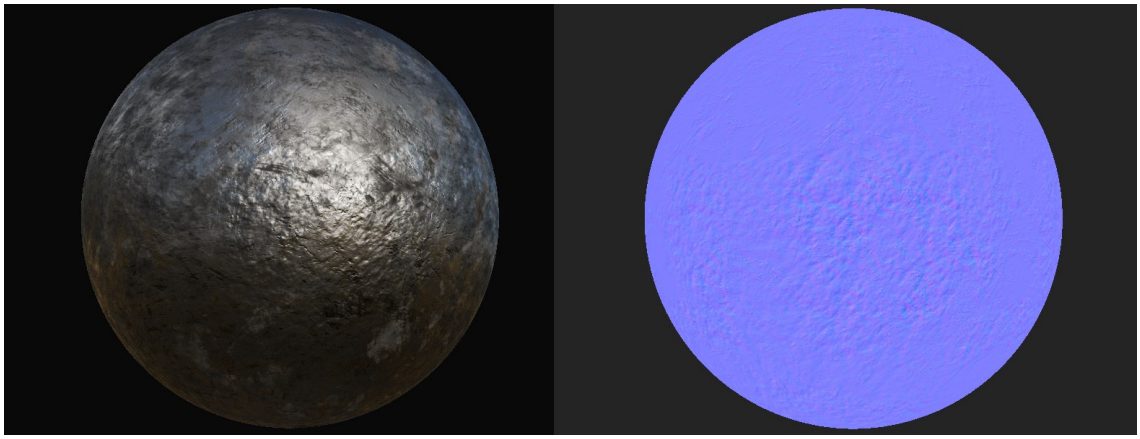
With the release of Unreal Engine 5, Epic Games introduced their own hybrid global illumination solution that provides support for both non-ray tracing and ray tracing graphic cards. The software ray tracing solution uses a Signed Distance Field (SDF) representation of the scene to efficiently trace rays and surface caches to quickly retrieve lighting data. The hardware solution focuses on the intelligent use of ray tracing (Toth 2021). Although NVIDIA's RTX graphics cards can improve performance and visual fidelity in Unreal Engine 5, they are not required for achieving real-time global illumination in the engine. Lumen ensures that games made with Unreal Engine 5 can still achieve realistic lighting on a wider range of systems thanks to its software implementation.

Nanite is another key feature introduced in Unreal Engine 5. It is a virtualized geometry system that uses a new rendering technology and internal mesh format to render high counts of detailed 3D assets. It eliminates the need for creating multiple LOD (Level of Detail) versions of an asset, making it easier to include detailed and complex geometry in a game while maintaining performance. High-poly models, such as photogrammetry scans or ZBrush sculpts, can be directly imported without having to decimate geometry and bake detail into normal map textures. (Unreal Engine n.d). The ability to create environments without needing to spend extra development time on creating LODs for assets streamlines the modeling process. According to Unreal Engine, Nanite should generally remain enabled, as any static mesh with it enabled will typically take up less memory and disk space and render faster. It is worth noting that the usage of many unnecessarily high-poly models can bloat the size of game files, and careful optimization is therefore still needed. As Nanite and Lumen are both still relatively new technologies, they may still have some performance issues.

2.3 Importance of Lighting and Texturing for Realism

A key component in making realistic video game graphics is called PBR, which stands for Physically Based Rendering. PBR materials use unique texture maps that control different properties of the materials and determine how light reacts to them. The shader automatically solves calculations based on the laws of physics (Mesquita 2021). PBR has become a fundamental aspect of modern game development because it provides consistency across different environments and programs. Standardizing the way materials are rendered helps make different 3D platforms more compatible with each other.

Instead of just using one base color texture for a 3D model's material, PBR allows the use of several types of texture maps. These include, but are not limited to, normal, roughness, and metalness maps. The metallic workflow adopted by several game engines uses these three as its main maps, but other maps, such as normal and roughness maps, can also be used (Mesquita 2021). Giving materials many different adjustable properties can make them appear more realistic in a game environment. A metal object that is only using an albedo (base color) map would look flat and not realistic at all. However, adding a metallic map with a high metalness value can effectively simulate the reflective qualities of real metal (picture 1). The degree of realism can become especially high when combined with surface details like scratches or dents included in a normal map.



PICTURE 1. 3D Cube textured using a PBR material. The left image shows the full material and the right image shows only the normal and height channels.

3 DEVELOPMENT PROCESS

3.1 Tools and Software Selection

For this thesis, I will use Unreal Engine 5 as the main application, as well as Blender and Adobe's Substance 3D Painter. It is crucial to understand that Unreal Engine 5 is a video game engine and such, other applications are necessary in creating realistic 3D assets. Although the engine has some simple modeling capabilities, it is almost impossible to create complex models within the engine itself.

For creating the environment's assets, I will be utilizing Blender. It is a free and open-source 3D modeling software tool. It is highly compatible with Unreal Engine 5 and 3D models can easily be exported from it via FBX or USD file formats. For texturing and material creation, I will use Adobe's Substance 3D Painter, a professional tool for creating PBR textures. Textures created with Substance 3D Painter can be exported in a format compatible with Unreal Engine 5's material editor.

These tools together make a suitable pipeline: modeling in Blender, texturing in Substance 3D Painter, and final assembly in Unreal Engine 5. This combination is efficient and makes sure that the final environment is as realistic as is needed for this project.

3.2 Environment Design Process

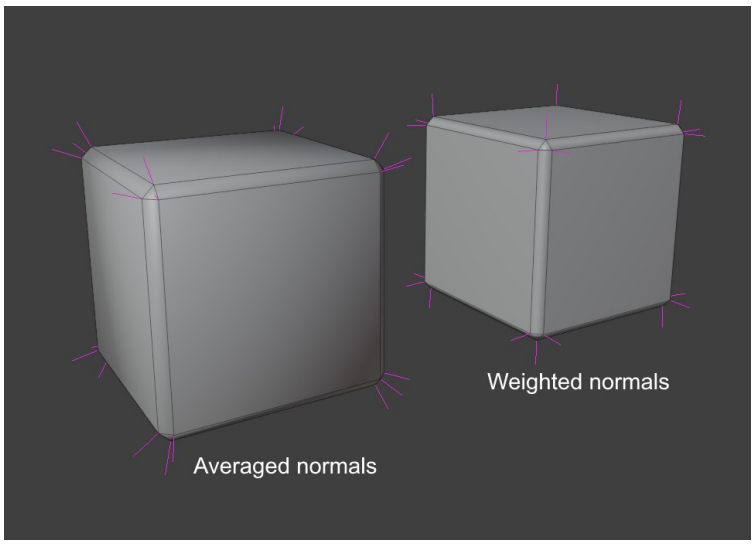
In order to make all the 3D assets look realistic and as if they are within the same environment, it is advisable that artists ensure that there is a constant texel density across all the models in a scene. This is to ensure that the different components of a scene look as if they are part of the same environment and thus create a more realistic scene. (GarageFarm.NET n.d). Having some objects appear noticeably blurrier than others is not good for creating realism.

Texel density is determined by the object's size and its texture resolution. A high-resolution texture increases detail while increasing the size of an object makes it appear blurrier. Texel density is measured in pixels per centimeter, and it can be manually calculated by dividing the texture resolution with the object's size. (Dries 2023). Blender does not come with a tool for checking texel density, but a useful add-on called Texel Density is available for free. Changing texel density later on can be a bit challenging so it is best to determine and set the correct texel density at the start of the environment design process.

3.3 Workflow for 3D Model Creation

The traditional approach to making 3D models is called the high-poly to low-poly workflow. However, another workflow for hard-surface modeling has recently been adopted in several AAA games, called the medium-poly or mid-poly workflow. In this workflow, just one version of the asset is created. Making changes to a baked asset requires a lot of time and effort, so the mid-poly workflow removes a destructive part of the pipeline. There are still many situations where the high-low poly workflow is preferable, such as smaller hero objects, objects with organic shapes, or assets with a lot of material variety. (Patscheider 2020). I won't be making any smaller hero objects or organic shapes for my project, so the medium-poly workflow is suitable for it.

When making 3D objects with the mid-poly workflow in Blender, the Weighted Normal Modifier should be used. Face-weighted vertex normals can make objects look better in terms of shading than objects that use standard smooth shading. Face-weighted normals consider the angle of the adjacent faces which allows for a smoother transition of light across surfaces (Alexandrov 2020). This results in better shading, particularly on hard edges and beveled surfaces, as opposed to the often inconsistent lighting seen with standard smooth shading. Picture 2 shows the difference between averaged and weighted normals on two beveled cubes that are identical in terms of topology.



PICTURE 2. Two beveled 3D cubes using different shading methods in Blender.

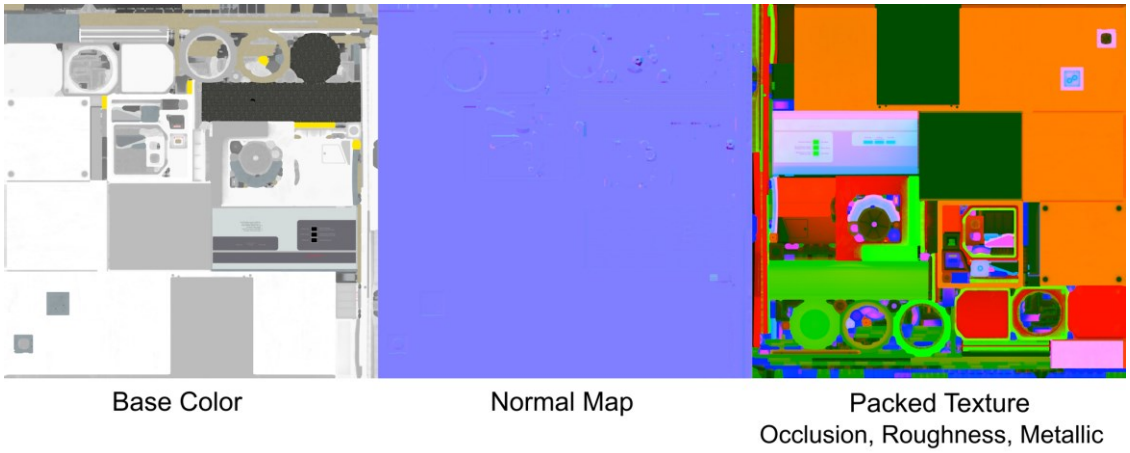
The mid-poly workflow is particularly well-suited for the creation of an interior environment consisting of mostly hard-surface models. I chose to implement this workflow in my project to test its advantages and simplify the modeling process for the interior environment.

UV unwrapping can be started after 3D modeling is finished. First, seams need be marked on the model in Blender. Essentially all sharp edges on the model should be marked as seams. To make sure the UVs are not too stretched, the Display Stretch option should be turned on in Blender. Stretching can generally be fixed by adding more seams. The resulting UV islands can be scaled individually to give specific parts more or less detail. The last step of UV unwrapping is called packing. Although Blender's default UV packing algorithm can be used, it is not very efficient. This is why I opted for a paid Blender addon called UVPackmaster 3, which provides a much more effective way to pack UV islands, minimizing wasted space.

3.4 Texturing and Material Techniques

Channel packing is a handy technique for optimizing PBR textures in Unreal Engine 5, as it allows four distinct textures to be stored in the same file. A maximum of four channels can be present in digital photographs, and each one may have a distinct texture. For instance, alpha can be utilized for a mask texture, blue for

ambient occlusion, green for roughness, and red for metallic. Channel-packed textures should be created in a lossless format like PNG or TIFF. Using only three channels results in one channel having one extra bit because these formats can compress to 16 bits. Since the roughness texture typically contains the most intricate information, this is often the green channel. Packed textures can easily be exported from Substance Painter to Unreal Engine 5 by using the Unreal Engine 4 (Packed) output template, which combines ambient occlusion, roughness, and metallic textures into one (Hof, 2023). The base color and normal map are exported as their own texture files. The textures can then be applied to a material in the Material Graph panel. Channel packing results fewer texture files and in optimized texture memory usage. Having fewer textures helps keep the project files tidy. A packed texture exported from Substance 3D Painter is shown in picture 3.



PICTURE 3. Exported PBR textures from Substance 3D Painter. The packed texture consists of occlusion, roughness, and metallic texture maps.

3.5 Lighting Setup

Within Unreal Engine 5, the available light types include directional lights, point lights, spot lights, rectangular area lights, and sky lights. Blender uses very similar types of lights. Using the right kind of light is important for each lamp object. A fluorescent lamp for example should use a rectangular area light instead of a spot light or point light to make the light spread wider and more realistic. It is important to adjust the color temperature and light intensity settings so that the lighting looks

natural and fits the surrounding environment. A warmer tone could fit in living space, while a colder tone could be more suitable for a more industrial environment.

3.6 Optimization for Performance

One method of optimizing performance in Unreal Engine 5 is called culling, which is enabled by default but can be manually adjusted according to performance needs. It helps reduce the rendering workload by omitting objects that aren't visible or necessary for the player's current viewpoint. It optimizes frame rates and can even reduce power consumption. This is particularly beneficial for games running on mobile or other low-power devices. (McCole 2023). If the player is facing away from an object, there is no real need to render it and it is better to focus rendering power on details in the player's immediate surroundings.

4 PRACTICAL PROJECT

4.1 Goals and Scope

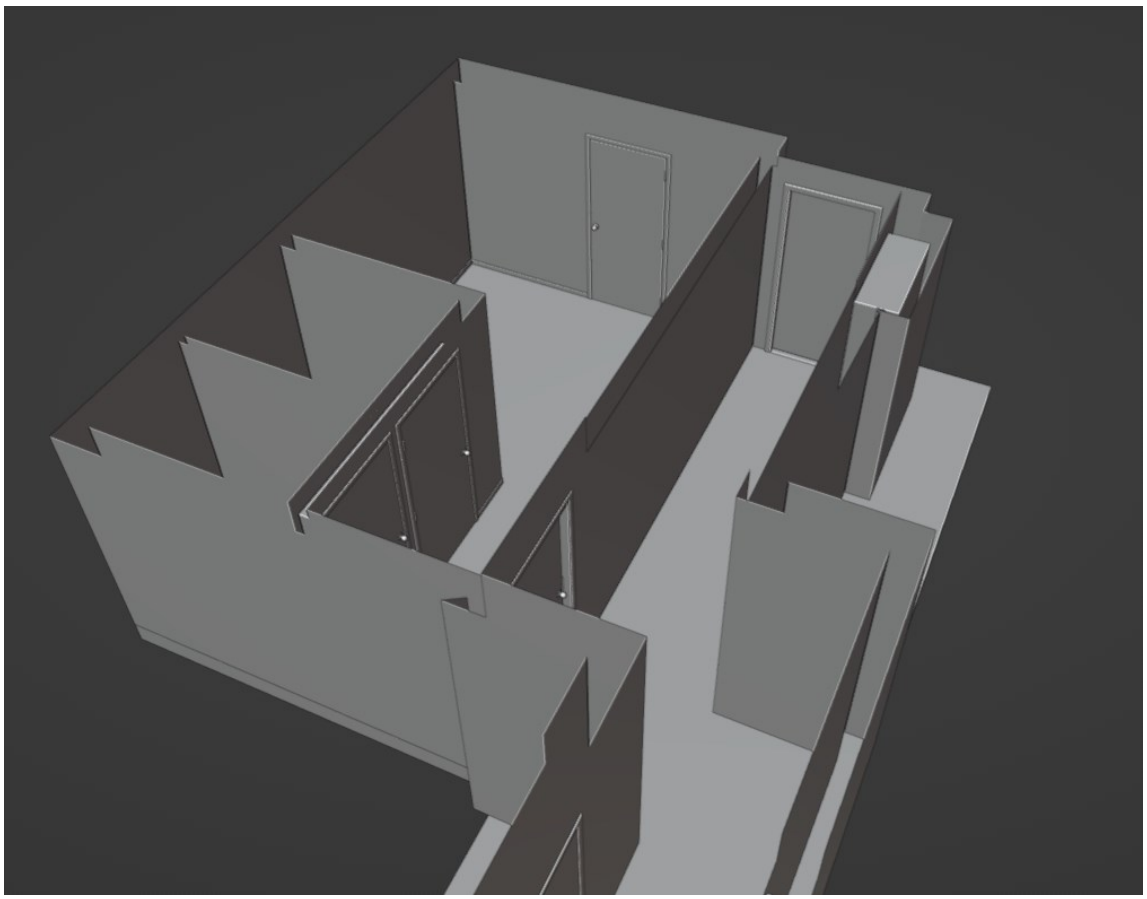
The primary goal of this practical project was to create a realistic interior environment in Unreal Engine 5 by using techniques and technologies discussed previously. I wanted the confined space to be quite small to not make the scope too large.

The interior environment was modeled after the basement floor of an apartment building, which features a hallway and a laundry room. In the modeling process I heavily focused on real-world accuracy. The environment was designed for real-time exploration from a first-person perspective.

4.2 Initial Design and Layout of the Environment

To start the project, some reference photos of hallways and laundry rooms were gathered. I used a free program called PureRef to collect and organize the reference images into one space.

The initial environment layout was created with the help of Blender's Archimesh add-on (picture 4). It is a free add-on that comes with Blender and can be enabled from the add-ons menu. It allows the generation of architectural elements such as ceilings, walls, doors, and windows. It also comes with different types of door handles and door types. Parameters such as wall height, thickness, and baseboard can be easily adjusted. Generated doors automatically have correct pivot points so that they can easily be rotated.



PICTURE 4. Blockout of an apartment building basement floor, with the ceiling removed.

4.3 3D Modeling and Asset Creation

All 3D assets made for this project were modeled in Blender, the modeling program I am most proficient with. Blender has many different addons made by the community that make the hard surface modeling workflow easier and faster. These include HardOps and BoxCutter, for example. HardOps mainly enhances Blender's native boolean operations, while BoxCutter allows the user to slice meshes fast and intuitively. I used both of these add-ons to speed up the 3D modeling.

Some of the props made for the environment include a washing machine, a utility sink, and a laundry basket. These were modeled after some reference photos. All the models were modeled using the medium-poly workflow, as it was found to provide a lot of benefits in flexibility, reusability, iterability, and performance. The

washing machine, utility sink, and laundry basket consisted of 9243, 5025, and 4532 triangles, respectively. I optimized the topologies to the best of my abilities while keeping a fair amount of detail by also modeling smaller parts such as buttons. These triangle counts are neither too low nor too high and are quite optimal for the mid-poly range. The models had no obvious shading artifacts, thanks to the weighted normal modifier and manual adjustment of the normals in Blender.

After the modeling process was finished, parts of the models were separated and then vertex painted to create an ID map. I then applied the triangulate modifier to the models with the keep normals option selected, as I have previously experienced issues with using quad-based models in Substance 3D Painter. The models were then exported from Blender as FBX files and imported into Substance 3D Painter. Mesh maps were first baked for each model, and texture sizes were set to 4096x4096 (4K). By also baking the ID map using the vertex colors applied in Blender, each material could be applied to its own object. A mask with color selection can be applied to any material, making texturing different parts easier.

The models were textured using a combination of base color, roughness, metallic, and normal maps to achieve a realistic look to different materials like plastic and metal. Photoshop was used to create some masks for fine details such as text, which were then applied to the models. I used the Unreal Engine 4 (Packed) output template, as it also works for Unreal Engine 5. As a result, each model got three PNG texture files: a base color, a normal map and a channel-packed texture consisting of occlusion, roughness and metallic maps. Picture 5 shows one of the finished models made for the scene.

PICTURE 5. Finished washing machine model, wireframe version on the left and rendered with textures on the right.

4.4 Integration into Unreal Engine 5

Once the modeling and texturing were done, I exported the scene from Blender as a Universal Scene Description (usd) file. Pixar released the format as open-source software in 2016. It allows for 3D scene data to be exchanged between different tools and software seamlessly (Autodesk n.d). I chose this format due to its versatility and efficiency, as USD files can encapsulate a large amount of information, including 3D geometry, materials, textures, and lighting.

To import a USD file to Unreal Engine 5, the USD Importer plugin needs to first be enabled from the plugins panel. Upon importing the USD to Unreal Engine 5, I noticed that some things, such as area lights, weren't quite imported properly, so I had to recreate them, which was fortunately quite easy. I deleted the imported lights and added in new area lights for lamps such as the fluorescent lights. I had some problems with lights leaking through walls, but I figured out that this could be fixed by simply increasing the thickness of the walls. After doing this, I simply exported a new USD file from Blender and reimported it into the engine.

4.5 Adding interactivity

As I wanted the scene to be explorable from a first-person view, I decided to make the environment interactable. I wanted the player to be able to open doors and turn light switches on and off. I used Unreal Engine's Blueprints to do this, which is a visual scripting system. The node-based interface makes it so that one can easily connect different variables, functions, and events to add interactivity.

Setting up the door interaction system was simple. I first placed a trigger box in front of each door to detect when the player is close. When entered in, a message appears on the screen telling them to press a certain key to open or close the door. The door's movement was controlled using a timeline node that animates the door from its closed to open position. This timeline was connected to a Set Relative Rotation node, which rotates the door in the scene.

I used a similar system for light switches. A trigger box was set next to a switch, and upon interacting, a Blueprint toggle function controlled light visibility and intensity in the scene. I used a timeline node to simulate the brightening and dimming of the lights. By using Blueprints I was able to implement some interactivity into the scene without having to do any programming in C++, the engine's main programming language. Adding interactive features made the environment feel more immersive and interesting while giving the player a feeling of control in the environment.

4.6 Achieving Realism

After walking around the scene myself, I was quite impressed by how realistic it ended up looking. The combination of real-time global illumination, realistic textures, detailed models, and interactivity achieved a result that could be considered realistic. As a final touch, I added some simple post-processing effects to the scene, such as grain, to help hide some of the imperfections.

4.7 Final Result

The result was a playable, realistic interior environment scene that the player could explore and interact with. The interactive features worked as I expected and they had no noticeable problems. Avoiding the classic low poly to high poly workflow and going with the medium poly workflow instead turned out to be a good decision. It not only saved considerable time during the modeling process, but also allowed for more flexibility in making corrections to the 3D assets created without having to bake texture maps all over again.

Nanite enabled the inclusion of highly detailed assets without sacrificing performance, while Lumen's dynamic lighting system contributed to the lifelike quality of the scene by reacting to changes in the environment, such as switching lights on and off in real-time. One of the finished rooms can be seen in picture 6.

The walls and floor in the scene used simple tiling PBR textures while each object used its own PBR texture set exported from Substance 3D Painter. The tiling thankfully ended up not looking too visually repetitive, as the textures used were quite high in resolution and the rooms quite small. The imported objects looked a bit different in Unreal Engine 5 compared to Substance 3D Painter, primarily due to differences in lighting, material settings and the way the engines handle various texture maps. Some minor adjustments to the lighting setup were made to ensure that the objects matched the intended look.

PICTURE 6. Screenshot from the Unreal Engine 5 project showcasing the laundry room area in the scene.

5 DISCUSSION

By using various modeling techniques and technologies, I was able to create a realistic interior environment in Unreal Engine 5. The result may not be considered fully photorealistic by current AAA standards, but I would still say that I met the criteria of realism in a game engine. Realism in the context of video games is about creating believable lighting, textures, and materials while ensuring smooth performance, which I believe was accomplished.

Understanding normals and weighted normals was a significant learning curve. From my experience, there aren't too many beginner-friendly tutorials that go in depth into what normals really are and how they can be properly manipulated in Blender. After many baking errors and much research, I eventually got the hang of them and was able to use things such as weighted normals along the mid-poly workflow. Avoiding the process of creating both a low and high poly version of each model turned out to be very effective in saving time.

As the only desktop computer I had access to had a powerful NVIDIA RTX 3080 graphics card, I wasn't able to properly benchmark the scene's performance across a range of different graphics cards. The frame rate did always stay above 80, so I would say that the various optimizations done did help to at least some degree. As the development of Unreal Engine 5 continues and technology advances further, the environment creation workflows and performance should only improve. Nanite and Lumen still have some quirks in terms of performance, but I believe they will become more powerful and optimized in the future.

This thesis can serve as a practical guide for creating realistic 3D interior environments in Unreal Engine 5. It gives some insight into the environment creation workflow and available tools. I hope it will serve as a valuable resource for 3D artists and developers learning the engine.

REFERENCES

Alexandrov, G. 2020. Weighted Normal | Blender 2.8x Tutorial. YouTube video. Watched on 2.10.2024 https://www.youtube.com/watch?v=sqGFhiP-2mc

Atreya, S. 2022. The evolution of video game graphics and their impact on the industry. From Neuroquantology, Volume 20 No 5. Read on 25.9.2024. https://neuroquantology.com/open-access/THE+EVOLUTION+OF+VIDEO+GAME+GRAPHICS+AND+THEIR+IMPACT+ON+THE+INDUSTRY_11224/

Autodesk n.d. Universal Scene Description: Digital worlds, open for business. Web page. Read on 8.10.2024. https://www.autodesk.com/solutions/universal-scene-description

Dries, T. Texel Density. Beyond Extent. Web page. Read on 3.10.2024. https://www.beyondextent.com/deep-dives/deepdive-texeldensity

Garagefarm.NET n.d. A texel density walkthrough for Blender artists. Web page. Read on 3.10.2024. https://garagefarm.net/blog/a-texel-density-walkthrough-for-blender-artists

Hof, Marteen. 2023. What Is Channel Packing? How To Set It Up In Substance Painter. Web page. Read on 12.10.2024. https://maartenhof.com/what-is-channel-packing-and-how-to-set-it-up-with-substance-painter/

Makuch, M. 2021. Unreal Engine 5 Gets Stunning Demo With Incredible Graphics, Enters Early Access. Web page. Read on 25.9.2024. https://www.gamespot.com/articles/unreal-engine-5-gets-stunning-demo-with-incredible-graphics-enters-early-access/1100-6491998/

McCole, C. 2023. Culling in UE4/UE5 (Precomputed Visibility Volumes, and Cull Distance Volumes). Web page. Read on 21.10.2024. https://www.chrismccole.com/blog/culling-in-ue4ue5

Mesquita, L. 2021. Everything About PBR Textures And A Little More - PART 1. Read on 13.11.2024. https://www.artstation.com/blogs/luismesquita/PwEm/everything-about-pbr-textures-and-a-little-more-part-1

Patscheider, M. 2020. Quarry Bank Mill 3D #08 The mid-poly workflow. Web page. Read on 8.10.2024. https://matthias-patscheider.artstation.com/blog/ZXnP/quarry-bank-mill-3d-08-the-mid-poly-workflow

Toth, B. 2021. The state-of-art of Dynamic Global Illumination in video-games. Read on 21.9.2024. https://enjmin.cnam.fr/medias/fichier/2021-the-state-of-art-of-dynamic-global-illumination-in-video-games_1647857353552-pdf

Unreal Engine n.d. Nanite Virtualized Geometry. Web page. Read on 1.10.2024. https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine