ARCHITECTURAL VISUALIZATION IN THE UNREAL ENGINE 4

GRADUATION REPORT

THE VIRTUAL DUTCH MEN

ALEXANDER ELZINGA
333979

22 JANUARY 2019
ENSCHEDE

SAXION UNIVERSITY OF APPLIED SCIENCES

SUMMARY

The client of this project, The Virtual Dutch Men, is a company specialized in architectural visualization. The client asked the author to do research on the Unreal Engine and develop a workflow that would fit into their current pipeline.

The company would like to know how a switch to the Unreal Engine would benefit them from a business' perspective, with the quality of work, production costs and training time in mind. Because making a switch to the Unreal Engine and training people to use it costs time which is otherwise used on their clients' work, they would like the author to take on the research and develop a workflow they can use in their current pipeline.

Based on the problem and goal of the client the following research question has been formulated.

> How does the process of constructing a photorealistic architectural visualization scene in the Unreal Engine compare to the current workflow of The Virtual Dutch Men in Unity with respect to time, quality, costs and reusability?

To help guide the research and approach, sub-questions must be answered first to find the answer to the main question. The author established a workflow that connects to the current pipeline of the client. The tools provided by the Unreal Engine proved to speed up the production time and therefore reduce the cost of creating a scene. The result of the project shows the quality of lighting and shadows in Unreal. The delivered project and documentation serve as a guide to start working with the Unreal Engine.

PREFACE

During the development of this project, I was guided by Erik Mostert and Richard de Paauw. I would like to thank both for their input and guidance within the company environment. Richard was our technical and creative advisor and his feedback and enthusiasm helped me find new techniques and achieve better results. Erik was my internship supervisor and helped me develop a better planning structure and become a better project leader.

As project lead, I would like to thank Kristiyana Georgieva and Keimpe Snip for their contribution to this project. They have helped me realise the result of this project with their modelling and material creation.

I would like to say thank you to Herman Statius Muller, who guided me through this project. His feedback and input helped me raise the quality of research, self-reflection and personal growth.

I would also like to thank The Virtual Dutch Men for providing a space and time for the team to work on a game project that I was programming lead of. This project was carried over from the previous study year and could continue because The Virtual Dutch Men provided a whole room in the office for the development team to work from. Because of the continuation of this project I was able to go to Geneva and attend the was the first-ever global event to focus on both air pollution and health to promote our game project.

At last, I would like to thank all other employees of The Virtual Dutch Men that helped me by answering questions, delivering content, providing tips and giving feedback. They made my time at the company a great experience.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

The Virtual Dutch Men is a company that has a large portfolio in making architectural visualizations. Architectural visualization is often abbreviated to Arch Viz. In Arch Viz you visualize a designed interior or exterior with 3D models in a photorealistic way. The Virtual Dutch Men uses several programs that help accomplish the final look of a designed scene. Depending on the type of visualization, the most suited programs are chosen to achieve the best result.

Static renders are images captures from one camera angle, these images can be made with very precise calculations on lights and shadows. The program The Virtual Dutch Men uses for this is V-Ray. For demo projects where customers can walk through the whole scene, the lighting of the scene needs to be real-time or baked. When the lights are baked all lighting data is stored in lightmaps. Right now, the company uses the Unity Engine to build their demos.

The company has been impressed by presentations and videos of people and companies using the Unreal Engine for Arch Viz scenes with real-time interactions. The visual impression of these scenes has made them curious about the possibilities in the engine. However, since they have an established workflow and lots of experience in the Unity Engine, they have not fully explored the possibility of making a switch. They have asked the author to do research on the Unreal Engine 4 (UE4) by making an architectural visualization scene in the engine.

Virtual Reality (VR) is also one of the main reasons to use a game engine. In game engines, lots of functionality can be built into the scene. The current demos of The Virtual Dutch Men run inside the Unity Engine. To run a project in VR, the scene needs to be optimized to run with a high count of frames per seconds (FPS). A minimum of 90 FPS is needed to make the image appear smoothly inside the VR headset.

The results of this research will be used by the company to decide whether they want to use the Unreal Engine in their pipeline. These results should show improvement in different areas in their current workflow pipeline. During the development of this project, the author will be maintaining a blog in which he reports his research and development steps. This blog can be used as a guide to start with Arch Viz in the UE4.

1.1 REASON

The client

The client, The Virtual Dutch Men (TVDM), is a company based in Almelo, The Netherlands. They have been doing visualization work for over twenty years now. Their work is focused on visualizations of the exterior and interior of buildings and houses. They also have a team that is focused on developing virtual and augmented reality experiences. TVDM have been industry leaders in the field of visualizations for over two decades. The company has grown quite a lot during that time as their development team now consists of fifteen people.

Figure 1.1 (The Virtual Dutch Men, n.d.)

"The roots of The Virtual Dutch Men go back twenty years and started with the 3D visualization of real estate objects. We bring projects and objects to life, make them tangible and make them speak to the imagination. Our core values, passion, quality, innovativeness, and sobriety have remained the same. Clients and employees that have been with us from the start lay the foundation for the growth of our company." (The Virtual Dutch Men (n.d.)).

Currently, the company is using the Unity Engine for their real-time demos and V-Ray for static image renders. Static image renders are often edited in image editing software like Adobe Photoshop after rendering in V-Ray to add extra detail. They have been interested in the Unreal Engine for a long time. The Unreal Engine is, as the Unity Engine, a game engine which can be used to build interactive scenes. Inside both engines, you can use VR to accomplish real-time interactable demos controllable through a VR headset and input controllers. Changing their current workflow pipeline from the Unity Engine to the Unreal Engine costs lots of time, training and knowledge. Because they are also busy with running projects for clients, they had decided to stick with the Unity Engine for now.

On conferences they have been to, they have been impressed by what other people accomplished using the Unreal Engine. Because their products need to have photorealistic looks, they would like to see what the Unreal Engine can offer them. They want the author to do research on the benefits of the Unreal Engine by making an architectural visualization scene.

The goal

The goal of the company is to find out if a switch to the Unreal Engine will benefit their Arch Viz work compared to the Unity Engine. They would like to see if this improves the quality of work and speeds up their current workflow process. If it reduces production time and maintains the high-quality results, it will reduce the development costs. With the author's findings, they have a guide and base to work from and therefore reduce training time.

The first step when they would be changing their pipeline is learning more about the start-up process and best practices of the engine. For this part, the author will be providing a development blog where the company can check on his research and explanation of features and functionality. This blog can be used as a guide into the engine and Arch Viz possibilities.

TVDM has a very high-quality standard which the project should meet or surpass. The time spent, and quality of the results must show a workflow or quality improvement. If the quality meets the company's standard, they can think about replacing a part of the pipeline where rendering is costing lots of time. Inside the engine, they can then make a scene composition and try out different iterations of positioning and lighting. When doing real-time demos, the project must run smoothly on the company's hardware.

What they would like to see produced is a demo showing the possibilities when building Arch Viz scenes in the Unreal Engine. A demo where functionality is built in which the client can use to show off their work or capture high-resolution images for professional use. Examples of functionality that they would like to see added are an animated camera, high-resolution screenshots and a controllable first-person character. These will be used for showcasing demos in real-time and the capturing of images which can be used for editing after.

1.2 PRELIMINARY DEFINITION OF THE PROBLEM

TVDM would like to see their models and scene inside the Unreal Engine. They believe the Unreal Engine is better than Unity and therefore will show a significantly better image quality. However, this is not exactly what the company wants the author to research. From the business' perspective of view, a change from Unity to the Unreal Engine will bring a lot of changes to the current workflow pipeline. Their actual question is to see the benefits of the Unreal Engine and its workflow compared to the currently established one.

To answer this question, the author needs to have a look at both engines' rendering system as well as the established workflow and programs used within the company. Since the company is also doing static image rendering in V-Ray, a lot of their models are not suitable for use inside a dynamic scene in either Unity or the Unreal Engine.

The client would like to know how a switch to the Unreal Engine would benefit them from a business' perspective, with the quality of work, production costs and training time in mind. Making a switch to the Unreal Engine and training people to use it costs time which is otherwise used on their clients' work. Therefore, they would like the author to take on the research and develop a workflow they can use based on their current pipeline.

1.3 GRADUATION SCHEDULE

The first goal of the research project will be to create a scene using only static lighting. What this means is that all lights need to be baked into lightmaps. The final goal is to be able to interact with objects in the room. One of these interactions is the possibility to turn lights on and off. With these types of interaction, the scene will have dynamic lighting. Because dynamic lighting and shadows are calculated in real-time, optimizations and testing will be done to keep the scene run smoothly.

To see whether it is possible to achieve what the team wants, the author must learn what the engine can do first. The research will focus on the rendering system and lighting possibilities. Since each engine has limitations when it comes to lighting, the author must figure out what these are and how to deal with those limitations in a specific situation.

In the approach chapter of this thesis, the author explains the steps that he performed during the research. A reflection on these steps can be found in the method chapter, followed by results.

Chapter 2

THEORY

2.1 THEORETICAL FRAMEWORK

In this chapter, the theoretical research of the thesis will be explained. Knowledge needed to understand the next chapters of the thesis can be found here.

The Unreal Engine 4

The Unreal Engine 4 (UE4) is a game engine, which is a complete suite of game development tools. Some of its components are the rendering engine, physics engine, sound engine, animation, networking and scripting. The complete engine features many more components which are ready to be used inside the development environment. The Unreal Engine version used by the author in this project is 4.20. During the development of this project, Unreal 4.21 was released but the author stuck to the version used from the start. Because a game engine provides all components and tools, content creators do not need to worry about programming an engine from themselves.

Figure 2.1 (Epic Games, n.d.)

In the UE4 a level can be created in which models can be dragged and placed as actors. For consistency in the report, the level will be addressed as the scene, and actors will be called objects.

Light types

Inside the UE4 there are five light types that can be placed. A list of them can be found in figure 2.2. Each type of lights emits light in a different way.

The Directional Light is a light that is cast from a source which travels over an infinite amount of distance. Shadows and light from the directional light is parallel. Sunlight is often represented by a directional light in the scene.

The Point Light emits light in every direction surrounding itself. This lamp is great for simulating lamp bulbs. With its properties, this light can also be changed to emit from a different shape.

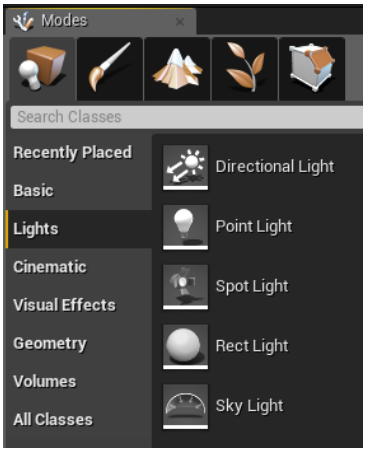


Figure 2.2

The Spot Light is emitted from a single point and travels in a cone. Properties for this light can be changed to increase or decrease the intensity, distance, spread and fall off.

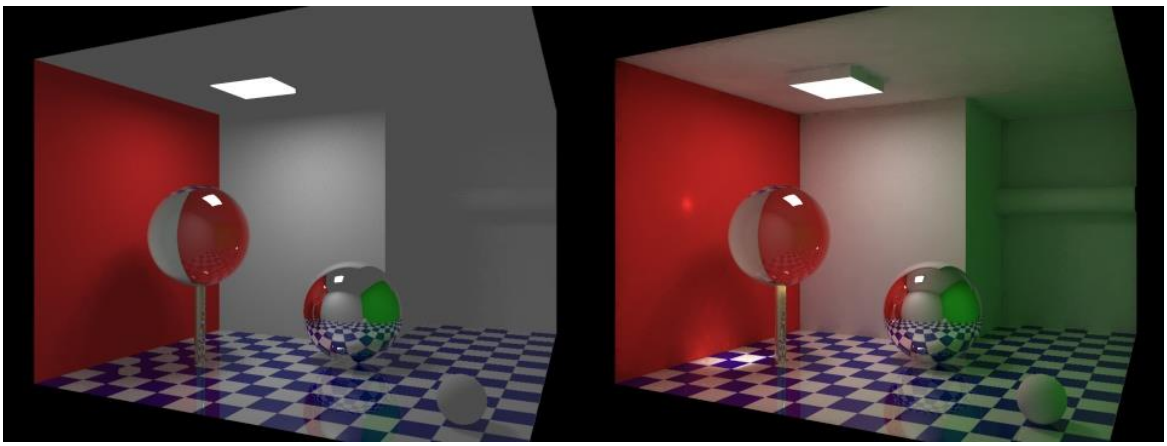
The Rect Light emits light into the scene from a rectangular plane with a defined width and height. You can use them to simulate any kind of light sources that have rectangular areas, such as televisions or monitor screens, overhead lighting fixtures, or wall sconces (Epic Games, n.d.).

The Sky Light applies light to the whole scene by capturing the distant parts in a scene and reflecting those on the objects in a scene. Another way to light your scene using this light type is by replacing the scene capture with a skybox texture. This texture or image will then be projected on the objects.

Global Illumination

Global illumination shows a realistic result of all lights and colours in the scene being bounced around as indirect lighting. Direct lighting hitting the surface of a model will bounce around and reflect the colour of the surface in its light bounces on to other surfaces or models. This lighting is also known as indirect lighting. In figure 2.3 results for global illumination can be seen.

“The global illumination algorithm based on photon maps is a two-pass method. The first pass builds the photon map by emitting photons from the light sources into the scene and storing them in a photon map when they hit non-specular objects. The second pass, the rendering pass, uses statistical techniques on the photon map to extract information about incoming flux and reflected radiance at any point in the scene. The photon map is decoupled from the geometric representation of the scene. This is a key feature of the algorithm, making it capable of simulating global illumination in complex scenes containing millions of triangles, instanced geometry, and complex procedurally defined objects. (Jensen et al., 2001)



Local illumination Global illumination

Figure 2.3 (Tanski, G., 2004, December 22)

Lightmass

The UE4 features Lightmass to get the best pre-calculated indirect lighting and a global illumination workflow. Lightmass can use photons from the Photon Mapping technique to backtrack to Directional, Point and Spot light sources.

The Lightmass has several objects that can be added to the scene to help with the results of the lighting calculations. One of these is the Lightmass Portal. When Lightmass is building the lighting data, the Lightmass Portals tell Lightmass that more light rays should come from the area of the portal, yielding higher quality light and shadows (Epic Games, n.d.). These portals are often placed on areas where the light comes through like windows, roof or wall openings.

Lightmass Importance Volumes are used to indicate parts of the level which have a high priority on light calculation or need more detail. In these areas, the engine will emit more photons and output more lighting data. Within these volumes, the engine will also create Volumetric Lightmap Samples.

The Volumetric Lightmap is a system the UE4 has, to share indirect lighting data to dynamic objects. When the static lighting of a scene is built, the engine stores precomputed lighting data in sample points throughout the scene. These sample points are generated inside of the Lightmass Importance Volume in cells of 4x4x4. In Figure 2.4 the sample points can be seen spread out through the volume with more samples around geometry. Higher density around geometry will result in better indirect lighting on objects. The density of these points can be changed for more accurate interpolation between data points.

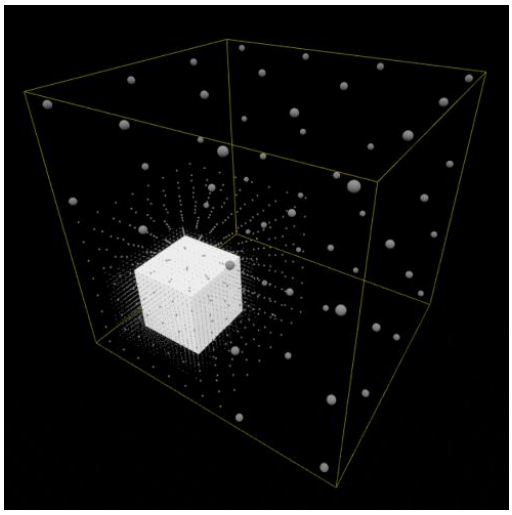


Figure 2.4 (Epic Games, n.d.)

In appendix C – V and C – VI the setup of the author's Lightmass Importance Volume and Volumetric Lightmap can be seen. To gain detail the author chose for a high density of Volumetric Lightmap samples.

Lightmass creates lightmaps with complex light interactions like area shadowing and diffuse interreflection. It is used to precompute portions of the lighting contribution of lights with stationary and static mobility (Epic Games, n.d.). Without indirect lighting, the scene would look black in parts where no lights hit the surface because only direct lighting would be shown. Diffuse interreflection makes sure lights that hit the surface are being bounced in many angles. Therefore, not only opposites of the projection view are calculated. This is why light bakes in the UE4 are processed for the whole scene and stored in lightmaps. Because the lights are being calculated for the total scene and stored in lightmaps, the light bake times increase when calculations are done with more precision and higher detail levels.

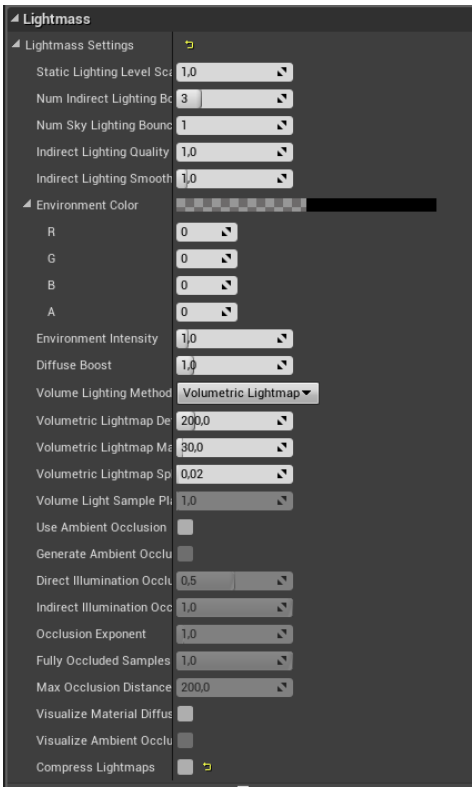


Figure 2.5

Figure 2.5 shows the settings available in the Lightmass. The top five settings are the most important for static lighting. Static lighting level scale will set the scale in which lighting is calculated. The lower this value is, the more detail is shown in lighting and shadows. This setting will, however, generate more noise and artifacts which will be addressed by the indirect lighting quality setting. The quality setting will increase the number of rays used for the final light gathering. The number of indirect bounces will increase the brightness of the scene by bouncing the lights multiple times. As shown in appendix E – II the light bounces increase the area which the lights can reach. Table 2.6 shows the influence the most important Lightmass setting will have.

To add ambient occlusion the box for ambient occlusion should be checked, together with the option to generate ambient occlusion for static lighting. To get the best results lightmaps should not be compressed.

Setting	**Influence on**	**Influence on build time**
Static lighting level scale	Details of Lightmass (Irradiance caching recording radius)	Medium
Num indirect lighting bounces	Brightness of scene (Number of photon reflections)	Low
Indirect lighting quality	Decrease artifact noise (Number of rays for final gathering)	High
Indirect lighting smoothness	Blurs indirect light component in the lightmap (Irradiance caching interpolation ratio)	Low

Table 2.6 (Platteaux, J., 2017, October 9)

Table 2.7 shows a formula is based on the effect of each Lightmass setting. Since going low on the level scale will result in more noise and detail. The quality setting will make sure to get rid of the introduced artifacts. With the lighting quality set to production, artifacts can be reduced to an acceptable level when the multiplication of the static lighting level scales and indirect lighting quality results in 1.0.

Static lighting level scale	Indirect lighting quality
0.1	10
0.2	5
0.25	4
0.5	2
1	1

Table 2.7

Dynamic and Static

In this part of the chapter, objects will be split into two types: Static Mesh Actors and Light Actors. All the light types mentioned above are light actors and can be added to the scene to contribute to the lighting. Static Mesh Actors are objects that hold a mesh, these objects will interact with the lighting and contribute shadows.

Objects in the UE4 have a mobility setting which controls whether an object can be changed or moved during gameplay. The three mobility setting options are static, stationary and movable. These will be explained in the following paragraphs.

Static objects have their mobility set to static and cannot change position during runtime. Light actors with static mobility use lightmaps and bake their shadows before runtime. Static light actors will only interact with objects that are static and will not cast lights or shadows during runtime. Static light is cheap on runtime because all lighting data is pre-calculated and stored into lightmaps using Lightmass. The UE4 uses Lightmass to do calculations on complex light interactions. Static objects in the form of mesh actors are great for objects in the scene that should or cannot move. Lighting for these types of objects can be pre-calculated and stored in lightmaps.

Dynamic objects have their mobility setting set to moveable. Light actors with this mobility will not contribute to lightmaps. This means that all lighting and shadows in your scene from dynamic light actors are calculated in real-time. The calculations for this type of are expensive for the overall performance. Because these lights are expensive, the UE4 has a restriction to four dynamic lights overlapping. When more than four lights interact with each other, shadow data of some lights will not be processed and lost on the final image. Static Mesh Actors that are dynamic can only be affected by stationary or moveable light actors. Their light and shadow data will not contribute to indirect lighting and lightmaps.

Stationary objects are different from static and dynamic objects because they have extra functionality. Stationary light actors will stay in one position but can have some properties changed during runtime. These lights will contribute to the pre-calculated lightmaps but can also cast dynamic shadows and lights on moveable objects. Therefore, the same lighting restriction applies for a maximum of four overlapping lights. On runtime, values like that change the direct lighting can be changed. Pre-calculated indirect lighting is stored and can therefore not be changed on runtime. Stationary static mesh actors do not contribute towards the pre-calculated lightmaps by Lightmass and are therefore lit like moveable objects. However, they will use cached shadow maps when lit by moveable lights when the light is not moving. This will increase the performance of the dynamic lighting.

Depending on the lighting and interactivity needed in a scene, the object mobility and lighting workflow should be chosen.

Unreal Swarm

Unreal Swarm is a tool that works with Lightmass. Depending on the size of projects and scenes, light baking can take lots of time. Light baking times can be significantly reduced by setting up the Unreal Swarm over the network. The Unreal Swarm consists of two applications that can be run. The first one is the Unreal Swarm Agent. This application takes on the job to calculate the lighting based on the geometry and lights of the scene.

The second application is the Unreal Swarm Coordinator. This application can be set up to distribute the workload of light baking to multiple connected swarm agents over the network. Each connected and available agent will get assigned to a task to contribute with computing power. When all jobs are finished, the lighting data is mapped into textures and imported back into the engine.

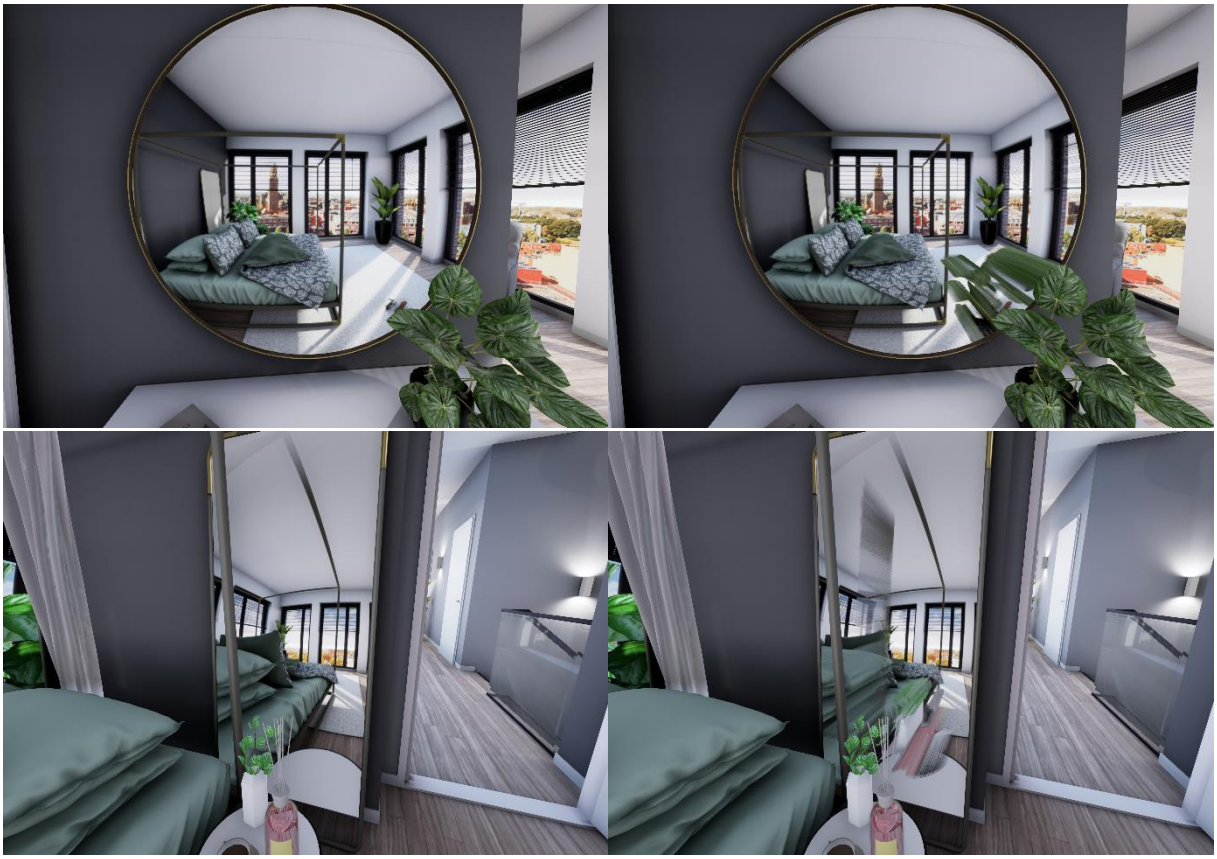
Reflections

Objects in the scene that hold reflective materials will have their surface be affected by reflection data. Reflections in the UE4 can be done using a couple of different solutions. Divided into real-time and static reflection captures they will be described.

For static reflection captures there are three options: sphere reflection captures, box reflection captures and camera capture to texture. First, the sphere and box reflection captures will be described, the camera capture to texture will be addressed in a separate paragraph because that option can also be done in real-time. The shape of our reflection capture objects is important because they decide the part of the scene that is captured and how it is projected on the affected objects inside the area of influence. These reflection captures will only capture static objects in the scene and are not updated during runtime.

Real-time reflections in the UE4 can be done using a planar reflection, screen space reflections or camera capture to texture. Because these reflections are done in real-time, the performance will be impacted. Planar reflections are the way Unreal fixes real-time reflections and positioning of the camera. At the point of the reflection, the level is rendered again and projected back from the point of impact.

Screen space reflections are calculated and projected on objects in the camera view which have reflective surfaces. This method will only project scene geometry which is seen on screen, therefore it will cut off reflections or try to fill up reflections which results in ugly lines (figures 2.8). An improvement for the screen space reflections would-be real-time raytracing, this a feature that is coming with the newer generation of graphics cards.



Figures 2.8 Screen space reflections OFF Screen space reflections ON

A final option for both static and dynamic is the camera capture to texture. What this option does is capture the scene from a second camera and project this onto a texture, which is then displayed on the surface of an object. This will also render the scene twice and thus have a higher rendering cost. Translating to the right position has to be done manually by calculating the original camera position towards the reflection. The static solution for this would be to capture the scene from the second camera's angle. This can be recaptured on runtime if needed but will only map the texture from this one specific angle. Resulting in a weird result when moving around the reflective surface because it does not update.

2.2 DEFINITION OF THE PROBLEM

The client would like to know how a switch to the Unreal Engine would benefit them from a business' perspective, with the quality of work, production costs and training time in mind. Because making a switch to the Unreal Engine and training people to use it costs time which is otherwise used on their clients' work, they would like the author to take on the research and develop a workflow they can use based in their current pipeline.

Condition of satisfaction

The client would like to see a working demo showing the best visual representation of their scene. They are satisfied when all techniques are applied and explained so they can use them again for later projects. A comparison between the current workflow and the workflow in Unreal should show a positive change in production time. Also, the quality of the produced scene should meet or surpass the high-quality standard of the company.

2.3 MAIN AND SUB QUESTIONS

Based on the problem and goal of the client the following research question has been formulated.

How does the process of constructing a photorealistic Arch Viz scene in the Unreal Engine compare to the current workflow of TVDM in Unity with respect to time, quality, costs and reusability?

To find the answer to the main question there are sub-questions to help guide the research and approach.

- What are the differences and limitations of static and dynamic lighting?
- How is the scene set up for the designed lighting solution to get the best possible quality of realism?
- How can the models, scene and lighting be optimized to maintain good performance in virtual reality?
- How are functionality and techniques migrated from one project to another in the Unreal Engine to speed up the development process?
- How does the company's current pipeline connect to working with the Unreal Engine?

2.4 SCOPE

Since this project is quite broad and the author has a technical background he will be assisted by artists who provide and fix the models for his scene. These artists work under the author's supervision, he will lead and plan the project. Decisions and adjustments will be made based on meetings and feedback of the client. In respect to time and cost, the artists will need to keep track of their workflow. The results will also be provided as knowledge and feedback to the client for when the client wants to work with the Unreal Engine.

The focus of the product will be on the visual quality and workflow in the engine. A blog will be kept showing all the steps the author took to accomplish the result. Created materials and shaders will be explained by how the author has made them and how they work. These will not be the attention of this research since there are other, more specialized, programs where you can create and export these. Because the goal is to show off the best image and lighting possibilities, optimizations for performance will be made in the later stage of development. The first goal of this project is to show the best possible lit scene with static lighting. When the client is satisfied with the first scene, the author will start implementing interaction into the scene. When interaction with lights is added, the scene will have dynamic lighting.

2.5 APPROACH

The project will start by analysing the question of the client. From their question, a problem statement can be derived with a couple of research questions. With the existing knowledge, a plan can be made to do research on converting the currently existing solutions to the new engine. With known knowledge of the Unity Engine and the techniques learned by the current study, the author has made his first design of the concept.

To get more knowledge on the possibilities for static and dynamic lighting in the Unreal Engine the author will do theoretical research by looking through the documentation of the engine and lighting techniques. From there a practical approach can be made by converting a concept scene into a prototype. A first design and implementation must be made, giving the first insight into the limitations and errors. These first tests will provide enough knowledge and feedback to start iterating and improve the scene. From these tests, the gained knowledge will be documented along with the results.

Achieving realism in the demo scene is one of the goals in Arch Viz. The designed prototype should show the designed realism of the concept. Based on feedback provided by the client and experts in the field of Arch Viz adjustments will be made to improve the feeling of realism in the scene. Light shapes and shadows need to be achieved in the best possible way. The research will be done to get the best quality of lighting and shadows inside the engine.

Real-time demos in game engines need to run smoothly and are therefore limited by the hardware's performance. The provided models that are often used in the client's visualization projects are not suited for use in a game engine. These models need to be optimized. By iterating through different steps, the impact on the performance can be reduced while maintaining the high-quality look.

Models and functionality developed in a company are going to be reused for many projects. Migrating them from one project to another should provide a basis of a workflow. This process will save time because work that has already been done does not need to be replicated. By making sure the functionality and models are well designed, money and development time is saved.

With these approach steps, the team is going through the process of each of the steps in the framework. Starting with practical analysis, followed by research. From this research comes a first design which will be translated into a concept or prototype. This prototype will provide a base to iterate upon going through a loop which will expand the practical and theoretical knowledge of the project.

Chapter 3

RESEARCH

Following the approach set up at the start of the project, several methods were used to acquire knowledge through practical and theoretical research.

3.1 METHOD

One of the methods to get to a goal is to conceptualize ideas. During this project, the team had made several prototypes and iterations of concept. The first prototype was designed from a small part of the total scene. TVDM had a render of the scene the author and his team were going to recreate (appendix C – I). This scene could also be viewed in a 360-photo tour, this is where the author looked at how the scene was set up and the final lighting was added.

Workflow

TVDM has an established workflow as shown in figure 3.1. The author will be looking at the production part of the current pipeline. He will compare this workflow to the new workflow in the UE4 and research how the new workflow can be integrated.

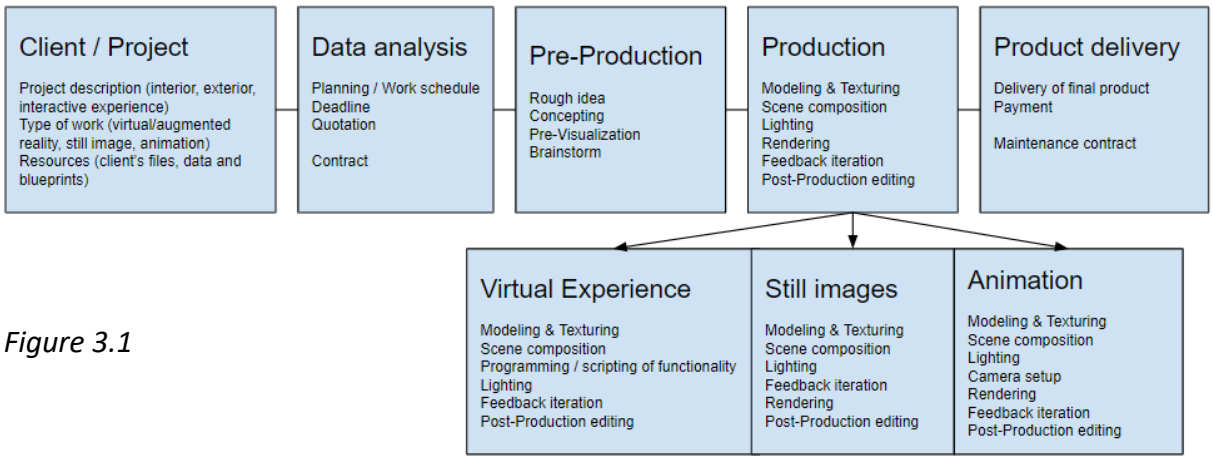


Figure 3.1

At the start of the assignment, the author was given the project description. From this description, the first concept and ideas were visualized with the client. He then proceeded to share this information with his team. Setting up the planning for the first concept demonstration was done by assigning roles to his team members. The author took on the role of project lead, lighting artist and functionality programmer. He was also responsible for the communication between the team and the client. The feedback from the client was written down and distributed to the person responsible for the task.

Translating the idea from a concept to a scene in the UE4 required research on: the provided 3D models, importing these models in the engine, materials, textures, lighting and engine settings. A theoretical scan was done to get started with the Arch Viz workflow in the engine. With the knowledge gained by the theoretical research, first tests were performed.

Test environment

A separate scene was created with a lighting setup to test the 3D models. Inside this test environment, models were placed individually and tested on the quality of lighting and shadow. Using the test environment workflow, the author tested each model to find errors and good lightmap sizes.

The author performed tests with different lightmap resolutions for each model. When the lighting showed errors or artifacts on the models it could be caused by several issues. Artifacts are errors as shown in figure 3.2a. Some of these errors can be caused by overlapping UV maps, which is the process of projecting a 2D image to a 3D model's surface. Another issue that caused errors which were encountered a couple of times was the complexity of the 3D models.

The bed sheet in figure 3.2a is a complex model with lots of polygons. The way the author fixed the model as shown in figure 3.2b was by assigning one of his modellers to adjust the UV map of the model so that it could be used by the UE4. The UE4 has an algorithm that generates lightmaps from an existing UV map. But because of complexity of the model, the lightmap that was generated had a ton of small pieces. A lot of information and space was lost because of this. By dissecting the model into separated parts and laying them out nicely over the whole area of the UV map, the algorithm of the lightmap generator could produce a better result.

Figure 3.2a

Figure 3.2b

Lightmap resolution

In the UE4 when you press Alt + 0 you get a nice overview of the lightmap density on objects in your scene. In this overview, the lightmap density colour shows whether the resolution is suitable for the selected objects. When the colour is blue it means the lightmap is too small and therefore the object loses detail in lighting and shadows. When the lightmap density shows a red colour, the resolution might be too high. A higher resolution lightmap increases the time for lighting to be built (appendix E – I).

Figure 3.3 shows an overview of the transition of colours. Epic Games' UE4 is an engine mainly used for games, that is why they recommend using a green lightmap density for objects. However, for architectural visualization we want objects to look the best they can. Therefore, we will allow red density during our final decision on some important objects that will need higher resolution lightmaps.

In appendix C – II an image was made by the author to show off the overview. In the image, some objects are shown with a blue resolution colour to demonstrate the different resolutions. This view is used to define objects that have a low, medium or high setting for the lightmaps. For Arch Viz you can go higher than usual to achieve better results. This means you can go past the green colour.

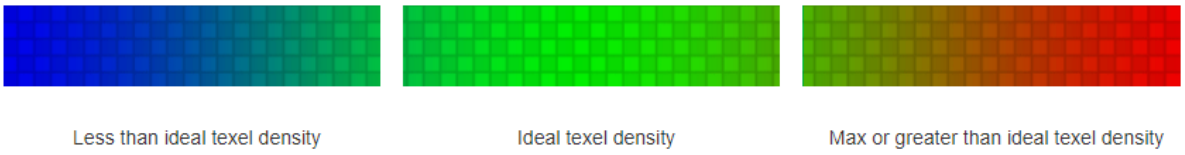


Figure 3.3 (Epic Games, n.d.)

There are multiple ways to change lightmap resolutions on objects in your scene. The simplest way of changing or overriding the lightmap resolution on a specific object is by editing the lightmap resolution on a selected object in your level. Firstly, you select the object in your level. Then in the details panel, go to Lighting -> Overridden Light Map Res. Mark the checkbox and enter a value for the resolution to be set.

A second option is to change the resolution on the model itself through the static mesh editor. To do this, you open the static mesh editor. This is opened by double-clicking the model's mesh in your content browser, or by opening the static mesh on an object in the world outline. By changing the Minimum Lightmap Resolution, the space between each UV island (padding) can be scaled together with the total texture space. By keeping this value the same as the actual lightmap sizes used on our assets, the padding distance will be better. The higher resolution and padding results in better quality calculated shadows and lighting data. Inside this editor, an option to let the Unreal Engine generate the lightmaps is available.

The author found two ways to edit multiple objects at once. The first way is to use method one but with more objects selected. The second way is to select multiple objects in your content browser and bulk edit them via the property matrix found under the asset actions window.

Scene realism

From the original scene and models, iterations were made to get the perfect lighting position. To answer the sub-question on how to set up the scene to get the best possible quality of realism, the author looked at some main focusses in the original render. The height of the scene, sun position and sky had to be considered to reflect the correct lighting direction.

To make iterations on the lighting positions, the author needed to rebuild the lighting of the scene to view the results. To speed up the process of building the lights for these iterations, the quality settings of the lighting were kept low. In a later section of this chapter, the Unreal Swarm topic will be explained and how it improved the workflow speed.

Materials and textures make your objects look realistic. As explained in the theoretical research, materials will interact with the lighting. For each object in our scene, we created a different material that would hold the texture, roughness and shininess of the surface. The team created a workflow for using materials. This workflow would consist of making a base material for each specific surface material, called the master material. These master materials were set up with all values and calculation. From this material, the team created instanced materials which act as the same material. The only difference is that instanced materials can only change exposed values from the master material. Examples of these are the base texture or colour, shininess and roughness. When the master material gets updated, all instanced materials also get this update. This will improve the speed of changing and updating all materials at once.

Lighting setup

The workflow of setting up the lighting came forth from understanding the theoretical research on indirect lighting and the Unreal Lightmass. After the models were imported into the scene, a first test could be done by positioning the directional light. A real-time preview will show the direction and positioning of the lighting. However, because the lighting is going to be static, only a preview of the lights can be seen until the lightning has been built into lightmaps. When the lights are built a first iteration is made. The results may vary depending on which Lightmass settings are used. As explained before, building the lighting takes time. The higher quality settings are used, the longer this time will be. The overall quality settings: preview, medium, high and production have a significant impact on the total build time. But they also improve the final result with lots of extra details. For final renders and demonstrations, a production build is needed.

After the directional light was positioned, a skylight was added to the scene. This light is used to fill up the dark areas which the bounced directional light did not fully reach. From these two lights, a base should be created on which fill or interior lights are added. A method learned from the client to get softer shadows around the windows and wall was the use of spotlights bouncing back inside. Spotlights were placed on the outside of the building, shining away from the scene with a big plane in front of them. As they bounce back from this plane, the light will benefit from the multi-bounce by providing softer shadow edges.

Inside the closet shown in Figure 3.4b on the next page, emissive lights were used. Emissive lights are great to use when trying to fill up an area or light. These are set through an emissive material and can also be baked into the lightmaps. For some of the lamps and plants in the scene, materials have been created that pass through a bit of lighting. These materials used the method of subsurface shading.

Dynamic lights where used inside lamps that could be interacted with. These lights do interact with the stored lightmaps but do not bounce indirect lights.

Lightmass

The lightmass importance volume is used to map an area in which the lighting is calculated with more accuracy. This area should extend a bit further than the edge of the models. When the scene has windows or open areas where the light should pass through, lightmass portals should be placed. As explained in the theory, the lightmass portals yield higher quality light and shadows.

For the light build of the final scene, always use the production quality setting. Because increasing the bounce settings gives very low return when the number is high, we keep the number of bounces for the indirect light and skylight the same as set up during the lighting set up iterations.

As explained in the theoretical research, the lighting level scale and indirect lighting quality have the highest impact on build times. Those settings increase the level of detail in our scene and are very important to be set to other values than the default 1:1. To gain detail, we want to decrease the lighting level scale to either 0.1 or 0.2. When the lighting level scale is being decreased, we need to increase the indirect lighting quality. A good practice is using the formula of “lighting level scale x indirect lighting quality” = 1.0. Platteaux, J. (2017, October 9).

IES profiles

The Unreal Engine features IES profiles for the following light types: Point Light, Spot Light and Rect Light. IES stands for the Illuminating Engineering Society. IES profile files are ASCII files containing a format for metadata about the distribution of light from a light source.

Dissecting the reference image, the author had gotten as a reference for the final scene (appendix C – I), exposed the use of fake lighting to get the desired effect. As shown in figure 3.4a, the lighting on the top part of the closet is not possible with the lamps hanging from the ceiling. What the author found out was, because the scene of figure 3.4a is a static scene, the original creator placed lights closer towards the closet to get a more lit view of the closet area. In figure 3.4b the author used an IES profile to get a close result. This IES profile made sure the light that was emitted had the right shape, intensity and fall off.

Figure 3.4a (The Virtual Dutch Men, n.d.)

Figure 3.4b

Functionality and project migration

In Unity the code language used is C#, making a switch to C++ in Unreal brings a lot of changes. In the UE4 functionality can be developed using C++ code or Blueprints Visual Scripting. Blueprints is the node-based scripting language of Unreal which uses a visual editor. Slower compile times is one of the results when you use C++ in Unreal. Blueprints is a great option to quickly prototype your functionality but can also be used for most of the use cases.

The functionality that is built into the scene was all done using Blueprints. Objects in the scene can be linked or converted into a blueprint to hold functionality. The author used a trigger area linked to a lamp to indicate possible interaction for turning the lamp on or off. A player controller was added to the scene that could be moved through the whole scene. Functionality to click on objects was added together with a user interface to change materials or save high-resolution screenshots.

From a business' perspective, functionality should be easily migratable from project to project. Unreal has a great built-in tool which allows you to migrate anything from one project to another. With this tool, it is easy to grab the functionality from an existing project and add it to a new one. It becomes simple to build a library of functions and models ready for any Unreal project. Unity can do this by copying scripts or by exporting content as a Unity Package.

Performance optimization

Running the scene in the highest quality settings requires good hardware and good models. Too many high polycount models, dynamic lighting and shadows will result in a drop of FPS and make the program run with lag. Especially when the scene needs to run in VR, optimizations should be made.

One optimization is to have all lights static. Resulting in no extra lighting and shadow calculations. Another method is the use of models with a lower polycount. Models for renders in V-Ray can be very complex for detail in Arch Viz. In a game engine like UE4, all models in the scene are drawn to the screen each frame. High complexity models will take extra computing time. Models that have a very high polycount, can be considered for optimizations by lowering the polycount while maintaining the details. Another option to lower the polycount is to use LODs. This stands for "level of detail" and switches a model to lower detailed versions when the distance between the model and the camera becomes greater.

Post Effects

Post Processing Volumes are used in unreal to map areas in which post processing effects should be applied. One approach the author took to start of the scene was disabling auto exposure on the camera. This makes sure the camera does not adjust its values to let in more light and make the scene brighter. Auto exposure works likes a human eye in darkness.

For Arch Viz it is very important to disable auto exposure. When it is enabled, moving the camera from a light to dark area will trigger the exposure and mess up your view of the scene. Resulting in you changing the lighting settings, and thereby also changing the total lighting of other parts in your scene. To avoid this, we force the value to 1.0 for minimum and maximum exposure.

Other options to turn off at the start of our scene are: motion blur, bloom, vignette, screen space ambient occlusion and screen space reflections. The last settings to set are the default tone mapper values. With all these options set you can start changing and editing the lighting settings. When the look of the scene is complete, we can come back to these values and change the ones we prefer to have turned on.

When the Arch Viz scene consists of multiple rooms in which you want different post effects. UE4 supports the use of multiple post processing volumes and can blend them into each other.

3.2 RESULTS

In the method chapter, the author explained how he used the techniques and information from the practical and theoretical research. Following his approach, the author tested each iteration and reviewed the results. His team helped him achieve insight into the modelling and texturing part of the workflow.

All prototypes and iterations took a lot of computing time. Unreal Swarm proved to be a very good tool to speed up development speed and the ability to make a final light build in a short time. Although the team started using Unreal Swarm late into development, there was no real loss of production time. Most of the high setting light builds were done over the weekend. However, once swarm was operating over the network it made a great impact on the development speed during office hours. For the client, swarm is potentially a great and must have tool when there are short deadlines or quick iterations need to be made on lighting results.

To meet the deadline set by the client in week 15 just before Christmas, the team used Swarm to meet the deadline. A normal light build on the development computer would have taken four to five hours. With Swarm hooked up to three other laptops and one extra computer the build time was only 18 minutes. Appendix C – IV shows the unreal swarm agent running with five computers.

Test results of the Lightmass, light types and lighting settings proved that the UE4 can deliver high quality Arch Viz results. The final scene surpassed the initial wishes of the client as new content and functionality was added.

During the project, the client and team shared their knowledge and research to design a scene that meets the client's high-quality standard. The client is happy to see their feedback and wishes were integrated into the final product.

Chapter 4

CONCLUSION

The conclusion chapter of this thesis is split into two parts. The first section contains a summary and conclusion about the research and methods that were implemented during the production phase of the workflow. In section 4.1 we will reflect on the methods used and address other areas that are open for more research.

In this project, the author constructed an Arch Viz scene in the UE4 with models used for original content by the company. Following the pipeline of the company, the author researched the process and workflow of constructing this Arch Viz scene in Unreal. During the project, the author informed the client of the steps in the new workflow and documented these in the blog or text documents.

After a slow start and encountering issues, the author had to learn more about the UE4 and the original models provided by the client. As the author iterated through newly learned techniques and tested the scene with more models and materials, a workflow was established. This workflow came forth from theoretical and practical research. With this workflow, the quality of materials and lighting improved, and the speed of the development increased significantly.

Looking back at the overall workflow of the company. Unreal would fit right into their production workflow. Unreal provides great tools for the company to speed up the current production workflow. One of those tools used by the author was the Unreal Swarm. Setting up the Swarm Coordinator was easy and should have been done as early as possible in the production phase. The Unreal Swarm Agent automatically starts when working with Lightmass, but sharing the workload required the coordinator to be active and set up with computers on the network running the swarm agent.

With the Unreal Swarm, migrate option and Lightmass, the Unreal Engine proves to produce a high-quality lighting solution for Arch Viz. The Unreal Swarm speeds up light baking time which is essential within the company's workflow. Migrating content and functionality proved to be very easy and helps build one base which can then be reused in multiple projects.

In the discussion part, extra plugins and components will be explained for further research. These can make the Unreal Engine stand out even more.

4.1 DISCUSSION

Although there are enough results to prove the quality results of static lighting in Unreal. There are still features left to test when a static scene must be transformed into a dynamically lit scene. Because the research mainly focussed on static lighting, the dynamic lighting options of Unreal are still open for more research. Unreal provides great solutions however for creating dynamically lit scenes.

Because this project was done with the Unreal Engine, the author has not yet tried the Datasmith solution provided by Unreal Studio. Datasmith is a workflow toolkit which helps to import your complete scene from many other 3D software programs and to translate them to Unreal. This tool will drastically speed up development and importing times since it can take your original composition of models and materials and import them directly. Another great aspect of Unreal Studio is that it features a 100% royalty-free license.

Chapter 5

RECOMMENDATIONS

Based on the conclusion and discussion of this research, the author recommends the Unreal Engine for a static light solution for real-time demonstrations. The workflow in Unreal fits right into the current workflow of the company and with a little bit of tweaking or cleaning up the models would run in a statically lit scene. The author believes the lighting and shadows to be of sufficient quality for projects to capture a cinematic or high-resolution screenshot from.

A highly recommended tool to use in the workflow when working with the UE4 is the Unreal Swarm. Together with the Lightmass settings and volume, the swarm proved to work well in distributing the workload and reducing the light bake times. The blog also contains these workflow decisions and shows their results.

The author also provides a recommendation in the discussion for extra research on features on dynamic lighting. The scope of this project did not include dynamic lighting, therefore only a few solutions and options have been tested and described throughout the theory. However, one feature currently in development is the Light Propagation Volume in Unreal. This is Unreal's solution to dynamic Global Illumination.

The Unreal Studio workflow with Datasmith is a workflow that should be looked at because of the benefits for the company. It will improve the development speed by importing the whole scene from the other 3D program. Unreal Studio is also beneficial in terms of costs because it is a 100% royalty-free license.

Chapter 6

GRADUATION PRODUCTS

With the research that has been done and implemented during the process of this project, a final demonstration of Arch Viz in the Unreal Engine has been created. This demonstration shows the possible result of the company's models inside the UE4.

With the research and recommendation, the author hopes to provide a good view of the possibilities with the Unreal Engine. Because the unreal workflow has been researched to fit in with the current workflow of the company, the company can follow my workflow and apply the techniques they need to convert their concept into a final product.

A development blog was created during the process of the project to write down documentation on the workflow and components in the Unreal Engine. The blog is an addition to the report and can be used and reviewed by the company when they want to reproduce the steps the author took during his process. With this blog, they learn the steps to work with Unreal or train their employees to work with the engine. The blog can be accessed with the URL found in appendix B.

SOURCES

Atlassian (n.d.). *Bitbucket.* Retrieved September 2018, from https://bitbucket.org

Autoriteit Persoonsgegevens (n.d.). *Algemene informatie AVG.* Retrieved January 2019, from Autoriteit Persoonsgegevens: https://autoriteitpersoonsgegevens.nl/nl/onderwerpen/avg-europese-privacywetgeving/algemene-informatie-avg

Crassin, C., Neyret, F., Sainz, M., Green, S., & Eisemann, E. (2011, September). *Interactive indirect illumination using voxel cone tracing*. In Computer Graphics Forum (Vol. 30, No. 7, pp. 1921-1930). Oxford, UK: Blackwell Publishing Ltd.

C-Media (2017, November 26). *Tutorial | UE4 Lighting Overview* [Video file]. Retrieved November 2018, from Youtube: https://www.youtube.com/watch?v=kFaEf8V8XYY

Cornish, D., Wright, D., & Nalezynski, R. (n.d.). *Unreal Swarm – Massive Application Distribution for Unreal Engine 3.* Retrieved November 2018, from https://api.unrealengine.com/udk/Three/Swarm.html

DanielW (2015, October 23). *Lets make Lightmass EPIC (and understandable)* [Online discussion group]. Retrieved October 2018, from Unreal Engine Forums: https://forums.unrealengine.com/development-discussion/architectural-and-design-visualization/60547-lets-make-lightmass-epic-and-understandable

Epic Games (n.d.). *Unreal Engine Academy Videos*. Retrieved September 2018, from Unreal Engine: https://academy.unrealengine.com/

Epic Games (n.d.). *Unreal Engine Answer Hub*. Retrieved October 2018, from Unreal Engine: https://answers.unrealengine.com/index.html

Epic Games (n.d.). *Unreal Engine Documentation*. Retrieved September 2018, from Unreal Engine: https://docs.unrealengine.com/en-us/

Epic Games (n.d.). *Unreal Engine Editor Viewport View Modes.* Retrieved September 2018, from Unreal Engine: https://docs.unrealengine.com/en-us/Engine/UI/LevelEditor/Viewports/ViewModes

Epic Games (n.d.). *Unreal Engine Forums*. Retrieved October 2018, from Unreal Engine: https://forums.unrealengine.com/

Epic Games (n.d.). *Unreal Engine Lighting and Shadows Documentation*. Retrieved September 2018, from Unreal Engine: https://docs.unrealengine.com/en-us/Engine/Rendering/LightingAndShadows

Epic Games (n.d.). *Unreal Engine Marketplace.* Retrieved from Unreal Engine: https://www.unrealengine.com/marketplace/store

Epic Games (n.d.). *Unreal Engine Rendering Documentation*. Retrieved September 2018, from Unreal Engine: https://docs.unrealengine.com/en-us/Engine/Rendering

Epic Games (n.d.). *Unreal Engine Resources*. Retrieved September 2018, from Unreal Engine: https://www.unrealengine.com/en-US/resources

Epic Games (n.d.). *Unreal Engine Wiki*. Retrieved October 2018, from Unreal Engine: https://wiki.unrealengine.com/index.php?title=Main_Page

Epic Games (n.d.). *Unreal Swarm.* Retrieved November 2018, from Unreal Engine Documentation: https://docs.unrealengine.com/en-us/Engine/Rendering/LightingAndShadows/Lightmass/UnrealSwarmOverview

Evermotion (n.d.). *Evermotion Tutorials.* Retrieved October 2018, from Evermotion: https://evermotion.org/tutorials

Franczak, M. (2015, September 15). *Seven pro tips that will improve your Unreal visualizations.* Retrieved November 2018, from Evermotion: https://evermotion.org/tutorials/show/9714/seven-pro-tips-that-will-improve-your-unreal-visualizations

Franczak, M. (2017, March 3). *Making of Bright Dining Room – Tip of the Week.* Retrieved October 2018, from Evermotion: https://evermotion.org/tutorials/show/10584/making-of-bright-dining-room-tip-of-the-week

Franczak, M (2017, June 29). *Creating custom sky from HDR image in Unreal Engine.* Retrieved October 2018, from Evermotion: https://evermotion.org/tutorials/show/10738/creating-custom-sky-from-hdr-image-in-unreal-engine

Franczak, M. (2017, October 11). *Unreal Engine Lighting for Arch-Viz.* Retrieved November 2018, from Evermotion: https://evermotion.org/tutorials/show/10860/unreal-engine-lighting-for-arch-viz

Franczak, M (2018, May 5). *Making an Unreal Loft – Tip of the Week.* Retrieved October 2018, from Evermotion: https://evermotion.org/tutorials/show/11039/making-of-unreal-loft-tip-of-the-week

Franczak, M (2018, September 11). *Making of Modern House in Unreal Engine – Tip of the Week.* Retrieved October 2018, from Evermotion: https://evermotion.org/tutorials/show/11133/making-of-modern-house-in-unreal-engine-tip-of-the-week

Franczak, M. (2018, October 15). *Unreal Engine 4: Making of Artistic Loft.* Retrieved October 2018, from Evermotion: https://evermotion.org/tutorials/show/11176/unreal-engine-4-making-of-artistic-loft

GitKraken (n.d.). *Git Kraken.* Retrieved September 2018, from https://www.gitkraken.com/

GitHub (n.d.). *Git Large File Storage.* Retrieved November 2018, from https://git-lfs.github.com/

Hadwiger, M., Ljung, P., Salama, C. R., & Ropinski, T. (2009). *GPU-Based Volume Ray-Casting with Advanced Illumination*. In Eurographics 2009.

Hobson, T. (n.d.). *Lighting Channels.* Retrieved December 2018, from Tim Hobson: http://timhobsonue4.snappages.com/lighting-lighting-channels

Hobson, T. (2017, January 24). *Lighting Techniques & Guides | Live Training | Unreal Engine* [Video File]. Retrieved December 2018, from Youtube: https://www.youtube.com/watch?v=jCsrWzt9F28

Hobson, T. (2018, April 23). *Swarm Agent Troubleshooting.* Retrieved November 2018, from Unreal Engine Wiki: https://wiki.unrealengine.com/Swarm_Agent_Troubleshooting

Hoffman, M. (2017, December 11) *Unreal Engine 4 Rendering Part 1: Introduction.* Retrieved October 2018, from Medium: https://medium.com/@lordned/unreal-engine-4-rendering-overview-part-1-c47f2da65346

Jensen, H. W., Christensen, P. H., Kato, T., & Suykens, F. (2001). *A practical guide to global illumination using photon mapping*. SIGGRAPH 2001 Course Notes.

Karis, B., & Games, E. (2013). *Real shading in unreal engine 4*. Proc. Physically Based Shading Theory Practice, 621-635.

Lauritzen, A. (2010). *Deferred rendering for current and future rendering pipelines*. SIGGRAPH Course: Beyond Programmable Shading, 1-34.

Leleux, B. (2018, July 13). *Setting Lighting in Unreal Engine 4.20.* Retrieved December 2018, from 80 level: https://80.lv/articles/setting-lighting-in-unreal-engine-4-20/

Looman, T. (2014, September 14). *Custom Depth in Unreal Engine 4.* Retrieved December 2018, from Tom Looman: https://www.tomlooman.com/the-many-uses-of-custom-depth-in-unreal-4/

Looman, T. (2016, January 10). *Getting Started with VR in Unreal Engine 4.* Retrieved December, from Tom Looman: https://www.tomlooman.com/getting-started-with-vr/

Looman, T. (2016, September 9). *VR Template Guide for Unreal Engine 4.* Retrieved December, from Tom Looman: https://www.tomlooman.com/vrtemplate/

Luoshuang (2018, April 13). *Luoshuang's GPULightmass* [Online discussion group]. Retrieved October 2018, from Unreal Engine Forums: https://forums.unrealengine.com/development-discussion/rendering/1460002-luoshuang-s-gpulightmass

Mittring, M. (2012). *The technology behind the unreal engine 4 elemental demo*. part of "Advances in Real-Time Rendering in 3D Graphics and Games," SIGGRAPH.

Owens, B. (2013, October 28). *Forward Rendering vs. Deferred Rendering.* Retrieved October 2018, from Tutsplus: https://gamedevelopment.tutsplus.com/articles/forward-rendering-vs-deferred-rendering--gamedev-12342

Peterson, M. & Tokarev, K. (2018, November 28). *Creating a Haunted House Animation Project.* Retrieved December 2018, from 80 level: https://80.lv/articles/creating-a-haunted-house-animation-project/

Peterson, M. & Tokarev, K. (2018, April 30). *Setting Up Interior Scenes in UE4.* Retrieved December 2018, from 80 level: https://80.lv/articles/setting-up-interior-scenes-in-ue4/

Platteaux, J. (2017, October 9). *Lighting with Unreal Engine Masterclass | Unreal Dev Day Montreal 2017 | Unreal Engine* [Video file]. Retrieved September 2018, from Youtube: https://www.youtube.com/watch?v=ihg4uirMcec

Reis, R. (2015, November 10). *Lighting Workflow.* Retrieved October 2018, from UE4Arch: https://ue4arch.com/tutorials/ue4archs-unreal-engine-4-lighting-workflow-part-1/

The Virtual Dutch Men (n.d.). *Over The Virtual Dutchmen.* Retrieved December 2018, from The Virtual Dutch Men: https://thevirtualdutchmen.com/over-the-virtual-dutchmen/

Tunno, F. & Sergeev, A. (2018, December 22) *Setting Up Lighting in UE4.* Retrieved December 2018, from 80 level: https://80.lv/articles/setting-up-lighting-in-ue4/

TortoiseSVN (n.d.). *TortoiseSVN.* Retrieved November 2018, from https://tortoisesvn.net/

Unreal Archviz Video Tutorial (n.d.). *Home* [Youtube channel]. Retrieved September 2018, from Youtube: https://www.youtube.com/channel/UCBA3Z4PYKbS83BO2hxRUxpQ/

Wikipedia (2019, January 9). *General Data Protection Regulation.* Retrieved January 2019, from Wikipedia: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

Wikipedia (2018, November 4). *Global Illumination.* Retrieved November 2018, from Wikipedia: https://en.wikipedia.org/wiki/Global_illumination

Wilson, H. (2017, August 15). *[UE4] Learning Lighting Art* [Online discussion group]. Retrieved November 2018, from Polycount: https://polycount.com/discussion/190402/ue4-learning-lighting-art

Wilson, H. & Tokarev, K. (2018, February 21). *Building a Darker Version of Bioshock Infinite Scene.* Retrieved December 2018, from 80 level: https://80.lv/articles/building-a-darker-version-of-bioshock-infinite-scene/

Wilson, H. & Tokarev, K. (2018, December 10). *Lighting Studies with Harley Wilson.* Retrieved December 2018, from 80 level: https://80.lv/articles/lighting-studies-with-harley-wilson/

3darchstuffs (n.d.). *Home* [Youtube channel]. Retrieved September 2018, from Youtube: https://www.youtube.com/user/3darchstuffs2/

APPENDICES

A – ABBREVIATIONS

Arch Viz	–	Architectural visualization
FPS	–	Frames per second
UE4	–	Unreal Engine 4
VR	–	Virtual Reality

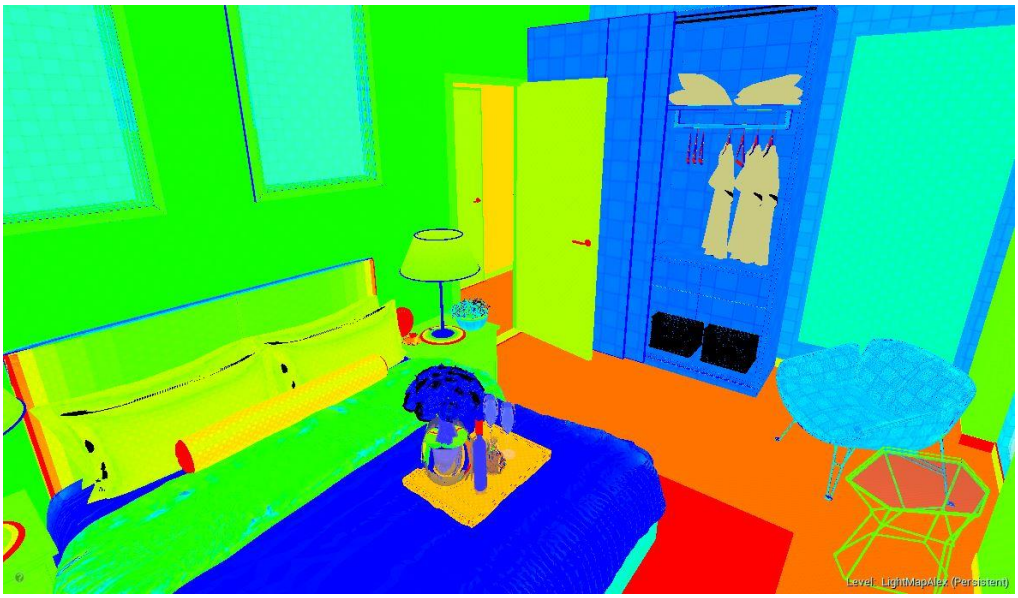
B – BLOG

The development blog can be accessed with the following URL:
https://unrealarchviztvdm.wordpress.com/

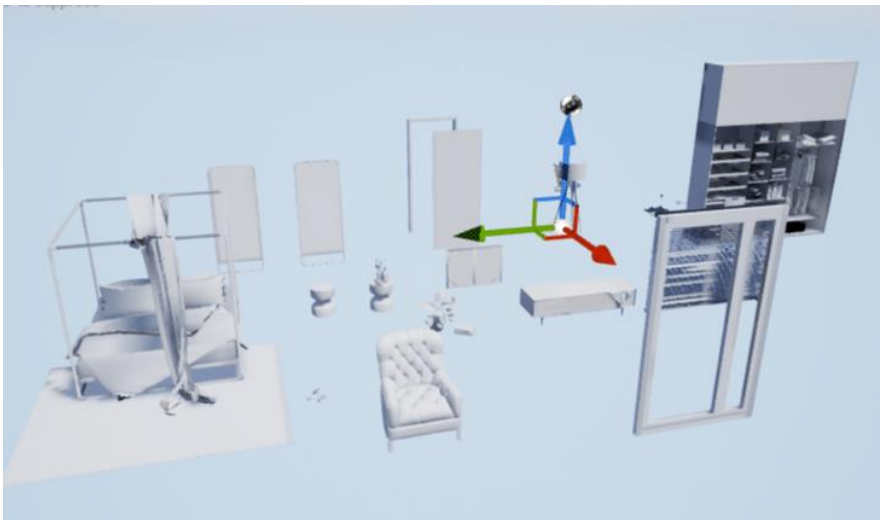
C – IMAGES

I – Reference image

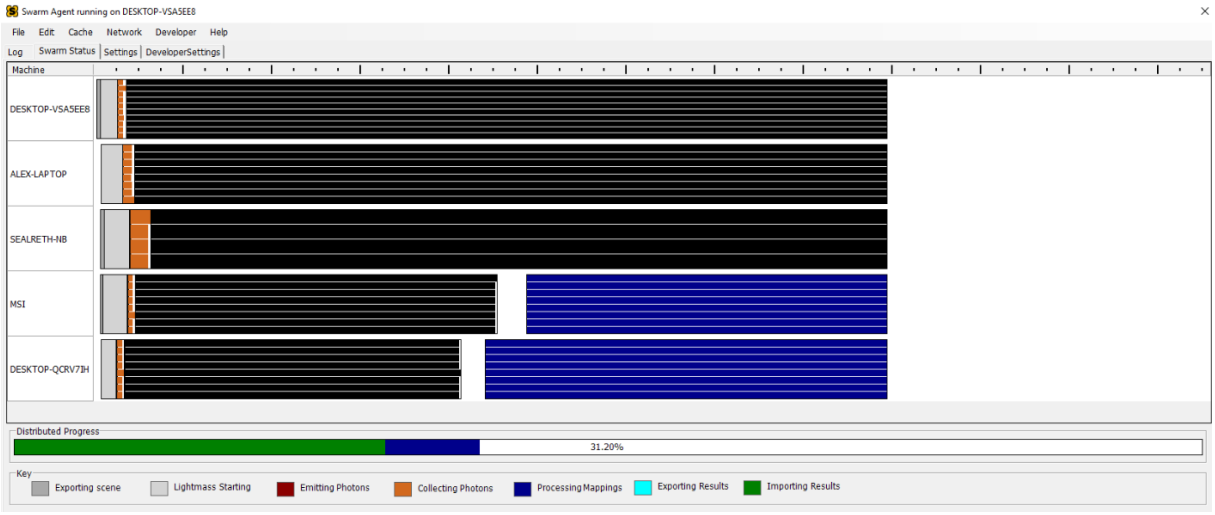
II – Lightmap resolution density view mode



III – First test environment



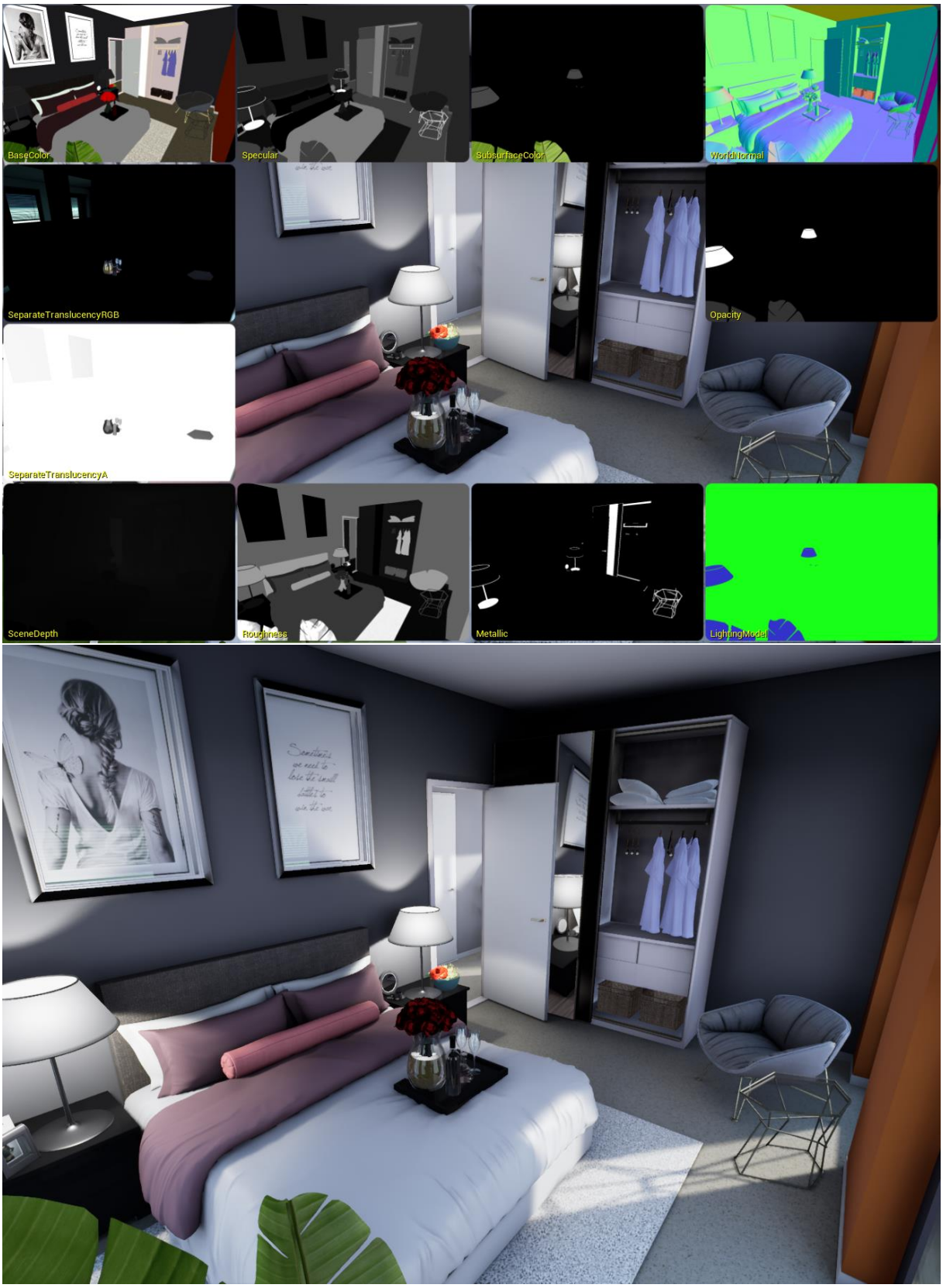
IV – Unreal Swarm



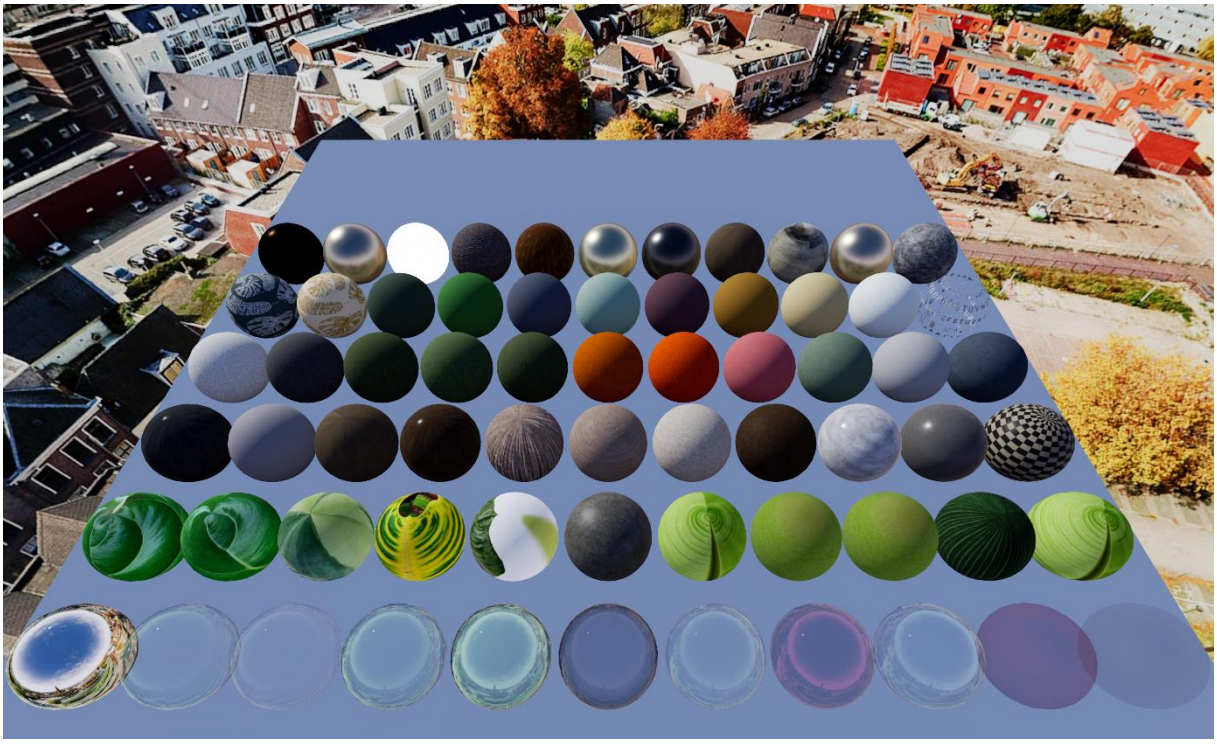
V – Lightmass Importance Volume

VI – Volumetric Lightmap samples visualization

VII – Buffer Overview view mode



VIII – Materials



IX – Final images

D – LIST OF TABLES AND FIGURES

E – TESTS

I - Test results of the scene shown in appendix C – III.

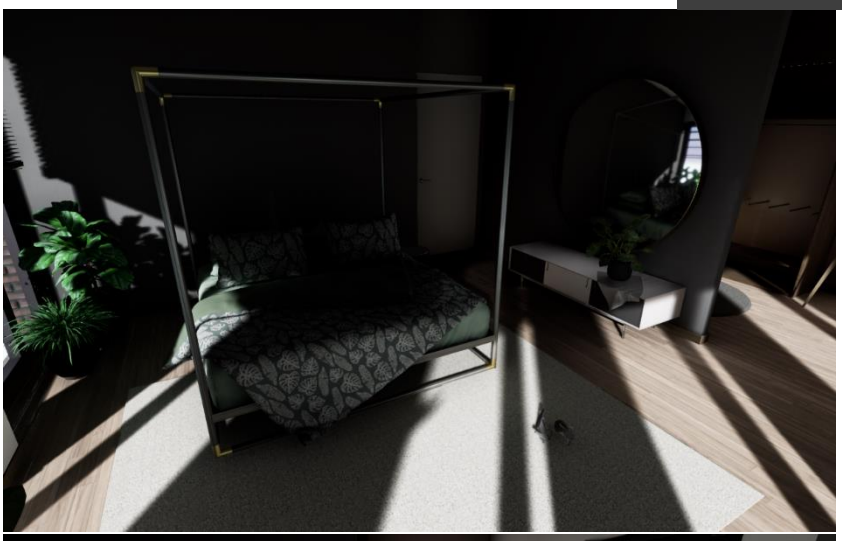
Lightmass on ALEX-LAPTOP:
Processor: Intel® Core™ i7-3610QM CPU @ 2.30 GHz (8 CPUs), ~2.3GHz

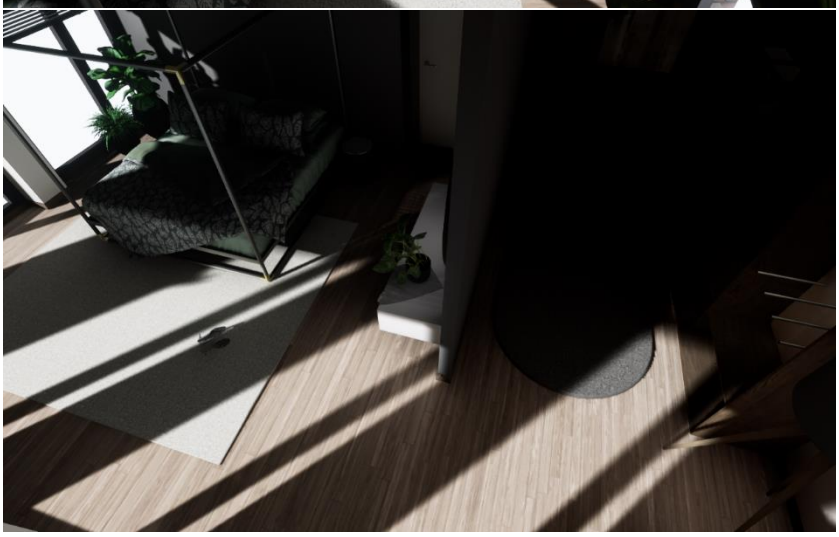
Lightmap resolution	64x64	128x128	256x256	512x512	1024x1024
Total time	1:01 min	1:21 min	2:15 min	8:25 min	52:26 min
Importing	9.46 sec	9.54 sec	10.8 sec	11.1 sec	11.2 sec
Setup	5.84 sec	6.08 sec	6.40 sec	6.19 sec	7.07 sec
Photons	17.7 sec	20.8 sec	26.5 sec	1:11 min	8:31 min
Processing	28.8 sec	44.8 sec	1:32 min	6:55 min	43:36 min
extra exporting [89/89 mappings]	0 ms	0 ms	0 ms	0 ms	0 ms
Threads:					
Total time	3:21 min	5:13 min	10:43 min	48:31 min	5:05:18 hours
Processing	2:26 min	3:31 min	7:48 min	37:28 min	4:09:17 hours

II – Light Bounces

Static Directional Light – 10 intensity

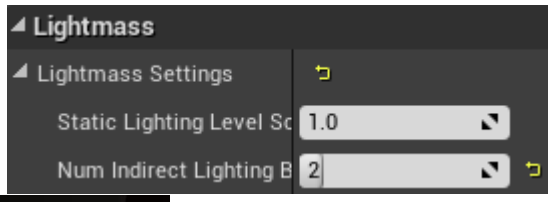
1 indirect lighting bounce

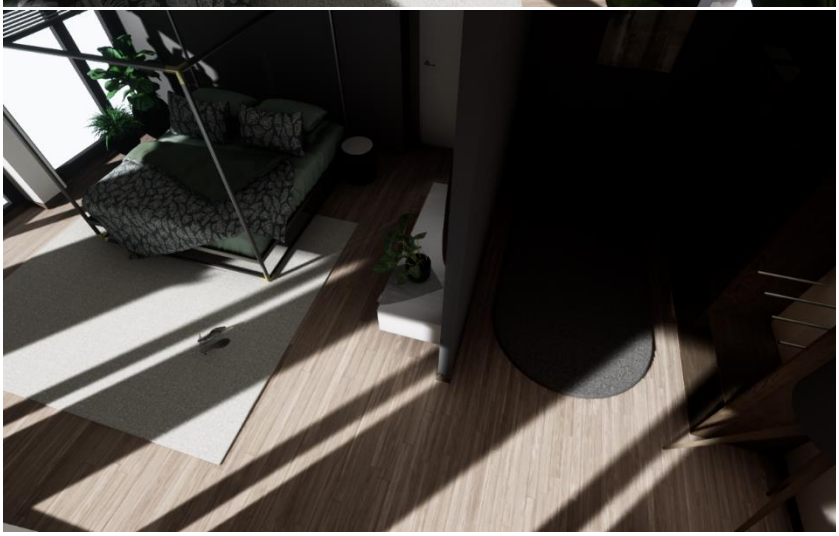




Static Directional Light – 10 intensity

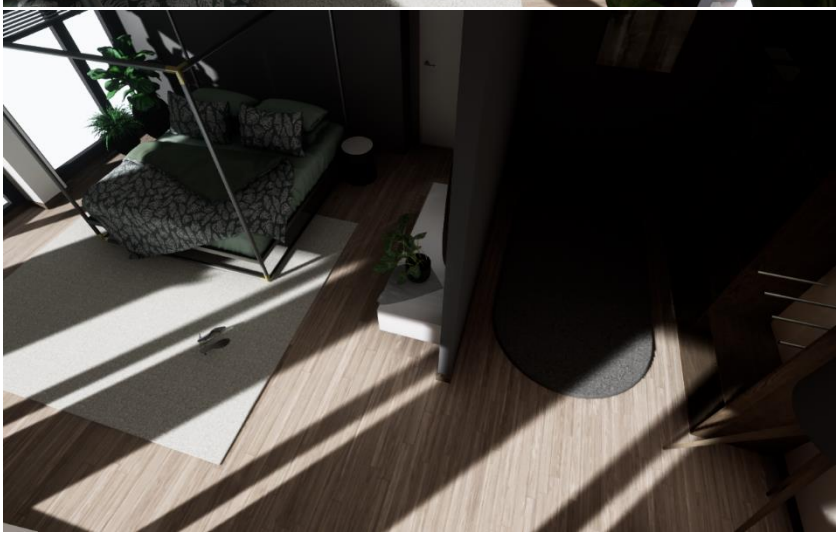
2 indirect lighting bounce





Static Directional Light – 10 intensity

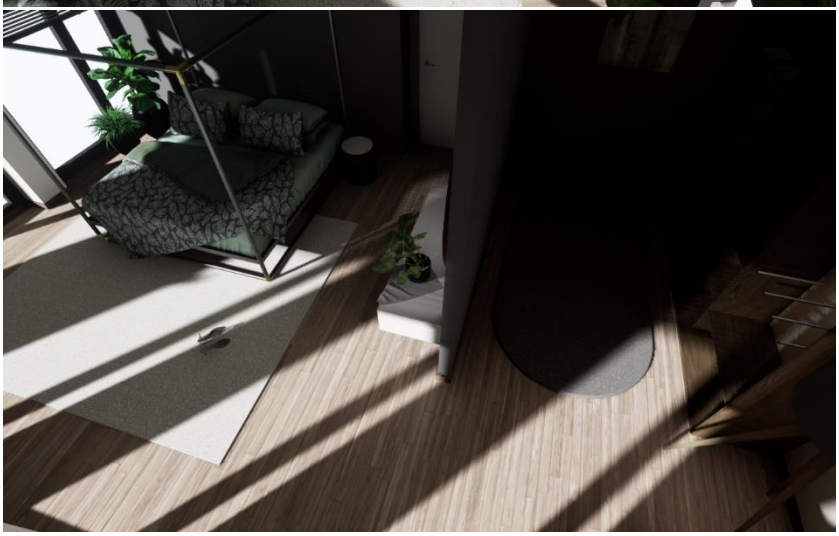
5 indirect lighting bounce



Stationary Directional Light – 10 intensity

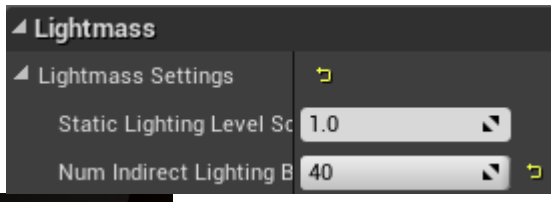
5 indirect lighting bounce

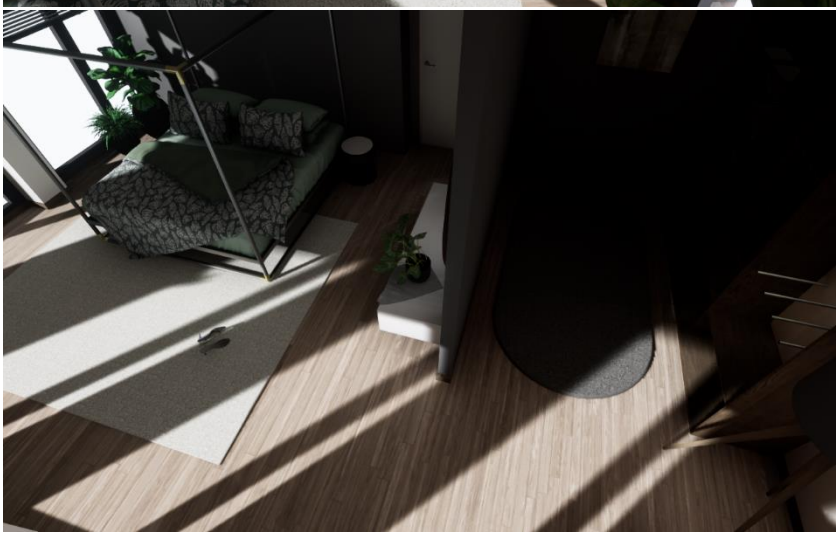
Area shadows enabled



Static Directional Light – 10 intensity

40 indirect lighting bounce





Stationary Directional Light – 10 intensity

5 indirect lighting bounce

Area shadows enabled

Skylight added with 3 bounces