Tampere University of Applied Sciences

Lighting a Mobile Game Scene without Real Time Lights

Mikko Tanskanen

BACHELOR'S THESIS
May 2023

Degree Programme in Media and Arts
Interactive media

ABSTRACT

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

MIKKO TANSKANEN
Lighting a Mobile Game Scene without Real Time Lights

Bachelor's thesis 36 pages, appendices 1 pages
May 2023

Lights are an important tool in a video game maker's toolkit. They lead the player, set the mood, and are an integral part of visual storytelling. However, real time lights have high performance costs and are difficult to run on mobile devices, and with the ever-growing interest in mobile gaming it is important to learn how to make optimized lights for mobile devices with limited resources.

The objective of this thesis was to study the best practices of optimizing light for mobile games. Optimization techniques for Unity game engine projects were also studied while conducting research for this thesis.

The gathered information from professionals, online articles and Unity’s own online manuals were then put into practice by creating a demonstration in the form of a fully lit mock-up of a video game lobby scene in Unity game engine. Multiple optimization techniques were used in the scene, including lightmaps, light probes, and batching. Use of the optimized lighting proved to be beneficial for the scene, providing an atmosphere unobtainable without lights.

Key words: light, unity game engine, mobile game, optimization

CONTENTS

GLOSSARY

Game engine	Software handling all the necessary calculation to run a video game.
Raytracing	Lighting technique that simulates physical light rays' behaviour.
Baking	Act of calculating and storing information in advance.
FPS	Frames per second.
UV unwrapping	Act of creating 3D objects UV Map; process of projecting 3D models surfaces onto a 2D texture.
Draw call	A "call" game engine sends to a graphical processing unit to draw something.
Polygon	A plane with at least three edges.
Thermal throttling	Process of cooling CPU and GPU under heavy workflow by lowering work speeds.
Hue	Colour's unique position on the colour spectrum. Pure colour.
Saturation	Measurement for the intensity of a colour.
Value	Measurement for the darkness or lightness of a colour. Also known as brightness.
Artifact	Unexpected visual in a Lightmap.

1 INTRODUCTION

The function of light in video games is much more than just lighting up a room. Video games transport people into other worlds and, it is important that they stay fully immersed in those worlds (Vertex School, 2022). In video games this is accomplished with game design, level design, art direction etc. Light however affects all of these, it enhances all the other work (Massive Entertainment, 2021). It can make or break a scene; in other words, it can make or break the player's immersion. That is why understanding good lighting and what that entails in video game design is important.

Typical lighting techniques from film can be applied to lighting up video game scenes (Houzé, R. 2019). Contrasts, colour temperature, colour gradients, different lighting compositions all must be thought of. However, when designing lighting for a film you only see it from one point of view, through the frame of the lens. Video games are interactive and can possibly be viewed from multiple different angles (Foundry, 2020). When designing lighting for a video game scene, it is recommended to think of it as a 360° view (Massive Entertainment, 2021). First impressions of a scene are important, but think about what comes next, where does the player go, and do they have a clearly defined path where they need to go?

Light has many uses in video games; it can highlight important key objects; it can illuminate pathways for players in the dark. Light can also be used to create emotions, it can be used to frame scenes, it can be used as a source of information, it can even be used as a gameplay element (Houzé, R. 2019). However, light is one of the most taxing aspects when it comes to video game performance. Therefore, the project's memory budget must be kept in mind when planning a scene; to know how much light can be afforded (Massive Entertainment, 2021). In mobile games, this budget is even smaller due to strict limitations of the devices.

1.1 Light in video games

To understand light in video games, it is beneficial to look at examples on how different games use light. In Mojang's Minecraft lighting is very basic, at first there is only one light source, the sun. Minecraft also does not have bounced light in the game as default, but ray tracing has been introduced to the game with mods and Nvidia's ray tracing technology, RTX (Picture 1). Light also works as a game-play element in Minecraft, with day and night cycles and with the lack of light creating enemies into the game world.

PICTURE 1. On the left of the image is Mojang Studios Minecraft without ray tracing, and on the right is Minecraft with NVidia's ray tracing feature on. (Hexus, 2019)

In Fatsharks first-person shooter, Darktide, light and volumetric fog are used to highlight the game world's massive and oppressive scale (Picture 2). Light also acts as a gameplay element in Darktide, be it glowing red lasers pointers from enemy snipers aiming at players or the player character's flashlights illuminating dark paths and corridors for the player.

PICTURE 2. Light and volumetric bloom in Darktide. (Rock Paper Shotgun, 2022)

Ironmaces dungeon crawler, Dark and Darker, creates dangerous and oppressive environments for the players to traverse through by using the lack of light as a dangerous gameplay element. Shadows and dark places are extremely dark in Dark and Darker (Picture 3) and can be used as hiding places by enemy players and can hide deadly traps inside them.

PICTURE 3. Ironmaces Dark and Darker. (Gamesradar, 2022)

In Remedy Entertainments action-adventure game, Alan Wake, light is a key gameplay element. Light must be shone on enemies from the flashlight to burn away a protective coat of darkness from them so that they can be hurt (Picture 4).

PICTURE 4. Alan wake burning enemy's protective coat of darkness away in Alan Wake Remastered. (VG247, 2021)

1.2 Light and colour in traditional media

As stated in the introduction, typical lighting techniques from film can be implemented when creating lighting for video games. To understand better how light and colour can be utilized together to evoke different emotions, it is beneficial to look at different lighting techniques in film, as well as, to take a quick overview of basic colour theory in fine art.

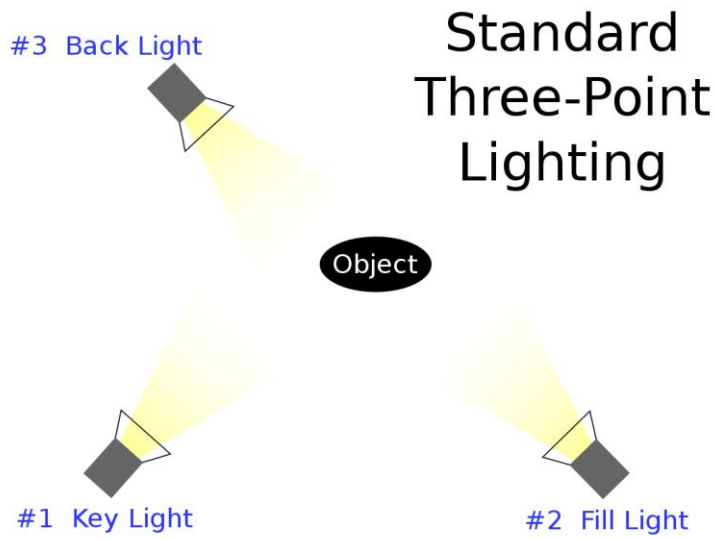
1.2.1 Light in films

Alex Depew (2021) describes cinematic lighting as: “Lighting that evokes feeling and has a style”. This is achieved by deliberately shaping, limiting, and framing lights using different lighting methods. One of the most used lighting methods is a lighting setup called ‘*Three-Point Lighting’* (Picture 5). Three-Point Lighting is designed to illuminate subjects in a scene from three different positions to fully express their three-dimensional shape. This is achieved by using three different lights called: *Key Light*, *Fill Light*, and *Backlight*. (Houzé, R. 2019)

Key Light is the strongest light in a scene, and usually the first light to be set up in a scene. Key Light can be place universally anywhere in a scene, depending on what kind of mood is wanted. However, placing the Key Light right next to the camera is not recommended, for it easily creates flat looking lighting. (Depew, A. 2021)

Fill Light is a light that is positioned opposite to the Key Light, and its job is to illuminate the shadows created by the Key Light. Fill Light should be lower in intensity than the Key Light. (Depew, A. 2021)

Backlight is a light that is placed behind the subject relative to the Key Light, illuminating a wide area of the subject's back, and separating it from the background. The backlight should also be lower in intensity than the Key Light. (Depew, A. 2021)



PICTURE 5. Basic Three-Point Lighting setup (Wikipedia)

Not all lights used in a scene must be industrial filming lights hiding off the screen. Common household light sources such as *candles*, *lamps* and *monitors* can also be employed to create illumination to the scene (Picture 6). Use of these kinds of light sources is called *'Practical Lighting*'. Practical lighting is commonly used to create a night-time scene. (Depew, A. 2021)

PICTURE 6. Practical lighting from candles illuminating a room in Stanley Kubrick's movie Barry Lyndon. (Rotolight, 2019)

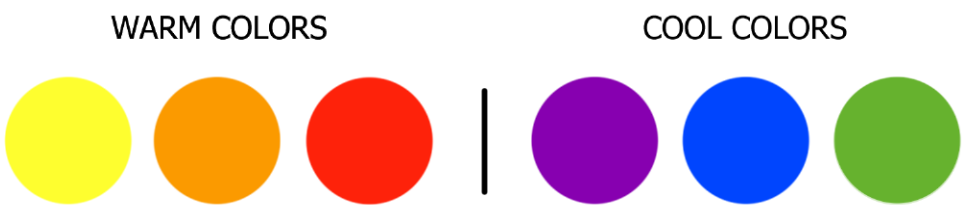
To give off the feeling of a natural light source, such as light from the sun or the moon, while having the complete control of the lights is called '*Motivated Lighting*'. This is done with lights that have been modified to *mimic* the natural light of the

wanted light source (Picture 7). Motivated Lighting can be used to accentuate existing light sources within the scene and to support Practical Lighting. If the *natural* light of the scene is used, it is then called, '*Ambient Lighting'*. (Depew, A. 2022)

PICTURE 7. Motivated lighting (sunlight) can be seen lighting up the two characters through a window in Quentin Tarantino's movie Django Unchained. (Studio-Binder, 2021)

1.2.2 Colour in fine art

In fine art, colour and their different temperatures have many functions. When using the subtractive colour mixing by using paints, dyes, and pigments as colours. The six-point colour wheel of primary (red, blue, and yellow) and secondary (orange, green, and purple) colours can be divided through the middle to get warm and cool colours (Picture 8). Warm colours, *reds*, *oranges*, and *yellows*, tend to be associated with warmth and passion, and cool colours, *blues*, *greens*, and *purples,* are usually associated with cold and calmness. Colours, however, are always relative to what is next to them. A certain red might look warm in some context, but in another, it might occur as cold. It is always important to check what colours are next to each other and how they influence each other. (Art In Context, 2022)



PICTURE 8. Primary and secondary colours divided into warm and cool colours.

The ability to use different colour temperatures in an art piece is an important skill. For example, in landscape paintings, different colour temperatures are utilized to create a sense of depth (Picture 9). Cool colours tend to push things back to the background, and the opposite is true for the warm colours. They tend to push things forward to the foreground. This effect can be used in portrait paintings as well, to express objects form. (Larson, E. 2023)

PICTURE 9. In Pieter Bruegel the Elders painting, *The Harvesters*, the sense of depth can be perceived in the warm yellows and oranges in the foreground, and the cool blues and greens in the background. (Wikipedia)

A common practice, to make sure an art piece would look good, is to decide beforehand what colours are going to be present in the painting, AKA. Deciding the painting's colour harmonies, also known as the *colour scheme*. Six popular colour schemes in the colour wheel are: *monochromatic, complementary*, *split-complementary*, *analogous, triadic* and *tetradic* (Picture 10). (Scott, D. 2019)

The monochromatic scheme utilizes only one colour and achieves contrast in the piece by using different levels of saturations and values of the colour. (Scott, D. 2019)

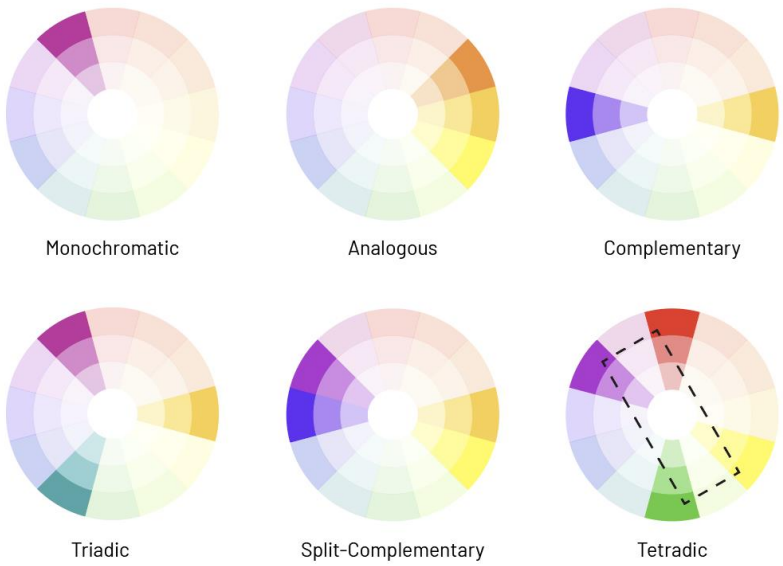
The complementary scheme uses colours opposite to each other in the colour wheel, creating a strong contrast in the art piece. When using a complementary scheme, it is recommended to pick a dominant colour and use the second colour as an accent colour. (Scott, D. 2019)

Split-complementary is almost the same as complementary colour scheme, but one of the complement colours is split into two secondary colours. (Scott, D. 2019)

The analogous colour scheme uses colours that reside next to each other in the colour wheel. Due to the colours being similar in hue, contrast is created by using different values and saturations of the colours. (Scott, D. 2019)

In a triadic colour scheme, three colours are evenly spaced out, creating a harmonious colour scheme. Picking a dominant colour and two secondary colours to work as accents is recommended when using this colour scheme. (Scott, D. 2019)

Finally, tetradic colour scheme, also known as *rectangular colour scheme* or *double-complementary colour scheme*, uses two complementary colour pairs on the colour wheel. When using this colour scheme, it is recommended to pick a dominant colour and use the three colours as accents. (Alscher, D. 2019)



PICTURE 10. Six common colour schemes. (G2, 2019)

1.3 The importance of optimizing light

In 2021 there were more than 2.1 billion mobile game users across the globe, Asia containing 1.29 billion users of that, accounting more than 48 percent of the global mobile game users (Clement, J. 2021). With so many users, comes a wide range of different devices with different levels of hardware (Unity 2021, 3). The point of optimizing games to run on low-end devices is to give games a better chance of passing through platform-specific certifications, and to have as wide as possible reach for potential users amidst the different handheld devices (Unity 2021, 3).

The modern standard for frames per second in consoles and PC single-player video games is 60 FPS, and in PC multiplayer first-person shooters the standard is 120 FPS (Branko, G. 2022). Mobile games however usually run around 30 to 60 FPS, this is done to preserve battery life and to fight thermal throttling (Bacioiu, C.; Unity 2021, 23).

Achieving 30 to 60 FPS on low-end devices can be a challenge, and some levels of optimizations are needed to achieve stable frame rates. For example, when targeting low-end devices, post-processing effects are not used preferably at all (Unity, 2020c). These effects can include Bloom, Depth of Field, Motion Blur, and different colour corrections. Heavy usage of real time lights is not recommended in a mobile game. Instead, methods of bringing lighting information to scenes and onto game objects without real time calculations are recommended (Unity, 2020c). These methods include Lightmaps and Light probes.

2 LIGHTING OPTIONS

To achieve “realistic” looking lighting, most modern games rely on ‘global illumination’, also known as ‘GI’. Global illumination simulates bounced light in a scene, which result in a more realistic looking scene, due to dark places receiving light from indirect lighting, and colours mixing from bounced light (Picture 11). This can be difficult and expensive to simulate properly, due to heavy real time calculations. For this reason, games tend to rely on different methods to determine and store lighting data before runtime.

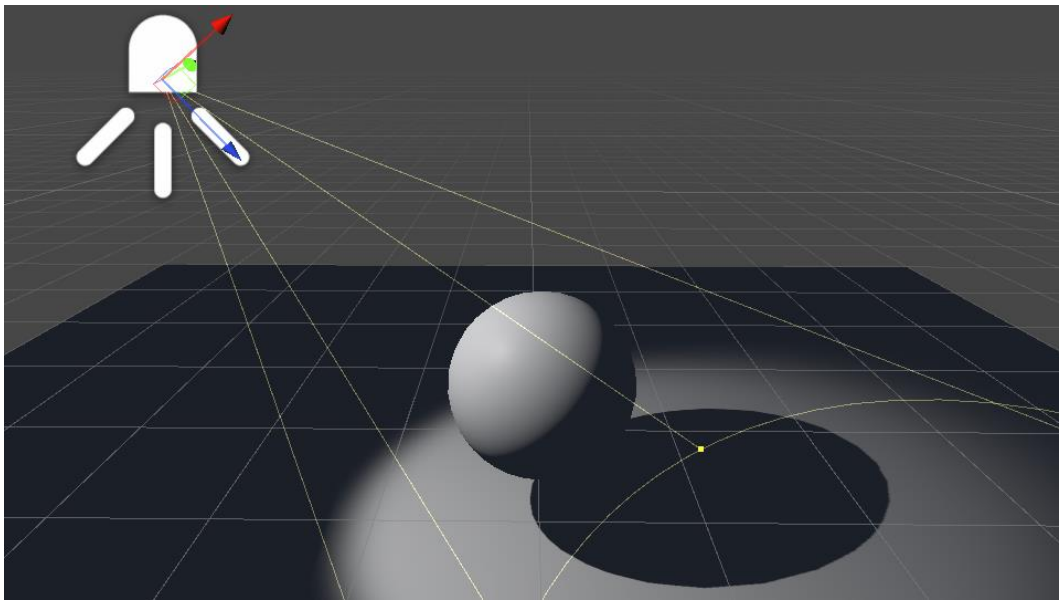
In Unity, lighting can be regarded as ‘Realtime’ or ‘Static’. Both can be used together to create a lighting option called ‘Mixed’. (Unity Learn, 2019b) Unity was chosen for this project out of the possible game engines (Unreal Engine, Godot, Amazon Lumberyard etc.) for its simple and easy to use design, and due to its familiarity with creating games for mobiles.

PICTURE 11. Bounced lights lighting undersides of trees and shadowy places in Santa Monica Studios God of War Ragnarök. (Gaming bolt, 2022)

2.1 Realtime

Realtime lights provide direct light to the scene that is updated on every frame. Benefits of Realtime lights are that they can be changed dynamically, making them interactive; they can be turned off or on, change colours, or be moved. Realtime lights also cast dynamic shadows on game objects, making them highly immersive lights. Realtime lights are in best use when lighting characters or movable game objects. (Unity Learn, 2019b)

By default, Realtime lights do not produce bounced lighting, as seen in picture 12, but if 'Realtime GI' is enabled from lighting settings, Realtime lights start to contribute to the scene's indirect lighting. However, using Realtime GI is highly taxing and is only recommended to be used when targeting mid- to high-end PC systems. (Unity, 2021f)

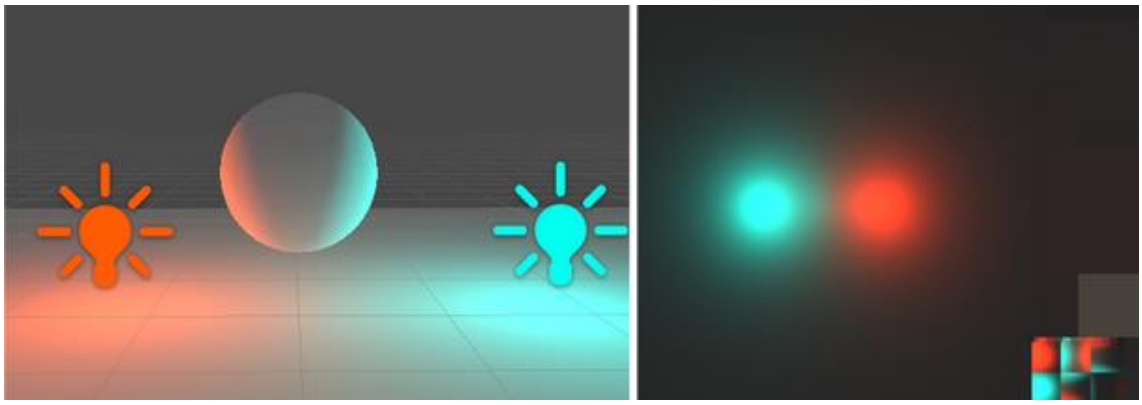


PICTURE 12. Realtime spotlight illuminating a sphere without Realtime GI.

2.2 Static

Static or more commonly referred to as ‘Baked’ lights are pre-calculated lights where light data has been baked into an image called ‘Lightmap’.

Lightmaps are textures that contain pre-rendered light information of a scene, the texture is then overlaid on top of the scene’s geometry to create an illusion of illumination (Picture 13). Lightmaps enable global illumination, ambient lights, and shadows to be present in a scene at runtime with low hindrance to performance (Unity Learn, 2019). Performance is saved due to renderer referring to pre-calculated values from Lightmaps at runtime instead of doing real time calculations (Unity, 2020c).



PICTURE 13. On the left of the image, a sphere is being lit with two light sources using a ‘baked’ light mode. On the right is the scene's Lightmap.

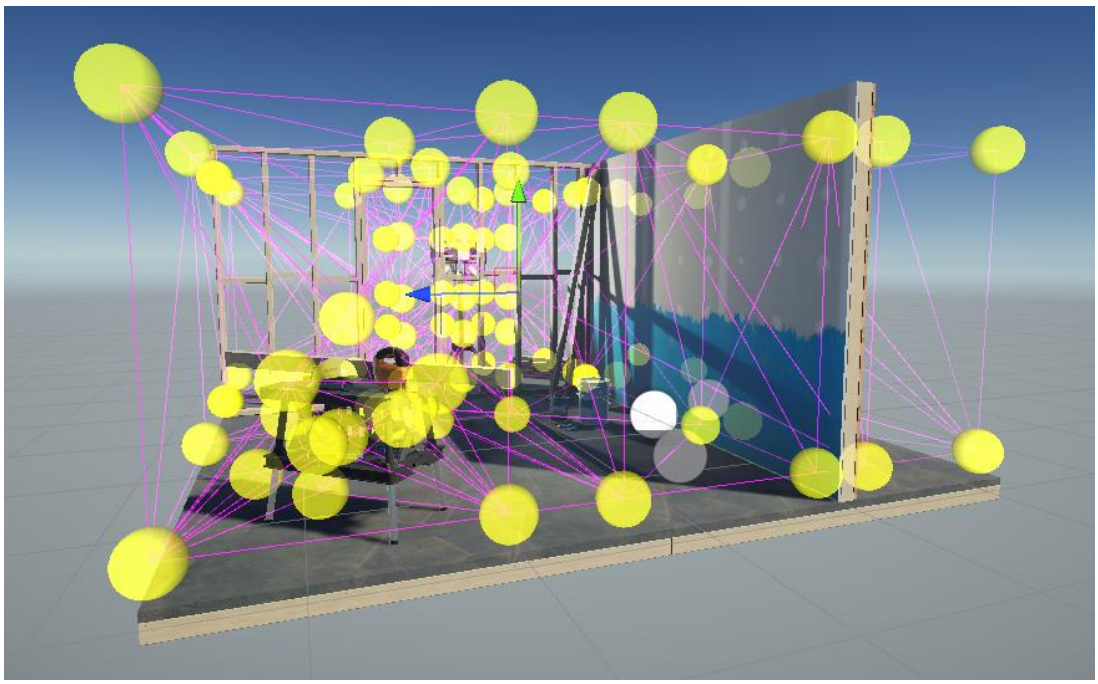
The drawbacks of Lightmaps are that they are textures, therefore, they will inevitably increase build sizes and cause more memory usage at runtime. They also cannot be moved or changed during runtime due to Lightmaps being unable to change during runtime. Realtime lights can be used on top of Lightmaps to create dynamic lights, but they cannot change Lightmaps (Unity, 2021b). Performance benefits, however, usually outweigh these drawbacks (Unity, 2020c). Another limitation of Lightmaps is that they only apply to static objects. When it is desired to reflect pre-calculated values onto a dynamic object, light probes are recommended.

2.2.1 Light probes

Light probes are a helpful tool in Unity, designed to bring light information from baked lights into non-static game objects. According to Unity (2019a), light probes are used for two primary reasons: “To add high-quality light, including indirect bounced light, to moving objects.” and “To provide lighting information for static scenery when that scenery is using Unity’s Level of Detail (LOD) system.”

Light probes work by capturing light information that is passing through empty spaces in a game scene. That baked light information is then reflected onto non-static game objects during runtime by calculating the values from the closest light probes relative to the game object. (Unity Documentation, 2021e)

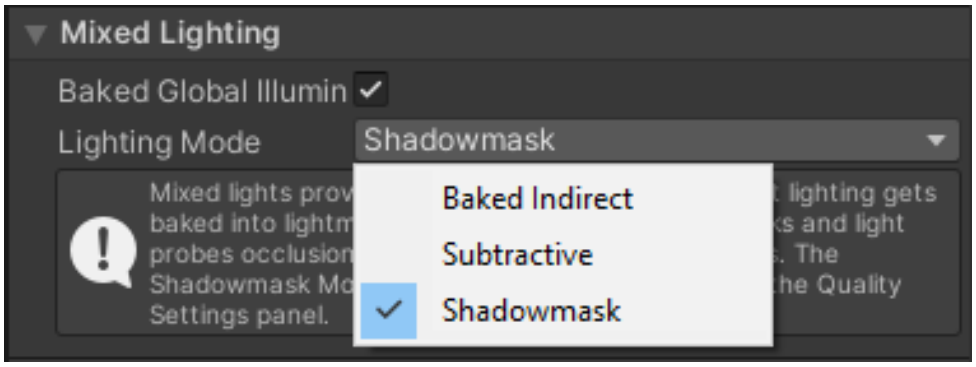
Light probes are used by placing them throughout the scene in a multi-layered network (Picture 14). Light probes are best placed in places where there are strong changes in light, I.e., shadows and strong light sources (Unity, 3, 2021).



PICTURE 14. Light probe network in Unity’s 3D Sample Scene (URP) project.

2.3 Mixed

Lights using Unity's 'Mixed' lighting mode contribute to both Realtime and Baked lights. They all behave according to the scenes lighting assets lighting mode that can be changed under Lighting settings mixed lighting tab, as shown in picture 15. These mixed lighting modes are 'Shadowmask', 'Baked Indirect' and 'Subtractive'.

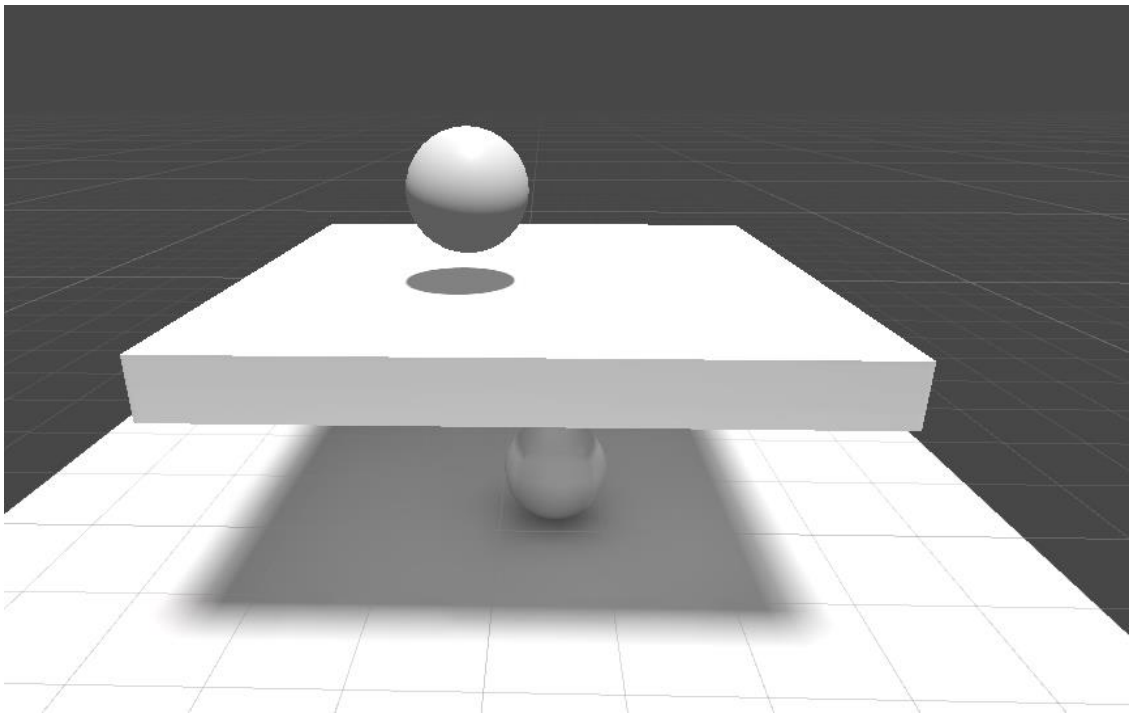


PICTURE 15. Unity's lighting assets mixed lighting window.

In 'Shadowmask' mode, direct lights marked as 'mixed' behave as Realtime lights, but their indirect light will be baked into Lightmaps (Unity, 2021c). Shadowmask mode's specialty is that it can bake shadows at distance and cast real time shadows at close, and then mix the two shadow types together dynamically (Unity, 2020a). This is accomplished by storing extra data into light probes and by creating an additional texture called 'Shadow Mask' (Unity, 2021c). Drawbacks to this mode include high performance and memory budget costs, but it also produces the most realistic shadows out of the three modes. Shadowmask mode is recommended when targeting high to mid-range devices. (Unity, 2021c)

In 'Baked Indirect' mode, mixed lights behave as Realtime lights casting real time shadows, but indirect lights will be baked into Lightmaps. Baked Indirect mode's benefits are savings on the computing power through baked indirect lights while also utilizing transformative aspects of direct lights. It is however recommended that the direct light changes should be slight during runtime, due to baked lights being unable to change during runtime. (Unity, 2021b)

In ‘Subtractive’ mode, all the mixed light’s direct and bounced light and shadows will be baked into Lightmaps, but subtractive mode will allow one directional light, also known as ‘Main light’, in the scene to cast real time shadows on top of static objects. This can be observed in picture 16. Subtractive mode is recommended when you are targeting low-end devices and the scene only calls for one Realtime light. (Unity, 2021d)

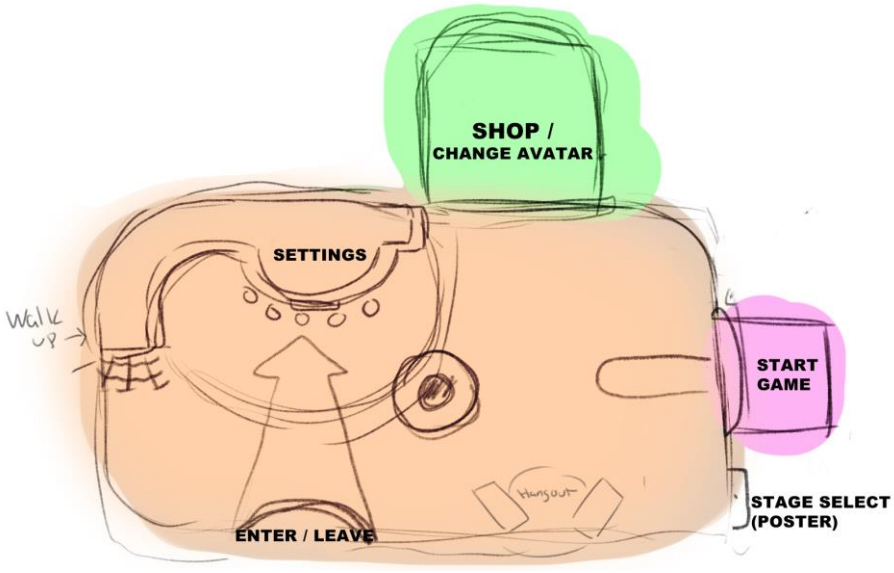


PICTURE 16. A scene lit with a directional mixed light using subtractive mode. On top is a dynamic sphere that is casting a real time shadow, and below is a static sphere that is in baked lighting.

3 PRACTICAL PROJECT

The goal of the project was to plan a mock-up of a lobby scene for a multiplayer online game where players could hang out and launch different types of games. The feel of the lobby was to be a dark and moody hiding place, with warm practical lights creating a comfortable and inviting environment.

Different sketches were done in Photoshop to place down important menu actions as physical places in the lobby scene, actions such as: *Options*, *Stage Select*, *Start Game*, and *In Game Shop* (Picture 17). After that, to help guide the player's interest towards the important locations, a plan of colours and lights associated with the key locations was created. A triadic colour scheme of purples, oranges, and greens was decided to be the lobby's colour scheme. Dark purples were used for the shadows, to invoke a mischievous and mysterious look. Oranges were used as the main colour for the lights throughout the lobby to bring warmth and liveliness to the scene. Finally, to complete the triadic scheme, green was used for the lobby's shop area. Green was used for the shop to associate positive feelings with the area. After the plans were done, a list of important game objects was created, and modelling of the 3D objects and lobby's framework started in Blender.

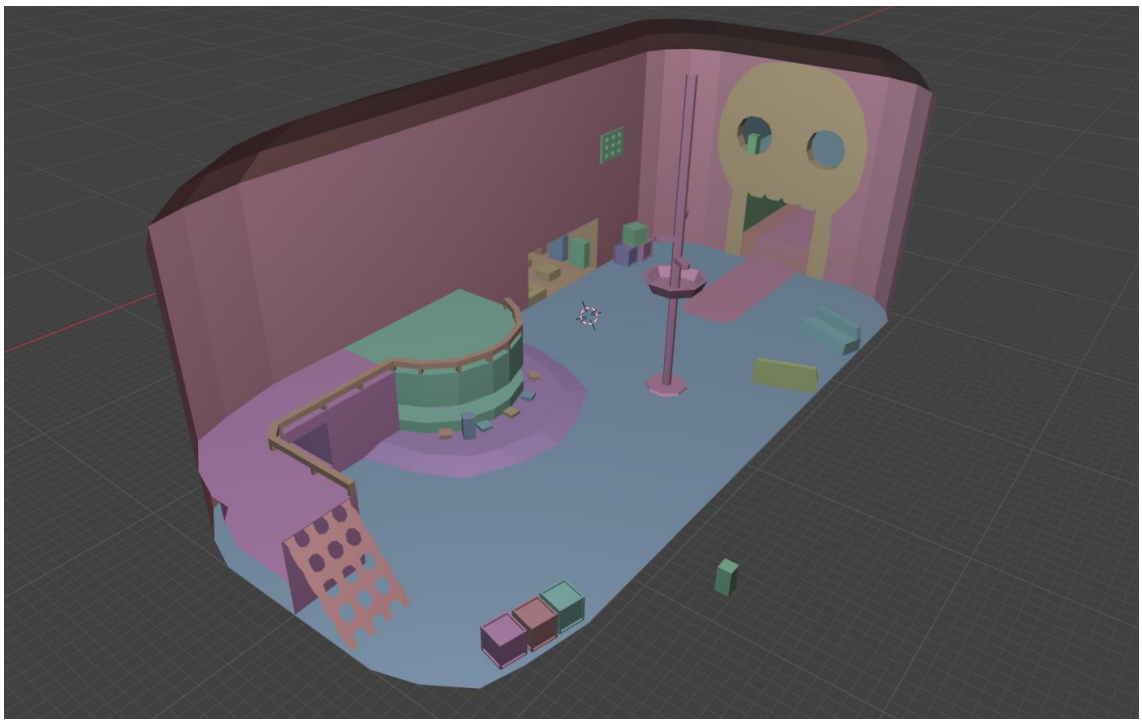


PICTURE 17. Early sketch of key areas in the lobby scene.

Blender was chosen to be the project’s 3D modelling software due to its versatile features for both 3D modelling and texture painting; Blender being a free 3D modelling software was also a factoring point in deciding the program.

The planned art style demanded low poly 3D models with pixel art textures. Creation of pixel art in Blender was achieved by using Texel Density add-on and by changing Blender's own settings. Before starting the prototype's construction, the player character's height needed to be determined to act as a reference point for the rest of the scene's game objects' scales.

The framework for the lobby (Picture 18) was created by using Blender's default cubes without textures. Key areas were placed, and scales were checked with the player's height. After everything was in their rightful place, pixel grid textures were introduced to objects to help point out texture stretching. Objects were then UV unwrapped and appropriate texture sizes were implemented on the objects. The resulting objects were low-poly objects with optimized pixel art textures.



PICTURE 18. Game lobby scenes framework in Blender.

3.1 Compiling the scene in Unity

To bring the lobby scene's 3D model from Blender into Unity, an FBX file had to be created. FBX file format was chosen due to it being well suited for video game purposes. It supports variety of data, including 3D meshes, animations, materials, etc., and is supported by most popular 3D modelling software's and game engines, including Unity, Blender, and Maya (Vivian, M. 2022). Before making an FBX file, individual 3D models had to be treated so that they would be ready to be exported. Transformations and locations were applied to the models in Blender. All the objects 'Origin points' were set in the middle of geometry to make sure objects rotate correctly in Unity. Lastly, individual objects were named appropriately in Blender so that they could be identified easily in Unity's 'Hierarchy' list later. When all the necessary preparations were done to the objects, the FBX file of the lobby was created by pressing 'File' → 'Export' → 'FBX (.fbx)' in Blender.

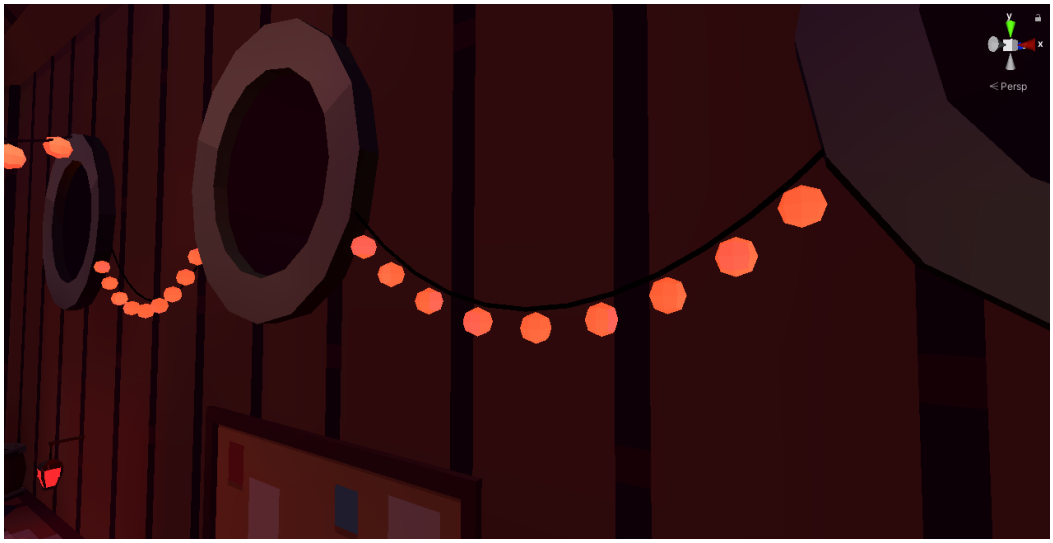
The created FBX file with its texture atlas was then imported into a project in Unity game engine. When the file was imported, animations were unchecked from the FBX file in 'Rig' and 'Animation' tabs to save unnecessary calculations. To help the future Lightmap to behave correctly, 'Generate Lightmap UV's' was also checked on in the 'Model' tab.

3.1.1 Setting up the lights

Before anything was done in *Unity,* 3D models of objects acting as lights sources in the scene were created in Blender. These objects included lamps, lanterns, candles, signs, and different lengths of light festoons. (Picture 19)

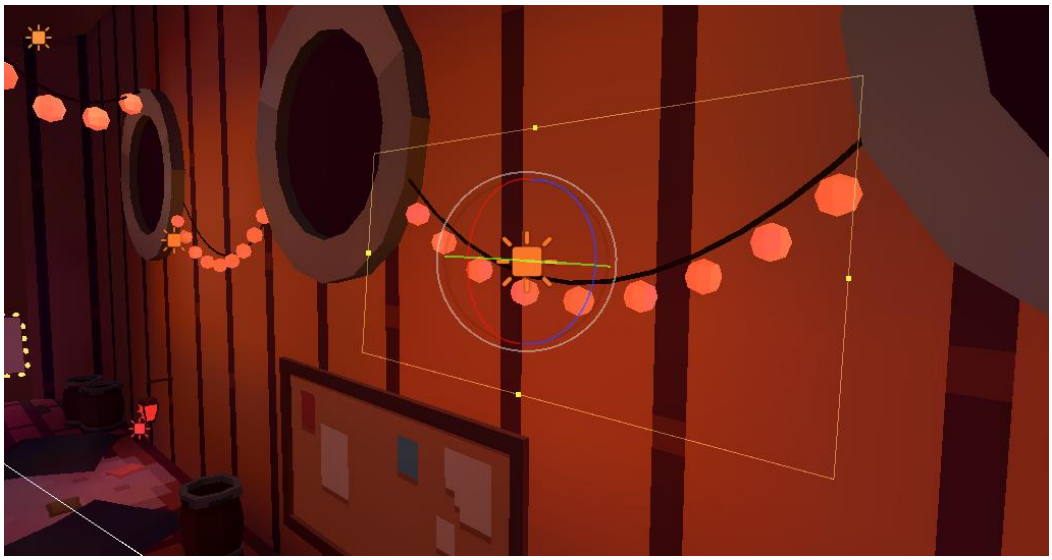
PICTURE 19. Different kind of objects meant to be lights in the lobby.

To stand out from the darkness, emissive materials were created for the objects in Unity by right-clicking assets list → selecting ‘Create’ → and from there selecting ‘Material’. The new materials options were changed in the ‘Inspector’ window; ‘Emissive’ was checked on and under the ‘HDR Colour’ the lights ‘Intensity’ was adjusted to be appropriate for the object. (Picture 20)



PICTURE 20. Emissive material on a light festoon game object.

To create actual illumination to the scene, Unity's lights were used (Picture 21). They were created by selecting 'GameObjects' → 'Light' → and from there, appropriate lights were selected, such as: *area*, *spot*, and *point* lights. All the placed lights were then selected to use 'Baked' mode and were placed throughout the scene.

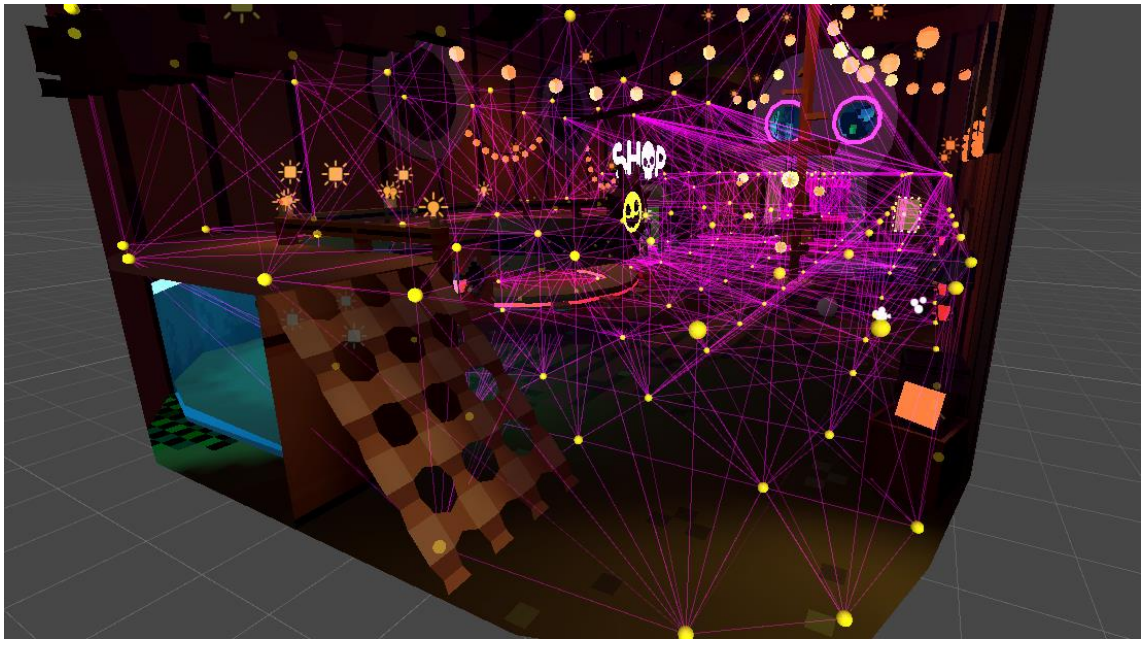


PICTURE 21. Area lights using baked mode illuminating a wall.

3.1.2 Setting up the light probes

Due to the scene's lights all being set to 'baked' mode, they do not provide light to non-static objects. For this reason, a light probe group was created for the scene (Picture 22). The group was created by clicking 'GameObject' → 'Light' →'Light Probe Group' in Unity. The light probes were then placed throughout the scene in light probes edit mode. Placing light probes above surfaces where they could receive light information properly was paid close attention, for if the probes are inside a mesh or are obscured when Lightmaps are calculated, the compromised light probes will not provide accurate light information.

Points of interest where light undergoes strong shifts of intensity or colour were paid close attention by placing increased number of light probes next to each other. This helps lights probes project the changes in the light more convincingly onto moving objects.



PICTURE 22. Light probe group in the game lobby scene in Unity.

3.1.3 Baking the Lightmap

When starting the process of making a Lightmap it is recommended to start with smaller values in the ‘lightmapping settings’ tab, and increasing them as the project goes along, for it helps to make iterations faster. (Unity, 2020b) For example ‘Lightmap Resolution’, it is advised starting with smaller values at first and then increasing them until the light map looks good, instead of starting at value of 40 to 50.

To minimize the Lightmap’s texture size, individual objects ‘Scale in Lightmap’ were changed in ‘Inspector’ window. ‘Lightmap Resolution’ was changed to 6 and ‘Max Lightmap Size’ was set to 512 pixels. To make light look smooth at lower resolutions, Lightmaps filtering was changed to ‘Advanced’ and under advanced settings, ‘Direct Denoiser’ was set to be ‘Optix’ and ‘Direct Filter’ to ‘Gaussian’ filter mode.

The final Lightmap was baked by clicking ‘Window’ → ‘Rendering’ → ‘Lighting’ and clicking ‘Generate Lighting’ at the bottom of the ‘Scene’ tab. The resulting Lightmap was a 256 x 256 file with the size of 128 KB. (Picture 23)

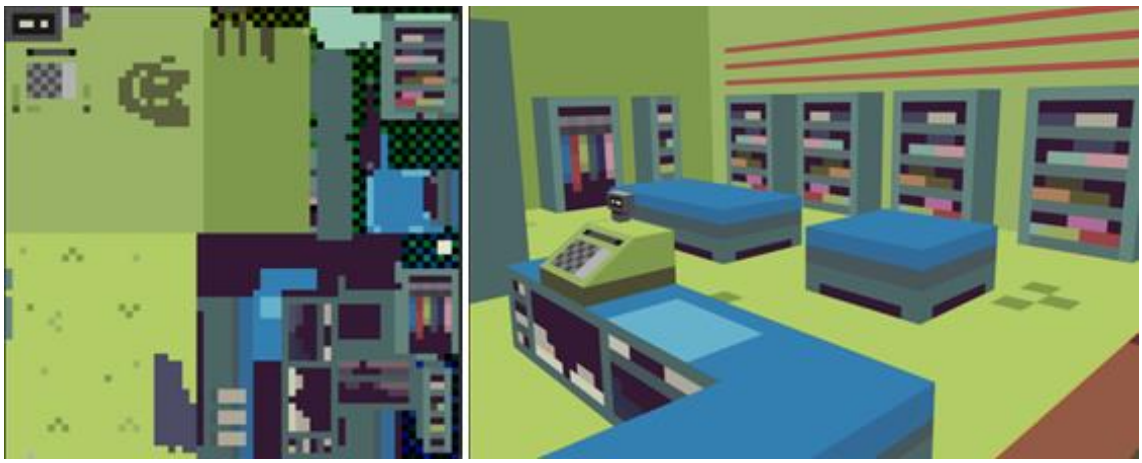
PICTURE 23. On the left of the image is the lobby scene in Unity using baked lights. On the right is the final Lightmap texture.

3.2 Optimization in the scene

To assure that the scene runs smoothly, certain optimization techniques were utilized when compiling the scene in Unity and when creating the 3D game objects in Blender.

3.2.1 Textures

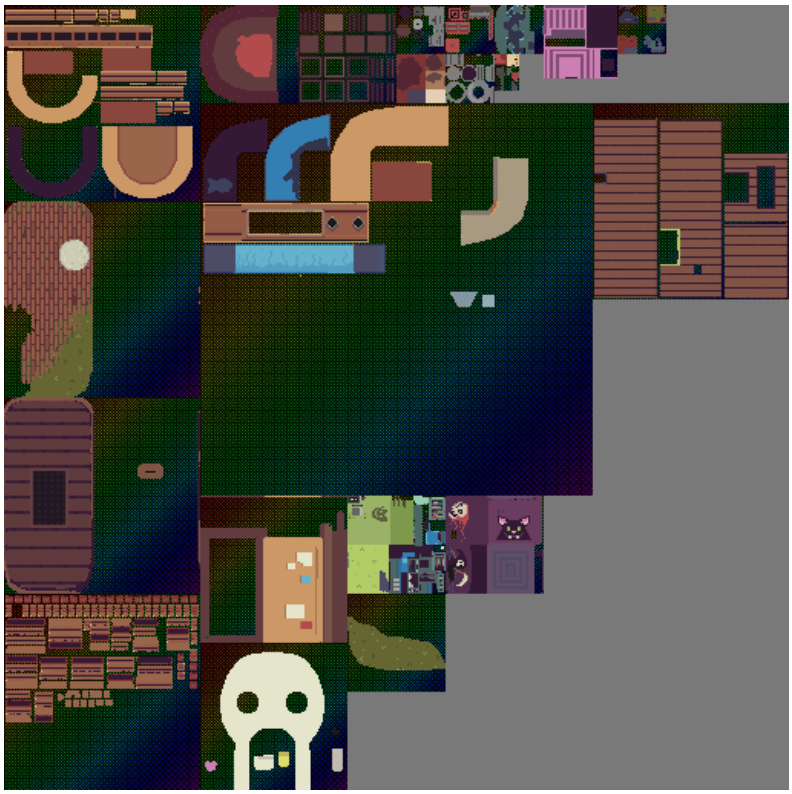
The process of creating textures for the game objects started by placing different sized pixel grid textures that followed power of two sizes (8x8, 16x16, 32x32 etc.) onto a game object to determine the appropriate texture size for said object. Determining an object's texture size always started by placing the smallest texture size first and increasing the size when it was called for. When the texture size was determined, pixel textures were painted onto the game object directly in Blender's 'texture paint' mode. Sometimes, multiple game objects were painted onto the same texture to save space (Picture 24).



PICTURE 24. Game lobby scenes 'shop' area. Left of the picture are all the shop areas textures. Right of the picture is the shop area with textures in Blender.

Power of two texture sizes were used due to them being optimized sizes for digital devices. They take up only the allocated space in memory they have, unlike non power of two textures that, if not compressed into a power of two, would take up more space in memory than is required for the texture; a 32x33 is not power of two size and would take up *64x64* space. (About Game Making, 2021)
When all the textures were painted, they were then collected into a larger image texture called a *texture atlas.* By combining all the different textures into a larger

texture atlas, Unity makes fewer draw calls when drawing the scene, lowering the processing power and draw times needed to draw up the scene in the process. When all the textures were combined, they created together a 512x512 texture atlas (Picture 25).

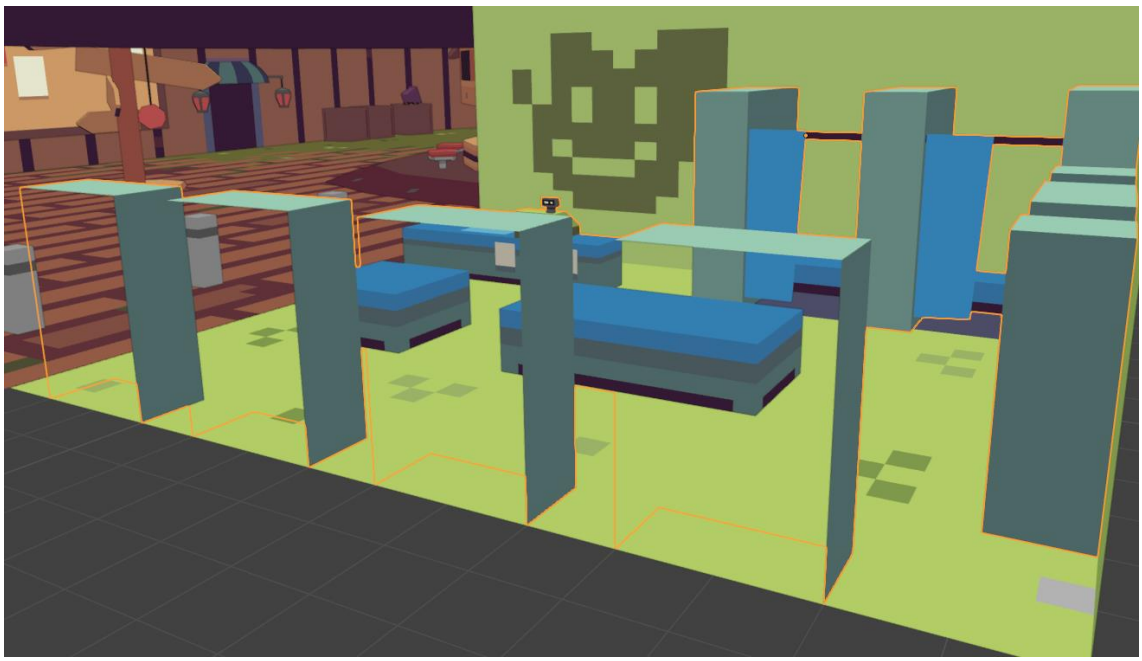


PICTURE 25. Picture of the lobby scene's texture atlas.

3.2.2 Mesh

3D game objects for the lobby scene were created to contain as little geometry as possible while still portraying the desired object convincingly. This was done to save on memory and to lessen the needed processing power to draw the objects in the game.

To lessen unnecessary calculation costs, faces that would not be visible in the scene were removed from the game models (faces facing walls and floors, e.g.). Back-face culling was also used in the project so that the faces would be drawn only from one side, thus minimizing more drawing calculations (Picture 26).



PICTURE 26. Lobby's shop area view in Blender.

3.2.3 Batching

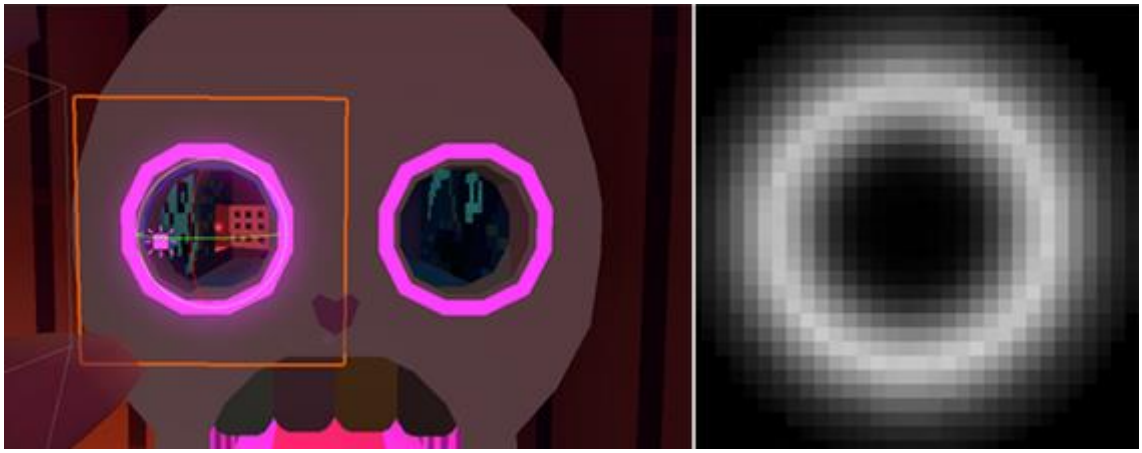
Before bringing objects into Unity, some of the game objects meshes were manually combined in Blender, this makes Unity draw the game objects at the same time, lowering future draw calls.

Stationary objects in the scene were marked as 'Static' in Unity's Inspector window, to make use of Unity's draw call batching technique called *'Static Batching'*. When objects are marked as *'static'* Unity will combine them together into bigger meshes and renders them together. (Unity Documentation, 2021g)

Game objects using the same material will also be batched together in Unity (Unity Documentation, 2021a). For this reason, the lobby scenes texture atlas was created, and the game objects were then UV unwrapped to use the same material using the texture atlas.

3.2.4 Tricks

To create an illusion of glow effect, quads with black and white gradient textures were placed behind certain game objects, acting as light sources in the scene. The quads' gradient textures were then changed to use Unity's 'Particles/Standard Unlit' shader and their rendering mode was changed to 'Additive'. Lastly, the texture's colour mode was changed to 'Overlay'. The resulting effect is a see-through texture with a slight glowing effect without using any post-processing effects (Picture 27). When the quads were placed in appropriate places throughout the scene, a billboard script was introduced into them, making the quads always face the player's camera. Without this effect, the 2D quads would not move, and the players could break the illusion of a three-dimensional glow effect simply by moving to the quads side.



PICTURE 27. Left of the picture behind the neon pink circle is a quad with a black and white texture creating a glow effect. On the right of the picture is the black and white texture that the quad uses.

4 DISCUSSION

When researching this topic and making the practical project, it became abundantly clear that light is an integral part of video games and should be paid close attention and utilized appropriately when creating video games. The value of knowledge on lighting techniques and colour theory in art became also clear when researching this topic and creating the project.

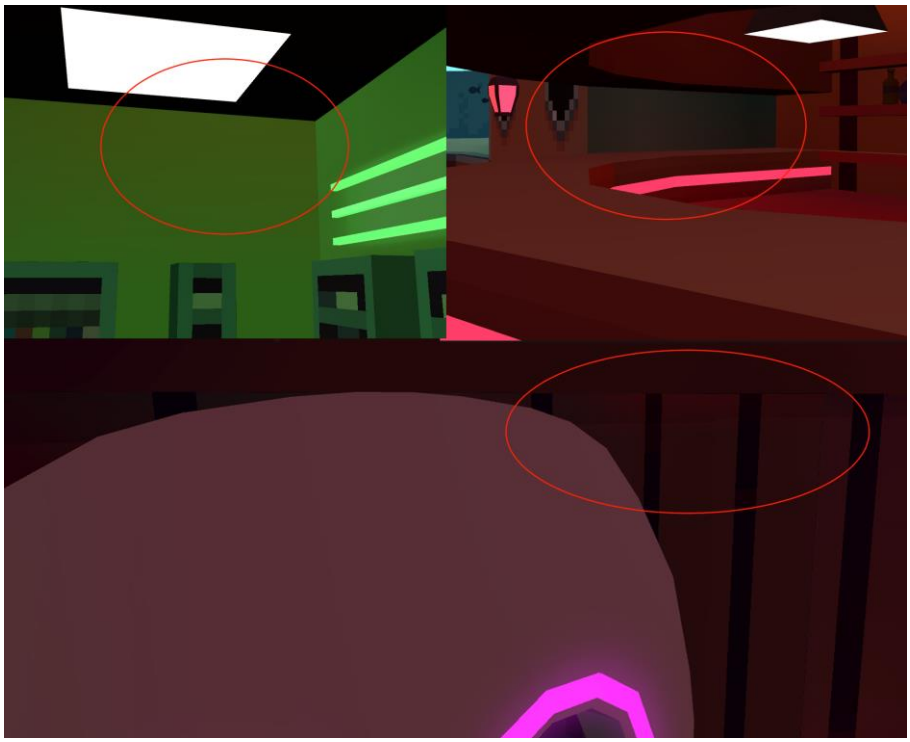
The importance of understanding proper practices of creating good and optimized lighting for mobile games became clear when researching limitations on mobile devices. Creation of the lights for the practical project backed up this notion as well.

4.1 Results

The result of the practical project was a fully lit 3D game lobby scene with pixel art textures that the players could traverse through and explore. The game scene benefitted greatly from lighting, creating a mood for the scene that could not be achieved without lights (Appendix 1). Different coloured lights gave personality to parts of the lobby, differentiating them from each other, and the lights acted at the same time as indicators for players to where to look at.

The low resolution of the final Lightmap and the use of the Gaussian blur in the filtering in Unity resulted in a smooth and believable looking light (Picture 28). Although, compressing the Lightmap to such low resolutions did result in some unwanted artifacts and imperfections in the final texture. These imperfections can be observed in picture 29, including slight hues of red in the shop area's green walls, the bar's red lights getting interrupted by a blue light, and in the roof of the lobby, a slight light can be seen shining where darkness should be. The low resolution also made it challenging to precisely control lights in the scene. Making harsh lights with hard edges, for example, was impossible. This effect directly hindered the result of an industrial light meant to light up a corner in the scene with harsh clear light.

PICTURE 28. Section of the lobby scene from bird's eye perspective. Smooth colourful lighting from different sections of the lobby can be seen mixing on the floor.



PICTURE 29. Artifacts and imperfections in the Lightmap.

The texture atlas for the scene was optimized and worked well, but the texture ultimately had a lot of wasted space that could be cleaned up to free up space. While it is good to keep textures to the power of two, individual textures inside the texture atlas do not need to follow this rule, for the larger atlas image is already in the power of two. Keeping the individual textures to the power two resulted in larger game objects taking up a lot of unnecessary space in the atlas. This was also the result of a messy UV unwrapping in Blender. A lot of overlapping islands were used when UV unwrapping objects and creating textures for them. This technique allows different sides of an object to use the same textures and minimizes texture sizes, but it can result in problems when making Lightmaps. Generally, every side of the object should have their own space in UVs to ensure no bleeding from different lights happens in the Lightmap. This can be avoided, and was avoided in this project, by creating Lightmap UV's in Unity, but to save up more on space creating UV maps for assets with Lightmaps in mind is a good practice.

The creation of the practical project brought a lot of hand on knowledge about Unity Game Engine. Including how Unity's lights work, how to optimize them, how Unity's batching systems work, and how those habits can be used to optimize game projects. Working on the project also offered a chance to learn how to create and handle game-ready objects in 3D modelling software's and how to prepare them for game engines.

REFERENCES

About Game Making. 2021. Unity : Why the resolution of images should be a power of two. Mar 23, 2021. Watched on 13.3.2023.
https://youtu.be/-D4iNFYZnOQ

Alscher, D. 2019. The 6 Color Schemes to Keep Everything Picture Perfect. Published on June 6, 2019. Read on 11.4.2023.
https://www.g2.com/articles/color-schemes

Art In Context. 2022. Warm Colors – What Is the Difference Between Cool and Warm Colors?. Published on September 8, 2022. Read on 6.4.2023.
https://artincontext.org/warm-colors/

Bacioiu, C. Why do most mobile games run at 30 FPS? » Best Smartphone Games. Read on 8.12.2022.
https://bestsmartphone.games/why-do-most-mobile-games-run-at-30-fps/

Branko, G. What Is A Good FPS For Gaming?. GPU Mag. 1.10.2022. Read on 27.2.2023.
https://www.gpumag.com/good-fps-for-gaming/

Clement, J. 2021. Number of mobile gaming users worldwide in 2021, by region. Survey. Published on Sep 7, 2021. Read on 28.3.2023.
https://www.statista.com/statistics/512112/number-mobile-gamers-world-by-region/

Depew, A. 2021. Types of Lighting in Film: Basic Techniques to Know. Released on April 5, 2021. Read on 5.4.2023.
https://www.adorama.com/alc/basic-cinematography-lighting-techniques/

Foundry. 2020. Leveling up by light: a look at lighting in video games. Released on July 17, 2020. Read on 8.12.2022
https://www.foundry.com/insights/film-tv/lighting-video-games-8-bit-bulb

Houzé, R. 2019. What is the effect of the lighting design process on game aesthetics and its influence on the gaming experience?. Study. Read on 8.12.2022.
https://www.gamedeveloper.com/design/what-is-the-effect-of-the-lighting-design-process-on-game-aesthetics-and-its-influence-on-the-gaming-experience-

Larson, E. 2023. How to Best Use Warm and Cool Colors in Your Art (and Painting in 2023). Last Updated on March 13, 2023. Read on 11.4.2023.
https://artstudiolife.com/warm-and-cool-colors/

Massive Entertainment. 2021. Shining a light on light in video games - The Fika Sessions [Episode 7]. Feb 17, 2021. Watched on 2.12.2022.
https://www.youtube.com/watch?v=ez5qOlMidAs

Scott, D. 2019. Color Schemes in Art. Published on March 7, 2019. Read on 11.4.2023
https://drawpaintacademy.com/color-schemes/

Unity Learn. 2019a. Configuring light probes. [Tutorial]. last updated on 12.10.2020. Read on 2.12.2022.
https://learn.unity.com/tutorial/configuring-light-probes-2019-3#

Unity Learn. 2019b. Introduction to Lighting and Rendering. [Tutorial]. Last updated on 2.4.2020. Read on 30.1.2023.

https://learn.unity.com/tutorial/introduction-to-lighting-and-rendering#5c7f8528edbc2a002053b528

Unity. 2020a. Harnessing Light with URP and the GPU Lightmapper | Unite Now 2020. Jul 22, 2020. Watched on 2.1.2023.
https://youtu.be/hMnetI4-dNY

Unity. 2020b. How to build Lightmaps in Unity 2020.1 | Tutorial. Nov 9, 2020. Watched on 3.3.2023.
https://youtu.be/KJ4fl-KBDR8

Unity. 2020c. Light baked Prefabs & other tips to get 60 fps on low-end phones. Read on 8.12.2022.
https://unity.com/how-to/advanced/optimize-lighting-mobile-games?utm_source=demand-gen&utm_medium=pdf&utm_campaign=asset-links-gmg-choose-unity-for-mobile&utm_content=optimize-mobile-game-performance-ebook

Unity. 2021. Optimize Your Mobile Game Performance. [E-book]. Read on 8.12.2022.
https://create.unity.com/optimize-mobile-game-eBook

Unity Documentation. 2021a. Draw call batching. [Online manual]. Read on 14.3.2023.
https://docs.unity3d.com/Manual/DrawCallBatching.html

Unity Documentation. 2021b. Lighting Mode: Baked Indirect. [Online Manual]. Read on 6.2.2023.
https://docs.unity3d.com/Manual/LightMode-Mixed-BakedIndirect.html

Unity Documentation. 2021c. Light Mode: Shadowmask. [Online Manual]. Read on 6.2.2023.
https://docs.unity3d.com/Manual/LightMode-Mixed-Shadowmask.html

Unity Documentation. 2021d. Lighting Mode: Subtractive. [Online Manual]. Read on 6.2.2023.
https://docs.unity3d.com/Manual/LightMode-Mixed-Subtractive.html

Unity Documentation. 2021e. Light probes. [Online manual]. Read on 30.11.2022.
https://docs.unity3d.com/Manual/LightProbes.html

Unity Documentation. 2021f. Realtime Global Illumination using Enlighten. [Online manual]. Read on 6.2.2023.
https://docs.unity3d.com/Manual/realtime-gi-using-enlighten.html

Unity Documentation. 2021g. Static batching. [Online manual]. Read on 15.3.2023.
https://docs.unity3d.com/Manual/static-batching.html

Vertex School. 2022. Is Video Game Lighting Actually Important?. Article. Read on 2.12.2022.
https://blog.vertexschool.com/is-video-game-lighting-actually-important/

Vivian, M. 2022. Everything You Need to Know About FBX Files: A Comprehensive Guide. Blog. Read on 27.3.2023.
https://vection-technologies.com/blog/Everything-You-Need-to-Know-About-FBX-Files-A-Comprehensive-Guide/

APPENDICES

Appendix 1. Lobby demo video

https://youtu.be/BwiXG77q5b4