Trung Le Dang

Level Designing in Game Engine

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Engineer

Bachelor's Thesis

23.04.2017

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Trung Le Dang Level designing in game engine 52 pages 23.04.2017
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	Digital Media
Instructor(s)	Antti Laiho, Senior Lecturer Petteri Kolehmainen, CEO Make Helsinki Joonas Turunen, COO Make Helsinki

Level designer is an important position in game development process. However, it requires expertise, which only can be adopted through work experience. The goal of this project is to create a quality product using Unreal Engine.

The project is divided into two parts. The first part focuses on the core requirements of creating a quality level game map. The second part focuses on the research project. The task was given by the customer who ordered a scene trailer. The study describes the process step by step from the beginning to the final product. The final product can also be previewed at an online address provided in this thesis.

The project was carried out by using Unreal Engine 4.13 version, Autodesk 3D max and Adobe Premiere CSS 5.5. Epic Games, which is a software company, provides Unreal Engine 4 which is free of charge. There are also various assets provided by Unreal Engine and the author's personal projects. The final product was rendered as images and imported into Adobe Premiere for future use.

As the result of this thesis, a complete level with animations was created by using Unreal Engine 4 in three and half weeks. Most of the time was spent on creating the map as well as rendering the product.

Keywords	Level Design, Sculpting, Foliage, UE4, Adobe Premiere, Blueprints, Game engine

Contents

Abbreviation and Terms

Sculpting	Designing an object by using mouse or pen as sculpting tools.
Blueprint	A system which supports users to use programming skills in the easiest way in order to running various effect which are not able to do in other 3D software. For example, creating particle effect, creating water flowing animation, wind, etc.
RAM	Random Access Memory.
GPU	Graphic Process Unit.
HLM	High Level Map.
LLM	Low Level Map.
Foliage	A combination of objective models which could be placed on map via painting mode.

1. Introduction

Nowadays, competition between game companies is hard everywhere.
Therefore, new products benefit the owner. Game engines have been around for over fifteen years but because the industry is growing rapidly, developers must expand their knowledge in order to stay in business.

Level designers, in particular, have an important role inside the game industry, since they are responsible for creating game rules as well as game concepts.
However, designing a product is a complicated process. There are steps which should be followed in order to maximize the outcome and minimize the time spent. One purpose of this study is to describe how I learned about level design during my internship in a project called Make Helsinki in October 2015.

2. Level Design in theoretical framework.

2.1. History of Level Design

In game develop, the person responsible for constructing the game spaces land letting the players to compete, is called a level designer, or a map designer. The level designer is mainly responsible for implementing the game play.

The term *level designer* is somewhat misleading. Originally the word *level* stood for some perks and traits one could gain during playing. A game consists of different levels of difficulty, starting from level one. Step by step, each level becomes more difficult than the previous one.

In 1970's and early 1980's, level designing stages were no doubt simpler than nowadays (Figure 1). The game *Breakout* was created for arcade game machines and it was one of the first two dimensional games with rectangle shapes. At the first level, a player was provided certain amount of color bricks to eliminate by hitting a ball. After reaching the goal, players advanced to the next level, which was more challenging; more color bricks appeared when there was an improvement in the ball's speed. The game map was two dimensional. See Figure 1.

Figure 1. The first level of old school *Breakout* 1970's – 1980's [8.]

In later game versions, when a player advanced into a higher level of gameplay, the screen would be reset or words such as *level two*, *level three* on the screen.

However, game developers were not satisfied with that formula because it was too simple. They decided to improve the game with subtle changes and transitions to a new

level were seamless. As the result, individual levels could be very complex, with a storyline blended in. Also, the player's choices could affect the outcome of end game.

The transition from two dimensions into three-dimensions was an important improvement in game developing industry. In 1983, a release of a three dimensions computer game-*Battlezone,* turned the history of level designing onto a new page of achievement (Figure 2). Although models and environments were made from rectangle shapes, they also displayed the movement in 3D. *Battlezone* became successful because it represented the first use of polygonal environments and obstacles in computers, along with providing players the ability to move through gameplay space, accordingly to X and Y axis [13]. (See Figure 2)

Figure 2. *Battlezone* gameplay in 1983. One of the first 3D games, which was created for Atari [13.]

The game provided not only basic of polygon environments but also all sides of a polygonal object, which were visible all the time. This made the game to look more futuristic.

2.2. 2D and 3D Level Design in Game Engine

Two dimensional games has been frequently one of the favourite styles for game developers ever since 1970's (Figure 3, Figure 4). Game Engine (Unity, Cry Engine, Unreal Engine) offered new ways to develop two dimensional games.

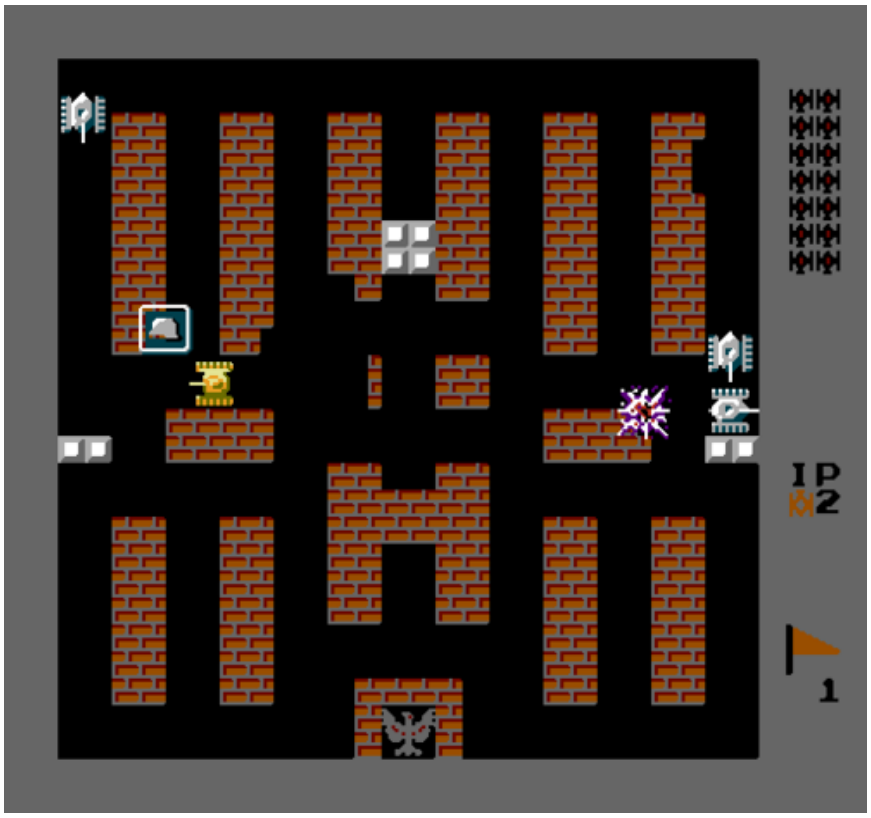


Figure 3. *Battle City Tanks* game in 1990's [14.]

Battle City was a two dimensional strategy top down view game for NES game console. In the game, players took role as alliance forces who were defending their base from the invasion of an enemy army. The User interface indicator was placed on the right side of the screen, showing to player the amount of enemies as well as player's scores [14].

There were several ways to make two dimensional games. Two of the most popular styles were *the Top Down view* (Figure 4) and *the Side view* (Figure 5).
In *The Top Down view*, a camera would be placed on the top and pointed down straight 90 degrees, capturing player's gameplay or even following an in-game specific character appointed to be a player. In *the Side view* game, however, a camera was at one spot or moving only with X's axis coordinates.

The Super Mario World, made by Nintendo was one of the best two dimensional side view games. Nintendo had made an effort to design the map, game play concepts and game logics as well as possible. The result was a very successful and many players became attracted to this new genre [16]. Even today many games are inspired by Super Mario World.

Figure 4. *Super Mario World* game in 1990's [16].

In game developing industry, those designing two dimensional games often aim at a mobile platform release because two dimension games require less power resources.

Recently, developers prefer to use a free style camera with special algorithms to detect and follow all actions of players. The *Angry Birds* game is an example of this (Figure 5) [17].

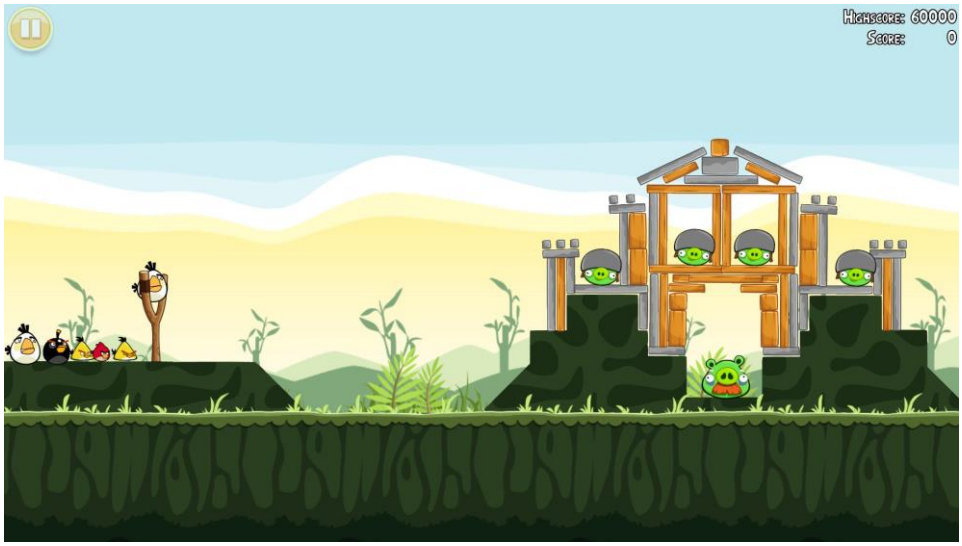


Figure 5. *Angry Birds* gameplay. The most downloaded mobile game ever, made by Rovio [9.]

Inside game play, the camera stays in focus on the slingshot. The player can zoom in and out or calculate the distance between the slingshots to targets. When players

give the command to shoot the birds, the camera is automatically attached and follows the flying birds. Then the camera is reset back to the slingshot so players can continue to aim and shoot.

When designing a level in two dimension games, designers can use two and three dimension objects, models and assets to create the map. After that, they can apply two dimension physics which only effect the X and Y axis. Modelling characters, objects and assets for two dimension games is usually less complex compared with three dimensional games because the game view is only from one perspective. For example, in a three dimensional game, a simple box is modelled from six sides. Textures are modelled on each side which increases the usage of computer memory when importing the box in the Game Engine.

In a two dimensional game, the box would only need to be modelled and textures applied on three sides (for *Top Down* view) or one side (*Side* view). This can cut the costs even by 50 percent. Figures 6 and 7 show how a city environment has been created for a two dimensional game. If the game was based on *Side* view, it would be simple because texture is added on one side only.

Figure 6. Two dimension *Side* view game back ground. Created by Mohit Kumar Rao [10.]

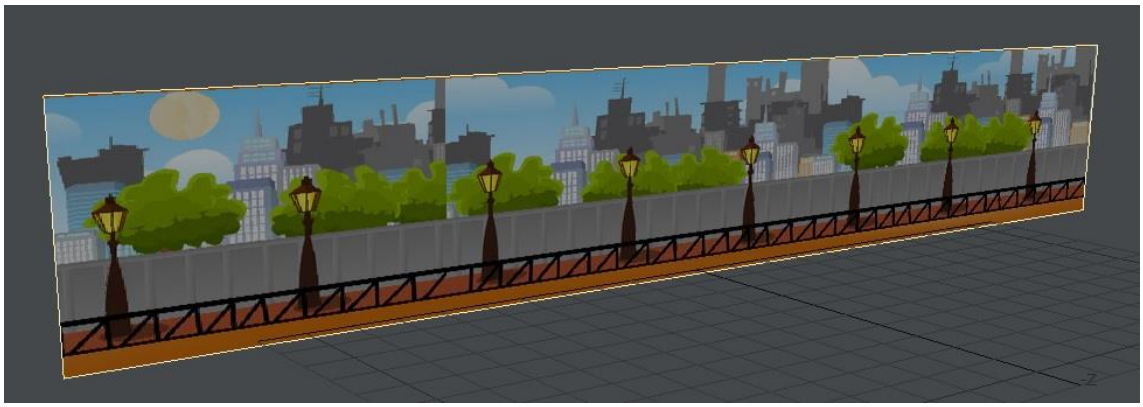


Figure 7. A plane was used for re-creating back ground for two dimension game in Autodesk 3D max.

Three dimension games require more complex models, characters and assets as well as skills to create. Based on the game's genres (such as RPG, MMORPG, Turn-base and MOBA), models are created to fit the demand.

3D level design as well as playing the games requires a lot of computer memory. Developers usually try to limit the polygons count in models and environments. Often they make different options for a specific model. When a main human character is created for a game, the details must be of high quality. As a result, one model can consist of 600 000 to over one or two millions polygons when finished. As illustrated in Figure 8, the enormous amount of polygons consists mostly of details such as hair, scars or details in uniform [12]. The cowboy model was modelled very complex in the beginning but at the end, the polygon count was significantly lower. Those two options, high and low polygon, could be used as close-ups or seen from the distance in a game.

Figure 8. High polygons versus low polygons [12.]

To make sure that millions moving polygons do not ruin the player's fun, developers must retopology that model down to less than one hundred thousand polygons, plus, divide textures into usually three options, called *Level of Detail* (LOD, such as LOD 0, LOD 1, LOD 2). See Figure 9.

At zero LOD, the object in Figure 9 provided the best quality and texture resolution. This first option was mainly used to render the game trailer, make game cut scenes

cinematic and provide various close-up cameras. The first LOD could be skipped due to poor efficiency.

The second LOD provided the lowest resolution of quality both of model and texture, so this LOD would be used for far distance view since the human eye can not recognize the difference between a million-polygon model with 2000 resolution textures and a 100 000-polygon model with high quality resolution texture at certain far distance of view [18].

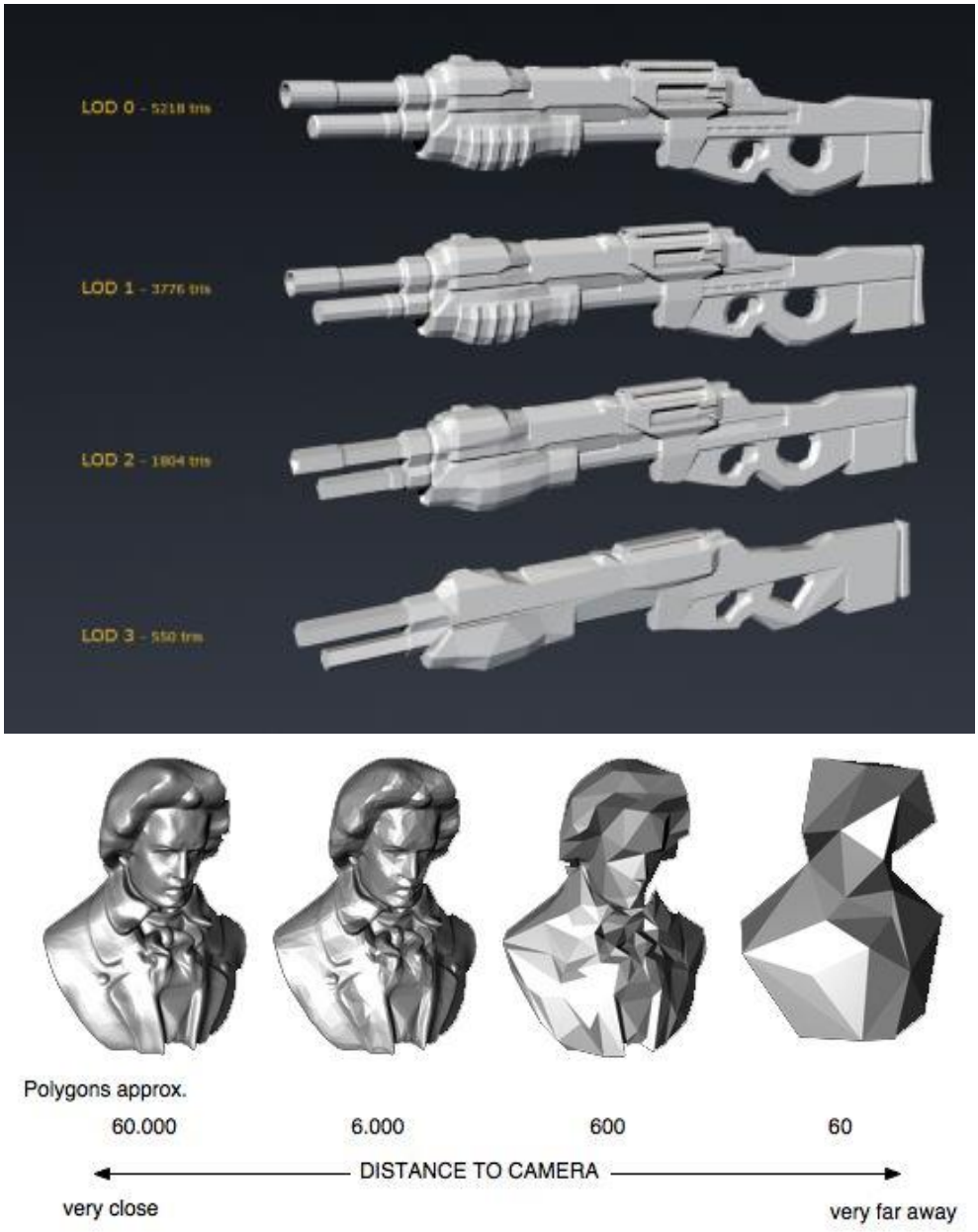


Figure 9. Level of Detail in difference. Another example of close and far distance view between complex and simple level of detail [18].

These levels of detail systems are important for level designers because. If they can master this workflow, the outcome will be of good quality without consuming the hardware too much [20].

2.3. Game Engine and setting up

There are numbers of game engines such as Cry Engine and Unity which provide fully professional functions for game development [1]. However, Unreal Engine in general and Unreal Engine 4 in particular were selected as the main game engine for this thesis.

In 1998 the company Epic Games, released the game Unreal. This marked the birth of one of biggest game engines (Figure 10). So far, there have been four iterations of the Unreal Engine [23].

Unreal Engine one provided the basic functions of making games, such as integrated rendering, collision detection, artificial intelligence and networking. This engine version relied on simple collision detection to achieve good frame rates 1998 hardware [24].

Figure 10. The Unreal's game level map was created in full combination between surprises plus scary elements which constantly provided to players the feeling of loneliness surviving in alien world [13.]

Unreal Engine two introduced a vast improvement in the rendering engine. This version also included the Karma physics SDK, known as ragdoll effects. The engine supported also current consoles such as Playstation 2, GameCube and Xbox [24].

Unreal Engine three introduced new functions for developers since it had incorporated a wide variety of technology such as Scripting, Physics, Editors, Animation and particle system. Written in C++, using the OpenGL and Direct X Graphics API, the third Unreal Engine featured an object-oriented design. Furthermore, one of the engine's goals was to make content creation and programming as easy as possible. As a result, the engine helped artists to develop assets with minimal programming assistance and give the programmers an extensible framework to create graphic games with [24].

Unreal Engine 4, the latest version of the engine, supports DirectX 9, 10, 11 PCs, Xbox360, Playstation 3 and Playstation 4 along with Virtual Reality such as Google cardboard and Samsung Gear VR. Most of the tools available in the engine, are based on previous tools but with upgrades in graphics, rendering and especially in light reflection as well as other physical systems [24]. The game Unreal Tournament was made using latest Unreal Engine version (Figure 11). With modern technology such as complex particles (smoke, explosion, electric effect), the game brought to players a fast paced first-person-shooting duel style. The level map was created in detail and it indicated directions clearly, because players did not have enough time to figure out which way they should run [14].

Figure 11. Unreal Tournament developed by Unreal Engine four nowadays. [14].

In order to develop a specific game, developers in general and level designers in particular would need to install Unreal Engine 4 via the official Unreal Engine website. At the main menu of Unreal Engine 4 editor client, users can choose the best version for their needs. So far, there have been 14 versions of Unreal Engine 4, the latest ones are 4.0 to 4.13. [25].

3. Essential requirements in Level Design

Even a veteran level designer must follow all the requirements to create one specific map or environment. A good level designer is also an environment artist be who creates all the assets included when making the game map.

A bad game map can ruin the whole game. To create a flawless map there can be no wrong twitches of light or placement of objectives. To sum up, setting up a blueprint sketch, building up assets, and designing right lighting during post process are all necessary steps in creating a game map.

3.1. Good Level Design

Good level design has many different approaches, although the basic elements are the same. There can be a robotic constructing level approach, or a fantasy and unrealistic approach. Sometimes even ideas that make no sense are included to satisfy the gamers. Each approach has its advantages, but when it comes to algorithm, they all follow the core elements of good level. Those core elements are presented below [5].

The good level design must be fun to navigate

In most cases, the player's core expectation and judgment are based on interacting with the map through navigation. Mastery layout, a touch of lighting, blinking signage and other visual cues create a natural flow to the level, which leads the player instinctively to follow it (Figure 12). From an aesthetic aspect view, a game's map should be stitched and worked all together to provide a consistent visual language by the use of colour and forms, so the player could learn, experience and be able to achieve their own progress intuitively through the level [2].

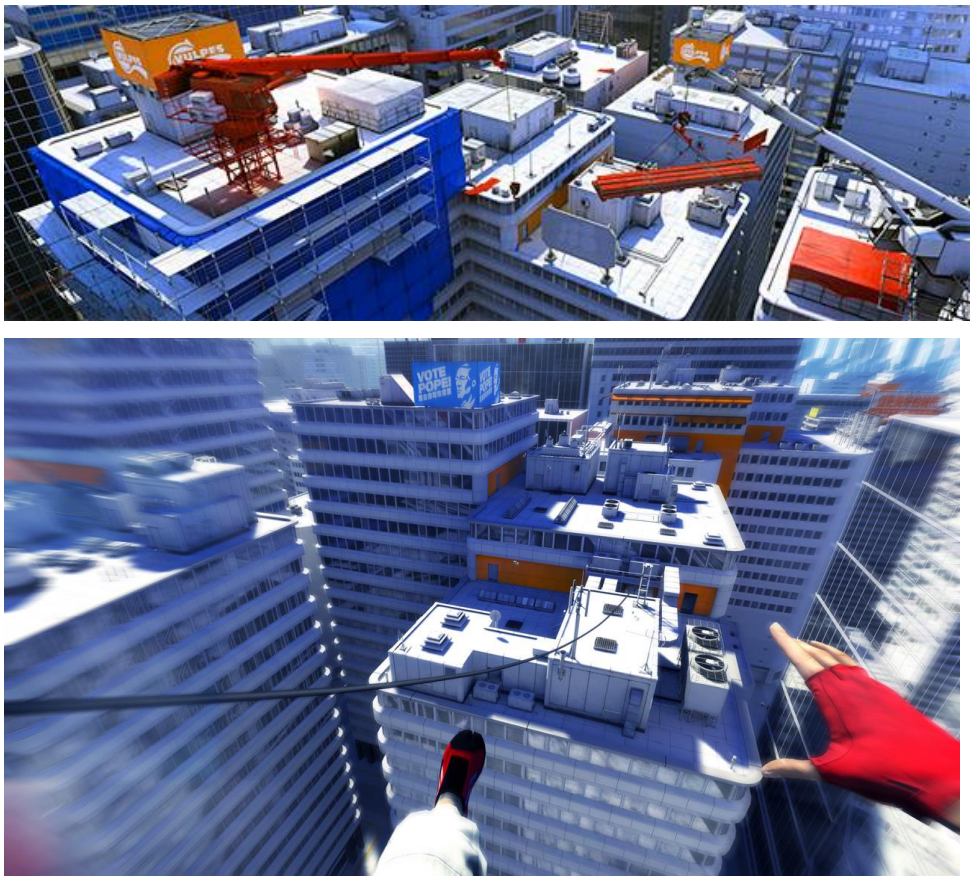


Figure 12. *Mirror Edge,* the game had brought to players the feeling of constantly keep moving in order to survive [15].

Contrary of those obvious guideline for path finding, various level designers also apply twitches of trickery to add depths, or create areas and hideouts. These cause players to feel lost and confused, bringing a sense of dramatic tension.

The good level design must not rely on words to tell stories

Solving puzzles or completing a broken circle are good ways of level design. A level designers can create a circle, but leave a broken gap for the players to fill in themselves. However, if the gap is too small, the players will not notice it; too big and it causes losing the players, who are not able to connect the circle [2;5].

There are three aspects to create the circle and the gap in a game level.

- The Explicit way; it is cleared and sent straight forwards to players by text or speech such as mission objectives or cut scenes.
- The Implicit way; it is told via the environments through gameplay. It must be carefully place at obvious area where players can notice it (Figure 13).

- The Emergent way; it is the player's own experience as they go through each level.

The aspects above tend to stimulate and trigger the player's intelligence. They also allow players to fill in the gap with their own actions and imagination as shown in Figure 14. [5].

Figure 13. A message was delivered to players whilst their exploration in gameplay [26].

Figure 14. *Hitman,* the game provided free-style game rules for players. Either players could go gun-blazing like Rambo or sneak around like a ninja to assassinate the target [27.]

The good level design informs the player what to do, but not how to do it

Offering players the power to experience and explore their own story through game-play is one of main targets of developers. This is typically created by simple, explicit, text- based objectives, waypoint markers or any other navigation aids in order to prevent the players to stumble into any doubts [4].

Those level objectives must be visually distinct, using any forms, lighting, animations or locations to clearly stand out from other surrounding environments. The key to make players fully excited each step is to create more open-end objectives. In other words, there must not only be an ultimate objective such as a destroyed place or eliminated target, but also a free choice of how to complete those objectives. (Figure 15)

These requirements are the key, because in one scene of game play players can use all arsenals of items and environments to complete the mission. [2.]

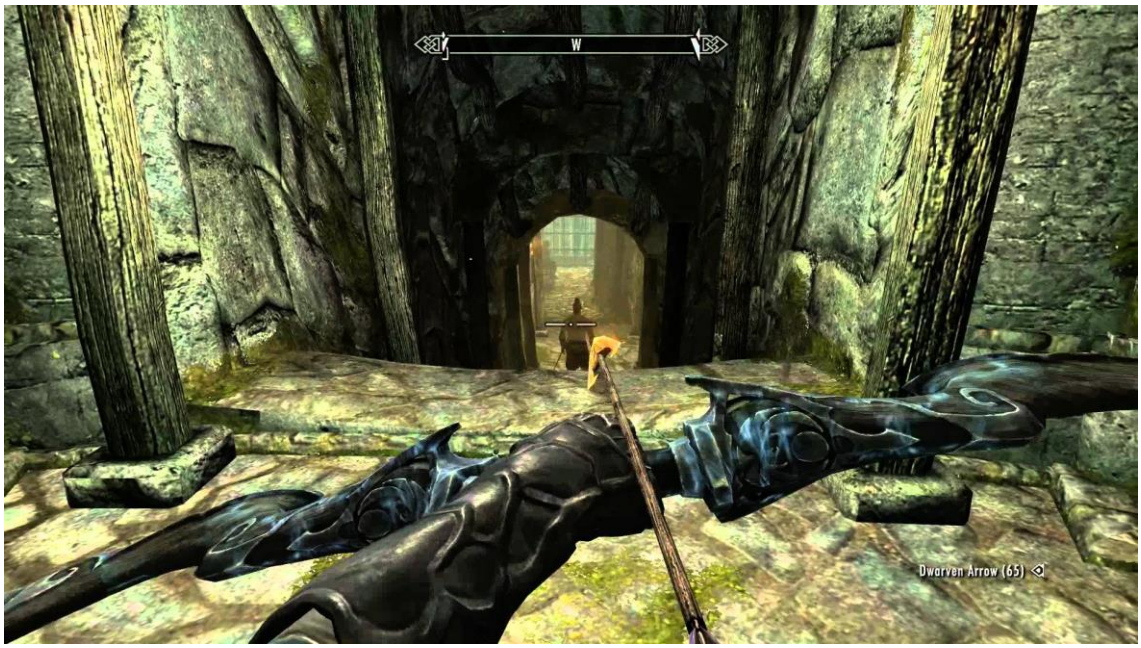


Figure 15. *Skyrim*, the innovated game which let players fully to select the ways to complete the objectives [28].

The good level design, teaches players something new while they play

There are various games that provide real life knowledge to players. When players actively play their game, they also learn new things. Sometime, the learning part in gameplay can be hard in order to process. If players understand the patterns and master the mechanics too easily, they will quickly become bored and stop playing. Therefore, level designers can put in various challenging missions which only can be completed by using specific items. Those items can be acquired via applying real life knowledge [2;6].

For example, a player needs to escape an underground cave where the main door is covered by unpassable trees and vines. By applying real life knowledge, the player should recognize that vines and trees would be destroyed by fire and search for items to burn vines in order to escape.

On the other hand, if the player does not know that, they would struggle to find out solutions by themselves, even throwing rocks or mashing their body to vines and trees hopelessly, until they discovered how fire was useful in that case. Thus, they learn new things through game play. In the future, whenever they encounter vines and trees, the word “fire” could first come in their mind.

Good level design empowers the players

Virtual world in general and video games in particular are made to escape the limited power in real life. Good level designers must not make players do things that they could do easily in real life, but encourage them to experience new things, which they would not or definitely could not do in their real life. A mission objective of a well-designed game should eliminate boring repetitive chores, and create interesting and exciting ones. [4].

In order to grant players true empowerment, their actions must have noticeable effects on the game world. For example, in a magical game world, if players choose “fire” element instead of “ice”, their path is bound to destruction. If players encounter lakes or liquid areas, where they might be frozen by using ice power, they must find another way to cross the path.

In various historical series games such as Medal of Honor, Company of Heroes, Call of Duty, there are also secondary objectives that sometimes provide more fun and breath taking than the main mission. Blending objectives could grant players the feeling of being a powerful hero [29].

Good level design supports the player to control the difficulty

Controlling the difficulty, sounds like a responsibility of programmers since this is one of the game system mechanics. However, a good level designer knows that blending the game's map, which runs parallel with the difficulty adjustment, would boost players experience highly up.

There are various mechanics to show players the difference of difficulty. By implementing a dynamic difficulty, most players notice the changes of areas, where enemies become more powerful and treasure more valuable. That kind of dynamic difficulty is applied by Massive Multiplayer Online Role Playing Game (MMORPG) since those mechanics turn different locations on one map into a seamless world [5].

However, the mentioned mechanics above were not always available. A well designed level must also let players manage difficulty themselves, through clever usage of risk and reward. The basic path through game missions should be properly paced for a player of moderate ability, with the appropriate peaks and troughs of challenge, but there should be specific areas off the main path, with clear opportunities for veteran players (Figure 16). As the result, whenever players make a path choice, they would feel the heat and responsibility of their choice. Therefore, big rewards come with mountains of risk.

For example, in the *Skyrim* game, player would often encounter various treasure chests which are located in visible but unreachable areas. Therefore, players need to have certain skills such as teleportation in order to claim the chest. A good level design map would content a chain link reactions which is triggered once players entered.

The following chart explains the relationship between link and chain reaction. Once player receives mission objective, they need to complete the puzzle and challenges in order to receive the standard rewards. Therefore, they must be able to unlock side missions with extra rewards. Those elements could be linked together in game play and once, one of them is triggered, the rest would follow.

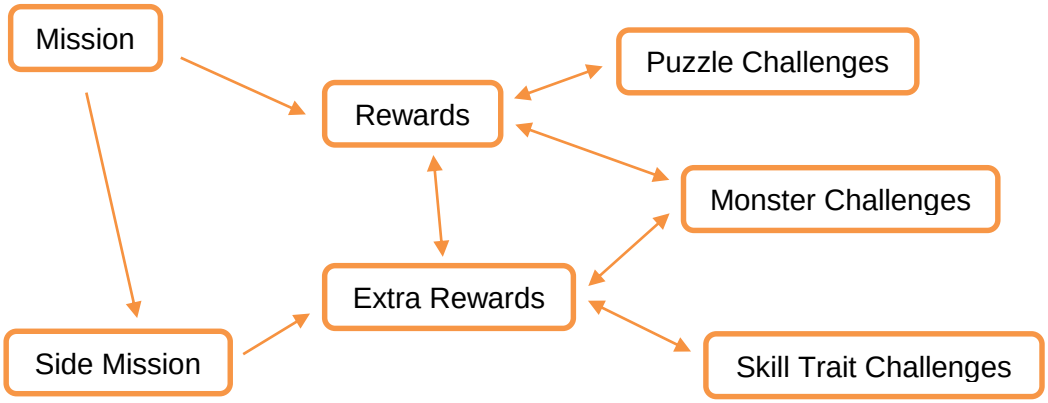


Chart 1. Chain link reaction during game play.

Figure 16. *Need for speed*, during game playing, players could access various shortcuts where they would face many obstacles or even zigzag roads [31].

Good level design constantly entertains players

No matter what modern technology is like, there is a limited amount of resources for a game to draw from, ranging from hardware limitations (like RAM= Random access memory) to graphical system (such as graphic memory). Clearly it is the level designer's responsibility to maximize the usage of those resources and create efficiency by good design.

Modular design is very important here. Veteran designers would not design a level/ map, instead they rather design a series of modular, mechanic-driven encounters, which could be linked and blended together to create a whole level/map and so on.

By following and applying simple modifiers to those modules, one could create variation, provide more levels with less efforts and less risks. This technique not only create series of familiar encounters but also keeps them fresh by applying constantly increasing challenges and surprises. Thus, the player can use to learn and master game mechanics via game play [2].

After developing two of the enormous open world maps and games of the year – *Skyrim* and *Fallout*, the Bethesda studio had proven how a relatively small team was able to create a lot of contents. Although a high level of modularity might not suit for every game, it could certainly be applied to any game in varying degrees. For example, *Call of Duty* brings to players "adrenaline rush battle moment" – sections of intensity combat gameplay, even approximately thirty seconds to five minutes, which players could quickly prototype and iterate on, then stitching them together in various different contexts in order to make a chain of exciting levels. These mechanics could help the level designers to build significant amounts of more contents in tremendous less time, yet still keep it interesting [29].

To make the levels and maps look flawless is time consuming. The point of re-using areas of the levels/maps is not only to save investment money, but also to alleviate the amount of level geometries which would be stored in memory. These tricks could sometime be referred as back tracking. As a level designer, one shall be careful to secure such spaces, were designed for bi-directional gameplay, as a key modifier on the second pass [30].

Good level design surprises players when they do not expect it

This approach mechanic has varied in many forms such as curve of high versus low intensity, exploration versus combats, rest versus action, etc. Those contrary actions serve as a good base line for level design, and are important for maintaining player engagement. In fact, constant repetition could quickly turn *de rigueur*. However, even great pacing levels would face trouble of being memorable without sudden spike in intensity which might come from surprise.
Surprise in particular does not need to be a big shock or a plot twisting but at its core, surprise could be considered as a rapid, constantly surge in uncertainly game play/story lines, which was the extreme essence of fun and joy.

In term of level design, surprise could disguise in form of a unique setting, a moment that teaches players something new and unpredictable. A mechanic, that has already been used for a while to create a beautiful vista or a radical change in pacing, can cause an adrenaline rush, satisfaction and be enjoyable (Figure 17) [5].

Figure 17. Resident Evil Series game, one of the horror survival games which constantly blended the surprise elements [19].

As fans of *Resident Evil* game series have experienced, once players enter from one location to another, there is a door opening scene. Cleverly, the game level designers have placed surprising concepts after players have entered a new location. For example, from the beginning, players can get acquainted with a so-called peaceful feeling when they see the door opening cut scene because, in fact it is a game mechanic that obviously keeps players not encountering any monsters, after entering a new location. The moment players are lowering their intense focus to enjoy a drop of peace as they should expect, monsters (sometime even a mini boss) would jump out from window/ drop down from ceiling or fully charge ahead to player's position. That surprise element can cause a panic reaction and drag player's intentional focus back in no time. Never let players predict what can happen next, is one of essential core keys of success [32].

Level designers must not be afraid of taking risks with their design. In fact, they should not just replicate a level from another game. There are millions of games in the same genre chart, so by trying something unusual and truly innovative could make an outstanding unique sparkle from the crowd (Figure 18). Visioning and managing the risks

– designing on paper – picturing the final product – creating a playable prototype should be done as early as possible.

Figure 18. *Urban Chaos,* the game which created by London's Rocksteady Studio by implemented vast of innovated elements. [33].

Creative ideas show to player another perspective of being a good police and a bad police. The studio developers broke the line of hero always being on the good side. There were consequences which effected by player's decisions.

After completing the game, and checking the credits, the game suddenly bring the player back to an extra mission – Some of the gangs, who were busted by player in game, had tracked down player's home and seek for vengeance. Therefore, the player must scramble through home, grab a sidearm from under the table, and put the end to the criminal mobs once and for all.
This post credit surprise mechanic was skillfully executed by level designers, providing a truly emotional end game for players, who were patient enough to watch credit from developers. And for those, who were skipping the credit part – it was obviously a mis-take.

3.2. Important aspects in Level Design

Besides those success keys mentioned above, there were also important aspects in level design, which bound up with graphical parts became a product of it own.

Music

Music is one of the most underrated parts of a game. A touching soundtrack could doubtlessly effect a gameplay. Indeed, music guides the players in the right frame of mind for incoming challenges. Thus, an epic boss battle could become even more special and unforgettable when accompanied by appropriate music. Practically, music is not the task for level designers as they are not responsible for composing a soundtrack, but if one take it away from a game, and it would feel naked.
Final Fantasy series in general and Final Fantasy VII in particular are members of hall of fames, due to their outstanding soundtrack as well as story line.

Movement constraints

Movement constraints may sound like a basic thing but this basic mechanic could be a headache for level designers if they do not carefully keep eyes on it. Most of game glitches nowadays is caused by movement constraints. Certain areas/ structures on the map were not set up at their right scale/ position as well as the movement limitation called Boundary. Therefore, if some lucky players find out and used it as advantages over others, they ruin the game's competitive experience. These kind of glitches occurs mostly in first person shooter online games – imagine a machine gunner who jumps out the map, drops under the ground and starts fire up to other player's foot. Indeed, it could be even more disasterous if that gunner had shown to his entire teammates the glitches.

In order to create a more interesting movement space, level designers must create more complicated level geometries. For example, staircases could behave like hallways – they have a start and an end, which provides a sense of path direction in the map – and they could be part of a room as well [2].

One of the best examples for movement constraints in level designing was The Legend of Zelda: Breath of the Wild (Figure 19). The game had been flawlessly designed from the story line to map geometries and game play, it was simply a combination between the best bits of the franchise's long history with the best bits of the rest of the gaming

world. The game was well designed in every single corner of the map, it gave to the player multiple choices so they would not feel like following the game default concepts. Indeed, it let player dived in and explored the game as their real life – each player had unique approaches – therefore, unique items accordingly rewarded.
In another smart design concept, there were four core boss monster placed at four corners of the map. Hence, in order to complete the story line, the player should explore almost all the map. That was a creative approach from the designers since they could place puzzles, side quests, unique NPCs on the way to the bosses.
Moreover, the weapon and item system in the game were also new. There was absolutely no durability in items – meaning, they would be shattered after certain time of usage. Those concepts made the player carefully to plan their strategy and tactics while encountering monsters through game play [35].

Figure 19. The Legend of Zelda: Breath of the Wild, one of four boss monsters combat scene. The game was created by Nintendo and it became one of top rated game by player choices [34].

In multiplayer first person shooter genre games, the sense of directions could be more subtle, and could offer more gameplay variety. One direction path or even dead end are dangerous because they obviously signal narrow and limited player options to escape. If there were an item or objective on the path, it could encourage the player to take this risk, these designs act like traps. A subtler form of this directional design created variation between offensive and defensive areas. However, if there are no reasons behind the direction design, or it had no role in the map, then it is a bad design.

For example, from the high ground (Figure 20), a player could decide, either jumping down or taking the stairs, to claim the treasure chest. However, a player who approaches the chest from the low ground must turn around because of the high ground is too tall to jump onto from the low ground. Therefore, stairs would be the only choice to reach the high ground. This geometry leaves one side vulnerable, and also provides a subtle sense of direction between defensive and vulnerable positions.

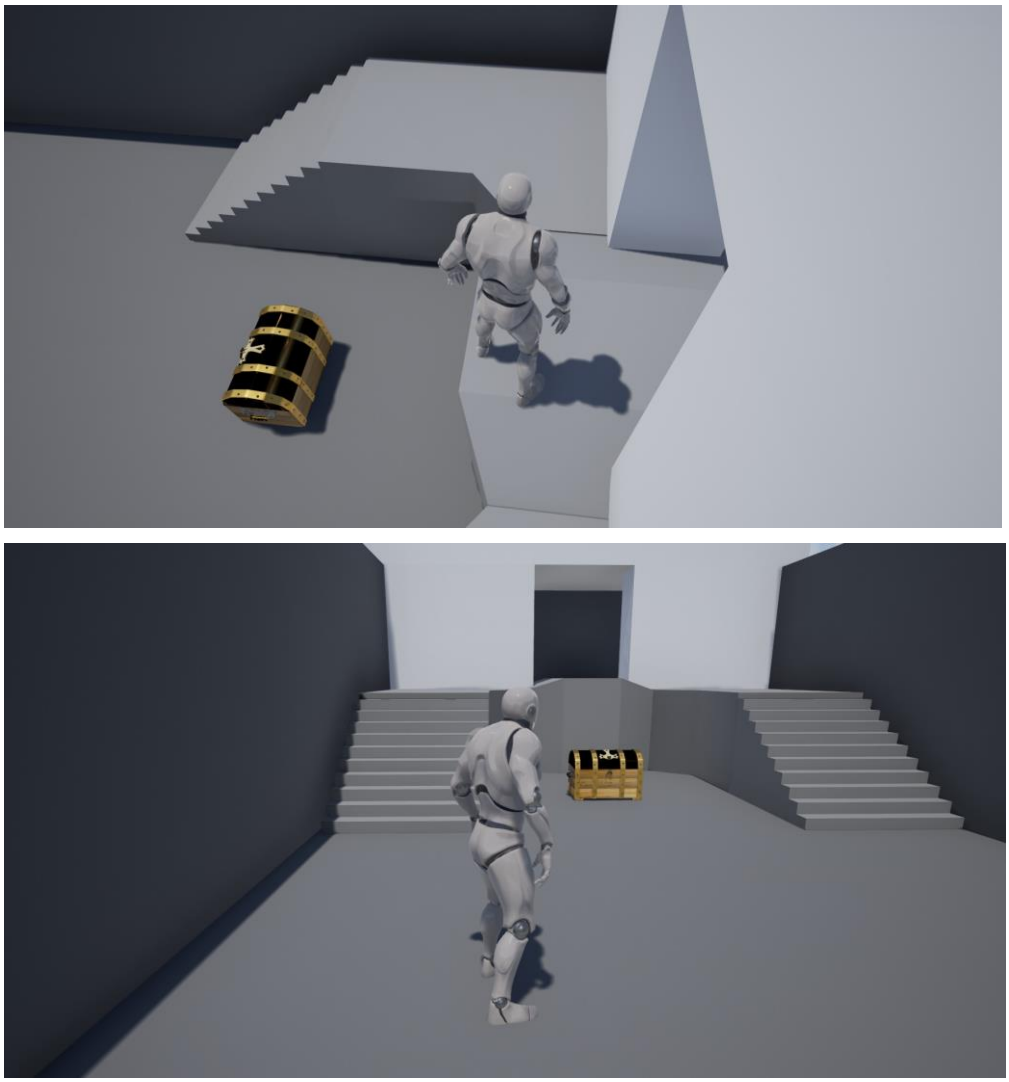


Figure 20. No complex directional level design.

Occlusion (Visible or hidden objectives for player, close sightlines and movement through the space open).

Besides constraining the path movement possibilities, map geometries also occluded sightlines. For example, a puzzle maze would be a heavily occluded space area, where

players could observe only their immediate location. Thus, their decisions would be informed about where to go and what to do. An enormous, open room could be almost as bad design because players could observe many objectives at once – if there is too much to see, then result is overwhelming, or if there is nothing to see, then result is boring.

The occlusion of space indeed strongly required the player to move and look around in order to observe everything. Thus, an interesting level should be expected to balance between these extremes which suited with the game's style, and evoked the feeling of player's demand.

Another important mechanics in occluding spaces is through doorways and other threshold usage. These frames, as the player crossed the threshold – slowly introduce it. This framing mechanic could also focus on the most important part of the next area, which provides the player a sense of priority to plan around once inside the room – the plan is a kind of gestalt, where the player observes and understand their local situation within better, bigger layout.
In addition, a space which is easy to adopt and interesting to explore, would be included in various global organizations or patterns, as well as local variation or noise. There should be a variety of closed and opened, difficult and easy spaces. These sort of variations shall create the pace of level.

A multiplayer first person shooter game is taken as an example, if all of the weapons are projectile based, then players are secured when distanced from their enemies, they can even see each other. If the weapons are auto-detect, then anything inside the crosshair would take damage when the player is shot. Therefore, it is more crucial for players to cover between themselves and their attackers via obstacles, and avoid open spaces.

Lighting resources

In fact, applying the right choice of light could improve the quality of a game and make it appear more realistic to players. In some games, light can be advantage or disadvantage during gameplay. A scene, where light is carefully and logically placed can affect player's emotion as well as playing style (Figure 21) [2]. Moreover, a good logically

placed lighting resource could bring liveliness into scene and provide the feeling of magnanimous or cozy to the players (shadows, illumination, light-shaft, emission).

For example, in first person shooting games, if a player had stood facing the sunlight source, the vision would definitely be blurred, narrowed and unable to observe movement from enemies – On the other hand, enemies could clearly observe if their back was facing the light source.

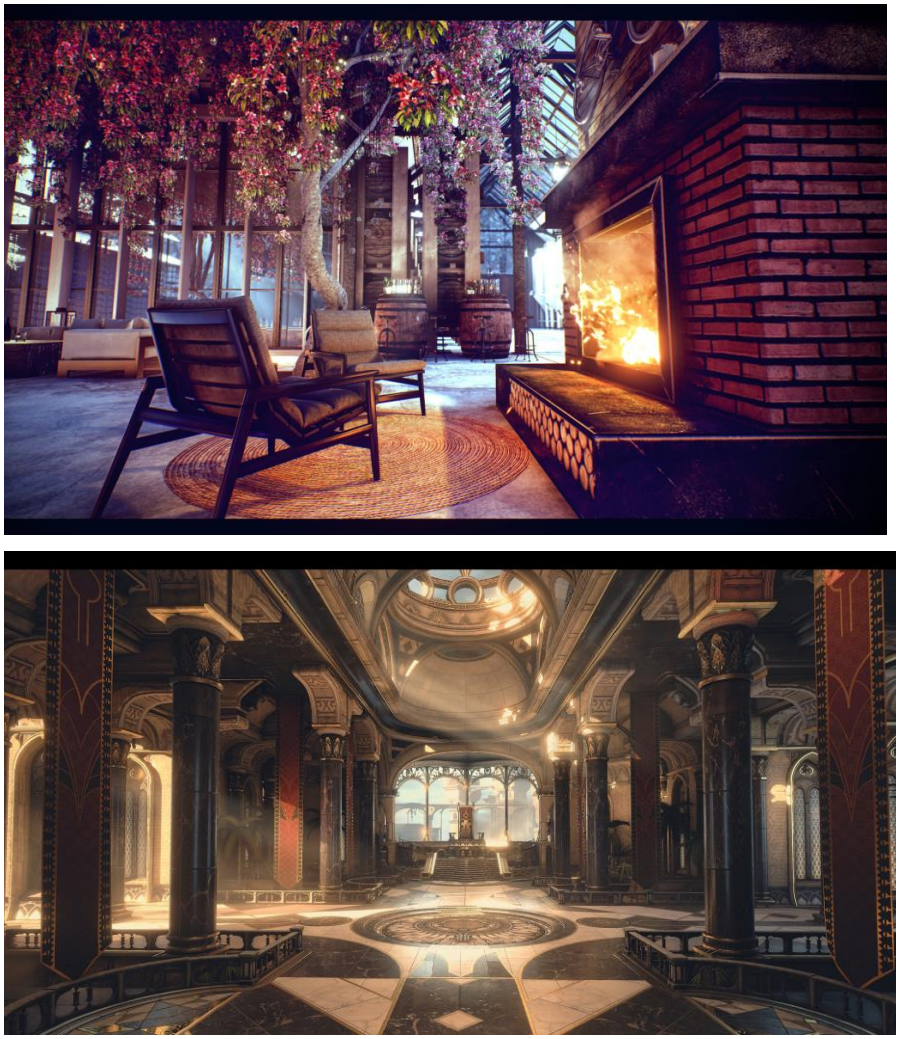


Figure 21. The magically works of lighting from Rostislav Nikolayev [21.]

3.3. First Person Shooters/ 3rd person/ Top-down level designing

Besides other game's categories, first person shooters and Third person and Top down, were often choices in developing [7.] There were reasons for developers to pick top three due of massive benefits, indeed at first person shooters and third person game genre, it

would be obvious opportunities to show off to players the outstanding and flawless well designed environment through game world.

For level designers in particular, there are not much differences between creating an environment level/ map for first person shooters and third person game genre. The only difference is to be able to tell the genre. On other words, there can be same map, same objectives, same items, same obstacles and even the same main model of characters, but there would be two cameras, detached and attached to the main character. One is placed in the main character's head – rendering according to human eye angle (Figure 22); one is placed behind main character, not so distantly – capturing character's movement and actions.

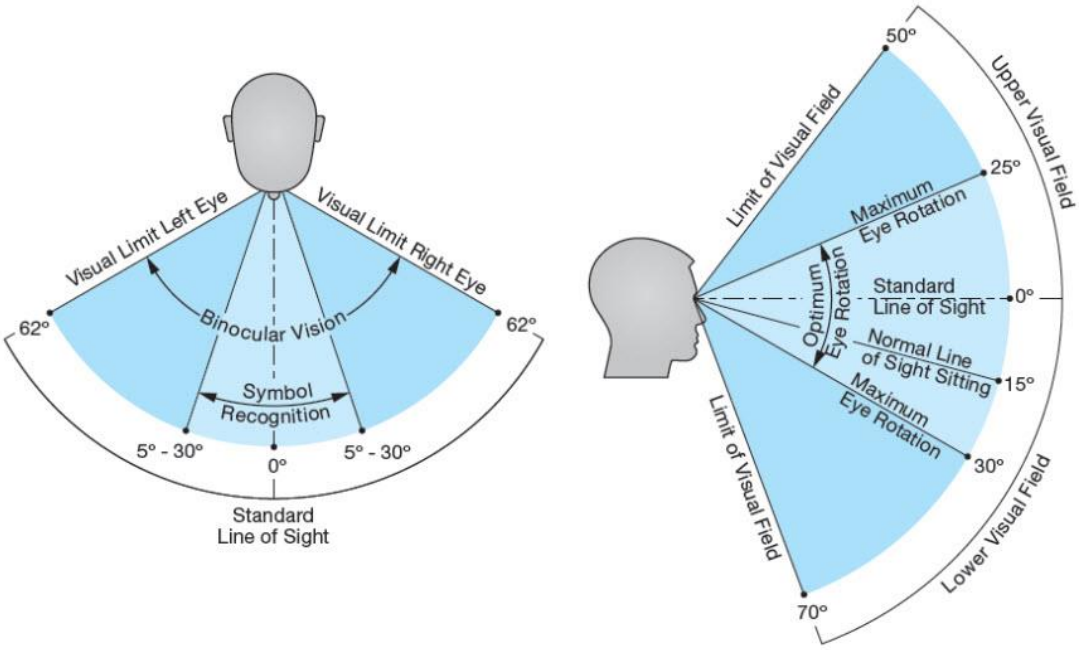


Figure 22. Human eyes angle of view. The camera's angle of view would use exactly same perspective view as human eyes [22].

In addition, there are various games such as Fallout 4, Company of Heroes 2, Skyrim… which provide the choice of switching from first person view to third person view and reverse. These mechanics would bring another new experience as well as challenges for both developers and players. If players chose the first person view, they could gain advantages in aiming, tracking movement in focused area since first person view allowed players observed in narrower view (Figure 23). However, they would pay

the price of tracking other players who would come from left or right sides and it could make them easier targets to be flanked, compared to third person view [36].

Figure 23. Third person view perspective versus First person view in *Star Wars –Battlefront* [37].

Those pictures explain viewers the advantages and disadvantages of perspective views in third person camera and first person camera. Within the first person camera, players could gain better vision in a specific area but pay the price of observing wider view from left and right side

The Top down perspective games could be a heavyweight competitor, compared to First and Third person games due to the blooming and renovation of graphical technology. Indeed, they could be parallel to each other – the more advances in graphic, the better top down games would be developed. Practically, the top down perspective games are not different from two other neighbors. They share the same game mechanics, same blueprints, even same models except cameras. A camera is placed on top and pointed down to track player's movements and actions as well as capture the gameplay. Also, the top down has a unique mechanic, which allows players to be observed through kinds of houses while entering inside. Due to its enormous capturing screen, the top down perspective would be mostly focused on developing Strategy genre games, Online Role Playing Game games, and new comer, trendy genre Multiplayer Online Battle Arena games.

4. Level Design Workflow in trailer creation

During internship, there were opportunities to participate in various projects such as; small games, interior designs, trailer creation, etc. With real life experiences, there were different techniques and tricks which could be helpful for a level designer. This project had typical goals for a level designer. During the workflow of creating a trailer by using Unreal Engine 4, various models and objectives were created from other projects, and they were re-used in order to avoid time consumption. Furthermore, there were also Autodesk 3D max and Adobe Premiere CSS5.5 which would be used for supporting the workflow.

After receiving an order from a customer, the product would be delivered to customer as a trailer video as soon as possible. The customer demanded to have a scene, where viewers could feel a peaceful nature environment and the trailer should be lively.

In order to satisfy the customer demands, the workflow was divided into separated parts which helps author to control the workflow. Those parts were not only important to the author but also for any level designers since it helped tracking all the process.

4.1. Storyline Scripting

Mission

- Creating a peaceful natural environment.

Demands

- No game play, only background/ trailer style.
- Animations in order to provide liveliness to the scene.
- Location is unbound but prefer places with lakes.
- Trailer could be maximum at 1 minutes and minimum at 30 seconds.

Story scripting

Based on the requirements, an environment which took place in a forest would be suitable and satisfy the customer.
Assuming viewers take a walk in a forest in the morning, enjoy their freedom and merge themselves into nature. There would also be a small waterfall and a slowly flowing river.

4.2. Map Creating

In order to create a scene mentioned above, a landscape in Unreal Engine 4 was created. There were two possibilities to create a map.
In the first method, the map could be created in 3D software such as Autodesk 3Dmax, Blender, Maya, then the map would be imported to Unreal Engine to continue decorating.
In the second method, level designers could create a map straight from Unreal Engine by using the engine landscape tool as well as other supporting tools. Due to the purposes of this thesis, the author chose the second method to demonstrate the workflow.

Unreal Engine 4.13 was selected to create this project. In the beginning, users could select the blueprints of the project such Third person, First person, Vehicle, Virtual Reality, Top down, etc. All the blueprints could be modified within the workflow. However, the best suitable blueprint was Blank, because the project only needed one camera to render at the end. Therefore, add-on blueprints would be unnecessary.
After opening the blueprint, users could delete unnecessary default models and select the Landscape option tab (Figure 24). This Landscape option provides to the user the ability to create the open world map.

Inside Landscape option, there were three different types of hierarchy children options - Manage, Sculpt and Paint option – those options had their own specification functions to support users in map creation. However, at this stage, the Sculpt and Paint option were disabled since the user needed to finish the first Manage layout in order to process further.

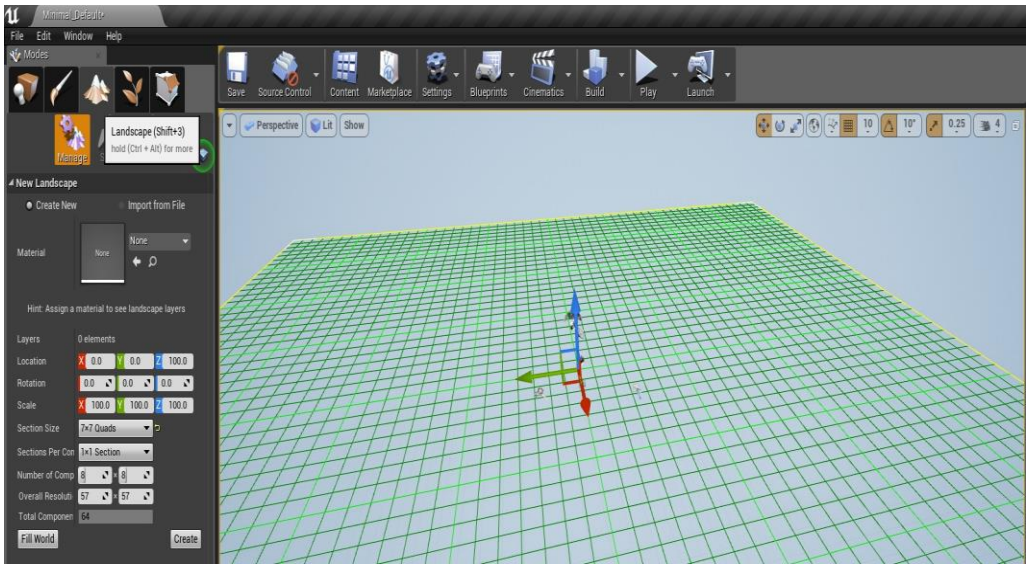


Figure 24. First step of creating a map, using Manage option under Landscape.

The map Section size was selected as 7x7 Quads size, since the scene would not need to be very big. The size could be adjusted from Section Size. Users should avoid the Fill world option on the left side at the end, because it would consume a vast amount of hardware usage to fill the world map, which was totally not fit for the purpose of the project (Figure 24).

The next step was selecting the suitable ground material for the scene. Due to the demand from customer, the scene would be a jungle theme so users could choose grasses and soils for the ground. There were various materials, some were provided by Unreal Engine from basic contents (red highlighted area – Figure 25). On the other hand, users could also create their own materials and import them into the engine.

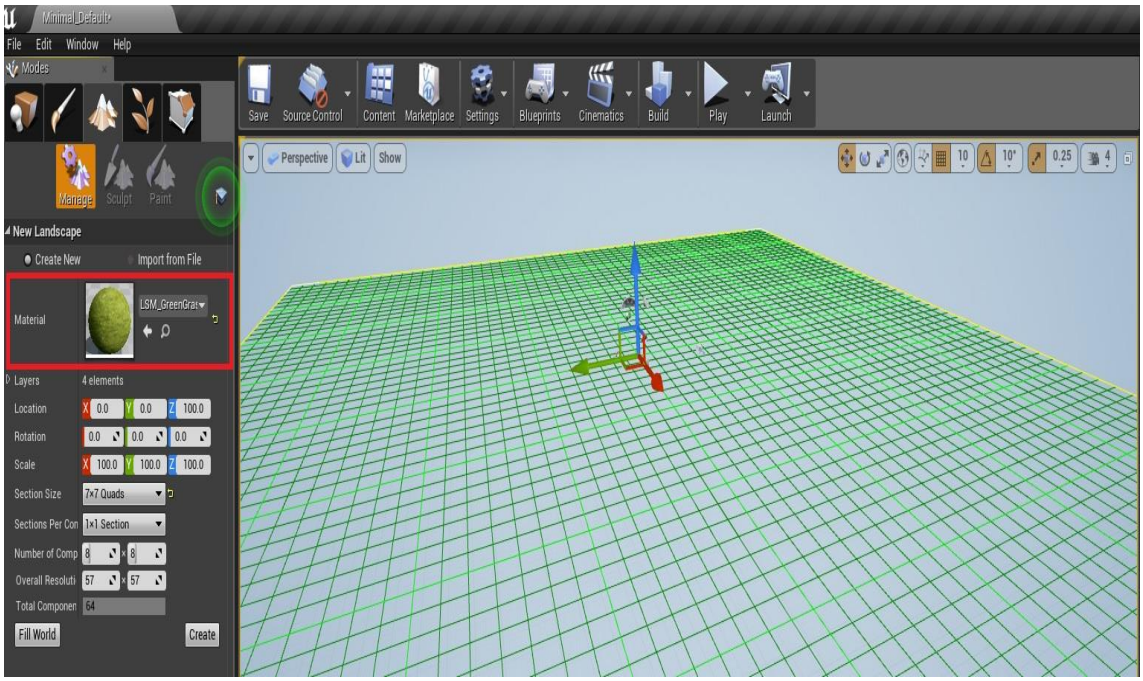


Figure 25. The ground material (grasses and soils) was selected and applied to the project.

After creating the basic layout for the map, under the Paint option tab users were allowed to apply materials by using painting brush mode. There were four defaults options which would be matched with the material name (red highlighted area - Figure 26). Those layout material options must be chosen in order to continue. As from top to bottom, there would be:

- First option, grass material – it was only basic grasses material.
- Second option, grass and soil material – it was mixed together.
- Third option, only soil material.
- Fourth option, water material.

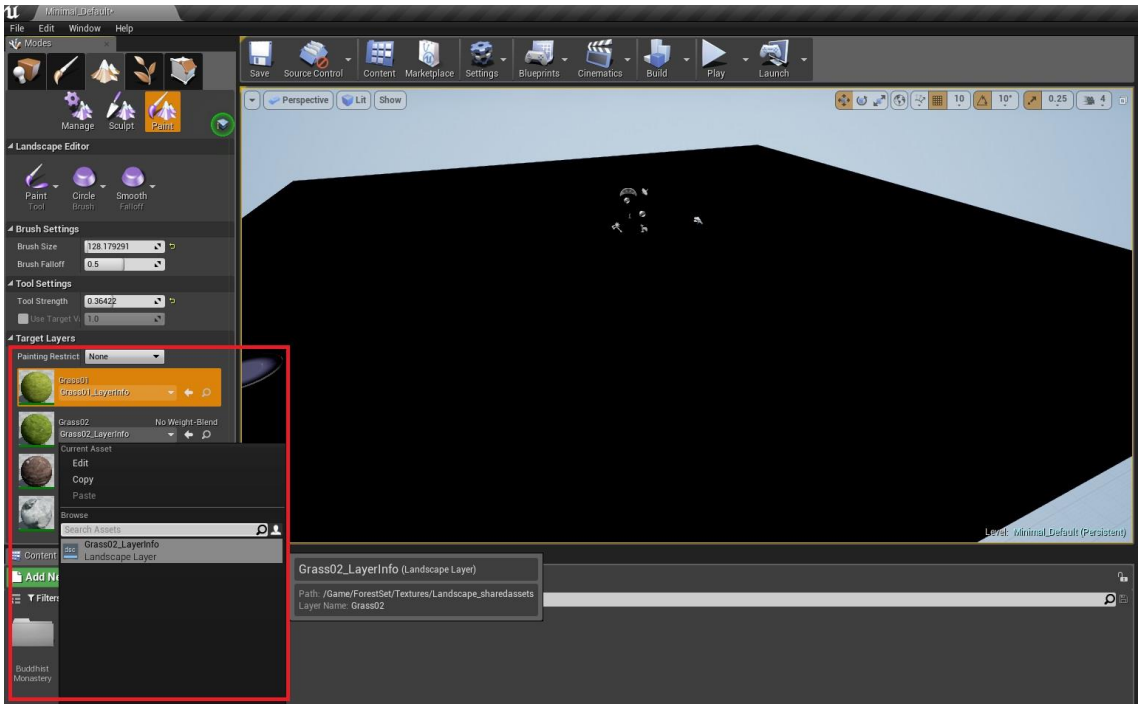


Figure 26. Paint option material layout.

At first sight after creating the map as well as setting up the materials, users could see the layout was covered in black color. Hence, users should use the paint brush to make the materials appear into the map. This process could take a short or a long time depending on the size of the brush.

The last option in Landscape is the Sculpt tool. A decent quality layout map could not appear in flat surface. Therefore, after covering the map by materials, users needed to modify the height of the map using a sculpting tool. Basically, the sculpt tool functioned like a paint tool but it defied how to raise on or fall off the area. According to the environment, which would be an inside jungle, fall off areas were created in order to adjust as a river afterwards. By using enough amount of brush falloff strength, the river area was modified while holding Shift and pressing left mouse button (Figure 27).

The appearances could be frustrating in the beginning but as a level designing job, there must be step by step approach to fix problems which usually occurred while using sculpt tool.

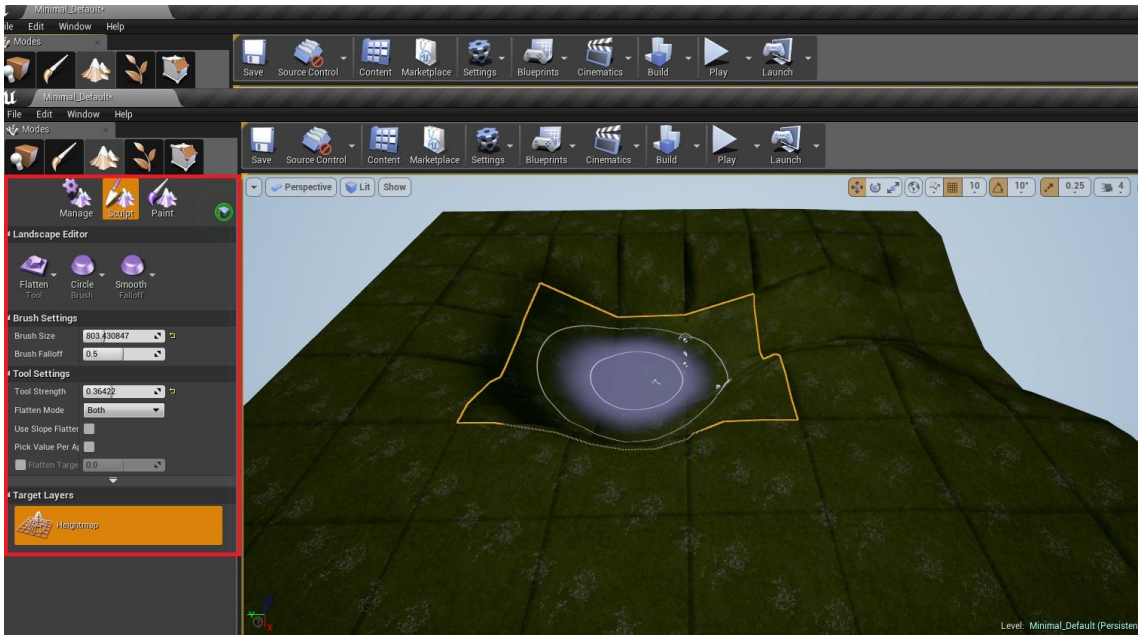


Figure 27. Sculpting the layout map, workflow of making river area and height map.

At this stage, there would be a basic level designed map, where all the materials were applied and the heights were adjusted. Foliage option was the next phase for the map. There were models of grasses, small trees, small rocks, etc (Foliage models could be used in massive amounts and they were automatically grouped together into batches that are rendered, using hardware instancing, meaning that many instances could be rendered with only a single draw call). Users could add the foliage models that were intended to use by “+ Add Foliage Type +” (Figure 28). There was no limitation for adding foliage. However, every foliage counted as RAM usage, so a good designed map should be calculated by the amount of models in used.

In addition, depending on users, painting the foliage would be different. For example, users could increase the Paint density to maximum 1 value in order to paint grass, and decrease to nearly minimum 0,1 for painting trees. Therefore, the result would be more legit looking (red highlighted area).

After that, the grass models were placed on the map by using paint tool. Moreover, all types of foliage models could be selected at once and placed on the map. Users had to be careful since it could easily become overwhelming.

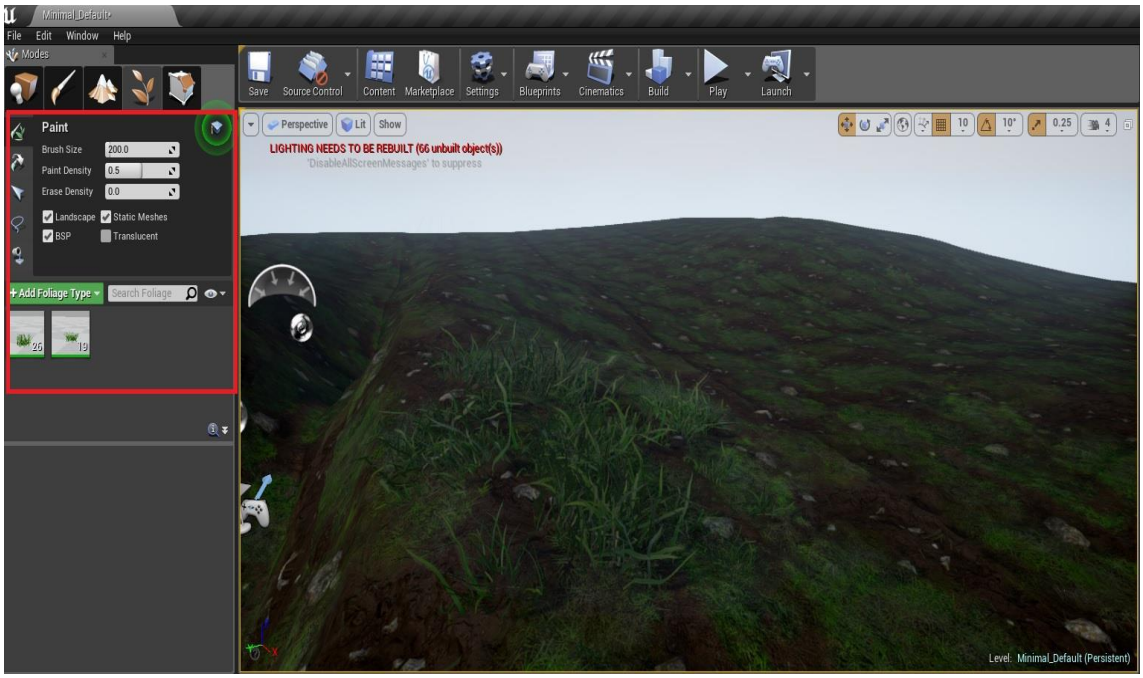


Figure 28. Placing and painting foliage grass models.

The next phase would be importing static models such as rocks and trees. This was the main job of level designer – placing and creating the scenes, based on models provided (Figure 29). The scale of rock model was modified by using scaling option (R – short key) or manually adjusting through XYZ axis (red highlighted circle) as well as the rock's position by rotating (E – short key) and moving (W – short key), even though those rocks would intersect with the ground but it would not be a problem. In additional, the model type should be Static selected (red highlighted box) because the rocks would not be able to move or be interacted, therefore Static option definitely improved the lighting effect later on.

The difference between Static, Stationary and Movable could be explained as:

- Static – used mostly to un-movable objects on which the light never changed. For example, grass, trees, rocks.
- Stationary – used mostly to objects which would not need to be moved around but still needed animation like changing color when receiving light.
- Movable – used mostly to character objects which constantly changed position or were moving around.

Sometimes, all of them would be twisted, depending on purpose. For example, rock models or tree models could be selected as stationary objects when the designer wanted to close up rendering light effect on that specific object. After all, the level designer will do what is best to achieve the goals.

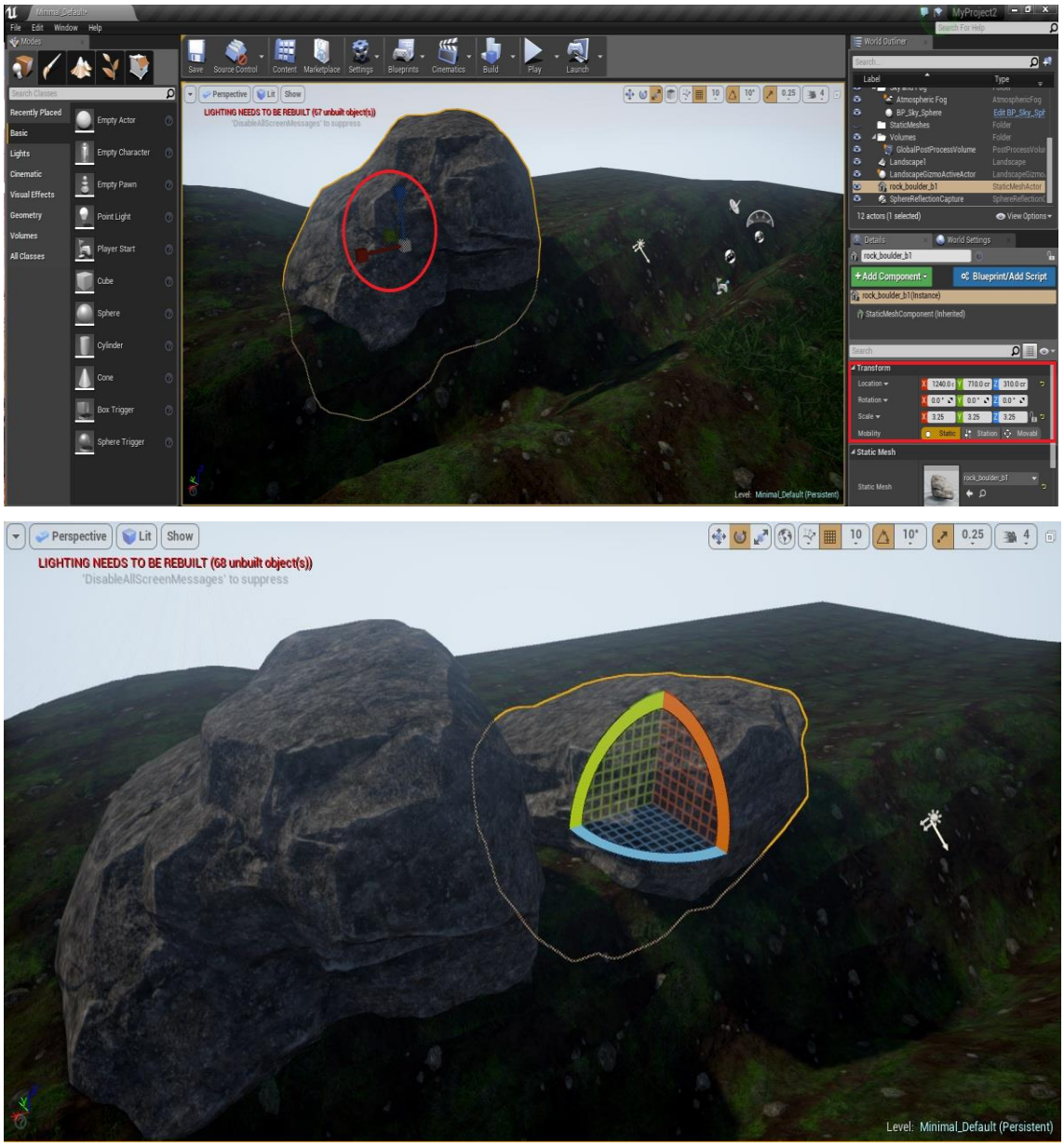


Figure 29. Importation and modification of rock models on the map in Unreal Engine.

In this step, a pipeline river was used by blueprint, it displayed as a simple plane with water material and flow animation (Figure 30). The plane could be extended by picking at the end point and extruding (Alt + drag to desired direction). Moreover, the position of each point could affect the flow animation.

For example, based on XY-axis coordinate, if the third point was lower than the second point at X-axis, the flow animation would change into flow down animation accordingly. Therefore, the lower the points, the more waterfall the flow animation would have.

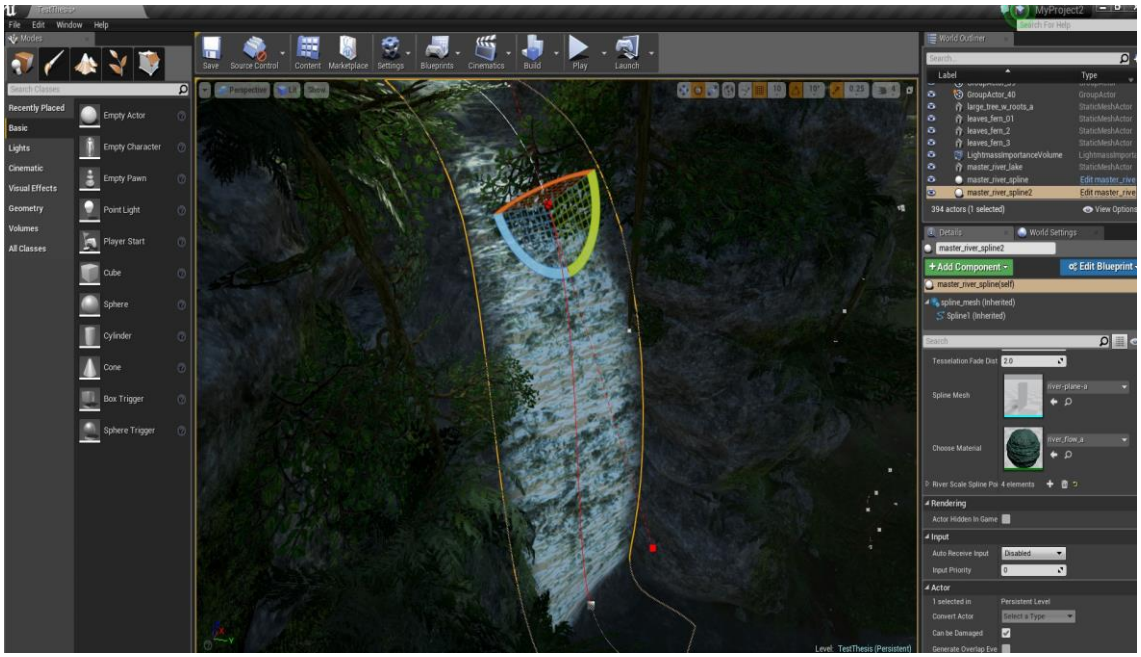


Figure 30. The creation of waterfall, based on adjusting points after points. The lower each points after points, the heavier waterfall effect occurred.

When the river flow was done, another blueprint was used to create pipeline trees. The blueprint functioned like the river pipeline but with different materials of course. In this blueprint, the starting point would be the root of a tree, then by extruding the connection points, users could create a full grown tree (Figure 31).

Those points could also be bent to achieve even more realistic branches. After separating tree's body and tree's branches, the final step was attaching the branches into the body.

Users could also adjust or modify how many branches they desired (red highlighted area). There was a ticked box "add branches" option (red highlighted area) but it only focused on decorating small branches.

To achieve a realistic looking tree, level designers should manually create other tree (smaller than the original one) by using the method above. They attached the tree to the original one and transform it into branches (small trees were attached into bigger trees to become big tree's branches). The result would be a huge tree with different size of branches pointing out at all directions.

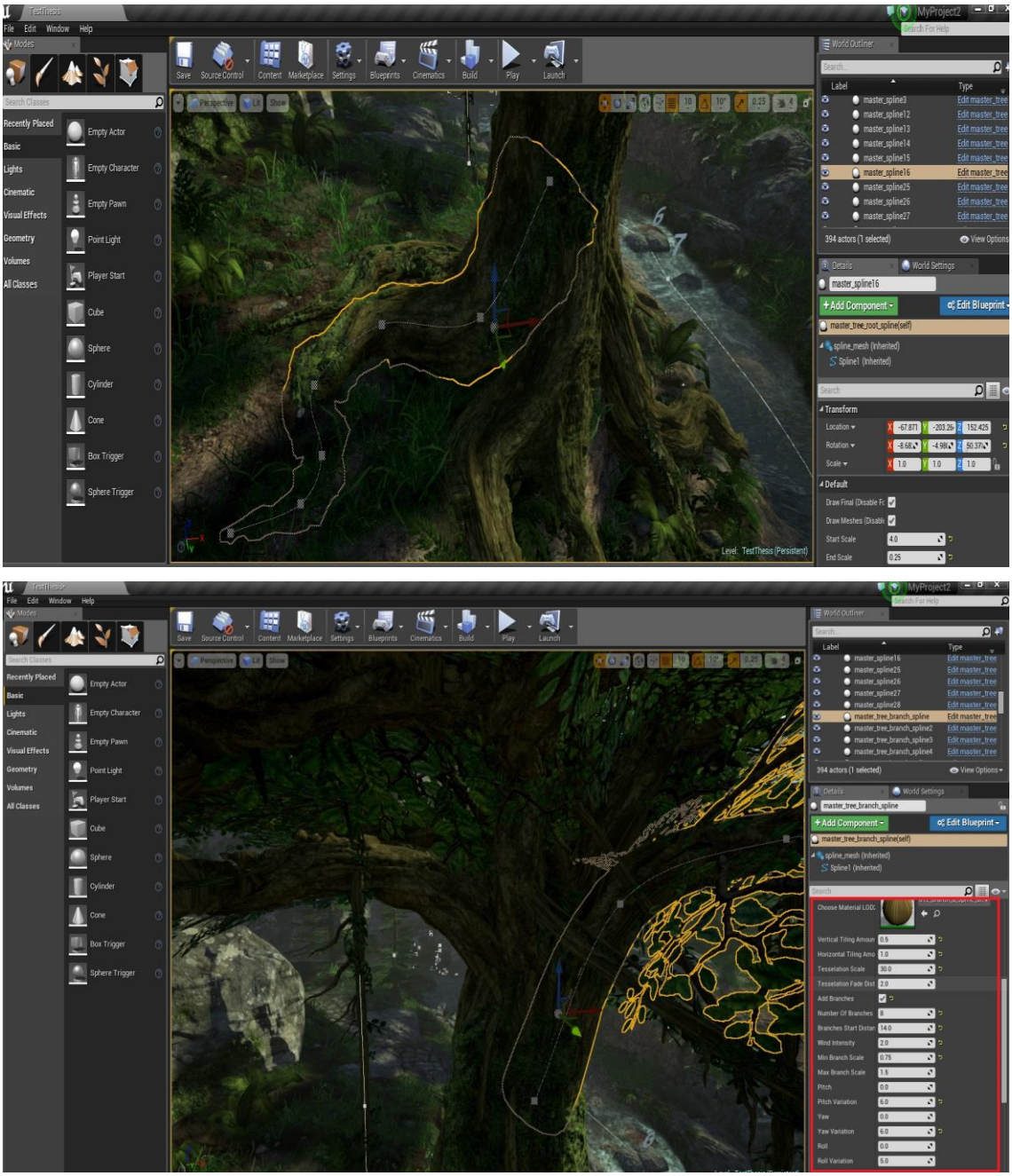


Figure 31. The process of creating a huge realistic looking tree, using blueprint pipeline and placement of other smaller trees.

4.3. Modelling and Importing

In order to create the props for this project, various amounts of ready-made models from the author's previous projects were used. Also, AutoDesk 3D max was used to create the rocks, trees and materials for the ground.

Models could be made and imported from another software into Unreal Engine 4. However, the easiest and most common way was just to drag-drop those supported file formats (.OBJ, .FBX) of desired models into the destination folders.

The modelling process began as the rock models were created in AutoDesk 3Dmax. At first, a cube was created and modified in editable mesh mode with turbo smooth effect (Figure32). In the final stage before exportation, the rock model was approximate five thousand and three hundred polygons.

The rock model was then imported into Unreal Engine by drag-drop method (Figure 33). In the red highlighted area, there were important options which users must pay attention to in order to avoid encountering the importation problems.
For example, for non-moving objects, users must not tick on the “skeletal mesh” since it was meant to be used for humanoid characters only (because those models have their skeleton animation within). Also users must always tick on “auto generate collision” and “import materials” as well as “import texture” unless they only wanted to use original non-materials or textures models.

Usually, if users intended to use models with seperated parts, the Combine meshes option should not be ticked because the engine would wrap up all parts into one big mesh. That process would destroy the animations which were placed within those separeted parts.

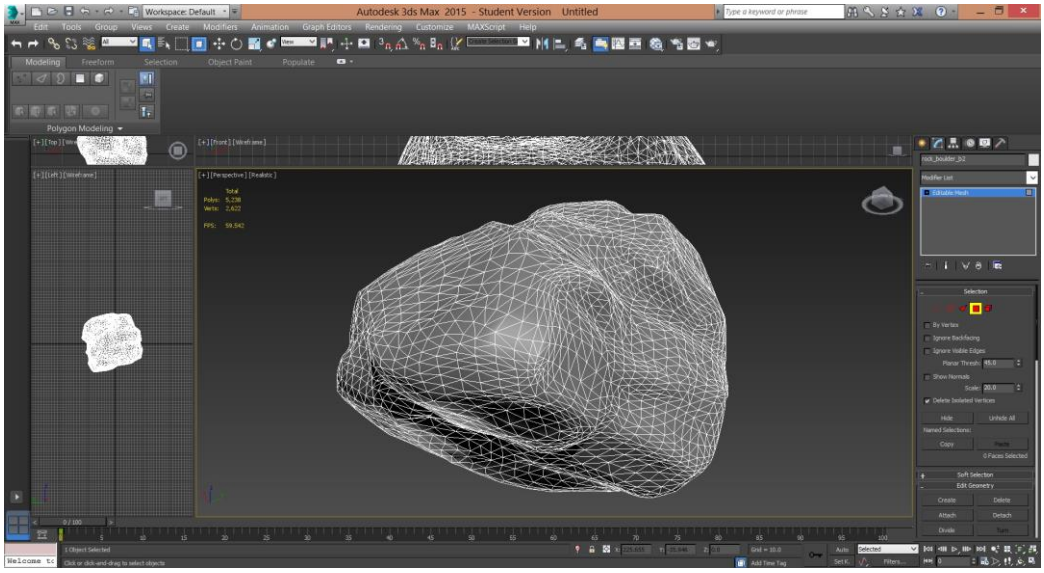


Figure 32. The rock model at final stage in AutoDesk 3D max.

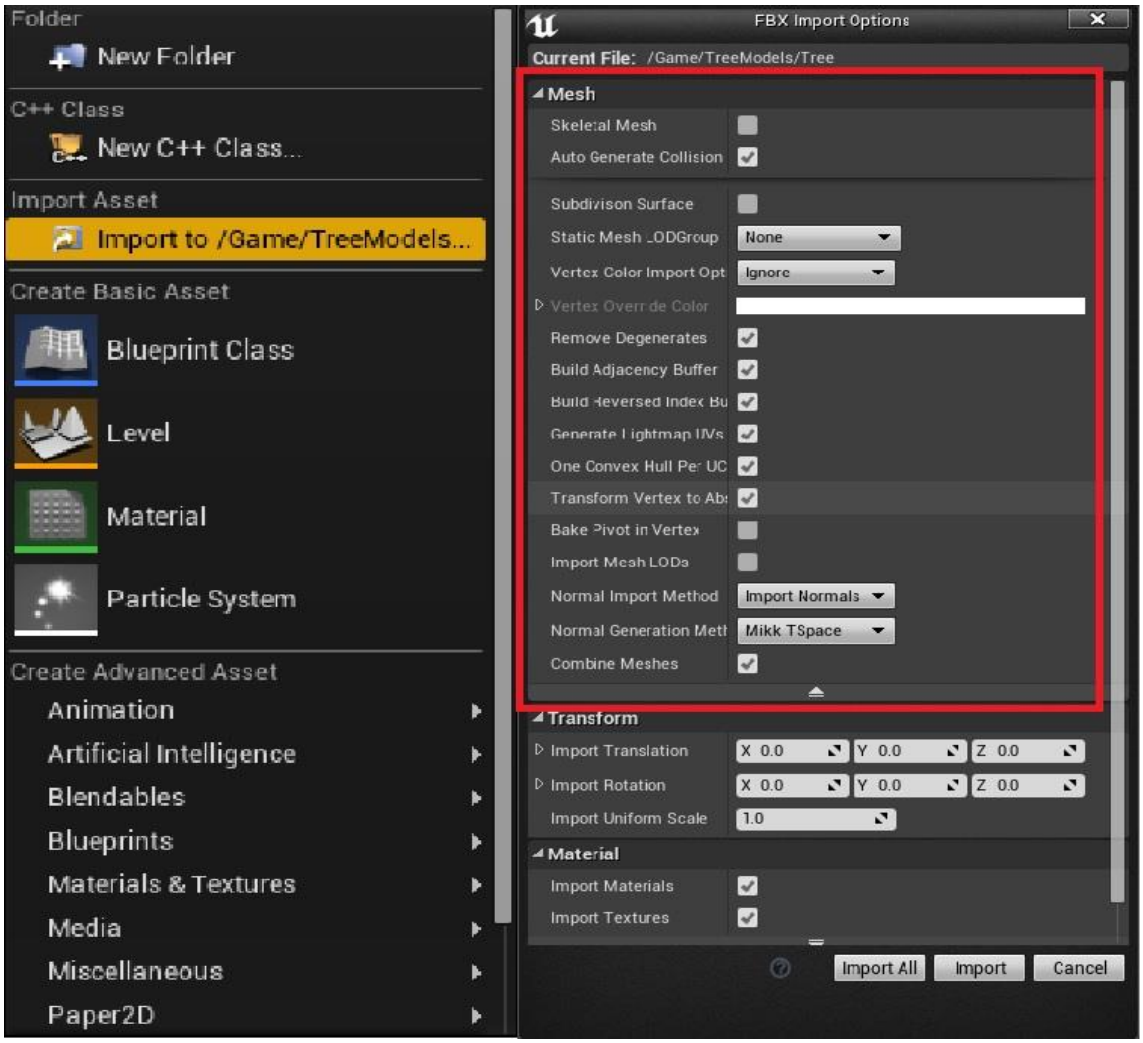


Figure 33. The important options when imported models into Unreal Engine.

After importing the rock models, the “auto generated UV map” option was selected to generate the rock’s UV map (Figure 34). Inside the red highlighted area, those options were set as default but the “destination lightmap index” value should be at 1 in order to apply the UV. On the left side of the picture, there was a UV map which was built after applying the generating process.

This method demonstrated the process of using Unreal Engine in order to generate an overlap UV map, which can sometimes occurr after importing model. When objects are made from another 3D software with UV map layout ready, this method is not neccesary.

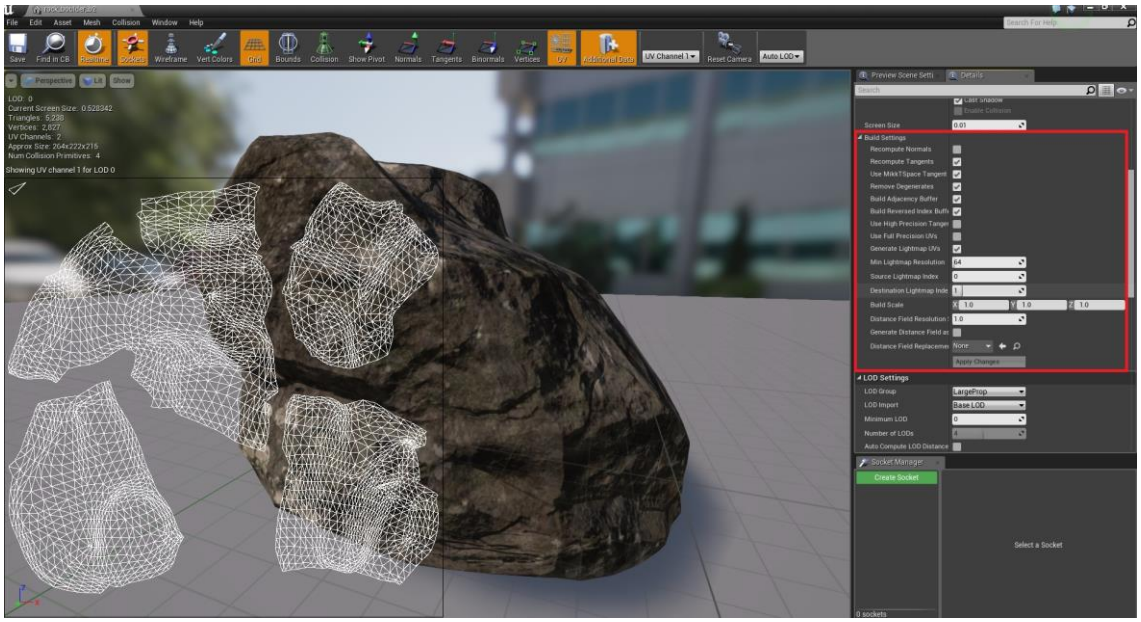


Figure 34. Generating UV map process after importing the rock model inside Unreal Engine.

4.4. Camera workflow

In Unreal Engine 4, camera workflow is one of an important features which can heavily effect the quality of the final product.

A good level designer must be able to controll the camera in order to maximize the benefit from it. In some cases, if the map has some problems and there is not enough time to fix it, then the camera can be very important to save the project.

Objects and models or even bad locations inside the map, can be hidden away by using camera workflow. Once the level designer understands what the camera can provide, the quality of project can improve.
This camera workflow process requires specific skill in cinematic observing. The process is described in details at below.

In Unreal Engine 4, users could choose “Add matinee” under “Cinematic” option tab (Figure 35). Then, after the engine had created the matinee actor, the “play on Level Load” under Play section must be ticked on, otherwise the camera would not be able to work. Next, by clicking the ”Open Matinee”, users could configure the workflow of the camera.

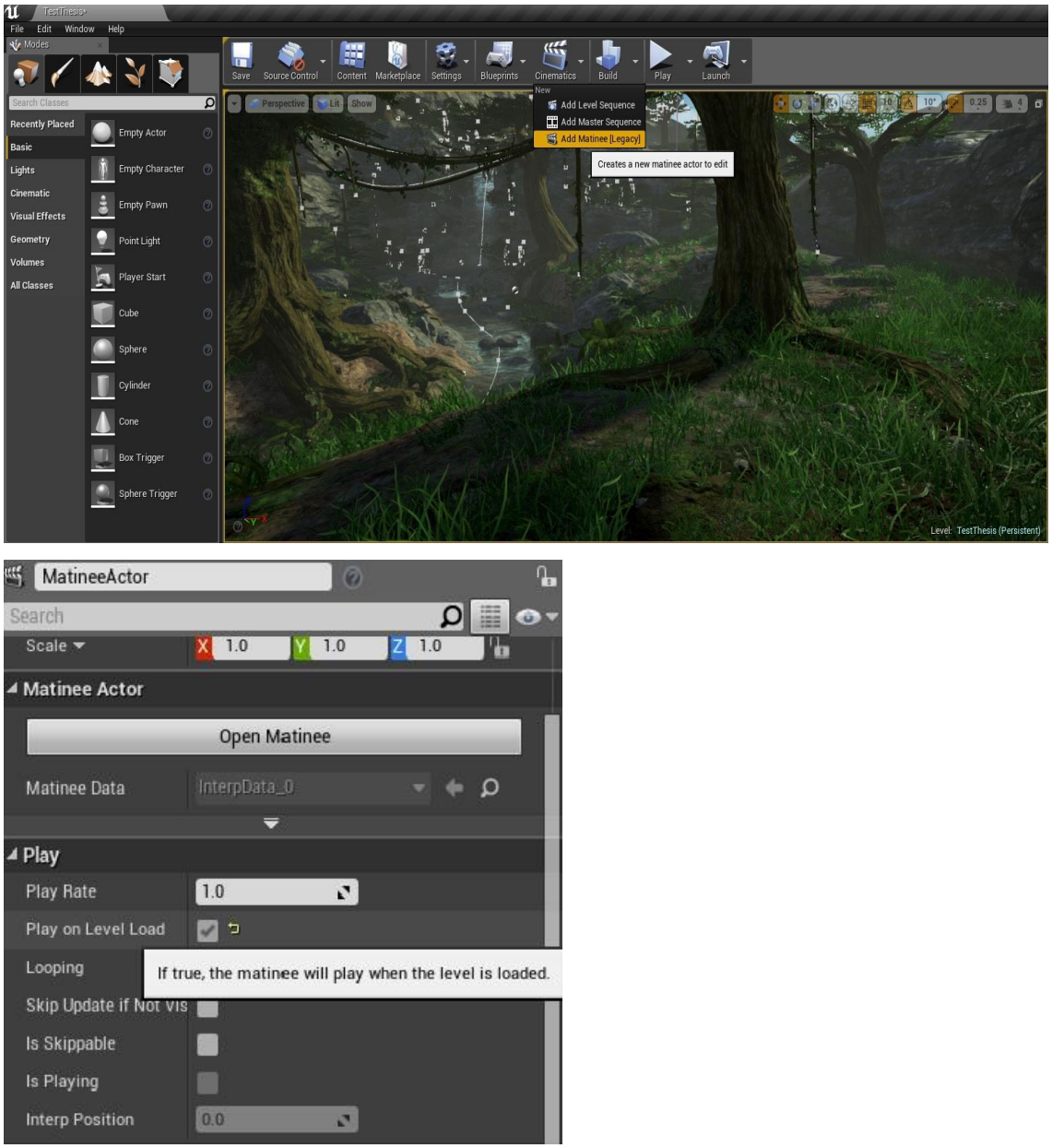


Figure 35. Setting up the first step for camera workflow process.

Once the Matinee actor was opened, there were two groups that should be created; Director Group and Camera Group (Figure 36). The Director Group acted as a director who controls all the functions or workflow executed by users and the Camera Group. Meanwhile, the Camera Group was the place where users could input and adjust all camera workflows.

There were also two types of dots (red circle area). From the left to right side, the red dots stood for the total length of the video (at 0:00 as the beginning of camera workflow to 45:00 second as the ending) and the green dots stood for actual length of the video (in this case, amount of total and actual time was same value).

Last but not least, there was also a black strip under a green dot on the left side, which stood for event camera position.

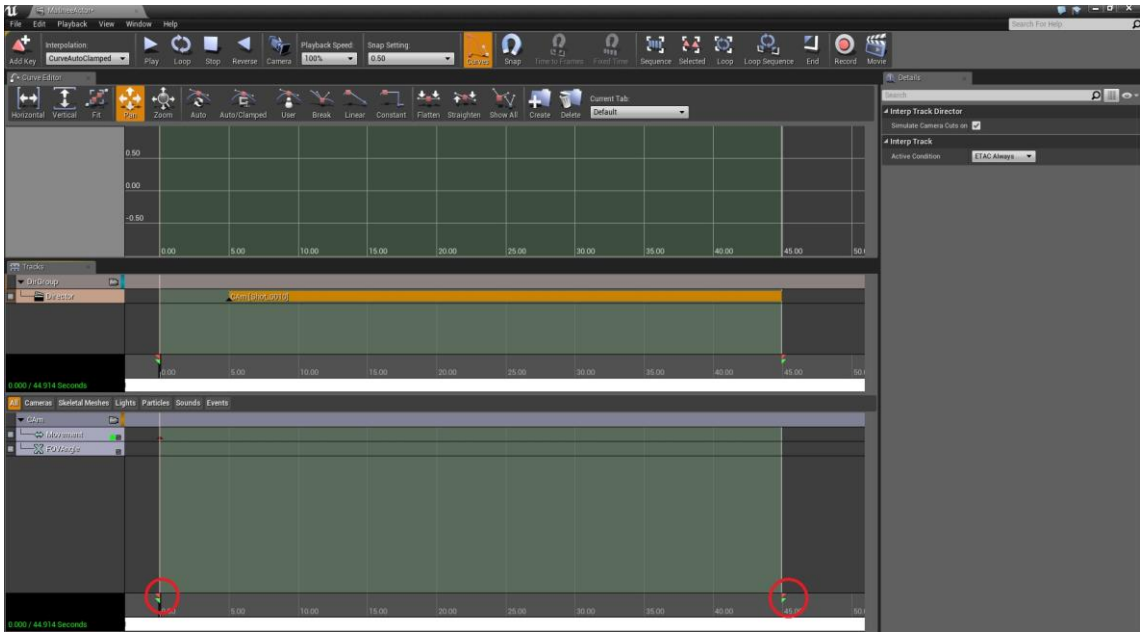


Figure 36. Setting up the duration of camera for the project rendering.

After adjusting the starting position of the camera at 0:00, users could press “Enter” to mark into the system. Indicators would appear as dark red arrow icons (inside and on top of green area; Figure 37).

There were six marks representing six locations of the camera.
During each movement of the camera, users should carefully place a reasonable amount of time in order to avoid the sudden changing angle of view causing an uncomfortable motion for viewers.
For example, from 0:00 to 10:00 second, the camera should have ten seconds to move from position at 0:00 to position at 10:00 and so on. This process required the actual experience from level designers to adjust since it totally depended on the purpose of the scene.

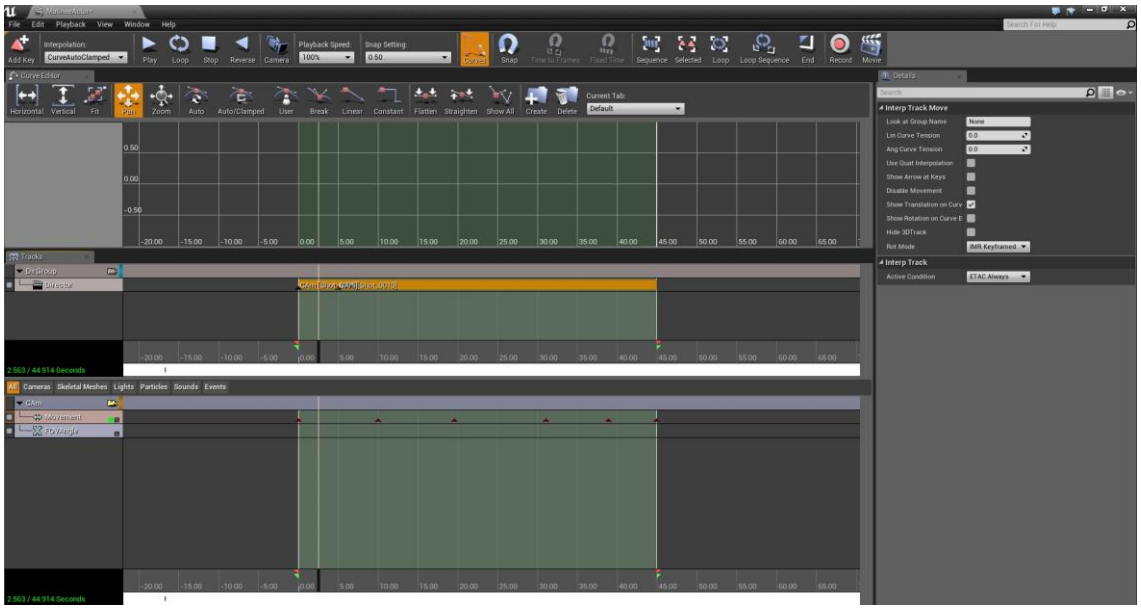


Figure 37. The process of placing and marking each camera position during the workflow.

4.5. Post Process

In this section, Adobe Premiere Pro CS5.5 was used in order to create the after effect for final product. The project was exported as rendered images (.PNG) and combined into video. The engine provided three options for outcome results such as Video format, Images format and sequence formats.

Rendering by image format was recommended as safer than a video format. If there would be problems which during the rendering time, users could always resume the process if they had chosen image rendering. Otherwise, they would lose all if video rendering selected.
When all the images were rendered, a new sequence was created in Adobe Premiere Pro CS5.5 (Figure 38). Then, a whole full images package was combined to become a video clip.
Furthermore, the “Level” option under “Effect” tab was selected in order to twitch the post process effect. Basically, users could select, drag and drop the “Level” icon into the video to activate the effect controls.

After activating the “Level” effect, “Effect Controls” tab was used for modifying the effect. By adjusting the “(RGB) Black” values from 0 to 20, the scene was put in a blurring foggy effect which was suitable for the purpose. In level designing, there were definitely no limitations on creating speciall effects for projects. With imagination, level

designers can twist, bend the values of White Balance, RGB coloring, or even reverse the camera workflow to achieve various special effects.

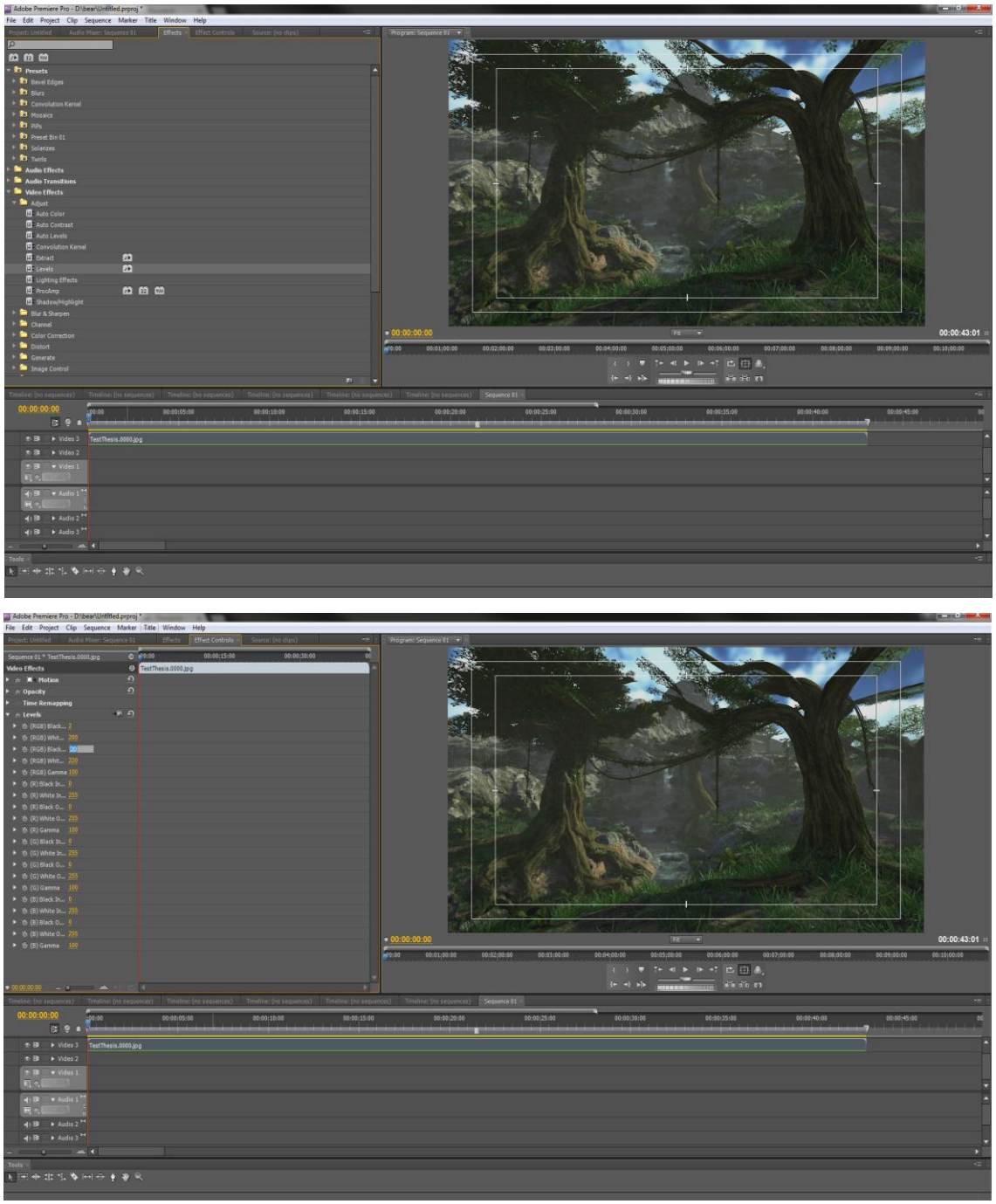


Figure 38. Adding effect and creating fog environment inside Adobe Premiere CSS5.5.

After polishing all effects, a music soundtrack was imported into the sequence (green area below the video, Figure 39) and adjusted to fit with the video length. As mentioned in the theoretical part, music brings users mental pleasure when observing the product.

Therefore, based on the genre and the scene, the sound track of Morning Mood by Edvard Grieg was purchased to be used in this project.

Usually, the music part has a longer duration than the video clip. However, users should not shorten the duration by compressing it. This causes the fast forward speed effect while playing. Instead, users could trim or cut the un neccesary parts out, so the soundtrack could play at its normal speed.

When all images were rendered in full high defination resolution (1920p x 1080p), the final product matched to the same export resolution.

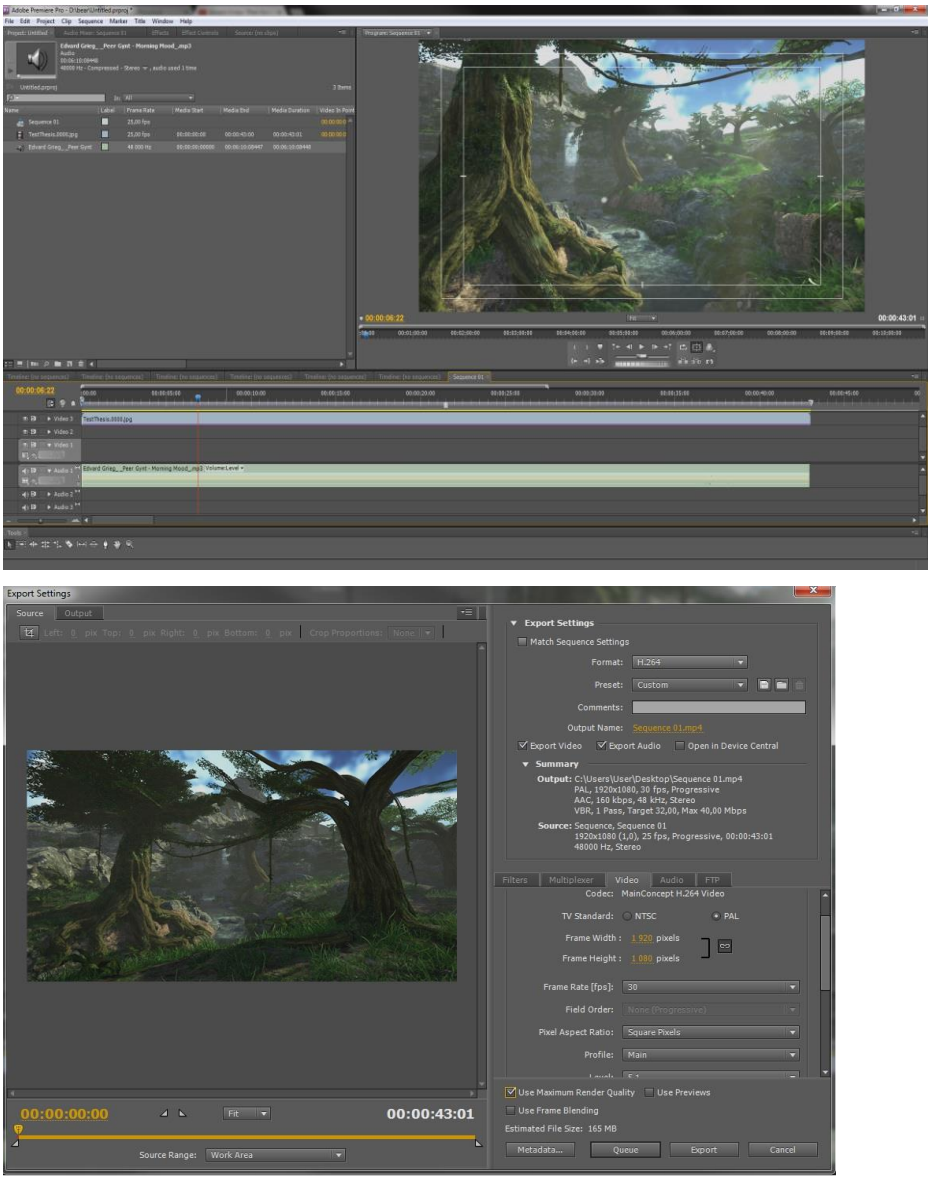


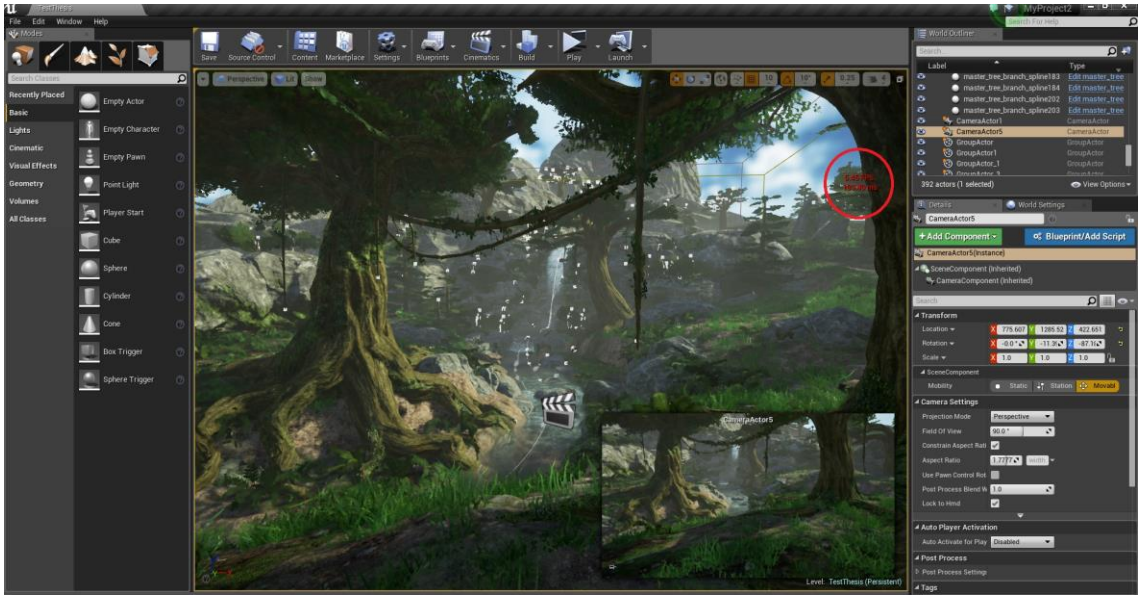
Figure 39. The final step, the final video was exported in 1920p x1080p (16:9 resolution).

4.6. Final and Future work

Even though the project was successfully deployed, there are still dozens of problems which could be improved in the future (Figure 40). For example, the Frame per second (Fps) of the project was dropped below ten frames per second, which was very effect the user view. Inside the red circle area, the system indicated the frame per second warning.

The reason of that suddent frame drop was not only due to low Graphic Proccesing Unit (GPU) as well as the Random Access Memory (RAM) from the computer, but also the numbers of props, models and foliage also increased the usage of computer's resources (Figure 40). In addition, those blueprints which were used to create water flow animation and wind effect were one of the main problems causing frame drop due to ”Event tick” function [3,55].

There are many solutions to those problems. Users could either improve their computer's hardware (GPU,RAM) or reduce the number of props and models used inside Unreal Engine. In addition, when developing games, users should be recommended not to use the unneccesary ”Event tick” function since it had an effect on frame per second.



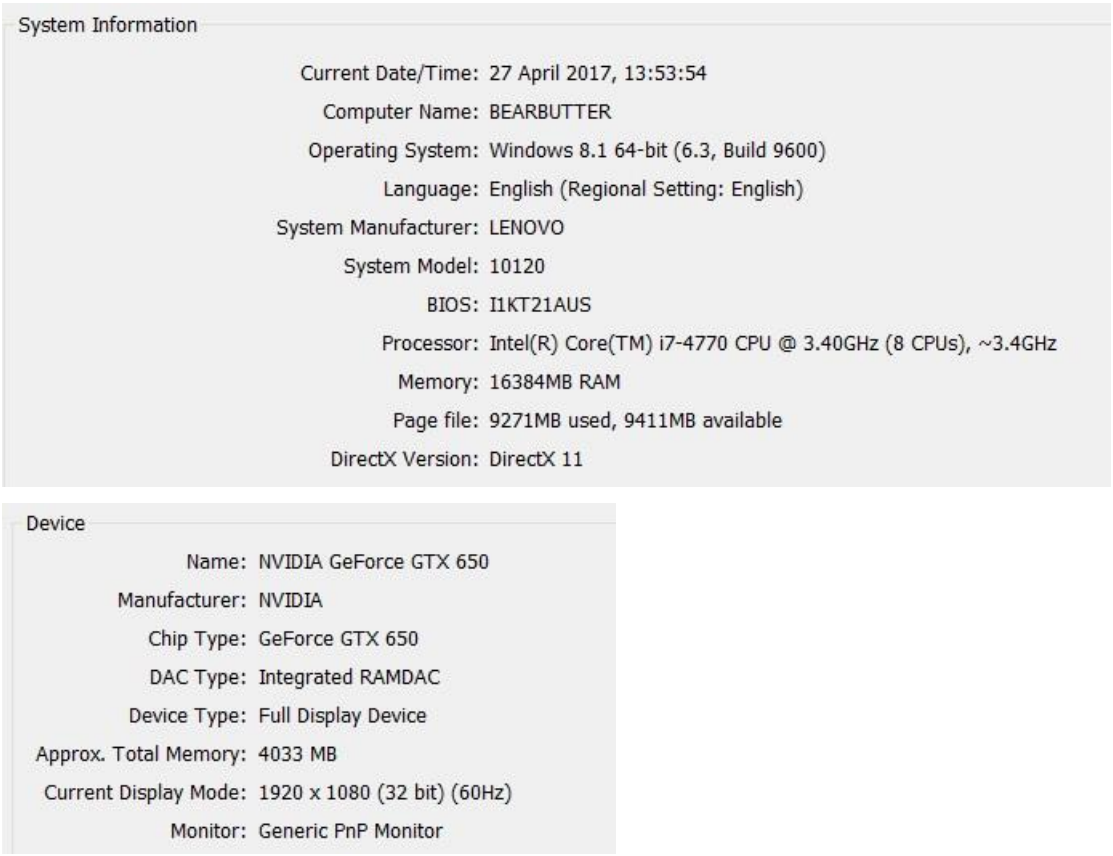


Figure 40. Frame per second dropped problem and the System Information of the computer which was used to build this project.

In this particular project, the purpose of the final product was meant to be a trailer so the frame per second was not a disadvantage. Because, as a video format, the frame would always stay at standard value twenty five frame per second.
To sum up, this project required:

- three weeks full time of work (Modelling, camera workflow and map creating included).
- Three hours of rendering time when the system boosted with 1075 images.
- Four hours of post process time.

The project can be reviewed at:
https://www.youtube.com/watch?v=vxq5xYbHwHE&feature=youtu.be.

5. Conclusion

Game Engine in general and Unreal Engine in particular not only plays an important role in game industry, but also in other related fields such as 3D advertisements, real estate advertisement and movie industry. It reduces the amount of time spent on game development and project costs. Moreover, the engine also connects programmers to graphic designers so they could both easily develop the same project by using the same engine.

The game engine cannot totally replace all other software. The users must also have knowledge in other software such as Adobe Photoshop, Autodesk 3d max, Autodesk Mudbox, Zbrush and Lightwave in order to maximize the benefit of the engine.

This thesis focuses on multiple ways to create different game levels. When creating 3D models for game levels, the creators need to practice 3D modeling to achieve enough knowledge to create playable levels. Finally, the Unreal Engine will certainly develop in the future because there is a need to create more complex games.

References

1. 23 recommended and available 3D game engines (updated); Category: Level design, Game environment design ; December 12, 2012 (Updated: July 28, 2016). URL:http://www.worldofleveldesign.com/categories/level_design_tutorials/recommended-game-engines.php Accessed in 22 January 2017.

2. Pete Ellis. Clever Level Design: More than Meets the Eye; 31 March 2016. URL: https://80.lv/articles/how-to-build-good-levels-for-games/ Accessed in 22 January 2017.

3. Unreal Engine 4 blueprints documents. URL:https://docs.unrealengine.com/latest/INT/Engine/Blueprints/UserGuide/Events/#eventtick Accessed in 28 March 2017.

4. Pluralsight. Keeping your player engaged-Tips for Great Game Level Design; Febuary 6, 2014.
URL: https://www.pluralsight.com/blog/film-games/keeping-players-engaged-tips-great-game-level-design Accessed in 3 Febuary 2017.

5. Jesse Schell. The Art of Game Design: A book of lenses; Chapter 9 to chapter 13.
URL: http://www.sg4adults.eu/files/art-game-design.pdf Accessed in 14 Febuary 2017.

6. New York Film Academy. Four Tips For Captivating Level Design; September 22, 2015
URL: https://www.nyfa.edu/student-resources/four-tips-for-captivating-level-design/ Accessed in 18 Febuary 2017.

7. Goohan. Top 10 Greatest Video Game Genres.
URL: https://www.thetoptens.com/video-game-genres/ Accessed in 5 March 2017.

8. Breakout Picture source.
URL: https://learnopengl.com/#!In-Practice/2D-Game/Breakout Accessed in 10 January 2017.

9. Nick Tylwalk. Top 20 Mobile Game Of All Time; 2014.

URL: http://gamesided.com/2013/08/24/top-20-mobile-games-of-all-time/ Accessed in 10 January 2017.

10. Mohit Kumar Rao. 2D City Background game art; 2016.
URL: http://mohitkumarrao.deviantart.com/art/city-2d-game-background-design-by-mohit-kumar-rao-623290321 Accessed in 10 January 2017.

11. Benji Edwards. Super Breakout 's Rainbow – Smashing Astronaut; October 16, 2006.
URL: http://www.vintagecomputing.com/index.php/archives/212 Accesed in 10 January 2017.

12. Potter1992. Level of Detail; September 26, 2013.
URL: https://potter1992.wordpress.com/2013/09/26/level-of-detail/ Accesed in 10 January 2017.

13. Retro Gamer Team. BattleZone; May 19, 2009.
URL: https://www.retrogamer.net/retro_games80/battlezone/ Accessed in 11 January 2017.

14. Rushabhgamedesign. Battle City (1990); January 7, 2013.
URL: http://rushabhgamedesign.blogspot.fi/2013/01/battle-city-tank-1990.html Accessed in 20 March 2017.

15. Felix Movie Thoughts. Mirror Edge Review.
URL: https://felixmoviethoughts.wordpress.com/2016/05/05/mirrors-edge-review/ Accessed in 20 January 2017.

16. Danny Biven & Bryan Rose. Super Mario Word Review Revisit; April 2, 2016
URL: http://www.nintendoworldreport.com/feature/42441/super-mario-world-review-revisit Accesed in 20 January 2017.

17. The Angry Bird picture source.
URL: http://angrybirdsbfm.weebly.com/transmedia-analysis-of-angry-birds.html
https://iamyourtargetdemographic.files.wordpress.com/2012/03/angry-birds-22.jpg Accesed in 20 January 2017.

18. Jordan NewMan. Level of Detail; September, 2013.
URL: https://jnewman96.wordpress.com/year-2/oo-design/level-of-detail/ Accessed in 20 January 2017.

19. Mike Williams. Resident Evil 7 E3 Demo: in Reality, It's Virtually P.T.
URL: http://www.usgamer.net/articles/resident-evil-7-e3-2016 Accesed in 21 January 2017.

20. Ernest Adams. Fundamentals of Game Design: Game Worlds; October 8, 2009.
URL: http://www.peachpit.com/articles/article.aspx?p=1398008&seqNum=3 Accessed in 21 January 2017.

21. Rostislav Nikolayev. Making An Award Winning in UE4; November 24, 2015.
URL: https://80.lv/articles/making-an-award-winning-scene-in-ue4-with-no-experience/ Accesed in 21 January 2017.

22. Pinterest. Explore vertical field, vertical space, and more!
URL: https://www.pinterest.com/pin/354869645610339224/ Accesed in 22 January 2017.

23. Cliff Bleszinski. History of Unreal Engine; Febuary 23, 2010.
URL: http://www.ign.com/articles/2010/02/23/history-of-the-unreal-engine?page=4 Accessed in 22 January 2017.

24. Classes.soe.ucsc.edu. History of Unreal Engine.
URL:https://classes.soe.ucsc.edu/cmps164/Spring09/ass1/btkachef/Unreal_Engine/History.html Accessed in 23 March 2017.

25. Unreal Engine. Patches released support.
URL: https://docs.unrealengine.com/latest/INT/Support/Builds/index.html
Accessed in 22 March 2017.

26. Joy Bossardet. The best and the worst of androids free zombie games; 2013.
URL: http://unrealitymag.com/video-games/the-best-and-the-worst-of-androids-free-zombie-games/ Accessed in 22 March 2017.

27. Simon Miller. Hitman Review; Febuary 2, 2017.
URL: http://www.trustedreviews.com/hitman-review Accessed in 22 March 2017.

28. Charles Onyett. The Elder Scrolls V: Skyrim Review; November 10, 2011.
URL: http://www.ign.com/articles/2011/11/10/the-elder-scrolls-v-skyrim-review
Accessed in 22 March 2017.

29. Matthew Rorie. Call of Duty 2 Review; January 18, 2006.
URL: https://www.gamespot.com/articles/call-of-duty-2-walkthrough/1100-6136916/
Accessed in 23 March 2017.

30. Chris Plante. Need For Speed: Most Wanted Review; October 30, 2012.
URL: http://www.polygon.com/2012/10/30/3575034/need-for-speed-most-wanted-review Accessed in 22 March 2017.

31. Jeff Gerstmann. Need For Speed: Most Wanted Review; October 30, 2012.
URL: https://www.giantbomb.com/reviews/need-for-speed-most-wanted-review/1900-537/ Accessed in 23 March 2017.

32. Chris Plante. Resident Evel 7 Boss Fight Director Koshi Nakanishi Interview; Febuary 20, 2017
URL: http://www.theverge.com/2017/2/20/14668330/resident-evil-7-boss-fight-director-koshi-nakanishi-interview Accessed in 23 March 2017.

33. Rosh Kelly. Urban Chaos: Most relevant police video game; October 17, 2016.
URL: https://www.vice.com/sv/article/why-2006s-urban-chaos-riot-response-is-2016s-most-relevant-police-video-game Accessed in 23 March 2017.

34. Metascore. The Legend of Zelda: Breath of the Wild game score rating.
URL: http://www.metacritic.com/game/switch/the-legend-of-zelda-breath-of-the-wild
Accessed in 23 March 2017.

35. Alex Hern. The Legend of Zelda: Breath of the Wild review; March 2, 2017
URL: https://www.theguardian.com/technology/2017/mar/02/the-legend-of-zelda-breath-of-the-wild-review-link-nintendo-switch Accessed in 23 March 2017.

36. Reddit. StarWar Battlefront First person vs Third Person threads. URL:https://www.reddit.com/r/StarWarsBattlefront/comments/3ta0gz/first_person_vs_third_person/ Accessed in 15 April 2017.

37. StarWar Battlefront picture source. URL: http://www.gamepur.com/news/18826-new-star-wars-battlefront-1080p-screenshot-shows-difference-between-first.html Accessed in 23 March 2017.