3D Modeling and videogame adaptation of the UJI Library for Smart UJI

David Castellà Corral

Final Degree Work
Bachelor's Degree in
Video Game Design and Development
Universitat Jaume I

May 22, 2022

Supervised by: Michael Gould.

To my parents, whose love gives me strength

Acknowledgments

First of all, I would like to thank my Final Degree Work supervisor, Michael Gould, for his support during the project and for recommending me the project's idea.

Secondly, I also would like to thank Juan Camilo Gómez for helping me obtain the licenses and providing me with tutorials and information about ArcGIS' products. And I also want to acknowledge Lluc Gauxachs Sanz, for taking the time to help me with CityEngine and showing me SketchUp.

Finally, I want to thank Sergio Barrachina Mir and José Vte. Martí Avilés for their inspiring LaTeX template for writing the Final Degree Work report, which I have used as a starting point in writing this report.

Abstract

This document is a detailed explanation of every step of the project that I have been working on for the past months, which aimed to create a 3D model of the Universitat Jaume I library using ArcGIS' CityEngine to be used on the Smart UJI project and to reuse the same model in a stealth videogame created with Unity.

The end result is, of course, the 3D model of both the interior and exterior of the UJI library and the videogame made in Unity that uses the library as its main environment where the action is located.

Contents

Chapter 1

Introduction

Contents

This first chapter's purpose is to state the main objectives of the project, but also the motivations behind choosing this project and working on it and what was the state from where it was started.

1.1 Work Motivation

The idea of the project was proposed by my supervisor, Michael Gould. He said that there was a university group called Geotec [1] that were working on a project called Smart UJI [2], which aimed to have a complete 3D environment of the college facilities. And so he explained the idea to model one of the buildings (in this case, the library) so it could be added to the project and then develop a videogame around the completed model.

That particular idea was chosen for a number of reasons: the first one and the most motivational one was that it gave creative freedom to do whatever videogame was wanted, and the second one was that by doing this project it would be possible to learn a variety of new programs that are rarely taught and talked about, like ArcGIS' CityEngine [3].

1.2 Objectives

As stated on the previous section, the reason behind the 3D model is so it can be used for Smart UJI, a project developed by the GEOTEC group, a group devoted to research in geospatial technologies. Smart UJI is, in short, a 3D environment filled with various 3D models placed in order to replicate the college as accurately as possible. And that's where the first and most important objective of this project stems from: to create a 3D model of the library so that it can replace the one that is currently being used on the Smart UJI environment.

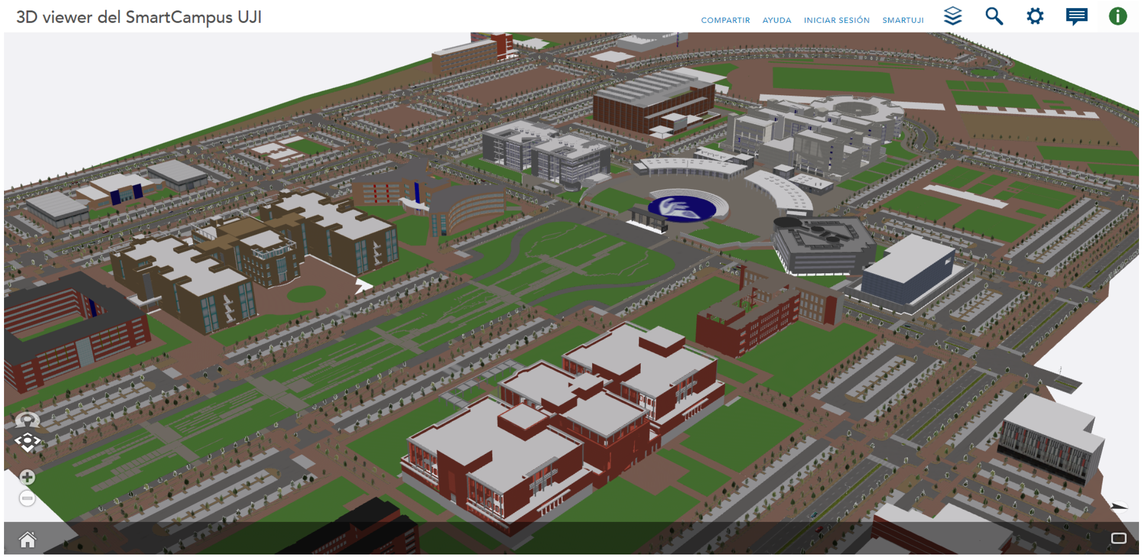


Figure 1.1: Aerial View of the Smart UJI project

That is one of the main objectives of this project, but of course there are many others. The second one, as also said previously, is to develop a first person stealth game in Unity that showcases one's knowledge as a developer and what their skills and strong points are. The game should also serve as a way to explore the interior of the library easily, since Smart UJI doesn't really let users move around the inside of the building. The third and final objective, which is a bit more abstract than the other two, is to become proficient at using CityEngine by the end of the project.

It is widely known that CityEngine is a very interesting software that can have a lot of uses in the videogame industry and so the final objective for this project is to at the very least have a decent understanding of how the program works and how to use it.

1.3 Environment and Initial State

The project started with the installation of CityEngine, thanks to the help and licenses provided by a member of Geotec. Along with those two, some files containing information about the library and the Smart UJI project were also sent. Those were a geodatabase file with all the 2D data about the campus, a zip file with the CityEngine project that Smart UJI was currently using, the UJI's architectural plans and some specific data about the library including some .dwg files. The work started on the CityEngine project that was provided, and it's worth noting the state of the file when it was received.

The 3D model that was inside the file had quite a few errors. The walls were flat and had no volume, a bunch of textures were missing or incorrect, many planes were displaced or even deleted, etc. And even the aspects of the model that were functional weren't the most correct, as they had many clipping issues, low precision issues, some vertices were connected to others they shouldn't be, and many more. However, it was better than starting from scratch, and so the project was started there. Here's a few images of the initial state of the model:

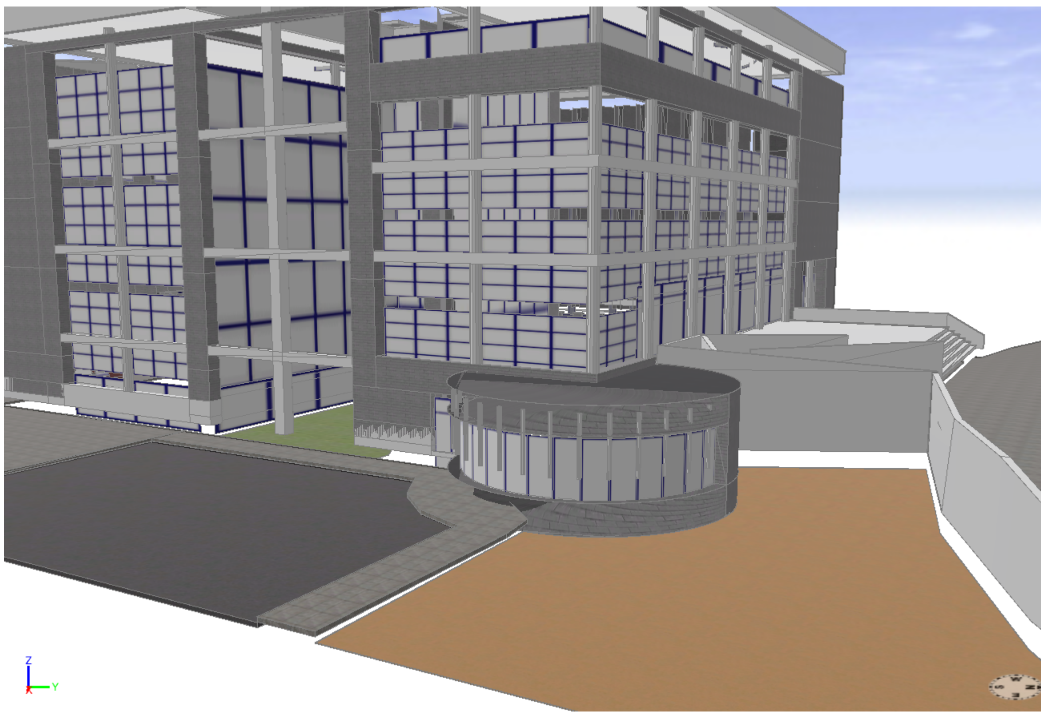


Figure 1.2: Initial State of the Library on CityEngine

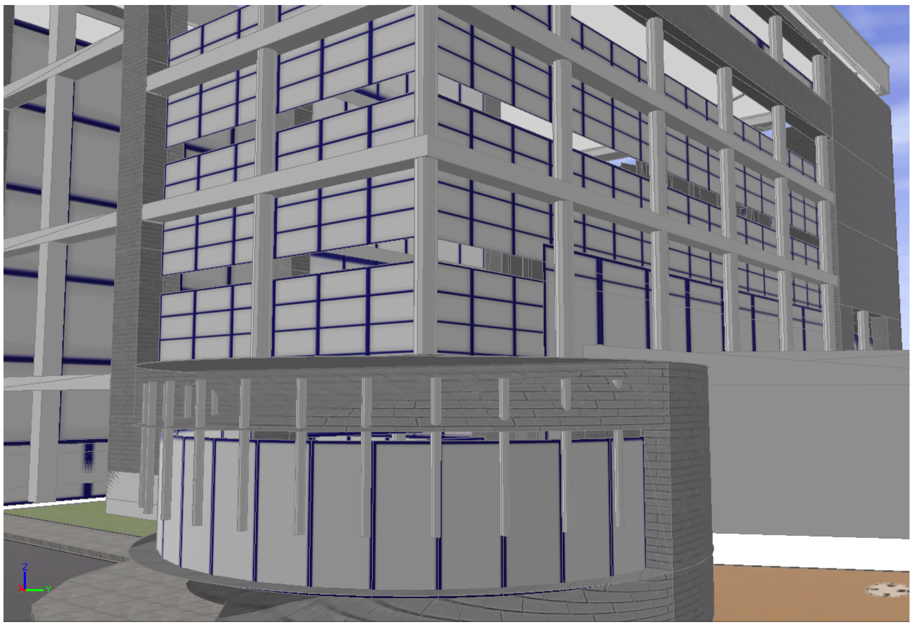


Figure 1.3: Initial State of the Library on CityEngine

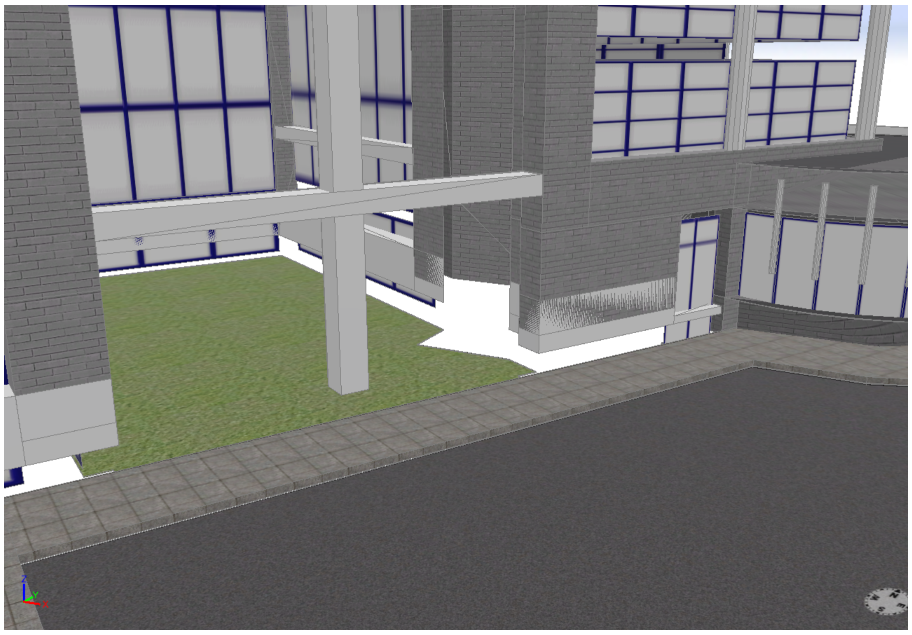


Figure 1.4: Initial State of the Library on CityEngine

Chapter 2

Planning and resources evaluation

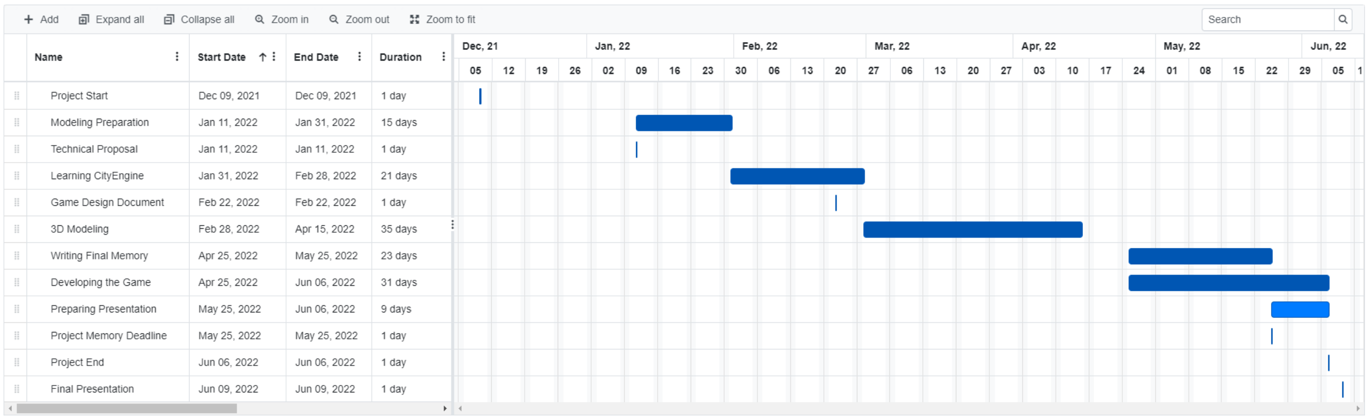
Contents

This second chapter will talk about what the planning for the project was originally and how that planning changed as time went on. There will also be a section talking about what resources have been used in the making of this project.

2.1 Initial Planning

The planning for this project was sort of difficult to begin with, seeing as there was so many tasks to be done and so little time. There was an initial planning that was made at the beginning of the year, but of course it's near impossible to follow a schedule perfectly and so there are a few differences between the ideal planning and the real one that ended up as the final version. On the next page there's a Gantt Chart that shows what the definitive planning ended up looking like. (see Figure 2.1)

The diagram is easy enough to understand. There are six main tasks that had to be completed for the project: prepare the modeling, learn how to use CityEngine, actually model the library (interior and exterior), write the final memory, develop the game and finally prepare the presentation.



Name	Start Date	End Date	Duration
Project Start	Dec 09, 2021	Dec 09, 2021	1 day
Modeling Preparation	Jan 11, 2022	Jan 31, 2022	15 days
Technical Proposal	Jan 11, 2022	Jan 11, 2022	1 day
Learning CityEngine	Jan 31, 2022	Feb 28, 2022	21 days
Game Design Document	Feb 22, 2022	Feb 22, 2022	1 day
3D Modeling	Feb 28, 2022	Apr 15, 2022	35 days
Writing Final Memory	Apr 25, 2022	May 25, 2022	23 days
Developing the Game	Apr 25, 2022	Jun 06, 2022	31 days
Preparing Presentation	May 25, 2022	Jun 06, 2022	9 days
Project Memory Deadline	May 25, 2022	May 25, 2022	1 day
Project End	Jun 06, 2022	Jun 06, 2022	1 day
Final Presentation	Jun 09, 2022	Jun 09, 2022	1 day

Figure 2.1: Project's Gantt Chart

Those tasks were all mentioned in the initial planning, though they didn't have fixed dates or even a schedule. Each task was assigned a rough estimate of hours, as seen in the chart below:

Project Task	**Estimated Hours**
Modeling Preparation	5
Learning CityEngine	25
3D Modeling	90
Writing Final Memory	20
Developing the Game	140
Preparing Presentation	10

These estimations turned out to not be the most accurate. Many tasks ended up taking a few more hours than was planned for, since the scale of the project was underestimated. In fact, it can be said that the recommended 300 hours that this Final Degree Work was intended to take have been far surpassed.

2.2 Detailing Tasks

While the project scheduling and deadlines ended being the same and were all met, the workflow and total hours of work that were invested into the project were different than it was intended. So, in this section there will be an brief explanation for what each task entailed and the work that was done. Next to each task will also be the actual number of hours that it took to complete. They are all in chronological order, so let's start:

- **Modeling Preparation (5h):** This task includes installing every software necessary for the project and both downloading and preparing all necessary files in order to begin working.

- **Learning CityEngine (40h):** CityEngine is a program that has not been taught here in UJI, and since it's not the easiest of programs it took a while to become accustomed to it. Luckily, the same Geotec member sent some tutorials from the official ArcGIS website [4] that were very helpful and showcased all the capabilities that the software provides.

- **3D Modeling (120h):** This ended up being the main bulk of the project, the task that took the longest to complete. Even though it was thought on the initial planning that developing the game would surely take longer, it wasn't taken into account that personal experience played a big part into how long each task would take. So even though developing the game should in theory be the harder task, the extra experience with Unity and programming in general and the lack of experience with CityEngine and SketchUp tipped the balance. This task includes both fixing the exterior of the library that was on the CityEngine project and modeling the interior of the library from scratch.

- **Writing Final Memory (50h):** Even though the name here is "writing final memory", it also includes all other documents inside this task. This is another task that took way longer than it was planned for, since it wasn't known that LaTeX was a program required in order to write the final memory. Without previous experience with LaTeX, learning how the software works slowed the project down quite a bit.

- **Developing the Game (110h):** The second highest task, in number of hours worked. In this one it is also included all the design for the game, which means both thinking what the game is going to be and translating that idea to paper are accounted in this task. The remaining hours were spent on the actual development on Unity. From animations, to UI, to programming, to many more. Every task done inside the Unity engine is considered here.

- **Preparing Presentation (15h):** The final task of the list, which is pretty much creating the presentation for the Final Degree Work and studying it enough to a point where one would feel comfortable talking about it.

Those are all the tasks that needed to be realized for this project. All in all the planning worked out quite nicely and even though it took more time than expected the quality of the project did not suffer because of it.

2.3 Resource Evaluation

This final section is a listing of every resource that was used in making this project and also contains an estimate of what the economic cost would have been without having a single resource beforehand.

The first and most important resource needed to develop the project was of course the hardware. This following list contains the specifications of the computer that was used for the project:

- **CPU:** Intel(R) Core(TM) i5-8400
- **GPU:** Gigabyte NVIDIA GeForce GTX 1060 3GB
- **RAM:** G.Skill Trident Z RGB DDR4 3200 PC4-25600 16GB 2x8GB CL16
- **Hard Drive:** Western Digital WD10EZEX - 1 TB (SATA, 3.5", Buffer 64 MB)

The hardware isn't the most up to date, though most people would consider it decent by modern standards. You could complete this whole project with a worse or better computer, but it is worth noting that there were some performance issues with CityEngine. Sometimes trying to extrude a surface would freeze the program, and once it finally unfroze the extrude would be either nonexistent or incorrect. It's probably true that having an HDD hard drive would have probably solved these issues; but it was not necessary to complete the project. Worth noting that a computer with these components has an estimated cost of around 900€ nowadays.

Next up on the list is the software, and the amount of programs needed to complete this project is quite large:

- **CityEngine:** This is a software developed and distributed by ESRI (Environmental Systems Research Institute), a business focused on developing and distributing software for geographic information systems. Their family of software is named ArcGIS, and CityEngine is a program from that family. CityEngine specializes in the generation of 3D urban environments, which is why it is being used for the Smart UJI project. This is the software that was used to open said project and where it was worked on the exterior of the library.
- **SketchUp [5]:** SketchUp is a 3D modeling and graphic design software, similar to others like 3DS Max and Blender. It was used in combination with CityEngine to model the interior of the library. There are a number of reasons why it wasn't just CityEngine being used for that purpose, but they will be touched upon later.
- **Unity [6]:** Unity was the game engine chosen to develop the videogame. Many others could've been chosen, like Unreal, but Unity was personally chosen because it was the one learned throughout the degree and had more time been spent to learn Unreal the project wouldn't have been able to be finished before the deadline.

- **Google Drive [7]:** This is an online cloud data storage, where not only you can store whatever files you wish to but also use their builtin editors for documents, presentations and others. It was used to store backups of some of the files, and some of their online editors were also used to write a few documents and the final presentation.
- **GitHub [8]:** This is an online platform commonly used to store projects using the Git system. It's where the Unity project has been stored because it's safer to keep the project stored both on GitHub and on the computer than only on one. It also allows to change versions in case something goes wrong while developing.
- **Blender [9]:** Blender is very similar to SketchUp, another 3D modeling software previously mentioned. The use of Blender was very scarce, but it is worth mentioning since it was used to fix some minor details about the library model before finally importing it to Unity.
- **Visual Studio [10]:** Visual Studio is the environment used to program scripts for Unity. It is one of the most important parts of the project, since without it the job of coding the game would have been much harder.
- **Google Meet [11]:** Google Meet is a platform that allows users to do video calls with other users on the same platform. This software has been used exclusively to contact with the supervisor for any meetings or to do regular checks on how the project was going.
- **Google Maps [12]:** Google Maps is an online map viewer that allows the user to check out any maps from the earth that they wish to see, and also check out any place in Street View. This was used to see the library's exterior as a point of reference when working on CityEngine.
- **Overleaf [13]:** Overleaf is an online LaTeX editor. It was used to write this document using the LaTeX template provided by the university. There are other LaTeX editors but this one was chosen because it doesn't require installation and it is free.
- **Discord [14]:** This is the last software that will be mentioned, and though its use was limited to pretty much only two days of the project, the role it played in bringing this project to fruition can't be understated. It was used to talk to ex-students from the university that had done their Final Degree Work with the same supervisor as me. Their counseling and overall help was invaluable, and so this is a thanks to both Lluc and Francisco for their time and attention.

Considering all the hardware and software that was used for this project and the individual cost of every single one of them, the estimated cost of the entirety of them is about 4500€. The item with the highest cost is CityEngine, with a cost of around 3.5k€, but since it is essential for this project there is no workaround.

Chapter 3

System Analysis and Design

Contents

This chapter's purpose is to present a complete analysis of the videogame and every part of it. Design, systems, interfaces and others are the main aspects that will be touched upon in these next sections.

3.1 Brief Videogame Summary

There are many systems and design choices that will be explained later, but it will be hard to understand them without proper information about the videogame and its characteristics. This section serves as a short description of the game, whose name is "Book Thief".

3.1.1 Initial Concept

The first concept that existed for the game was somewhere between horror and stealth for the genre. In the end a stealth game was chosen because of having more experience with the genre and not being a huge fan of horror games. The main inspiration for the game was "PAYDAY 2" [15], one of the most known stealth games in videogame history. Many of the mechanics in PAYDAY 2 fit this vision of a stealth game and the library the best, and so most of the mechanics and overall gameplay resemble those from PAYDAY 2 the most.

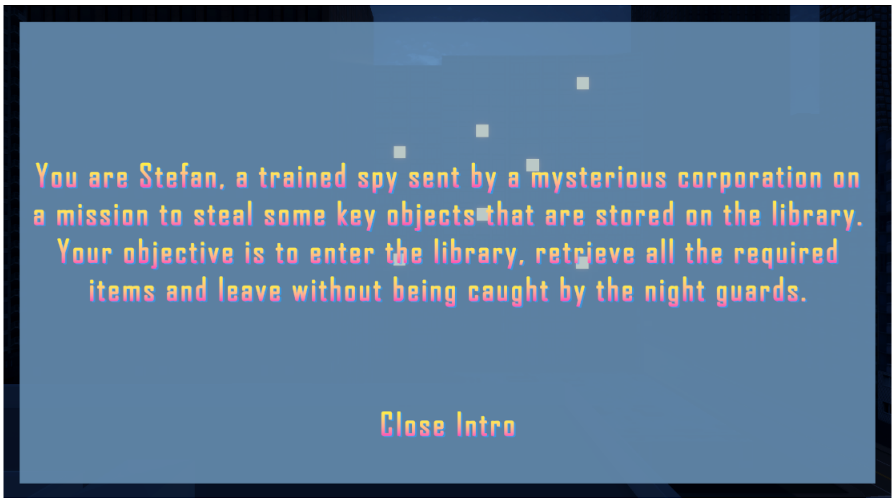
Figure 3.1: Gameplay Image of PAYDAY 2

3.1.2 Story Elements

Book Thief has a very simple story that is explained before the gameplay starts. You're Stefan, a professional thief sent by a mysterious corporation on a mission to steal some key objects that are stored on the library. Your objective is to enter the library, retrieve all the required items and leave without being caught by the night guards.

The story is very brief and simple, but that's because the game has a focus on gameplay. Such a short stealth game wouldn't really benefit from having a more complex story.

It would also be hard to somehow tell the story through gameplay without introducing cinematics or other methods. So it was chosen to just show the player a brief text before giving them control of the character, just to get them a bit more immersed in the game.



You are Stefan, a trained spy sent by a mysterious corporation on a mission to steal some key objects that are stored on the library. Your objective is to enter the library, retrieve all the required items and leave without being caught by the night guards.

Figure 3.2: Story Screen

3.1.3 Level Design

As stated on previous chapters, the main environment the game will take place in is the library. After the user presses the "Start" button on the main menu and the story summary shows up, the player character will spawn on the outside of the library. There will be 3 main zones: the outside, where the library and other buildings will be located; the first floor of the library, and the second floor of the library.

The outside of the library is the starting area, and as such there will be no night guard NPCs around it. The player can roam around the area to look around the library and see what entrance they prefer to use to get inside the building. The first and second floors of the library is where the action will happen and where the player has to complete the main objective of the game.

3.1.4 Gameplay and Mechanics

The gameplay is based around stealth. The player is forced to avoid the guards that patrol around the library, there is no other option. While dodging the enemy NPCs, they have to complete the main objective of the game which is to collect every single key item. These objects are scattered around the interior of the library, so the player will have to pretty much explore the entire library in order to finish the game. The premise of the game might seem difficult at first, but of course there are many mechanics that will help the player.

The first set of mechanics worth mentioning are tied to the movement of the character. In order to make moving around the guards an easier task, the player can both sprint and crouch. Sprinting makes you move faster but make more noise, and crouching makes you slower but less noisy. The player has to combine both these mechanics and choose when to use them to more easily avoid the enemy NPCs. There's another movement mechanic, jumping, though that one is mostly useless and it was only introduced as a novelty option.

A mechanic mentioned earlier is the noise system. The player has different amounts of noise they can be making, and the higher the volume the bigger the range of detection of the night guards is. So even though sprinting is the fastest method of movement, it's near impossible to sprint around every enemy NPC since they will hear you from very far away.

Obviously, if the videogame is from the stealth genre, then one of the main mechanics is hiding. Every night guard has a frontal area of detection, their eyesight. In order to not be seen, the player has to stay either behind them or behind cover.
There are many spots on the library where the player can take cover by simply crouching to avoid detection.

Another mechanic that makes the player have more control over their actions is marking. With the press of a key while aiming at a night guard, the mechanic activates and that guard becomes marked. A player can see all enemy NPCs that are marked at all times, even through walls. That allows the user to make better decisions based on the position of the night guards, and see where they are without losing cover.

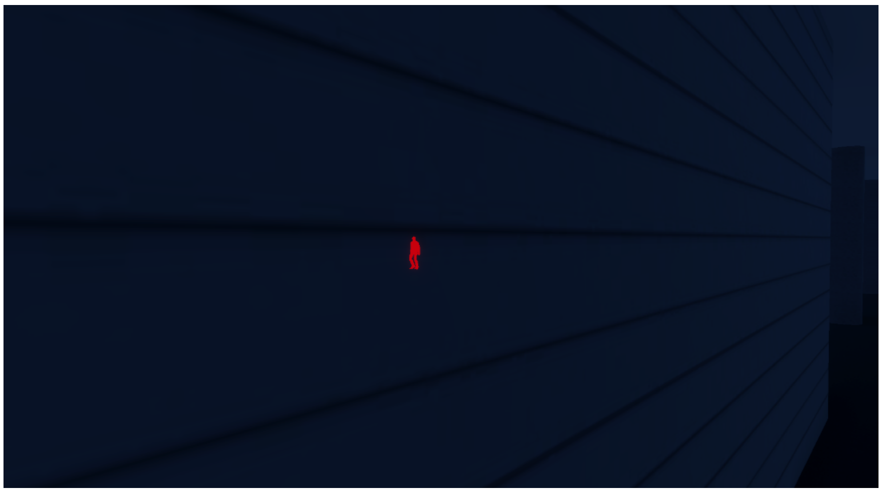


Figure 3.3: Marked Enemy shown through wall

The final mechanic mentioned here is very simple, interacting with objects. The player obviously needs to collect all the key items in order to win. To obtain them, it's as simple as getting close to them and pressing a key. They will be added to the player's inventory.

3.1.5 Art and Sound Design

This final subsection talks about pretty much all other aspects of the game. Programming is the area of expertise, and gameplay will be the main focus of Book Thief. But that doesn't mean the game won't have good characters or models, though they won't be made in this project. Not only because of a clear lack of skills, but also because time is limited and it would be almost impossible to work on modeling the library, programming the game and also modeling the character and any other objects in the game.

There needs to be a character that the player can control, key objects that they can collect and also some furniture and other miscellaneous objects to decorate the library. Most of these models will be acquired from the Unity Asset Store [16]. It is an online library filled with assets for Unity that you can obtain and use them on your personal Unity project. Since my economic resources are very limited, the assets used for Book Thief will all be free.

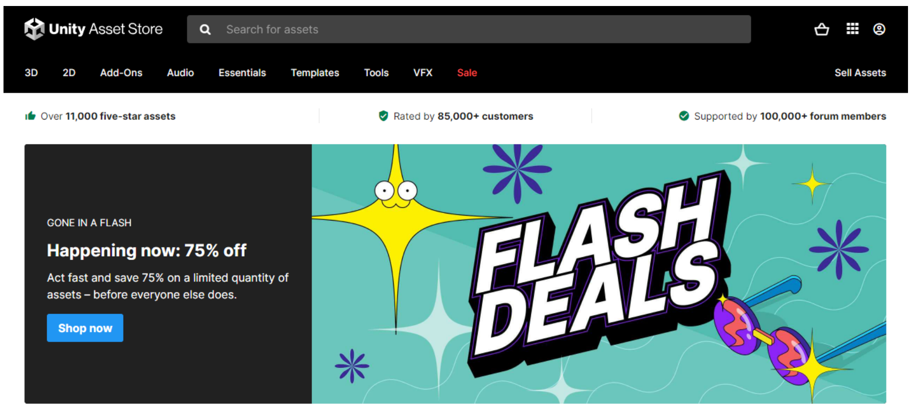

Unity Asset Store
Search for assets
3D
2D
Add-Ons
Audio
Essentials
Templates
Tools
VFX
Sale
Sell Assets
Over 11,000 five-star assets
Rated by 85,000+ customers
Supported by 100,000+ forum members
GONE IN A FLASH
Happening now: 75% off
Act fast and save 75% on a limited quantity of assets – before everyone else does.
Shop now
FLASH DEALS

Figure 3.4: Unity Asset Store page

A similar method will be used for the sounds of the game. This time, they will be procured from the Open Game Arts [17], another online library that has a specific sound collection with many files than you can download and then use on your Unity project (or other game engine).

3.2 Requirement Analysis

This section will be a detailed listing of every requirement of the videogame, both functional and non-functional. Designing and developing a game from scratch is not an easy feat, so it is heavily recommended to have a list of requirements before working inside the game engine to know exactly what needs to be implemented and how to do it.

The two types of requirements have been stated previously, and they are quite different. Functional requirements are, in easy to understand terms, simple sentences that define what the system (in this case, the videogame) should do. Non-functional requirements are conditions that the system should meet, and they're usually related to either performance or reliability constraints; in the case of videogames.

These next two subsections will contain a list of every single requirement about the videogame, stated in simple short sentences. There's a plethora of ways to represent requirements, but this one has been personally chosen for its ease of reading.

3.2.1 Functional Requirements

The requirements will be all numbered, since it will make the next section easier to read and understand. Here are the functional requirements:

- **R1:** The player can move around.
- **R2:** The player can crouch.
- **R3:** The player can jump.
- **R4:** The player can sprint.
- **R5:** The player can pick up objects.
- **R6:** The player can quit the game.
- **R7:** The player can mark enemy NPCs.
- **R8:** The player can start the game from the main menu.
- **R9:** The enemy NPCs can spot the player.
- **R10:** The enemy NPCs can chase the player.
- **R11:** The player can change settings from the main menu.
- **R12:** The player can finish the game.

3.2.2 Non-functional Requirements

And here are the non-functional requirements:

- **R13:** The game will run on stable FPS.
- **R14:** The game will have proper illumination.
- **R15:** The loading screens will be shorter than 5 seconds.
- **R16:** The game will be playable on lower end PCs.
- **R17:** The interface will be simple and clean.
- **R18:** The difficulty will be challenging.
- **R19:** The movement will be smooth.

3.3 System Design

This section presents the entire design of the videogame, both logical and operational. Some diagrams and charts will be used to simplify this section, and it starts with a detailed analysis of the functional requirements using Case Use charts:

Requirement	R1
Description	The player can move around.
Actors	Player
Preconditions	1. The player must be not on a menu.
Normal Sequence	1. The player presses one of the movement keys. 2. The player moves in the correspondent direction.
Alternative Sequence	1. The player presses one of the movement keys. 2. The player doesn't move because of collisions with other objects.

Requirement	R2
Description	The player can crouch.
Actors	Player
Preconditions	1. The player must be not on a menu.
Normal Sequence	1. The player presses the crouch key. 2. The player character crouches.
Alternative Sequence	1. None.

Requirement	R3
Description	The player can jump.
Actors	Player
Preconditions	1. The player must be not on a menu.
Normal Sequence	1. The player presses the jump key. 2. The player character jumps.
Alternative Sequence	1. The player presses the jump key. 2. The player character doesn't jump because of collisions with other objects.

Requirement	R4
Description	The player can sprint.
Actors	Player
Preconditions	1. The player must be not on a menu.
Normal Sequence	1. The player presses the sprint key and a movement key. 2. The player character sprints on the corresponding direction.
Alternative Sequence	1. The player presses the sprint key and a movement key. 2. The player character doesn't sprint because of collisions with other objects.

Requirement	R5
Description	The player can pick up objects.
Actors	Player
Preconditions	1. The player must be not on a menu. 2. The player must be aiming at an object and close to it.
Normal Sequence	1. The player presses the pick up key. 2. The object gets added to the player inventory.
Alternative Sequence	1. None.

Requirement	R6
Description	The player can quit the game.
Actors	Player
Preconditions	1. The player must be on the pause or main menu.
Normal Sequence	1. The player clicks on the "Quit Game" button.
Alternative Sequence	1. None.

Requirement	R7
Description	The player can mark enemy NPCs.
Actors	Player
Preconditions	1. The player must be not on a menu. 2. The player must be aiming at an enemy NPC and have a clear line of sight.
Normal Sequence	1. The player presses the mark key. 2. The enemy NPC gets marked for the player to see.
Alternative Sequence	1. None.

Requirement	R8
Description	The player can start the game from the main menu.
Actors	Player
Preconditions	1. The player must on the main menu.
Normal Sequence	1. The player presses the "Start Game" button. 2. The game starts.
Alternative Sequence	1. None.

Requirement	R9
Description	The enemy NPCs can spot the player.
Actors	Enemies
Preconditions	1. The player must be on their line of sight. 2. The player must be in range.
Normal Sequence	1. The player moves into the enemy's line of sight. 2. The enemy NPC spots the player.
Alternative Sequence	1. The player moves into the enemy's line of sight. 2. The enemy can't spot the player because they're behind cover.

Requirement	R10
Description	The enemy NPCs can chase the player.
Actors	Enemies
Preconditions	1. The player must be on their line of sight. 2. The player must be in range. 3. The player must be spotted.
Normal Sequence	1. The enemy spots the player. 2. The enemy starts chasing the player.
Alternative Sequence	1. The enemy spots the player. 2. The enemy can't chase the player because they've moved behind cover.

Requirement	R11
Description	The player can change settings from the main menu.
Actors	Player
Preconditions	1. The player must be on the main menu.
Normal Sequence	1. The player clicks on the settings button. 2. The player changes the settings.
Alternative Sequence	1. None.

Requirement	R12
Description	The player can finish the game.
Actors	Player
Preconditions	1. The player must be on the exit. 2. The player must have all key items in their inventory.
Normal Sequence	1. The player steps onto the exit. 2. The game ends.
Alternative Sequence	1. None.

To wrap up this section there's an activity diagram on the next page that gives some more details about the gameplay loop and its structure.

3.4 System Architecture

This section is a short description of the hardware and software requirements that the system needs to meet in order to be able to play Book Thief properly. The computer specifications have been listed in a previous chapter, but they will also be shown here for easier reading:

- **CPU:** Intel(R) Core(TM) i5-8400
- **GPU:** Gigabyte NVIDIA GeForce GTX 1060 3GB
- **RAM:** G.Skill Trident Z RGB DDR4 3200 PC4-25600 16GB 2x8GB CL16
- **Hard Drive:** Western Digital WD10EZEX - 1 TB (SATA, 3.5", Buffer 64 MB)

But obviously, the game doesn't require nearly as much. Book Thief is not really a complicated or resource intensive game, so the minimum requirements in order to play the game are way lower than the ones listed before. It's very hard to know exactly what they are, since there would have to be tests on multiple computers with different components and check their performance. But there's a list of minimum requirements on the official Unity site in order to run games developed in Unity. Here's the list:

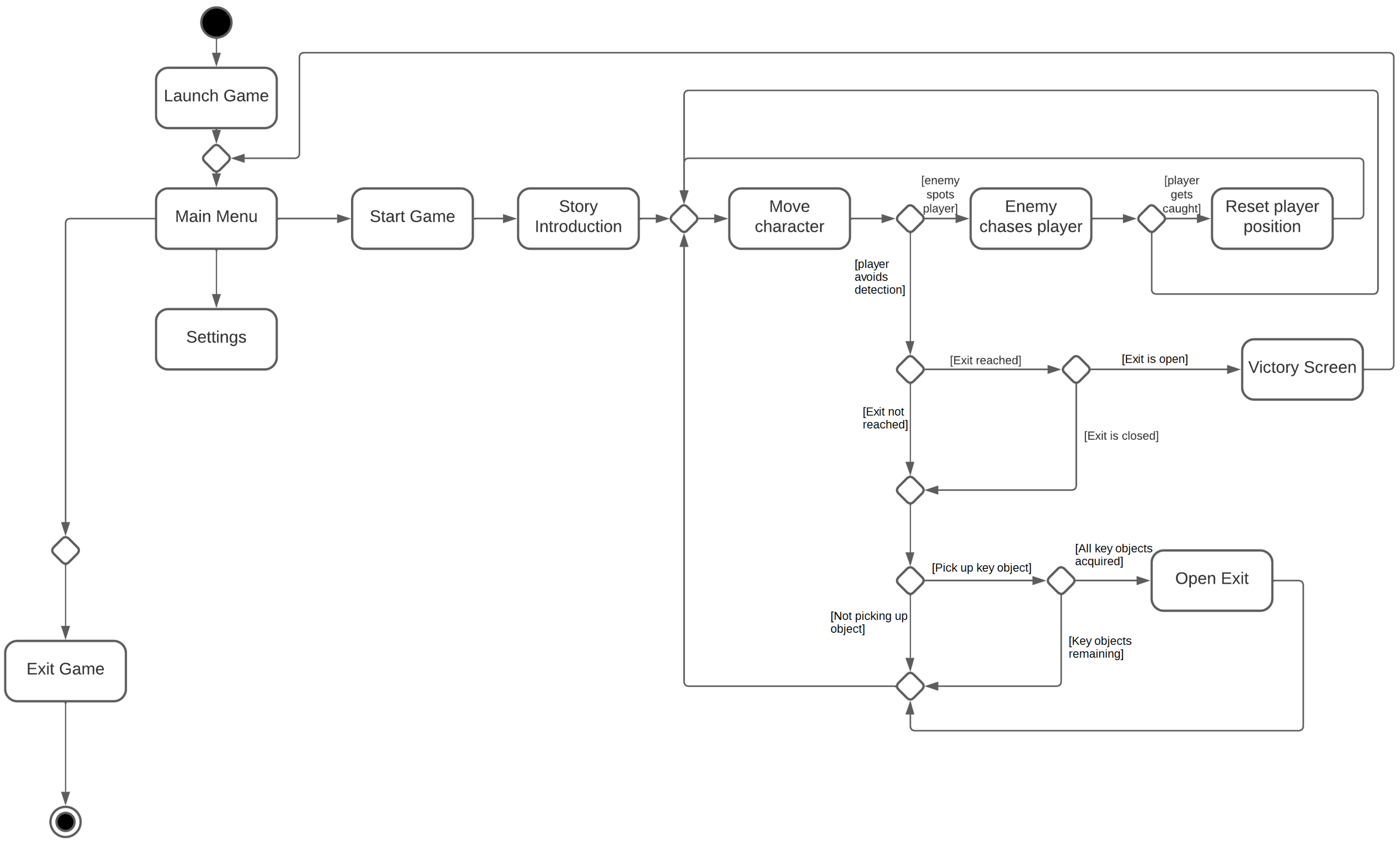


Figure 3.5: Activity diagram of Book Thief

- **OS:** Windows 7 SP1+, macOS 10.13+, Ubuntu 18.04+
- **GPU:** Graphics card with DX10 (shader model 4.0) capabilities.
- **CPU:** SSE2 instruction set support.

These requirements are for Unity version 2020.3.13f1, which is the version that Book Thief runs on. They are very low requirements, but the Unity site also states that "how well it (the game) runs depends on the complexity of the project" [18]. A rough estimate says that any computer that isn't older than 7 years will be able to run the game at a stable 60 FPS. It is however worth mentioning that keyboard and mouse are necessary, since there is no controller support for the game.

3.5 Interface Design

This final section of the chapter will serve as a showcase of the interface (also known as UI) of Book Thief. The first UI element the player can see is the main menu and all of its buttons. The player can click on them to either start the game, open the options or close the game. The background is an image from the game and the buttons are very minimalist, in order to draw the players' focus towards the background and the title of the game.

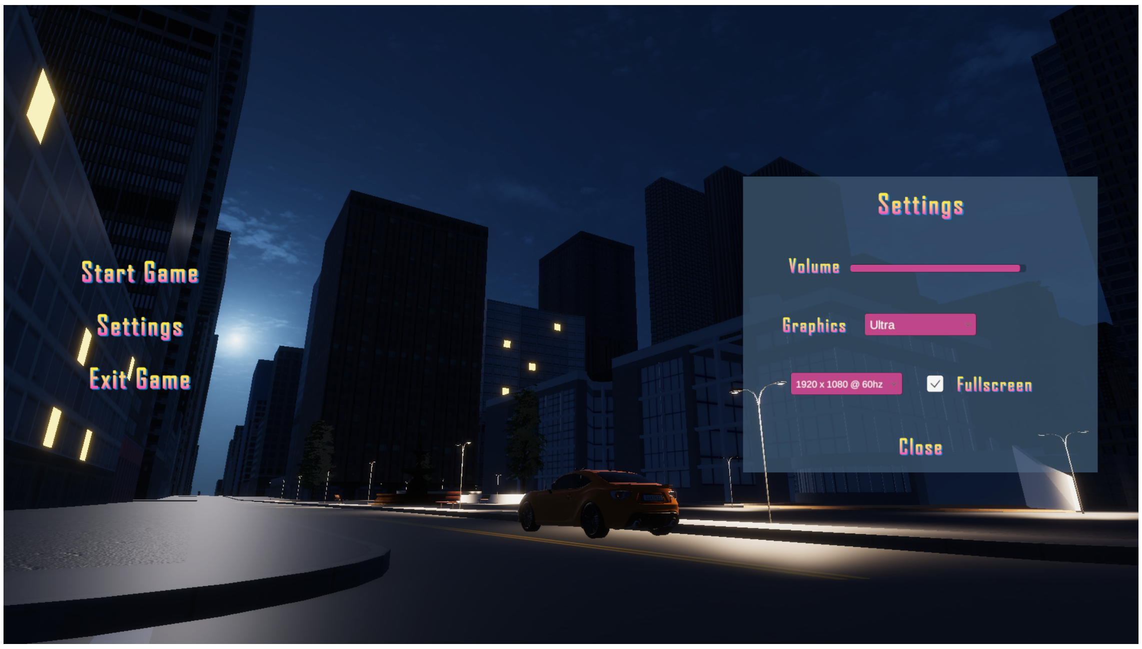


Figure 3.6: Main Menu of Book Thief

Inside the game itself, there's a few more UI elements that need to be mentioned. First is the pause menu, which is just a simple transparent background that pauses the game and allows the player to quit back to the main menu. The final interface of the game is the HUD (heads up display). In Book Thief, the player can see whether they're sprinting, crouching or running based on a little icon placed on a corner of the screen. On a different corner they can see what key items they have collected, and how many remain to finish the game.

The design of the interface of Book Thief as a whole has been focused around simplicity. There's no need to add many UI elements to a game that doesn't need them and the few that are there take up as little space as possible. The idea was to keep the player focused on the gameplay, since it's the main selling point of the videogame.

CHAPTER 4

WORK DEVELOPMENT AND RESULTS

Contents

This chapter is a detailed timeline of the whole project, what work was done at what stages and dates and what things changed about the planning and the expected results during the development of the project. The final results of the project are also included in this chapter, to compare them to the objectives that were set on earlier stages of production.

4.1 Work Development

This section is a chronological recounting of all the events that happened during the work development. Any tasks, any milestones reached, any problems or other things that took place during the making of this project will be talked about here. Every important task will be divided into subsections.

4.1.1 Modeling Preparation

The project had begun a few months earlier, but since all that was done between then and now (late January) was writing the Technical Proposal there's no point in talking about it. On January 31st took place the first reunion with the supervisor, Michael Gould. Also on the meeting call was Juan Camilo, a member of the Geotec group who was to provide help with anything related to CityEngine. After the meeting was over, Juan Camilo sent all of the files and information necessary to start working with CityEngine. They've already been mentioned in an earlier chapter, but here's a list of everything:

- A geodatabase file containing all 2D information of UJI's campus.
- A zip file with the SmartCampus project for CityEngine.
- The most recent version of the university's architectural plans.
- Some miscellaneous information (plans, images, other files) about the library.
- A license for CityEngine and the installer.
- Tutorials for CityEngine and the SDK for ESRI products (including CityEngine).

The small task included in the Gantt Chart with the same name as this subsection entailed installing CityEngine, making sure it was working correctly, and check that all the files that Juan Camilo provided were correct and usable. This task only took a few hours, and on the same day it was possible to start working with CityEngine.

4.1.2 Learning CityEngine

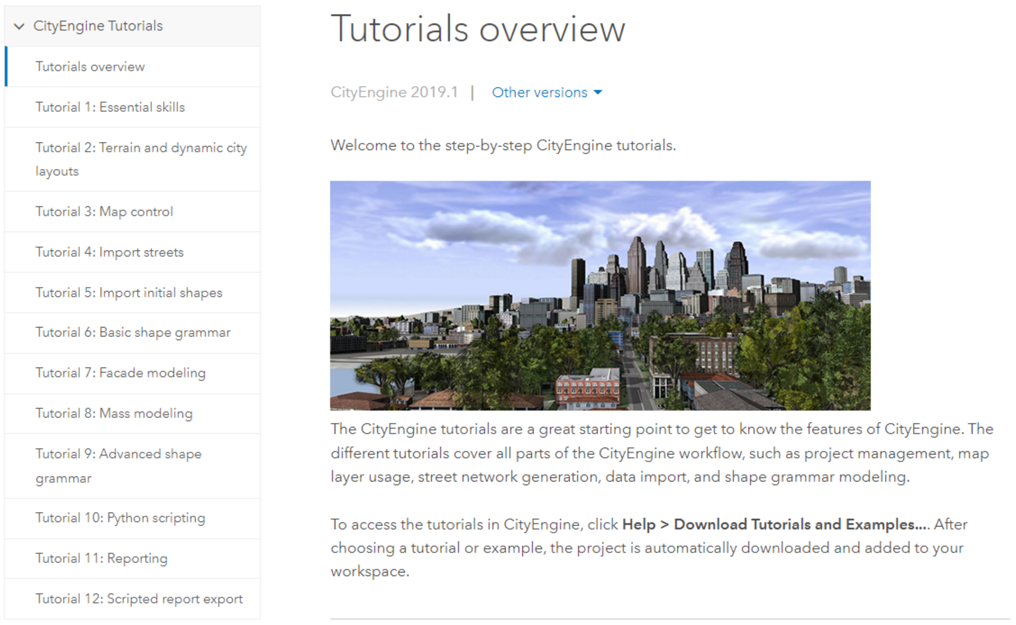
Once CityEngine was installed and with access to the project, it was decided to open the project to see exactly what the contents were. On the previously mentioned reunion we had established that the focus would be on the library interior, since according to them the exterior was pretty much done and didn't need an update. However, when the project was opened with CityEngine the reality was a bit different.

There were many other buildings and details in the project, but the main focus is only on the library. And there were quite a few issues with the exterior of the library. Here's a list of all of them:

- Textures clipping
- Faces out of place

- Faces missing
- Faces incorrectly shaped
- Textures not adequate
- Different shapes compared to the real library
- Flipped normals
- Incorrect measurements

Not all of them were noticed on the first time the project file was opened, but they were there and most of them were seen. They said that the project could be focused on modeling the interior of the library, but there was no way the exterior could be left on that state since it was needed for the videogame. But with a very clear lack of experience with CityEngine, it was decided to first follow the tutorials that Juan Camilo had kindly sent.


CityEngine Tutorials
Tutorials overview
Tutorial 1: Essential skills
Tutorial 2: Terrain and dynamic city layouts
Tutorial 3: Map control
Tutorial 4: Import streets
Tutorial 5: Import initial shapes
Tutorial 6: Basic shape grammar
Tutorial 7: Facade modeling
Tutorial 8: Mass modeling
Tutorial 9: Advanced shape grammar
Tutorial 10: Python scripting
Tutorial 11: Reporting
Tutorial 12: Scripted report export
Tutorials overview
CityEngine 2019.1 | Other versions

Welcome to the step-by-step CityEngine tutorials.

The CityEngine tutorials are a great starting point to get to know the features of CityEngine. The different tutorials cover all parts of the CityEngine workflow, such as project management, map layer usage, street network generation, data import, and shape grammar modeling.

To access the tutorials in CityEngine, click **Help > Download Tutorials and Examples....** After choosing a tutorial or example, the project is automatically downloaded and added to your workspace.

Figure 4.1: ArcGIS' tutorials

This task was entirely about completing all the tutorials (there were 20 of them) and getting as much experience as possible with CityEngine in the limited time there was. All the tutorials were finished (except a few that involved other software), and in the process much knowledge was gained of how CityEngine works and how to use it. Looking back now, most of the tutorials were not really useful to this project and they could've been skipped in order to save time and use that time to make the game better.

But there was no way back then to know what skills would be required to complete the library, and also the knowledge acquired from those tutorials might end up being very important for the future.

4.1.3 3D Modeling in CityEngine

This was the longest task of the whole project (it includes the next section as well, they have been split for easier reading), by quite a large margin. It was heavily underestimated how long it would take to model the library (partly due to the fact that it was thought that only the interior would need modeling work), and so the time spent building the library ended up being a decently big number.

Once all the tutorials were finished, it was decided to open the CityEngine project once more to take a more detailed look at the entire building. It was here where all the errors that weren't seen before were noticed, and where it was also checked something about the project and CityEngine. The entire interior of the library was empty (as expected), but there were quite a few problems if the objective was to model it from scratch. First of all, many faces from the exterior clipped into the interior or were wrongly placed and ended up inside anyway. Secondly, the blueprints for the interior of the library were not placed on the CityEngine project, so it would be needed to import them and somehow make them fit with the already existing exterior. Lastly (and arguably the most important factor), it was becoming more and more clear to me that CityEngine was not exactly made to model interiors. Navigating around the inside of the library was proving to be a more difficult task than was intended.

Seeing that even after completing all tutorials there wasn't really a clear idea of how to work, it was decided to once again contact the supervisor and ask him for advice. On the 28th of February we had the reunion. They provided a few tutorials on how to create interiors and said that there was another program called ArcGIS Pro that could maybe be used for the project. However, after careful consideration it could be seen that none of those would really help much, and so it was asked if they could somehow contact students who had worked with CityEngine on previous years, so that they could be asked some questions. They agreed, and on the same day after the meeting had passed they said they had managed to contact with 2 UJI ex-students that had worked on CityEngine.

A day after, on March 1st, there was a meeting with one of the two students, Lluc. His experience, knowledge and expertise with CityEngine and the whole project proved to be really helpful, and one could go as far to say all this would not have turned out as well without his help. He explained how the timeline of his own final degree work turned out, and gave some very important tips on how to make sure to meet all the deadlines without issues. He also showcased how to more efficiently fix the exterior of the library and also about SketchUp, a 3D modeling software that he had used in combination with CityEngine to build the interior. A few days later a different meeting took place with the other student, Francisco. His help was also invaluable, and said pretty much the same things that Lluc had explained and advised to start working as early as possible, since there was a lot of work to be done. With my newfound knowledge it was time to go back yet again to the CityEngine project, this time to actually start working. There were many errors to fix, and the main objective was to have the exterior have a proper look for when it was imported into Unity. The workflow that was used to work on the exterior was the following:

1. Pick a new section of the exterior to work on.
2. Check that section on Google Maps to have a point of reference.
3. Delete any unnecessary faces, from bottom to top.
4. Extend any faces that aren't the proper size, same order.
5. Create any missing faces, same order.
6. Apply/reapply textures to any faces that need them, same order.
7. Move around the section and check everything is correct and there are no errors.
8. Start again from step 1 until no sections remain.

This went on for quite a while, until the exterior of the library was finally free of errors (most of them, at the very least) and it looked nice enough to be put into the videogame on Unity. The next few pages contain images of both the finished library and some comparisons between the initial state and the final result.

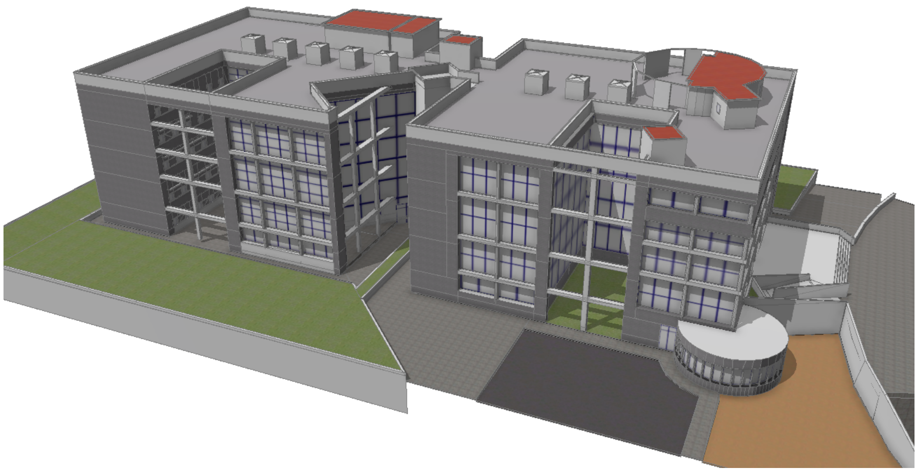


Figure 4.2: Final version of the library's exterior

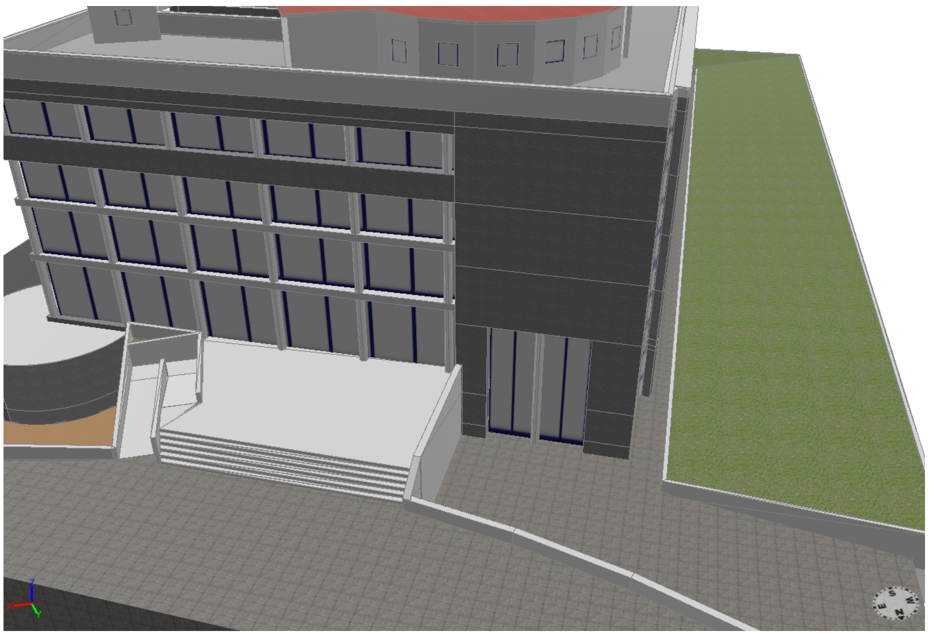


Figure 4.3: Frontal angle of the library's exterior

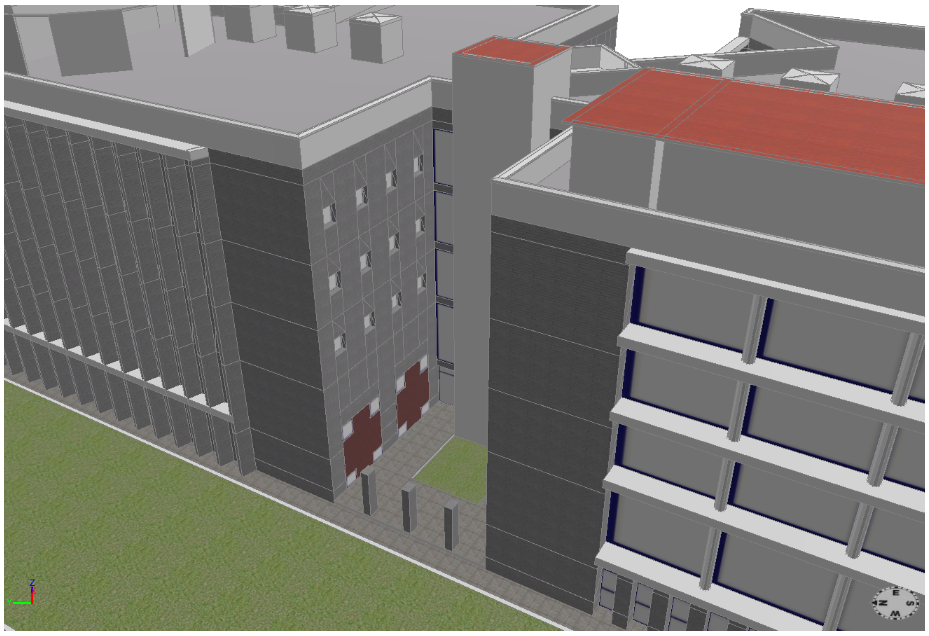


Figure 4.4: Side of the library's exterior

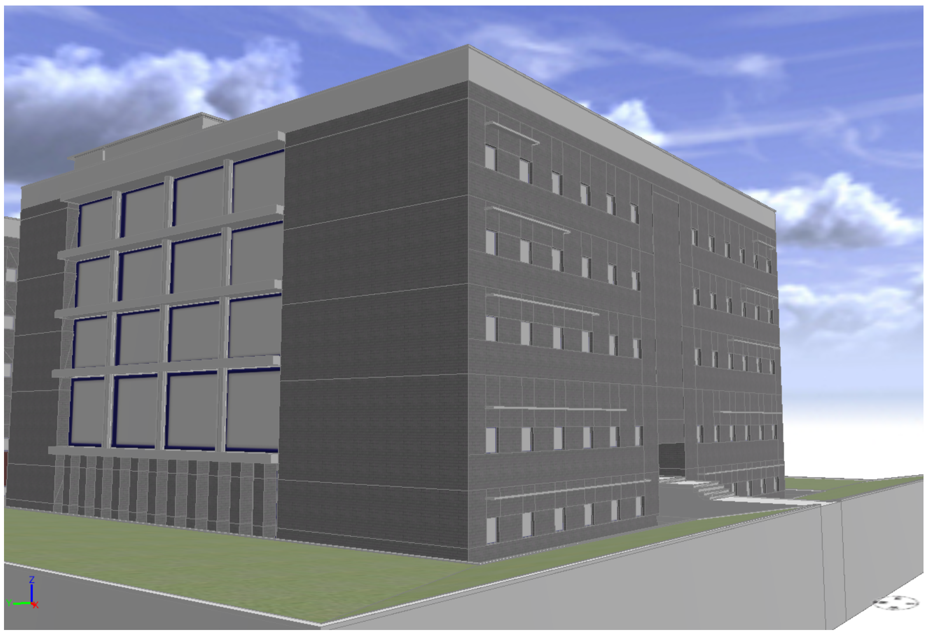


Figure 4.5: Back of the library's exterior

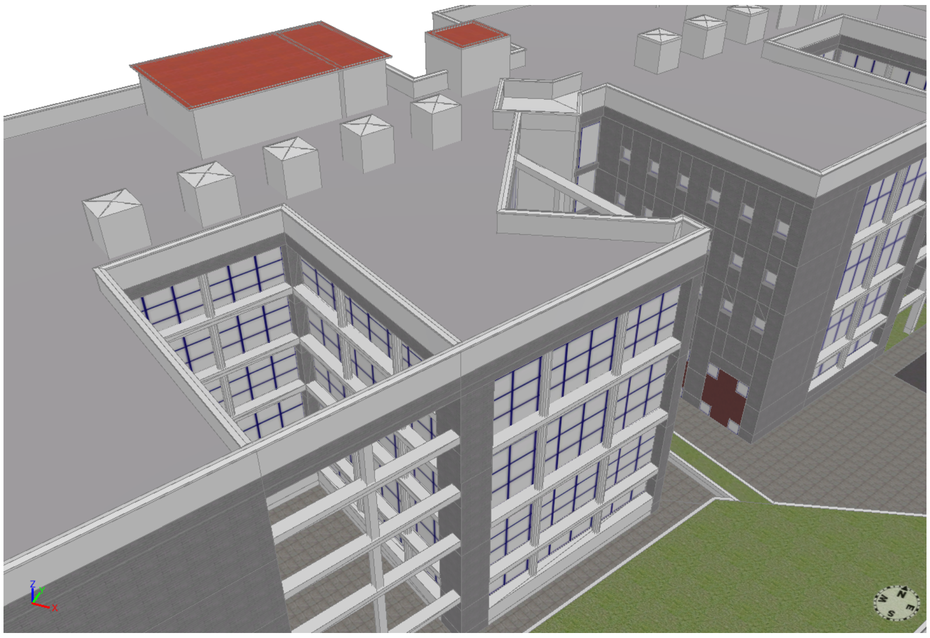


Figure 4.6: Top view of the library's exterior

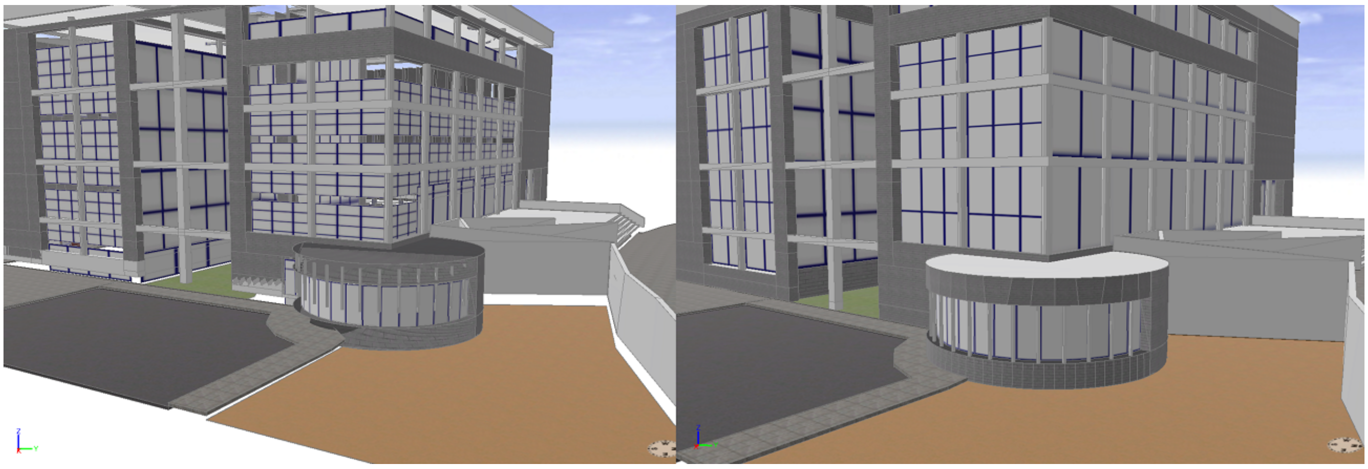


Figure 4.7: Before and after fixing 1

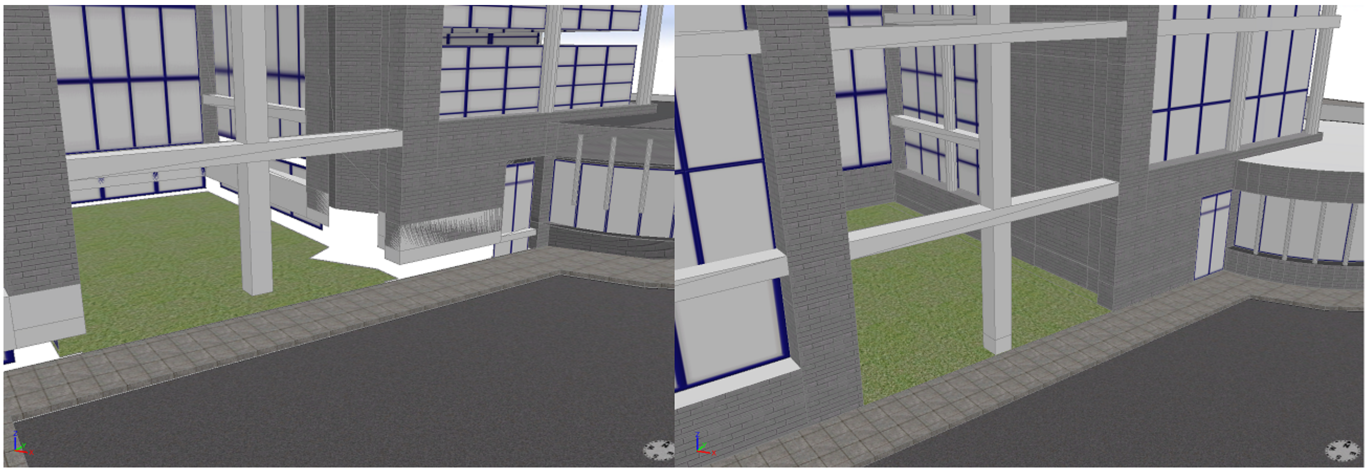


Figure 4.8: Before and after fixing 2

4.1.4 3D Modeling in SketchUp

This subsection is part of the same task as the previous one, "3D Modeling" on the Gantt Chart. With the exterior of the library finally finished, the interior was immediately worked on after realizing how much time it had taken to fix the outside of the building. A few tutorials [19] about SketchUp were watched, because of a lack of experience with the program, and there was no way it would be able to be used without some basic knowledge. After learning about the software the architectural plans of both the first and second floor of the library were imported into SketchUp and started being working on.

Only the first and second floor of the library were going to be modeled for a few reasons.

The main issue was time, and with more time it would've been possible to model all the floors without problem. So only the first and second floor were chosen, because that fit the deadlines properly (according to the planning).
It also made the most sense to model those two particular floors for the videogame. The underground floor could've been done instead of the second floor, but the underground probably didn't have enough furniture or sections.

With the two plans imported into SketchUp and scaled to the proper measurements, the process of modeling the entire interior finally started. Here's the workflow that was followed:

1. Pick a section of the interior.
2. Make all the faces necessary by tracing along the lines.
3. Extrude all the faces to the correct height.
4. Remove any faces that aren't necessary.
5. Reverse the normals of any face that requires it.
6. Repeat step 1 until all sections of the floor are done.
7. Create the floor face.
8. Add texture to all the faces in the floor.
9. Repeat the process for the second floor.

That workflow was followed until finally both floors of the library's interior were finished. The tools used were mostly very basic, only needing the pencil and the extrude function of SketchUp to model everything. Though some other tools were also sparingly used, like a plugin which created some of the faces of the plan, therefore making the job a little easier. Also used was another plugin that allowed to close all edge gaps, since the architectural plans of the library had a few problems. Much like the exterior on CityEngine, the plans also had a few errors (at least when imported to SketchUp).

Some of the lines on the plans were not connected to other lines when they should've been, causing faces not to form when tracing over existing lines. This caused quite the amount of issues, though part of the fault is mine for my inexperience with SketchUp. Seeing as some of the faces would not form when tracing over (and not yet knowing what caused it), the rectangle tool was used to form those faces. But the rectangle tool didn't make the face from those lines, it created four new lines on top of the existing ones and made a face with them. This caused some issues when extruding, causing some faces to be incorrect by a few pixels and also when adding textures and creating the floor face, since some faces would be duplicated or disconnected from others causing texture issues and faces not forming.

Fortunately, those errors were spotted not too late and in the end it was managed to complete the interior of the first two floors of the library. In the next few pages there's some images, to showcase both of the floors' final result and how they looked at different stages of the project.

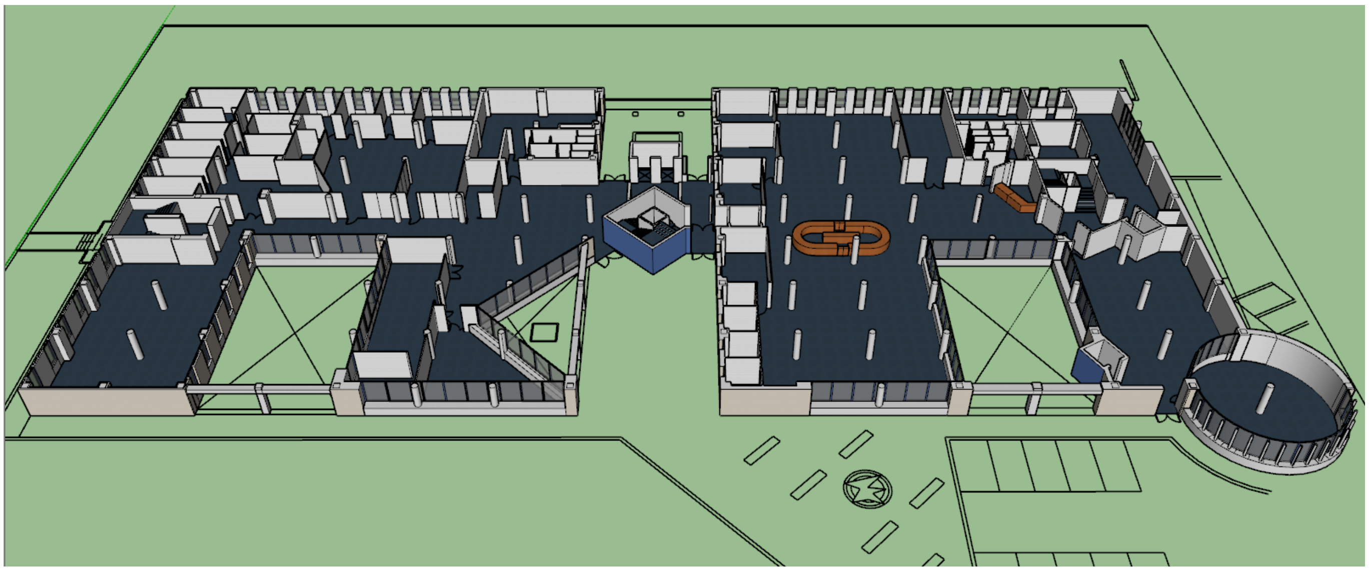


Figure 4.9: Final version of the library's floor 0

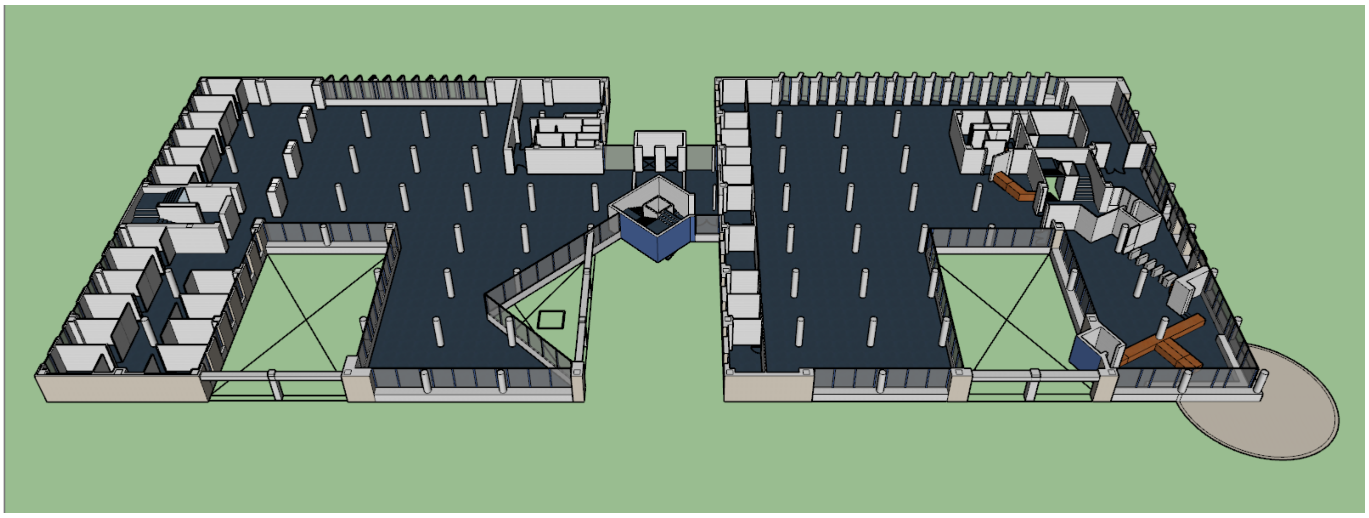


Figure 4.10: Final version of the library's floor 1

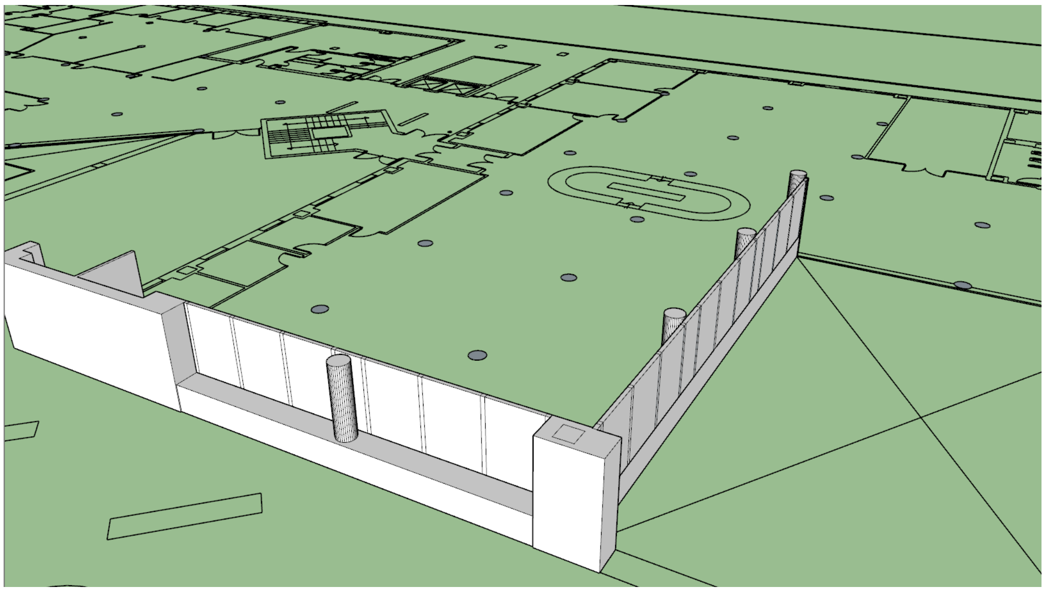


Figure 4.11: Starting Modeling

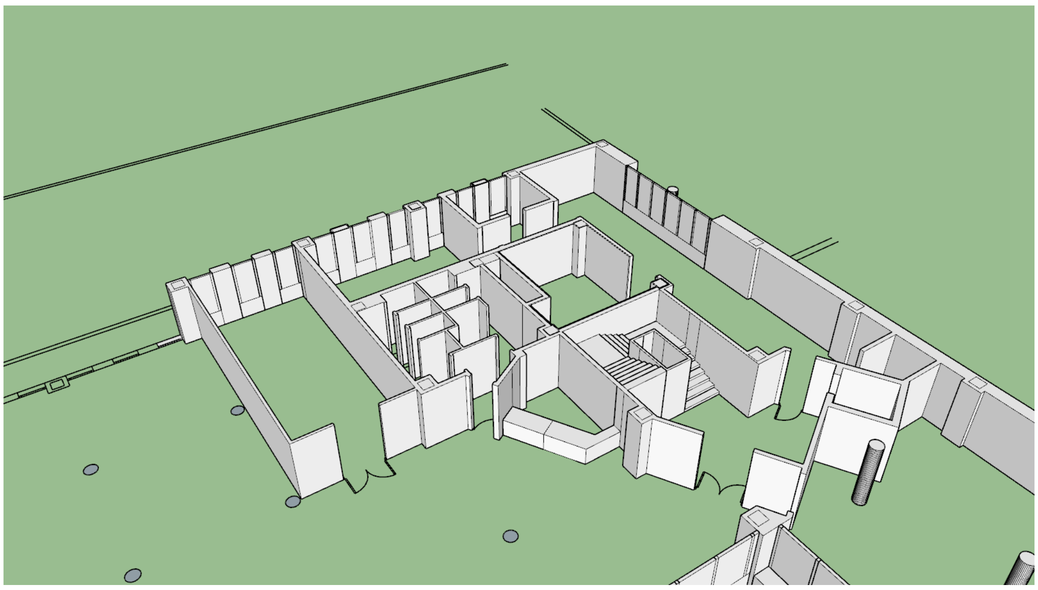


Figure 4.12: Advanced Modeling

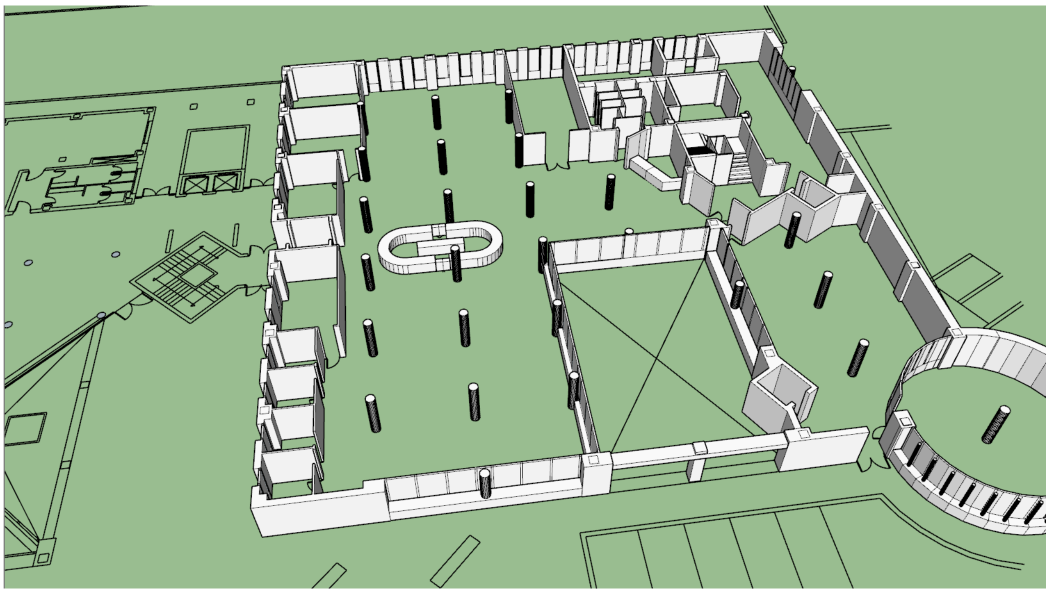


Figure 4.13: First half done

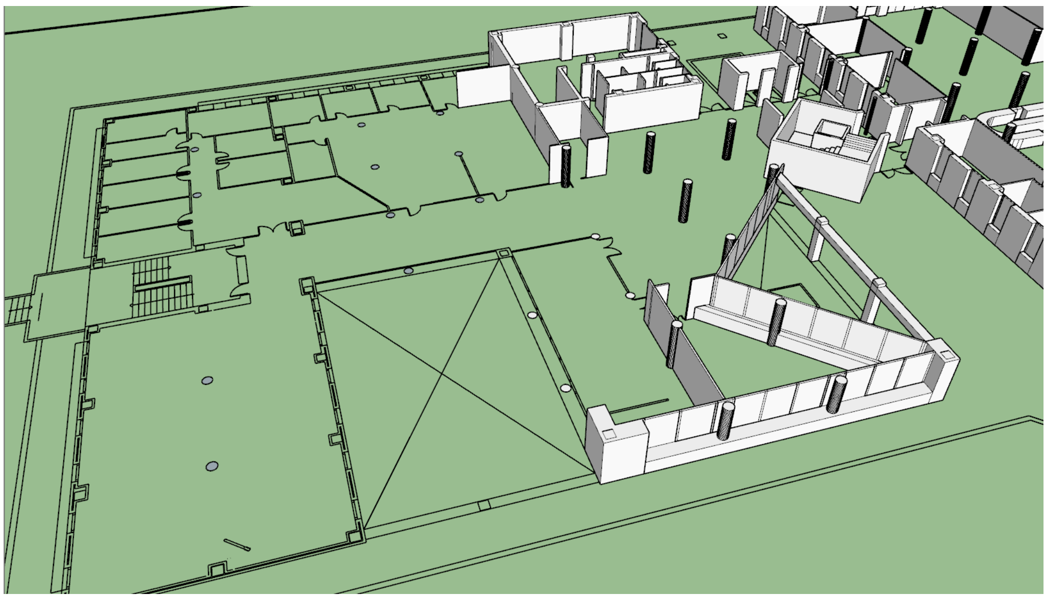


Figure 4.14: Working on second half

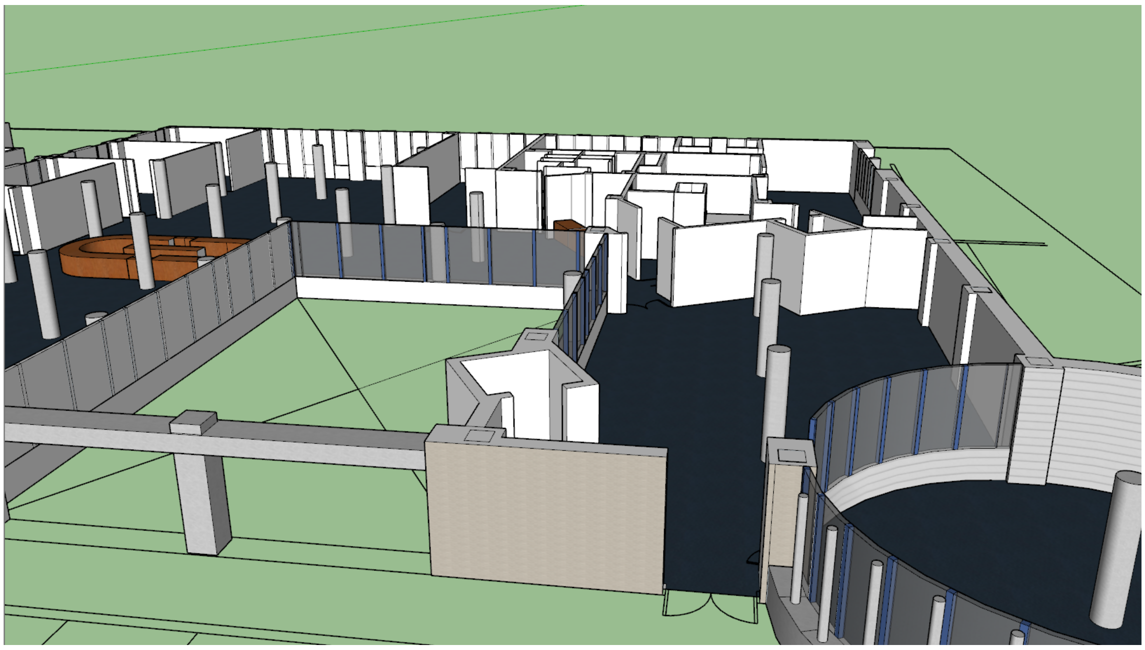


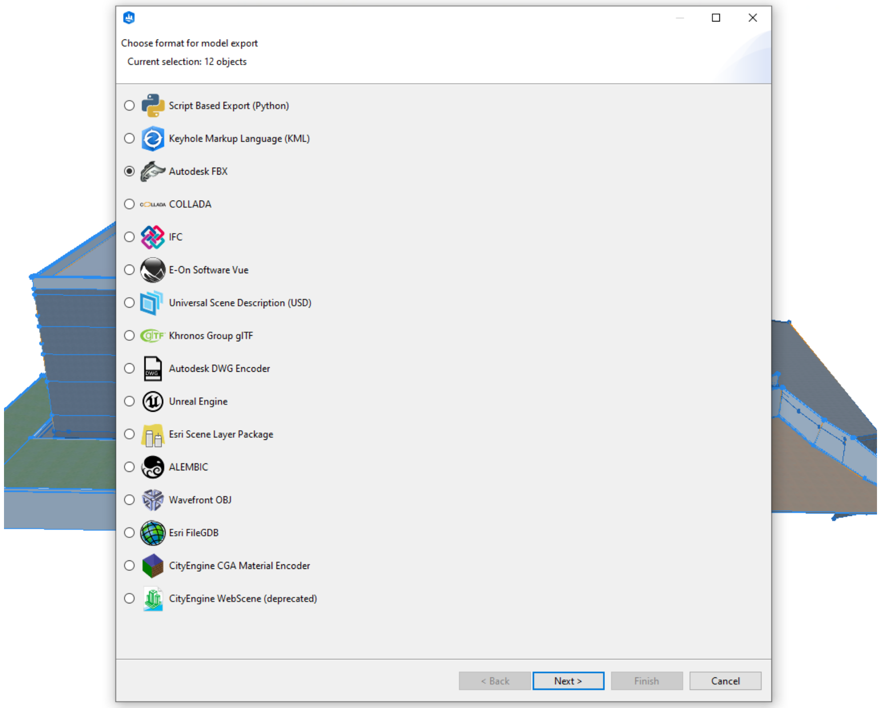
Figure 4.15: Painting shapes

4.1.5 Importing the library into Unity

With the interior of the library completely built on SketchUp and the exterior of the library fixed on the CityEngine project, it was time to import them both to Unity so they could be used on the videogame. Importing the interior of the library from SketchUp was relatively easy. By using the "export" button on SketchUp itself the program converted the scene into a COLLADA file, which then was imported into Unity where it turned into a prefab that could be dragged into the scene. Very simple, no issues encountered.

The exterior of the library on the other hand, caused a few more problems. It was known before importing to Unity that the exterior had a bunch of flipped normals, but the plan was to either duplicate the model on Blender with flipped normals and stack both of them on top of each other; or change the rendering shader of Unity so it was two-faced. Both of those solutions were not the most optimal, but luckily neither of those ended up being used. After importing the model from CityEngine to Blender and trying out the duplicate method, it could be seen that there were many clipping issues and also that illumination was wrong on some faces. So it was decided that even though it would take a while to manually reverse every flipped plane, it was the most sensible solution. The process of flipping normals on Blender went on until it was realized that CityEngine itself had that function, so the plan was to go back to CityEngine and fixed the model yet again. While fixing planes there it was noticed there were left a few mistakes and holes on the model, so those were patched up too. It sounds simple but it's worth noting that CityEngine doesn't exactly show you which way the plane is facing, so it was required to open both Unity and CityEngine to see on Unity what planes were flipped and find and fix them on CityEngine.

With the exterior finally on perfect condition to be put into the videogame, the model was exported directly from CityEngine. Then converted the 3D model into a .fbx file following some instructions from the official ESRI web page and imported that file into Unity. Finally, the entire library was inside the Unity project and it was decided to push the project to the GitHub repository in case something went wrong later. There were also stored some more security copies, just to be extra safe.


Choose format for model export
Current selection: 12 objects

- Script Based Export (Python)
- Keyhole Markup Language (KML)
- [✓ Autodesk FBX]
- COLLADA
- IFC
- E-On Software Vue
- Universal Scene Description (USD)
- Khronos Group glTF
- Autodesk DWG Encoder
- Unreal Engine
- Esri Scene Layer Package
- ALEMBIC
- Wavefront OBJ
- Esri FileGDB
- CityEngine CGA Material Encoder
- CityEngine WebScene (deprecated)

Figure 4.16: Export from CityEngine

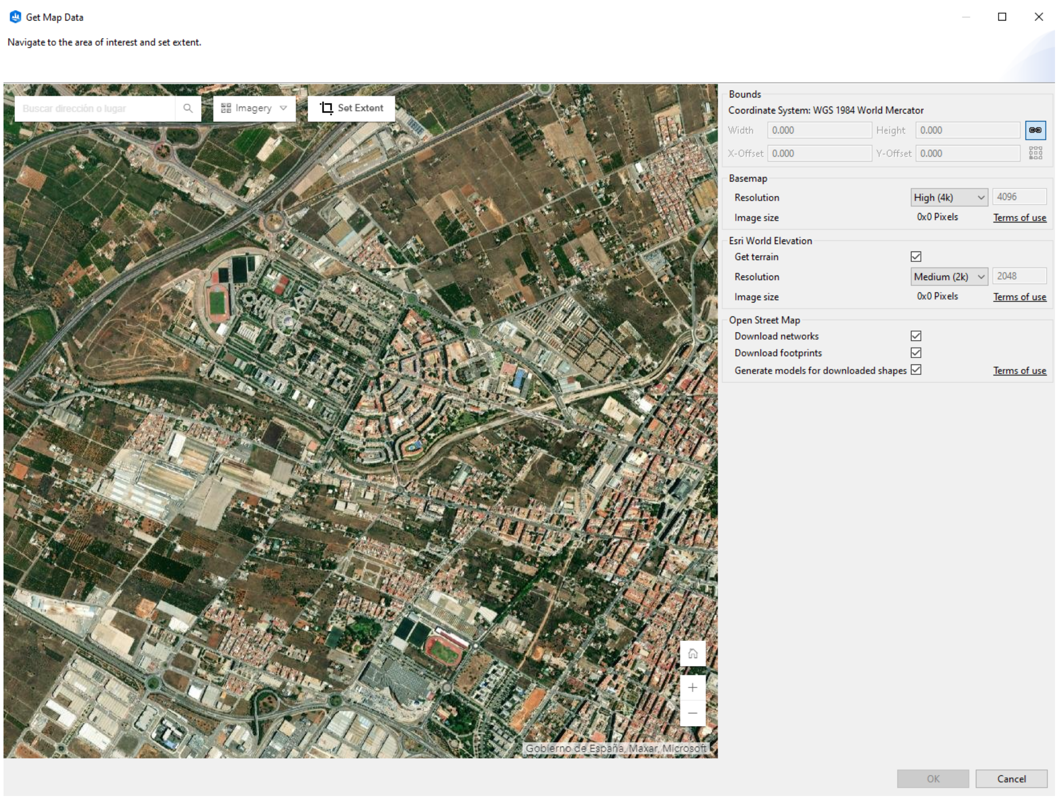
This process is included on the Gantt Chart as part of the "3D Modeling" task, since it made the most sense to throw it in there. Exporting and importing is also part of the modeling job and as such it fits with the name of the task.

4.1.6 Improving the scenery for the game

This subsection isn't exactly separated on the Gantt Chart, since there was no plans for it on the initial planning but it took so little time that it wasn't worth making a new category for it. This task however is accounted for on the "Developing the Game " task, since it took place during that period.

With the model of the library finally imported into Unity, it was becoming more and more clear that only having the building as an environment was never going to work. There was a need of something else to place the library on, so that the exterior didn't look completely empty. It was decided that CityEngine would be used for this task, since the main purpose of the program is to build cities and that was pretty much what was needed for the game. It would also help showcase on this project some of the more common functionalities of the software and how powerful it is as a tool.

First, this part started by trying to generate a city procedurally, with the tools that CityEngine has and with the knowledge I had gained from all the tutorials. However even if one of the tutorials from the official ArcGIS documentation which gives you a basic city as a result had been followed, it would've taken quite a bit of time and the end result wouldn't necessarily be the best, it would lack textures and other important aspects. So the plan was to find a more reasonable solution, and shortly after one was found [20].



Navigate to the area of interest and set extent.

Bounds
Coordinate System: WGS 1984 World Mercator
Width 0.000 Height 0.000
X-Offset 0.000 Y-Offset 0.000

Basemap
Resolution High (4k) 4096
Image size 0x0 Pixels Terms of use

Esri World Elevation
Get terrain ✓
Resolution Medium (2k) 2048
Image size 0x0 Pixels Terms of use

Open Street Map
Download networks ✓
Download footprints ✓
Generate models for downloaded shapes ✓ Terms of use

Figure 4.17: Map Data tool in CityEngine

There's an incredibly powerful tool in CityEngine called "Get Map Data". It is located on the "File" section of the program bar, and when you click on it a new window opens.

It shows a 2D map of the world, where you can select an area inside the map and change some settings, like the resolution of the base map and the elevation and if you want to also get the networks and the polygons. After selecting the area, choosing the settings and pressing on OK, a 3D environment of the entire area you selected will be placed on the scene.

This aspect of CityEngine is amazing and very useful, since it allows the user to build entire 3D models of cities and other places in a matter of seconds. It creates everything and lets you choose what to import, in case you don't want specific buildings or areas on your final 3D model. The only issue is that it relies on open data to build the environment, so depending on which area you pick the result might be better or worse, since every city has different levels of detail for their data.

So the plan was settled on using this function for the scenery. A random city with lots of tall buildings was picked (in this case, Chicago), selected an area of the city and the tool instantly placed a 3D model of the whole area on the scene. All that had to be done from there was export it to a .fbx file and then import it to Unity. Here's an image of the model on CityEngine:

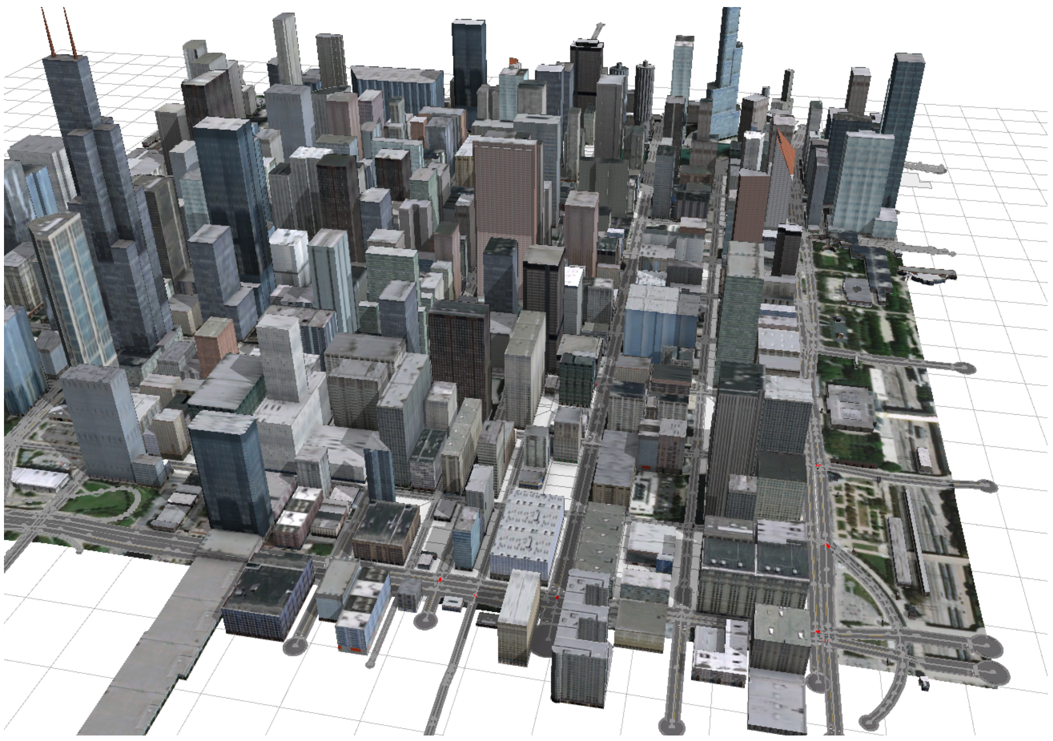


Figure 4.18: Model from map data in CityEngine

4.1.7 Developing the Game

With all of the modeling required for the project done and every object necessary imported into Unity (barring a few that would be added later), it was time to finally start developing Book Thief. Every aspect of the game was divided into several tasks and sorted them accordingly in order of their importance or when they are naturally done in most game studios. Here's the full list of tasks that were done to develop the game:

1. Create a scene for each part of the game.
2. Place all the imported items on their corresponding positions and scenes.
3. Import any needed models from the Unity Asset Store, like the main character and the enemy NPCs.
4. Place all the newly imported models on their positions.
5. Create any assets necessary for the main menu.
6. Build the main menu and code the functionalities of it.
7. Code the player movement, interactions, and others.
8. Code the enemy movement.
9. Tweak any other details necessary, like settings or lighting.
10. Revise the whole project to make sure everything is correct.
11. Test the game multiple times to find and fix any missing bugs.

Those are the steps that were followed in order to bring Book Thief to completion. The next subsections will elaborate on some of those tasks, to give a better understanding of the work that was done on every part of the development.

4.1.8 Dividing Scenes and Importing

Even though these four tasks sound simple enough, creating all scenes and placing all 3D models (both the buildings and other imported items) on them, it caused quite a few problems. For starters, the area where it was wanted to place the library wasn't completely flat, and so it was necessary to make some of the outside areas of the building taller. Without changing them, the library might seem like it was floating and the player would be able to see under the library. It was also really hard to place the library and the ground floor on the exact height where there would be no clipping but the model wouldn't be floating or separated from the ground. In the end it was managed to get a decent result, though it's not as perfect as it could be with better methods. With that the first two tasks were done.

The next two tasks of this subsection were importing all necessary content from the Unity Asset Store and place it on the proper scenes, much like the building before. This part gave much less troubles and went smoothly, for the most part. The player character was imported, the enemy NPCs, some furniture to place inside the library, some decoration to place outside of it and around the block, and other miscellaneous models. In the next pages are a few images of the game:

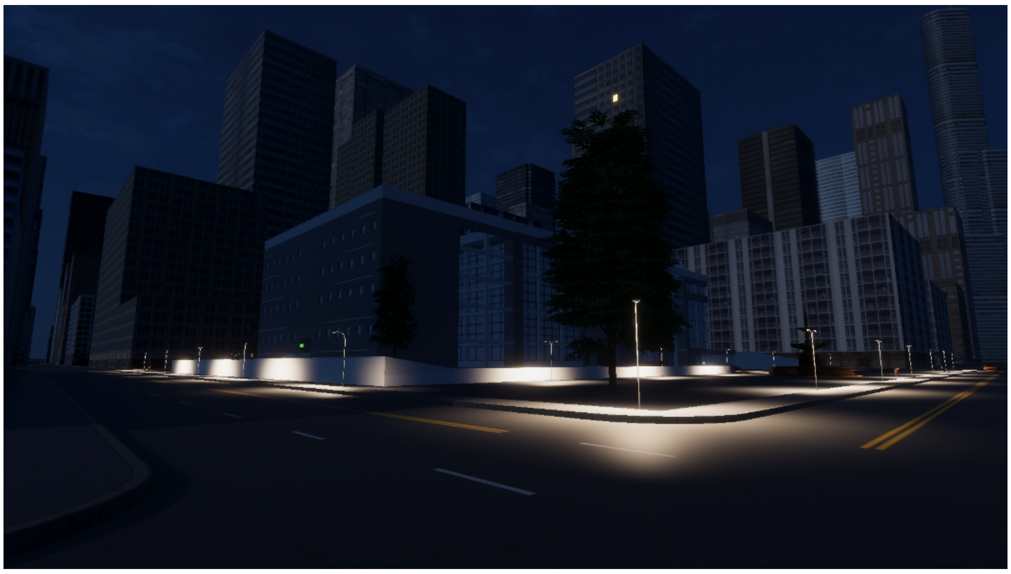


Figure 4.19: Exterior Scene 1

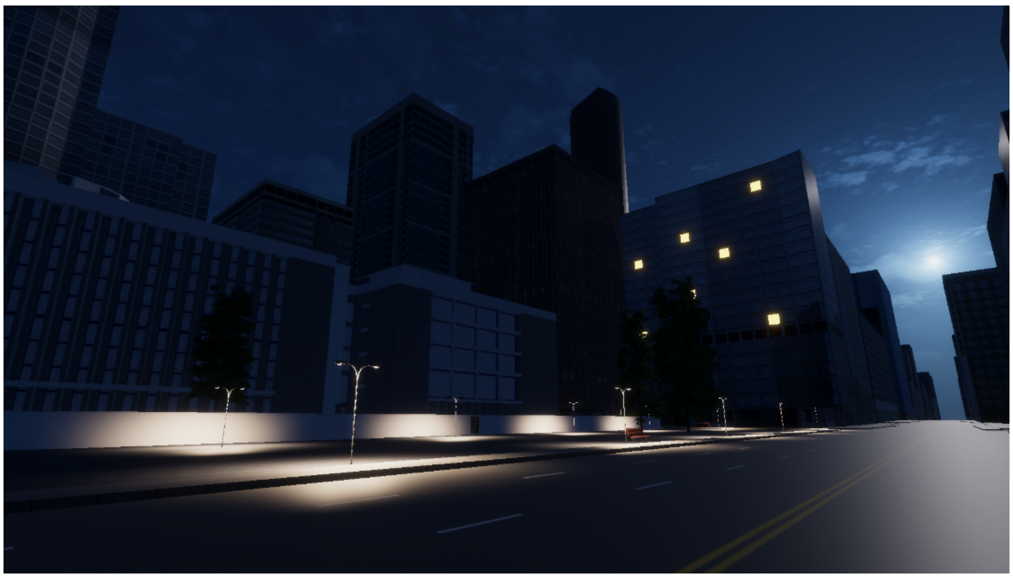


Figure 4.20: Exterior Scene 2

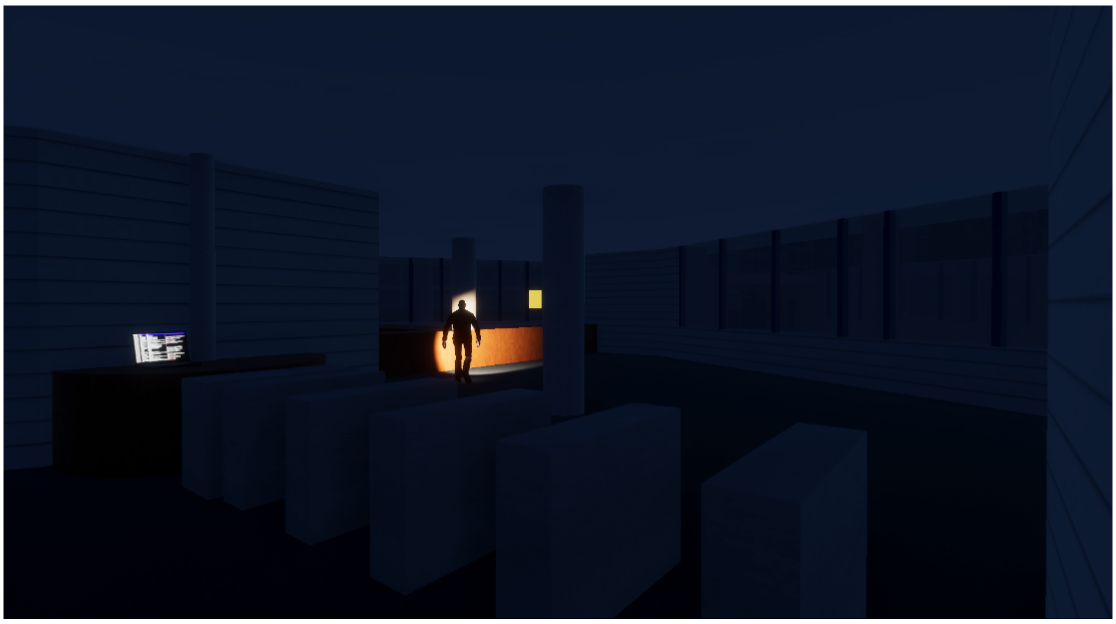


Figure 4.21: Interior Scene 1

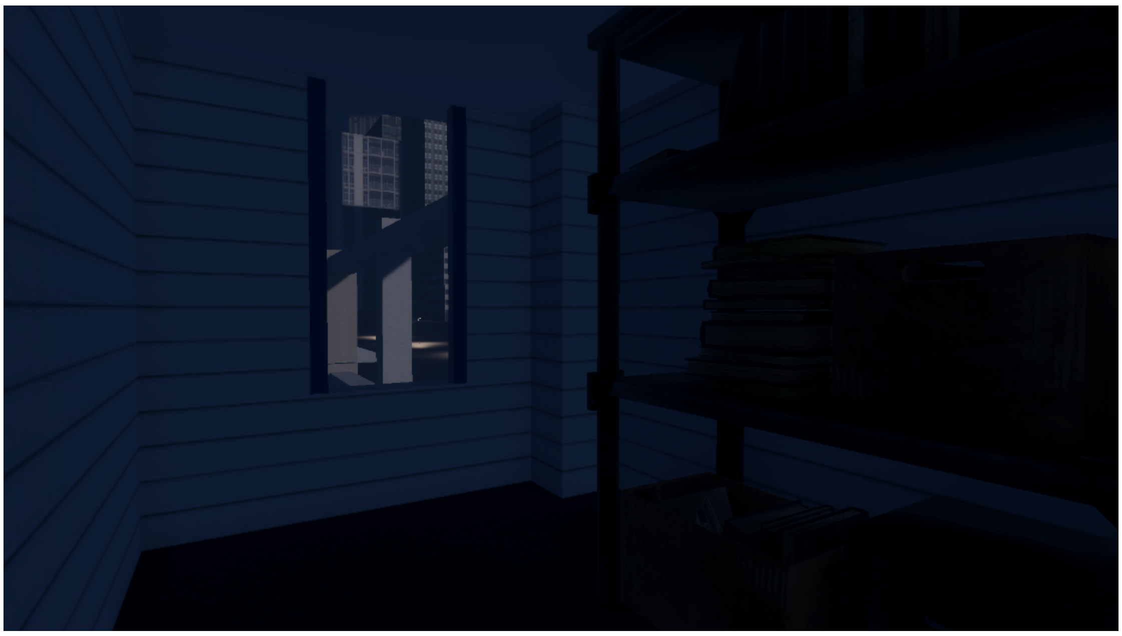


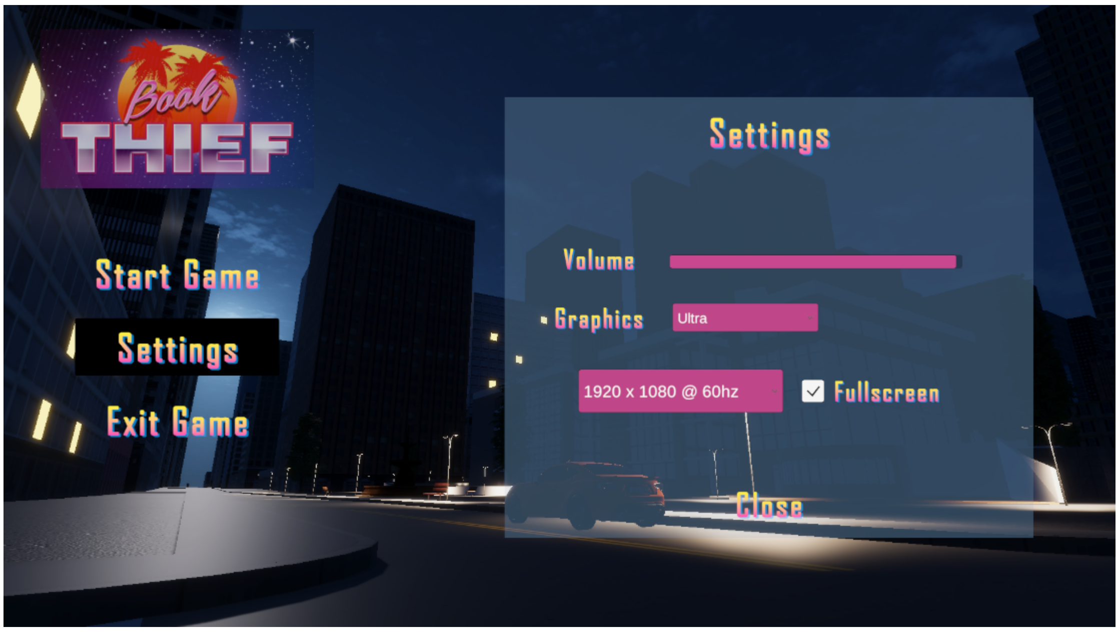
Figure 4.22: Interior Scene 2

4.1.9 Working on the Main Menu

With every piece on their place it was time to start actually designing and developing. The first scene that was worked on was the main menu. It made the most sense to start there since it's the first thing the player sees when loading into the game. The main objective here was to make the scene as appealing as possible, since the main menu is pretty much like the cover of a physical videogame.

The background of the scene is simply the exterior scene, from an angle that allows the player to see the whole area so that they're not lost once they load into the game. The title of the game is displayed on the top center of the screen, and below are the three buttons: Start, Settings and Quit Game. They are all self explanatory. There wasn't much coding to work on in this scene, just making sure the buttons worked properly and did their functions.

When opening the settings menu the player can change the quality of the game, make it full screen, change the resolution and lower the volume of the music and sounds. These options are the bare minimum a game should have in order to adapt to the player's needs.



Graphics: Ultra
1920 x 1080 @ 60hz
✓ Fullscreen

Figure 4.23: Settings Menu

4.1.10 Coding player and enemy functions

With the main menu done it was time to work on what was pretty much the backbone of the game, the scripts. Since there's quite a bit of them and there's no point talking about extremely technical details here, there's only a list the functionalities that were coded and in what order they were worked on. Here's the list:

1. The player's movement.
2. The player's ability to jump.
3. The player's ability to crouch.
4. The player's ability to sprint.
5. The player's ability to mark enemies.
6. The player's ability to move between scenes and floors.
7. The player's ability to interact with objects.
8. The enemy's patrol movement.
9. The enemy's pursuit movement.
10. The enemy's line of sight and hearing radius.
11. The inventory system (includes also designing it).
12. The system that controls the exit and finishes the game.

Though the list is quite large there's not that much code, and most of it is pretty basic. It won't be explained here as stated before, but there's the GitHub repository that contains every script and they can be checked there.

4.1.11 Tweaking Details and Improving Graphics

This task is a bit more diverse than the others and involves many different aspects of the game. The first thing done here was add some more models to the scene, since it needed them very badly. Those were furniture all inside the library so the player could hide behind them, and decorations and others around the outside of the library. Without them, the scenery of the exterior scene just looked way too barren.

One more detail that was worked on was the sounds and the music. The game was way too empty and dead without those, so a few were added to both scenes. Some very quiet ambient music to liven up the action, and a few sounds like footsteps and streetlight noises were added to give a bit more realism to the game.

Now that the scenery of both scenes was looking a bit better, it was time to actually improve the graphics of the game. And though the models are pretty much what they are and can't be changed, another big aspect of graphics that plays a big role on how good a game looks is lighting. There's a Unity package called "High-Definition Render Pipeline" (HDRP), which is pretty much just an upgrade to the already existing Universal Render Pipeline (URP). By adding the package to the project placing a light and an environment object to the scene, you can start working with HDRP. With those two objects placed you can tweak their settings and parameters to change the lighting to whatever you want it to be.

A tutorial from the official Unity YouTube channel [21] was followed, but here's a brief summary. The two previously mentioned objects were added, changed and assigned some parameters given by Unity themselves to make the ambient light nighttime, added a bit of fog to better appreciate the environment and reflections, and added some street lights around the exterior of the library (if they weren't there it might have been too dark for the player to see).

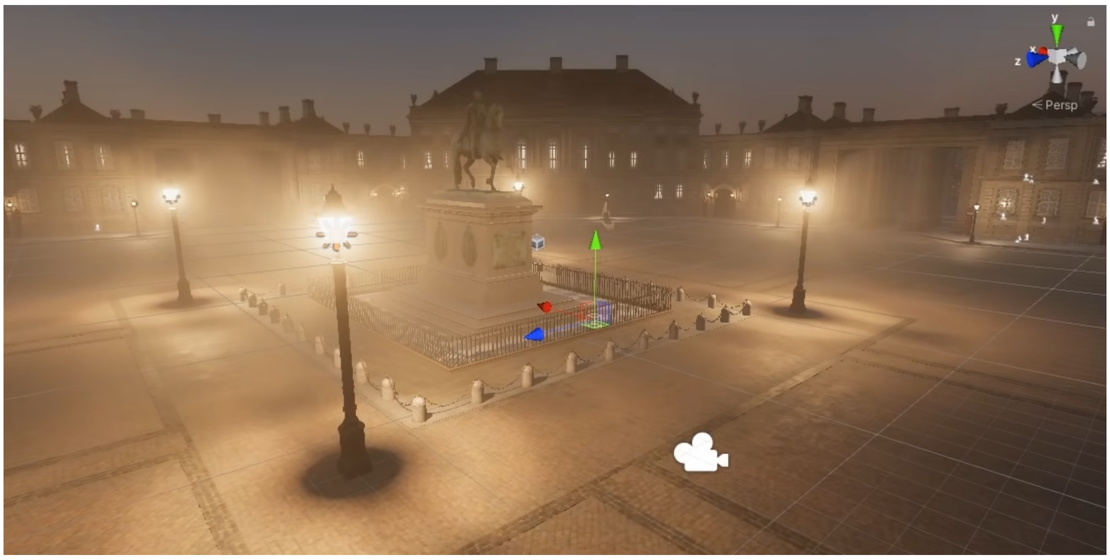


Figure 4.24: HDRP Tutorial Showcase

4.1.12 Revising and Testing

Though this task maybe isn't really complicated and some might think it's not worth to talk about, the importance of testing your game once it's finished cannot be understated. No single videogame is free of bugs, and it's up to the developer to find them and fix them. Without properly testing a game, it's impossible to find bugs and errors that might remain there.

A pretty thorough testing of the game was done, playing it multiple times while testing that every script, collision, model and others were working correctly. Some bugs might remain (since it's almost impossible to remove every single bug from a game), but the player shouldn't notice any of them during a normal playthrough.

And with that the work development section is done. There was a bit more work after completing the game development, such as some last minute adjustments and also the final presentation and document; but those won't be mentioned here.

4.2 Results

There were three main objectives mentioned on chapter 1 section 2 of this document. Those were to firstly, create a 3D model of the library so that it can replace the one that is currently being used on the Smart UJI project. Second, to develop a first person stealth game in Unity that showcases how much experience as a programmer has been gained these past four years. And third and final, to become proficient at CityEngine and reach a point where one would be satisfied with the knowledge of the software.

For the first objective, I would say the result is satisfactory. Even though it might not be the most accurate compared to reality, I would say that for the purpose of being used in the Smart UJI project it works well enough, especially compared to the model that is currently inside the web viewer. I would've liked to somehow combine the interior and exterior in one single file, but building the game didn't require it and doing that task on CityEngine would've been very painful, especially if the measurements on both sides differed even by a pixel. I would've also preferred it if I had more time to model more floors of the library. I only managed to do the first and second floors since I also spent time fixing the exterior.

For the second objective, I would also say that I'm happy with the result. The game might not be the most technically advanced of the prettiest, but I used the tools at my disposal and managed to create a pretty decent environment with CityEngine inside Unity. The gameplay isn't revolutionary but it accomplishes its purpose well and allows the players to explore the library while also adding some difficulty and skill involvement.

For the third and final objective, the result is a bit more subjective, since it depends on if I feel like I've become proficient at CityEngine. And to be honest, I don't really think so. I've been able to use the tools necessary to fix the exterior and I've learned about many other functions of CityEngine while doing the tutorials, but I personally don't think I can call myself proficient in CityEngine. It's a very powerful software with an incredible amount of functions, tools and capabilities at the user's disposal, and I personally think it would take way more time than I've spent to become an expert at CityEngine. So for this result, even though I've managed to learn quite a bit about the program and used it to complete the other 2 results, I'm not gonna consider myself proficient with CityEngine.

Chapter 5
Conclusions and Future Work

Contents

This final chapter talks about the conclusions of the work, as well as its future extensions and other ways to keep working on it.

5.1 Conclusions

This section will mainly talk about my personal opinion about my final degree work. I think the development went rather smoothly and I didn't encounter many issues that I couldn't solve with enough time. But I will say, the only reason this went as well as it did was because of two factors: first, because I started the project very early this year; and second because of the counseling that the ex-students gave me.

If I hadn't started working on the whole project so early the results might have been very different. Though the university says that the recommended hours this work should take are around 300, I can confidently say that I've worked way more than that. And that's considering I didn't encounter many problems, thanks to the meetings with both Lluc and Francisco.

There were many ways I could have approached this whole project, considering the data that was given to me. I could've started the exterior from scratch, either from the geodatabase files or from the architectural plans; I could've tried using many other programs to model the interior, I also had the choice to maybe using ArcGIS Pro to help me with the exterior, etc. But I avoided all those (and by doing so escaped losing many hours without results) thanks to knowing what problems and difficulties the previous students had when working on similar projects. Had I tried any of those ways mentioned before and failed, due to my own inexperience or to issues with the softwares involved; I honestly don't think I would've been able to finish this project in time for the deadline.

My final stance on this whole project is that overall, it wasn't that bad. I learned a very useful program to procedurally generate cities (CityEngine), improved my lacking 3D modeling skills, and refined my programming and design by working on the videogame. I'm also happy with the final results (the library and Book Thief, the game) and looking back, I wouldn't change any of my choices that lead me here.

5.2 Future work

The work done in this project can be continued in many ways, either through continuing the modeling process for the rest of the floors of the library (and even the rest of the building inside the Smart UJI project) or through improving, upgrading and expanding the videogame. Any number of things could be added to the videogame: a proper story, better gameplay, voice acting, more enemies and areas, better player progression, allowing the player to pick difficulty, etc.

Another way the project could continue being worked on (and this is something I recommend to future students working on similar projects), is to redo the entire library, this time building it entirely on a dedicated 3D modeling software like SketchUp and making sure that all measurements and shapes are correct. By building both the interior and exterior on the same program it would save much time and also avoid issues. Though I consider CityEngine an amazing software I personally don't think it should be used to do detailed 3D modeling. It is not the program's main function, and it shows. Had I had more time (and the option) I would've probably taken this route myself for this project. However since time was limited, I chose to focus more on the videogame, since that's my area of expertise anyway.

Bibliography

[1] Geotec, "Official Geotec website." http://geotec.uji.es/, 2022. [Online; accessed 13-January-2022].

[2] Geotec, "Smart UJI Project." http://smart.uji.es/, 2022. [Online; accessed 13-January-2022].

[3] ArcGIS, "CityEngine's official website." https://www.esri.com/es-es/arcgis/products/arcgis-cityengine/overview, 2022. [Online; accessed 13-January-2022].

[4] ArcGIS, "ArcGIS' CityEngine tutorials." https://doc.arcgis.com/en/cityengine/2019.1/tutorials/introduction-to-the-cityengine-tutorials.htm, 2022. [Online; accessed 13-January-2022].

[5] Trimble, "SketchUp's official website." https://www.sketchup.com/es, 2022. [Online; accessed 26-January-2022].

[6] U. Technologies, "Unity's official website." https://unity.com/es, 2022. [Online; accessed 1-January-2022].

[7] Google, "Google Drive's official website." https://www.google.com/intl/es_es/drive/, 2022. [Online; accessed 1-January-2022].

[8] GitHub, "GitHub's official website." https://github.com/, 2022. [Online; accessed 1-January-2022].

[9] B. Foundation, "Blender's official website." https://www.blender.org/, 2022. [Online; accessed 1-January-2022].

[10] Microsoft, "Visual Studio's official website." https://visualstudio.microsoft.com/es/, 2022. [Online; accessed 1-January-2022].

[11] Google, "Google Meet's official website." https://meet.google.com/, 2022. [Online; accessed 1-January-2022].

[12] Google, "Google Maps's official website." https://www.google.es/maps/?hl=es, 2022. [Online; accessed 1-January-2022].

[13] WriteLatex, "Overleaf's official website." https://es.overleaf.com/, 2022. [Online; accessed 15-April-2022].

[14] Discord, “Discord’s official website.” https://discord.com/, 2022. [Online; accessed 1-January-2022].

[15] Valve, “PAYDAY 2’s official Steam page.” https://store.steampowered.com/app/218620/PAYDAY_2/, 2022. [Online; accessed 1-January-2022].

[16] Unity, “Unity Asset Store website.” https://assetstore.unity.com/, 2022. [Online; accessed 1-January-2022].

[17] OpenGameArt, “Official website.” https://opengameart.org/content/library-of-game-sounds, 2022. [Online; accessed 1-January-2022].

[18] Unity, “Site with the Unity requirements.” https://unity3d.com/es/unity/whats-new/2020.3.13, 2022. [Online; accessed 25-April-2022].

[19] TheSketchUpEssentials, “Tutorial about SketchUp.” https://www.youtube.com/watch?v=c2HDmKFhGWM, 2022. [Online; accessed 1-March-2022].

[20] E. Events, “Video showcase of the Get Map Data function.” https://www.youtube.com/watch?v=sO0iDzfsFU4, 2022. [Online; accessed 24-April-2022].

[21] Unity, “Video showcase of HDRP in Unity.” https://www.youtube.com/watch?v=OXCB-LKCK_Y, 2022. [Online; accessed 1-May-2022].