Design and development of a low poly puzzle game

Jorge Bartol Guillamón

Final Degree Work

Bachelor's Degree in
Video Game Design and Development

Universitat Jaume I

June 30, 2022

Supervised by: Carlos Marín Lora

Acknowledgments

First of all, I would like to thank my Final Degree Work supervisor, Carlos Marín Lora, for all the help and guidance during the whole development of the project

I also would like 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 presents the project report of the Video Game Design and Development Degree Final project by Jorge Bartol Guillamón. It is a video game that consist of a series of puzzles separated into small scenarios which the player must solve in order to progress through the game.

To complete them, the player will control a small character who will interact with all the elements on the stage. Each of this levels will have a different set of mechanics to solve the puzzles.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Contents

This chapter is an explanation about which were the motivations that took to project's idea, which were the objectives initially fixed and how the idea started to be developed

1.1 Work Motivation

The aim of the project is to create a complete video game in which all the knowledge learned throughout the degree is applied, both at programming and artistic level.

One of the game genres that has always caught my attention has been puzzle games, which challenge the player and can be combined with almost any other type of game such as a shooter or an adventure game.

For this reason i decided to create a video game focused on solving puzzles which are separated into different rooms. Being divided into individual levels makes it more attractive and less time-consuming to solve the puzzles.

1.2 Objectives

- To create a project in which all the knowledge that has been learnt throughout the degree has been applied, resulting in a complete and playable video game.

- Modelling and designing all the elements that appear within the game and programming the scripts to make the game logic work.
- Create several levels with a series of puzzles that are attractive to players and entertaining to play, with mechanics that are correctly applied to the game play and environment design.

1.3 Environment and Initial State

The project is based on the idea of developing a game in which the player must solve the different obstacles that are presented throughout the game in order to overcome the different levels.

For this purpose, a puzzle game will be developed, separated by levels, seen in isometric perspective and low poly aesthetics. Each level will have a series of elements with which the player will have to interact in order to complete it and advance to the next one, increasing the difficulty progressively.

The work will be programmed entirely in Unity, and all the artistic section of the game will be modeled on Blender.

First I will start programming the main mechanics of the game and base models that are important in it, such as the movement of the character. This will be used to see if any adjustments or changes need to be made in order to continue with the project.

It will be developed over a period of three months while it is combined with the external internships of the degree that entail about five hours a day, so this work will have a daily dedication of approximately three hours.

CHAPTER 2

Planning and resources evaluation

Contents

This chapter shows the planning that has been followed to complete the project and the resources used to accomplish that purpose.

2.1 Planning

This section details the approximate time distribution of all the tasks that have been carried out for the development of the project (see figure 2.1).

This includes all the aspects worked on during the development, such as level design and modelling, or the programming required for the implementation of the game mechanics. Also, a part of the total of these hours is dedicated to the creation of all the necessary documents for the creation of the final degree project, including this one.

Table 2.1: Table of time distribution

Task	Time (Hours)
Design of puzzles, characters and levels	30
Modelling of scenarios and characters	60
Modelling of objects of the environment	40
Scenario implementation	2
Puzzle mechanics	40
Player movement	10
Visual effects	10
Material design	10
Lighting implementation	5
Animations	5
Interface	3
Main menu	7
Game Design Document	10
Memory	60
Project presentation	10
Total	302

2.2 Resource Evaluation

The resources used in the development of the project, both software and hardware, are as follows:

- **Desktop computer (procesador i5-7600k , 16GB de RAM and a MSI 1050 ti 4GB graphics card):** This will be the core team with which the majority of the project will be carried out.
- **Laptop (Acer Nitro 5):** Used to a lesser extent for when working away from home
- **Unity 3D version 2020.3.25f1:** The version of the game engine to be used for the project.
- **Visual Studio 2022:** Programming environment for game scripts.
- **GitHub (GitHub Desktop):** A Git repository where all the changes made during the project will be stored. Useful for not losing the changes made, avoiding conflicts in the code or sharing data between several computers.
- **Blender version 3.1:** Free 3D modelling program with which all the models that appear in the game will be made.
- **Krita:** Drawing program to design the images that will make up the game interface.

- **Mixamo:** Website for downloading and importing animations for 3D models within Unity.
- **Lucidchart:** Website to create diagrams.

CHAPTER 3

System Analysis and Design

Contents

This chapter presents the requirements analysis, design and architecture of the proposed work, as well as, the flowchart of the game, the references and its interface design.

3.1 Requirement Analysis

This project consists of the development of a puzzle video game within the Unity 3D game engine. Specifically, it is a third-person game in which you control a character in small scenarios where each one will be part of a different level, reaching a total of three, in each of them will be different types of puzzles to solve to complete the game.

To solve the puzzles you will have to use the player's own mechanics, which are walk, run and jump to move around the stage; and the mechanics of each puzzle, which are push and pull objects and use items.

As mentioned before, each level has its own puzzle and scenario. These will have a pre-established order, from less to more difficult, and in order to unlock them the player has to complete them following this order. Once unlocked, any of them can be accessed freely without any limitation.

To guide the player on what to do at all times, a tutorial level will first load and explain all the mechanics of the character. Once completed, it can be revisited in the level selector at any time.

At all times the player will have a pause menu that will allow him to restart the level, resume the game and exit to the main menu.

Regarding the artwork, all the elements that make up the scenery will have a low poly aesthetic.

3.1.1 Functional Requirements

Functional requirements define which features of the game are to be developed:

- **R1** The player can start the game.
- **R2** The player can select levels.
- **R3** The player can move the character in all directions around the stage by pressing the W,A,S and D keys.
- **R4** The player can make the character perform a jump by pressing Space.
- **R5** The player can make the character use objects by pressing the E key.
- **R6** The player can push and pull certain objects that appear on the map by pressing the B key and moving in one direction.
- **R7** The player can open the pause menu.
- **R8** The player can exit the game.
- **R9** The player can return to the main menu.
- **R10** The player can control the sound volume.

3.1.2 Non-Functional Requirements

Non-functional requirements will impose conditions on the design and implementation of the elements that appear in the game:

- **R11** The game will be playable on PC
- **R12** The game will use low poly models
- **R13** The mechanics will be easy to learn
- **R14** The interface will be simple, without obstructing any element of the game.
- **R15** The mechanics will be fluid, providing feedback to the player.

3.2 System Design

This section presents the logical and operational design of the system to be carried out. To design it, case of use diagrams will be used for each of the functional requirements presented above.

Requirement:	R1
Actor:	Player
Description:	At the main screen the player can push the button “Play” to start the game
Preconditions:	1 The player is on the main screen.
Normal sequence:	1 The player pushes the “Play” button. 2 The game loads new scene.
Alternative sequence:	None

Table 3.1: Case of use «R1. Play»

Requirement:	R2
Actor:	Player
Description:	At the scene selector the player can select a level.
Preconditions:	1 The player is on the scene selector.
Normal sequence:	1 The player pushes the scene icon. 2 The game loads the scene selected.
Alternative sequence:	None

Table 3.2: Case of use «R2. Select Level»

Requirement:	R3
Actor:	Player
Description:	The player can move the character using the keys W,A,S and D.
Preconditions:	1 A level has been loaded.
Normal sequence:	1 The player pushes one or two key of the movement. 2 The game moves the character in the direction specified. 3 The game rotated the character in the direction specified.
Alternative sequence:	None

Table 3.3: Case of use «R3. Movement»

Requirement:	R4
Actor:	Player
Description:	The player can make the character jump with the space key.
Preconditions:	1 A level has been loaded
	2 The lower part of the character is touching some object.
Normal sequence:	1 The player pushes the space
	2 The game moves the character upwards.
Alternative sequence:	None

Table 3.4: Case of use «R4. Jump»

Requirement:	R5
Actor:	Player
Description:	The player can use the available object.
Preconditions:	1 A level has been loaded
	2 The character is not jumping.
Normal sequence:	1 The player pushes the E key
	2 The character uses the object
Alternative sequence:	None

Table 3.5: Case of use «R5. Use Object»

Requirement:	R6
Actor:	Player
Description:	The player can push and pull some objects on the map.
Preconditions:	1 A level has been loaded
	2 The character is touching the floor.
Normal sequence:	1 The player pushes the B
	2 The player moves the character to one direction.
	3 The game moves the object on this direction
Alternative sequence:	None

Table 3.6: Case of use «R6. Push and Pull objects»

Requirement:	R7
Actor:	Player
Description:	The player can open the pause menu
Preconditions:	1 A level has been loaded
Normal sequence:	1 The player pushes the Escape key
Alternative sequence:	None

Table 3.7: Case of use «R7. Pause menu»

Requirement:	R8
Actor:	Player
Description:	The player can close the game
Preconditions:	1 The player is on the main scene
	2 The player has opened the pause menu
Normal sequence:	1 The player pushes the Quit button
Alternative sequence:	Closing the program from windows

Table 3.8: Case of use «R8. Close game»

Requirement:	R9
Actor:	Player
Description:	The player can go back to the main scene.
Preconditions:	1 A level has been loaded.
Normal sequence:	1 The player pushes the Escape key.
	2 The player presses the Main Menu button.
Alternative sequence:	None

Table 3.9: Case of use «R9. Main Scene»

Requirement:	R10
Actor:	Player
Description:	The player can regulate the volume.
Preconditions:	1 The player is on the main scene.
Normal sequence:	1 The player presses the Options button.
	2 The player moves the volume slider.
Alternative sequence:	None

Table 3.10: Case of use «R10. Change volume»

3.3 Flowchart

In order to identify which steps the player follows throughout the game, a flowchart will be created which describes all the processes from the start of the game until the completion of a selected level (see figure 3.1).

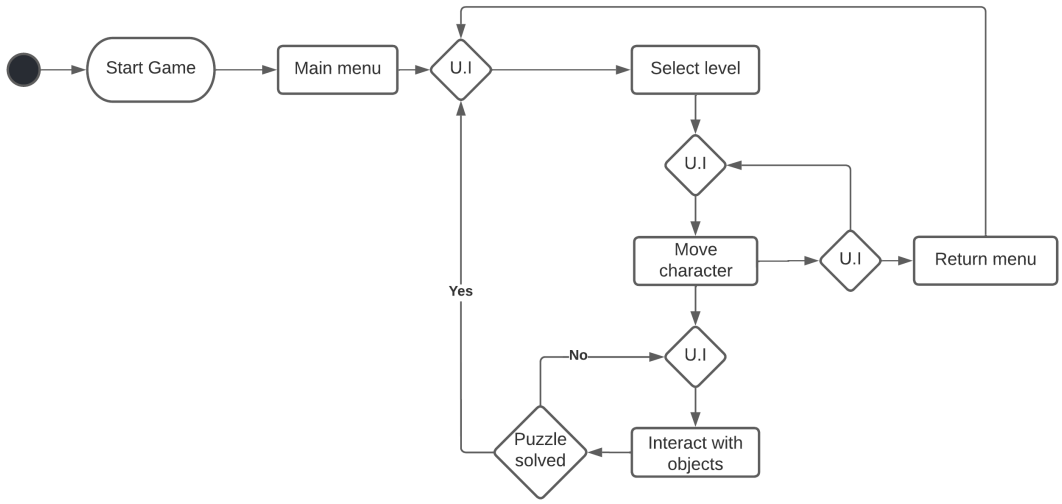

Figure 3.1: Flowchart of the game (U.I is user input)

3.4 Interface Design

The only interface element visible during the game will be the item that the character has equipped to be used in that particular level. This will be in the top left corner of the screen, where it will not cover any element of the stage.

On the other hand, if the escape key is pressed, the pause menu will appear in the centre of the screen, where there will be three buttons: one to return to the game, one to restart the current level from scratch, and the last one to return to the main menu.

Figure 3.2: UI object

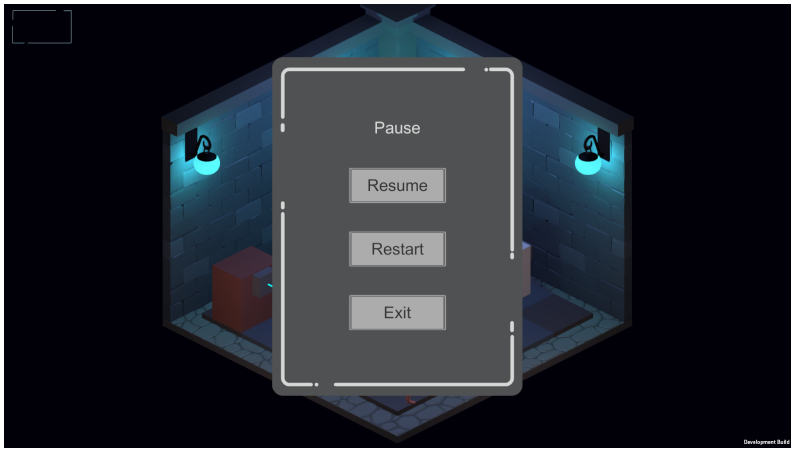

Figure 3.3: Pause menu design

3.5 Game design research and references

The main games that have inspired the development of the project are Monument Valley and The house of Da Vinci.

Monument Valley [2] is a puzzle game developed for mobile devices by ustwo games. It consists of a series of small levels seen in isometric perspective with minimalist aesthetics on which the player will have to interact with the environment to complete the level. These levels are characterised in that the player has to play with the perspective of the objects, generating mostly impossible geometries in order to continue advancing and achieve the objectives(see figure 3.4).

Figure 3.4: Monument Valley level

The House of Da Vinci [1] is a puzzle game available for all devices, developed by Blue Brain Games. It is a first-person game in which the player progresses through different rooms in which he is presented with a variety of puzzles distributed around them. To advance, the player must complete all the puzzles in a certain order, as completing some of them unlocks objects needed to complete other puzzles; once completed, the player moves on to the next room and repeats the same process (see figure 3.5).

Figure 3.5: The House of Da Vinci level

Chapter 4

Work Development and Results

Contents

This section will comment on the work carried out in each of the sections involved on the development of the project, commenting the design changes that have been applied, together with the problems that have arisen during development.

4.1 Player movement

The implementation of a movement system for the character, which feels responsive and fluid for each action performed, is one of the most important aspects of the development of the project, as this will be the element with which the player interacts with the elements of the game.

This will be controlled with Unity's new input system[5], which allows users to easily modify all the actions assigned to the character, creating different action maps with their respective actions to which one or several keys are assigned that will respond when that action is executed (see figure 4.1).

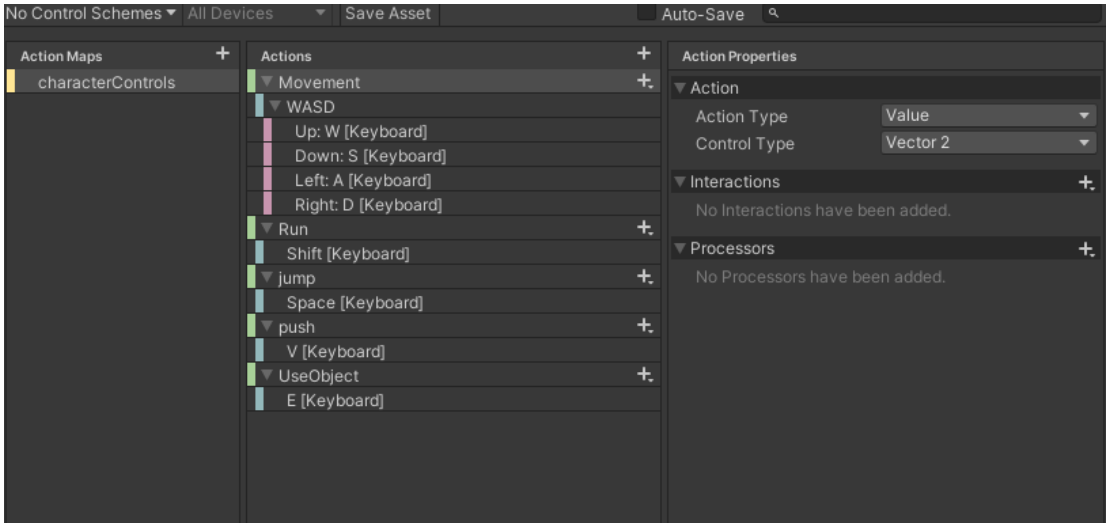

Figure 4.1: Input system window

Using this system, the keys W, A, S and D will be used to move the character, which will return a Vector2 that will represent the direction in which the player wants to move, being W and S for the Y axis and A and D for the X axis. As the game scenarios are represented in three dimensions, once this vector has been obtained, it will have to be transformed into a Vector3 in which we will assign the values to the X and Z components, which correspond to the axes of the horizontal plane of Unity. Besides, the shift key has also been configured so that the character can run around the map, the only thing it will do is to increase by three the normal movement speed of the character. Since the camera is rotated to give the isometric perspective to the scene, the X and Z components must be multiplied by the respective camera positions components to move the character relative to the camera

Regarding the jumping mechanics, the character was initially configured with a rigidbody next to a collider so that the unity physics system would take care of the collision with objects and the character's gravity; and to perform the jump, all that is done is to apply an impulse force on the character in the vertical axis. When checking that everything worked correctly, it was observed that when jumping against an object, the character would stick to the collider randomly, even changing the friction of the materials to zero.

As a solution we will use a character controller[4], a unity component that provides all the basic features of character movement such as step detection, but is not affected by all the physics of the engine, only the collision detection so the gravity will have to be calculated manually. The new jump implementation will use the ground detection of this component, which tells if the bottom of the collider is touching an object, and if it doesn't collide a value will be subtracted from the Y component of the movement depending on the gravitational constant and a velocity multiplier.

4.2 Animations

For each of the actions that the character performs on the stage, such as moving or jumping, an animation corresponding to each movement is executed to give visual feedback to the player. All these animations are downloaded from the Mixamo website which provides a wide variety of animations for the characters.

Within unity these animations are managed by an animation tree (see figure 4.2), which will activate the different nodes by code depending on the action performed.

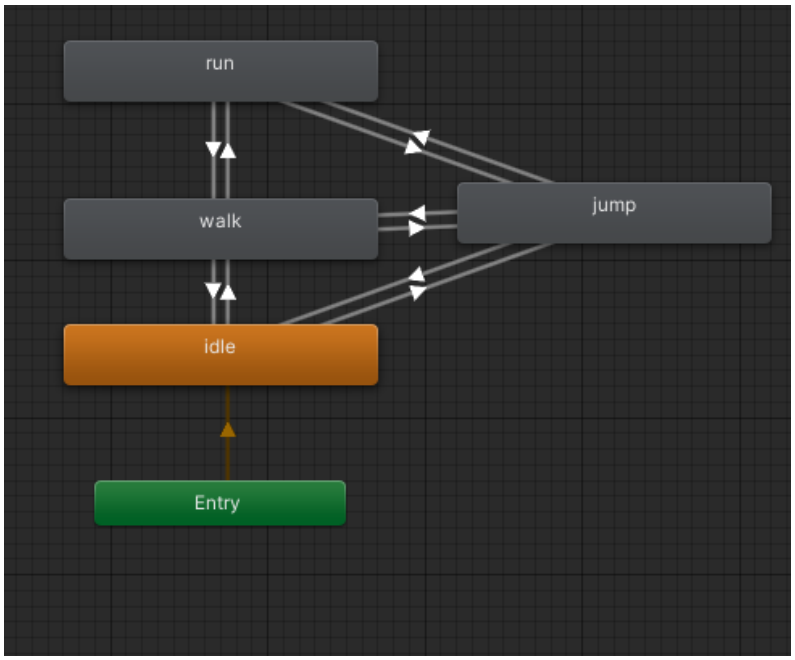

Figure 4.2: Animator Graph

4.3 Mechanics

To solve the puzzles that appear throughout the levels, the player will have two mechanics to interact with the stage: the use of items and pushing and pulling objects.

Pushing and pulling objects is executed by pressing the B key. Once pressed, a raycast will be launched in front of the character which will detect whether the detected object in front of it can be pushed or not. If so, it will wait for the player to make a movement in one of the directions to move the object. While the object is moving, the player's controls will be disabled to prevent other actions from being performed.

All movements are performed within a grid (see figure 4.3) which will limit the movement of the objects to each of the cells that compose it. Before each movement is made, the cell where the selected object is to be moved will be calculated and checked

to ensure that it is free. This system will ensure that all the pieces end up in exact positions that do not give errors when solving the puzzle.

Regarding the use of items, it will be executed with the E key. Once pressed, the function assigned to the key of the item that the character has equipped at that moment will be executed.

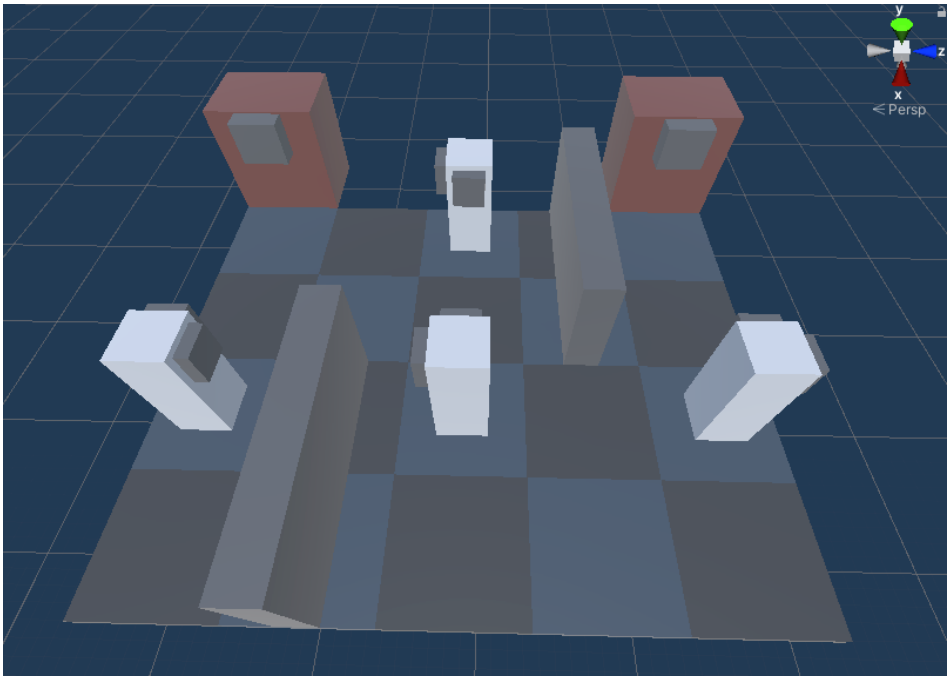

Figure 4.3: Grid prefab

4.4 Game Levels

There are three different puzzles in the game, each separated into different scenes, as well as a tutorial level.

- **Tutorial:** The tutorial level will introduce the two main mechanics of the game: pushing and pulling objects, and the use of items (see figure 4.4).

  A grid will appear with a box which must be moved to a square marked in green. Once taken to this point we unlock a key which we can use to open the door and advance to the first level.

  During this level all the mechanics will be explained with text on screen so that the player knows how to interact with the elements that are presented during the development of the game. First, it will be explained that you can move around the stage with the W,A,S and D keys, run with the SHIFT key pressed, and jump with the Space key. Next, the mechanics of pushing objects with the B key will be introduced, which you will have to use to move the box to the marked point. And finally the use of objects with the E key, in which you will have to use a key in front of the door that appears on the stage to continue in the game.

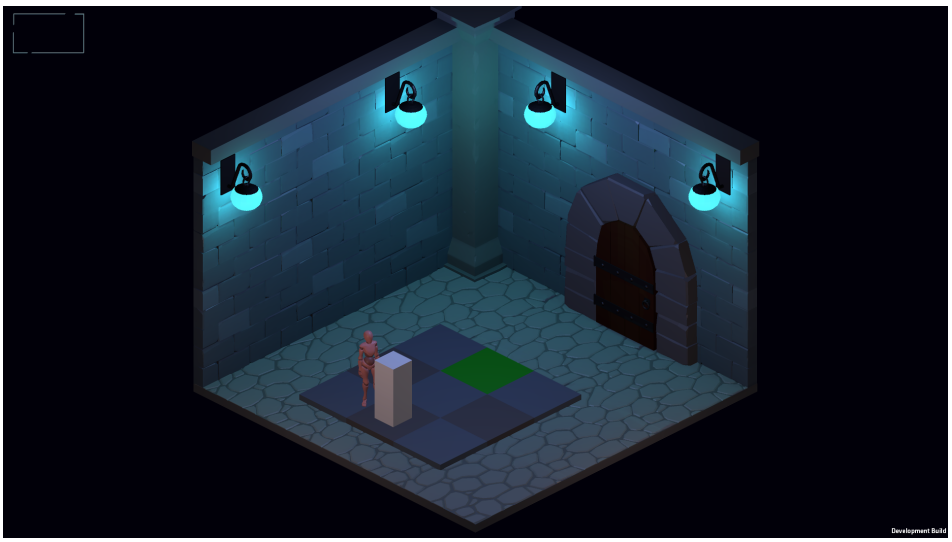

Figure 4.4: Tutorial layout

- **Level 1:** The first level will use the mechanics of pushing objects to solve the puzzle (see figure 4.6).

  In this puzzle there are four types of objects: a light emitter, reflectors, obstacles and a receiver. The objective is to move the reflectors around the grid and place them in such a way that the beam emitted by the emitter is reflected by the reflectors and reaches the receiver to complete the level.

  The operation is based on a raycast that spreads through space. Each reflector is formed by a pair of objects located on different sides of the base that will be in charge of propagating the light ray through the puzzle. First a raycast is launched from the emitter. If it collides with one of the objects in the reflector, it will launch a ray from the pair, which will follow the same logic as the emitter. When the object that detects the raycast is the light receiver, the two objects will be connected, thus completing the puzzle and moving on to the next level.

  For the drawing of the light ray we have used the unity line renderer component (see figure 4.5), which draws a line in space given a set of points. These points will be the result of the collision of the propagating beam. Each time a reflector is moved, the points will be calculated to join only those that are connected to the emitter.

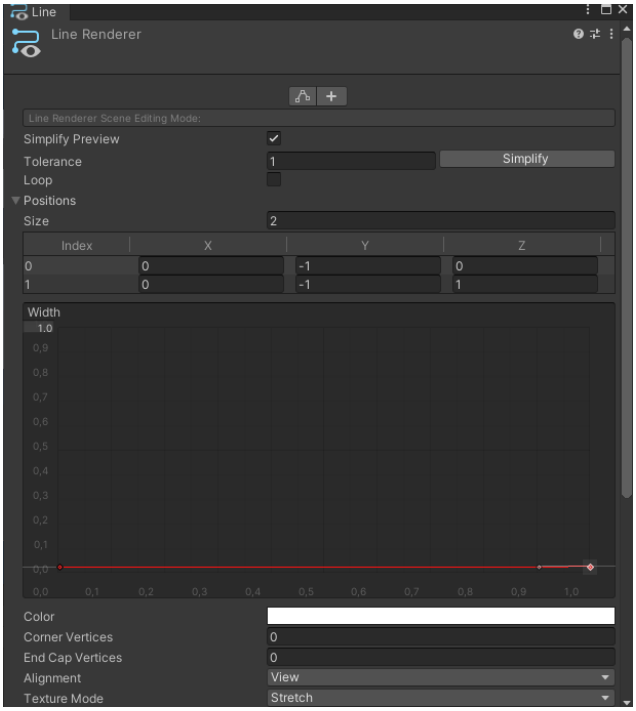

Figure 4.5: Line renderer component

To update the beam as the reflectors move, we will use UnityEvents[9]. These allow to subscribe the functions that you want to associate to the event, and execute them when the event is called at some point. In this case two events will be created, one which will eliminate the points of the light ray when an object starts to move; and another one which will re-do the whole raycast from the emitter to recalculate the light ray when the object has finished moving.

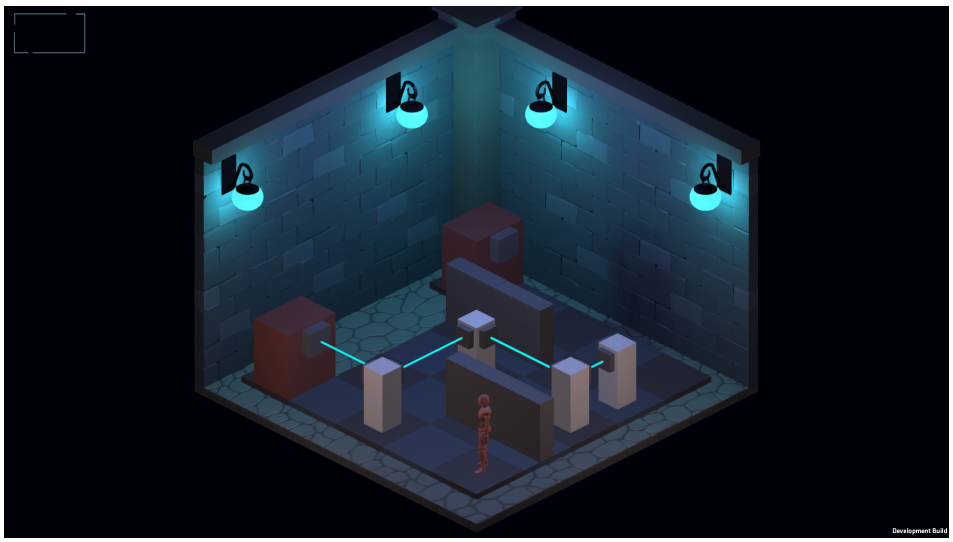

Figure 4.6: Level 1 layout

- **Level 2:** This second level employs the use of objects to solve the puzzle. In it we will find four objects located on the walls which are linked to a sphere that changes colour when we interact with the equipped object, which in this case is a flashlight (see figure 4.7).

  Each of these objects will rotate when a button in the centre of the room is activated. Depending on the colour selected in each of the spheres, the associated objects will rotate in different ways: Red colour, rotation of the object and the object to its left; yellow colour, rotation of the object and the object to its right; Purple colour, no rotation of the object. Even if the non-rotation mode is selected, it will still be affected by the rotations of the other objects.

  The aim of the puzzle is to alternate the different rotation modes with the help of the flashlight, so that one of the selected faces of each of the objects, which will be highlighted by a different colour to the others, faces the character.

  To rotate all the pieces when the button is pressed, a rotation function is first created in each of the objects which detects which is its rotation mode, and which are its left and right neighbours to perform the rotation. Once applied from the puzzle controller, each of these functions is called in turn so that the turns are

carried out in a fluid and orderly manner. When all the turns have been completed, each element is checked to see if it is in the correct position to see if the puzzle has been completed.

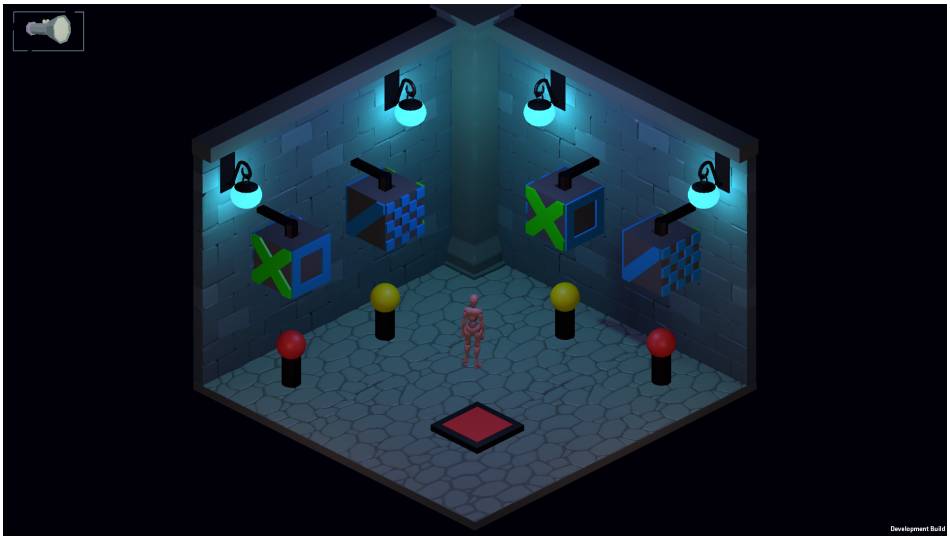

Figure 4.7: Level 2 layout

- **Level 3:**The third level will make use of the two previously mentioned techniques: pushing and pulling objects and the use of items. This will be a variation of the first level of the game where the objective was to connect the light beam from one end to the other (see figure 4.8).

  In this level there will be two light emitters with their respective receivers. As in the first level we will have to move the reflectors inside the cells to connect both emitters and receivers. Each of the reflectors will have a direct colour indicator which will specify the specific light beam they can spread, which will prevent them from reflecting any of the rays.

  With the help of the item, which in this case will be a small bell, we will be able to interact with the reflectors so that they make a 90º turn clockwise. When the object is used, a raycast will be launched in front of the player, which, if it detects that the object that has been interacted with can be rotated to make the turn.

  Unlike the first level in which there are a series of obstacles that prevent the movement of objects, in this one all the cells are free to move the objects, except for one, which is the point where the two receivers converge. This is occupied by a special reflector which cannot be moved out of the cell but can be rotated, and which has four reflection points, two for each light beam. Because this reflector is located in a key position which prevents both light beams from connecting directly with a single object, it will be the main piece to be taken into account for solving the puzzle.

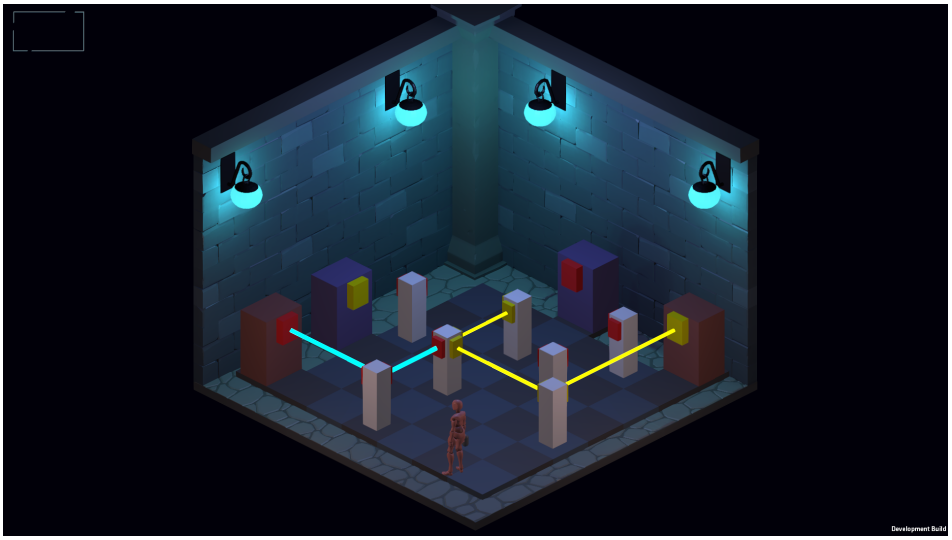

Figure 4.8: Level 3 layout

4.5 Object Modelling

All the elements that appear in the game like the items, the character and the scenery will be modelled in 3D within the Blender program. For this, the low poly modelling technique will be applied, which focuses on representing the models in a simple way and with few polygons, resulting in a simple and striking aesthetic.

Figure 4.9: Example of an in game item (flashlight)

The floors and walls of the scenarios will be modelled as "tiles", which have a continuity on the parallel sides. This design allows the scene to be modelled in any size desired and can be enlarged at any time, as the models are placed side by side in such a way as to give the sensation of a single solid object.

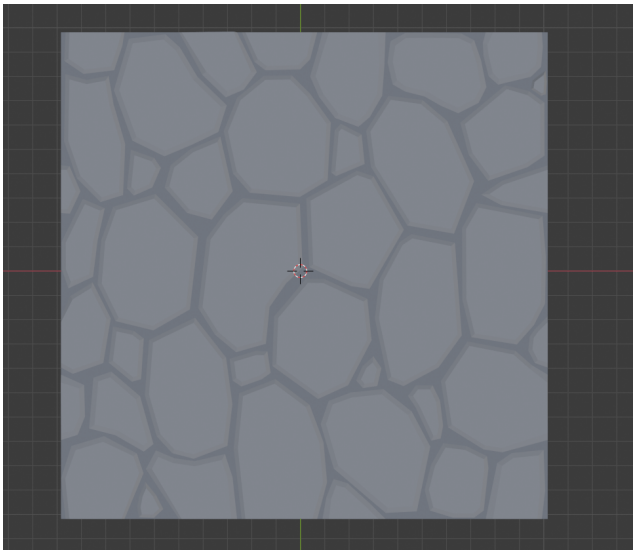

Figure 4.10: Floor tile on Blender

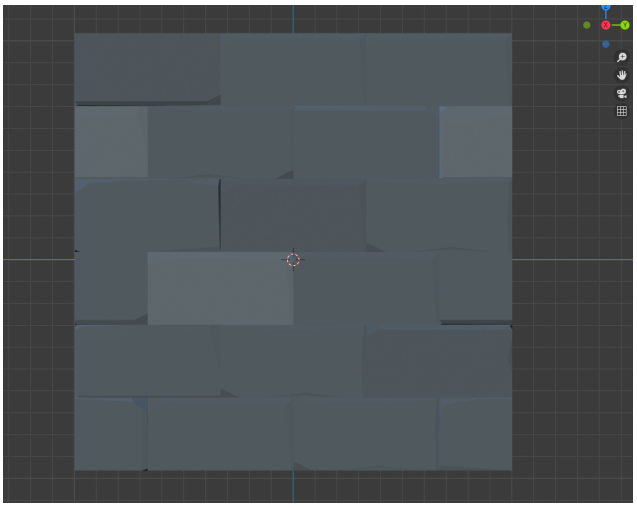

Figure 4.11: Wall tile on Blender

Also within blender an approximation of the materials that the object will have within unity will be created, due to the fact that many of the properties such as the reflection are not imported correctly into the engine.

4.6 Scenes

During the course of the game we will encounter three different types of scenes:

- **Main scene:** It will display the title of the game, the play button that will take you to the next scene, a settings button that will allow you to adjust the volume of the music and sound effects, and a button to exit the game (see figure 4.12).

Figure 4.12: Main scene design

- **Level selector:** This is the scene that will appear when we press the play button. It will show four images indicating the levels to select. If you have never played the game before, the only scene available to select will be the tutorial, which is mandatory to continue to the other levels. Once the first level is completed, the second level is unlocked and so on (see figure 4.13).

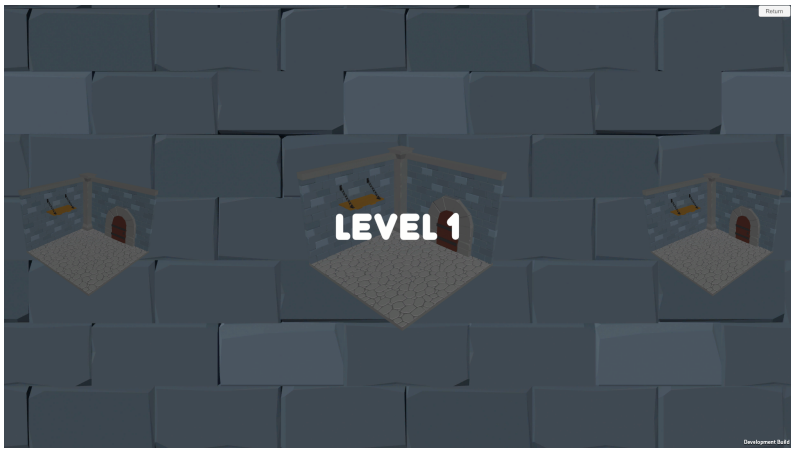

Figure 4.13: Level selector design

- **Game level:** The scene of the level we have previously selected. Once inside, you can start moving the character and solve the puzzle. If you open the pause menu and press the button to return to the main menu, the main scene will be reloaded.

All the scenes are assembled within the blender program which allows very easily to put together several elements. With tools such as the magnet you can adjust the different geometries so that everything fits together correctly without any errors (see figure 4.14).

As all the 3D models are modelled within blender, when putting all the pieces together, the same scaling will be maintained at all times. Once assembled, only the fbx file will be imported into the engine to introduce the scenes.

Figure 4.14: Base level model on Blender

On the other hand, to save the player's progress we will make use of Unity's integrated class called PlayerPrefs[7], which allows us to store different types of values such as int, floats or strings which will not be lost when the application is closed (see figure 4.15). To do this, each level will be assigned a string with an associated numerical value, initially set to -1. If a level has been completed, the value will be changed from -1 to 1 and it will be unlocked for the rest of the game.

```
public void ReadPlayerData()
{
    int level_1 = PlayerPrefs.GetInt("Level_1");
    int level_2 = PlayerPrefs.GetInt("Level_2");
    int level_3 = PlayerPrefs.GetInt("Level_3");

    //
    if (level_1 == 1)
        buttonLevel_1.enabled = true;
    else
        buttonLevel_1.enabled = false;
    //
    if (level_2 == 1)
        buttonLevel_2.enabled = true;
    else
        buttonLevel_2.enabled = false;
    //
    if (level_3 == 1)
        buttonLevel_3.enabled = true;
    else
        buttonLevel_3.enabled = false;
}
```

Figure 4.15: Use of PlayerPrefs to ulock levels

For the level selector we have used a ScrollRect component which allows, given a content, to move the objects in it horizontally or vertically, in this case each of the buttons that give access to the level.

To make this movement more dynamic and provide a greater visual response we have used a free asset from the Unity Asset Store called Simple Scroll-Snap[3] developed by Daniell Lochner. This is an add-on for this component which provides a large number of extra features that are not implemented by default in Unity. In this case the magnetisation of the objects will be used so that one of the buttons is always centred on the screen when the scrolling is finished, and the scaling of the elements so that the button that is centred has a larger size than the others.

4.7 Light and Post Processing

For the lighting of the scenes, a series of light points have been used throughout the stage that will simulate the light coming from the lanterns that are scattered along the walls (see figure 4.16). This light will use the Mixed lighting mode which combines the elements of real time lighting, which is generated in real time for the elements that move in the scene; and those of baked lighting, which is pre-calculated to generate a light map on the models that remain static within the scene [6].

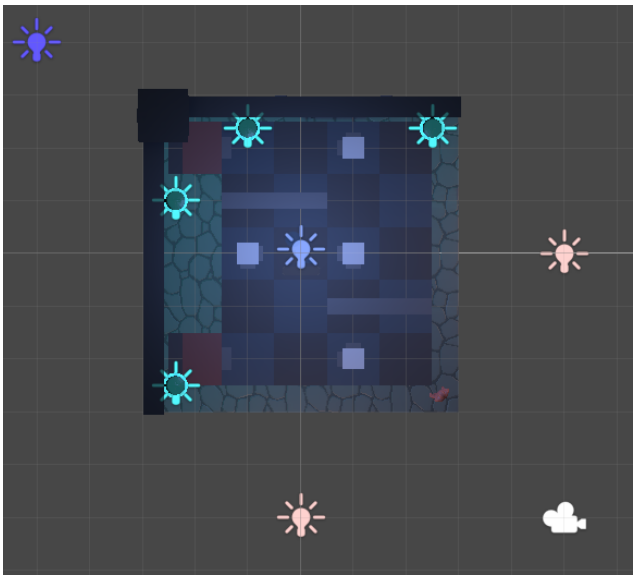

Figure 4.16: Light setup of the scenes

On the other hand, we have also configured the material of some objects such as the line renderer of the first puzzle so that they have the property of light emission, thus generating a more realistic result to the scene. For this we have used the Post-Processing package provided by unity [8]. This provides a series of filters that are applied to the final result of the scene, one of them is the light emission of the materials. To add it, a Post-processing Layer is added to the scene's camera, and once implemented, each of the filters can be added to the image (see figure 4.17).

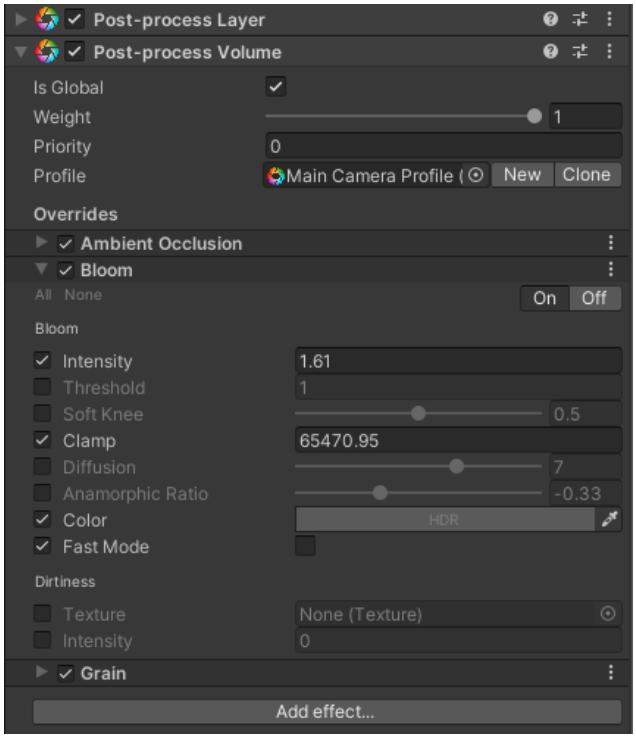

Figure 4.17: Post Process component

CHAPTER 5

Conclusions and Future Work

Contents

In this chapter, the conclusions of the work, as well as its future extensions are shown.

5.1 Conclusions

In this project I have applied all the knowledge I have learnt throughout the degree, both at a programming level, programming all the scripts within the Unity engine, and at an artistic level, modelling all the 3D objects.

I have also learned new techniques such as the use of the new input system or event-driven programming, which will be very useful when starting to develop new projects.

5.2 Future work

One of the main areas for future expansion would be to add a wider variety of levels with new mechanics for the player to interact with. This would make the game more complete and much longer.

Another aspect to consider would be to add more detail to the scenes. On the background, which is currently just a flat colour, small visual effects or moving objects could appear. For the 3D models of the scenes, different types of objects could be designed with different themes depending on the mechanics to be used or the theme of the puzzle.

Finally, the implementation of a varied soundtrack would also be an addition to enhance the player's experience throughout the game.

Bibliography

[1] Blue Brain Games. The house of da vinci. https://www.bluebraingames.com/. Accessed: 2022-06-16.

[2] Ustwo Games. Monument valley. https://www.ustwogames.co.uk/. Accessed: 2022-06-16.

[3] Daniel Lochner. Simple scroll-snap. https://assetstore.unity.com/packages/tools/gui/simple-scroll-snap-140884. Accessed: 2022-06-29.

[4] Unity. Character controller. https://docs.unity3d.com/Manual/class-CharacterController.html. Accessed: 2022-06-16.

[5] Unity. Input system. https://docs.unity3d.com/Packages/com.unity.inputsystem@1.3/manual/index.html. Accessed: 2022-06-16.

[6] Unity. Light mode: Mixed. https://docs.unity3d.com/Manual/LightMode-Mixed.html. Accessed: 2022-06-30.

[7] Unity. Playerprefs. https://docs.unity3d.com/ScriptReference/PlayerPrefs.html. Accessed: 2022-06-29.

[8] Unity. Post-processing. https://docs.unity3d.com/Packages/com.unity.postprocessing@2.3/manual/index.h Accessed: 2022-06-30.

[9] Unity. Unityevents. https://docs.unity3d.com/Manual/UnityEvents.html. Accessed: 2022-06-16.

Appendix A

Other considerations

A.1 Source Code

All the assets used for the development of the videogame, both the scripts and the rest of the game elements can be found in the following github link, which contains the complete Unity project: https://github.com/jorgebg12/TFG*jorgeBartolGuillamon*