Czech Technical University in Prague
Faculty of electrical engineering
Department of computer graphics and interaction

Bachelor thesis

First-person puzzle game with switching playable characters as the main mechanic

Aleksandr Murugov

Supervisor: Ing. Ladislav Čmolík, Ph.D.

Study program: Open informatics, Bachelor

Specialization: Computer games and graphics

January, 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Murugov**	Jméno: **Aleksandr**	Osobní číslo: **466360**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Zadávající katedra/ústav:	**Katedra počítačové grafiky a interakce**		
Studijní program:	**Otevřená informatika**		
Studijní obor:	**Počítačové hry a grafika**		

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Logická hra, kde přepínání mezi různými postavami je hlavním herním mechanismem

Název bakalářské práce anglicky:

First-person puzzle game with switching playable characters as the main mechanic

Pokyny pro vypracování:

Analyzujte herní principy v logických počítačových hrách z pohledu první osoby (např. Portal, The Talos Principle, The Turing Test) a v počítačových hrách, kde je hráč nucen přepínat mezi několika herními postavami. Dále analyzujte modulární komponenty a jejich využití v návrhu herních úrovní. Na základě analýzy vytvořte design dokument pro logickou počítačovou hru z pohledu první osoby využívající přepínání mezi herními postavami. Na základě design dokumentu vytvořte v Unreal Engine alespoň tři hratelné herní úrovně, které budou ve svém návrhu využívat modulární komponenty. Proveďte kvalitativní evaluaci výsledné hry s alespoň pěti uživateli.

Seznam doporučené literatury:

R. Koster, Theory of Fun for Game Design, 2nd edition, O'Reilly Media, 2013.
J. Schell, The Art of Game Design: A book of lenses, CRC Press, 2008.
J. Lee, Learning Unreal Engine Game Development, Packt Publishing, 2016.
B. Sewell, Blueprints Visual Scripting for Unreal Engine, Packt Publishing, 2015.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ladislav Čmolík, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **21.09.2020** Termín odevzdání bakalářské práce: **05.01.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Ladislav Čmolík, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.

Datum převzetí zadání	Podpis studenta

Declaration

I hereby declare I have written this thesis independently and quoted all the sourcces of information used in accordance with methodological instructions on ethical principles for writing and academic thesis.

Prague 5. 1. 2020

....

Acknowledgments

I would like to express my gratitude to my supervisor Ladislav Čmolík, Ph.D. for his useful advice, comments, and remarks. Furthermore, I would like to thank my parents and friends for all the support.

Abstrakt

Cílem tohoto projektu bylo vytvoření logické hry z pohledu první osoby, kde hráč má schopnost ovládat jiné postavy pro řešení hádanek. Herní vývoj byl předcházen analýzou existujicích podobných her jako logické hry z pohledu první osoby: Portal, The Talos Principle, Turing test; Hry s prohozením hrajicích postav: GTA V, Driver: San Francisco, Asterix & Obelix XXL; Immersiv sim Deus Ex s jeho systémem předmětů. Pak výběrem nejvhodnějšího herního enginu mezi Unity a Unreal. Dále jsem analýzoval modulární komponenty a jejích použití v level-designů. Mým úkolem bylo zkombinovat mechaniky těch třech druhů her a použit modulární komponenty pro vytvoření úrovně a vytvořit něco nového a originálního. Práce popisuje navrh a implementaci hry s nasledujicím kvalitativním testováním s šesti uživateli . Výsledkem je hra s třemi úrovněmi.
Klíčová slova: Unreal Engine, Logická hra, Modulární komponenty

Abstract

The aim of this project was to create a first-person puzzle game where the player has the ability to take control of other characters on the fly to solve puzzles. The game development was preceded by analysis of existing similar games such as first-person puzzles: Portal, The Talos Principle, Turing test; Games with switching playable characters: GTA V, Driver: San Francisco, Asterix & Obelix XXL; Immersive sim Deus Ex with its item system. Then choosing the best fitting engine for such game between Unity and Unreal. Futher I analyzed modular components and their use in level design. My objective was to combine mechanics of these 3 types of games and use modular components for level creation to create something new and original. The thesis describes design and impementation of the game followed by qualitative testing of 6 users. The result is a 3-level game.

Keywords: Unreal Engine, Puzzle game, Modular components

Contents

1 Introduction 1
2 Analysis 2
  2.1 First-person puzzle games 2
    2.1.1 Portal 2
    2.1.2 The Talos Principle 3
    2.1.3 The Turing test 4
  2.2 Games with switching playable characters 5
    2.2.1 GTA V 5
    2.2.2 Asterix & Obelix XXL 5
    2.2.3 Driver: San Francisco 6
  2.3 Item system in Deus Ex 7
  2.4 Game engine 8
3 Game design 9
  3.1 Mechanics 9
    3.1.1 Movement 9
    3.1.2 Switching characters 9
    3.1.3 Items and inventory 9
    3.1.4 Doors and Keys 10
    3.1.5 Jumping boots 11
    3.1.6 Cubes 12
    3.1.7 Platforms 12
    3.1.8 Checkpoints 13
    3.1.9 Menus 13
    3.1.10 Sounds 14
    3.1.11 Music 14
  3.2 Puzzles 15
    3.2.1 Level 1 (tutorial) 15
    3.2.2 Level 2 21
    3.2.3 Level 3 22
4 Implementation 25
  4.1 Game mechanics 25
    4.1.1 Base Character 25
    4.1.2 Game mode 27
    4.1.3 Character switching 27
    4.1.4 Inventory system 29
    4.1.5 Doors 34
    4.1.6 Moving platforms 36
    4.1.7 Containers 36
    4.1.8 Moving objects 36
    4.1.9 Outline 37
    4.1.10 Sounds 38
    4.1.11 Checkpoints 38
    4.1.12 Lighting 39
  4.2 Modular components 41
    4.2.1 Walls, floors, ceilings 41
    4.2.2 Doorways 42

4.2.3 Platforms ... 42
5 Testing ... 43
5.1 Testing 1 ... 43
5.1.1 Level 1 finish time ... 43
5.1.2 How is the main mechanic of switching characters in this form (first-person, controlling only one at a time) new to you? ... 43
5.1.3 Level duration (expect more levels of the same duration) ... 44
5.1.4 Pace of the game ... 44
5.1.5 Introduction to new mechanics ... 44
5.1.6 Rooms size ... 45
5.1.7 Which way did you finish the last puzzle? ... 45
5.1.8 Anything particularly annoying or bugged? ... 46
5.1.8 General suggestions ... 46
5.1.9 Conclusion ... 46
5.2 Testing 2 ... 47
5.2.1 Level 2 finish time ... 47
5.2.2 Level 3 finish time ... 47
5.2.3 Originality of puzzles ... 48
5.2.4 Your favorite puzzle/room (up to 3, whole game, very simple not included) ... 48
5.2.5 How do visual elements fit together? Interior, buttons, platforms, items etc. (not graphics) ... 48
5.2.7 Perfomance ... 49
5.2.8 Bugs ... 49
5.2.9 What needs to be done to make you buy this game? ... 49
5.2.10 Conclusion ... 49
6 Conclusions ... 50
Appendix ... 52

1 Introduction

From ancient times people of all ages loved to solve abstract problems as a part of a game. People love when their sequence of right actions and logical conclusions leads to a desirable result.

Thanks to technological progress the amount of problems, puzzles and riddles can be created in video games is limitless nowadays. From simple kids‘ games to complex horrors such as Resident Evil. One particular subgenre was established in 2007 with the game Portal. It was such a success that people started to call games with similar mechanics, even without portals, Portal-like. Although the right name would be first-person puzzle-platform game.

I came up with concept of my game by bringing the idea of character switching as in GTA V, Driver: San Francisco, Asterix & Obelix XXL, etc. where 2 or more agents controlled by 1 player cooperate, to the world of Portal-like games. Surprisingly I discovered only 2 first-person puzzles that combine such mechanics: The Talos Principle and Turing Test, although in slightly different form. Another feature I wanted to bring is items and inventory. Each agent would have its own items he can make use of. For example, a note with a keycode to a door. The main inspiration was the Deus Ex game. Further, I will describe games and their particular elements that inspired me, reasoning behind the use of Unreal Engine and the implementation of my game I would like to name "Such simple minds".

In Chapter 2 I will analyze existing games that inspired me. It is split into 3 parts: Portal-like games, games where you can switch characters and inventory system in immersive sim Deus Ex. In the end I will choose between engines Unity and Unreal.

In Chapter 3 I will describe all game mechanics and elements, show how player-switching can bring more depth to puzzle-solving and walk through the 3 levels to show how I created unobtrusive tutorial where the player learns new mechanics naturally by himself without much reading.

Chapter 4 will cover the implementation of my game. First part will show how I implemented game logic in Unreal Engine 4. In second part I will show how modular components can be used for level creation.

In Chapter 5 I conduct 2 qualitative tests of my game. The first took place in the middle of the development after implementing basic mechanics and designing the first level. The second one was at the end of the development after implementing all mechanics and designing 3 levels.

2 Analysis

2.1 First-person puzzle games

2.1.1 Portal

Portal [1], a game that basically defined the genre of First-Person puzzle-platformers. In 2007 Valve released it as a part of Orange Box, often earning more praise than either Half-Life 2: Episode Two or Team Fortress 2, two titles also included in The Orange Box. It was praised for its unique gameplay and dark deadpan humor. [2]

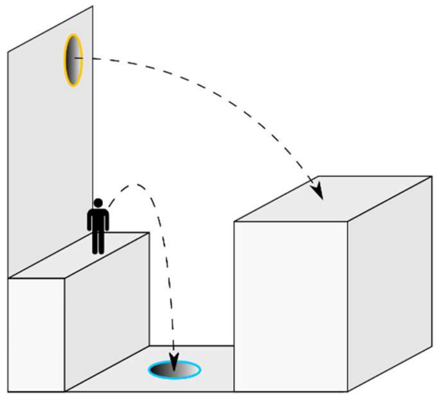


Figure 2.1 Portal mechanic

In Portal, the player controls the protagonist, Chell, from a first-person perspective as she is challenged to navigate through a series of rooms using the Aperture Science Handheld Portal Device, or portal gun, under the watchful supervision of the artificial intelligence GLaDOS. The portal gun can create two distinct portal ends, orange and blue. The portals create a visual and physical connection between two different locations in three-dimensional space. Neither end is specifically an entrance or exit; all objects that travel through one portal will exit through the other. An important aspect of the game's physics is momentum redirection. [3]

The game was a huge success. It's simplicity and accessibility made it enjoyable for people of all ages and previous gaming experiences. Valve wrote that they felt that Portal "makes physics, math, logic, spatial reasoning, probability, and problem-solving interesting, cool, and fun", a necessary feature to draw children into learning. [4] Every game after Portal release that can be described as First-Person puzzle-platformer with door switches, buttons, physical interaction with cubes, atmospheric, with a little bit of story is doomed to be called portal-like, even when there are no portals.

2.1.2 The Talos Principle

The Talos Principle [5] is a first-person puzzle video game created by the Croatian developer Croteam. Released in 2014.

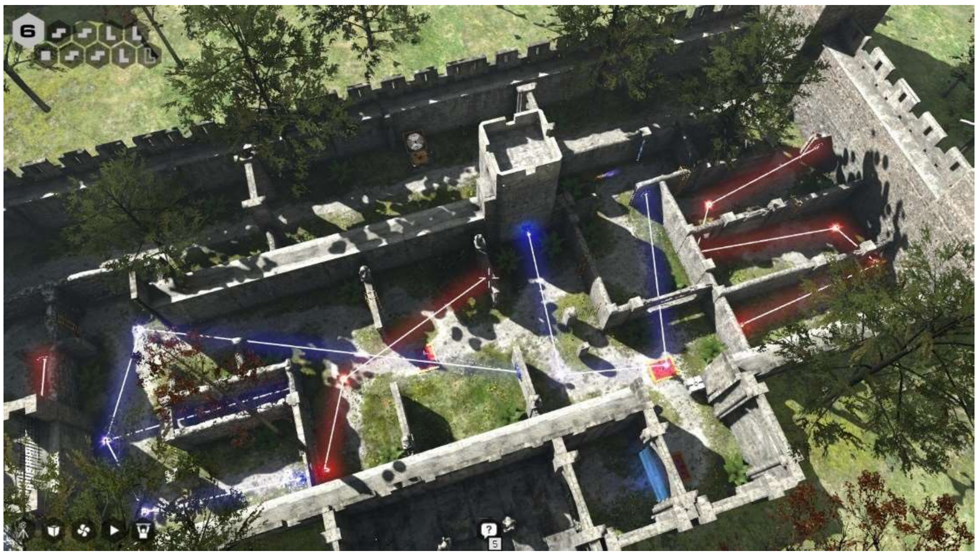


Figure 2.2 The Talos Principal typical level

The game revolves about finding "sigils" by navigating enclosed areas and overcoming obstacles with them. These include drones and turrets. They can be disabled using portable jammers, which also disable force fields blocking the way. As the game progresses, more puzzle elements become available. Portable refractors that activate light-switches, boxes to climb or block drones and large fans launching the player or other objects. Later, the player gains access to machine that creates a time-recording of their actions. These recordings and a main hero must cooperate. Brief description of recording machine: main character creates a clone that is controlled by the player, starts recording, does a sequence of actions, stops recording. Then the main protagonist can observe the recording and adjust to changes he is making in the process. All changes revert to the beginning when the recording stops. Inexperienced players may struggle trying to grasp the concept of playing several characters simultaneously. In relation to my project that is the most similar mechanic to that I wanted to implement.

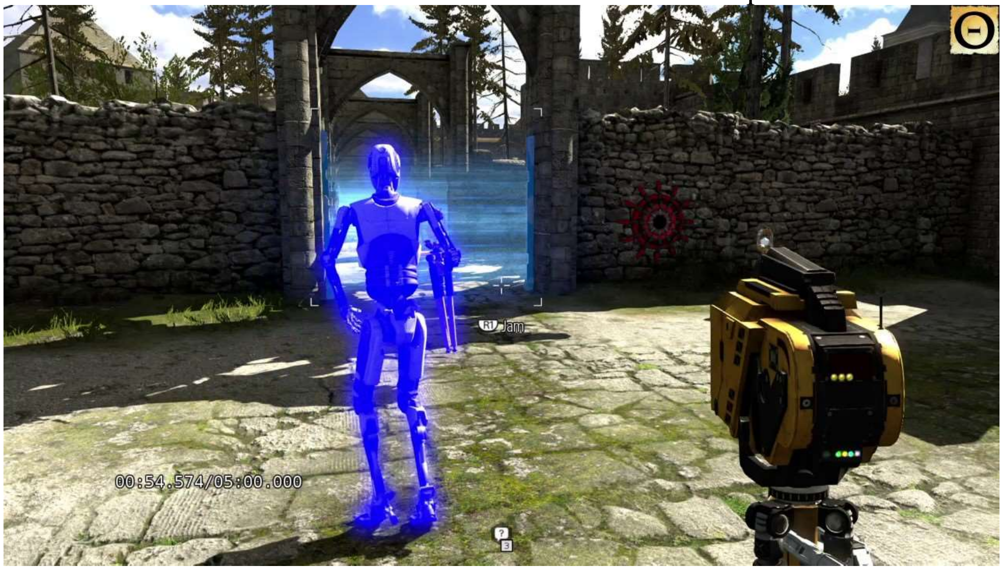


Figure 2.3 Player recording

2.1.3 The Turing test

The Turing Test [6] is a first-person puzzle video game developed by Bulkhead Interactive in 2016.

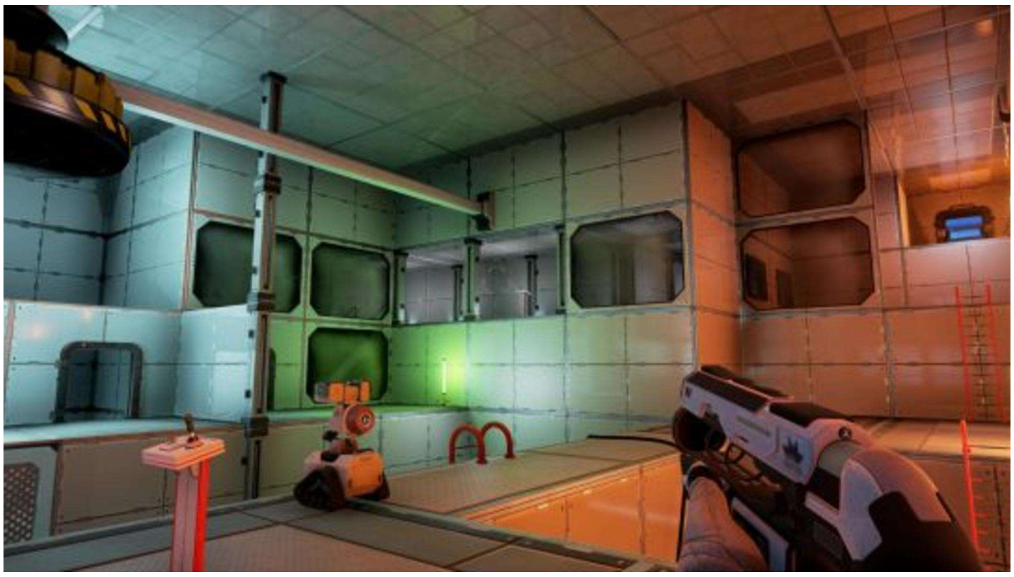


Figure 2.4 The Turing test

The puzzles are solved with a gun that collects and fires spheres into special sockets to unlock doors or activate other mechanisms. Further in the game the player is given the ability to watch through monitoring cameras and control mobile robots. The robot-controlling part is also similar to my vision of character switching. But robots there are very limited in possibilities.

2.2 Games with switching playable characters

2.2.1 GTA V

Rockstar released **GTA V** [7] in 2013. It was the first GTA that had multiple playable characters. In several missions switching characters brings more cinematographic effect. Player can observe the situation from different point of view, though there is no complex interaction between heroes. Each character has his own stats, money, clothes and available missions.

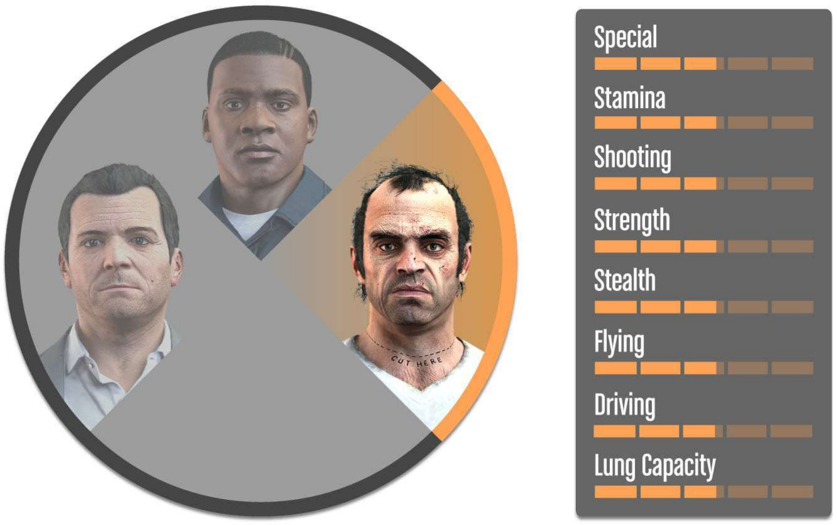


Figure 2.5 Character selection in GTA V

2.2.2 Asterix & Obelix XXL

Asterix & Obelix XXL [8] is an action adventure video game first released in 2003 developed by Étranges Libellules

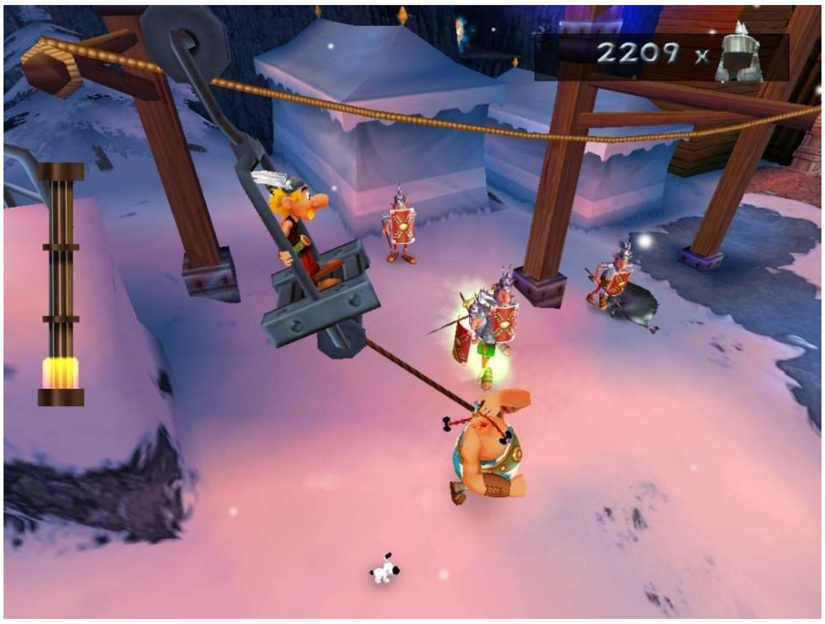


Figure 2.6 Obelix pulls Asterix on a platform

The gameplay varies from fighting romans to solving puzzles that require cooperation from both Asterix and Obelix. The protagonists have slight differences. Obelix is big and strong, he can break enforced crates and push and pull larger platforms than Asterix. Asterix is small and agile, which lets him to crawl into small tunnels and be carried by Obelix on special platforms. Switching activates automatically or by entering special relay station.

2.2.3 Driver: San Francisco

Driver: San Francisco [9] is an action-adventure racing video game released in 2011.

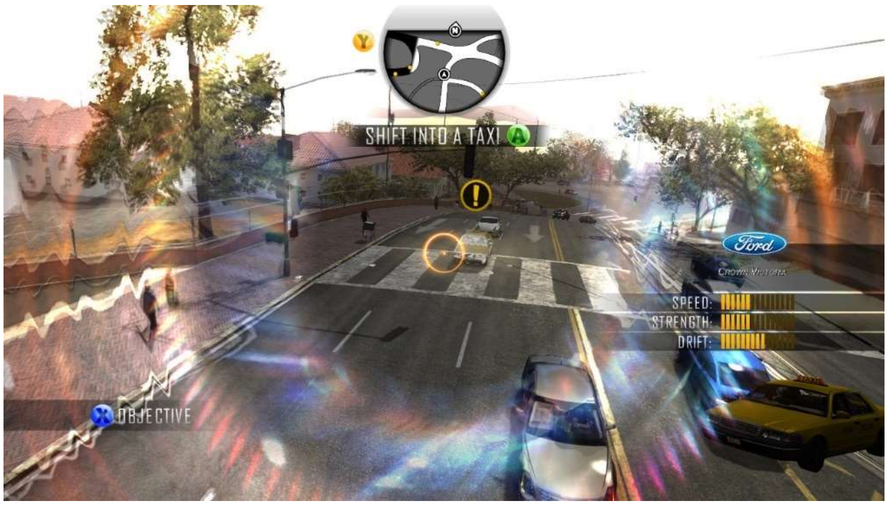
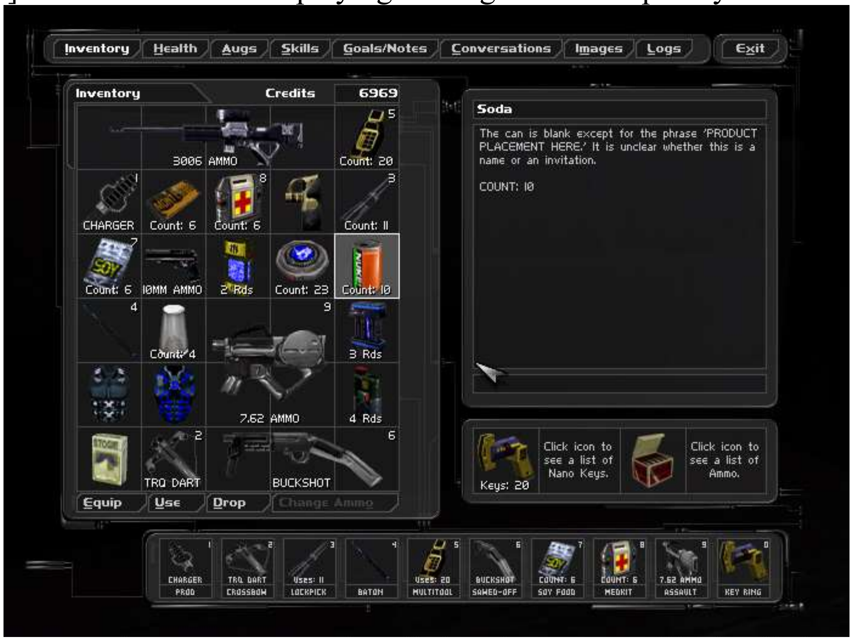


Figure 2.7 Shifting into another car

The game is quite unique in racing genre. Shift is the feature that lets you possess other drivers on the road resulting in car crashes that let player escape chases. The plot heavily relies on that ability. Once player jumps into another car, he can overhear the dialogue between driver and unsuspecting passengers.

2.3 Item system in Deus Ex

Deus Ex [10] is a 2000 action role-playing video game developed by Ion Storm.



The can is blank except for the phrase 'PRODUCT PLACEMENT HERE.' It is unclear whether this is a name or an invitation.

COUNT: 10

Figure 2.8 Inventory in Deus Ex

This game inspired me to include items and inventory into my game. The keys, notes with important information or lore, keycodes to doors or safes, lockpicks. Everything combined creates interesting non-linear paths for problem solving. Adding inventory to every character in my project expands possibilities for designing puzzles.

2.4 Game engine

Figure 2.9

When amateur game-developers like me want to create 3D game they mostly choose between 2 engines: Unity and Unreal Engine. Both have its pros and cons. The debate is never-ending since 2015 when Unreal Engine became free. Generally, it is thought that Unity is better for beginners, suits better for mobile and 2D. While Unreal excels at graphics and 3D shooters. Unity uses C# and Unreal uses C++ but has Blueprints Visual scripting system.

The Blueprints Visual Scripting system in Unreal Engine is a complete gameplay scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor. As with many common scripting languages, it is used to define object-oriented classes or objects in the engine. As you use UE4, you'll often find that objects defined using Blueprint are usually referred to as just "Blueprints." [11]

Like most of the students at FEL I started with Unity during “Computer games” course. At that time, I thought it was the perfect tool for creating games, I even tried to implement some ideas from this project. At some point I realized that I should try Unreal Engine as it was getting more recognition from new developers and more tutorials started to appear. Turned out it was a great engine for my purposes. I instantly liked the blueprint system and the editor. It was easier and faster to work with. It took much less time to implement player-switching and inventory than with Unity.

After all, Unreal is just my personal preference because the same game can be created in Unity, CryEngine, Source or any other 3D engine. The version used in the project: 4.22.3.

3 Game design

The player starts in a maze. There are doors and windows connecting the rooms. The goal is to get to the end of the level solving puzzles on the way.

3.1 Mechanics

3.1.1 Movement

Movement is as in any first-person game, WASD to walk. ‘Space’ to jump. Rotate camera with mouse. Hands and legs of the character the player is controlling are visible.

3.1.2 Switching characters

If you look at another character, he becomes highlighted and you can take control of him. Press ‘E’ and after short animation you are in another character’s body.

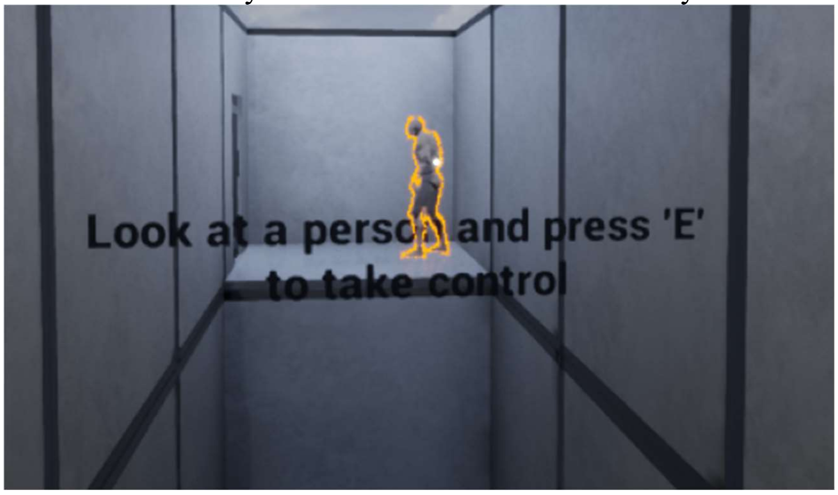


Figure 3.1 Highlighted character

3.1.3 Items and inventory

Every character has his own 4-slot inventory. You can pick items that are close enough with ‘F’ button. If an item is highlighted, you can pick it up. Open inventory with ‘Tab’. Press again to close or click on top-right button. Hover on an item to see its name and description.

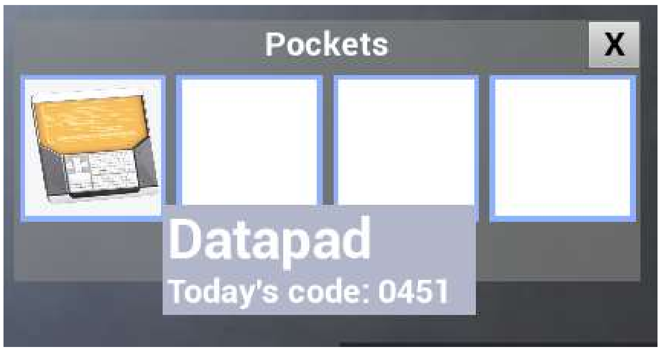


Figure 3.2 Open inventory

There are 3 types of items in the game:

1. Datapads and notes

Contain useful information such as codes to locked doors or general tips.
2. Keycards
   Keycards of different color are used to open the doors with matching color.
3. Jumping boots
   When in inventory, let the character jump twice as high.

Containers have the same inventory as characters. Player can look inside of them by pressing 'F' and take items with right click. It is not possible to put items back.

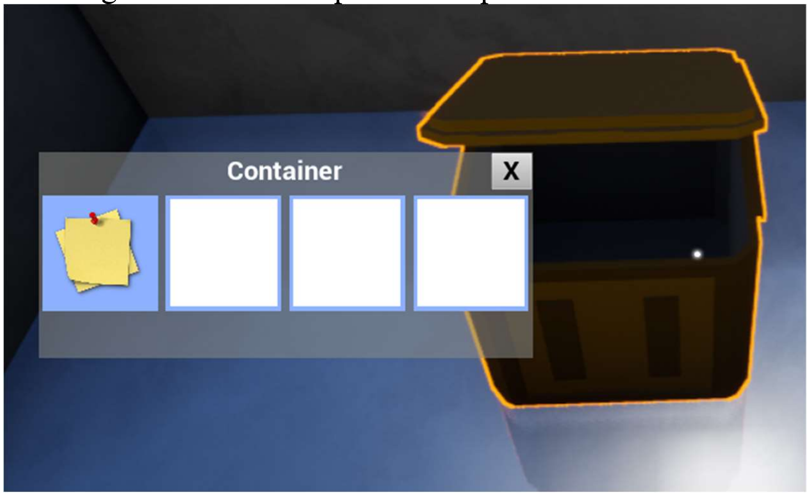


Figure 3.3 Open Container

3.1.4 Doors and Keys

There are 4 types of the doors:

1. Proximity doors: open when player is near. Doors can also be one-sided, blocking the way back.
2. Keycard doors: player should have a keycard in his inventory to open doors with respective color on them. There are 3 keys: red, green and blue. Available keys are also shown at the top-right corner.

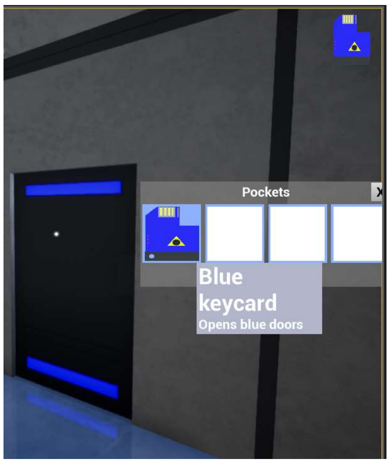


Figure 3.4 Player can see his keycard in open inventory or at the top-right corner.

3. Button doors: The doors open when something is on corresponding button: either character or a cube (movable object, described in 3.1.6). Button doors may require multiple (e.g. three) buttons to be pressed simultaneously to open. When an object is removed from a button, the door closes.

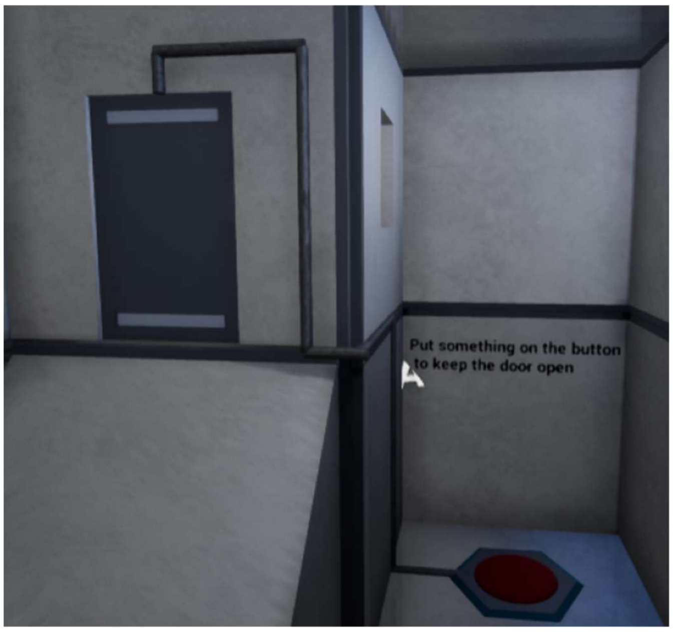


Figure 3.5 Button door

4. Keypad doors: The doors require 4-digit code to be opened. Player left clicks on buttons on a panel to enter password. If the password is incorrect "Error" appears. Click on 'C' to clear the screen of the panel.

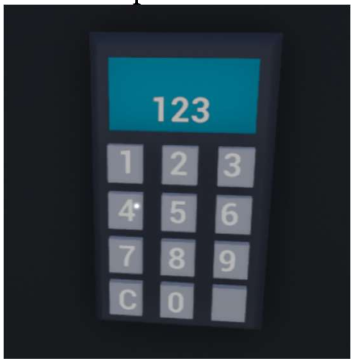


Figure 3.6 Keypad panel

3.1.5 Jumping boots

Jumping boots let you jump two times higher, expanding the platforming potential of the game.

Figure 3.7 Jumping boots

3.1.6 Cubes

Highlighted cubes can be grabbed with ‘F’. They can be used either as a climbing tool or something that you can put on buttons.

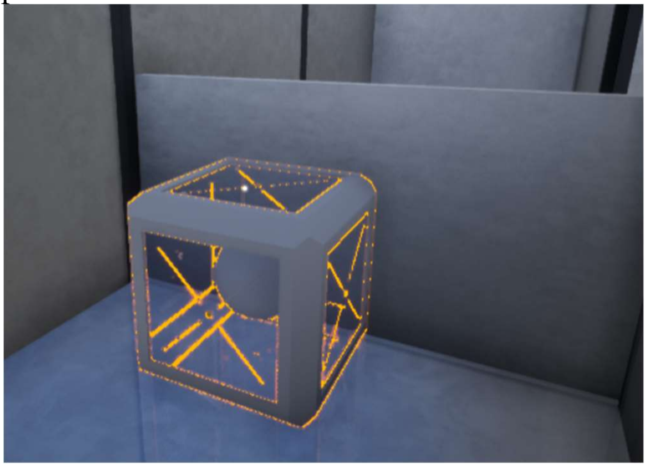


Figure 3.8 Cube can be picked. Player can climb onto it to overcome an obstacle.

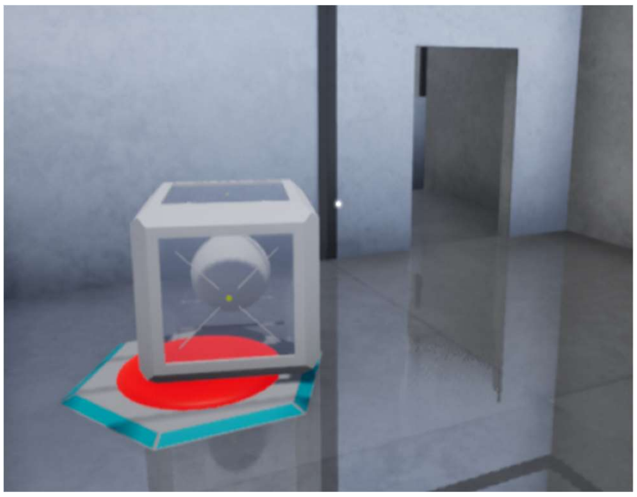


Figure 3.9 Cube on a button opening the door.

3.1.7 Platforms

Platforms can move vertically or horizontally. Activate them by pressing a small button on a wall with ‘F’ (1) or putting something on a corresponding button (2).

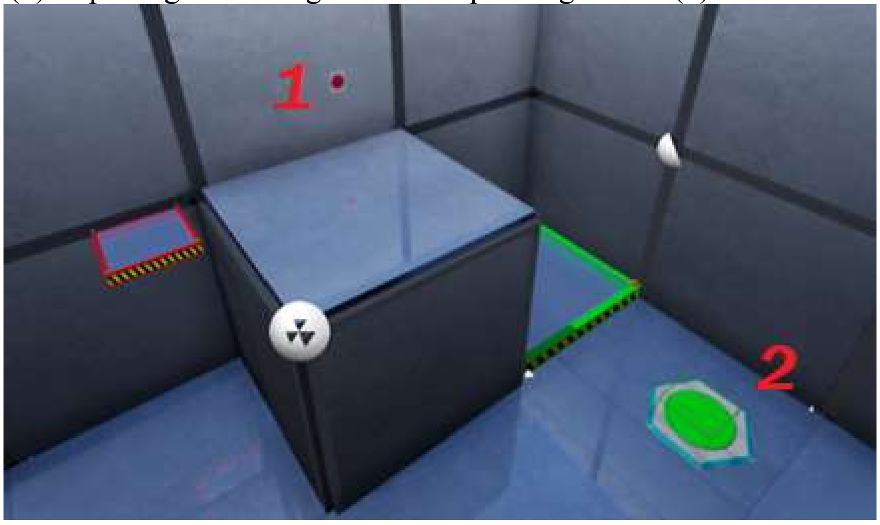
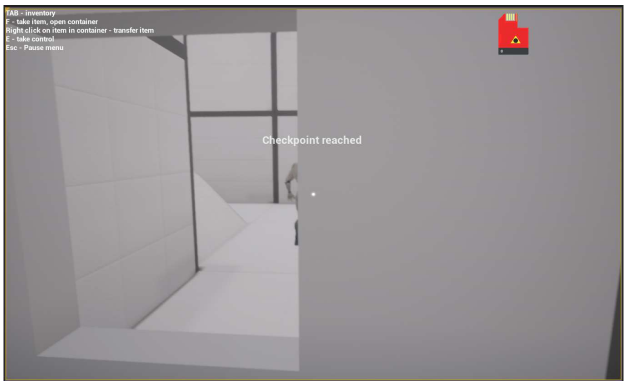


Figure 3.10 Red button activates horizontally moving red platform. Green button activates green elevator.

3.1.8 Checkpoints

If the player enters a checkpoint, the game is saved. The small pop-up message appears on the screen. Next time the game will be loaded, the player will appear at that checkpoint position. There is an option in pause menu to load the last checkpoint or restart the level.



TAB - inventory
F - take item, open container
Right click on item in container - transfer item
E - take control
Esc - Pause menu

Checkpoint reached

Figure 3.11 Player reaches a checkpoint

3.1.9 Menus

The game starts with main menu. Further Figures are self-explanatory.

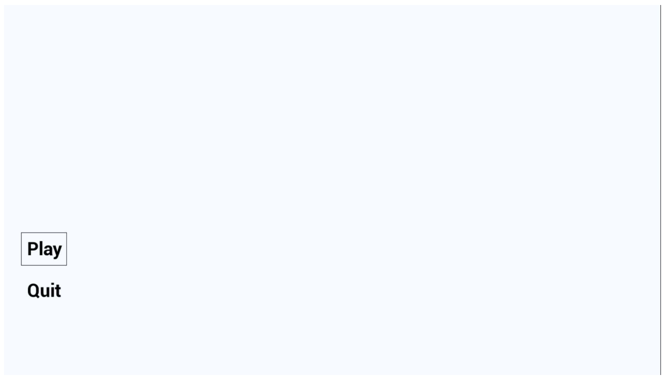


Figure 3.12 Main menu

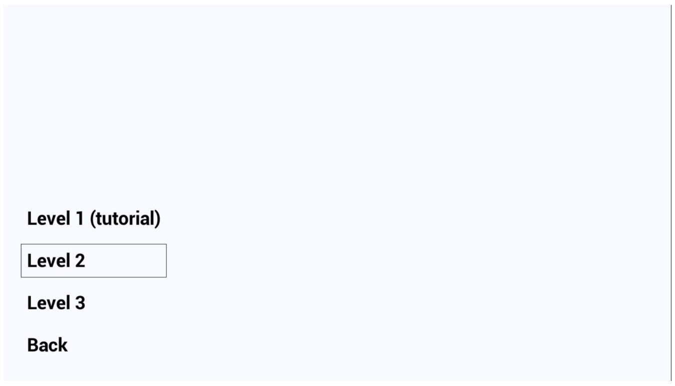


Figure 3.13 Choosing level

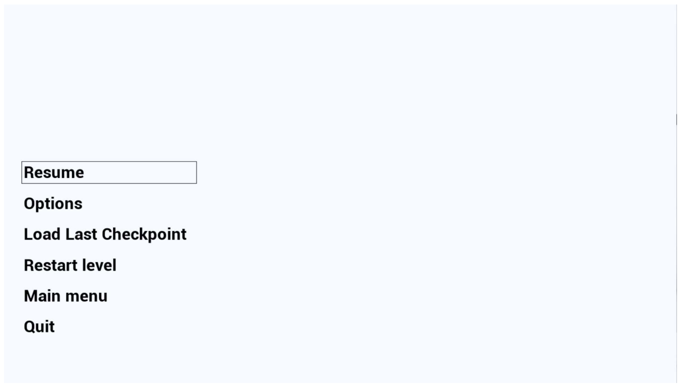


Figure 3.14 Pause menu

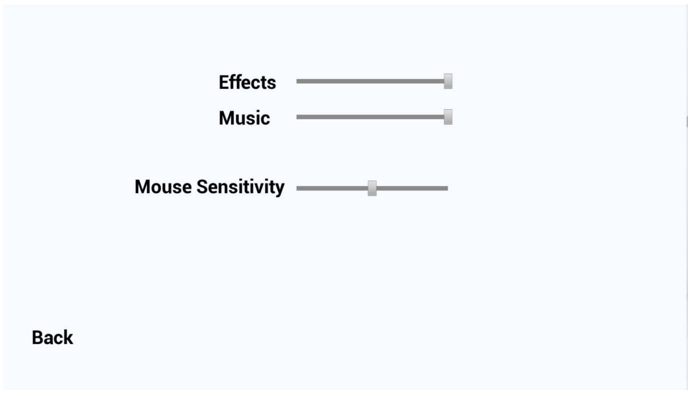


Figure 3.15 Options

3.1.10 Sounds

Following actions have sounds attached. Sounds are taken from freesound.org [12]

- Inventory open/close
- Picking up an item
- Container open/close
- Character switching
- Door open/close
- Platform moving
- Button press

These sounds are from Bartosz Kamiński [13]

- Footsteps
- Cube hitting

3.1.11 Music

Each level has its own ambient soundtrack. The copyright free tracks are taken from freemusicarchive.org [14]

3.2 Puzzles

My general approach to level design is to give the player a new mechanic, explain or show how it works, give him an easy puzzle that can be solved using only that mechanic followed by a harder puzzle that combines new with previously known mechanics. Puzzles alternate in difficulty but get harder in general as the game progresses. The aim here is to get the player into the state often called “flow”. This is the state you enter when you are experiencing absolute concentration on a task. When you’re in absolute control, the challenges that come at you are met precisely by your skills. New mechanics are evenly stretched along the game, so the player would not lose an interest. [15]

Further I describe levels that include several puzzles I designed myself, inspired by portal-like games described in Chapter 1. Expected playtime for the whole game is about 30-60 minutes. For the better visual demonstration, I have removed ceiling somewhere for the screenshots.

3.2.1 Level 1 (tutorial)

Firstly, the player is introduced to picking up items and inventory.

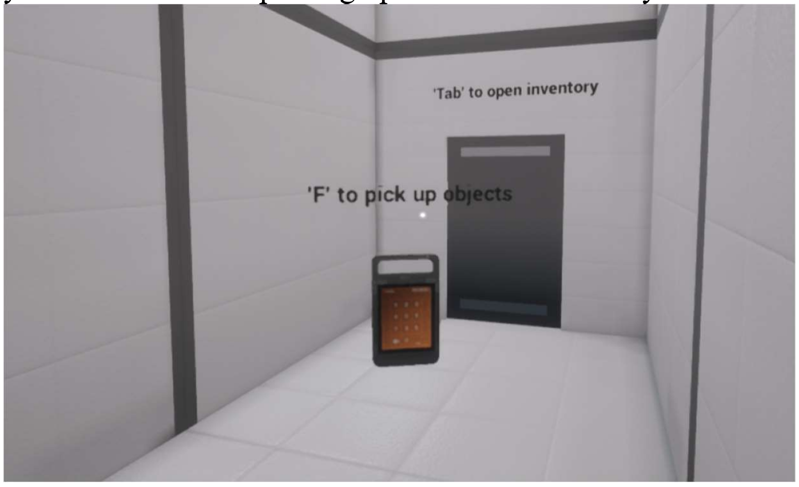


Figure 3.16

Then he needs to jump across the gap.

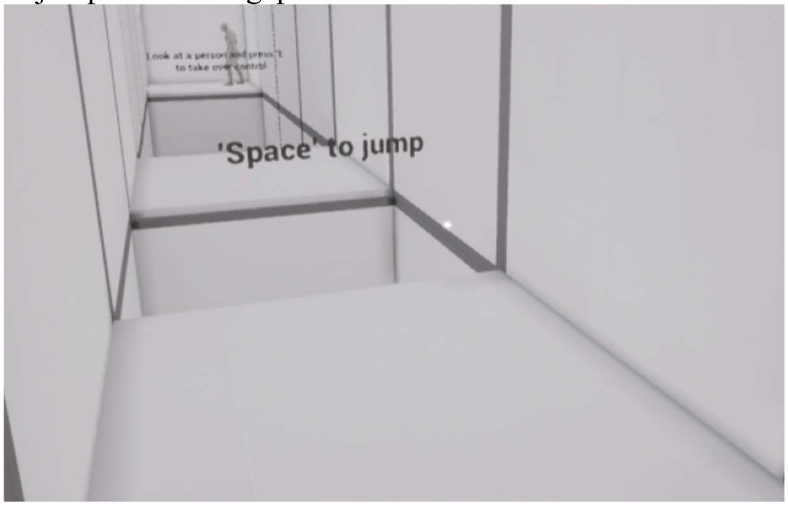


Figure 3.17

Take control of another character and move further. Leave the first one behind. There is no main or side characters, all are equal.

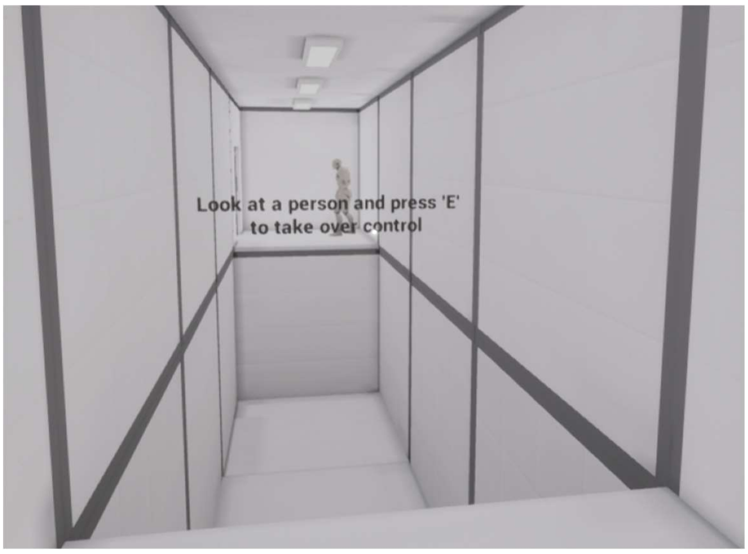


Figure 3.18

Combine previous knowledge and switch while jumping.

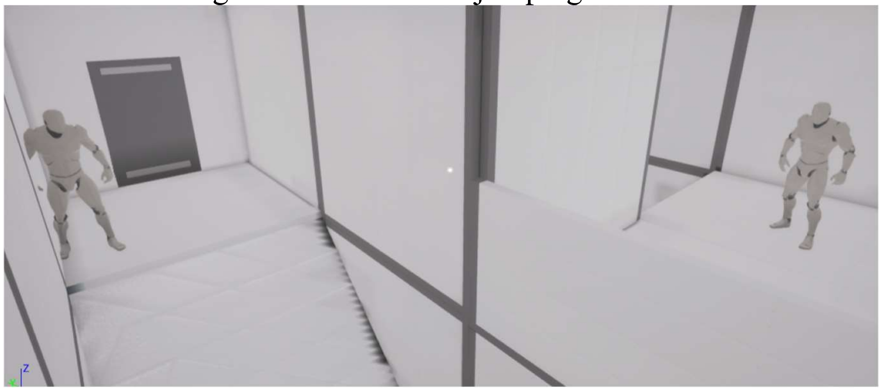


Figure 3.19 The left character is active, he cannot see the other one without jumping

The player encounters a new red door which is locked, a red keycard nearby opens it.

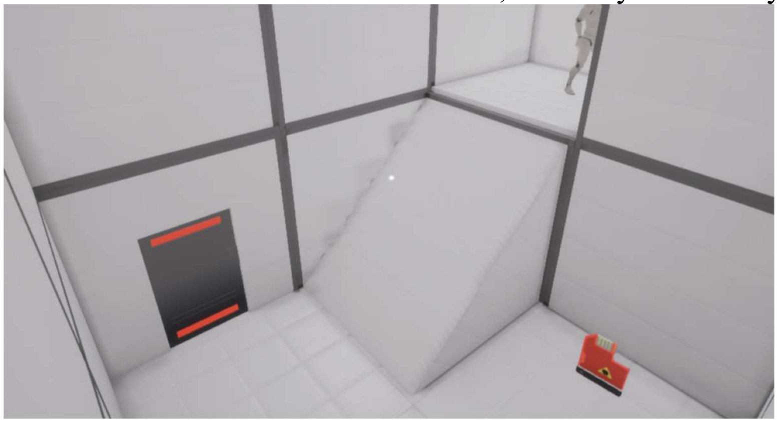


Figure 3.20

Switch to another character (1), pick up the green keycard (2), open the green door (3), switch (4) and open the red door (5).

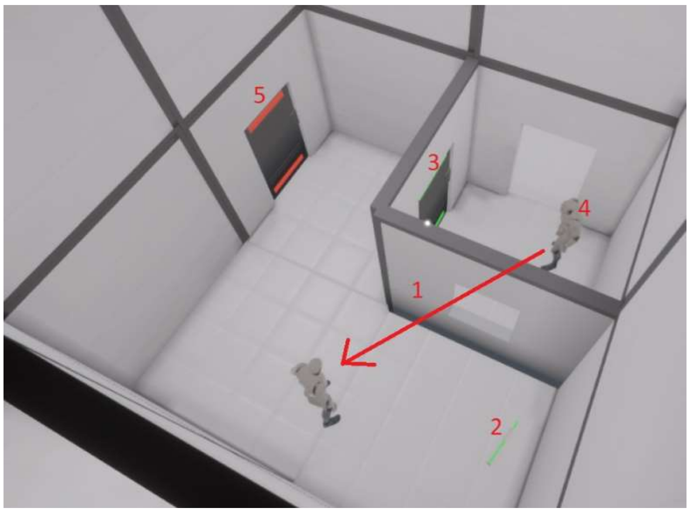


Figure 3.21

Two doors, one needs an unknown code. The second could be opened with a button on the floor. Use one character to stand on the button (1), switch (2), and proceed through the opened door (3). There is another person with a desired code in his pockets. Enter the code and move further (4).

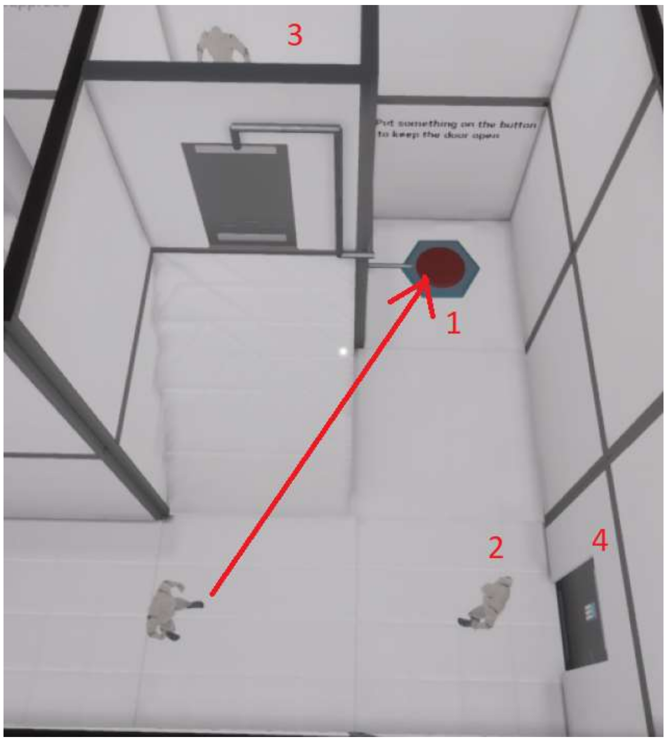


Figure 3.22

This puzzle needs just a little bit of thinking. Open trash bin using ‘F’ as the tip suggests (1). A note in it tells that the code is Larry’s birthday year and he is 30 today. A warning sign indicates that current year is in fact 2076 (2). Subtract 30 from 2076 and get the right code: 2046. Another note in a container tells that the blue keycard for the next door is kept by the person at the desk (3). Open the first door (4) with the code, switch (5) and open the blue door (6).

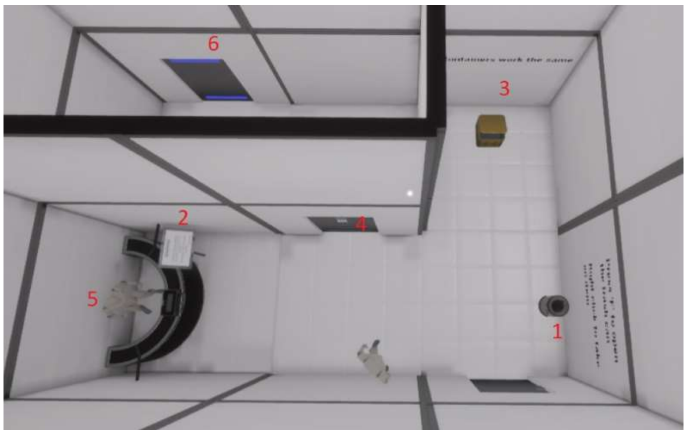


Figure 3.23

Grab this cube with 'F', move it closer to an obstacle and jump over.

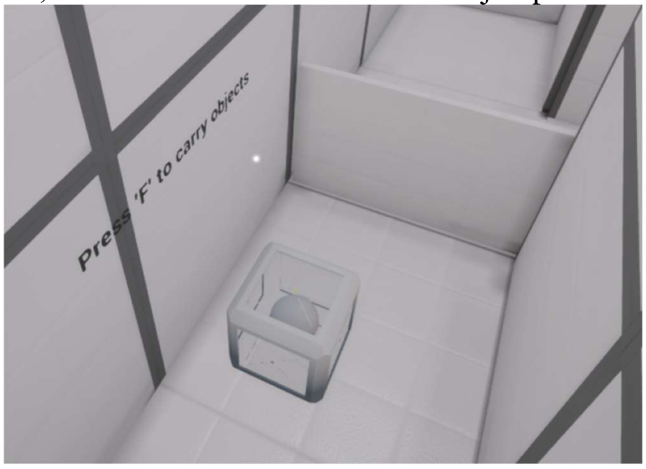


Figure 3.24

This obstacle is a bit higher. Switch to a person upstairs (1) and throw smaller cubes down (2). Switch back and build a ladder (3).

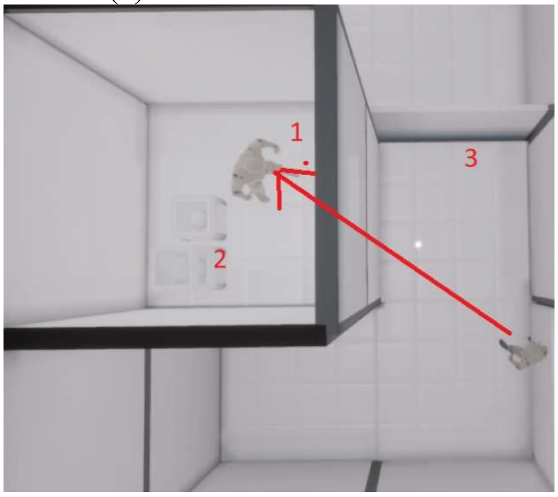


Figure 3.25

Next room shows that not only humans can stand on buttons. Put the cube and a button and pick up the green keycard in the container

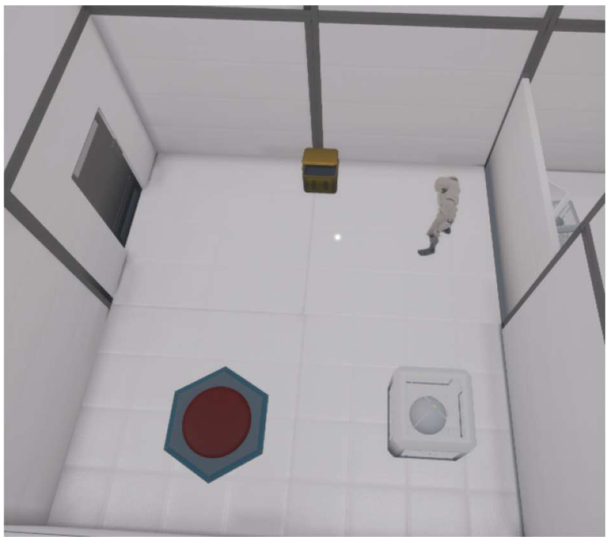


Figure 3.26

The player is separated from the next room by a fence. He can see the green door, another character and a cube on the platform above. The only way is to switch (1), pick up the datapad with the code: 8888 (2), open the door (3), stand on a button (4), watch the cube fall from a blue platform (5), switch back (6), climb over, using the cube and open the green door (7).

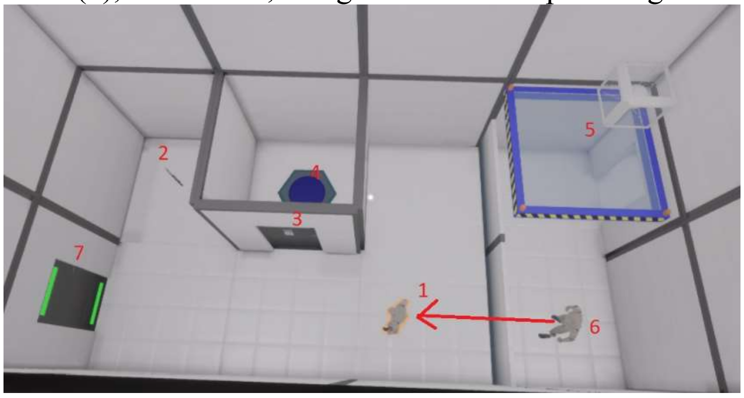


Figure 3.27

Elevator that works by pressing the button, player is supposed to switch while the elevator is moving, leaving button-pressing character behind.

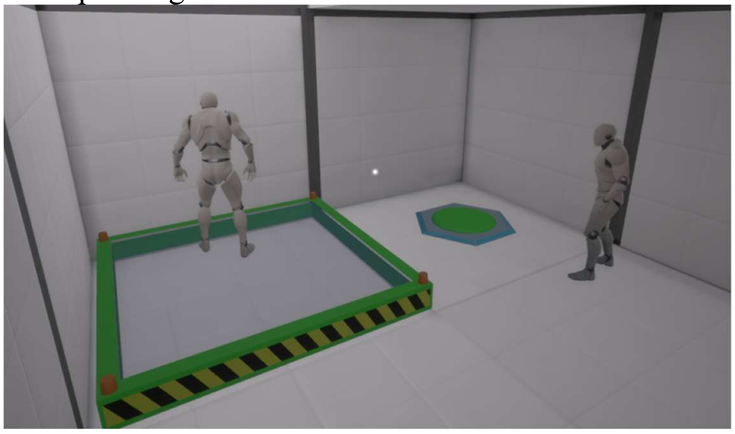


Figure 3.28

New type of button that needs to be pressed once with ‘F’. Player cannot stand on a moving platform and press the button at the same time, he needs a companion, which can be found near the elevator. The platform is moving back and forth. If the player somehow falls from it, he can climb back using the stairs.

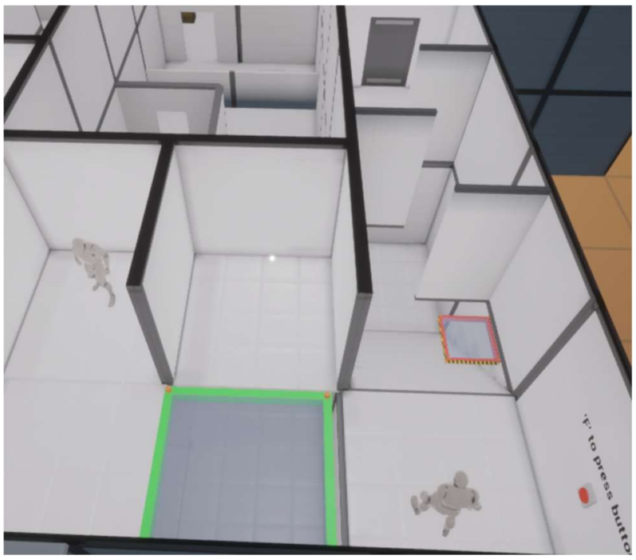


Figure 3.29

This one is much harder than previous rooms. Right sequence of actions is:

1. Get 1 character with a cube on an elevator.
2. Switch to another and press the button on the floor, lifting the character with the cube.
3. Switch back, put the cube on a platform and climb onto it.
4. Use the third character to get up, using the elevator and press the button the wall.
5. Move on the platform to the other side and use the cube to climb over an obstacle.
6. Press the blue button.
7. Pick up the cube that fell from the blue platform.
8. Repeat the steps 1-4 with that new cube and use it to climb over the second obstacle.

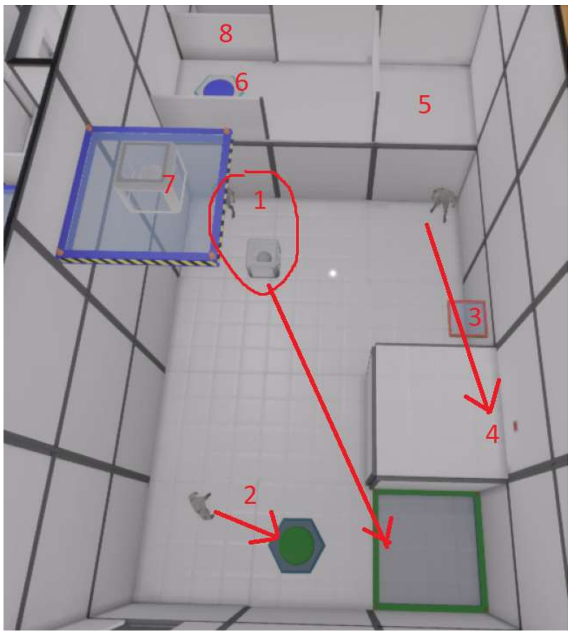


Figure 3.30

There is also another solution. You can get the cubes up from below simply by throwing them. In the first testing there is question on how the players solved this puzzle.

As you can see the first level introduces new mechanics one by one without walls of text. There are only 10 hints, mostly explaining controls. After the player learns a mechanic, he is expected to use it in the next puzzle, often combined with previous known mechanics.

3.2.2 Level 2

The first puzzle is working around the fact that cubes jump up after getting lifted with elevator (1). You need to switch while cube is in the air and grab it (2), then put it on the button to open the door (3). Pick up the red keycard and proceed to the next room (4).

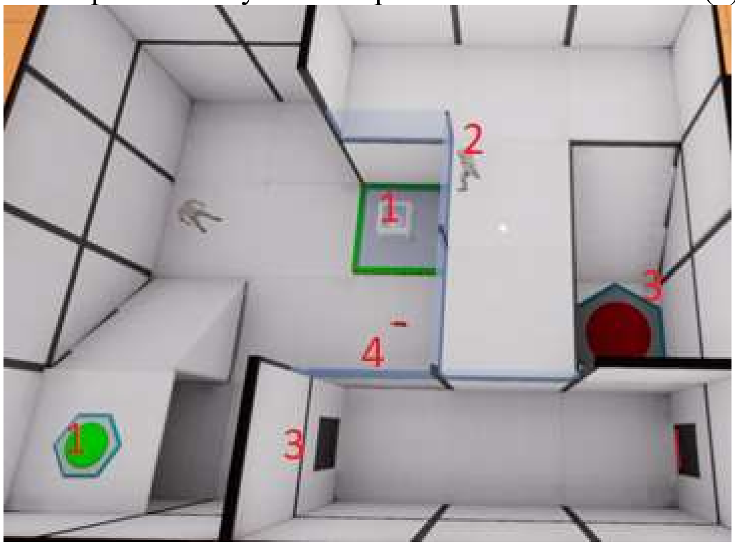


Figure 3.31

The second room requires a lot of cooperation between 2 characters.

1. Character 2 gives the cube to 1 through a window for him to climb over the obstacle.
2. Put character 2 on the platform.
3. Lift him by pressing the button.
4. Get the green keycard.
5. Pick up 2 datapads with clues to opening the door.
6. 2 goes through the green door
7. 1 opens the door for 2

The minimum switch count here is 4 but regular players need at least 2 times more.

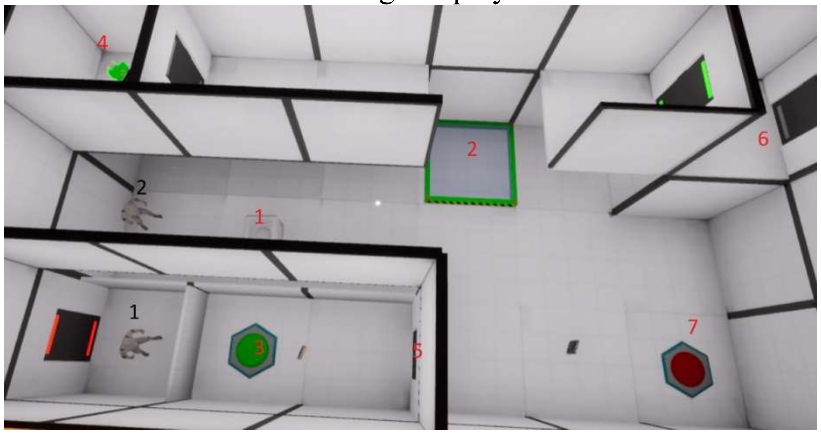


Figure 3.32

The third puzzle introduces the jumping boots. Find them in a container in a second room (1), use their ability to jump higher to get the cube (2) and jump with it back to the first room as the door you came in is one-way. Put the cube on a floor button to keep the other door open (3) where you find a new character with blue keycard (4). Switch to him and walk on the elevator (5). Jump on an elevator-lifting button with the character with boots (6) and use another character to open the blue door (7).

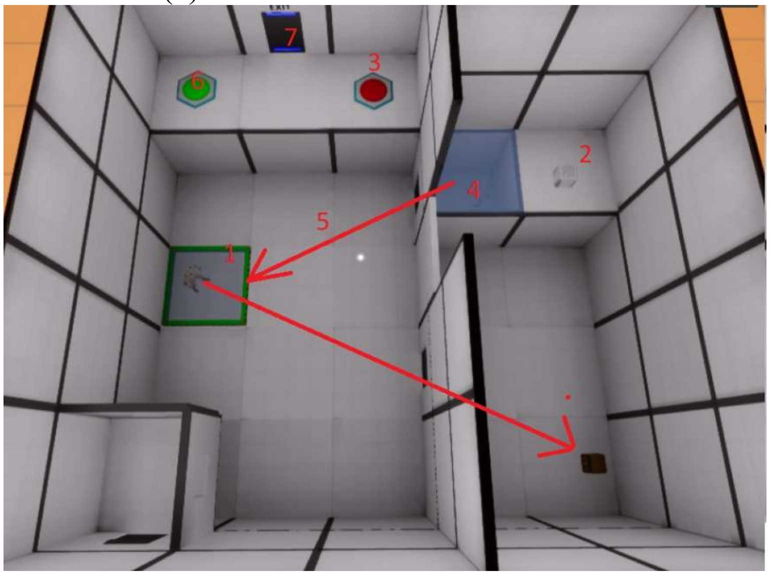


Figure 3.33

3.2.3 Level 3

Level design of level 3 is a bit different from previous. The first thing the player sees is a red door and 2 corridors. Right corridor leads to a room where you need to jump your way up to get the keycard (1). You certainly notice some numbers on the walls. After you open the red door, you see a keypad door and a datapad saying ‘4 digits in ascending order’ (2). 2 of them can be found in the left room, 2 in the right. Digit 9 is a little bit harder to find.

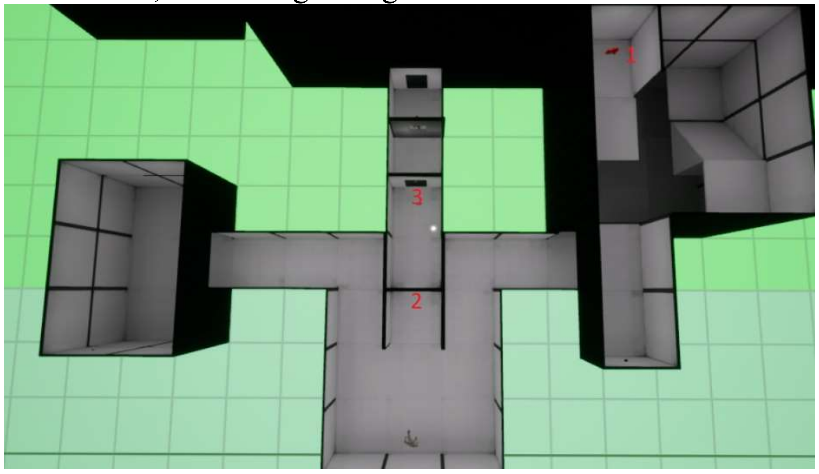


Figure 3.34

After unlocking the keypad door, the player turns up in a big room. One door requires a blue keycard, the other with an ‘Exit’ sign requires 3 buttons to be pressed at once. Switch to the character peeking out the window on the left.

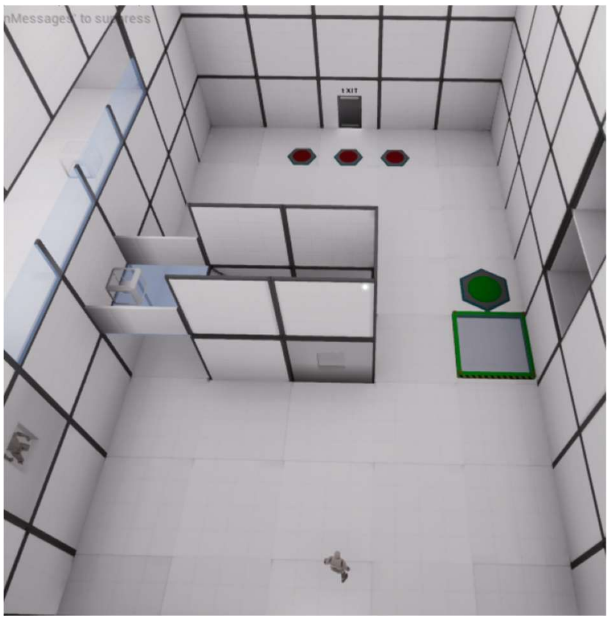


Figure 3.35

Using jumping boots (1) get the cube (2) and put it on a button (3) to open the door (4). Here lies another cube, drop it down and switch back to initial character.

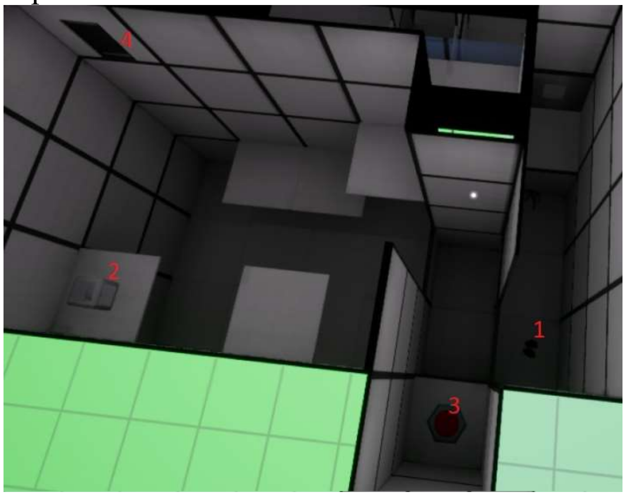


Figure 3.36

As you may notice, one cube and one character is not enough to open the door but enough to lift yourself up with an elevator to get to the right room.

This room is the most complex in the game. Get onto the sliding platform (1), switch and press the according button on the upper balcony (2), switch back. Jump at the end of the platform's route and put the cube (3) on the button (4). This will open the small room nearby (5), where you find a green keycard and a note in a container saying 'Look up'. If you look up, you will notice a code to the door (6). Use the platform (7) as before and open the door (8). Grab the cube and put it on a green button to activate the elevator (9). Lift yourself up and make a big jump from the high platform (10). You will find another cube a blue keycard. Drop the cube down in the main room.

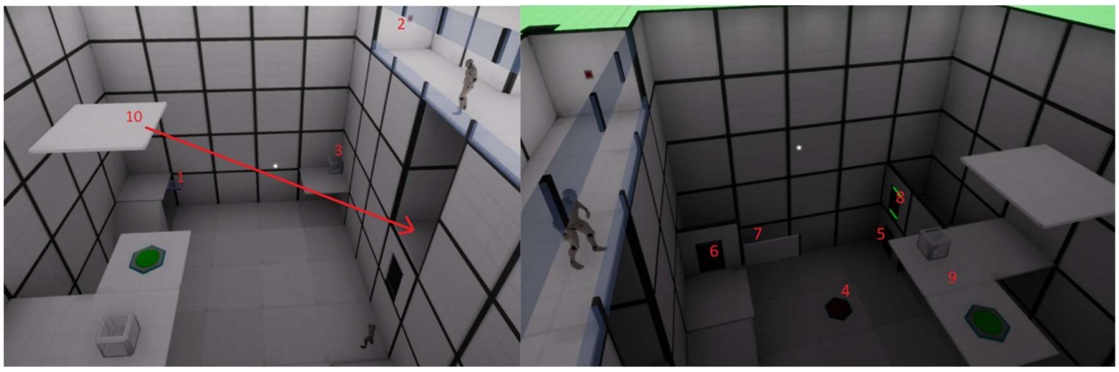


Figure 3.37

Open the blue door and drop down the cube. Now you have all 3 cubes needed to open the exit door. One from left section, that should be on a green button at that time, one from right section and one from middle section.

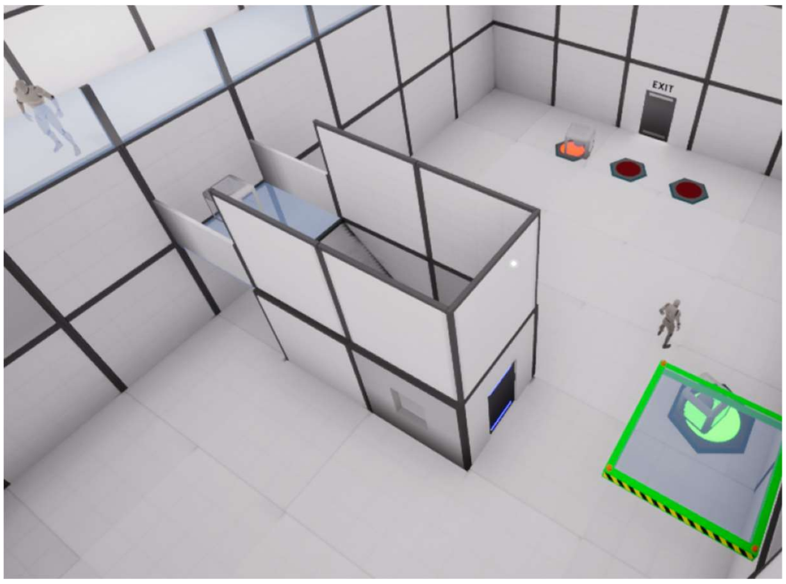


Figure 3.38

The game finishes after entering the room.

4 Implementation

4.1 Game mechanics

4.1.1 Base Character

Every character in the game is the same. All are equally controllable, have the same stats and have an inventory. As a base for playable character I have chosen ThirdPersonCharacter blueprint which comes with Unreal Engine. It has model, animations and movement already set up and ready to be modified. On the other hand, FirstPersonCharacter does not have own model except only arms and is more suitable for first-person shooters, which is unacceptable because I have multiple playable characters that should always be seen by other characters.

The character (actor) has several components attached to him.

- **ThirdPersonCharacter** – Base class (certain nodes and logic will be described later)

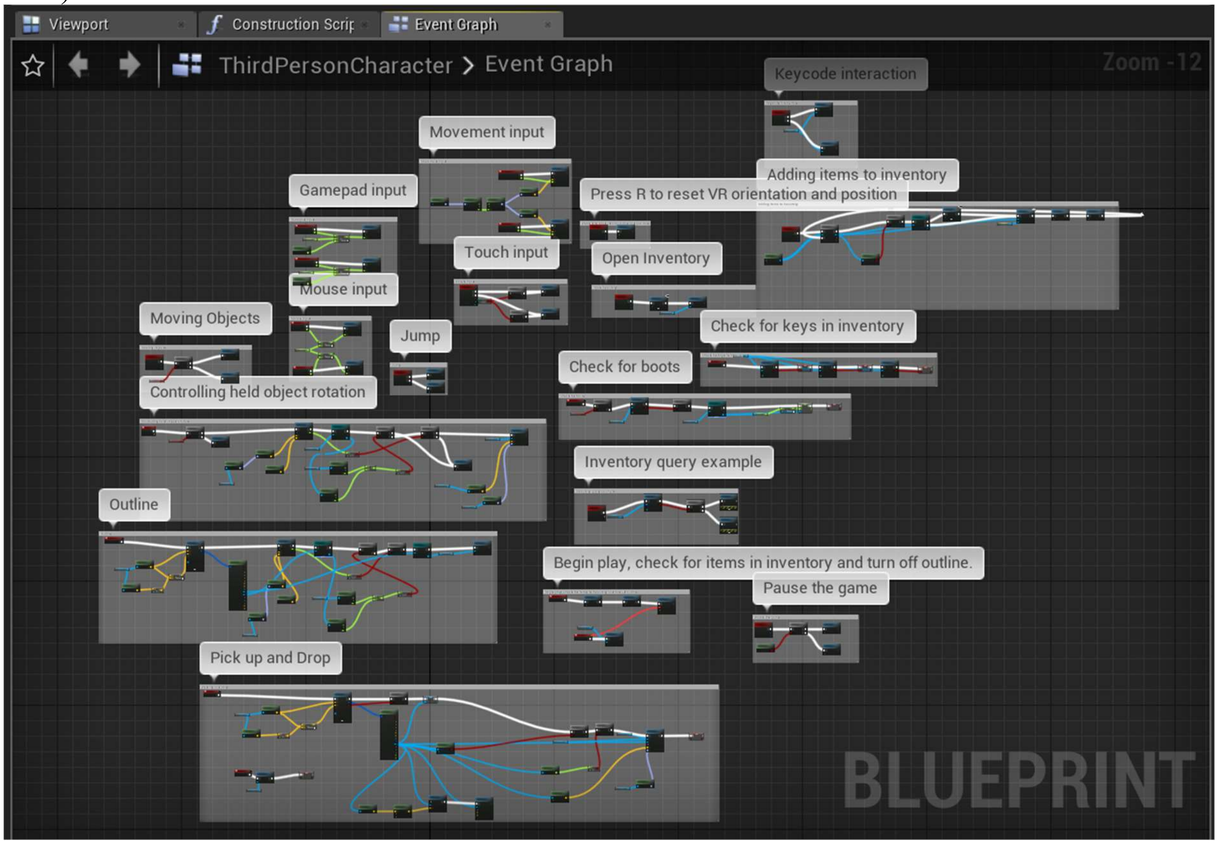
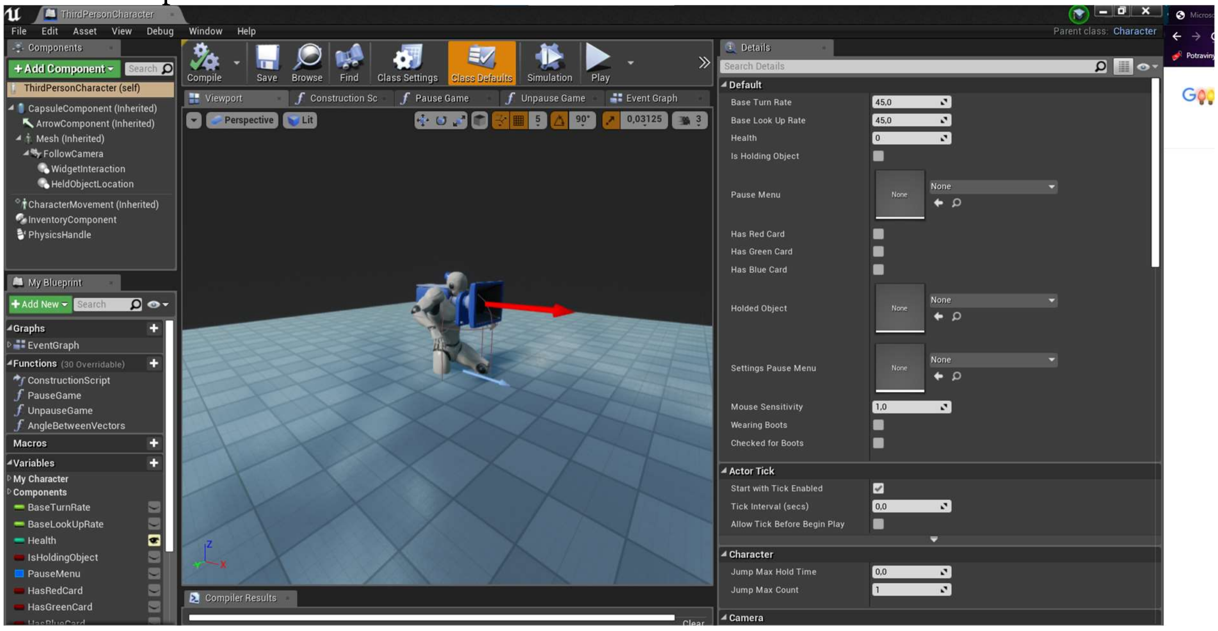


Figure 4.1 ThirdPersonCharacter blueprint

- CapsuleComponent – A capsule used for simple collision.
- ArrowComponent – Useful for indicating which way an object is facing
- Mesh – Character model to draw, uses animations from ThirdPerson_AnimBP_C (sample animation blueprint).
- **FollowCamera** – Represent a camera viewpoint and settings, such as projection type, field of view, and post-process overrides. Parent is Mesh.

- **WidgetInteraction(new)** – This is a component to allow interaction with the Widget Component. This class allows you to simulate a sort of laser pointer device, when it hovers over widgets it will send the basic signals to show as if the mouse was moving on top of it. Here used for interaction with panels on doors which require a code to be opened. Parent is FollowCamera.
- **HeldObjectLocation(new)** – Location where the picked-up cube goes (about 1 meter from the characters head). Parent is FollowCamera.
- **CharacterMovement** – handles movement logic for the associated Character owner.
- **InventoryComponent(new)** – Blueprint class, logic for inventory of a character, what items the player has, how to add a new item, arrangement of items etc. (described later)
- **PhysicsHandle(new)** – Utility object for moving physics objects around (cubes).

Components in **bold** are modified or new.



Default	
Base Turn Rate	45,0
Base Look Up Rate	45,0
Health	0
Is Holding Object	☐
Pause Menu	None
Has Red Card	☐
Has Green Card	☐
Has Blue Card	☐
Holded Object	None
Settings Pause Menu	None
Mouse Sensitivity	1,0
Wearing Boots	☐
Checked for Boots	☐
Actor Tick	
Start with Tick Enabled	✓
Tick Interval (secs)	0,0
Allow Tick Before Begin Play	☐
Character	
Jump Max Hold Time	0,0
Jump Max Count	1
Camera	

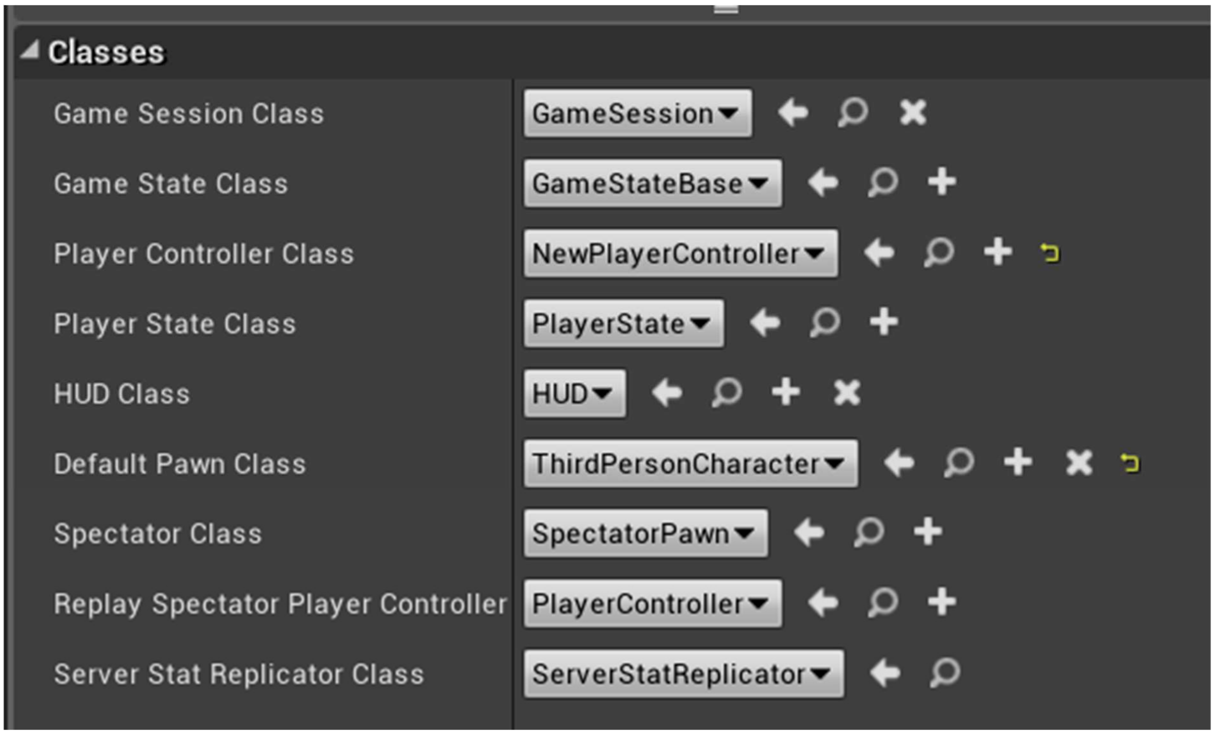
Figure 4.2 ThirdPersonCharacter viewport

Modifications of a base character:

- Moved the camera near the head so the player can see his arms and legs.
- Camera's Use Pawn Control Rotation set to true and Lock to Hmd set to false, so the camera behaves as expected in first person.
- The first main character has Auto Possess Player set to Player 0, other characters have Disabled. Player 0 is the character to start with.
- In Mesh: Collision Presets set to No Collision, it caused the physical objects the player stood on to move.
- In Character Movement: Run physics with no controller set to true, so the uncontrolled characters would fall down if appeared to be in air. For example, switching mid-air. Jump Z Velocity increased from 420 to 450 and Air Control increased from 0.05 to 0.1 for more natural jumping. Use Controller Desired Rotation and Orient Rotation to Movement set to true, in order to move the mesh to where the camera is looking. Reduced Max Step Height for better interaction with climbable cubes from 45 to 40.

4.1.2 Game mode

Every game in Unreal Engine must have a class called Game Mode. It tells what default classes to use.



Classes	
Game Session Class	GameSession
Game State Class	GameStateBase
Player Controller Class	NewPlayerController
Player State Class	PlayerState
HUD Class	HUD
Default Pawn Class	ThirdPersonCharacter
Spectator Class	SpectatorPawn
Replay Spectator Player Controller	PlayerController
Server Stat Replicator Class	ServerStatReplicator

Figure 4.3 GameMode

For my game I needed to change Player Controller Class to NewPlayerController and Default Pawn Class to ThirdPersonCharacter.

4.1.3 Character switching

Character switching works in 2 parts:

1. In NewPlayerController blueprint: Every EventTick (every tick of game loop) LineTraceByChannel function is called shooting rays forward from camera position (actor 1).

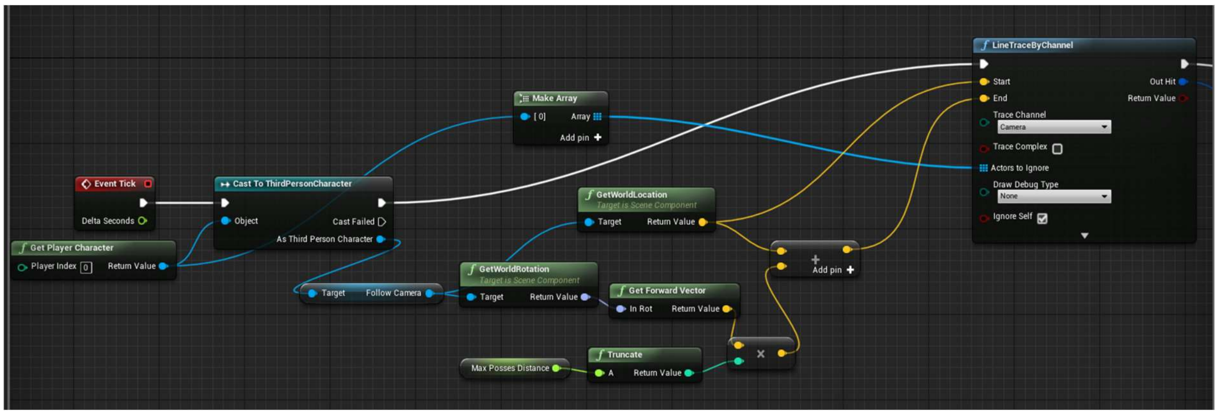


Figure 4.4 Character switching part 1.1

If it hits another character (actor 2) that can be switched to, this character is highlighted and actor 1's variable CanPossess is set to true, otherwise it is false.

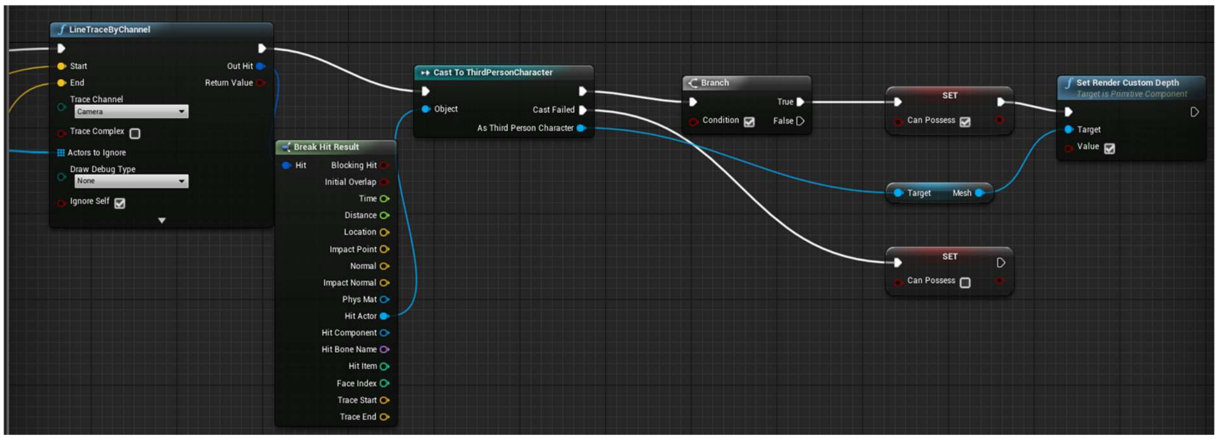


Figure 4.5 Character switching part 1.2

2. If the player presses ‘E’ key, and is not switching to other character at moment, the switch animation occurs. Small sound effect is played,

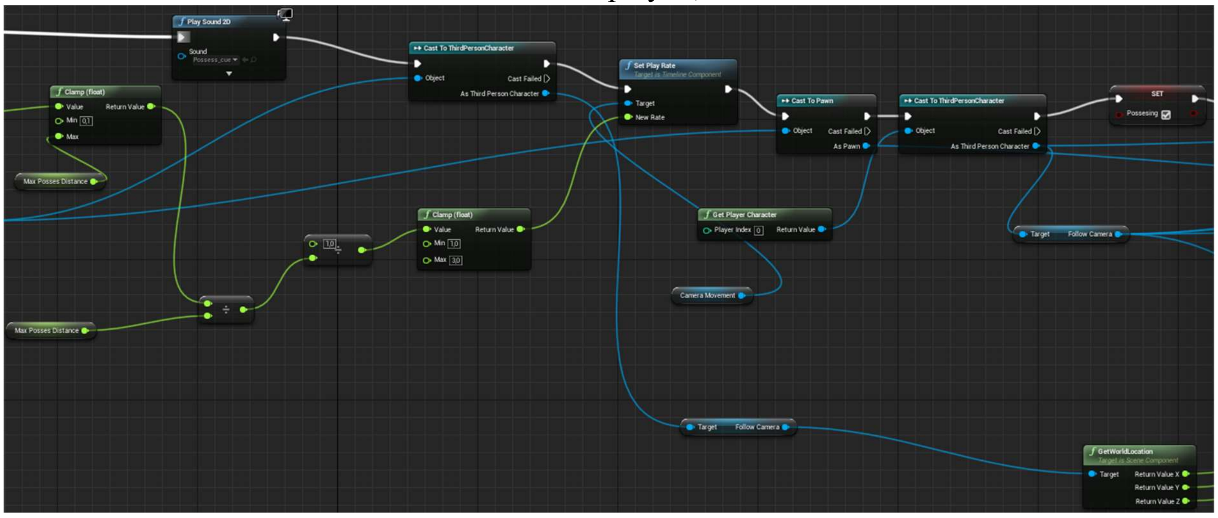


Figure 4.6 Character switching part 2.1

the time slows down to 10% with Set Global Time Dilation set to 0.1, input is disabled, camera flies toward the desired character's head for some time, depending on the distance,

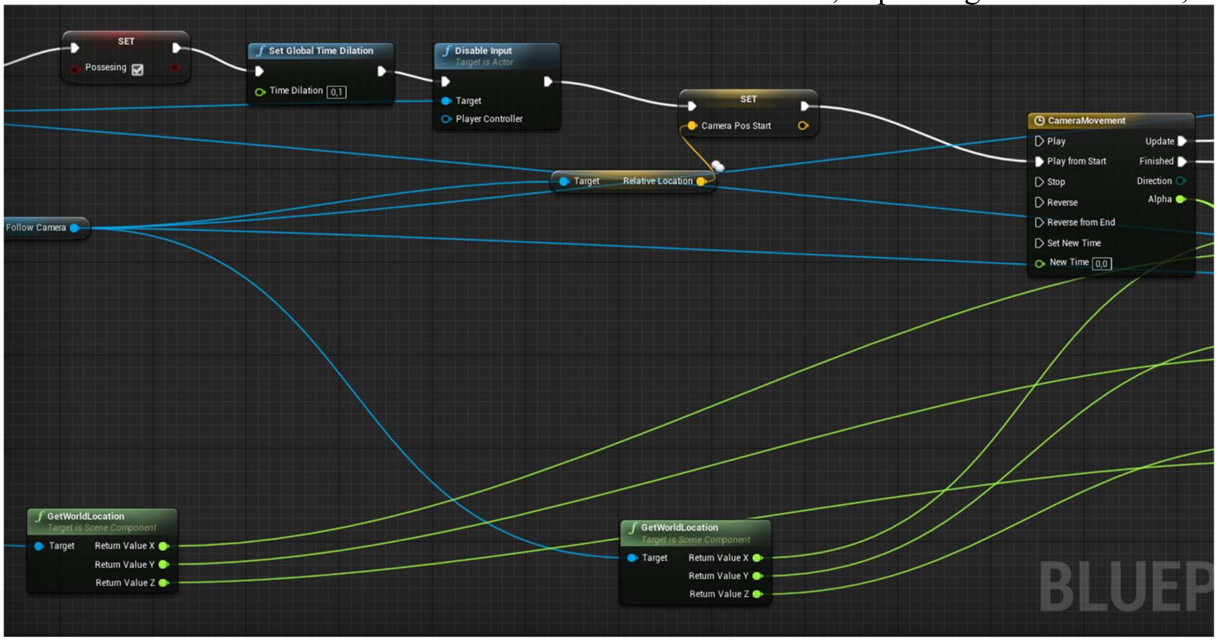


Figure 4.7 Character switching part 2.2

then the player possesses actor 2 with built-in function Possess and the actor 1's camera goes to the default position.

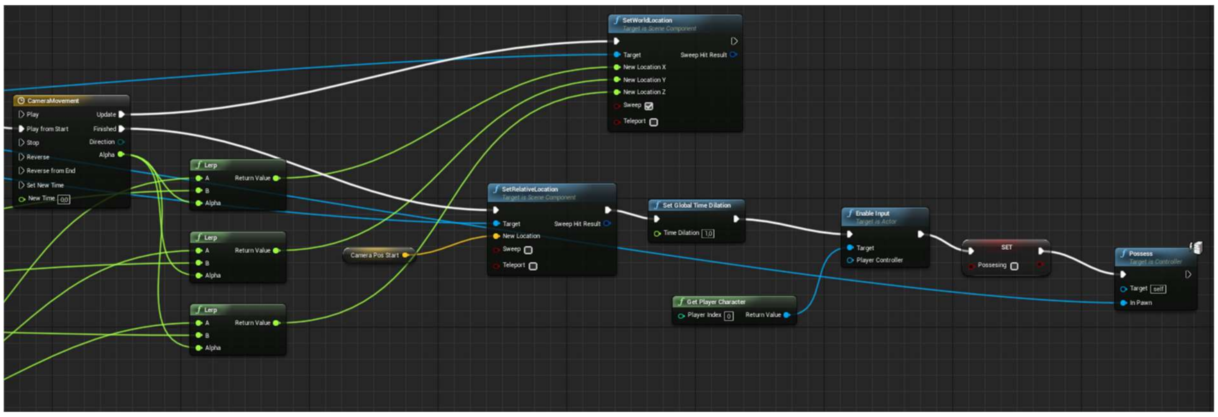


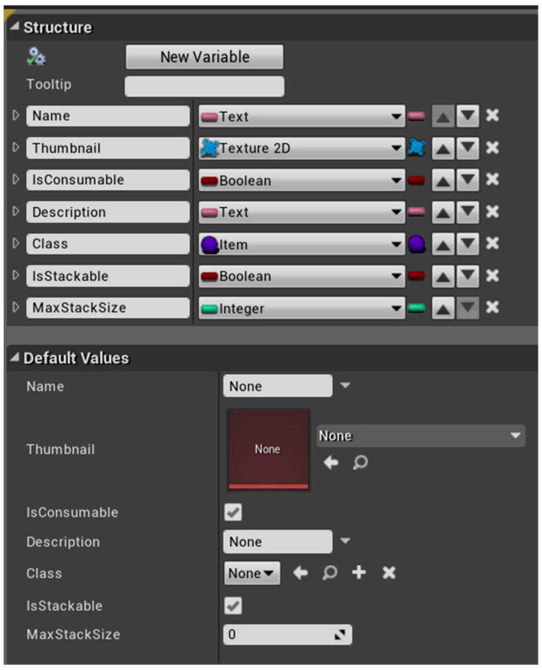
Figure 4.8 Character switching part 2.3

4.1.4 Inventory system

Inventory system was inspired by Ryan Laley tutorial on YouTube [16]. I have added or changed several features as it was too basic: adding items to characters before game starts, changed picking up behavior and container interaction and added sounds.

Item

Base class for all items is Item. Every item should be a child of this class. It has Sphere Collision and local variable of custom type ItemStructure.


Structure

New Variable

Tooltip

Name	Text	
Thumbnail	Texture 2D	
IsConsumable	Boolean	
Description	Text	
Class	Item	
IsStackable	Boolean	
MaxStackSize	Integer	

Default Values

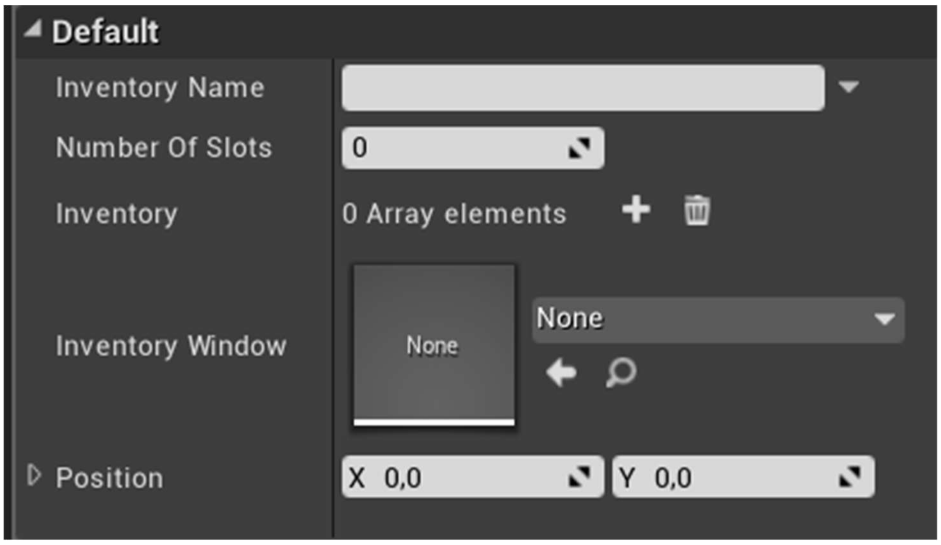
Name	None
Thumbnail	None
IsConsumable	✓
Description	None
Class	None
IsStackable	✓
MaxStackSize	0

Figure 4.9 ItemStructure

Inventory

ThirdPersonCharacter and containers implement InventoryComponent. Attributes are:

1. InventoryName ("pockets" for humans)
2. Number of slots
3. Inventory (array of ItemStructure)
4. InventoryWindow (UI widget)
5. Position (for InventoryWindow, pockets on the right, container on the left)


Default

Inventory Name	
Number Of Slots	0
Inventory	0 Array elements
Inventory Window	None
Position	X 0,0 Y 0,0

Figure 4.10 InventoryComponent

Inventory functions:

1. PrepareInventory: Some characters (or containers) have items by default at the start of the game.

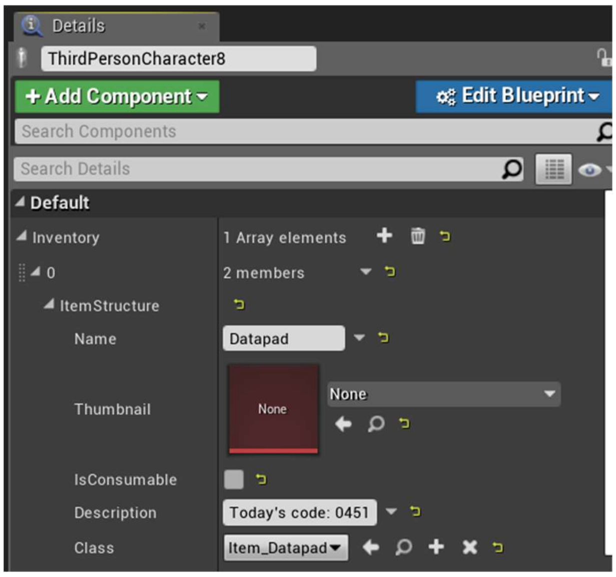

Details
ThirdPersonCharacter8
+ Add Component
Edit Blueprint
Search Components
Search Details
Default
Inventory 1 Array elements
0 2 members
ItemStructure
Name Datapad
Thumbnail None None
IsConsumable
Description Today's code: 0451
Class Item_Datapad

Figure 4.11 This character has datapad in his pocket

The function sets default properties of an item based on class (Item_Datapad here, will get thumbnail when the game starts), except for name and description, those can be changed.

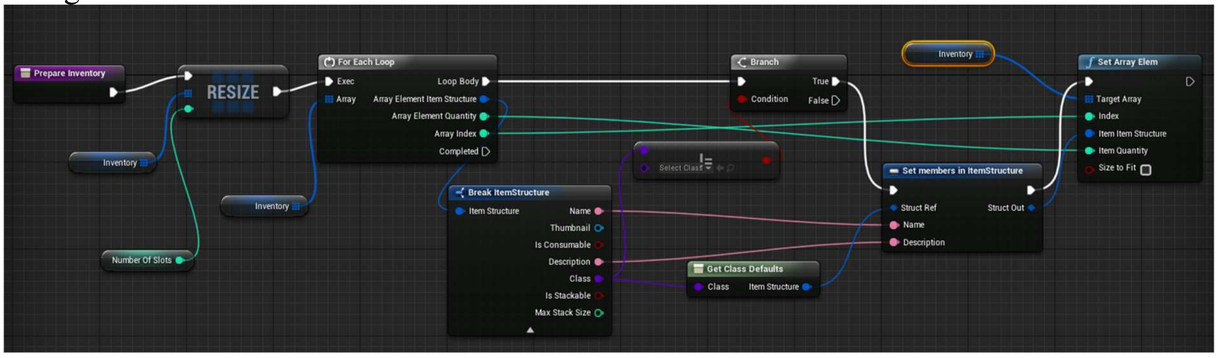


Figure 4.12 Preparing Inventory

2. CreateStack and AddStack

At first, I had items that were stackable but later in the development I got rid of them because I found no use of them in the world of my game. So, these functions can be used but are redundant for now.

3. AddToInventory
The function is called when the player presses 'Interact button' ('F' button) near objects that have 'Interact' interface.

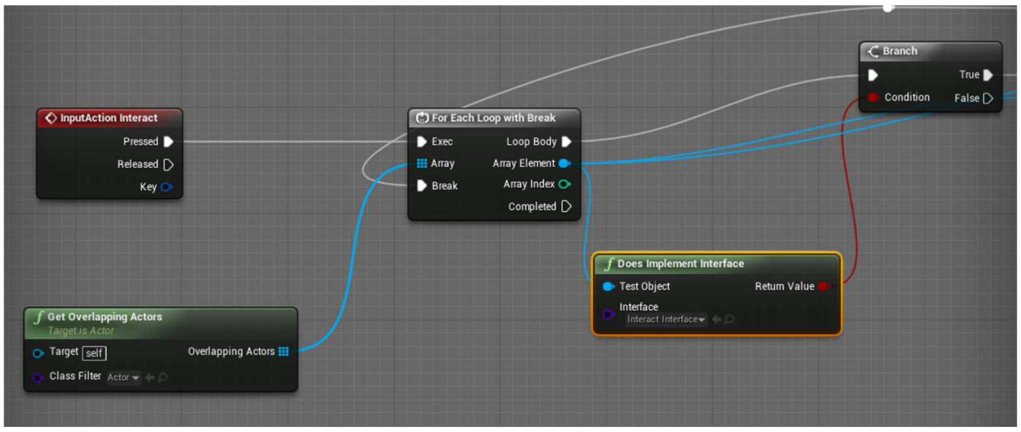


Figure 4.13 Picking up items (in ThirdPersonCharacter)

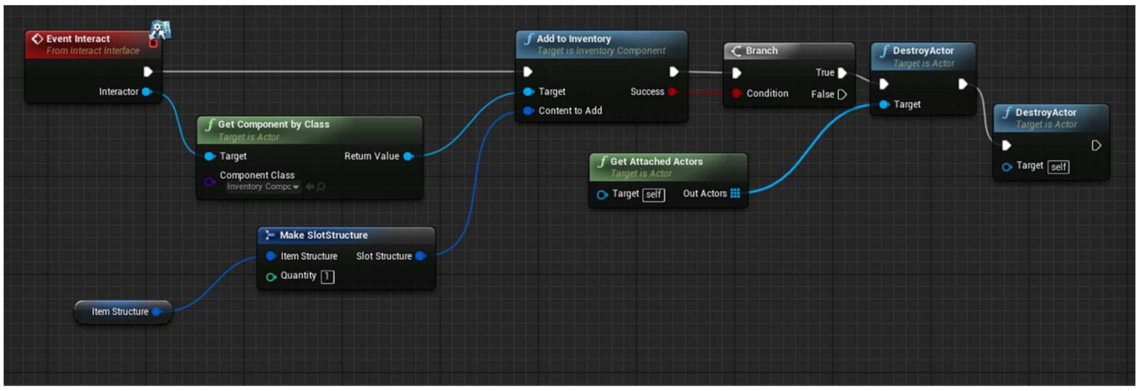


Figure 4.14 Interact Event (in Item parent class with Interact interface)

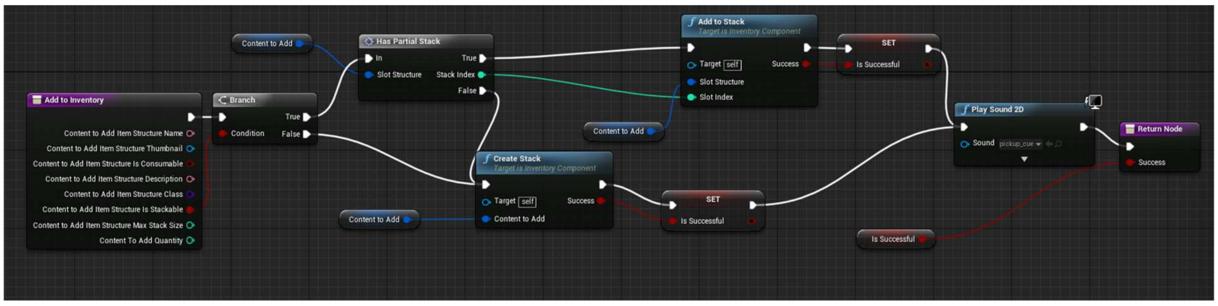


Figure 4.15 AddToInventory (in InventoryComponent)

4. ToggleInventory

ToggleInventory function creates Inventory Window Widget in right (left for containers) part of the screen showing available items (‘Tab’). Closes window by pressing again.

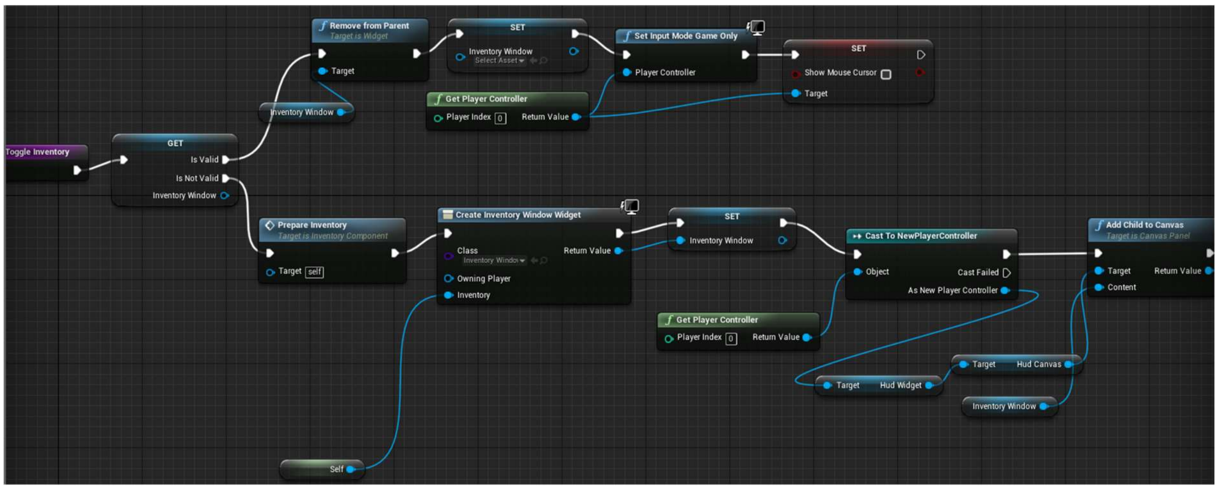


Figure 4.16 ToggleInventory (part 1)

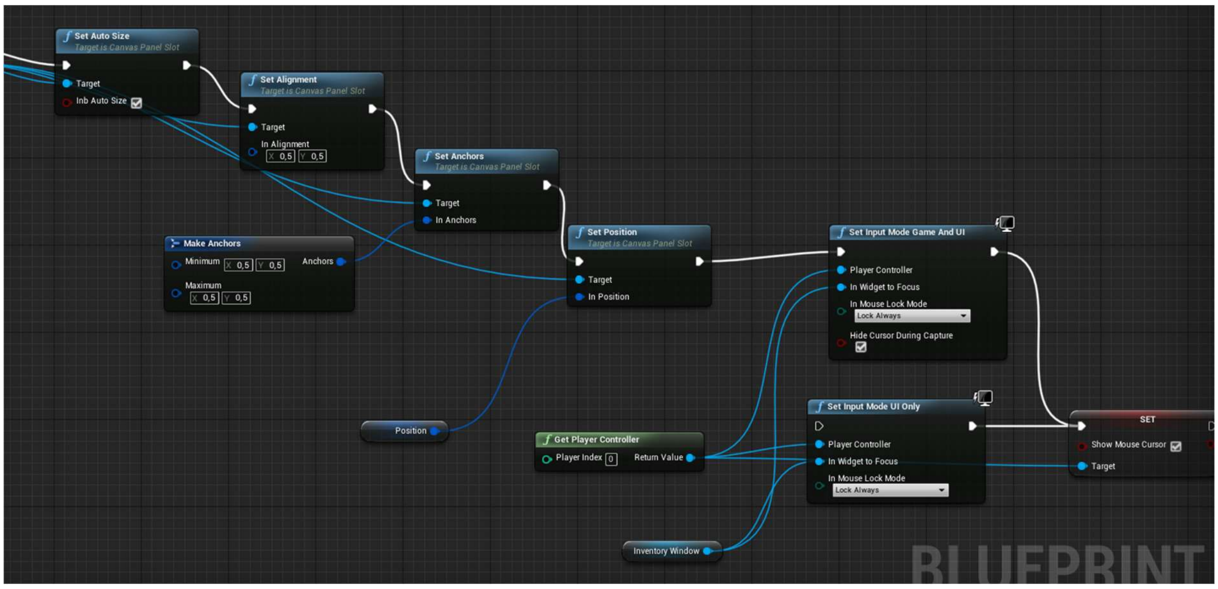


Figure 4.17 ToggleInventory (part 2)

5. InventoryQuery

The function is called whenever there is a need to find an object of certain class in inventory (such as keycards).

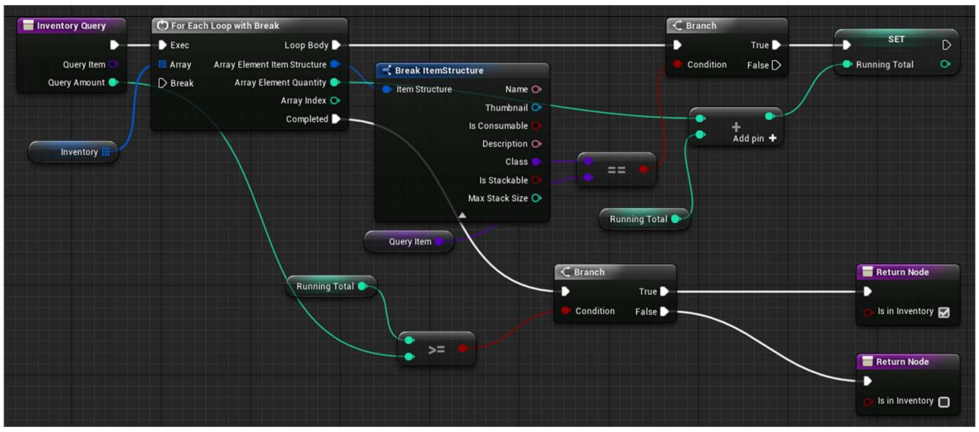


Figure 4.18 InventoryQuery

Inventory UI:

1. HUD (only has Canvas Panel)
2. InventorySlot (small square with item thumbnail)
3. InventoryTooltip (showed when mouse is on InventorySlot, consists of Item Name and Description)
4. InventoryWindow (arrangement of InventorySlots according to NumberOfSlots, can be opened/closed by pressing 'Tab' or clicking on small button in top-right corner, added to HUD)

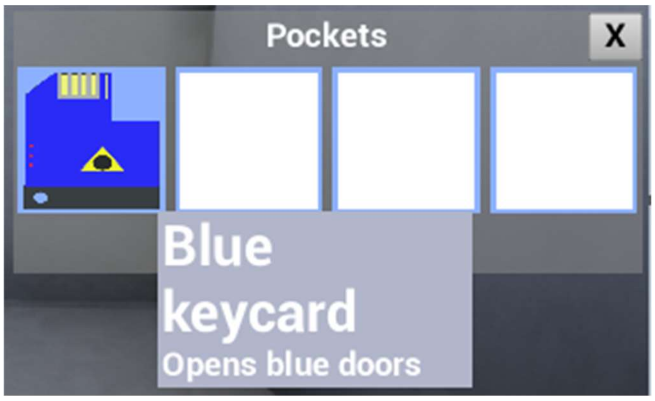


Figure 4.19 Inventory UI

4.1.5 Doors

I have implemented 4 types of doors (proximity, keycard, keypad and button). In Figure 4.20 I show blueprint for the proximity door. When the player enters a trigger box, a sound effect is played and the door slides to the side. When the player leaves the trigger box, the animation is played in reverse and the door is closing.

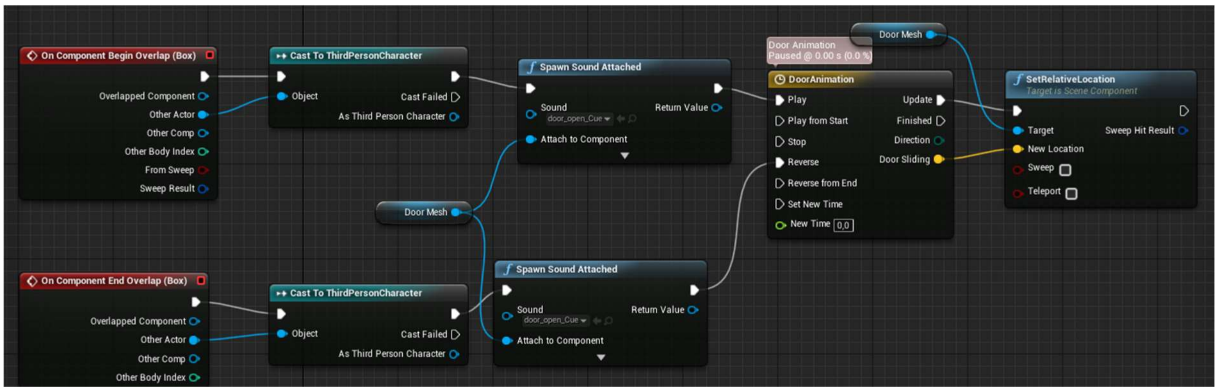


Figure 4.20 Proximity Door (Box component is a trigger near door)

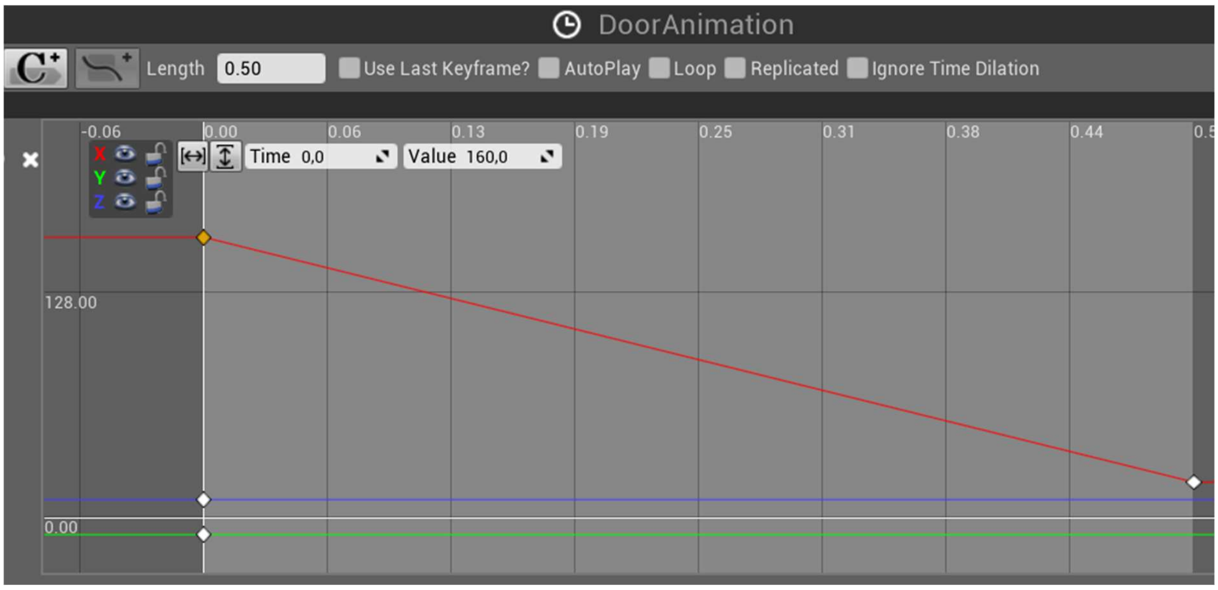


Figure 4.21 Inside DoorAnimation, the X position of the door is changed over time

Button door works the same, but the trigger box is located on the button and also works with cubes.

Keycard doors check if the player has a key in his inventory using InventoryQuery function.

Keypad doors have a small panel attached to it. It consists of a mesh and Keypad_UI widget. When the player is near enough, he can click on buttons on the widget to enter the code.

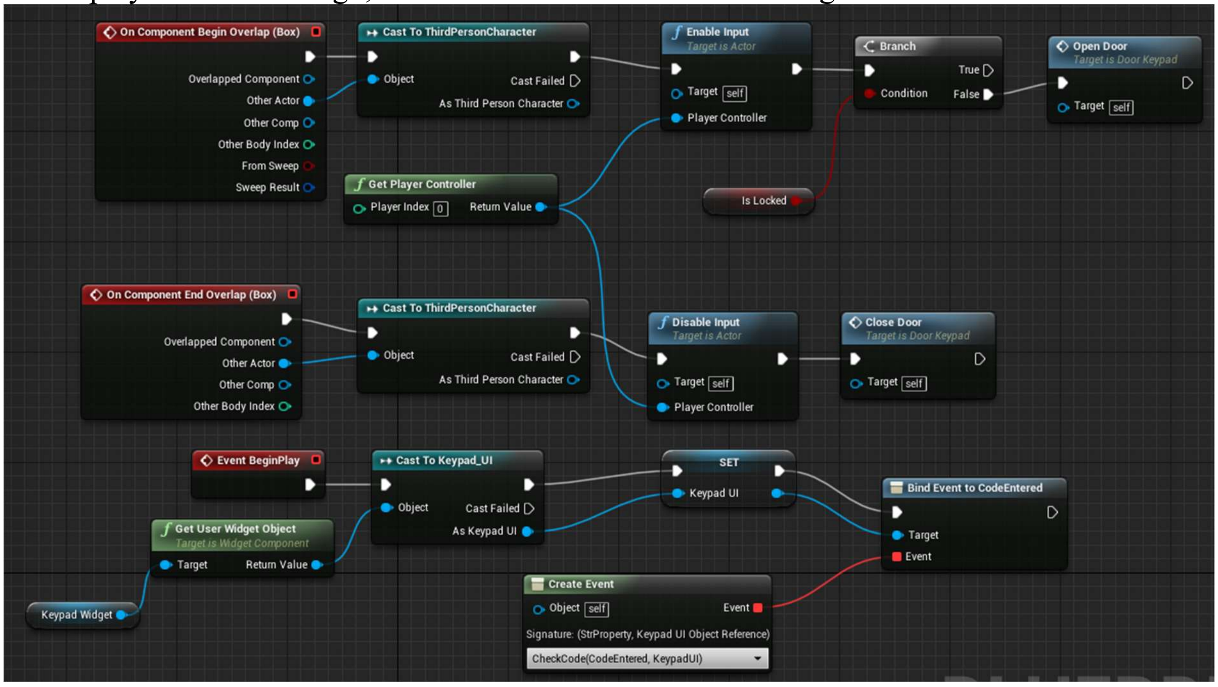

On Component Begin Overlap (Box)
Cast To ThirdPersonCharacter
Enable Input
Branch
Open Door
Get Player Controller
Is Locked
On Component End Overlap (Box)
Cast To ThirdPersonCharacter
Disable Input
Close Door
Event BeginPlay
Cast To Keypad_UI
SET
Bind Event to CodeEntered
Get User Widget Object
Keypad Widget
Create Event
CheckCode(CodeEntered, KeypadUI)

Figure 4.22 Keypad door

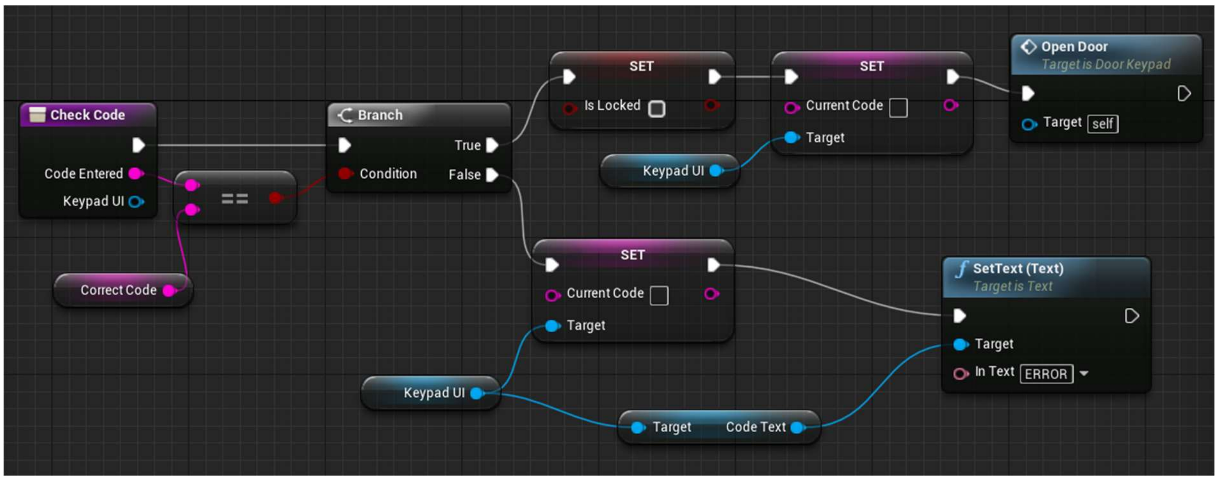


Figure 4.23 Helping function CheckCode inside keypad door blueprint

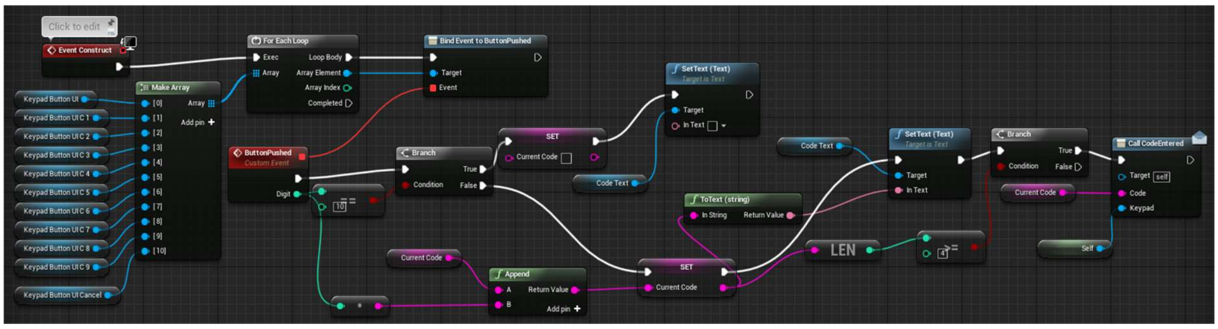


Figure 4.24 Keypad_UI widget. Logic for entering the code

4.1.6 Moving platforms

Platforms' blueprints are basically the same as the doors'. The platforms have different animations similar to Figure 4.21.

4.1.7 Containers

Containers implement InventoryComponent, that means they have the same inventory as humans. If the player is in container's trigger box and presses 'F', the inventory window for container is open. Right clicking on an item will transfer it into the player's inventory.

4.1.8 Moving objects

Inspired by TeslaDev YouTube tutorial [17]. Press 'F' looking at a physical object such as cube. The object is then attached to HeldObjectLocation which is a child of a camera in ThirdPersonCharacter and has the same rotation as the player. Press 'F' again to drop the object. The player cannot pick up objects that are below him, objects that are already picked up will be dropped.

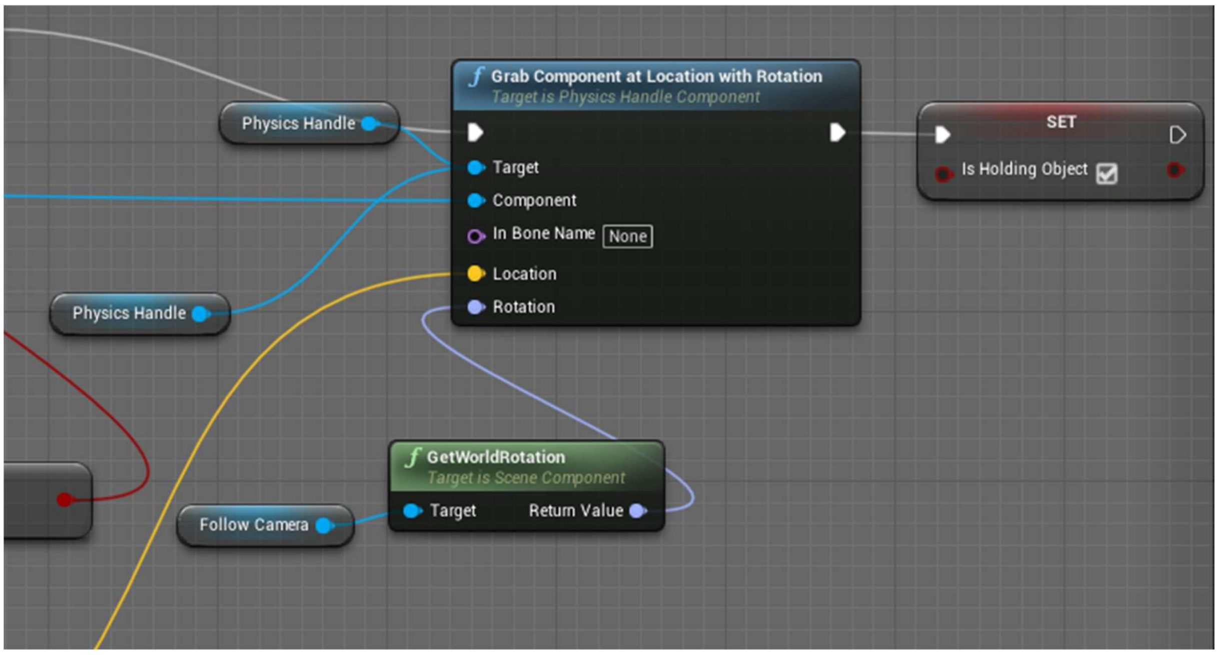


Figure 4.25 Grabbing objects

4.1.9 Outline

Outline highlights the character you can possess, items that can be picked or cubes that can be grabbed. In Unreal engine there is no simple way to implement outline. There are 3 steps needed.

1. Create PostProcessVolume of the size of the whole level.

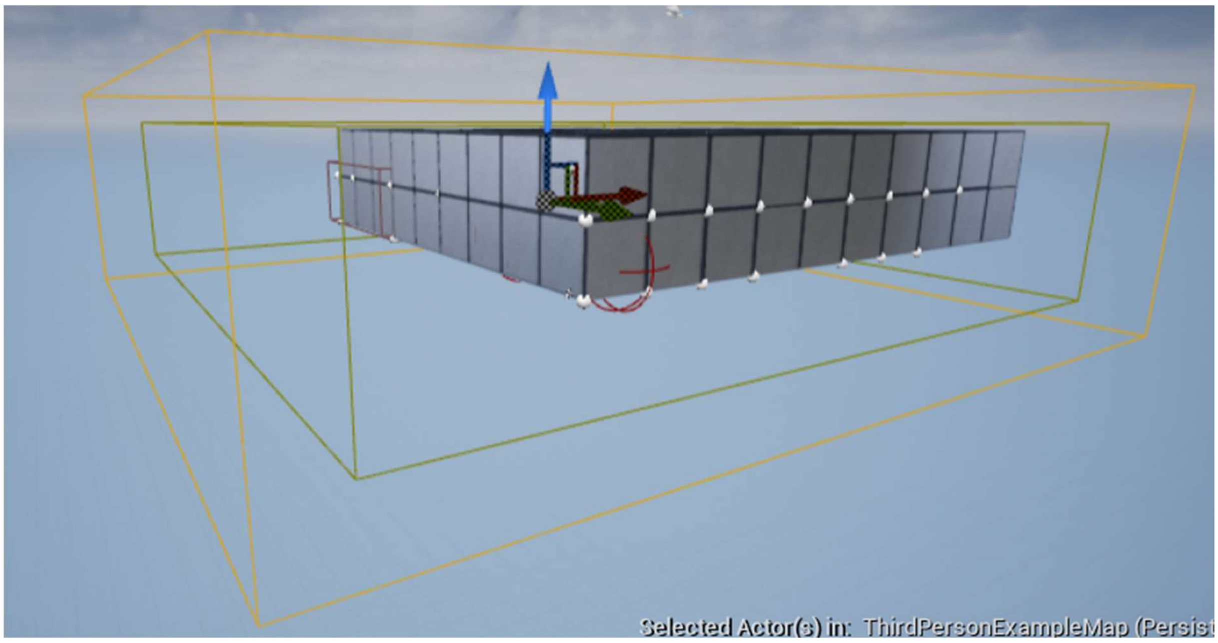


Figure 4.26 PostProcessVolume (yellow box)

2. Set its post process material to PP_Outliner_Inst [18]

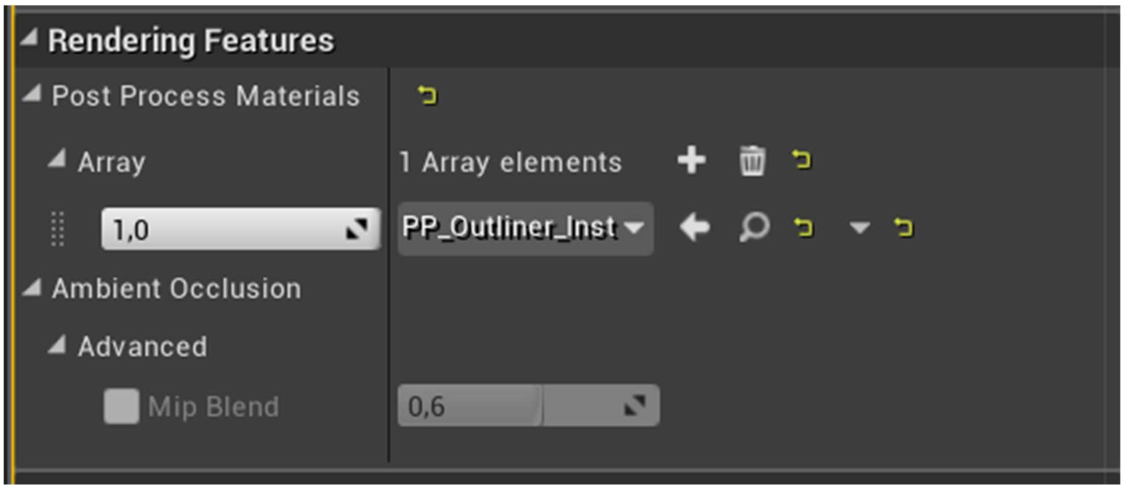


Figure 4.27

3. In blueprint for desired highlighted item or character whenever it is needed to draw outline set RenderCustomDepth to true (or false when not needed).

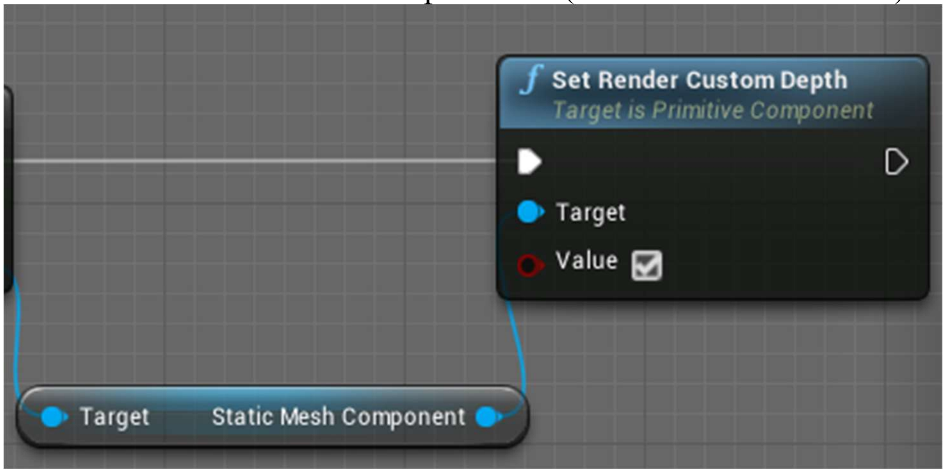


Figure 4.28

4.1.10 Sounds

I have taken all royalty free sounds from freesound.org [12]. For each sound I have created a cue, then I play it when needed with PlaySound2D (for UI sounds without exact location) or SpawnSoundAttached (for things in world with location, e.g. opening doors).

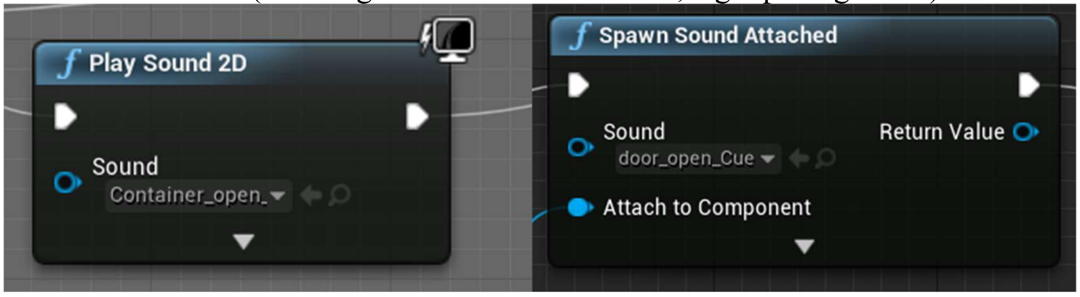


Figure 4.29

4.1.11 Checkpoints

When character enters the trigger box for checkpoint (Checkpoint actor) the game saves character with items and his position to save slot, if no save slot exists a new one is created. The saved information is stored in SaveGameObject. By pressing the 'Load last checkpoint' button from menu or opening the game with existing save (logic in level blueprint) the player will load at the last reached checkpoint. Restarting level will delete previous save. [19]

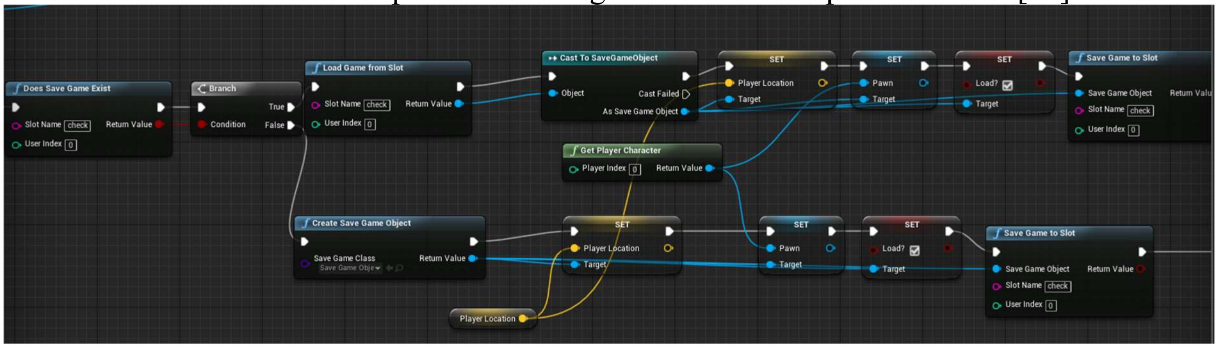


Figure 4.30 Saving game (Level blueprint)

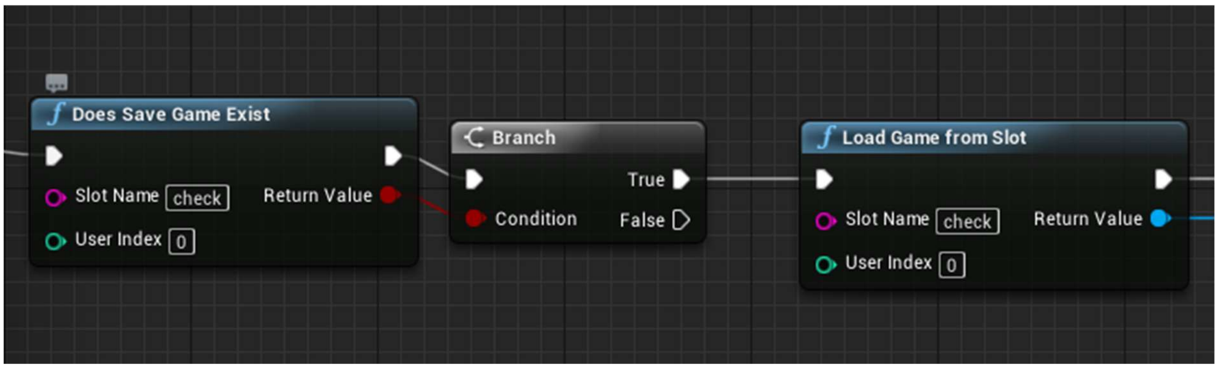


Figure 4.31 Loading game

4.1.12 Lighting

For the lighting I have created Lightmass Importance Volume and Post Process Volume stretched for the whole level. Lightmass creates lightmaps with complex light interactions like area shadowing and diffuse interreflection. It is used to precompute portions of the lighting contribution of lights with stationary and static mobility. The Post Process Volume is a special type of volume that can be added to a level and, because Unreal Engine 4 does not make use of post processing chains, these volumes are currently the only way to manipulate post processing parameters. Then I have created a ceiling prefab. It consists of the floor320x320 mesh, a lamp with emissive material and a rectangle light with following attributes.

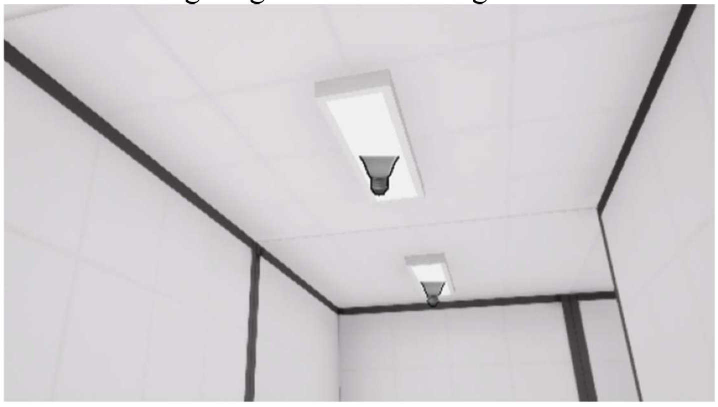
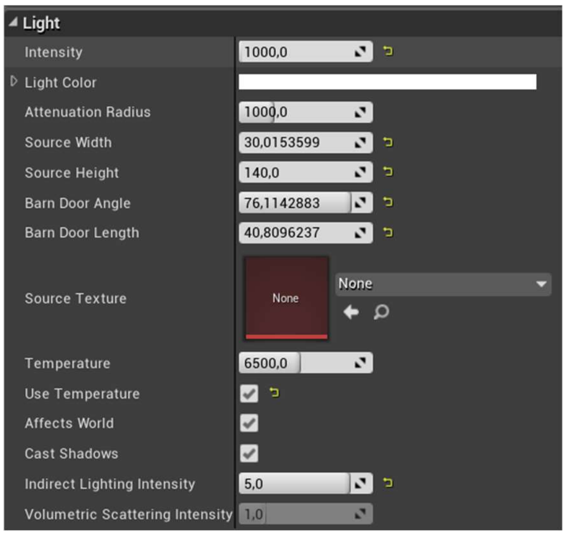


Figure 4.32 Ceiling



Light

Intensity	1000,0
Light Color	⬜
Attenuation Radius	1000,0
Source Width	30,0153599
Source Height	140,0
Barn Door Angle	76,1142883
Barn Door Length	40,8096237
Source Texture	None
Temperature	6500,0
Use Temperature	✓
Affects World	✓
Cast Shadows	✓
Indirect Lighting Intensity	5,0
Volumetric Scattering Intensity	1,0

Figure 4.33 Light attributes

4.2 Modular components

With polygon counts off the charts and hardware pushing more detail than you care to create, environments can be simply staggering. But with every step forward in graphics capabilities, many find themselves wondering how to utilize those advances. When we can draw limitless detail, it seems limitless talent is required to create those worlds. How do we quickly generate enough consistent quality content to fill a highly detailed world? How do we make sure those worlds can be easily modified and made flexible to design whims? The modular level design solution arose from the need to have greatlooking, high-detail levels without having to build and texture every nook and cranny of the environment. [20]

My aim was not to create a great looking, high-detailed world but using modular components definitely speeded up the process of designing levels. With the set of prefabs, I describe further, I could spend more time thinking about more interesting puzzles than modelling those rooms instead.

4.2.1 Walls, floors, ceilings

The first thing we need in a level is of course floors and walls. I have modeled 300x300x10cm wall and 320x320x10 floor in blender. For the walls to align and form a grid in a game world I have created a pole with size 300x10x10cm and a cube with size 10x10x10cm. Combining these 4 elements I created a set of building blocks that align perfectly. The size of each cubic building block is 320x320x320. I have already described the ceiling with lighting in 4.1.12. The materials I use are from free Necro's Utility Material Pack from Unreal Engine Marketplace. [21]

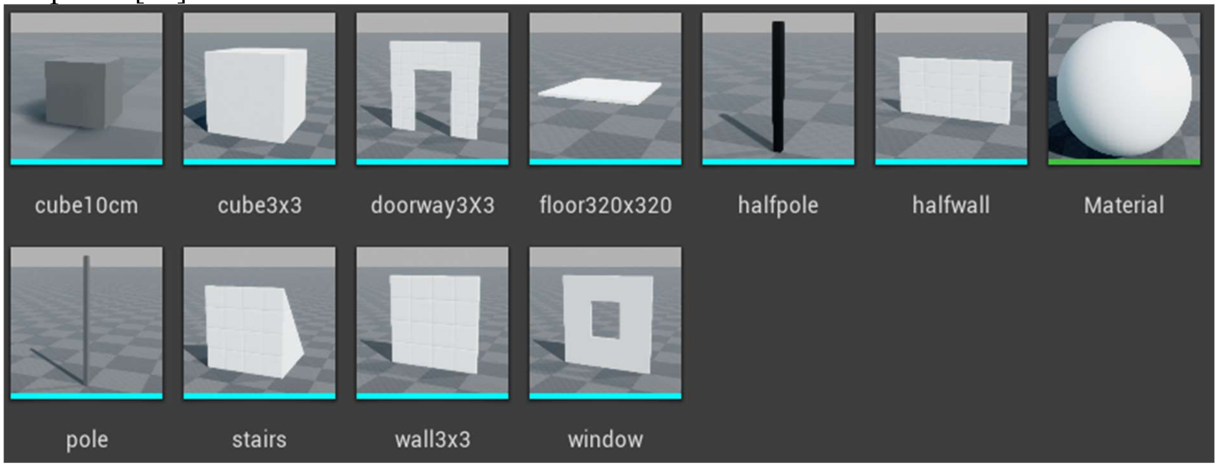


Figure 4.34 Basic elements

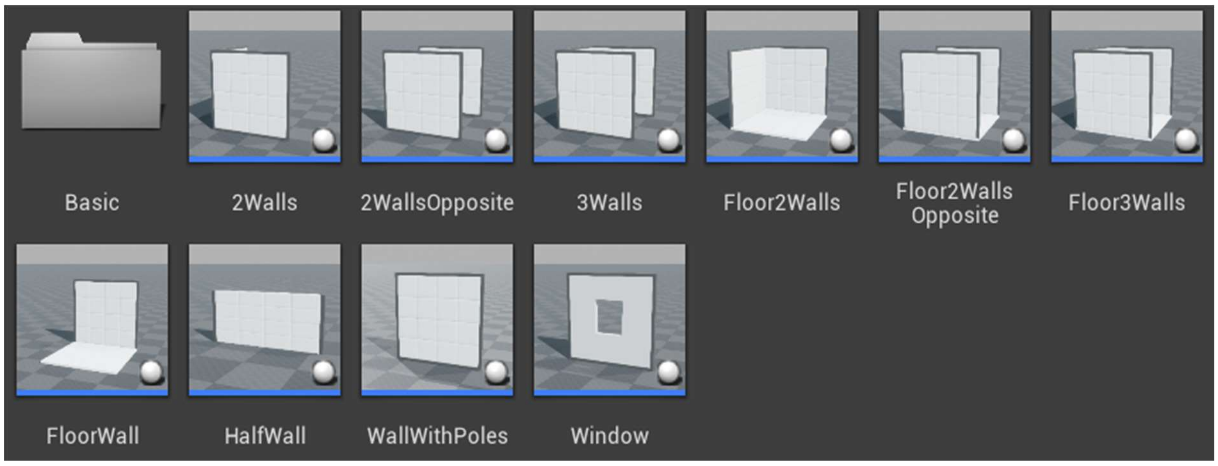


Figure 4.35 Modular components

4.2.2 Doorways

For the doorways I use doorway3x3, poles, cubes and a respective door which can have different lights or keypad attached. Some doors have buttons which can be moved freely once placed in the world.

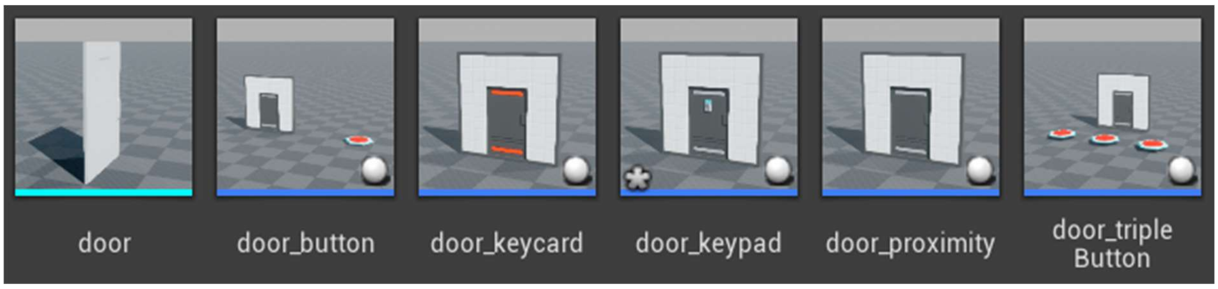


Figure 4.36 Doors

4.2.3 Platforms

Platforms are the two-part modular component of the same 300x300 size. Small platforms are ¼ of that. Each platform has a floor or wall button which can also be moved as the door button.

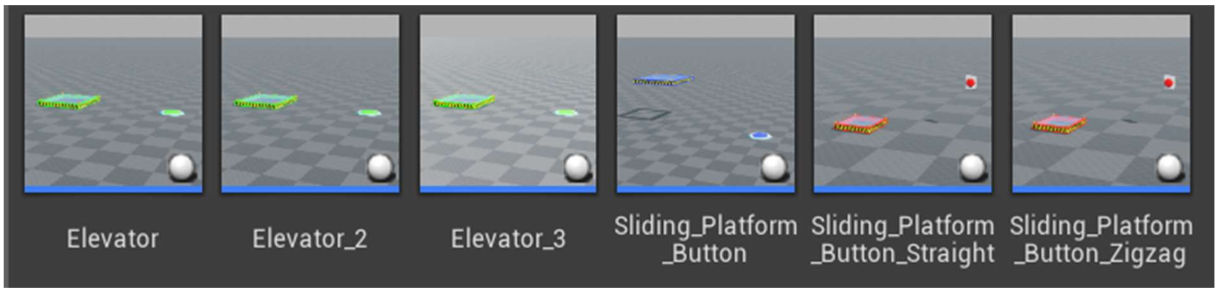

Elevator
Elevator_2
Elevator_3
Sliding_Platform _Button
Sliding_Platform _Button_Straight
Sliding_Platform _Button_Zigzag

Figure 4.37 Different types of platforms vary in moving patterns and speed.

Overall, the usage of modular components greatly helps with level design. By changing the Position Grid Snap value in editor to 320, blocks of the same size can be put quickly at desired location without adjusting coordinates. Any change in geometry of basic shapes or materials propagates to all components. However, I still have an option to customize particular elements placed in the world by moving meshes or changing the color.

5 Testing

5.1 Testing 1

It is easy, when developing a game, to fantasize about the player experience and to imagine how great it will be. Playtesting is necessary to serve as a wakeup call and force you to solve the ugly problems you have been putting off. [22] I have conducted the testing of the game among 6 of my friends in the middle of the development. All of them have played Portal at least. Their skill and experience in games varied from casual to very skilled. By that time, I have only designed the first level. The aim of the testing was to find out how long it takes to finish the level, get feedback on mechanics and level design, to find bugs early in the development and to get suggestions and new ideas for the future. Testers launched the game on their personal machines, no technical difficulties were met. I watched them play personally when possible and made notes, for others I watched their screen stream with Discord. After playing session they were asked to fill in short Google form.

5.1.1 Level 1 finish time

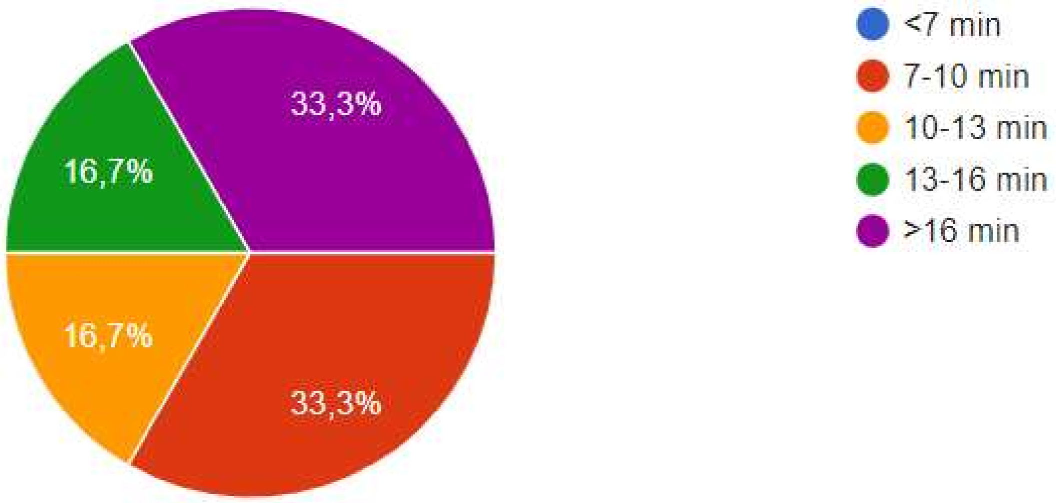


Figure 5.1

Nothing unusual was observed, more skilled gamers would finish the first level in about 10 minutes as expected, while less experienced, although familiar with this type of games, struggled up to 20 minutes.

5.1.2 How is the main mechanic of switching characters in this form (first-person, controlling only one at a time) new to you?

Figure 5.2

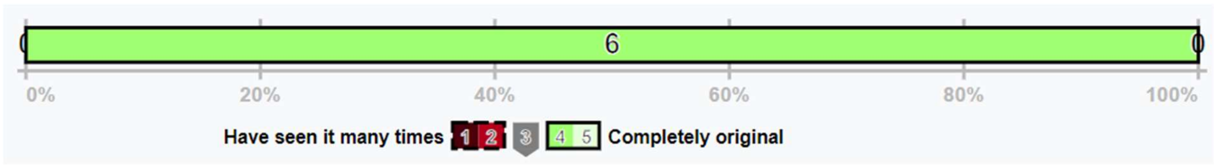
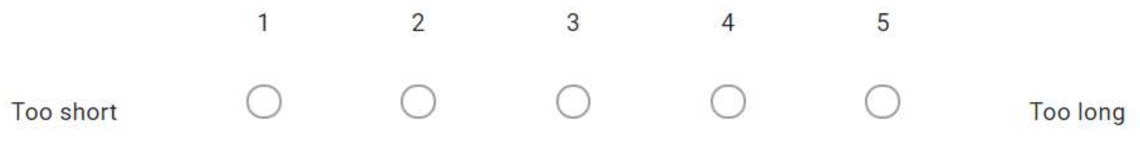


Figure 5.3

All 6 testers said that the mechanic was new to them but not completely original.

5.1.3 Level duration (expect more levels of the same duration)



	1	2	3	4	5	
Too short	○	○	○	○	○	Too long

Figure 5.4

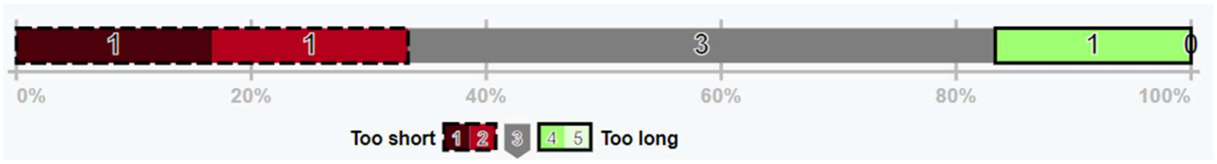
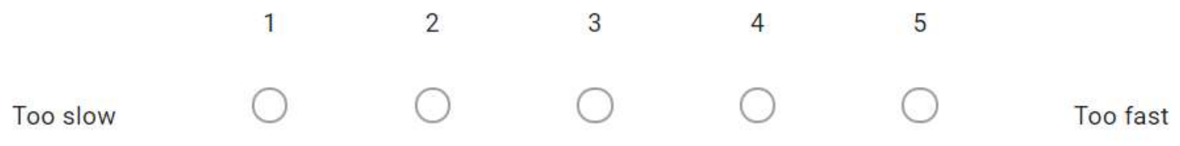


Figure 5.4

Half of the testers were satisfied with level duration. Two found it underwhelming and wanted to play more. Only one said it was a little long.

5.1.4 Pace of the game



	1	2	3	4	5	
Too slow	○	○	○	○	○	Too fast

Figure 5.6

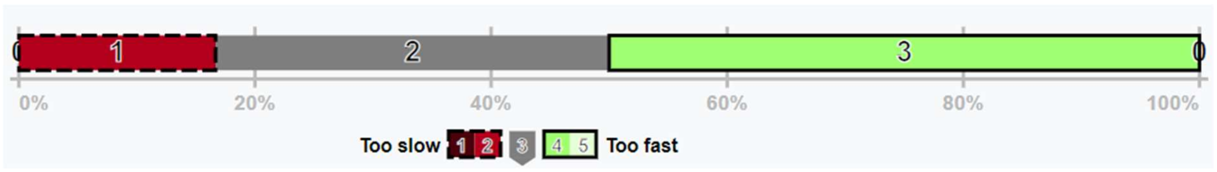
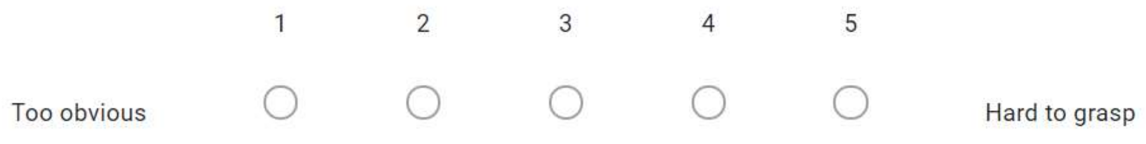


Figure 5.7

Not too fast, not too slow. Expected results for tutorial level, where most of the puzzles were the introduction to new mechanics.

5.1.5 Introduction to new mechanics



	1	2	3	4	5	
Too obvious	○	○	○	○	○	Hard to grasp

Figure 5.8

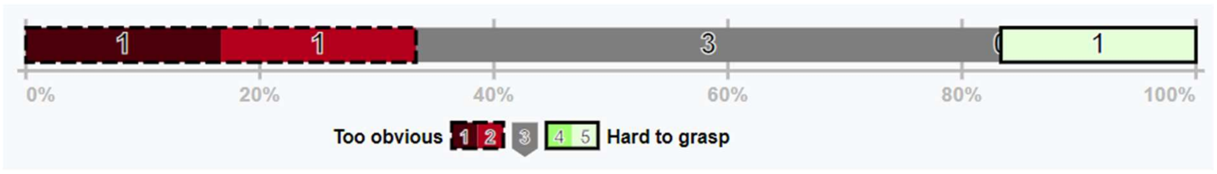


Figure 5.9

Generally, the tutorial system was praised for consistency. New mechanics were easy to get into. The only difficulty 2 testers reported was the first character swap as there was no hint of what the player was expected to do. I have added the hint later.

5.1.6 Rooms size

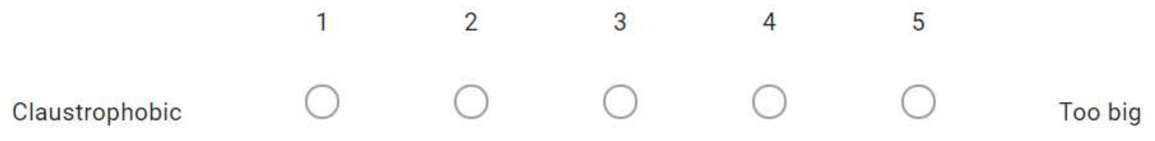


Figure 5.10

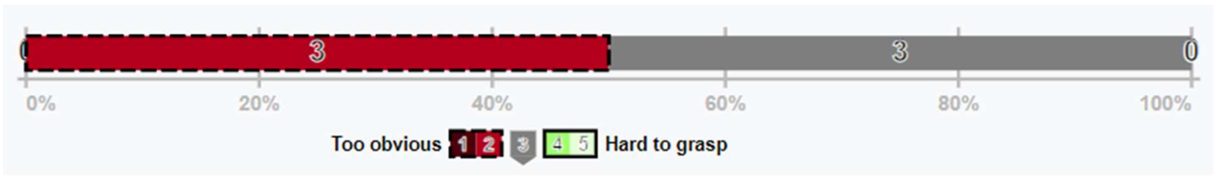


Figure 5.11

Half of the players reported the rooms were a little bit small. Which is fine for the first tutorial level. Next levels would have more complex puzzles therefore bigger rooms.

5.1.7 Which way did you finish the last puzzle?

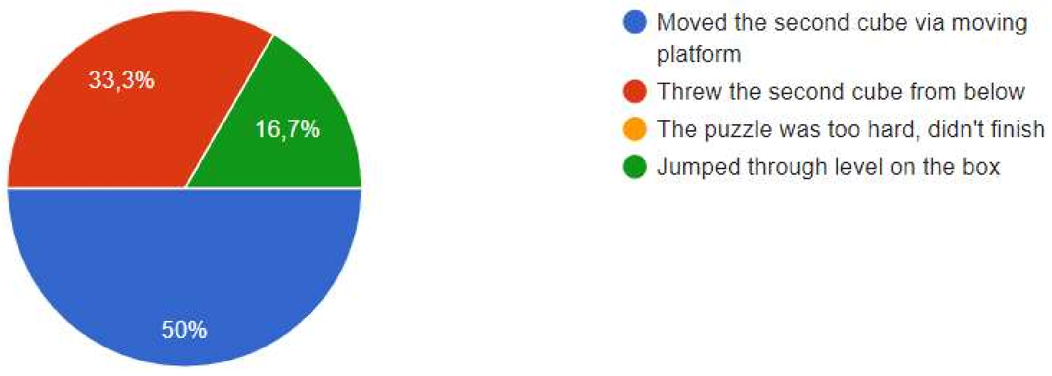


Figure 5.12

For the last and the hardest puzzle, I originally had only one solution in mind. However, during the previous informal testing, the tester solved it in a different way. I decided not to change anything so the puzzle would have 2 solutions which is even better. Luckily, he was not the only one and two players solved it the same way, while three solved it how intended. One individual abused the bug of jumping on a box that he is holding (known mechanic for speedrunning some games).

5.1.8 Anything particularly annoying or bugged?

Switching characters

Two testers suggested that switching animation could be faster for when the distance between two characters is short.

Player movement

For some players mouse sensitivity was too high and wanted the setting to change that.

Codes

One tester would like to have keyboard input, bigger buttons or less sensitivity for easier button press on code panels.

Another tester had trouble with the second keycode. He requested a hint because he thought he needed to subtract 30 from the current 2020 year and kept entering 1990, not from 2076 that is written on a warning sign to get the right 2046 code.

However, two other players found this puzzle interesting.

Moving objects

- As in the previous question stated there is a bug which allows you to jump on a held box.
- Sometimes playing around with boxes would launch the player up in the sky.
- There is a bug that lets two characters hold the same box at the same time.
- Boxes could be squeezed through the doorways.

Platforms

It is possible to get stuck at the elevator.

5.1.8 General suggestions

- Make boxes easier to rotate.
- Hold button to open the inventory instead of toggle.
- Mouse sensitivity setting.
- Make visual indications that the button is pressed.

5.1.9 Conclusion

The testing showed that the development is on the right track. The testers liked the game and would like to play more. Some adjustments are needed to be made to tutorial and controls. Several bugs were discovered, all of them were fixed by the second testing. Overall, the testing experience was very useful.

5.2 Testing 2

Rules from the first testing apply. The 6 testers are the same ones that participated in the first testing. They were asked to play level 2 and 3. In the beggining I found out that a lot of them have forgotten some of the mechanics and replayed level 1 to refresh their memory. The players played the final version of the game, which strongly differs from previous version. I have added a few new mechanics, changed the visual style, added several models, sounds, music and menus.

5.2.1 Level 2 finish time

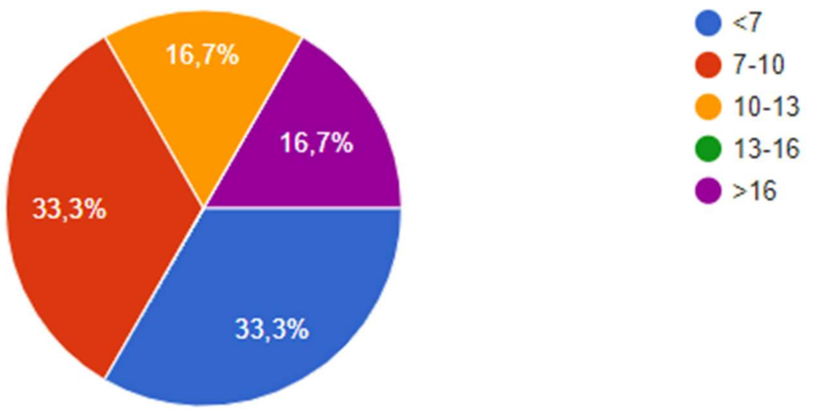


Figure 5.13

The finish time in level 2 varies a lot. 1 player has lost a lot of time in the first room, not understanding how to grab the cube in the air. Room 2 was probably the most challenging in the game for the players but nobody got stuck completely. Room 3 was easy but fun, because of the new jumping boots mechanic.

5.2.2 Level 3 finish time

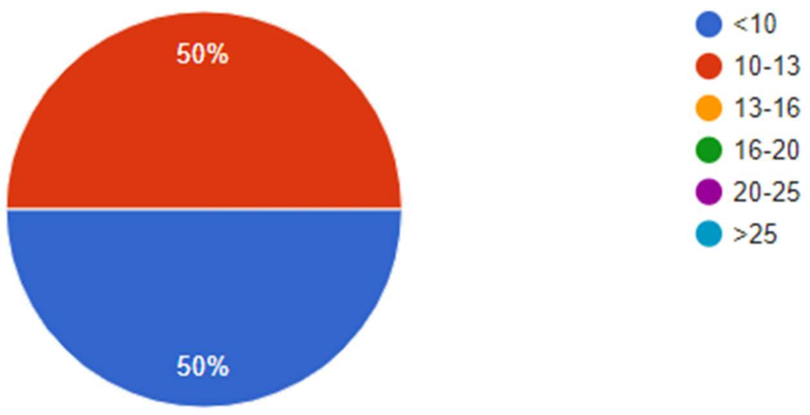


Figure 5.14

All 6 finished the level in under 13 minutes. The players enjoyed pretty much everything and did not have problems. Less experienced players were a bit slower in the room with the balcony. The whole level was praised because it felt like one big puzzle.

5.2.3 Originality of puzzles

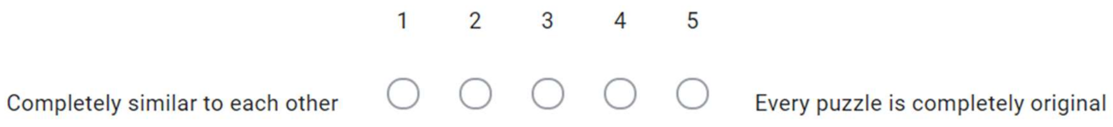

1
2
3
4
5
Completely similar to each other
Every puzzle is completely original

Figure 5.15

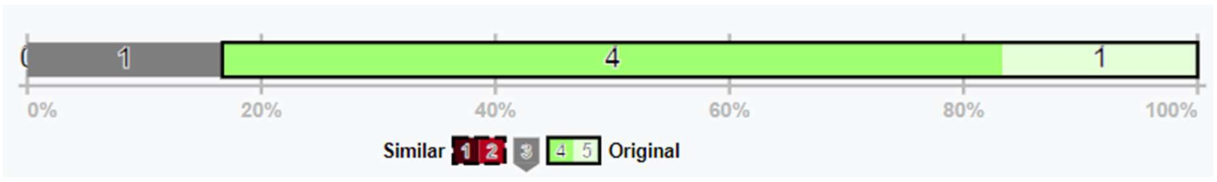


Figure 5.16

Great result, considering how much puzzles the game has. Having every puzzle completely distinct from each other is impossible even in games such as Portal and The Talos Principle, it would shorten the game by a lot.

5.2.4 Your favorite puzzle/room (up to 3, whole game, very simple not included)

1. 4 players named the room with jumping boots in level 3 their favorite. They enjoyed a little change in gameplay after constant door opening and character switching.
2. 3 players chose the puzzle with numbers on the walls. They found it interesting to notice some numbers and only after some time realize they are needed to open the door.
3. Room 2 in level 2 got the third place. Testers liked its non-linearness.

   The rest of the puzzles got 1 vote each.

5.2.5 How do visual elements fit together? Interior, buttons, platforms, items etc. (not graphics)

Figure 5.17

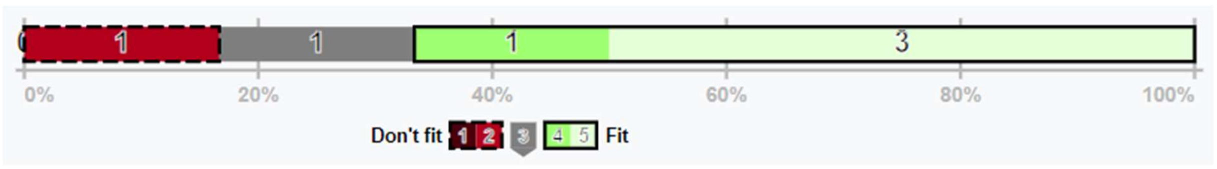


Figure 5.18

Overall, they fit almost perfectly, except testers reported that it is not always clear which door will be opened after pressing the button and some distinctions between different types of doors (proximity and button) would be good.

5.2.6 Sound design (not quality of the recorded sounds)

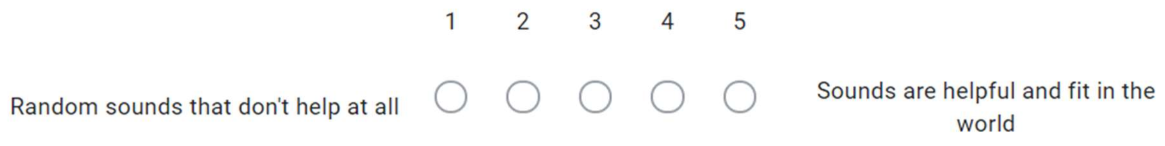

1
2
3
4
5
Random sounds that don't help at all
Sounds are helpful and fit in the world

Figure 5.19

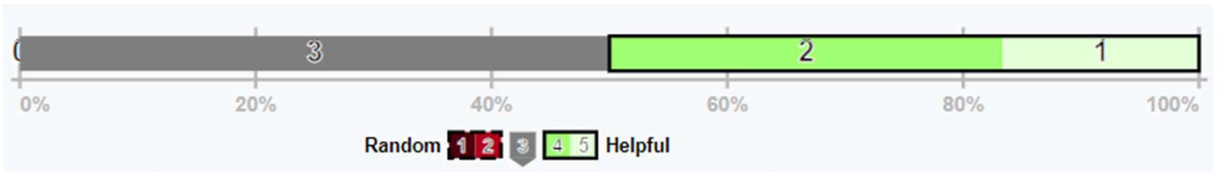


Figure 5.20

Everybody agreed that the addition of sounds improved the experience. However, the volume of some elements should be corrected. Most of the tetsers enjoyed the ambient music.

5.2.7 Perfomance

Testers got stable frames per second on machines from middle-tier laptops to high-end PCs.

5.2.8 Bugs

1. Being able to grab a cube while something or somebody is standing on it.
2. Inventory window does not close when switching the character.
3. Postiotions of some trigger boxes should be adjusted.

5.2.9 What needs to be done to make you buy this game?

The ranking is the following:

1. More puzzles (5 votes)
1. Harder puzzles (5 votes)
3. More mechanics relative to switching charaters (3 votes)
4. Non-linear puzzles (2 votes)
5. More mechanics not related to charater switching (1 vote)
6. Story (1 vote)
7. Better graphics (1 vote)

So, mostly the players enjoyed the game as is and would like the game just to have more puzzles and harder ones.

5.2.10 Conclusion

The second testing was a big success. The testers spent almost as much time solving puzzles as I expected. They expressed genuine interest and would like to play more. Several minor bugs were discovered. Some changes to visual style and sound design need to be done.

6 Conclusions

In his bachelor thesis I have managed to create a puzzle game with switching playable characters as the main mechanic. I have built 3 levels using modular components.

After analyzing successful first-person puzzle platformers, games with switching playable characters and an inventory system in Deus Ex game in Chapter 2, I chose Unreal Engine in contrast to Unity to be the engine of the game.

In Chapter 3 I described all of the game mechanics and their usage in building puzzles. I have created 3 levels of roughly the same length which require logic and spatial thinking.

Chapter 4 focuses on transforming these ideas to the blueprint language of Unreal Engine, as well as describing the usefullness of modular components in modern game development and using them to build the 3 levels.

In the middle and at the end of the development I have conducted qualitative tests. The first test showed the interest of the testers in such game and revealed several bugs that were fixed later. The second test took place after creating two new levels, changing several mechanics and adding some new, as well as changing the visual style. The testers appreciated the changes and new puzzles.

The experience I have gained making the game is invaluable. I now have deeper understanding of Unreal Engine and game development in general. The work on the game is not finished. I have plans about future development in my free time alone or with other enthusiasts.

References

[1] 2007. [Online]. Available: https://store.steampowered.com/app/400/Portal/.

[2] M. Keil, „"G4 Review — The Orange Box",“ 19 October 2007. [Online]. Available: http://www.g4tv.com/games/xbox-360/38752/the-orange-box/.

[3] Valve, *Portal. Level/area: In-game developer commentary.,* 2007.

[4] B. Kuchera, 16 Spetember 2011. [Online]. Available: https://arstechnica.com/gaming/2011/09/portal-is-used-to-teach-science-as-valve-gives-game-away-for-limited-time/.

[5] 2014. [Online]. Available: https://store.steampowered.com/app/257510/The_Talos_Principle/.

[6] 2016. [Online]. Available: https://store.steampowered.com/app/499520/The_Turing_Test/.

[7] 2013. [Online]. Available: https://store.steampowered.com/app/271590/Grand_Theft_Auto_V/.

[8] 2003. [Online]. Available: https://store.steampowered.com/app/1261520/Asterix__Obelix_XXL_Romastered/.

[9] 2011. [Online]. Available: https://www.ubisoft.com/en-us/game/driver-san-francisco/.

[10] 2000. [Online]. Available: https://store.steampowered.com/app/6910/Deus_Ex_Game_of_the_Year_Edition/.

[11] B. Sewell, Blueprints Visual Scripting for Unreal Engine, Packt Publishing, 2015.

[12] „freesound,“ [Online]. Available: https://freesound.org/.

[13] B. Kamiński. [Online]. Available: https://gumroad.com/bartkamski .

[14] „freemusicarchive,“ [Online]. Available: https://freemusicarchive.org/.

[15] R. Koster, Theory of Fun for Game Design, O'Riley Media, 2013.

[16] R. Laley, „Unreal Engine 4 Tutorial - Inventory System,“ 22 3 2018. [Online]. Available: https://www.youtube.com/watch?v=yxqSkFNAzE0.

[17] TeslaDev, „Unreal Engine 4 Tutorial - Pickup and Drop Objects (Physics Handle),“ 12 11 2015. [Online]. Available: https://www.youtube.com/watch?v=OltZ-xFz8IQ.

[18] PawelM, „michalorzelek,“ [Online]. Available: http://www.michalorzelek.com/blog/tutorial-creating-outline-effect-around-objects/.

[19] Kaosrio. [Online]. Available: https://www.youtube.com/watch?v=T0LcEedZkRQ.

[20] L. Perry, "Modular Level and Component Design," *Game Developer,* no. November, 2002.

[21] Necrophob30, „Necro's Utility Material Pack, Unreal Engine Marketplace,“ 2014. [Online].

[22] J. Schell, The Art of Game Design, CRC Press, 2008.

Appendix

The attached archives contain the game and the source project. Minimum requirements for the game are 1 GB of free space and a Windows PC with a discrete GPU. Unreal Engine 4.22.3 is required to open the project.