Development of a Virtual Reality Puzzle Video Game

Design and Development of Video Games Degree
VJ1241 - Bachelor's Thesis
07/07/2020

Author: Pau Fenollosa i Ángeles

Tutor: José Vicente Martí Avilés

Summary

This document presents the Degree's Final Project in the Design and Development of Video Games Degree. The work to develop will consists of a virtual reality puzzle video game for the *Oculus Quest* lens. In this game the player plays as a figure toy in a child's disorganized room. His objective is to throw a ball and make it reach the goal point. To do this, he builds blocks and other items to make a path for the ball. Care has been taken to avoid the most common causes of motion sickness with a resulting of a paused gameplay.

Keywords: Motion sickness, Virtual reality, Oculus Quest, Unreal, Paused Gameplay, Puzzle.

Index

List of tables

List of figures

1 PROJECT MOTIVATION

This section presents a summary of the motivation of the project, and the related subjects done on the degree.

1.1 Introduction and Project Motivation

Nowadays, virtual reality is getting stronger year after year and it is used a lot in video games, but also in other industries for training people, showing products, treat some diseases and so on. That is because virtual reality offers six degree of freedom in the head and both hands. This creates a feeling of immersion and connectedness that is a useful tool to help users to generate and retain memories.

For this, it is important to know the basics of virtual reality to make good products without bugs and spatial dissonances. That can make the product uncomfortable and lead the player to leave it. Or worst, to make the player have motion sickness and lead them to dizziness and have a bad experience. Something that we really do not want specially when we are treating a patient or showing a product to a customer.

Most of the virtual reality video games on the market are in first person point of view. All this games exploit the sense of scale that virtual reality offers, but they are very dynamic and with a lot of fast camera moves. Most people have not had an experience in virtual reality, and they feel overwhelmed on their first try. Too many new sensations. Therefore, there is a need of something on the half way that helps customers to introduce them smoothly in the virtual reality.

This project has the intention to cover this lack, using the first person point of view in virtual reality too for a natural placement of the body in the virtual world. The project also has a paused gameplay in order to obtain a more comfortable and relaxed experience and avoid extremely fast camera moves in the game. That is why camera in the game is designed to be moved with the body natural movements and using the teleport only. That's the best way to avoid motion sickness.

The video game tries to be a simple experience with all the tricks and tips for avoiding motion sickness.

1.2 Related Subjects

Several subjects of the degree design and development of video games are related to this project, but the most relevant are the following:

- **VJ1235 Advanced Interaction Techniques:** The objective of this subject is to learn new interaction techniques that go beyond the traditional ones. The virtual reality is applied in this work thanks to this subject.
- **VJ1227 Game Engines:** This subject introduces the basic architecture of a game engine with special emphasis on the graphics engine, physics engine and programming behaviors of game objects (scripting).
- **VJ1222 Video Game Conceptual Design:** This subject provides techniques to learn how to balance video games' elements such the level designs, the mechanics and the objectives for making it all work perfectly together. The work considers some of these techniques to be in balance.

2 PLANNING

The section presents the desired objectives to be achieved, the forecast time to do the project and the expected results after the realization of the project.

2.1 Objectives

The following objectives are those that are intended to be achieved with the realization of this project.

- Create an immersive experience with the virtual reality, exploring his possibilities and evading typical errors detected by professionals.
- Learn to properly use from zero the game engine *Unreal*.
- Build cohesion between the mechanics and the level design to make a good gameplay for the player.

2.2 Project Schedule

This section describes the division of project's tasks and its estimated time in hours. The project has been divided into three phases. Each one contains a set of tasks and its estimated duration.

2.2.1 Documentation Phase

Table 1 shows all the required tasks, the estimated time for the writing of the documents, the edition of the videos and the presentation.

Table 1 Documentation phase planning

Documentation	
Tasks	*Estimated duration (in hours)*
Technical proposal	10
Technical report	40
Video	5
Project defense presentation	5
Total	**60**

2.2.2 Game Design Phase

This phase shows all the subtasks for the elaboration of the Game Design Document (GDD) which specify all the design elements of the video game. All these elements are shown in the Table 2.

Table 2 Game design phase planning

Game design	
Tasks	*Estimated duration (in hours)*
Gameplay and mechanics	9
Level design	7
User interface	4
Total	**20**

2.2.3 Game Development Phase

In this subsection, the development tasks are presented. These tasks represents all the programming tasks and the adjustments in the engine for creating the video game. These are divided into six blocks for simplicity and are shown in the Table 3.

Table 3 Game development phase planning

Game development	
Tasks	*Estimated duration (in hours)*
Set up the virtual reality environment	10
Physics	30
Gameplay and mechanics	70
Environment	30
User interface	20
Debugging	60
Total	**220**

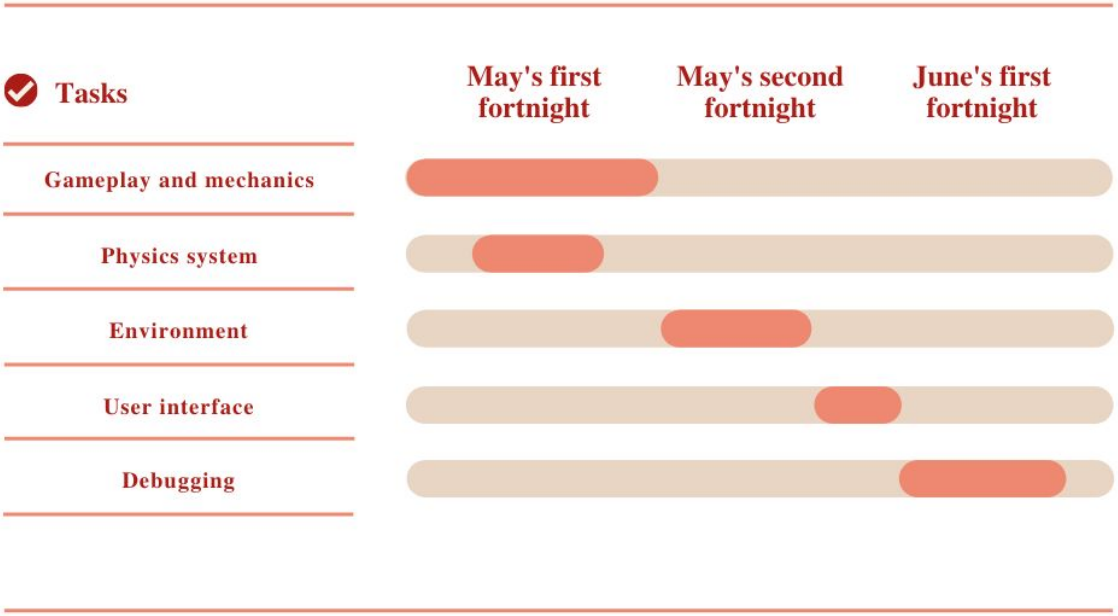
2.3 Development Plan

This section describes the development plan of the project, that is a reference and can be changed and adapted during the development of the project.

Each workday will be splitted in two work sessions of 4 hours each one. Following the Pomodoro technique, every 20 min of work will be a 5 min break and 1 hour for lunch.

The totality of the project has been divided into three fortnight each one containing a set of tasks.

Figure 1 Toys' Game development plan Gantt chart



2.3.1 May's First Fortnight

The first fortnight of the project will be exclusively dedicated to the most important part of the project, the mechanics. But before this, all the involved devices in the project have to be set up in the virtual reality development environment. The *Oculus Quest* lens needs to get the first set up and update all the drivers. Also needs to be linked to the PC and unlock the development mode. The PC needs to be configured with drivers and activate some options to be a friendly environment to the *Oculus Quest* device. Also, the Unreal engine needs to be configured for the *Oculus Quest* virtual reality environment.

After setting correctly the virtual reality programing environment, the basic mechanics like movement and interact actions will be programmed. This will be created alongside with the physics of the game to fit better with the mechanics and reduce the debug time.

2.3.2 May's Second Fortnight

During this fortnight, the environment of the game will be settled down. First, a basic design of the levels with primitive forms will be constructed. That will help to

change the levels faster and improve them if any problem appears during the development.

Once this mock-up is done, his appearance will be improved with free assets and textures that fit in the aesthetics of the game.

After that, the same process will be done with the UI. Firstly, will be created a basic UI for more easily debug, and then the appearance will be beautified.

2.3.3 June's First Fortnight

In this last step, all the gameplay and scenarios will be revised and debugged for make the player comfortable and do not get motion sickness. After that, the game will be tested by different kind of users for gathering valuable information like how different users respond to the game and discover new bugs. Then, the project will be debugged once again to better fit to the users.

If after all of this, there is still time to develop, new features will be added.

2.4 Expected Results

The resulting project is expected:

- To be a short but completed video game, at least in terms of programming.
- To be easily scalable for future expansion.
- To be a project with a good finish.
- To be enjoyed by the player.
- To be a video game that draws the attention for people who never played virtual reality games before.

3 ANALYSIS OF THE PROJECT

This section presents an analysis of the project before starting the implementation. It describes the main idea of the project through the Game Design Document, the troubles found for the realization of the project and the strategy to realize it. It also details the possible tools to work in this kind of projects and those chosen for this one.

3.1 Game Design Document

A GDD is a design document of a video game. It's used by the developer as a guiding vision throughout the game development process. This document describes all the elements of the game, the mechanics, the story, the gameplay, UI, etc.

The next sections will present every part of the design document of this project.

3.1.1 Video Game Data Sheet

Platform: Oculus Quest

Perspective: First person

Genre: Puzzle

Language: English

PEGI: 7

3.1.2 Storyline

When nobody is at home, the toys come alive. They are bored of being still and quiet and start to play their own game, the Toy's Game.

3.1.3 Synopsis

Toys' Game takes place in a child's disorganized room. In here, the player embodies a toy figure, that moves around when nobody is at home. This figure called Enzo likes to play a game called the *Toys' Game*. The game consists of making a ball pass through the disorganized room and reach his goal in one throw. This particular toy wants to be the very best in this game. To reach this title, he needs to beat the level that no one has ever beat: *The Train Pass*. To succeed, he needs a lot of training to learn the basics of the game and mastering them. To do this, he decides to beat all existing levels from the bottom to the top. Along the way, he will have to overcome different challenges and hazards to reach his dream.

3.1.4 References

The video game pretends to be an immersive puzzle, where the world around the player is enormous and colorful. The idea is to take out the child inside the player and make him have fun without nothing worrying him.

The main reference of this project is Pixar's movie Toy Story. In there, the world is colorful and from the point of view of a toy. That makes it the perfect reference for Toys' Game.

Figure 2 Toy Story 2 movie scene (Image reference)

Another important reference is the virtual reality video game Tin Hearts. The game represents a world where the toys of a workshop gets alive. The player has to guide them to make a safe path to his goal. To do this, he places obstacles to change the toys path to make them avoid dangerous situations for them.

Figure 3 Tin Hearts (Image reference)

3.1.5 Gameplay

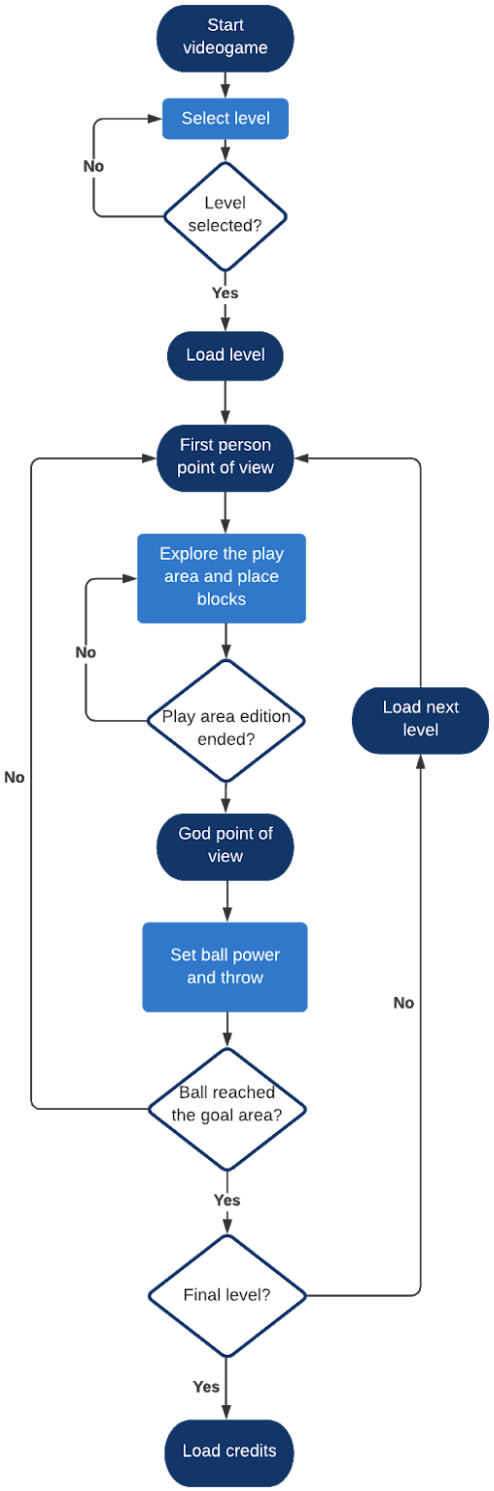
The video game is a virtual reality puzzle where you play as a toy in a child's room. The toy's objective is to make a ball reach her goal in one throw. Before the ball is thrown, the toy needs to prepare the area in order of making the ball bounce correctly to evade obstacles and hazards that redirect the ball to the goal. To do this, the toy can create and move child blocks.

When the area is ready, the ball is thrown and if the toy's calculations are correct, the ball will reach the goal and the level is completed.

3.1.6 Flowchart

The flowchart of a video game is a diagram of the different scenes of the game, that represents all the actions the player can do and how the game responds.

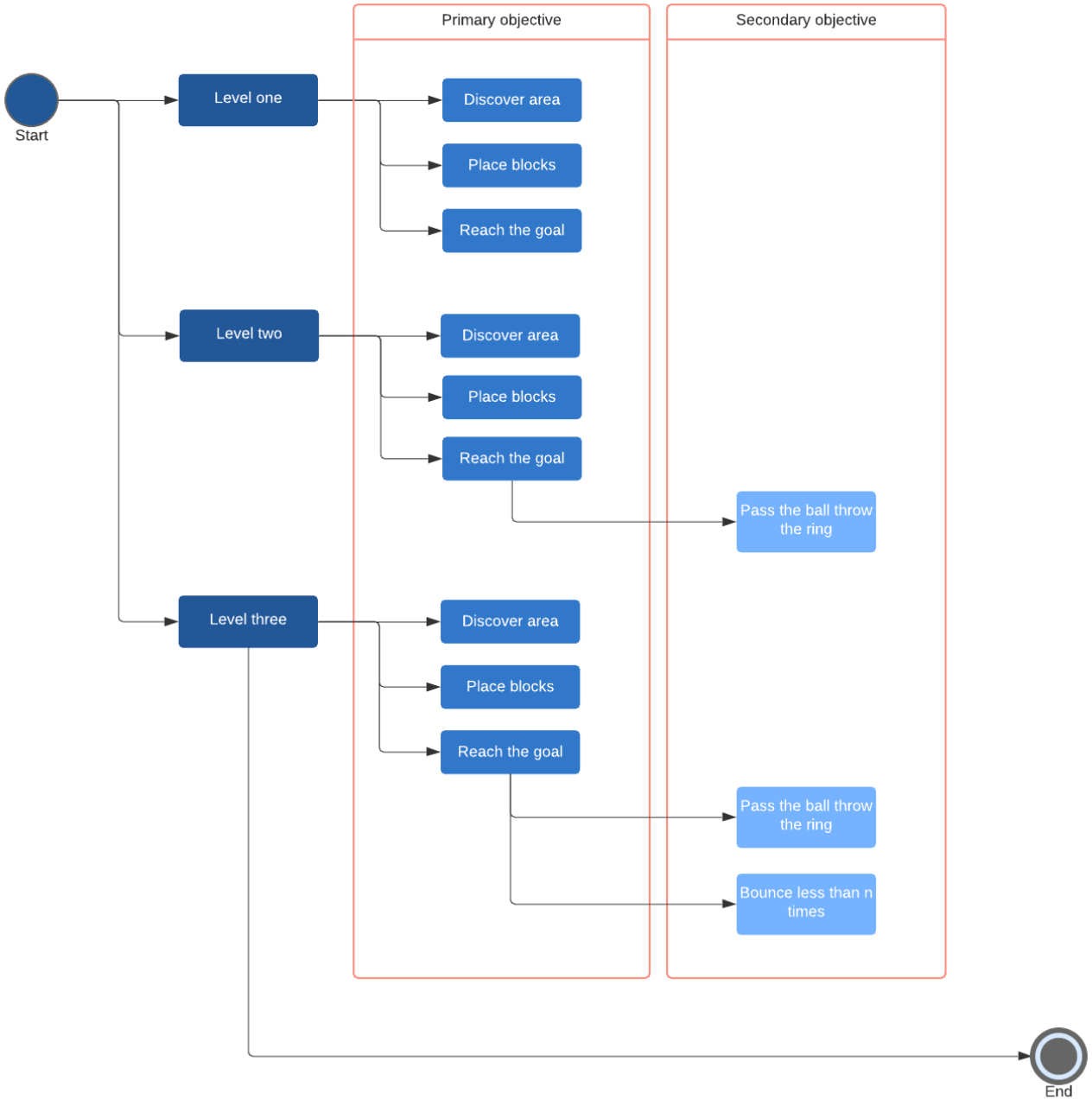
Figure 4 Toys' Game flowchart



3.1.7 Beat Chart: Challenges Structures

The beat chart of a video game is a diagram of the challenges structure, where the objectives of every level are represented.

Figure 5 Toys' Game beat chart



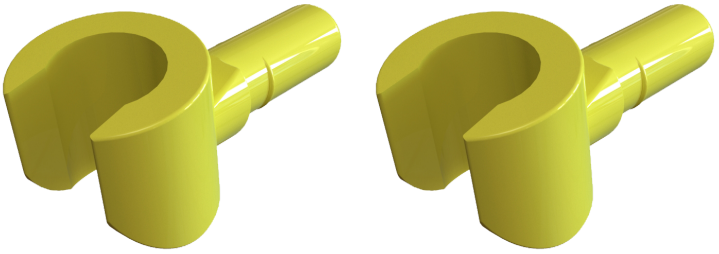
3.1.8 Checkpoint Structure

Every time you beat a level, you unlock the next one while being able to replay the previous levels at any time. There is no checkpoint system during a level play.

3.1.9 Entities

- **The player hands:** The player is represented like a Lego toy, but he only can see his own hands since we are using virtual reality and the camera is in the eyes of the player. These are used to grab and drag the blocks. The aspect of the virtual hands would be something similar to Figure 6.

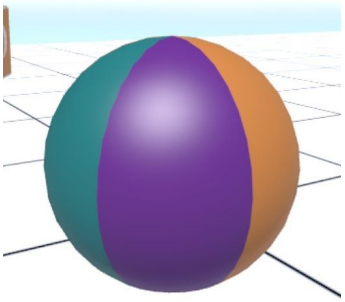
Figure 6 Lego's hands model (Image reference)



.

- **The ball:** The ball that has to reach the goal. It's a simple spherical object. Something similar to Figure 7.

Figure 7 Toys' Game ball (Project snap)



- **Child blocks:** These entities are used to block or change the path of the ball like Figure 8.

Figure 8 Toys' Game child block (Project snap)

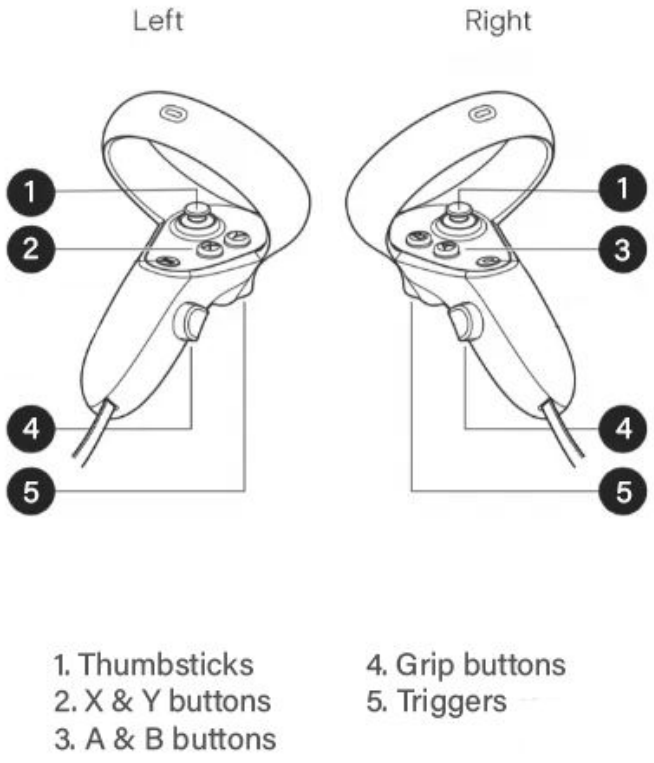
3.1.10 Scoring System

Every time the ball reaches the goal, the player scores points. The amount of points varies depending on how many bounces the ball have done before reaching the goal. Independently, the player can get extra points, making pass the ball through a ring.

3.1.11 Game Controls

The Figure 9 shows the *Oculus Quest* controllers and a mapping of his buttons. The related mechanic to each button it is shown below.

Figure 9 Oculus controller button guide (Image reference modified by the author)



- **Thumbsticks:** View rotation
- **Y & B buttons:** Show/hide the block creator UI.
- **X & A button:** Teleport
- **Grip buttons:** Grab action
- **Triggers:** Build object

3.1.12 UI System

The User Interface (UI) is an interactable panel inside of the video game that makes it easy and enjoyable to operate some actions like choose one option or another.

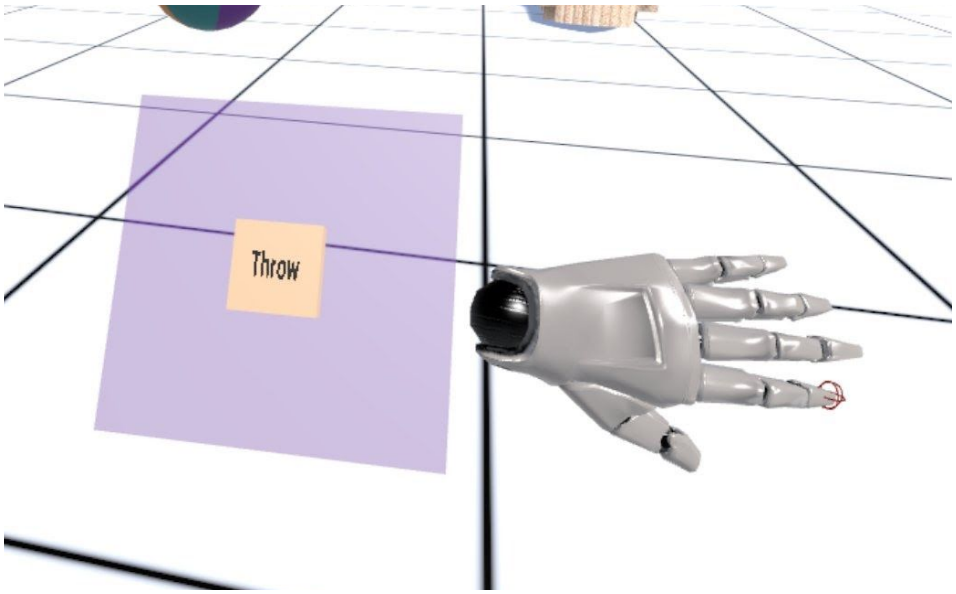
All the UIs in this game are non-invasive. They are in the world space and not overlapped in the camera of the player.

The main menu UI is a static panel placed in a room. The player moves around the room without being followed by the UI. Figure 10 shows an example.

Figure 10 Static UI panel of a Samsung Gear VR app (Image reference)

There is another UI in-game that is hidden, until the player request to being opened with the button B or Y. Once opened, the UI is like a watch, that follows the corresponding wrist like Figure 11.

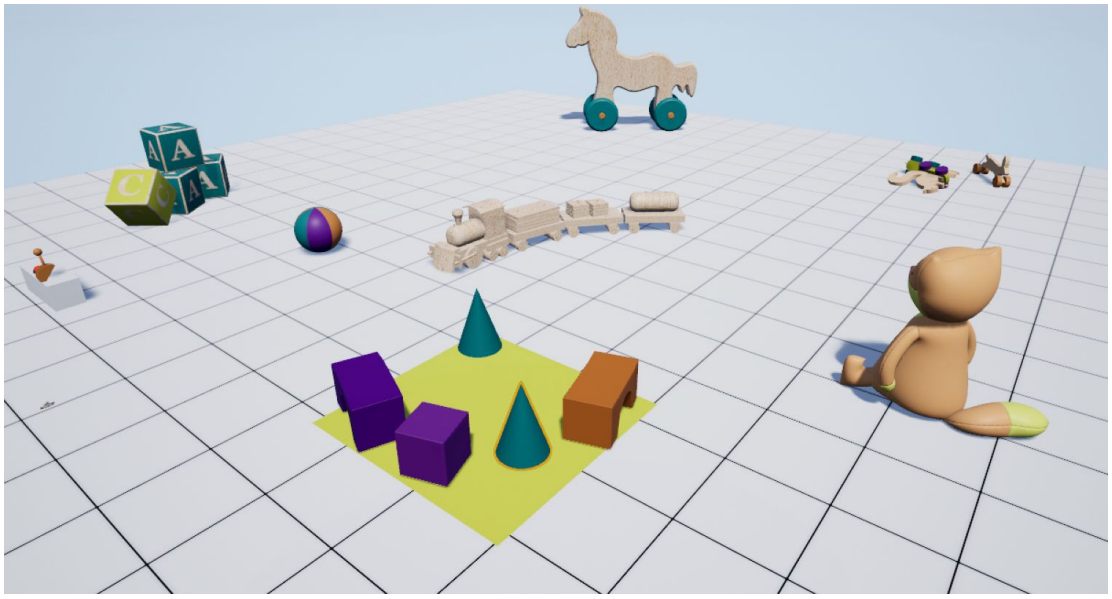
Figure 11 Toys' Game wrist UI (Game snap)



3.1.12 Level Structure, Design and Composition

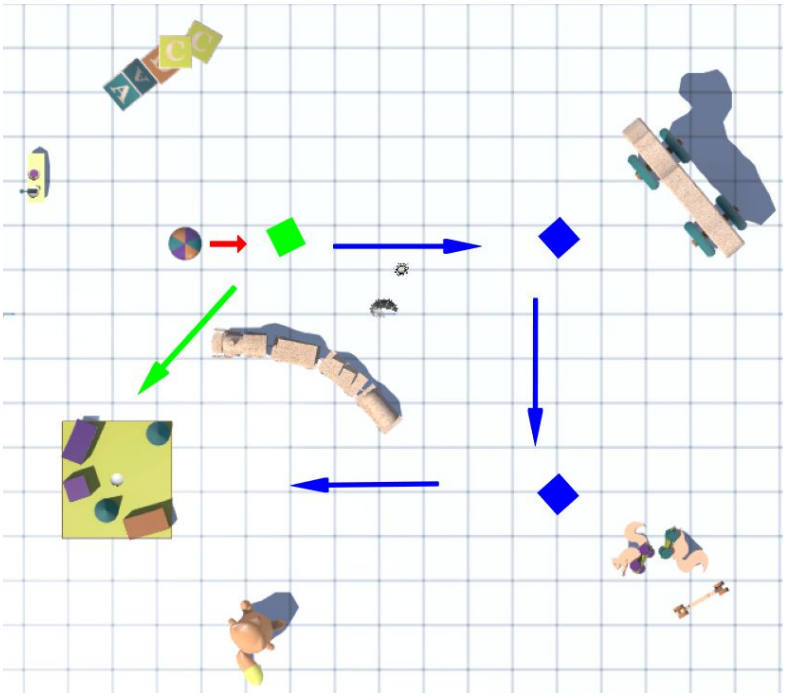
The level structure is similar to a child's disorganized room like Figure 12. All kinds of children stuff blocks the easiest path to goal. But some alternative paths are arranged to let the player build a path through them placing blocks. Although the player can place the blocks wherever he want, there are some clear and spaced areas to let the player place the blocks for redirect the ball easily. Also, there is a grid on the ground to help the player in his trigonometric calculations.

Figure 12 Assembled scene (Project snap)



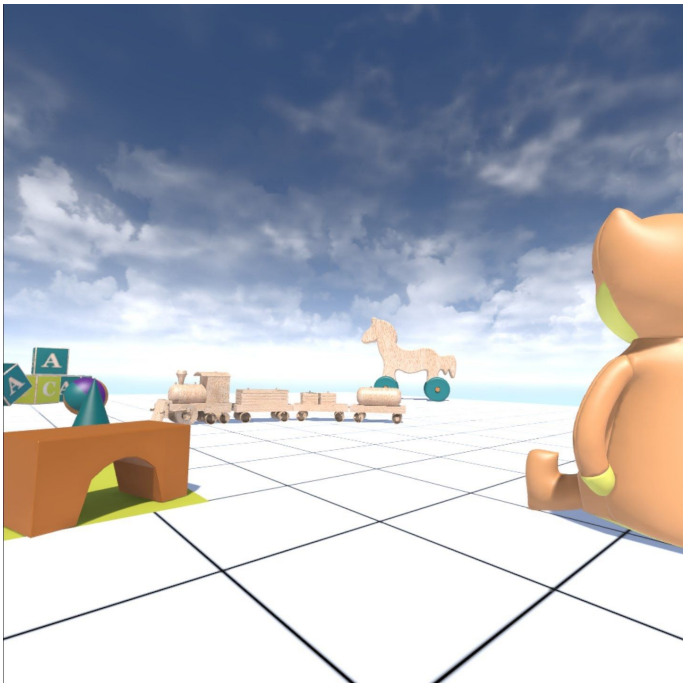
When the ball is thrown, always starts going in the same direction. That reduces the number of solutions. Figure 13 shows two different solutions in the same level. In this Figure, the arrows represent the direction of the ball and the boxes represents the blocks placed by the player. The red arrow shows the starting direction of the ball. The blue arrows and boxes represent the easiest solution path to reach the goal and the green ones the fastest but hardest solution path.

Figure 13 Level design (Project snap edited by the author)



All the objects in this world have real world size. But in the game, the player is tiny like a toy. More or less like a Lego figure. So the world and all the objects are scaled according to a toy point of view. Figure 14 is a good example of this kind of point o view and the world scale.

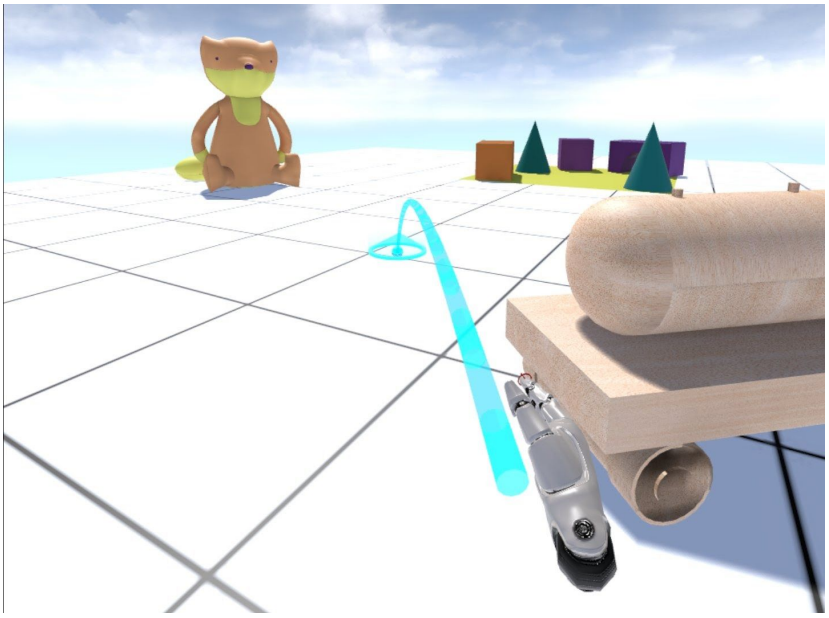
Figure 14 Player point of view (Video game snap)



3.1.13 Player Mechanics

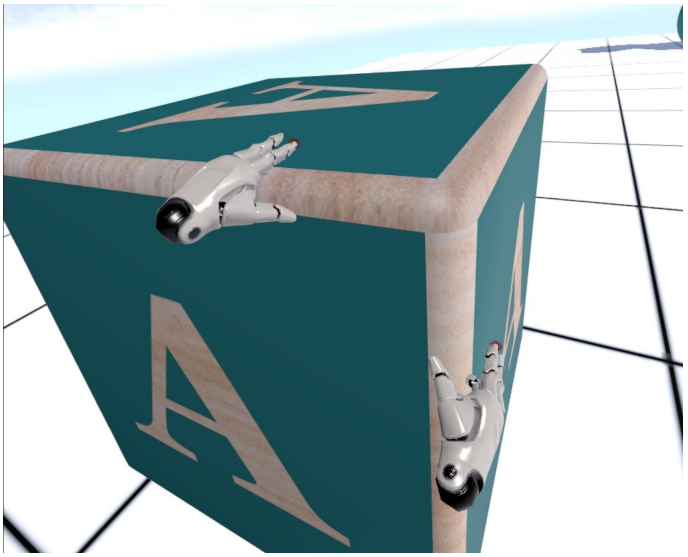
- **Teleport:** The player can move around the play area teleporting himself using the virtual reality standard "fishing rod". Figure 15 shows an example of a virtual reality standard fishing rod.

Figure 15 Toys' Game standard "fishing rod" (Game snap)



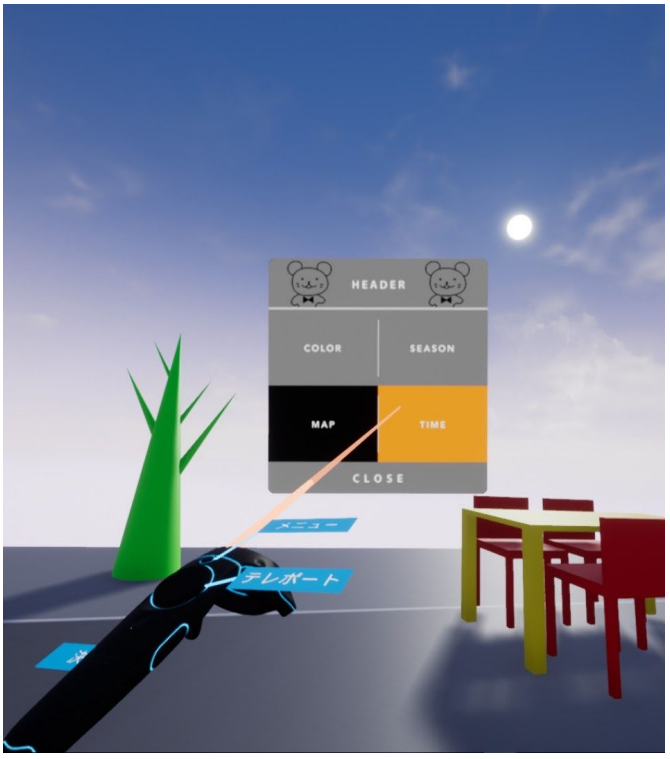
- **Grab:** Some objects are interactive. Can be grabbed, dragged and dropped using the controllers. Figure 16 show a virtual reality grab example.

Figure 16 Toys' Game grab example (Game snap)



- **Interact/select:** The player can interact or select with the UI elements using a pointer that is emitted from the controller. Figure 17 shows an example.

Figure 17 Virtual reality's UI laser interact example (Image reference)

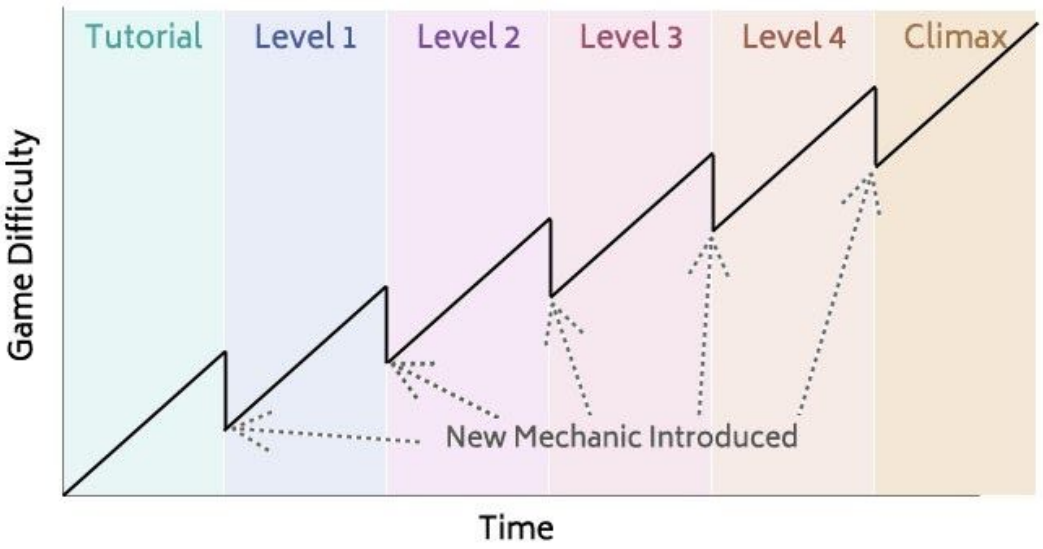


- **Block generation:** A limited number of blocks can be generated from a UI menu. Then these blocks can be used to change the path of the ball.
- **Throw power:** Via a lever, the player can adjust the thrust of the ball.
- **World rules:** In every level there are a limit of blocks that can be generated. The win condition is only to make the ball reach the goal. If after the throw, the ball stops before reaching the goal, it's a failed throw and the level starts again.

3.1.14 Game Balancing and Difficulty Characteristics

The difficulty curve is like a ladder. The first levels are introductory, and let the player learn the basics of the game. After that, the levels are a bit more difficult, forcing the player to think before throwing and try new things. The last levels are complex and let the experimented players have fun.

Figure 18 Ladder difficulty curve example (Image reference)



3.1.15 Game Instructions

When the game starts, the player is in the play area. Here, he needs to do a little exploring to find where the ball starts, and where the goal is. Once found, the player need to think which route will make the ball. Then, he needs to prepare the path, placing the blocks to help the ball reach the goal.

Once all is done, the player goes to the place where the throw lever is (normally in a higher place to see all the level) and sets the throw power. Then the ball is thrown. If the ball reaches the goal, the next level is loaded. If not, the level is reset and the player does the same procedure until the ball reaches the goal.

3.1.16 Sound and Music

This game is intended to be peaceful and without stress. With all the time in the world to prepare the path. That's why the background music is chill and with low beats. The world sounds are harmonic and peaceful. Intending to remember a childhood.

3.2 Troubles Found

The analysis of the project was a bit difficult due a couple of things. On the one hand we have that the virtual reality is something new, the software and the hardware is constantly evolving. That generates a situation of instability that makes it hard to find support or tutorials to realize projects. On the other hand, to develop a virtual reality project you need an expensive virtual reality lens and a mid-end/high-end computer with enough power to run this technology.

Also, the lack of experience playing these games makes it harder to find references and standards for the project.

3.3 Strategy to Realize It

To the hardware problem there were two options. The first one was use the high-end PCs and his virtual reality lens that the INIT lends to the students. The other one was use lenses and PCs from the external work placement. That analysis was before the Covid19 lockdown. When the lockdown started, and both places closed, there was no other option but to buy an *Oculus Quest* lens.

The software problem was easier to solve. In the external work placement, the work mates have a lot of experience in virtual reality and they helped to find documentation and tutorials to do this projects.

The lack of experience in virtual reality games was solved by doing some research and looking others opinions.

3.4 Possible Tools to Work

The current best engines to develop a project in virtual reality are *Unity* and *Unreal*. Each one has its own pros and cons. *Unity* is easier than *Unreal* to learn and use it. Also *Unity*s' rendering and computing time of the project is less than *Unreal* because it's simpler, but that means worse graphics than *Unreal*. In terms of versions, the support of *Unreal* for virtual reality projects was more stable than *Unity* during these

years. *Unity* has done a lot of different versions that makes incompatible older projects with the new ones. In terms of coding, *Unity* uses C# language and *Unreal* uses Blueprints.

Table 4 Engine comparisons

	Unity	**Unreal**
Graphics	Simple	Complex
Compiling time	Fast	Slow
Coding language	C#	Blueprints
Support on virtual reality	Instable	Stable

3.5 Tools Used

The following section shows the tools used in this project.

3.5.1 Project Tools

Since the *Oculus Quest* environment has been used, all tools have to be adapted to it. The list below are the tools that were used for this project:

- **Unreal Engine 4.24.3:** Is an engine that fits perfect in virtual reality due that proved during these years that have been stable in a matter of versions and have been made just a few changes since the start. That minds a perfect stability if we want to make changes in the future after the release.
- **Github Desktop:** It will be used for the version control of the project. It is easy to use and store *Unreal* projects.
- **Adobe Photoshop CC 2020:** Some textures will be created or modified for the project. With this professional tool, it is fast and easy make textures.
- **Oculus Desktop:** This software is necessary to set up and configure the *Oculus Quest* lens.

There was no need of a programming environment program because *Unreal* has its own one: the blueprint node programming.

3.5.2 Documentation tools

To have some flexibility during the documentation process and to work with different PCs, the documentation tools should be online and stored on the cloud. The list below shows the tools that were chosen for this:

- **Google Docs:** With his Microsoft Word version that works in .docx format, was perfect for the writing of the final report.
- **Google Sheets:** Ideal for the daily time record via tables and the planning hour control.

Due the video editing does not have an online tool yet, the Adobe Premiere CC 2020 has been used for creating the video of this project.

4. IMPLEMENTATION

The next section describes the project implementation process with the necessary technical details. Starts with a literature review, then the set up of the virtual reality environment and finally with all the elements and behaviours created for the game.

4.1 Previous Literature Review

The last four years the Virtual Reality industry has grown a lot. That's thanks to the new technologies emerged during these years that make possible a stable virtual reality interaction [1]. During these years, many new interactions have been created with virtual reality, both in the way of moving and interacting with the environment. Since virtual reality is constantly evolving, it still does not have many standards for setting a comfortable environment for the player. This new situations may be great and make the user have a lot of fun, but if the player gets motion sickness, it ruins the experience. In terms of movement of the player, some research point that the most comfortable way to move in virtual reality is teleport from one point to another doing a little blink as a transition (a fade in/fade out) [2]. Other options like moving from one point to another doing an interpolation or simulating to walk are more prone to create motion sickness [2]. Some experienced developers provide some best practices for avoiding motion sickness. They do not ensure you that it's the best way, but seems to be the first standards in the virtual reality industry [3] [4].

4.2 Set Up the Virtual Reality Environment

After the little research, the first step was to configure an *Oculus Quest* environment on the working PC to detect this kind of devices via USB. For this, an *Oculus* app was downloaded in a mobile for connect the mobile with the *Oculus Quest*

via Bluetooth and do the first set up of the lens [5] [6]. Then, using as well the mobile app, the developer mode was activated [5] [6]. Finally, the *Oculus* ADB Drivers [7] were installed on the PC for recognizing the *Oculus Quest* connected via USB.

The next step was to set up the engine *Unreal* with the correct android environment to develop projects for the *Oculus Quest*. To do this easily, NVIDIA Codeworks for Android was used to set up all automatically [8].

Finally, the last step was to set up a virtual reality project in *Unreal*. A new project was created based on the engine virtual engine template and the android support setting. Then the project was configured following the *Oculus* guide [9].

4.3 Build Time Reduction

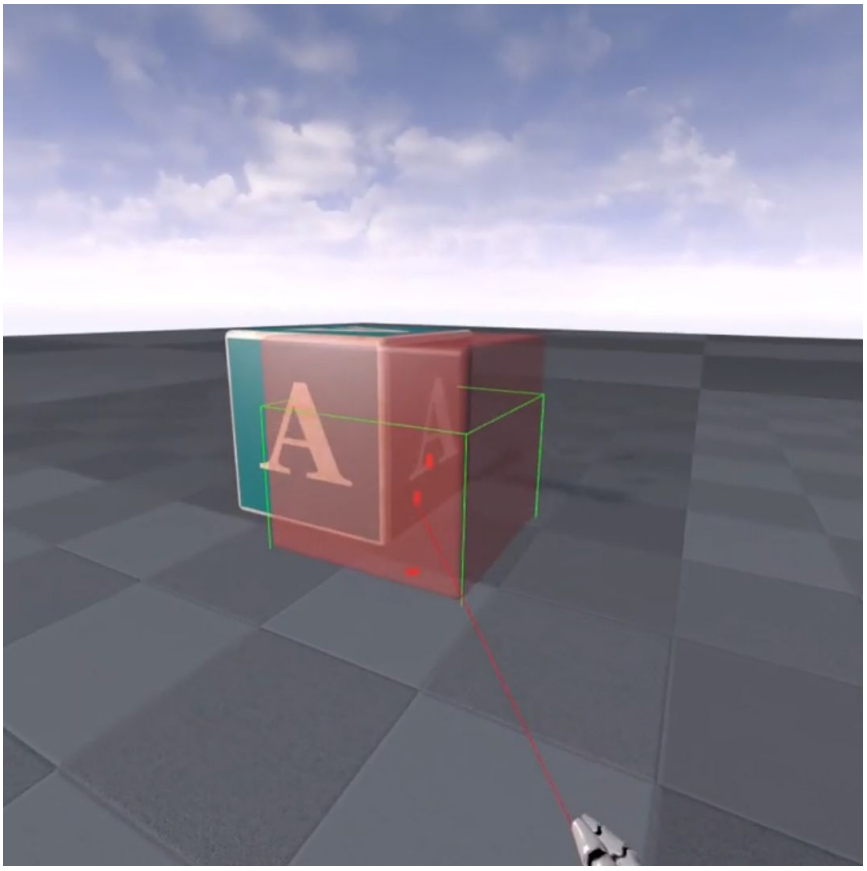
Every time new changes were added to the project, there was a need to build the application and installed in the *Oculus Quest* device. Since the build default times are about 10 to 15 minutes per build, and optimization was needed. To do this some settings of *Unreal* were deactivated or downgraded to reduce the build time using an online tutorial as a reference [10]. After that, the build time was improved to 3 min.

4.4 Building System

The system works as follows. When we press and hold the button that triggers the building system, a ray is thrown from the hand with the desired length. This ray it can be infinite or a certain amount of units. Even if the ray hits something or not, a preview of the model we want to build spawns at the end of the ray (if the ray hits something, the end is the hit point). While we hold the building button, the preview follows the end of the ray until we release the button. If this ray touches something, the hit point information is analyzed in addition to a box collider surrounding the preview object. This can lead to different situations. The preview uses the hit point as his center point to spawn unless the preview collides with something else. If the ray does not hit anything, the preview object it is spawned at the end of the ray. If the ray hits

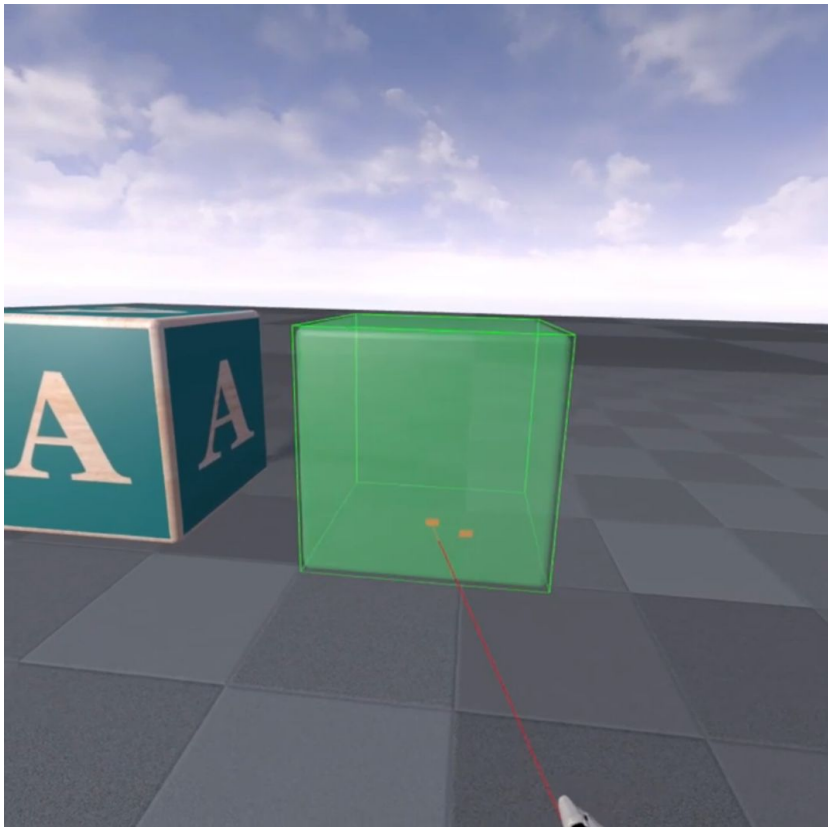
something, the preview object spawns at the hit point unless he is hitting the floor with his box collider, that in that situation, it spawns touching his base the floor, placing him on. The Figure 19 shows an example of this behaviour. The ray is hitting another block and the collider box is colliding with this block and the floor. So that leads to put the preview sitting on the floor, using the hit point plus an offset as the spawn point.

Figure 19 Building system: Unbuildable case (Project snap)



The preview object can be spawned with two different materials, both translucent. One is green like Figure 20, to sign that the object can be build in that place. The other is red like Figure 19, to indicate that this place it is not adequate for building. The preview object always spawns with the red material unless the ray hit point is touching the floor and the box collider of the preview object is not colliding with nothing else than the floor.

Figure 20 Building system: Buildable case (Project snap)



If the build button is released, an order to build is launched. If the conditions are favorable, the desired build object it is spawned in the place where the preview was. If not, the preview is destroyed and nothing happens.

An online tutorial and its code were followed as a reference [11]. By default, this build system was good, but it wasn't adapted for virtual reality. Also it does not fit with the original idea of this project. After doing the tutorial, some modifications were made to this code to work using the Oculus Quest controllers. Then, some other changes were added to make the build system more intuitive to the user.

Since new objects are placed in the scene and can be moved around, there was a need to update the navigation mesh of the map for evading teleport in places where there is a block already. After a search, the solution was in a tutorial where explain that the nav mesh can be rebuild in real time activating one option [12]. The problem was solved just with one click.

4.5 Bouncing Ball

After the build system, it was the turn for the physics. The idea was to add some physics to a simple sphere to simulate a ball that can bounce realistically with other objects.

To start, at the beginning the ball mass was modified to be 1 kg and each bounce of the ball subtract the 20% of his impulse forces. Having this as a central point, different physics damping were tested to have a realistic ball movement. The best fit was having a value of 0,015 units in the linear and in the angular damping. The final result of the damping was good.

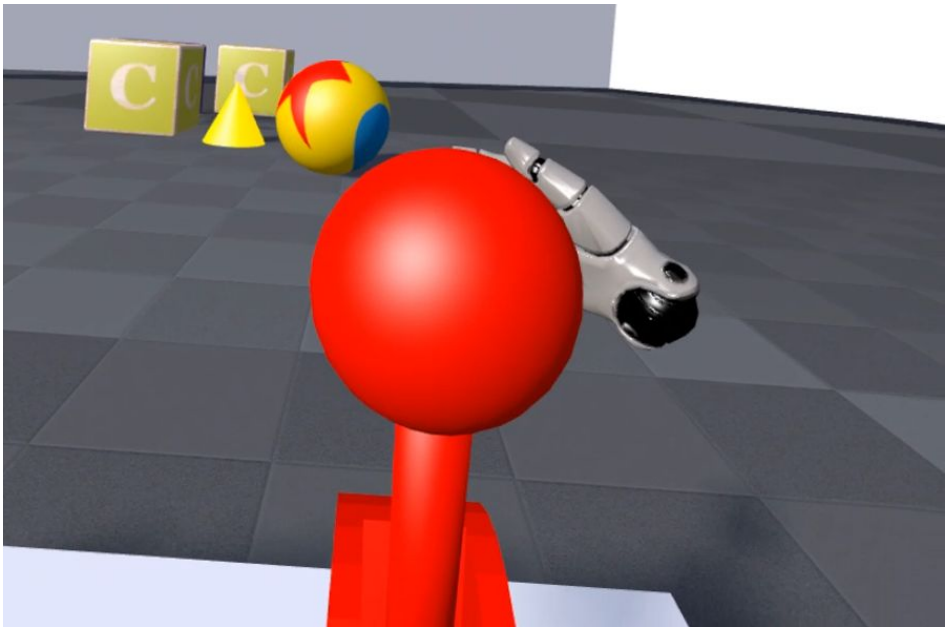
Once the ball has the correct physics set, a function was added to the ball blueprint. The function applies an impulse in the desired direction. This function can be triggered from other actors.

This bouncing behaviour were based on the physics system of *Unreal.* Some documentation were readed to make this physics and understand the physics system [13] [14]. Then a pair of tutorials were followed to have a template [15] [16]. The template was modified for adjusting the ball to the desired behavior for the project.

4.6 Virtual Reality Interactables

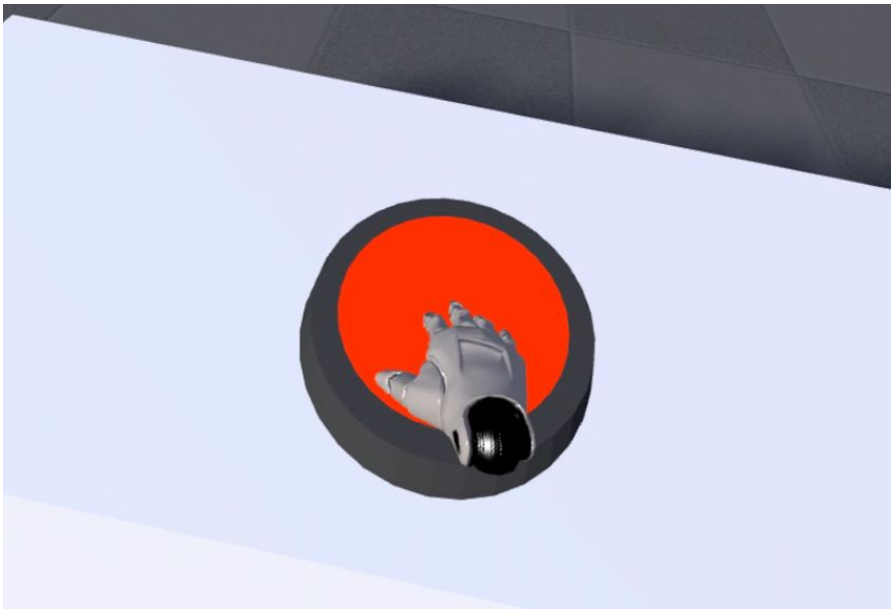
With the ball behaviors done, there was a need of having interactables to trigger them. The original idea was to use a lever to launch the ball, so a tutorial was followed to create a physics lever [17]. The final result, shown in Figure 21, was adapted to fit in virtual reality. It also was linked to the ball to trigger the launch if the lever was pulled. This actor was discarded because had a clunky user experience.

Figure 21 Lever (Project snap)



After this, a virtual reality button was created to have an alternative to this uncomfortable lever. The resulting button, shown in Figure 22, was more responsive than the lever, leading to a better user experience. A tutorial was followed as a starting point [18]. With some adjustments the button was more friendly to the user experience.

Figure 22 Button (Project snap)



4.7 Rotating the Camera

The default virtual reality movement from *Unreal* engine seemed poor, and a turn around mechanic was needed. To add this mechanic, a tutorial was followed [19]. The resulting behavior was good, but some features were added to improve it. The original behavior just turns around the camera 90º, using the Joystick X axis from the controller as a reference. So the first behavior added was turn 180º if the negative Joystick Y axis was triggered. The positive Joystick Y axis does not have a rotating behavior. The other feature added was restricting the rotation until the Joystick was not released. If this restriction does not exist, the user can trigger the rotate camera multiple times in a row holding the Joystick in one direction. This bug can lead to an uncomfortable experience and probably motion sickness.

4.8 Wrist UI

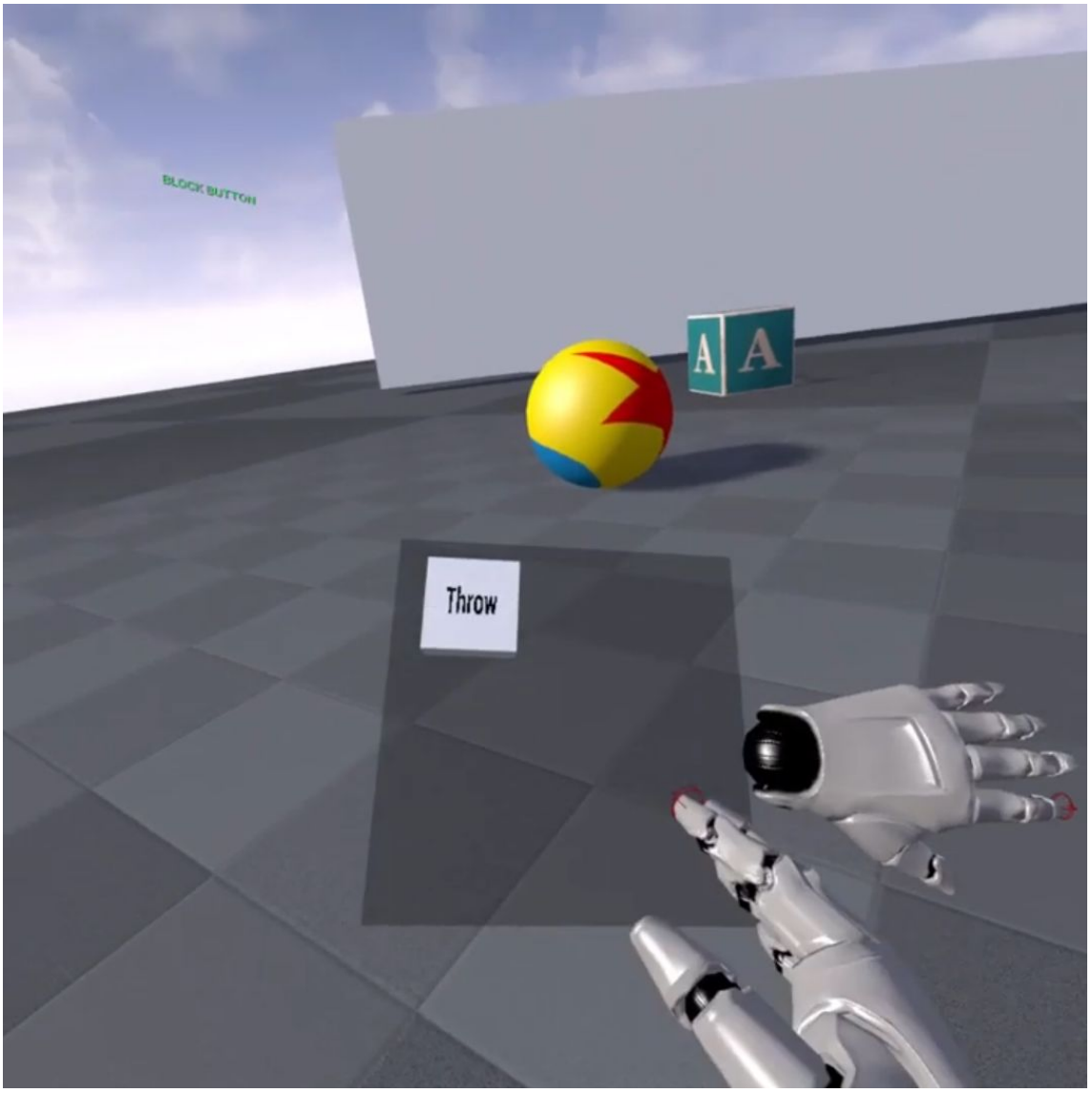
The next step was doing a UI that follows the left wrist like a watch.

The UI works as follows. When the user looks at his left wrist, the UI appears. It is visible until one second after the user does not look directly at the left wrist. Then it disappears until the user looks another time to his left wrist. When the UI appears, shows a semi translucent plane with a button on it. This button can be pressed with the index finger thanks to a sphere collider at the top of the finger.

An online tutorial and his code with similar behaviors were taken as a template [20]. The original behavior of this UI was adjusted to be more comfortable to the user, reducing the spawn time and repositioning better the sphere collider of the index finger. Also, the button was edited to trigger any desired functionality, in this case the button was setted to impulse the ball.

This wrist UI, his button and the sphere collider of the index are shown in the Figure 23.

Figure 23 Wrist UI (Project snap)



4.9 Environment

Firstly, 3D Toy models for the scene were searched in an Internet marketplace [21]. After gathering some models, there was a need of chose a color palette for the game aesthetics. Most of the models have wood material, so the palette should be complementary to this material. After trying different combinations of colors in a color scheme designer web [22], the scene colors were decided. This palette is shown in Figure 24.

Figure 24 Project color palette

The next thing done was create the child blocks. To do this, a block mesh from *Unreals* basic assets was taken as a template. To start, the static mesh editor of *Unreal* was studied [23]. Then, the block mesh was edited with this editor for apply it more easily different materials and simplify his texture. Finally, four different textures were created using Photoshop and the edited block UVs. The block UVs and one of his textures created are shown in Figures 25 and 26.

Figure 25 Block UVs

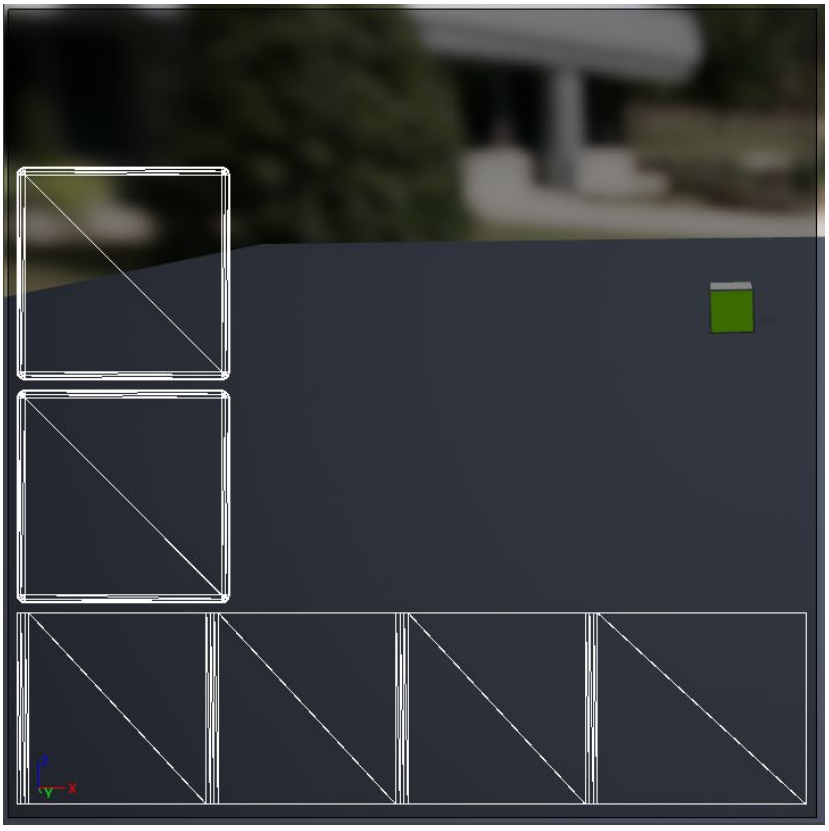


Figure 26 Block A texture

Each time a block is spawned, one of the four created textures is randomly applied to him. These four different blocks are shown in Figure 27.

Figure 27 Child blocks

A similar process was done for texturing the ball. His UVs were taken as a reference and a texture was created in Photoshop.

Finally, the scene was assembled with all the 3D models to seem a child disorganized room. All the models were placed strategically to make an obvious circuit for the player. After that, the models were colored with the palette colors and textured with wood textures. The Figure 12 presents a point of view of the assembled scene.

5. RESULTS

This chapter describes the objectives reached, compare the original schedule with the resulting one and justify the project deviations.

5.1 Objectives Reached

In section 2.1 of this document, the objectives to reach of this project were presented. Two of these three objectives proposed has been accomplished.

The resulting project has been an immersive experience and the typical errors of the virtual reality that leads to bad user experiences have been avoided. So the first objective was reached.

The second objective was also accomplished. Thanks to the hard work and the dedicated hours in the project an intermediate knowledge of *Unreal* engine was reached in a short time.

The current project has a robust mechanics individually, but in group, they do not work harmoniously. That leads to a bad gameplay to the player, so the third objective was not accomplished.

5.2 Planning Comparison

In section 2.2 of this document, an initial planning was presented to confront the development of the project, however, at the time of being undertake it has been modified. Tables 5, 6 and 7 show the comparison between the hours estimated and the actual work hours done. Table 8 compares the summary planning.

Table 5 Documentation time comparisons

Documentation		
Tasks	*Estimated duration (in hours)*	*Actual duration (in hours)*
Technical proposal	10	10
Technical report	40	70
Video	5	5
Project defense presentation	5	5
Total	**60**	**90**

Table 6 Game design time comparisons

Game design		
Tasks	*Estimated duration (in hours)*	*Actual duration (in hours)*
Gameplay and mechanics	9	7
Level design	7	5
User interface	4	1
Total	**20**	**13**

Table 7 Game development time comparisons

Game development		
Tasks	*Estimated duration (in hours)*	*Actual duration (in hours)*
Set up the virtual reality environment	10	7
Physics	30	21
Gameplay and mechanics	70	113
Environment	30	20
User interface	20	10
Debugging	60	46
Total	**220**	**217**

Table 8 Summary planning time comparisons

Summary table		
Tasks	*Estimated duration (in hours)*	*Actual duration (in hours)*
Documentation	60	90
Game design	20	13
Game development	220	217
Total	**300**	**320**

5.3 Project Deviations

The project has suffered some modifications from its initial idea. The first technical proposal was also a virtual reality video game, but with some differences. This proposed video game was an experience similar to playing with a doll house. In there, the player interacts with his companion and the mini world around him via the virtual reality controllers. The idea was to have an “alive” world that has its own behaviour.

The artificial intelligence was ripped apart because does the project too big and unreachable with only 260 work hours. With this part deleted, the main idea of the project was reworked. The idea was limited to a simple mechanics to learn properly from scratch how to do this interactions. So the idea changed to a first person game where you have the basic movement and a set of simple mechanics.

6. CONCLUSIONS

One of the biggest problems of the virtual reality for the new users is the motion sickness. Many video games in the market are categorized as uncomfortable because they are prone to provoke motion sickness. The developers best practices recommendations help to avoid motion sickness in most of the cases, but there is still a need of a scientific explanations in each case. Doing a video game with the basic movement mechanics helps to avoid the motion sickness, but also limits the creativity of the designers and this leads to limit the fun.

Nowadays, it is more easy do a virtual reality project thanks to the advances of the visualisation and tracking technology and the software that provides a good development environments. *Unreal* engine seems to be leading this thanks to his integrated virtual reality support and his templates for start.

This document presents a virtual reality puzzle video game that intends to cover the lack of comfortable environments in the market. The video game implements the best practices recommended and has a soft and paused gameplay that is needed to avoid the motion sickness. Also has a main mechanic to solve different puzzles. This means that the challenge of creating a virtual reality puzzle video game with a comfortable environment that avoids the motion sickness has been fulfilled.

7. FUTURE WORK

It has been thought several expansions of this project as a future work that are depicted at the following sections.

7.1 Project Expansion

This project will be used as a template in order to elaborate new mechanics and satisfy the author concerns. The work done will be used in future projects or taken as a reference for new ones. Also, the project will be improved and uploaded to itch.io video game community.

7.2 Future Research

With the knowledge gained during the realization of this project and the information gathered, in-depth research will be done in a future to expand the author professional knowledge.

8. REFERENCES

[1] Wikipedia. *Virtual reality* [Last Access 29/05/2020]

[2] Kasra Rahimi, Colin Banigan, and Eric D. Ragan (2018 November 30). *Scene Transitions and Teleportation in VirtualReality and the Implications for SpatialAwareness and Sickness*.

[3] Unreal Engine Docs. *Virtual Reality Best Practices* [Last Access 1/06/2020]

[4] Developers Oculus. *VR Design Best Practices/Locomotion* [Last Access 1/06/2020]

[5] Support Oculus. *First steps with Oculus Quest* [Last Access 3/06/2020]

[6] Play Store. *Oculus app* [Last Access 3/06/2020]

[7] Developers Oculus. *Oculus ADB Drivers* [Last Access 3/06/2020]

[8] Developer Nvidia. *Nvidia Codeworks for Android* [Last Access 3/06/2020]

[9] Developers Oculus. *Unreal quick start* [Last Access 3/06/2020]

[10] Youtube. *Optimizing UE4 projects for reduce the Oculus Quest build time* [Last Access 3/06/2020]

[11] Youtube. *Build system for Unreal engine* [Last Access 3/06/2020]

[12] Youtube. *UE4 Navmesh Rebuild Realtime* [Last Access 3/06/2020]

[13] Unreal Documentation. *Collisions overview* [Last Access 5/06/2020]

[14] Unreal Documentation. *Physics damping* [Last Access 5/06/2020]

[15] Youtube. *UE4 Physics: Bouncing ball* [Last Access 6/06/2020]

[16] Youtube. *UE4 Physics: Use physics material* [Last Access 6/06/2020]

[17] Youtube. *UE4 How to make Lever Radial* [Last Access 7/06/2020]

[18] Youtube. *VR interactables from scratch* [Last Access 7/06/2020]

[19] Youtube. *How to rotate VR Pawn* [Last Access 7/06/2020]

[20] Youtube. *VR interactive watch menu* [Last Access 12/06/2020]

[21] CGTrader. *Free Toy 3D Models* [Last Access 14/06/2020]

[22] Paletton. *The Color Scheme Designer* [Last Access 12/06/2020]

[23] Unreal Documentation. *Modify static mesh geometry* [Last Access 12/06/2020]

- Scott Rogers (2010): *Level Up: The Guide To Great Video Game Design.* United Kingdom: Chichester
- Scheme created from *https://www.lucidchart.com*
- Gantt chart created from *https://www.canva.com*
- Toy Story 1995 © Pixar
- Tin Hearts 2018 © Rogue Sun
- Oculus Quest 2019 © Facebook
- Unreal 1998 © Epic Games