



# **Serious Game Concepts in the Development of a Video Game with VR Technology in Unity**

**Araceli Cazorla González**

Final Degree Work

Bachelor Degree in

Video Game Design and Development

Universitat Jaume I de Castelló

June 27, 2019

Supervised by: Juan Miguel Vilar Torres, PhD.



For my parents,  
who always lend me a hand in everything I needed,  
for Alberto,  
who encouraged me in the hardest moments,  
and for Antonio,  
who gave me the boost I needed in many bumps in this degree.

## **Abstract**

Nowadays, VR video games are filling malls and entertainment locals. Also, little by little VR video games are reaching homes and schools, allowing casual players to live new experiences that traditional console games can not offer.

In this document, Bone Playhouse is shown. Bone Playhouse is the alpha project and the research necessary to a bigger project that will allow one or two players to play at the same time a serious game about learning how something is built, in this case human anatomy, with virtual reality technology. The objective is to make the players learn something new, with a double purpose: getting the players more motivated due to the unusual game, and making them assume the concepts with more staying power than in traditional learning techniques.

This project is meant to be executed with the Leap Motion technology, that will allow the players to evade controllers or touchscreens, and let them to use their own hands. Moreover, it is intended to be executed with the virtual reality headset HTC, although due to the flexibility of the game engine Unity3d, other glasses could be compatible in the alpha phase.

Due to the scale of the project, throughout this document it will be shown how the different parts of the development of the game are executed, with the purpose of showing the academic progress of the student in all the areas of the degree.

## **Key words**

Serious game, VR, Anatomy, Immersion.

<b>TECHNICAL PROPOSAL</b>	<b>9</b>
1.1 Introduction and project motivation	9
1.2 Context analysis	10
1.3 Related subjects	10
1.4 Task and temporal scheduling	11
1.5 Project objectives and expected results	12
1.6 Tools and environments	12
1.6.1 Programming	12
1.6.2 Art	12
1.6.3 Documentation	13
1.7 SDK	13
1.8 Resources evaluation	14
1.9 Risk Management	14
<b>GAME DESIGN DOCUMENT</b>	<b>15</b>
2.1 Introduction	15
2.2 Entities	15
2.2.1 Players	15
2.2.2 Human model	16
2.2.3 Bones	16
2.2.5 Hands	17
2.3 Gameplay	17
2.3.1 What does the player	17
2.3.2 Tutorials	17
2.4 Game experience	18
2.5 Visual style	18
2.6 Mechanics	19
2.7 Game modes	19
2.8 Scenery design	19
2.8.1 The Sims	19
2.8.2 Toy Story 3 The video game	19
2.8.3 Among the sleep	20
2.8.4 Ikea	20

2.9 Sound and music	20
<b>PROJECT DEVELOPMENT</b>	<b>21</b>
3.1 Previous research	21
3.1.1 Research about serious games	21
3.1.2 Research about anatomy	22
3.2 Game development	23
3.2.1 Adding the project to Github	23
3.2.2 General information classes	23
3.2.2.1 Created for the project itself	23
3.2.2.2 Used from Leap Motion libraries	24
3.2.2.3 Used from Photon libraries	24
3.2.3 Game modes and difficulties	24
3.2.3.1 Solo	24
3.2.3.2 Cooperative	25
3.2.3.3 Free	25
3.2.3.4 Against the clock	25
3.2.3.5 Easy	25
3.2.3.6 Medium	25
3.2.4 User interfaces	26
3.2.4.1 UI Main menu	26
3.2.4.1.1 Solo - Cooperative	27
3.2.4.1.2 Free - Against the clock	27
3.2.4.1.3 Easy - Medium	27
3.2.4.2 UI Bones	27
3.2.4.3. UI Hands	28
3.2.4.3.1 Spawning buttons	28
3.2.4.3.2 Menu button	29
3.2.4.3.3 Unlock button	29
3.2.4.3.4 Help button	29
3.2.5 Multiplayer and network connection	29
3.3 Art	30
3.3.1 2D Art	30
3.3.1.1 Tileable textures	30
3.3.1.2 Handmade textures	31

3.3.1.3 UI Sprites	32
3.3.2 3D Art	33
3.3.2.1 Previous design	34
3.3.2.2 First steps of modelling	34
3.3.2.3 Fixing the topology	34
3.3.2.4. Unwrapping the models	35
3.3.2.5 Scenery	36
3.3.2.6 Organic models	37
3.3.2.6.1 Rabbit	37
3.3.2.6.2 Skeleton	37
3.4 Testing	39
3.4.1 Sanity check	39
3.4.2 Functional test	39
3.4.3 Free testing	40
<b>RESULTS OBTAINED</b>	<b>41</b>
4.1 Game Development	41
4.1.1 Game modes	41
4.1.2 User Interfaces	41
4.1.3 Network	42
4.2 Art	42
4.2.1 2D Art	42
4.2.2 3D Art	42
4.3 Testing	43
4.4 Documentation	43
4.5 Final planification	44
<b>NEXT OBJECTIVES</b>	<b>45</b>
<b>CONCLUSIONS</b>	<b>47</b>
<b>BIBLIOGRAPHY</b>	<b>49</b>

## **List of Tables**

[Table 1: Task and temporal scheduling](#)

[Table 2: Gantt diagram](#)

[Table 3: Programming tools](#)

[Table 4: Art tools](#)

[Table 5: Documentation tools](#)

[Table 6: SDK](#)

[Table 7: Risk Management](#)

[Table 8: Different types of bones](#)

[Table 9: UI Button Sprites](#)

[Table 10: Scenery](#)

[Table 11: Bones](#)

[Table 12: Test Results 1](#)

[Table 13: Test Results 2](#)

[Table 14: Test Results 3](#)

[Table 15: Element recount](#)

[Table 16: Final planification result](#)

## **List of Figures**

[Figure 1: HTC Vive](#)

[Figure 2: Leap Motion](#)

[Figure 3: Base bones](#)

[Figure 4: Definitive bones](#)

[Figure 5: Error bones](#)

[Figure 6: Score panel](#)

[Figure 7: Leap Motion Controller](#)

[Figure 8: Text in-game](#)

[Figure 9: Hand painted style](#)

[Figure 10: Bone icons](#)

[Figure 11: The sims reference](#)

[Figure 12: Toy Story Reference](#)

[Figure 13: Among the sleep reference](#)

[Figure 14: Ikea 1 reference](#)

[Figure 15: Ikea 2 reference](#)

[Figure 16: Handmade anatomy sketches](#)

[Figure 17: Ulna Differences between hand moves](#)

[Figure 18: UML classes diagram](#)

[Figure 19: Game Object Skeleton](#)

[Figure 20: Starting game modes](#)

[Figure 21: UIButtons](#)

[Figure 22: First design of bone spawn buttons](#)

[Figure 23: Final design of bone spawn buttons](#)

[Figure 24: Bad tileable texture](#)

[Figure 25: Example of how the tileable tool works](#)

[Figure 26: Good tileable texture](#)

[Figure 27: 3D Result of tileable texture](#)

[Figure 28: Handmade textures](#)

[Figure 29: Handmade textures for a wall](#)

[Figure 30: Handmade bed design](#)

[Figure 31: Preliminary bunny draw](#)

[Figure 32: Bone model](#)

[Figure 33: Rabbit rigged](#)

[Figure 34: Room 3D Art](#)



## Contents

---

1.1 Introduction and project motivation	12
1.2 Context analysis	13
1.3 Related subjects	13
1.4 Task and temporal scheduling	13
1.5 Project objectives and expected results	14
1.6 Tools and environments	14
1.7 SDK	15
1.8 Resources evaluation	16
1.9 Risk Management	16

---

This chapter will reflect what is the origin of the idea and which resources are needed for the development of the project from a technical point of view.

### 1.1 Introduction and project motivation

This project is based on the creation of a tridimensional video game prepared to be executed by one or two players with virtual reality technology (See figure 1).

Players will stand up while playing the game. In front of them, they will find a empty human body that they should fill with bones in the correct places.

The objective of this serious video game [6] is teaching children and teenagers how anatomy works in a practical way.

Since the target of this game are children, it is important to show a friendly scenery that motivates them to keep playing. Skeletons, organs, and anatomy in general is a taboo topic, specially in some cultures. That is why a cartoon style, hand-painted with post-process effects will be used, looking for a comfortable environment.

It will be possible also to use Leap Motion technology [4] (See Figure 2). For that reason players can move their hands in the tridimensional space and interact with the environment, in order to solve the different minigames. This means that the immersion will be bigger than in other VR projects where the input is a controller.



**Figure 1: HTC Vive**



**Figure 2: Leap Motion**

It is also important to mention that players can cooperate between them thanks to the network connection with Photon plugin [3] that will allow them not only to interact with the same virtual room where their partners are, but also see their actions in real time.

## 1.2 Context analysis

This project was born from the idea of developing video games that could teach somebody how something is built in the inside. It is a common issue that students lose interest in the subject because it is hard to learn so many concepts and naming off different things from a book. That is because this project is aimed on keeping players learning while they play.

Furthermore, this project will be realized with VR technology because the immersion and concentration in VR games is usually higher. The shape of the bones, the names and the positions are easier to remember if the player can move and place them with their own hands in a virtual world where he is involved.

There are two problems that we have to take care of:

- **Education games are not attractive:** we have to design the game properly to create something interesting even for a child.
- **Dizziness and headaches:** in experiences of this type, consumers usually get tired, dizzy and sometimes have headaches and nausea, due to different reasons:
  - **The activity duration:** that usually takes a long time.
  - **Proximity:** the screen is too close to the eyes.
  - **Framerate:** it is necessary to be at 60fps minimum.
  - **Motion sickness:** dissonance between what the vision is capturing and what other senses, like balance, are interpreting.

## 1.3 Related subjects

The following subjects have direct impact on this project:

- VJ1227 – Game Engines.
- VJ1223 – Video Game Art.
- VJ1228 – Multiplayer Systems and Networks.
- VJ1216 – 3D Design.

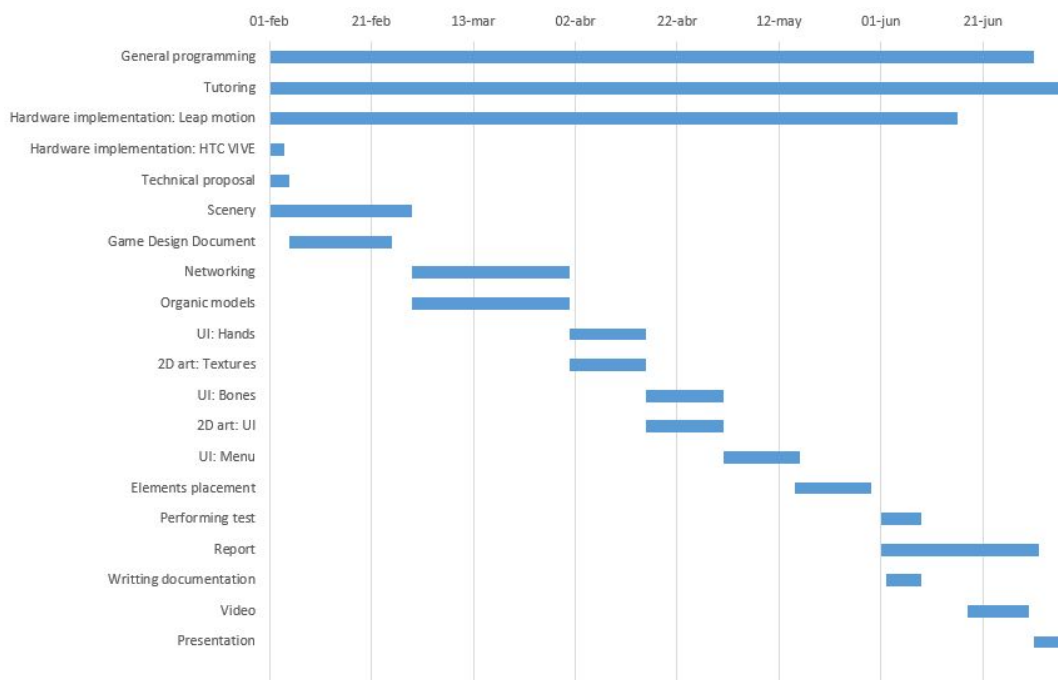
## 1.4 Task and temporal scheduling

In the next schedule we can see that this project has four main components that need to be completed: Game Development, Art, Testing and Documentation. This project is thought to take three hundred hours.

Code	Task	Dates	Time planned
<b>Game development</b>			
GD01	General programming	1st February to 1st July	15
GD02	Hardware implementation: HTC VIVE	1st February to 3th February	20
GD03	Hardware implementation: Leap motion	1st February to 15th May	15
GD04	Elements placement	1st May to 15th May	20
GD05	Networking	1st March to 31th March	25
GD06	UI: Hands	1st April to 15th April	19
GD07	UI: Bones	16th April to 31th April	17
GD08	UI: Menu	1st May to 15th May	12
<b>Art</b>			
A01	Scenery	1st February to 28th of February	42
A02	Organic models	1st March to 31th March	30
A03	2D art: Textures	1st April to 15th April	8
A04	2D art: UI	16th April to 31th April	5
<b>Testing</b>			
T01	Performing test	1st Juny to 8th July	10
T02	Writing documentation	2nd Juny to 8th July	5
<b>Documentation</b>			
D01	Technical proposal	1st February to 4th February	3
D02	Game Design Document	5th February to 25th February	10
D03	Report	1st Juny to 1st July	30
D04	Tutoring	1st November to 12th July	6
D05	Presentation	1st July to 8th July	3
D06	Video	18th Juny to 1st July	5
<b>TOTAL</b>			300

**Table 1: Task and temporal scheduling**

In the table below, you can see the Gantt diagram:



**Table 2: Gantt diagram**

## 1.5 Project objectives and expected results



- As many gaming modes as possible (bones, organs, muscles), at least one of them.
- One stage totally decorated.
- Total immersion.
- Challenging gameplay
- Well-designed project where dizziness will not be a problem.
- Project that can involve 2 players.
- To develop a project for everyone, where friends can play cooperatively.
- Possibility of moving hands inside the project, in order to interact with other elements.
- Bonus: add different gameplay modes, or buffs for the players.

## 1.6 Tools and environments

In this section we can find the tools that will be used during this project.

### 1.6.1 Programming



Below you can see the programming tools used during this project.


Unity 3D [1]	
This is the main tool used for the game development, along with Visual Studio. It is a free game engine that supports HTC Vive [5], Photon and Leap Motion. Also, it is prepared for Blender FBX and 2D elements. All these features made this engine the chosen for this project.	
Visual Studio	
Integrated development environment. It is used to develop computer programs and accepts many languages, as C#, needed in this project. With Unity 3D, is the main core of the development of the project.	

**Table 3: Programming tools**

### 1.6.2 Art

Below you can see the art tools used during this project.




Blender [2]	
It is the main tool used for the 3D modelling. Free and open-source 3D software used for creating 3D models, organics and inorganic. Models are exported in FBX.	
Adobe Photoshop	
Raster graphics editor used for creating the UI sprites and some textures used on the 3D models and the logo.	

<b>Krita [7]</b>	
<p>Raster graphics editor used for creating most part of the 3D textures. It has many pencils and options for creating pictures where the hand-painted style is present.</p>	

**Table 4: Art tools**

### 1.6.3 Documentation


Below you can see the documentation tools used during this project.


<b>Google Docs and Google Slides</b>	
<p>The main tools used for creating the Technical proposal, Game Design Document, Final Report and Final Presentation. With this tool users can create and share their files through Google Drive platform. It is a free Web-based application.</p>	
<b>Microsoft Excel</b>	
<p>Is a spreadsheet program included in the Microsoft Office suite of applications. Spreadsheets present tables of values arranged in rows and columns that can be manipulated easily. I will use this program for creating my testing reports.</p>	
<b>Vegas Pro</b>	
<p>Is a video editing software package for non-linear editing (NLE) originally published by Sonic Foundry, then by Sony Creative Software, and now by Magix. The software runs on the Windows operating system. I will use this program for creating the final video.</p>	

**Table 5: Documentation tools**

### 1.7 SDK

Below you can see the SDK that will be used during this project.

<b>Photon Unity Networking [3]</b>	
<p>Unity package for multiplayer games. It allows to get a flexible matchmaking where players can get into rooms where objects can be synced over the network.</p>	

Leap motion	
Unity package for games with AR/VR technology. It allows the project to use the hands of the player as the main controller.	

**Table 6: SDK**

## 1.8 Resources evaluation

For the creation of this project it will be needed some hardware equipment:

- **HTC Vive with all the elements (600€):** other VR glasses could be compatible, as Oculus Rift, for example. Because as today I have access to this model, HTC vive is the option.
- **Leap motion (80€):** this game could be played with mouse, HTC Vive controllers, keyboard or gamepad controller. Leap motion was chosen because it creates more immersion.
- **A PC capable to handle the items above (800€):** The minimum requirements are GTX 970, Intel i5-4590, 4GB ram.

Also, a software license will be needed for: Adobe Photoshop (120.95€), Vegas (299€) and Microsoft excel (135€).

Regard to the human resources, for this alpha project only a person will be needed for the programming, art, and production. In the future, at least it will be necessary to have a person specialized in each field, or even more.

Considering that the average wage of a programmer is 11€ per hour, an artist 10€ per hour, and a producer 12€/hour, the results of the hours dedicated are:

- **Programmer:** 11€x143h = 1573€.
- **Artist:** 10€x85h = 850€.
- **Producer:** 12€x72h = 864€.

The total cost of this project is 5321,95€.

## 1.9 Risk Management

The number of hours planned are 300. However, this schedule (See table 7) is not fixed, because some unexpected issues could come up and change the program. To anticipate the possible problems, this section includes some of the possible issues that could appear containing them.

ID	Name	Probability	Impact	Effect	Containment plan
R1	Miscalculate task durations	High	High	Some of the features could not be completed and the game could not be finished as planned	Focus on the most important task of the project
R2	Lack of skill in Leap Motion technology	High	High	The main point of the project could not be finished	Invest some time in investigation
R3	Lack of skill in general game programming	Medium	Medium	Some task could take more time than expected and other tasks could be punished by that	Invest some time in investigation
R4	Data loss	Low	Medium	Some files can get lost	Start the project with a git repository

**Table 7: Risk Management**

# CHAPTER 2

## GAME DESIGN DOCUMENT

### Contents

---

2.1 Introduction	19
2.2 Entities	19
2.3 Gameplay	20
2.4 Game experience	21
2.5 Visual style	21
2.6 Mechanics	21
2.7 Game modes	21
2.8 Scenery design	21
2.9 Sound and music	23

---

This section will show the details about the project related with gameplay, design and art.

## 2.1 Introduction

Bone Playhouse is a 3D video game, with VR model, cooperative, with cartoon style. It's genre is serious game-minigame, and its objective is to beat the minigames that are focused on making people learn Anatomy.

In those minigames, the player can create, move and place bones in an human model. The objective is to fill the human model with all the bones in the correct places. Different rules are stated for achieving the objective, depending on the difficulty and game mode.

## 2.2 Entities

### 2.2.1 Players

People who are playing the video game. Their objective is to play the game and beat the minigames while learning about anatomy in the process. The target public is children between 8-10 years which are starting to learn how the human body works and teachers who want to play with them. It is thought to have two kind of players:

- **Learners:** the players that wants to learn about anatomy.
- **Teachers:** the players that accompany the learners in the game.

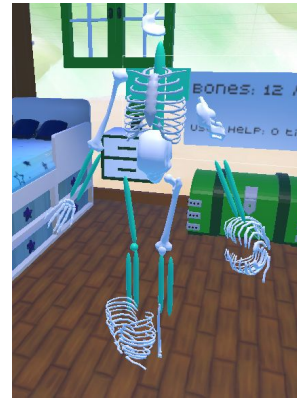
Learners could also play together in competitive modes, like Against The Clock game mode.

### 2.2.2 Human model

The human model that has to be filled with bones. It is an static 3D model formed by base bones, definitive bones and error bones.

At the start of the game, all the base bones are activated (See Figure 3). As the game avances, the bones are substituted by the definitive bones (See Figure 4) or error bones by the player (See Figure 5).

Depending of the game mode, the player can place wrong bones in the base positions. When a bone is not well placed, the model give clues to the player. (See Figure 5).







**Figure 3: Base bones      Figure 4: Definitive bones      Figure 5: Error bones**

### 2.2.3 Bones

The elements that should be placed in the model. We can find twenty five bones in the game, fifteen of them are different from each other.

There are four kind of bones:

- **The spawned bones:** these bones are the ones that are spawned. Players can grab, move and place these bones. Fifteen types of bones.
- **The base bones:** these bones are in the human model. They are placed in the 3D model from the beginning as a placeholder and can be substituted. Twenty five types of bones.
- **The definitive bones:** these bones are in the human model. They are placed in the 3D model and only appear when the base bones or error bones are substituted. Twenty five types of bones.
- **The error bones:** these bones are in the human model. They are placed in the 3D model and only appear when the base bones or error bones are substituted. Twenty five types of bones.

Spawned	Base	Definitive	Error
			

**Table 8: Different types of bones**



### 2.2.4 Score panel

It shows crucial information needed for the game (See Figure 6). There are three fields, that are visible or not depending on the game mode and difficulty:

- **Bone counter:** it shows how many bones are placed.
- **Timer counter:** it shows how much time is left to lose the game.
- **Help counter:** it shows how many times the button Help has been pressed.

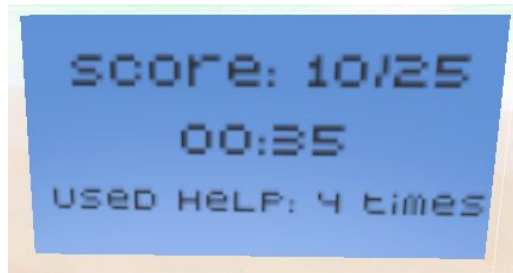


Figure 6: Score panel

### 2.2.5 Hands

It is the controller for the game (See figure 3). Players can interact with the video game using their own hands. This is possible thanks to Leap Motion Technology [4], which throws infrared rays to the hands in order to show them in game.

The main ways to interact with other elements are grabbing bones and touching the UI.



Figure 7: Leap motion controller

## 2.3 Gameplay

### 2.3.1 What does the player

Players main goal will be finish the game mode they choose, and place all the bones in their correct places, with the rules given by the minigame. If two players are playing, then they must play the same game mode in order to play together.

To win the minigames, they should create bones, grab, move and place them in the correct places.

### 2.3.2 Tutorials

Tutorials will appear as text in the game, depending on the game mode that the players choose. Also, a Security Rules text will appear at the beginning of the game (See Figure 8).



**Figure 8: Text in-game**

## 2.4 Game experience

It is expected that some players will be inexperienced in VR video games. For that reason, minigames should be well designed for everybody.

We hope that the player will be completely involved in the game. That is why the interface is completely in-game: they are not 2D images that follow the camera, but interactable 3D models.

The interface is very intuitive and easy to use. We want any player who can get used to the game easily. For that reason, every button has an icon and a label.

The general feeling that we want to make the player sense is relax. That is why the music and the environment helps to accomplish this objective.

## 2.5 Visual style

It have stylized, hand painted 3D models (See Figure 9). This style is suitable for project, because it is thought for a young public target.

In order to carry it out, we are mainly using 3D low-poly models. Some normal and occlusion maps will be used as a reinforcement if it is necessary to remark lines or illumination.

The sprites created for this project are mainly the bone icons, solid images that creates a silhouette of the shape of the bone (See Figure 10).



**Figure 9: Hand painted style**



**Figure 10: Bone icons**

## 2.6 Mechanics

The main mechanic consist in creating bones and placing them into a human body. Depending of the game mode (easy or medium), the player can or not lock and expulse the bone.

Once an element is locked, if the player wants to remove it, he has to use a special tool for that. This tool will be placed in the right hand.

In addition, the player has a tool if he needs aid for solving the minigame.

## 2.7 Game modes

Different game modes will be implemented:

- Solo/Cooperative: one or two players can play
- Free/Against the clock: the player can play with freedom or with a timer as a challenge
- Easy/Medium: it changes if the bones could be placed in incorrect places or not

## 2.8 Scenery design

We need to pay special attention for this scenery, because it is where players will pass the most part of their time, and should not be limited or boring.

The room will have a lot of elements. In order to not to associate it to a gender, the walls and general colours will be pink, blue and green.

### 2.8.1 The Sims

The first picture is very useful because it show a palett colour very similar to the scenery in the project. Also, the staircase of the bed is useful. In the second picture we can see some elements hand-painted, like the dinosaur or the plushies. The bookshelf, dinosaur and chest are interesting.



Figure 11: The sims reference

### 2.8.2 Toy Story 3 The video game

These references are useful because the furniture has round corners. The room does not have a specific palette colour, but there is not any element who doesn't fit. The environment is densely decorated. It is important to remark the desk and the dartboard.



Figure 12: Toy Story Reference

### 2.8.3 Among the sleep

These references are useful because it shows a room from a very low point of view. This will be the point of view used in the project. It is important to highlight the cubes.



**Figure 13: Among the sleep reference**

### 2.8.4 Ikea

In the first picture we can remark the stool. In the second, the staircase and basketball court could be useful.



**Figure 14: Ikea 1 reference**

In the first picture the furniture with three colors and de carpet are interesting. In the second, the wall painting is beautiful.



**Figure 15: Ikea 2 reference**

## 2.9 Sound and music

All the sound in the video game while the game is on will be funny, childish, and chill. We have three sound effects and one background song, all of them are CCO:

- **Place bone:** <https://bit.ly/2REgtKR>
- **Create bone:** <https://bit.ly/2XfrJ6g>
- **Menu and buttons:** <https://bit.ly/2Xgzmtl>
- **Background song:** Ukulele beach from <https://bit.ly/1wwplPp>

# CHAPTER 3

## PROJECT DEVELOPMENT

### Contents

---

3.1 Previous research	24
3.2 Game development	25
3.3 Art	36
3.4 Testing	49

---

In this section I will show the progress during this project in the two different main points: development and art. In addition, I will talk a little about the testing sessions I did for this project and the previous research

### 3.1 Previous research

#### 3.1.1 Research about serious games

During the first weeks, I studied the concept of serious games. This kind of games are designed for two purposes: entertainment and education. The concept of “education” could be very large, because a serious game could be also related to engineering, politics or science, but in this project, we will use the basic education concept.

The use of serious games has been rising these days. The "game based-learning" is finding his own space in schools over time.

That is why this project is a serious game. I liked the idea of creating a project where you can have fun and learn something new. Mixing it with VR, could give it an attractive, innovative and immersive extra that makes the children being interested in something that it is usually not.

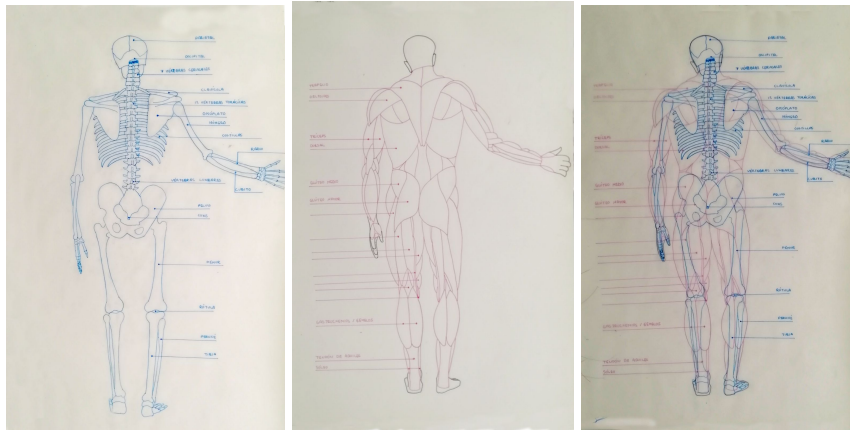
I got inspired in these two serious games:

- **Surgeon simulator:** in this game the player has to save a patient, beating minigames and placing all the organs in the body in a limited time. I like the idea of placing elements in a body from this game. Video: <https://bit.ly/2FCbqpm>
- **Cat explorer:** in this game the player can see the insides of a cat, using his hands as controller. in this game I like the idea of explore the anatomy of a cat. Video: <https://bit.ly/2PcR3m2>

### 3.1.2 Research about anatomy

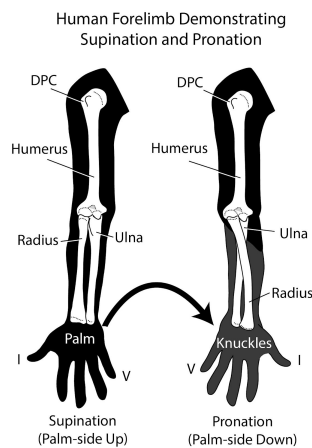
During the first weeks, a study about anatomy was done, specially using “An Atlas of Anatomy for Artist” [9]. This was necessary not only for knowing the details of all the bones that will be included in the game, but also because it is important to know how these bones are connected to each other and move with the human dynamics.

In order to understand better how bones work, some sketches were made by hand. This helped me to have a mental picture of how each bone is connected to other, and what muscles are involved in the process (See Figure 9).



**Figure 16: Handmade anatomy sketches**

One example is the strange move of the Radius and Ulna (See Figure 10). When the arm is rotated, the position of the bones changes completely. Placing the bones in a wrong place will be a big mistake, if I want to make other people learn with my game.



**Figure 17: Ulna Differences between hand moves**

All of this details were taken in mind in order to create a didactic game where mistakes were not an issue.

## 3.2 Game development

### 3.2.1 Adding the project to Github

The first step that I took when I created this game was creating a repository for the project. I found that Github has a plug-in for making the commits easier in the program, so I imported it into my project.

This tool was found in Window - Github. Once it was open, the process was simple, as in console: the changes were shown in the commit screen, and push sent to my internet repository.

### 3.2.2 General information classes

In this section of the document we can find the basic information of the scripts used in this project:

#### 3.2.2.1 Created for the project itself

- **GameManager:** This script is the connection between many scripts. Not only for the networking part, but also for the skeleton itself. It is inside an gameObject placed in the scenery.
- **Info:** Each bone have this script. It is necessary for storing information about the bone.
- **BoneButtons:** This script is used in the bones that are placed at first. Its main objective is showing the buttons for the Medium Mode.
- **HandButtons:** This script is used in the buttons of the hands. Its main objective is to manage the spawning of the bones.
- **UIButtons:** This script is used in the main menu buttons. Its main objective is to give the **GameManager** the correct selections of the player.
- **Sustitute:** This script is used in the bones that are placed at the start in the body. It manages its behaviour depending on the difficulty. Also, it makes a UI appear when a bone is touched.
- **NetworkManager:** This script creates the version and room of the network settings. Also, it manages when a player joins to a room and adds it to a list.
- **NetworkPlayer:** This script keeps the connection between the player and the server. Also, makes the movement fluent.
- **AgainstTheClock:** This script manages the Against the clock game mode. It has a Timer. Also, it manages the counters in-game.
- **Trash:** This script is used for destroying a bone when it collides with an specific object.
- **DestroyBone:** This script is used for destroying a bone after some time.
- **Movement:** basic movement for the player in multiplayer mode.

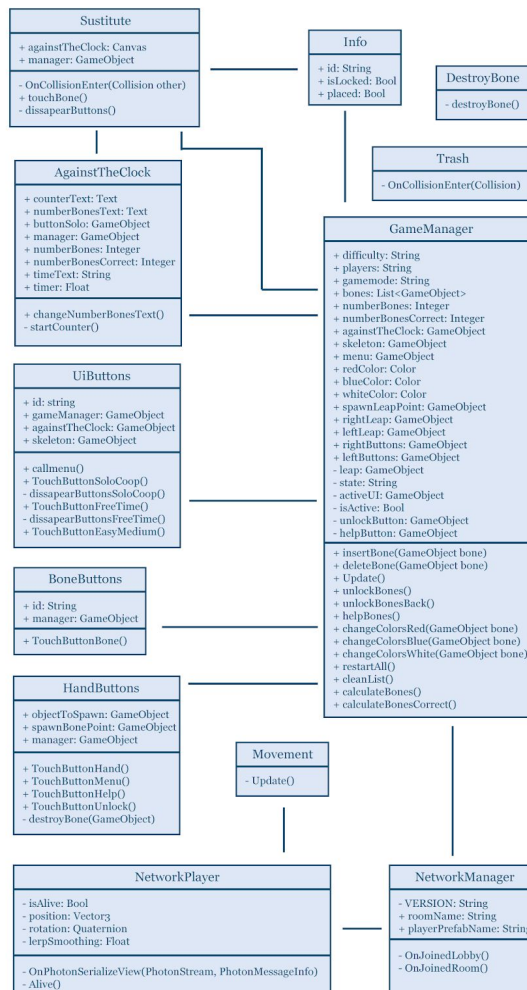


Figure 18: UML classes diagram

### 3.2.2.2 Used from Leap Motion libraries

- **InteractionBehaviour:** This script manages the interaction between hands and objects.
- **InteractionManager:** This script manages that the Leap Motion hands are being tracked. Each object that is interactable needs a reference of this manager in the **InteractionBehaviour** component.
- **InteractionButton:** This script manages that the Leap Motion hands can interact with buttons.
- **SimpleInteractionGlow:** This script makes a small animation in order to let the player know that the button that he is touching is being pressed.
- **SimpleFacingCameraCallbacks:** This script makes possible to see the hand UI while the player is looking at the palm of the hand.

### 3.2.2.3 Used from Photon libraries

- **PhotonNetwork:** main script that makes possible to create and joint to rooms.
- **PhotonServerSettings:** This script stores the settings of the project.
- **PhotonView:** This script makes the gameObject visible for other players.

## 3.2.3 Game modes and difficulties

### 3.2.3.1 Solo

In this game mode the player plays alone.



### 3.2.3.2 Cooperative

In this game mode the player plays in a cooperative mode. Will be explained in the multiplayer section.

### 3.2.3.3 Free

The Free mode is thought to be focused on children that want to learn and take time to pay attention to all the details in the bones and the names.

### 3.2.3.4 Against the clock

This mode is thought to be a challenging game mode where the player can prove their improvements. The player has 2 minutes for placing everything in their position, or he will lose.

The script ***againstTheClock*** and the function ***void startCounter()*** are the managers of this mode.

We have two important gameObjects in this game mode. The first is ***numberBonesText***, that shows up a string with how many bones are placed in that moment. The second gameObject is ***counterText***, that shows up a string that contains the timer.

***numberBonesText*** is updated in ***changeNumberBonesText()*** function, whereas ***counterText*** is updated in ***startCounter()*** function.

In addition, when the player loses, the game will show up a “You lose!” in the ***numberbonesText*** gameObject.

### 3.2.3.5 Easy

In the Easy mode, the player can place the bones in their places without penalty. If he tries to place it in a wrong place, the bone will not collide and will not be replaced. If he tries to place it in the correct place, the definitive bone will appear and the base bone will disappear.

To achieve this, each time that the function detects that a gameObject is colliding and it is the correct one, it deactivates the base bone and activates the definitive bone.

The script used for replacing bones in both difficulties is ***Sustitute***. With an ***void OnCollisionEnter(Collision other)***, we check if something is trying to collide with the body.

Before leaving the function, we call the ***void insertBone()*** function in ***GameManager*** script. This function adds the bone to a ***List bones*** and incrementates the variable value of ***numberBones*** by one. Also, it calls ***void changeNumberBonesText()*** that changes the ***numberBonesText*** variable in the ***AgainstTheClock*** script.

To finish the process, the bone spawned is destroyed.

Bones can be destroyed in two ways: placing them in a destroyer item, calling ***destroyBone()*** function in ***DestroyBone*** script, or waiting a minute after its creation, with ***destroyBone()*** function in ***HandButtons*** script.

### 3.2.3.6 Medium

The Medium mode works in a pretty similar way. The player can make mistakes and place bones in every position allowed. To achieve this, each time a bone collides with another, five possibilities are checked and the activated elements in the hierarchy (See Figure 17) are changed.

- **The base bone is activated and the bone spawned is correct:** the base bone (first in hierarchy) is deactivated and the definitive bone (second in hierarchy) is activated. ***numberBonesCorrect*** and ***numberBones*** is incremented, and a bone is added to the ***bones*** List.

- **The base bone is activated and the bone spawned is incorrect:** the base bone is deactivated and the error bone (fourth in hierarchy) is activated. The mesh of the error bone changes.
- **The definitive bone is activated and the bone spawned is incorrect:** the base bone is deactivated and the error bone is activated. *numberBonesCorrect* is decremented.
- **The error bone is activated and the bone spawned is correct:** the definitive bone is activated and the error bone is deactivated. *numberBonesCorrect* is incremented.
- **The error bone is activated and the bone spawned is incorrect:** the mesh of the error bone changes.

All this comparisons are checked with the collider in the collider element (fifth in the hierarchy) and the *id* in the *Info* script in the parent (skull in the example of the Figure 17).



**Figure 19: Game Object Skeleton**

This hierarchy and this way to proceed, enabling and disabling gameObjects or rendering components could be complex to understand, but it is the best way that I found for keeping the positions, rotations, meshes and components in the objects. Its a solid system once it is completed.

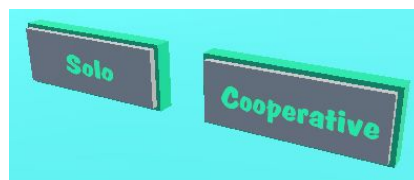
The game is easier for the player if he has a tool that allows to him to lock bones that are correct for him. Also, if this tool allows to expulse bones that he thinks that is not correct. That is why we need to spawn two buttons in each bone. This issue will be explained later.

### 3.2.4 User interfaces

In this project we find three user interfaces that are the bridge between the game and the player. All of them are made in 3D, with some labels on the top.

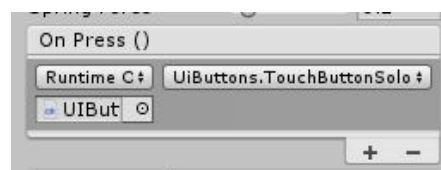
#### 3.2.4.1 UI Main menu

At the start of the game we find a menu where the player can choose the mode that he wants to play (See figure 18).



**Figure 20: Starting game modes**

This menu is programmed in the script *UIButtons*, and the call of the functions is made in the *InteractionButton* component (See Figure 19).



**Figure 21: UIButtons**

In all the cases the code works the same. When a button is pressed, a function is called. This function deactivates the buttons, and activates the next two with a coroutine.

In addition, the function sends to the **GameManager** the information given by the player. For achieving this, the button pressed gives its **id** to the **GameManager**.

There is a specific function in this Script, **called void callMenu()**. This function is the responsible for calling back the menu when necessary, that is, when the user presses the main menu button on his right hand or when the player finishes a level.

I found a problem developing this part of the project. When I pressed a button, the next two were pressed immediately too. That is why it was impossible to select the options I needed. The solution that I found was making the next two buttons wait three seconds to appear in the coroutines functions **dissapearButtons()**.

#### 3.2.4.1.1 Solo - Cooperative

In this first decision the player can decide if he want to play alone or with a friend. The functions called in each case are **void TouchButtonSoloCoop()**.

For making these buttons disappear, we call **IEnumerator dissapearButtonsSoloCoop()**.

#### 3.2.4.1.2 Free - Against the clock

In this decision the player can decide to play the “Free” game mode or the “Against the clock” game mode. The functions called in each case are **void TouchButtonFreeTime()**.

For making these buttons disappear, we call **IEnumerator dissapearButtonsFreeTime()**.

#### 3.2.4.1.3 Easy - Medium

In this last decision the player can decide to play in the easy mode or the medium mode. These functions called in each case are **void TouchButtonEasyMedium()**.

These buttons do not need a coroutine for calling new buttons.

In this function we also check if the gamemode selected in the “Free” / “AgainstTheClock” decision was “AgainstTheClock”. If that is the case, we call the function **void startCounter()** in **againstTheClock** script.

Also, if the difficulty decided is “Easy” the buttons help and unlock are activated.

#### 3.2.4.2 UI Bones

When we touch a bone in the Medium mode, a UI shows up with two buttons.

The first button is called Lock: it makes the bone not interactable anymore. For achieving this, it changes the variable **isLocked** in **Info** script. Everytime we collide a bone with a hand or with a bone, we check if the variable **isLocked** is not false for continuing the process. After pressing Lock, we make the UI inactive again. The second one is Expulse, it makes the bone recover its base form. After pressing the Expulse button, we make the UI inactive too.

To achieve this, we have a function called **void TouchButtonBone()** in the **BoneButtons** script. This function is called in the component **InteractionButton**, inside of each button, and makes the info and the behaviour of the bone change.

This UI had problems at the beginning, because more than one UI was activated at the same time and it was confusing for the player. This is why I created a **isActive** bool in **GameManager**, that checks if an UI is activated when other UI is called.

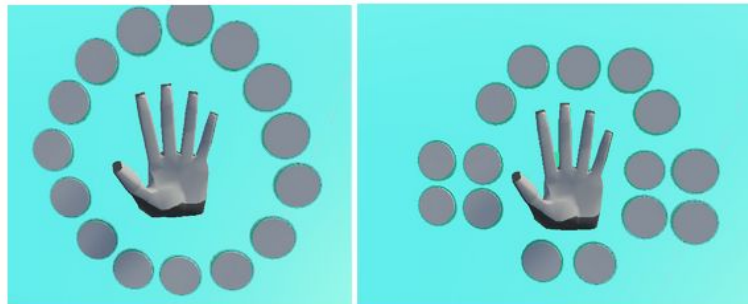
These buttons disappear automatically with **dissapearButtons()** function in **Sustitute** script.

In the case that a player decided to lock a bone, and regrets it later, he can press the unlock button in the right hand, that will be explained later.

### 3.2.4.3. UI Hands

#### 3.2.4.3.1 Spawning buttons

These buttons can spawn bones and are in the left hand. There are twenty five kinds of spawned bones in this project, and fifteen of them are different. That is why we have twenty five kind of spawn bones. At first, this kind of organization was decided (See figure 20):



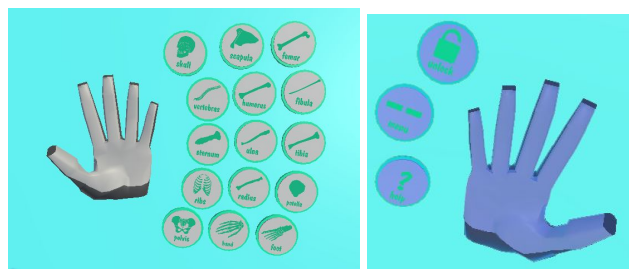
**Figure 22: First design of bone spawn buttons**

This two models were not very useful because the buttons on the left part of the hand are hard to press with the right hand. It is necessary to cross arms and it is not comfortable.

Also, due to Leap motion works with infrared, if we cross hands the left hand sometimes disappears.

That is why I decided to place the buttons in the left part of the hand (See figure 21). It is less beautiful to the eye, because it is concentrated, but is the most useful and intuitive way to play the game.

With the right hand, I followed the same schema (See Figure 21)



**Figure 23: Final design of bone spawn buttons**

When we touch a button in our right hand, a bone spawn up in a specific spawning position, in a gameObject transform called **SpawnBonePoint**. There, we call the function **void TouchButtonHand()**, located in the **HandButtons** script and we instantiate the correct bone depending on the identifier of the button itself.

At the beginning I found an issue related with these new bones spawned. The **InteractionManager** was not automatically placed, because it was public and dragged to the inspector. But it seems that when Unity instances a prefab it loses its dependencies. That is why I placed the component of the **InteractionManager** in the script just after creating the bone in running time.

Other issue found with this buttons was that everytime that the game was running, lots of bones were spawned without reason. I realized that the bones were spawning by the left hand itself. After checking some documentation, I found that it was possible to make these buttons only interactable for one of both hands, and I solved the problem.

### 3.2.4.3.2 Menu button

This button returns you to the main menu. You can press it anytime you want and find it in the right hand. We call the function ***void callMenu()*** in the ***UiButtons*** script.

With the script ***GameManager*** and the function ***restartAll()*** we restart the variables, players, gamemode and difficulty, and we clean the ***bones*** List.

### 3.2.4.3.3 Unlock button

Sometimes the player can decide that a bone is well placed, and lock it. But maybe later he changes his mind. He can press the unlock button, unset those bones that are incorrect, and press again the unset button to go back to the normal mode.

You can find it in the right hand. For achieving this result, we use the component ***InteractionButton*** in the button itself. There, we call the function ***TouchButtonUnlock()***, located in the ***HandButtons*** script, that calls the ***unlockBones()*** function in the ***GameManager*** script. There, we check with a loop all the elements in bone list and we change the color of the bones locked with ***void ChangeColorsRed()*** function in ***GameManager*** script.

After that, if the player press again the Unlock button, the bones are changed to white with ***unlockBonesBack()*** in ***GameManager*** script

### 3.2.4.3.4 Help button

Sometimes the player may need help. This is why we make the correct buttons change to blue, in order to give a clue to them.

You can find that button in the right hand. We use the component ***InteractionButton*** in the button itself. There, we call the function ***TouchButtonHelp()***, located in the ***HandButtons*** script, that calls the ***helpBones()*** function in the ***GameManager*** script. There, we check with a loop all the elements and we change to color blue the correct bones, calling the ***ChangeColorsBlue()*** function in ***GameManager*** script.

## 3.2.5 Multiplayer and network connection

This game is designed to be player in a multiplayer mode. Both of the players have an own HTC Vive [5] visor and Leap Motion [4] controller, and are playing in different PC.

In order to achieve this point, it was investigated how can we make two projects connect each other, and tried to get a fluent result where lag is not a problem.

The first step was importing the Photon Networking Plugin [3] to the project. Once this is done, the next step is setting up the server, creating an account in the website and logging in.

In my account, I found an app id, so it was time to add it to the project with the region. Also, at this point I selected the EU Region and I clicked Save.

At this point I created the two scripts that are necessary for making the communication possible: ***NetworkPlayer*** and ***NetworkManager***.

We added a Photon View to the player and we dragged his transform to the component ***ObservedTransforms***, so the position will be changing and will be visible for other players.

In the inspector, I created a ***spawnPoint*** for creating the players.

The ***NetworkManager*** script is not very complex. On the ***Start()***, we called the ***ConnectingUsingSettings(string version)*** for connecting my project with my settings to my Photon Server. It uses the version of my game.

In this script we can also find an ***OnJoinedLobby()*** function, that creates a room with specific options. It checks if a room is created, if that is the case, it tries to join the room. If not, it creates one.

We also have a void ***OnJoinedRoom()*** function. This one add the players that joins to the room to a List of players, and instantiate a gameObject ***PlayerPhoton*** for each player.

By its side, in the ***NetworkPlayer*** script we can find a void ***OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)***, that is the responsible for keeping the position and rotation updated all the time in the server. For doing that, it uses an stream object.

We also have other function, ***Alive()***, that checks if the user is still in the room. Once all of this was prepared, the player was ready for communicating with other players through internet. It worked, and I saw how two players can move and rotate in the same scene.

The problem came when I tried to add this functionality to Leap motion hands. Due to Leap motion being created procedurally, it is not easy to add this component to it. It is not formed for different parts with different transforms, as other gameObjects.

On internet I found that many people had the same problem as me. I tried to use other hands, that were supposed to be more prepared for physics projects, but it did not work either. It seems that it is not as easy issue at it seems and needs more time to be fixed, changing some code from the libraries of Leap Motion.

For finishing this part of the project, I realized that the gameObjects that were moving through internet were not very fluent. This is due to the information not being transferred constantly, it has a small delay being received. I investigated on internet how other games solve this problem and I found that some of them create an interpolation between the last reception of information and the last one before that one. That is how with a Lerp() function I can make a more fluent communication between gameObjects.

## 3.3 Art

### 3.3.1 2D Art

In this section of the document I will talk about how the textures, 2d animations, and walls were made. All of them were created with Krita or Photoshop.

#### 3.3.1.1 Tileable textures

To create this textures, Krita was a important factor. Tileable textures should be well created if we want to simulate a texture that can be connected in all of his parts.

In this example, we can see a bad tileable texture (See Figure 22). When we place it repeatedly, it is not well connected and the result is not satisfying.

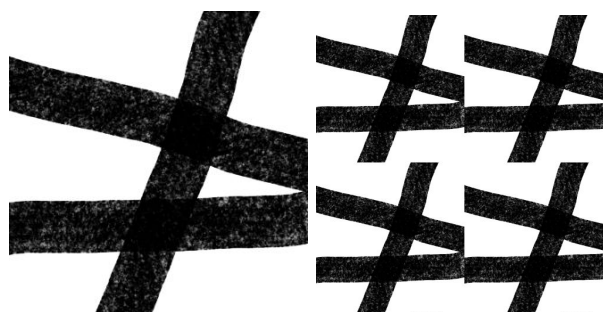
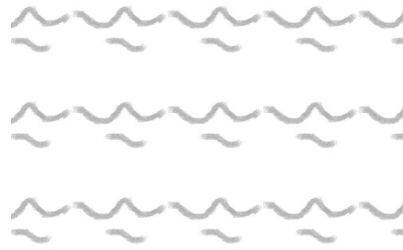


Figure 24: Bad tileable texture

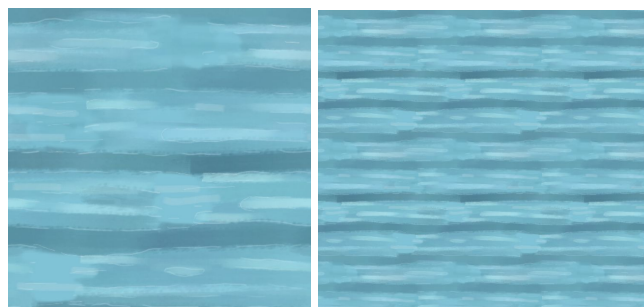
This is due to the ending part of the lines are not connecting to the beginning of the other ones.

Krita have a tool that makes this easy to do, so the artist do not have to worry about the final result a lot (See Figure 23). He can see the progression of the texture in real time.



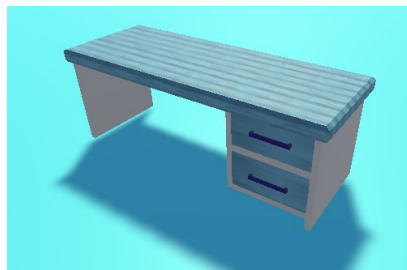
**Figure 25: Example of how the tileable tool works**

Using this tool with some patience, this texture was made for the desk (See Figure 24):



**Figure 26: Good tileable texture**

As we can see, the result is well connected. If we add it in a 3D model, we give it a extra point to stylized style (See Figure 25).

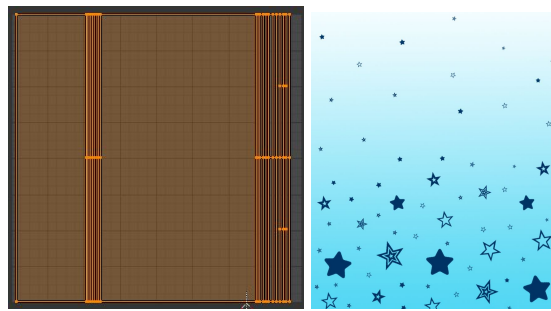


**Figure 27: 3D Result of tileable texture**

### **3.3.1.2 Handmade textures**

Other textures were made by hand, for a specific model. That is the case of the bed, where we can see that the stars are different from each other.

For doing this texture, the first step were creating a good topology for the object and extracting a UV Map. After that, we add it to a painting program (Photoshop in this case, but Krita is good too) and paint it by hand (See Figure 26).



**Figure 28: Handmade textures**

In the last step, we added it to the model in Blender. When the result is good and well placed, the model is ready to be exported in FBX to Unity.

Other 3D models with custom textures were made with the same basis.

It is important to mention the textures created for the walls, because they have a lot of details and are made for faking painted paper (See Figure 27).

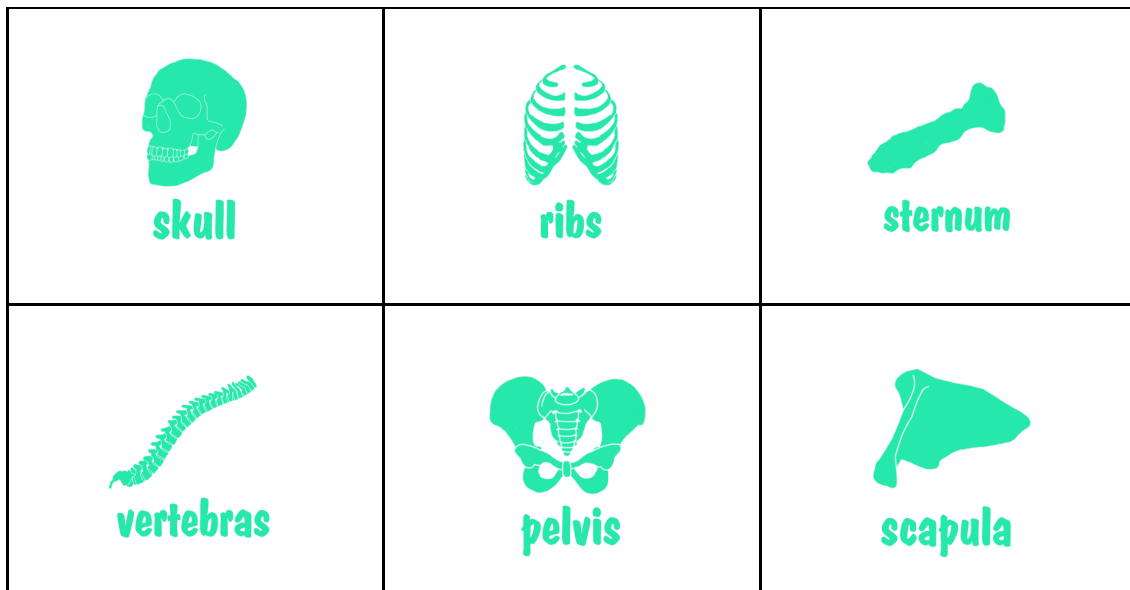


**Figure 29: Handmade texture for a wall**



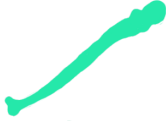

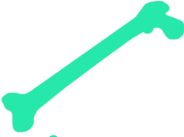


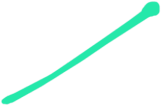




The furniture close to those walls are similar to the wall color palette.

### 3.3.1.3 UI Sprites

Here we can see the art created for the buttons of the game, created with Photoshop:





 humerus	 radius	 ulna
 hand	 femur	 patella
 tibia	 fibula	 foot
 unlock	 menu	 help

**Table 9: UI Button Sprites**

### 3.3.2 3D Art

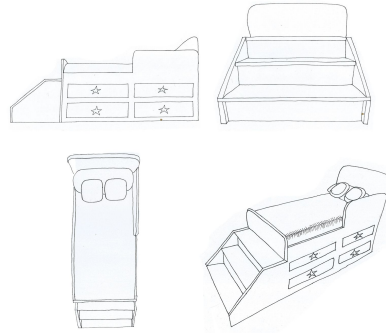
All the 3D models in this project has been made with the open source program Blender. This program is useful to model organic and not organic models.

As explained in the GDD, as this scenery is designated to be for a child, furniture follows simple designs, with patterns and without prominences. Also, I don't want to make it for a girl of a boy. That is why I decided to use different colors depending on the wall. Furniture close to the wall had the same palette color than the walls itself

The process for modelling a 3D element for the scenery was the following:

### 3.3.2.1 Previous design

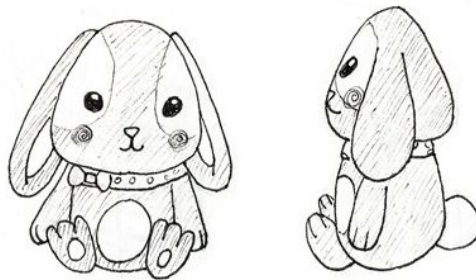
Before 3d could be modeled, a previous design was demanded (See Figure 28).



**Figure 30: Handmade bed design**

In this example you can see the different elements that the bed have. The mattress, the protection wall, the stairs and the drawers.

In case of organic or difficult models, different points of view were mandatory too (See Figure 29):



**Figure 31: Preliminary bunny draw**

In the organic 3D objects, a preliminary study were demanded for each bone. Not only for the shape, but also for knowing how the bone is created and can interact with other bones.

### 3.3.2.2 First steps of modelling

At first, 3D models were created without thinking a lot about the correct topology, and number of triangles. The high number of triangles can be a performance issue that has to be solved in a close future.

The big number of triangles created lag moments in-game: a single heavy model is not a problem, but when a lot of them where in the scenery, the framerate goes down. And, as I have said before, changes in the framerate could make the player dizzy.

### 3.3.2.3 Fixing the topology

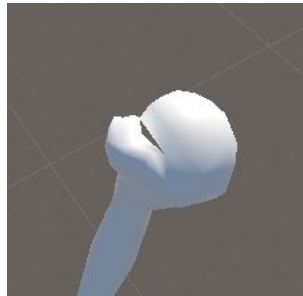
To solve this problems, I investigated in Retopology. This technique consists on recreating an existing surface with more optimal geometry. It is a hard and slow technique but with good results.

If the model were not very complex, the topology could be fixed by hand. This technique was not very effective with organic elements, because it took me many time and the result were not as good as with Retopology techniques.

Related to the bad topology, when the ambient occlusion was activated in Unity, topology shown how the models were really created. Ugly lines crossed supposedly plain surfaces, and made shadows that were not necessary and strange to the eye.

For that reason, some edges and faces had to be removed. With a more logical topology and deleting the unnecessary polygons we achieve a more polish result.

Other problem found when I was modelling was the order of the normals. Usually, the models have all the normals in the same direction, and it is not a problem. But sometimes, due to the order of the polygons is not understood by Blender, the model in Unity showed up the reality: some polygons were flipped (See figure 30).



**Figure 32: Bone model**

This could be easily fixed flipping the normals. The problem was that sometimes, even when I flipped those polygons, the problem was not fixed. This issue made me lose a lot of time, until I discovered a way to fix it, deleting the closer polygons and recreating that part of the mesh.

The last models done had less problems that the first ones, because I took care of the order of the polygons and the exceed of triangles since the first moment.


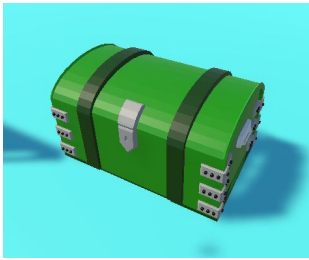
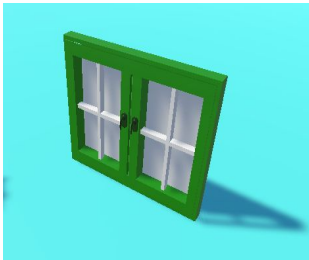
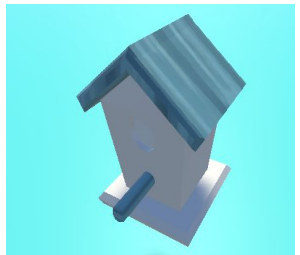
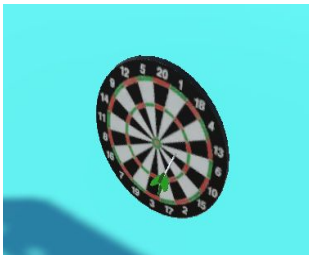
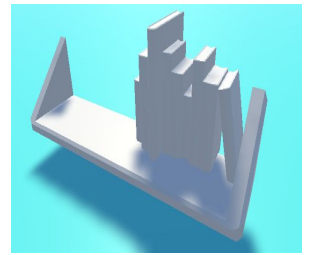
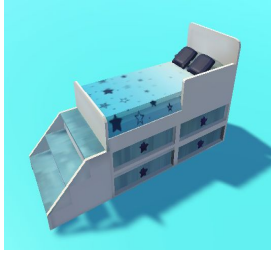


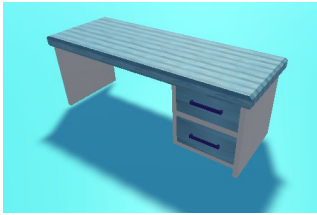


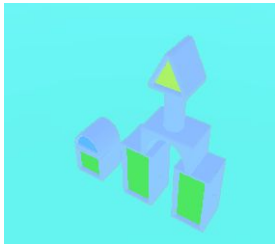


#### **3.3.2.4. Unwrapping the models**

At first, I thought that only some parts of some models were being unwrapped, specifically, that ones with patterns of textures. But, when ambient occlusion was activated in Unity, the reality was shown up: it is necessary to have an UW MAP to create a bake with ambient occlusion.

This issue made me fix again all the models.

### 3.3.2.5 Scenery

The scenery has different elements:

<p style="text-align: center;"><b>Vertical shelf</b></p>	<p style="text-align: center;"><b>Chest</b></p>	<p style="text-align: center;"><b>Window</b></p>
		
<p style="text-align: center;"><b>Bird house</b></p>	<p style="text-align: center;"><b>Diana</b></p>	<p style="text-align: center;"><b>Book shelf</b></p>
		
<p style="text-align: center;"><b>Bed</b></p>	<p style="text-align: center;"><b>Closet</b></p>	<p style="text-align: center;"><b>Rabbit</b></p>
		
<p style="text-align: center;"><b>Desk</b></p>	<p style="text-align: center;"><b>Night table</b></p>	<p style="text-align: center;"><b>Dresser</b></p>
		
<p style="text-align: center;"><b>Cube toys</b></p>	<p style="text-align: center;"><b>Mannequin</b></p>	
		

**Table 10: Scenery**

### 3.3.2.6 Organic models

Two organic models were created for this project.

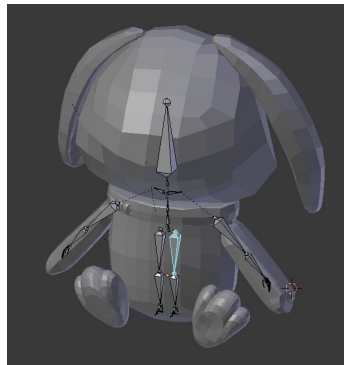
#### 3.3.2.6.1 Rabbit

The model is rigged and have an internal skeleton (See Figure 31).

For achieving this result, it was necessary to create a good mesh where vertex had a similar distance between others.

Also, it was important to have in mind the weights between elements. At the beginning, the ear and the head was not well weighted. This made an ugly effect, each time the ear moved, the head was influenced by it too.

That is why it is important to check the automatic weights that Blender sets to a rigged object.







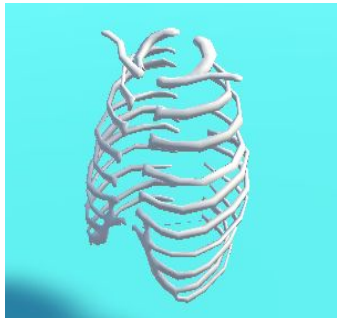

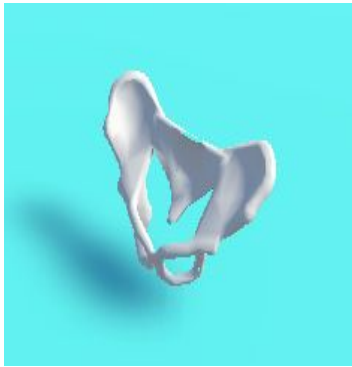


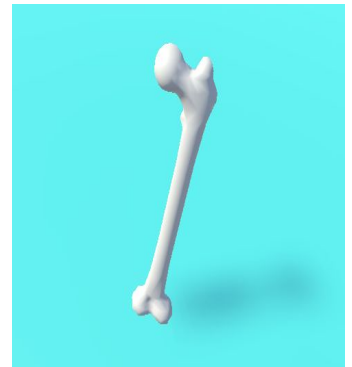





**Figure 33: Rabbit rigged**

#### 3.3.2.6.2 Skeleton

This model is formed by twenty five bones, fifteen of them are different from each other. In addition to those bones, we can find twenty five pieces that are the base bones, before the player places each bone in its place.

The bones are:

Skull	Vertebras	Sternum
		

<b>Humerus</b>	<b>Ribs</b>	<b>Ulna</b>
		
<b>Pelvis</b>	<b>Scapula</b>	<b>Radius</b>
		
<b>Femur</b>	<b>Hand</b>	<b>Tibia</b>
		
<b>Patella</b>	<b>Fibula</b>	<b>Foot</b>
		

**Table 11: Bones**

### 3.4 Testing

While this project was in an alpha phase, many problems were shown up in the first testing sessions. Thanks to the results, some issues could be solved, specially those that could not be easily predicted a intuitive.

Other issues more related with game programming were also shown up during the sessions.

I realized that many of this problems were not found by myself because I always play in the same way, and I did not took in mind that, for example, people not used to virtual reality tends to try everything they have in their minds, and not only those actions said by the game.

Because people who tested my game are not game testers, and some of them, even are not gamers, I recorded their experience, to not lose any detail of their thoughts.

Also, while virtual reality is not comfortable to test (you have to put on and put off the headset everytime you want to report something) I accompanied the players during their test, trying not to answer any of their needs until strictly necessary, and writing down myself the reports with their opinions.

After finishing the testing, I checked the video recorded and I fixed the document I wrote before, if it was necessary.

The testing process has three steps:

#### 3.4.1 Sanity check

In these first steps the tester checked very simple issues related with technical aspects of the videogame. Even if those issues seems to be not important, they are: testing the game in the wrong version or not having the Leap Motion correctly configured could result in a failed test.

Here you can see one of the tests filled:

Tester:	Loli	Date:	12/06/2019	Version	1.0.0
Task	State				
Headset	ok				
Leap motion	ok				
Version	ok	States:	Ok		
Boot	ok		Need fix		
Position	ok		Wrong		
Internet connection	ok				
Freezes	ok				

**Table 12: Test Results 1**

In general, all the Sanity checks were successful. Only the last one had a “Wrong” in the Leap Motion field. It was not well connected and it was necessary to reconnect the hardware.

#### 3.4.2 Functional test

In this second step I asked the player for specific issues of the game that I knew that were problematic, or in the case of the second and third testing session, I asked for issues that other players had found and were supposed to be solved.

This test is more complicated and sometimes the players got stressed if they did not understand exactly what they had to reproduce, or if they were reproducing the bug correctly. That is why, for the third testing, I provided my testers with some footage where the old bug was shown.

Comparing that footage with their own experience in real time made the functional test easier to do.

Here you can see one of the tests filled:

Number	Task	Status			
1	You can select the different game modes	Ok			
2	You can place the bones in their place	Need fix	States:	Ok	
3	You can touch the buttons in the hand	Ok			Need fix
4	When you press the buttons, some bones spawn	Need fix			Wrong
5	The bones spawned are correct according to the button	Ok			

**Table 13: Test Results 2**

In general, most of the issues noted in this test were similar between players. Majority of the issues were fixed when checked (70% Ok, 20% Need fix, 10% Wrong), and, when something was wrong, most of the players found it easily (90%). In those questions that were more personal, like “Is the environment well illuminated” or “Is the game well explained” the results varied more.

### 3.4.3 Free testing

This third step was not exactly sequential. Sometimes the players, from the first minute, started to complain or compliment the game, so every comment they gave as feedback was recorded for this step.

After finishing the first two steps, I gave the players some time for exploring the game. At this point, some interesting feedback was showing up, the players tried to do many things that were not expected: walking to cross the walls, touch the environment, creating bones in crazy places, lay down in the ground...

That is why in the beginning of the game you can see a list of advices for security.

Here you can see one of the tests solved:

Number	Report
1	The books need colors
2	The second shelf is empty
3	Some bones are moving
4	I dont know what to do
5	I see in the bed some strange colors
6	It is hard to place the vertebrae if the ribs are placed first

**Table 14: Test Results 3**

The most interesting points in this testing were:

- **Bugs not found by myself:** bones moving automatically, bones that were not working properly, counters not working properly...
- **User experience issues:** make the buttons disappear when the hand is not in the screen, create a button for rotating the skeleton, moving the spawning buttons...
- **Artistic issues:** change colors of some elements, add or remove elements, switch furniture position...



## Contents

---

4.1 Game Development	53
4.2 Art	53
4.3 Testing	54
4.4 Documentation	54
4.5 Final planification	55

---

In this section the results obtained in the different aspects of this project will be exposed. Also, a general overview of the objectives accomplished will be shown.

## 4.1 Game Development

### 4.1.1 Game modes

All the games work properly. In the objectives of the Technical Report it was promised to create one game mode, and two were implemented, with two variations in each case of difficulty.

The hardest part of these modes were not to overlap one to other. At the beginning all the modes were mixed in one script and the behaviour was not good.

Many problems related with collisions and interaction with Leap Motion and Unity objects were found. All of them were solved.

### 4.1.2 User Interfaces

The implementation of the Leap Motion and the interfaces are working properly. Players can use this interfaces easily and do not need to learn a lot how to play.

Respecting to the main menu, it works perfectly. It calls the functions that are necessary and stores in the GameManager all the information needed.

The hands could spawn bones according to the type. This bones work properly in each game mode.

Also the UI on the bones allows the player to Lock and Expulse bones when needed.

### 4.1.3 Network

The final result of the networking is successful. Players can share the same virtual room and play together.

Due to the complexity of making Leap Motion hands being an multiplayer object, and the limited time for this project, the Leap Motion hands could not being shown through internet. The gameObject is shared, but not the transform and rotations, due to how the software is implemented.

Respecting to the fluency, part of the rotation and translation is processed locally, and the result is more accurate that only showing the result of the other player moving through internet.

## 4.2 Art

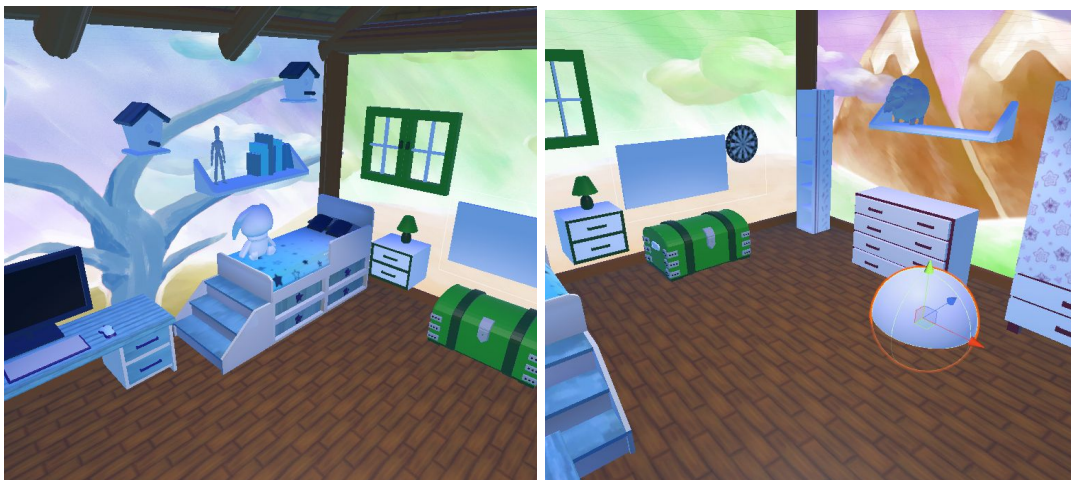
### 4.2.1 2D Art

Hand-painted stylized style was a complete adventure for me. I spent many time learning from tutorials and other artists. But the result is good. The room and the general environment seems to be childish and sweet. All the people who checked my game are agree with that and I am happy for hearing it.

The sprites for the bones buttons are simple, but useful. Testers could recognise easily what they were spawning.

### 4.2.2 3D Art

As I am used to model in low poly, this part of the project was not really difficult. But it is true that took a lot of time to do, because there are many elements in the scenery. Mixing these elements with their textures created a beautiful environment where the player can feel involved (See Figure 32). That is something beautiful and I am glad to have achieved it.



**Figure 34: Room 3D Art**

My skills with Blender related with unwrap and retopology improved a lot. My first models had many problems with normal flipping, bad topology, wrong unwraps, and other issues. The last ones did not gave me many bad surprises when I imported the models ingame.

## 4.3 Testing

The three testing done were very satisfactory and beneficial for this project. Thanks to them, many issues were discovered that would not had been discovered in other way.

Those test can be checked here: <https://bit.ly/2MW82M9>

For example, one of them was related with the hands button position. I realized that when I played, I moved my left hand first and later, with my right hand I pressed the button. In the first testing I did, I realized that people usually move their right hand, and not the left hand. That is why the design was changed.

In general, the free testing was the most useful. Sanity checks and functional test did worked as I was expecting mostly.

Testing this game was more useful than expected. That is why I'm going to continue this practice in my following projects.

## 4.4 Documentation

At the beginning of this project the Technical proposal was sent and most of the objectives have been completed.

At the end of February, Game design document was sent. Some elements have changed, mainly related with Leap Motion hands. The lack of knowledge about this tool made necessary to change some organization of the buttons.

This Final Report was sent at the beginning of July with the video, and in the following days the presentation was finished.

In order to summarize the work done for the project, this table shows a final recount of different elements done for the project.

Item	Value
Scripts	12
3D Models	30
2D Images	24
Game levels	1

**Table 15: Element recount**

## 4.5 Final planification

Code	Task	Dates	Time planned	Time spent
<b>Game development</b>				
GD01	General programming	1st February to 1st July	15	5
GD02	Hardware implementation: HTC VIVE	1st February to 3th February	20	2
GD03	Hardware implementation: Leap motion	1st February to 15th May	15	8
GD04	Elements placement	1st May to 15th May	20	25
GD05	Networking	1st March to 31th March	25	60
GD06	UI: Hands	1st April to 15th April	19	30
GD07	UI: Bones	16th April to 31th April	17	36
GD08	UI: Menu	1st May to 15th May	12	15
<b>Art</b>				
A01	Scenery	1st February to 28th of February	42	66
A02	Organic models	1st March to 31th March	30	40
A03	2D art: Textures	1st April to 15th April	8	15
A04	2D art: UI	16th April to 31th April	5	7
<b>Testing</b>				
T01	Performing test	1st Juny to 8th July	10	5
T02	Writting documentation	2nd Juny to 8th July	5	3
<b>Documentation</b>				
D01	Technical proposal	1st February to 4th February	3	10
D02	Game Design Document	5th February to 25th February	10	20
D03	Report	1st Juny to 1st July	30	50
D04	Tutoring	1st November to 12th July	6	6
D05	Presentation	1st July to 8th July	3	10
D06	Video	18th Juny to 1st July	5	10
<b>TOTAL</b>			300	423

**Table 16: Final planification result**

As you can see, in the most part of the project the time needed was expanded (See Table 15). This was related to the learning curve needed for the SDKs employed.

The installation of Leap Motion and HTC Vive took me less time than expected. Leap Motion took me more time because they update the device frequently and the hands stopped working often because of this.

Also, the art part has plenty of elements, so it took many time to develop it.

The testing part was not as difficult as I thought, and gave many good feedback that improved my game.

Writing all the documentation took way more time than expected due to writing in a foreign language because it is not something I am used to.

---

The next objectives for this project are:

- **Add the network connection to the hands:** I want this game to be completely multiplayer. This issue is going to take me some time and investigation. A user in Reddit gave me a possible solution for this problem, but it is very hard to implement and did not work the first time I tried. I want to try again that solution and improve my project.
- **Model my own 3D hands:** Now I am using 3D hands that are from Leap Motion itself. I would like to model a good one, that could fit better with this project. To achieve this, I will use Blender again as my main tool and I will investigate the motricity of the human hands.
- **Implement other minigames with organs and muscles, or other animals:** the options are unlimited and very attractive. I would like to implement more game modes, so I can expand this game. The minigames that I would start to implement are the skeleton of a bird and the organs of a person. That will require many art and more implementation, but will be possible to do with time, because all the basis is implemented in this project.
- **Create other two sceneries:** Each scenery could be used for a different minigame. One of them I would like to be a farm, because it is a good environment for an animal minigame. Other one could be an orchestra environment, where some musical instruments need to be built.
- **Adapt this project to HTC Vive controllers:** very few VR players have Leap Motion at home. That is why adapting this project to the usual controllers will be a good solution.

This page is left blank intentionally.

---

I have learnt that creating a video game from beginning to end with all the different areas could be very difficult. I have spent a lot of time learning how to deal with different topics of the game: 2D art, 3D art, and programming different objects and technologies. That took a lot of time and effort to do, but the results are good.

About the art part, I realized that the key to create a beautiful scenery lies in not only the 3D models, and how are they designed, but also in illumination, textures, shadows and synergy between the different elements could create a real ambient where a player can feel integrated.

Working with unknown hardware is not as easy as it seems. Sometimes documentation is too hard and limited for unfamiliar developers. Another times, the best documentation is not in the official website. Also, trying to understand code that connect software and hardware to create something new can be really hard. It could be even worse if almost no one in your environment is familiar with the technology. That is why I crossed many hard moments in this project, and I am glad to see that in the main result I have a game where I can play from beginning to end using a tool that nobody taught me.

To summarize, I have a game where a player can start playing without help, with an scenery that helps him to be involved, a technology well implemented that surprises them, and minigames that challenges them to learn and have fun at the same time.

In general terms, I am satisfied with this project because I have solved the problems that I have found and I feel that this project could be a good one with some time. Serious games and Virtual Reality are growing each day and this work is a good beginning for them.

This page is left blank intentionally.



- 
1. **Unity Technologies.** *Unity User Manual*. From <https://docs.unity3d.com/es/current/Manual/UnityManual.html>
  2. **Blender.** *3D Creation suite*. From <https://www.blender.org/>
  3. **Exit Games.** *Photon SDK API Photon Server, Photon Unity Networking*. From <https://doc-api.photonengine.com/>
  4. **Ultrahaptics Ltd.** **Leap Motion technologies** From <https://www.leapmotion.com/>
  5. **HTC.** **HTC VIVE Hardware**. from <https://www.vive.com/us/>
  6. Víctor J. Osma-Ruiz, Nicolás Sáenz-Lechon, Juana M. Gutiérrez-Arriola, Irina Argüelles-Álvarez, Rubén Fraile, Roberto Marcano-Ganzo, . (2015). **LEARNING ENGLISH IS FUN! INCREASING MOTIVATION THROUGH VIDEO GAMES. SPAIN:** Departamento de Teoría de la Señal y Comunicaciones, E.T.S. Ingeniería y Sistema de Telecomunicación, Universidad Politécnica de Madrid from [http://oa.upm.es/44606/1/INVE\\_MEM\\_2015\\_241732.pdf](http://oa.upm.es/44606/1/INVE_MEM_2015_241732.pdf).
  7. **Krita Foundation.** **Krita** | Digital Painting. Creative Freedom. From <https://krita.org/es/>
  8. Fritz Schider. **An Atlas of Anatomy for Artists**. From [https://www.goodreads.com/book/show/327249.An\\_Atlas\\_of\\_Anatomy\\_for\\_Artists](https://www.goodreads.com/book/show/327249.An_Atlas_of_Anatomy_for_Artists)
  9. **Gameplay video, exe and pictures:** <https://bit.ly/2LumrwB>. To play, launch "Videojuego Anatomía" .exe in the .rar. It is necessary to have Leap motion and HTC Vive to play.
  10. **Repository:** <https://github.com/ikoknight/AnatomyGame>

This page is left blank intentionally.