2022

# Pawblade GDD

PROFESSIONAL GAME DEVELOPMENT ENVIRONMENTS (CI5515)

BELLIDO CHUECO, DANIEL – K1925456

# Game Design Document

## Contents

## DESCRIPTION OF THE GAME

This video game developed in Unity is a game that belongs to the classic platform genre where the gameplay is based on running, jumping, hitting, dodging hazards, and collecting different objects.

In terms of narrative, the main character of this story is a knight who has been the victim of a spell that has turned him into a dog and now, equipped with a sword and a shield, has as an objective to break the spell that has turned all the inhabitants of the kingdom into different animals and other wild creatures.To do this, the player must advance level after level until reaching the final boss and defeat it to undo the spell.

In each level, the player will have to make his way by eliminating some enemies, dodging some dangers and above all jumping from platform to platform to advance through the level.In addition, throughout the level, the player will find different collectibles such as gold coins and potions that will allow the player to have an extra attempt in case he has fallen into a trap or has been hit by an enemy. The player will also have to be careful since he does not have a health bar, so at the slightest impact, the player will lose a life and return to the last checkpoint.

Some collectables such as potions provide an extra life and other collectables such as coins are automatically redeemed for another extra life when a total of 100 coins have been collected.Other collectibles that could not be implemented are the level-hidden keys that give access to the level of the boss in each area. The intention of these keys was to make the game more re-playable by having hidden keys only in some levels where the player must find them by himself.

The target platform of this project is designed to work mainly on systems that have a controller since playing with this type of peripheral is more comfortable for the public and also preserves the essence of the gameplay of the genre itself. Some examples of these systems are Nintendo Switch, PlayStation 4, Xbox and the PC.


*Figure 1 Screenshot of the game*

## VISUAL DESIGN, LEVEL DESIGN AND GAME MECHANICS

The visual design intends to be one of the main appeals for the users as it presents a cartoony family-friendly style full of attractive colours. The aesthetics of the selected assets have a friendly appearance, and an attempt has been made to cohere their appearance as much as possible to communicate to the users that they are in a fun world full of comic resources such as animations, ambient sound, and sound effects typical of humorous cartoons.Even though some elements such as the user interface and the choice of fonts do not adjust to the level proposed by the rest of the elements such as the Main Menu, the choice of colours, the animations, and the character design, the game presents a visual aspect very attractive that is interesting to the platform game lovers.

As for level design, the master class of level 1-1 of Mario Bros for the NES has been taken as a reference. In the first level of this project, the player is in a safe and totally isolated area, which forces the user to start moving to find out what the game is about. The level design quickly gives feedback on which direction to go, as the level is a wide corridor that they must go through, making the game a linear game. However, in front of the player, they find a high wall that cannot be overcome, which forces the player to go to the left, there is where there are three platforms of different elevation levels that resemble a staircase. It is here where the player

discovers the most basic gameplay mechanic: jumping. And on the last step, the user finds themselves with the first object with which the player is going to collide if they want to continue advancing: a coin (Figure 2). The coin quickly gives positive feedback through sound and by raising the UI coin counter from zero to one, meaning coins are objects the player can collect.

*Figure 2: Collectable Coin*

*Figure 3: Set of Boxes*

Once the player reaches the top of the high wall more coins are found, and they are arranged in such a way that they guide the player towards the introduction of a new element: the box (Figure 3). The player can collide with the box, but this time nothing will happen, not even when the player jumps on it. This is when the level design introduces the next mechanic: hitting with the sword (Figure 4).

*Figure 4: Player breaking a box*

*Figure 5: Collectable Life*

Once the box is hit and broken, another object immediately appears inside the box with behaviour similar to the coin, but in this case, upon colliding with it, the feedback through the UI will indicate that the player has gained an extra life (Figure 5).Having learned the two basic mechanics of jumping and attacking with the sword, the player is ready to face the first enemy: a mutated mushroom that approaches the player (Figure 6). The player can avoid being hit by the enemy or they can attack the enemy with the sword, if the player is hit by the enemy, the player will lose a life and return to the starting position having to repeat the previous path. This was the purpose of the life given before, to allow the player to discover that not all objects on

the screen are allies. The enemy, or independent agent, is equipped with an Artificial Intelligence based on four state machines: Idle, Patrol, Chasing, and Attack (Figure 7). The agent follows a predetermined path using waypoints.

Figure 6: Basic Enemy

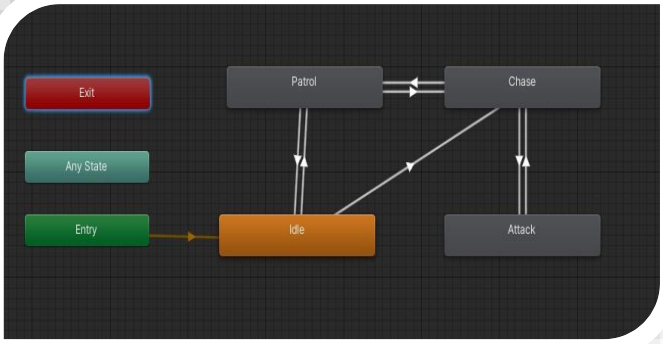


Figure 7: Enemy Artificial Intelligence

After getting past the first enemy, the player comes across more fixed platforms where they must jump to cross the small lake. If the player falls into the water, they will discover a new element that works against them: environmental hazards(Figure 8).

Figure 8: Environmental Hazard (Water)

From this point on, it can be said that the player has passed the invisible tutorial that the level design has proposed and now the difficulty curve will progressively increase as the player progresses through the different areas of the level and the rest of the levels until completing the game.One of the most important elements that will add difficulty to the game will be the mobile platforms that will start to appear from this point. The platforms will have different behaviours as some will move vertically, others horizontally and others will rotate around a common axis (Figure 9).

Figure 9: Mobile Platform

As the difficulty increases, new environmental hazards will appear, such as spikes (Figure 10), but one of the most interesting game mechanics that will help the player to progress during the game is the implementation of checkpoints.

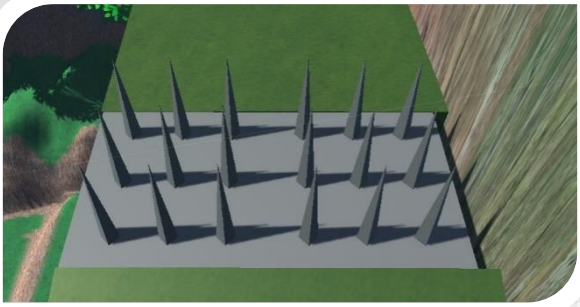

*Figure 10: Environmental Hazard (Spikes)*

Checkpoints are temporary save points that the player will have to activate as they progress through the level. These checkpoints are red when they are deactivated and to activate them the player only has to collide with them, in this way the checkpoint will turn green indicating to the player that in case of death the reset point will be the last checkpoint activated (Figures 11 and 12).

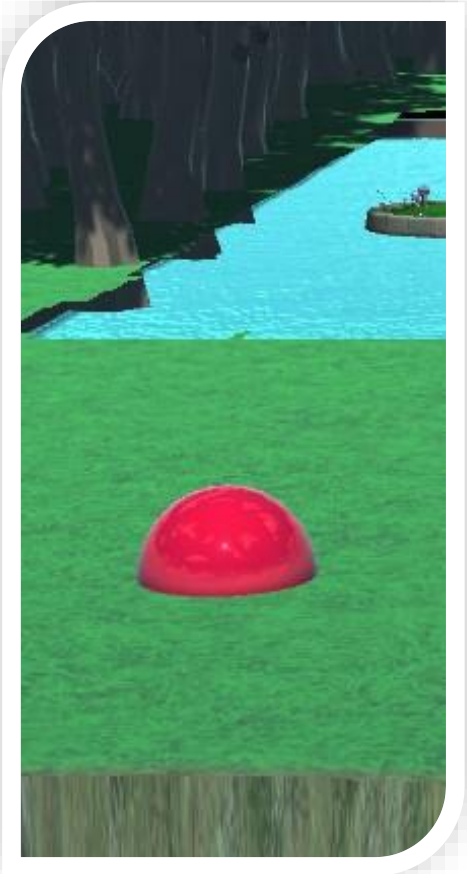

*Figure 11: Checkpoint deactivated*

*Figure 12: Checkpoint Activated*

To finish with the level design, at the end of each level there is a platform totally different from the rest that visually communicates to the player that it is a special platform. This platform is the goal or end of the level that takes the player to the next level with more enemies and more hazards (Figure 13).

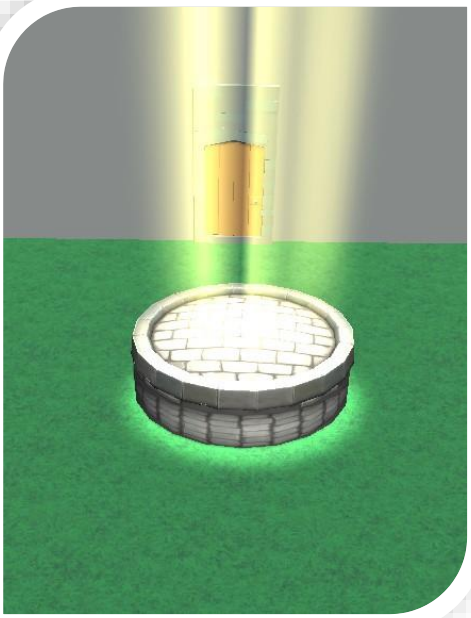

*Figure 13: Goal at the end of the level*

*All the references for the assets at page 17**

## TECHNICAL DESIGN

The architecture of the game is supported almost entirely by the Monobehaviour class (Figure 14)and the technical design is simple since the vast majority of classes are derived from Monobehaviour with a few exceptions, such as the Artificial Intelligence of the enemies that derive from the StateMachineBehaviour class.

Most of the classes created perform very specific tasks, such as the LifeCollision class that performs a simple trigger operation. Others like ObjectRotation simply do what the name suggests, and this is a very useful script as it has been used on different types of objects such as coins, lives, or platforms. Another example of reusable code is HazardDamage that has been applied to objects such as spikes or the object that controls the behaviour of water. This class has the basic function, in addition to managing certain sound effects, of triggering the death of the player when there is a collision between them.More examples of reusable code can be found, such as the DestroyingBox class. While it is true that the name suggests that it only destroys boxes, that was its initial function, the code has been reused so that the player can attack both boxes and enemies in the same way.

Regarding the technical design of the player, this object is perhaps one of the most complex at the code level since the PlayerController class is a script that handles more data and interacts with many more elements, such as the camera, sound effects, LayerMask, , enemies, invisible objects like groundCheck, to check if the player is grounded or not, or even other scripts like the sword script, SwordAttack, whose object is a child of the player object and they interact as a whole.

The GameManager is also an object that interacts with many elements and oversees the data displayed on the screen through the UI, the scene management, some sound effects, and some game rules such as checkpoint, the Game Over or the Restart function among others.

But as mentioned above, not all classes derive from the Monobehaviour class. This is the case of the behaviour of the enemies who use a set of scripts that derive from the StateMachineBehaviourclass; therefore, these scripts are not assigned to the enemy object but to the Animator component. This set of states is divided into four scripts that give the enemy independent behaviour and react based on the player's position.

The idleBehaviour is responsible for keeping the static agent in a position for a short period of time, once the time of this state is over it passes to the PatrolBehaviour state which interacts with a list of other objects, waypoints, that determine the agent's path. This loop repeats over and over until the player gets close enough to trigger the ChaseBehaviour state. This script interacts with the player's Transform and makes the enemy go to the player's position in order to attack. When the enemy enters the attack range, the following script called AttackBehaviour is activated, which will make the enemy execute the attack animation, which will call a function of the KillingPlayer class, which derives from the Monobehaviour class.

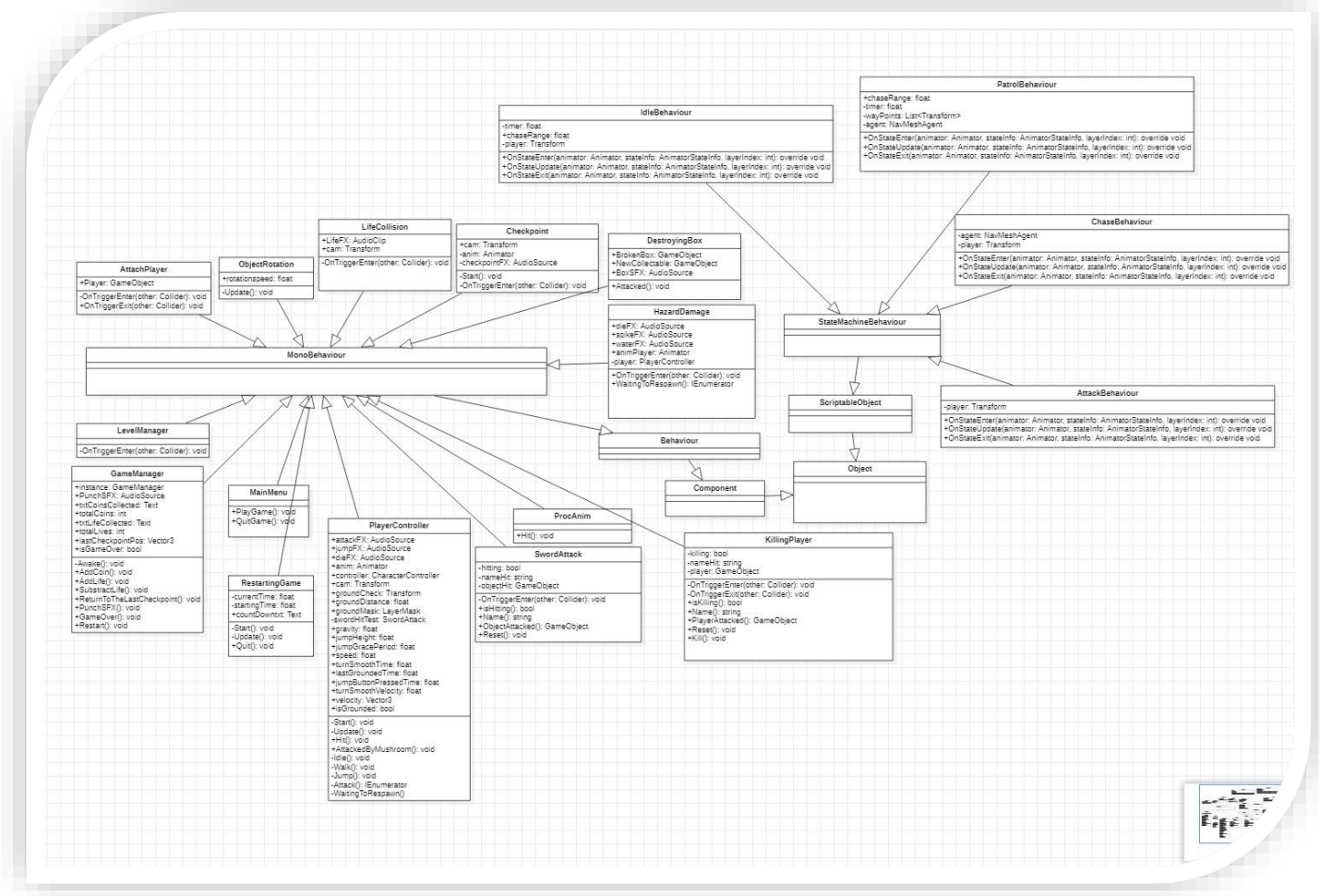


*Figure 14: UML Class Diagram*

## IMPLEMENTATION

The game has been implemented in C# and is mostly based on the Monobehaviour class. Some interesting aspects of the code are the Hit() and Kill() functions, which communicate with other scripts and make objects behave in the desired way. As can be seen in figure 15, up to a total of 3 scripts cooperate to execute a given action in a given animation frame.

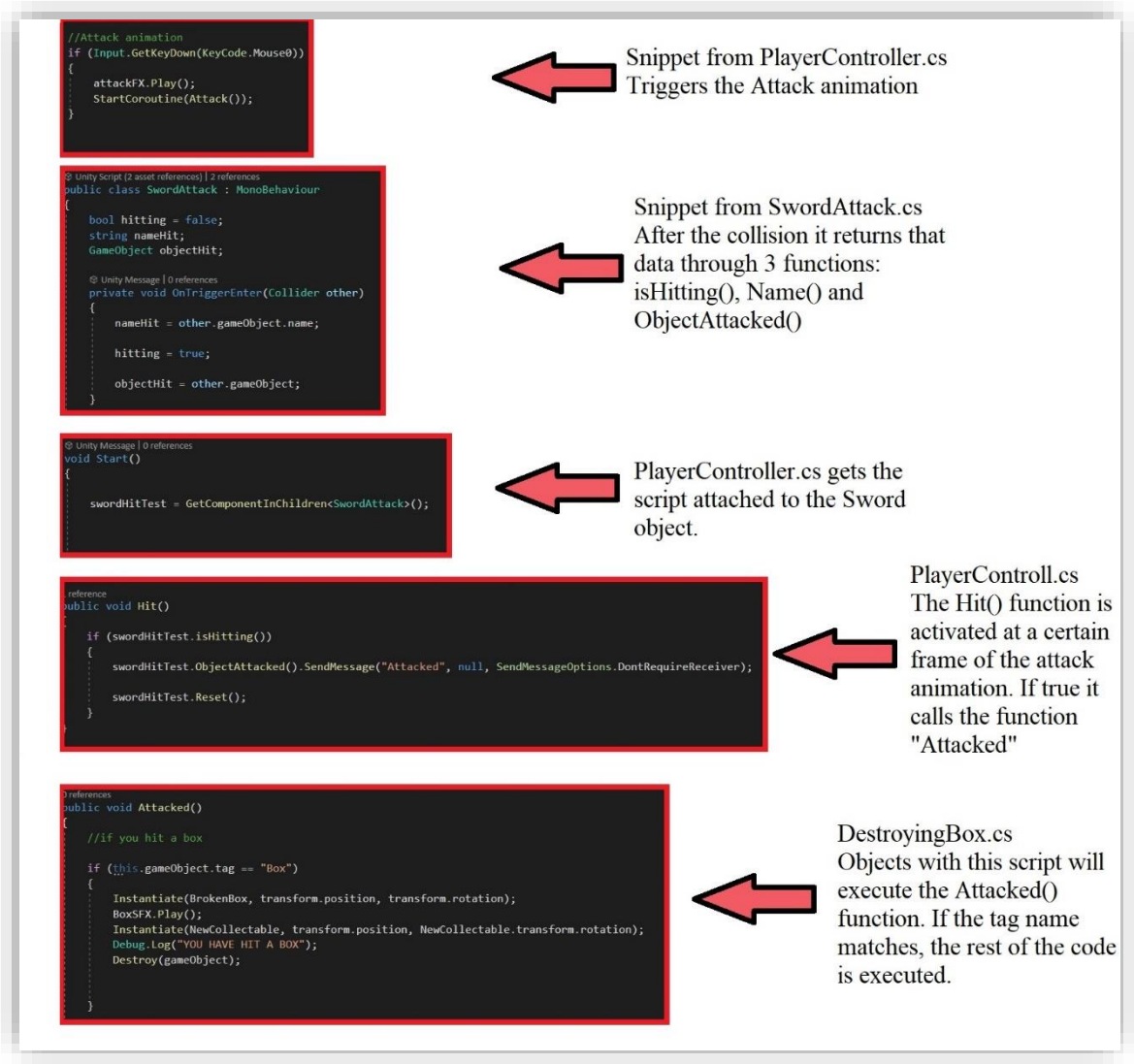


*Figure 15: Example of three different scripts performing one simple action through three different objects: Player, Sword and Box.*

The PlayerController runs a coroutine that triggers the attack animation,when the right mouse click is pressed, which will execute the Hit() function at a given frame of the animation. For the code in this function to work, it needs data that the Sword object is going to return when there is a collision between the sword and the other object. These data return true if there has been a collision and the name of the object with which it has collided.

The data is passed from the SwordAttack script to the PlayerController script which is executing the Hit () function. Inside the Hit() function there is an if statement that if it returns true, it will send a message to the hit object to trigger the Attacked function found in the hit object's script. This last script, being reused by different types of objects, will require a tag to be able to execute the correct code block.

The following example in Figure 16 is very similar but, in this case, only two scripts are communicated, the PlayerController, hosted on the Player object, and the KillingPlayer, attached to the Enemy Object.


*Figure 16: Example how the Enemy kills the Player.*

## TESTING AND EVALUATION

The plan to test the game has been basic since the prototype has been developed in a total of twenty days and there has not been much time to carry out more exhaustive tests due to multiple technical setbacks.

Still, the game has been tested daily by me as new features have been implemented, so the first approach to the testing plan has been to test if things worked and look for what could go wrong. For instance, the level has been created using the terrain tool, and for the player to know if the Boolean isGrounded to perform the jump was true or false, a LayerMask called "ground" was created that would not allow the player to jump if the ground layer, together with other variables, it was false. The implementation worked but later a problem was found with this implementation: The player could perform jumps if they were colliding with the walls of the terrain. This ruined the mechanics of the game since if the player didn't need platforms to overcome obstacles like walls, the game would miss the essence of the genre. The solution to this problem was to strategically place walls created with ProBuilder and not render the MeshRenderer, so the player would be colliding with something that didn't have the terrain layer mask.

Even though we didn't have much time to thoroughly test the game, the possibility that two people could play the game and the feedback was quite positive since both of them really liked the graphical potential of the game. They were able to find various details that could be improved such as jump accuracy, the inclusion of more enemy types, and other minor details.

The inclusion of more enemies could not be possible because there was not enough time for it and other problems needed to be solved. However, as for jumping improvement, I did some research on how I could improve the jumping since jumping is an essential game mechanic, and I found a basic principle about platformers that helped improve the gameplay: the coyote jump

This technique consists of giving the player a short space of time to be able to jump shortly after they stop touching the platform. This principle has made the game much more playable, and the result is quite satisfying despite a lot of work that must be done.

## LEGAL, SOCIAL AND ETHICAL ISSUES

The PEGI rating for this product would be 7 because according to NICAM, the organization responsible for rating PEGI 3 and PEGI 7 games, the product contains some acceptable violence in a humorous context and the characters in the game are fantasy, but the game also contains sounds, such as spikes, that could frighten young children under 7 years of age.Therefore, the game would be classified as PEGI 7 like other similar games of the same genre such as Crash Bandicoot, Mario Odyssey, Spyro, or Yooka-Laylee among others.

Regarding legal use of assets, according to Unity Support, once you have purchased and downloaded an asset you can use it commercially with no additional cost. This also includes assets that are offered for free if they are not restricted.Unity restricted assets cannot be used commercially, and these restrictions will be made explicit in the description of the asset page in question or in a .txt format file within the downloaded asset.

Unity also allows to sell commercial games using the free version of Unity if the requirements to use the Unity Personal Edition are met. These requirements are not to earn more than $100,000 a year with your products created in Unity either in the form of online purchases or income from advertising.

Finally, make it clear that this game will not be marketed,as the work is not polished enough, and some audio assets are copyrighted. In addition, the game does not use the personal data of the players in any way since it neither requires them nor stores them in any database.

## CONCLUSIONS

This project has been developed in a total period of 20 days and the management of the project has been decent considering that it is the first project carried out individually using a completely unknown graphic engine and framework.

Unfortunately, it was not possible to present an initial project proposal in October because I personally did not know what the possibilities were that I had using Unity and I needed to know and feel comfortable in this new environment to have a clear idea of what I could do.This is also the reason why a Gantt chart could not be implemented from the beginning since there was no type of idea that I really wanted to work on and because other academic and personal tasks did not allow me to think calmly about this project.In the end, I decided to develop a type of game whose genre I am passionate about: platformers.

The platform genre has always caught my attention because I find it to be a very fun genre and one that has driven the video game industry to enjoy the popularity it has today. So as a video game developer it seems to me an obligatory task to learn and develop a game with these characteristics.

The development of the project began after the Christmas break with a first commit on Gitlab, and the idea began to be built around the main character, a level, and some obstacles. Designing the level took a few days as it was experimenting with the use of the terrain tool. Regarding the main character, it took about a week to develop it until it reached its current state since many hours were invested in researching different techniques and approaches to develop the character with their respective animations, with a camera following it and the implementation of the animation events, which took me several days to understand why it was not working.I have also spent a lot of time researching how some components work, such as the Rigid Body, the absence of which meant that some collisions were not detected,and this led to problems with mobile platforms.

The implementation of enemies and their artificial intelligence was also a challenge in which I had to spend a few days, but it was an enriching experience since it helped me

to better understand state machines and the different ways of using them, not only in Unity but also in other type of frameworks.Other aspects that have helped to understand the operation and development of video games with Unity are the use of more abstract objects such as the Game Manager, the Level Manager, or the Main Menu.

In conclusion, for future projects, this learning process has made me discover the true potential of the Game Engine and above all, it has helped me understand the importance of planning, especially at the technical design level, of a project to avoid errors and other setbacks that reduce productivity and efficiency.

# REFERENCES

## Articles

1. Extra Credits (2014) *Design club - Super Mario Bros: Level 1-1 - how super Mario mastered level design*. Youtube. Available at: https://www.youtube.com/watch?v=ZH2wGpEZVgE (Accessed: January 26, 2022).
2. *PEGI explained - kijkwijzer.nl* (no date) *Kijkwijzer.nl*. Available at: https://www.kijkwijzer.nl/about-pegi (Accessed: January 26, 2022).
3. *Unity.com* (2019)Can I use assets from the Asset Store in my commercial game? Available at: https://support.unity.com/hc/en-us/articles/205623589-Can-I-use-assets-from-the-Asset-Store-in-my-commercial-game-?mobile_site=true (Accessed: January 26, 2022).
4. *Unity.com* (2020) Can I make a commercial game with Unity Free/Personal Edition?Available at: https://support.unity.com/hc/en-us/articles/205253119-Can-I-make-a-commercial-game-with-Unity-Free-Personal-Edition-?_ga=2.23825082.1689951378.1643175976-1963045512.1632855563&_gac=1.52777050.1641242141.Cj0KCQiA2sqOBhCGARIsAPuPK0hM_4uXx83D9Q7jfYJ9ZQMxecvxwTDy1-2KU-milHxWiUNENQ99UOoaAp2hEALw_wcB (Accessed: January 26, 2022).

All the images of this report are own source

# ASSET LIST

## 3D Models

Dog Knight PBR Polyart

https://assetstore.unity.com/packages/3d/characters/animals/dog-knight-pbr-polyart-135227

Potions, Coin And Box of Pandora Pack

https://assetstore.unity.com/packages/3d/props/potions-coin-and-box-of-pandora-pack-71778

Fantasy Mushroom Mon

https://assetstore.unity.com/packages/3d/characters/creatures/fantasy-mushroom-mon-115406

Handpainted Turntable Platforms

https://assetstore.unity.com/packages/3d/props/handpainted-turntable-platforms-66599

Lowpoly Environment - Nature Pack Free

https://assetstore.unity.com/packages/3d/environments/lowpoly-environment-nature-pack-free-187052

CastlePack_3D environment

https://assetstore.unity.com/packages/3d/environments/castlepack-3d-enviroment-123650

## Audio


1. VGH Synthetic Orchestra (2015) *Crash Bandicoot: Warped - Medieval Theme (VGH synthetic orchestra)*. Youtube. Available at: https://www.youtube.com/watch?v=JTCJMim7Gts (Accessed: January 26, 2022).

2. Pacifica Productions (2013) *Croc - Legend Of the Gobbos - 02 - volcano island map*. Youtube. Available at: https://www.youtube.com/watch?v=KKHQWzSSSHA (Accessed: January 26, 2022).

3. GamingSoundEffects (2015) *Sword slash sound effect*. Youtube. Available at: https://www.youtube.com/watch?v=Oa1mfoOInbA (Accessed: January 26, 2022).

4. Pandesal, T. V. (2020) *Best cartoon jump sound effects | used by youtubers / vloggers*. Youtube. Available at: https://www.youtube.com/watch?v=scOXprR0NCE (Accessed: January 26, 2022).

5. 2MirrorsDialogue (2017) *Wooden box breaking sound effects*. Youtube. Available at: https://www.youtube.com/watch?v=oD9ZUXDEoyA (Accessed: January 26, 2022).

6. Sound Library (2020) *All spike trap sounds (fortnite) - sound effects for editing*. Youtube. Available at: https://www.youtube.com/watch?v=NnSuTjsNstY (Accessed: January 26, 2022).

7. Think Media (2020) *FREE Sound Effects Pack YouTubers use! (royalty free)*. Youtube. Available at: https://www.youtube.com/watch?v=m2QAH1SpEQ8 (Accessed: January 26, 2022).

8. Adventures of Katie (2019) *These are the sound effects for the Scott Fee Dizzy game*. Youtube. Available at: https://www.youtube.com/watch?v=9_V1wYUGnFs (Accessed: January 26, 2022).

9. SoundMeFreely (2019) *Super Mario 64 complete Sound Effects*. Youtube. Available at: https://www.youtube.com/watch?v=vHtQmFD5mxg (Accessed: January 26, 2022).

10. TheGoldenJiggy (2015) *Collect Musical Note - Banjo-Kazooie Music*. Youtube. Available at: https://www.youtube.com/watch?v=CZ3O60DeqFQ (Accessed: January 26, 2022).

11. Gfx, S. A. (2017) *Water splashing sound effect - free download HD*. Youtube. Available at: https://www.youtube.com/watch?v=h1-Wu0VlWbQ (Accessed: January 26, 2022).

## Code

1. Brackeys (2017) *SHATTER / DESTRUCTION in unity (tutorial)*. Youtube. Available at: https://www.youtube.com/watch?v=EgNV0PWVaS8 (Accessed: January 26, 2022).
2. Brackeys (2017b) *START MENU in Unity*. Youtube. Available at: https://www.youtube.com/watch?v=zc8ac_qUXQY (Accessed: January 26, 2022).
3. Brackeys (2020) *THIRD PERSON MOVEMENT in unity*. Youtube. Available at: https://www.youtube.com/watch?v=4HpC--2iowE (Accessed: January 26, 2022).
4. GDTitans (2021a) *Enemy AI in unity - (E01): STATE MACHINE BEHAVIORS*. Youtube. Available at: https://www.youtube.com/watch?v=CmzJtM5BIKE (Accessed: January 26, 2022).
5. GDTitans (2021b) *Enemy AI in unity - (E02): CHASE AND ATTACK*. Youtube. Available at: https://www.youtube.com/watch?v=PK3gDP70pTo (Accessed: January 26, 2022
6. GDTitans (2021) *Simple Checkpoint System in Unity*. Youtube. Available at: https://www.youtube.com/watch?v=mBn7ZIB5Zhw (Accessed: January 26, 2022).
7. Games, K. (2021) *Improve annoying jump controls with coyote time and jump buffering (Unity tutorial)*. Youtube. Available at:

https://www.youtube.com/watch?v=oQ9877FdR8o (Accessed: January 26, 2022).

8. Jayanam (2018) *Unity moving platform tutorial*. Youtube. Available at: https://www.youtube.com/watch?v=rO19dA2jksk (Accessed: January 27, 2022).