Timmy Adventure

Game Design Document

Professor

Carmelo Macrì

Students

Thomas Voce, 214719

Gianfranco Sapia, 223954

Andrea De Seta, 227755

Course of Virtual Reality

UNIVERSITÀ DELLA CALABRIA DIPARTIMENTO DI MATEMATICA E INFORMATICA

Made with unity

Academic Year 2020-2021

Sommario

Sommario ... 2
1. Overview ... 5
  1.1. General information ... 5
  1.2. Introduction ... 5
  1.3. External influences ... 5
2. History ... 7
  2.1. Story ... 7
  2.2. Characters ... 7
  2.3. Enemies ... 8
3. Gameplay ... 9
  3.1. Introduction ... 9
  3.2. Mechanics ... 9
  3.3. Victory and Game Over conditions ... 10
    3.3.1. Victory Conditions ... 10
    3.3.2. Game Over Conditions ... 10
  3.4. Save system ... 11
  3.5. Dynamics ... 11
    3.5.1. Player ... 11
    3.5.2. Enemies ... 11
  3.6. Commands ... 12
4. Levels ... 13
  4.1. HUB ... 13
  4.2. Level 1 ... 14
  4.3. Level 2 ... 15
  4.4. Level 3 ... 16
5. UI Design ... 17
  5.1. Main Menu ... 17
  5.2. Main Menu Options ... 18
  5.3. HUD ... 19
  5.4. Pause screen ... 19
  5.5. Pause Screen Option ... 20
  5.6. Dialogues ... 21

5.7. Victory Screen 21
5.8. Game Over Screen 22
5.9. UI flowchart 23
6. Art Assets 24
6.1. 3D models 24
6.1.1. Player 24
6.1.2. Enemies 24
6.1.3. Collectables 27
6.1.4. Interactive object 28
6.1.5. Environmental Object 30
6.1.6. Obstacles 31
6.1.7. 3D models for creating levels 32
6.2. 2D sprites 33
6.2.1. Logo 33
6.2.2. HUD 33
6.2.3. Menu UI 34
6.2.4. Skybox 35
6.3. Animations 35
6.3.1. Player 35
6.3.2. Enemies 36
6.3.3. Other animations 37
6.4. Visuals 37
6.5. Audio 41
6.5.1. Music 41
6.5.2. SFX 41
7. Implementation 42
7.1. External Packages 42
7.1.1. MAST 42
7.1.2. Cinemachine 42
7.1.3. Path Creator 42
7.2. Managers 43
7.2.1. Managers Class Diagram 43
7.2.2. Managers description 43
7.3. Player 45
7.3.1. Player Class Diagram 45

7.3.2. Player description 46
7.4. Enemies 49
7.4.1. Enemies Class Diagram 49
7.4.2. Enemies Description 50
7.5. Obstacles 51
7.5.1. Cannon 51
7.5.2. Log waves 52
7.5.3. Rotating logs 52
7.5.4. Falling Blocks 52
7.5.5. Lava pools 53
7.5.6. Boulder 53
7.6. Camera 53
7.7. UI 55
7.7.1. UI Diagram 55
7.7.2. Brief description of the classes 56
7.8. Others 58
7.8.1. Interactive objects 58
7.8.2. Environmental object 60
7.8.3. Collectables 60
7.8.4. Scene Loader 61
7.8.5. Save system 61
7.9. Performance 62
7.9.1. Lighting 62
7.9.2. Fog 62
7.9.3. Occlusion Culling 63
8. Future development 64

1. Overview

1.1. General information

- **Name**: Timmy Adventure;
- **Number of players**: Singleplayer;
- **Genre**: Platform, Adventure;
- **Prospective Type**: Third person;
- **Perspective type**: Perspective;
- **Platforms**: PC;
- **Game Tools**: Mouse + KeyBoard, Joypad;
- **Development tools used**:
    - **Game Engine**: Unity 3D;
    - **Scripting Software**: Microsoft Visual Studio;
    - **3D Modelling**: Blender;
    - **Graphics**: Adobe Photoshop;
    - **Animations**: Mixamo;
    - **Audio**: Audacity;
    - **Version Control System**: Github;

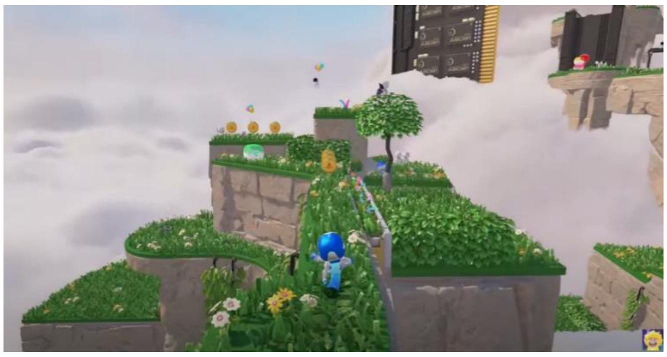
1.2. Introduction

Timmy Adventure is a 3D videogame belonging to platform and adventure genre, developed by ***Gemini*** group for *Virtual Reality* exam of Professor Carmelo Macrì at University of Calabria. The videogame is presented as a demo and it wants to recreate "childhood memories" of those videogames back at time of retro-gaming.

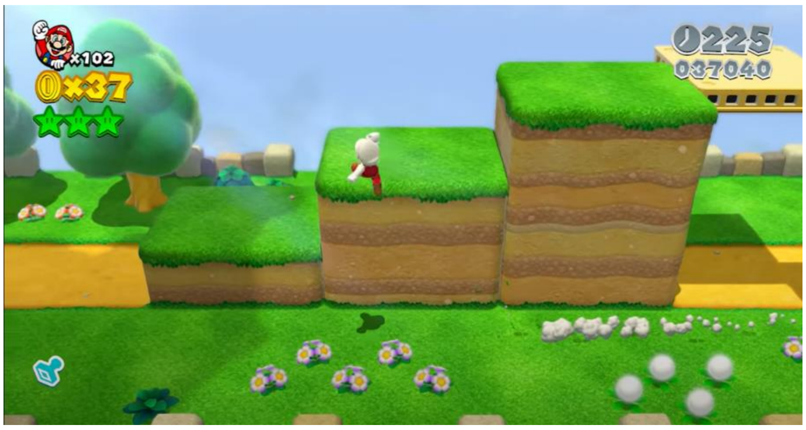
1.3. External influences

Videogames

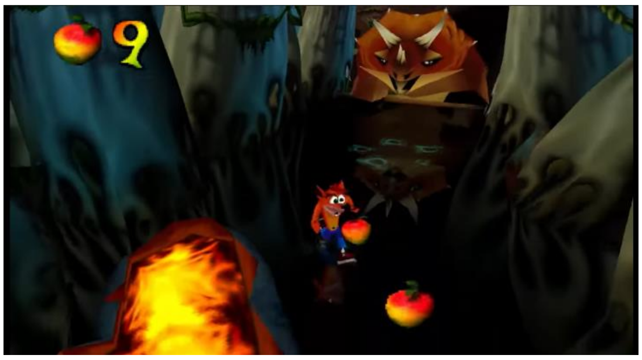
Astro's Playroom is a 3D platformer in which the player controls Astro Bot. Astro Bot can jump, hover, punch enemies and objects as well as spin attack by charging his punch.



Super Mario 3D World combines the free-roaming gameplay of the Super Mario franchise's 3D games with the mechanics of its 2D side-scrollers, including a level-ending flagpole and score system.



Crash 3 Warped is a platform game in which the player controls Crash and Coco Bandicoot, who must travel through time and gather 25 crystals scattered across time before Doctor Neo Cortex and Uka Uka do so. In particular, the triceratops's level has been recreated.



TV's series

Rick and Morty follows the misadventures of cynical mad scientist Rick Sanchez and his good-hearted but fretful grandson Morty Smith, who split their time between domestic life and interdimensional adventures.

2. History

2.1. Story

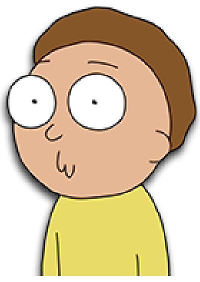
Scientist Rick Sanchez has discovered a new parallel world and to be able to explore it, he sends his latest invention to discover this world: the 626-robot, for friends Timmy.
Our protagonist Timmy gets on the interdimensional spaceship and sets out for this new world. While flying over the area, our hero is attacked by the local natives who make him lose control of the vehicle and falls into a forest. The ship is destroyed, and the pieces have scattered around the area. To return home, Timmy will have to recover all spaceship pieces by crossing various areas with different obstacles and dangers.

2.2. Characters

Timmy is the protagonist of the game, a robot created by scientist Rick Sanchez.

Rick Sanchez, a mad scientist.

Morty, Rick's stupid nephew.



2.3. Enemies

Each area is populated by different creatures, some similar and others completely absent from our world. They do not seem to accept strangers!
There are:

- *Normal Enemies*: these are those enemies that attack the player with melee attacks only. These enemies are **Skeleton, Slime, Spider, Rat, Wasp, Snake** and **Frog.**

- *Elemental Enemies*: these are those enemies that attack the player with elemental and, if it can, melee attacks. These enemies are **Fire Dragon, Ice Dragon** and **Electrical Bat.**

3. Gameplay

3.1. Introduction

The goal of the game is to complete all the levels to recover the spaceship components, so to allow Timmy to return home. To complete each level, the player will have to overcome various obstacles and enemies to reach the end of the level represented by electronic device.

3.2. Mechanics

In this section are described the mechanics of the elements in the game. At first, we want to have a look at the player, played by the user, and the enemies, controlled by AI.

- ***Player***
  - Life (start with 3);
  - Health, represented by a battery (start with 3, full battery);
  - Walk;
  - Jump;
  - Perform a slide in the direction where the player is moving;
  - Crouch down;
  - Attack enemies, both when grounded or when jumping;
  - Interact with some object;
  - Collect collectable object;
  - Place bombs
- **Enemies**
  - Walk;
  - Hit the player with melee attack;
  - Hit the player with elemental attack if they are elemental enemies.

Within the game the player can interact with various object that the player can found in the map described as following:

- ***Collectables***:
  - **Gears**: object useful to increase the score obtained at the end of the level. A gear adds 10 points to the final score;
  - **Gear Bonus**: obtainable by activating Bonus Block. A bonus gear adds 50 points to the final score;
  - **HP Battery**: when collect it allows the player to restore 1 health point, up to 3;
  - **Lives**: objects that allow player to come back to life after he lose all health points. If a life is collected is added to the current amount of life.
  - **Bombs**: object useful to destroy walls to unlock a new path. If an entity is in the range of the explosion, the bomb will damage them.
- ***Interactive Objects***:

  - **Bonus Block**: if hit, some collectables will appear around it which will disappear after 10 seconds.
  - **Checkpoint**: whenever the player loses a life, he will restart the level from the last activated checkpoint.
  - **Platform Button**: if jumped on it, this will enable some platforms to move.
  - **Destroyable Wall**: it blocks a passage to an accessible area. It can be destroyed placing a bomb in front of it to free the passage;
  - **Hive**: it is a Wasp spawner, when the player is close it spawn an Enemy Wasp every few seconds.
  - **Endpoint**: whenever this object is collected the player finishes the level.
- ***Environmental Objects***:
  - **Moving platform**: It can move horizontally or vertically;
  - **Popup platform**: After player jump on it after 3 seconds it will disappear, it reappears after 2 seconds;
  - **Trampoline**: It allows player to make a higher jump;
- ***Obstacles***:
  - **Cannon**: fires cannon balls every few seconds;
  - **Log waves**: more logs roll along a path;
  - **Rotating logs**: which spin and can hit the player;
  - **Falling Blocks**: blocks that fall to the ground and smash player;
  - **Lava pools**: a lava pool that must be avoided to not die;
  - **Boulder**: a giant rock that fall from a mountain and the player must run away from it.

3.3. Victory and Game Over conditions

3.3.1. Victory Conditions

To win, the player must reach the end of each level where the electronic device is located. The secondary objective is to collect all the collectables to increase the final score and make a new record.

3.3.2. Game Over Conditions

The player can receive two types of damage:

- **Normal**: the player will just lose 1 hp.
- **Fatal**: the player will die.

If the player loses all health points he dies, one life is removed and he re-spawns at the last checkpoint activated.
If the player loses all lives, it will be shown the Game Over screen and player will have to re-start the level from the beginning. When he respawns after the game over, he will restart with 3 lives.

3.4. Save system

The game allows you to keep track of the progress and the game settings. On two different files will be stored settings about the config and progress regarding to the player. Every time that the user opens the game last settings about config are loaded. The config settings are saved every time that the user close main menu option or pause menu. The progress made by the user are loaded if the player wants to continue where he left, otherwise he can start a new game and reset all the progress made. The progresses are saved every time that the user change level.

3.5. Dynamics

3.5.1. Player

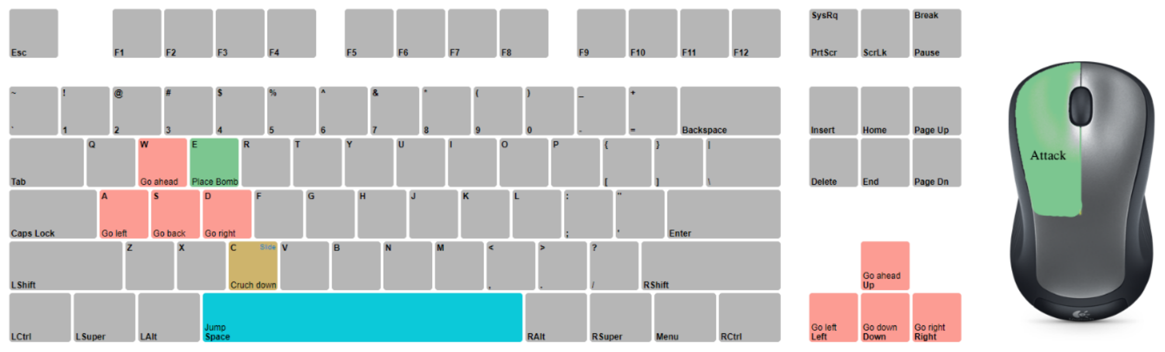
The player starts in the certain point of a level and his main aim is to reach the endpoint represented by electronic device. The path is populated by various enemies that make more difficult to reach the end of level. He can kill these enemies with various attacks (Melee attack, Slide, jumping above them) and, when he kills them, he obtains a gear.
These gears can be collected, they are useful to increase the final score; after the player has collected 100 gears an extra life is added. There are also gear bonus obtainable by activating the Bonus Block, that spawn some gear bonus which disappear after 10 seconds, so the player must be fast to collect all of them.
Along the path there are other obstacles that player must overcome, like Rotating Logs, Log Waves and Falling Block that can hit or kill the player.
To make more difficult the path there are some parkour sections formed by moving platform or higher point that the player can reach by jumping on bouncy objects like mushrooms; some of these moving platforms are activatable by a red button. The player must be careful to not fall from the edge of the lands where he moves, otherwise he will die. Lastly, the player can access secret zone hidden by a wall that can be destroyed placing a bomb in front of it; the bomb could be used also to hurt enemies, but it can also damage the player.

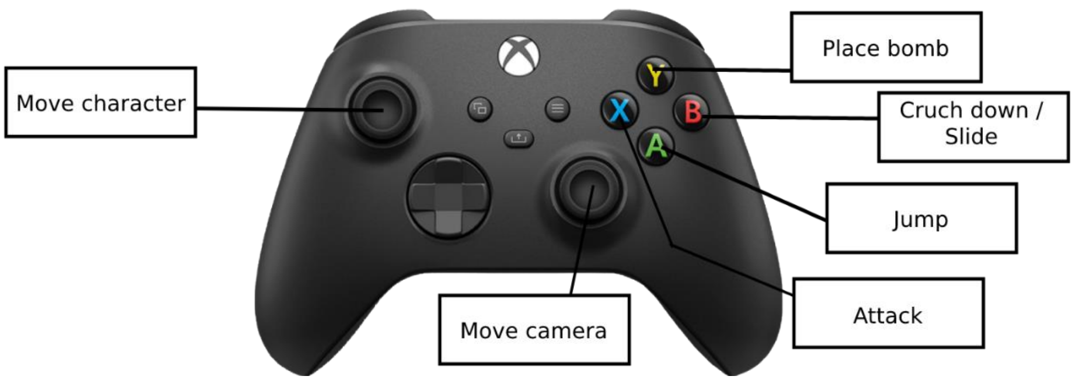
3.5.2. Enemies

The enemies are placed in the various levels. When player enter in the enemy's field of view, the enemy become warned and can reach player to attack him or wait until player is so close to attack him.
Some enemies can go anyway, others follow a patrol path and others can stay still.
Some enemies are in a sleeping mode, they wake up only if player satisfy some conditions, for example, if player reach a certain point.
Only the elemental enemies can hit player with elemental attacks, the dragon have also a melee attack to hit player if he is a lot close.

3.6. Commands

The player can use both **Keyboard and Mouse** or **Joypad** to interact with the game:





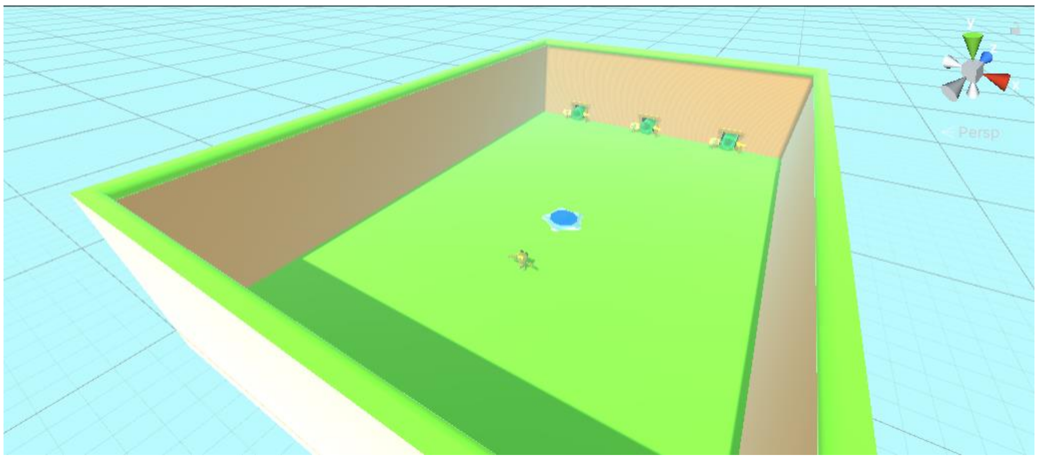
Button ESC (for Keyboard) or **Options** (for Joypad) is used to pause the game.

4. Levels

The goal of every level, except for HUB, is to reach the endpoint. The player, as optional task, can collect every collectable to get the maximum score for that level.

4.1. HUB

This is the first "level" where the player starts to play. In this scene the player can enter in a pipe for a specific level.



Dialogues:

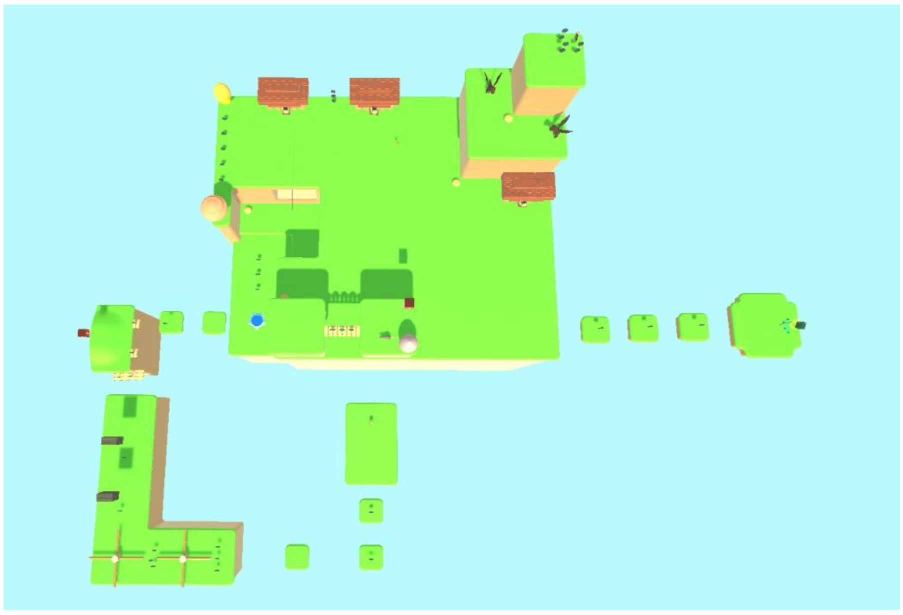
- D1: when player starts the game.
    - "Stupido Timmy, ti mando ad esplorare nuove dimensioni al posto mio e la prima cosa che fai e' rompere l'astronave!!"
    - "Per poter tornare qui e farti resettare, devi trovare tutti i componenti sparsi in giro e portarli qui."
    - "Perfino tu non dovresti avere difficolta', tanto per muoverti devi solo usare WASD o l'analogico sinistro."
    - "Per guardarti intorno usa il Mouse oppure l'analogico destro."
    - "Suppongo che per cambiare zona ti devi ficcare in uno di quei tubi."
    - "Nel frattempo io vado a scolarmi una bottiglia di whisky."
- D2: when player is near to pipes.
    - "Stupido Timmy, devi accovacciarti con il tasto C, o B del Joypad, per poter entrare in quel tubo."
- D3: is showed when player finish all three levels.
    - "Mmmm, non vedo altri tubi in cui ficcarsi."
    - "Mi sa che il gioco per il momento finisce qui."
    - "Ma dato che non hai ottenuto abbastanza pezzi di ricambio, mi sa che dovrai rimanere li."
    - "Non verro' di certo io a riprenderti!"
    - "Buona permanenza!!"

- “Ah un’altra cosa… Il Gemini team ti ringrazia per aver giocato!”
- “Crediti:\nThomas Voce\nGianfranco Sapia\nAndrea De Seta”

Another feature of this “level” is that when player is near a sign, he can press submit button to display the level score.
If the level was already completed, it shows: “Record: “plus the level score.
Otherwise, it shows:
“Non hai fatto ancora nessun punteggio!\nDatti una mossa!!”

4.2. Level 1

This first level works like a tutorial and it is useful to get knowledge with the various mechanics of the game.



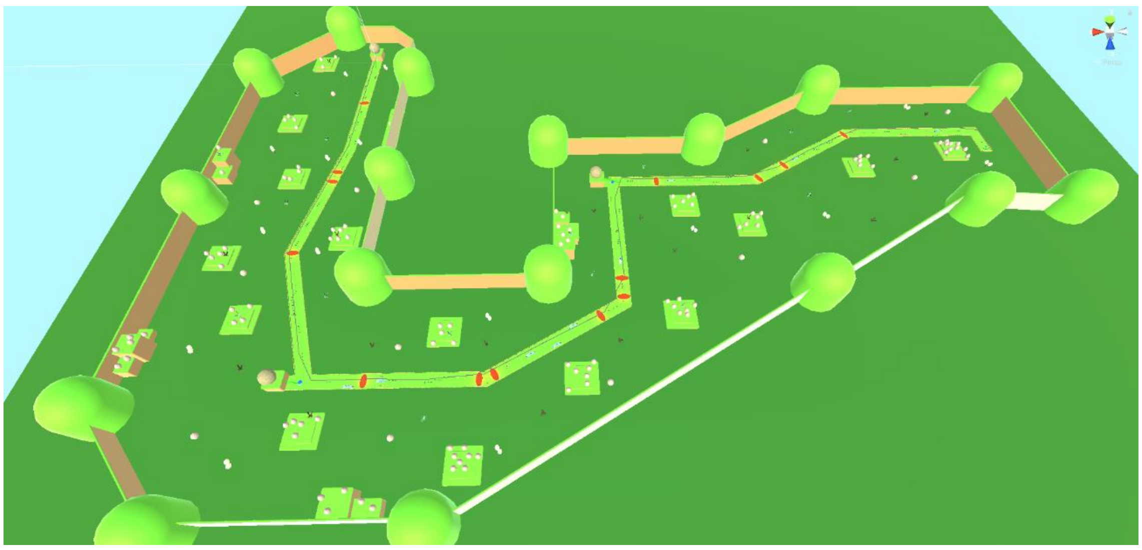
Dialogues:

- D4: when level 1 starts.
    - “Dannazione, ma che razza di mondo e’ questo?!”
    - “Per proseguire dovrai saltare… Premi SPAZIO o il tasto A del Joypad.”
    - “Effettivamente avrei potuto montarti dei razzi, ma per un rottame come te sarebbe uno spreco!”
- D5: in front of the first BonusBlock
    - “Ehi, quella e’ una cassa Bonus!”
    - “Attivala saltandoci di sotto per ottenere ingranaggi extra!”
- D6: in front of breakable wall
    - “Mmm… Quel muro sembra fragile… Li vicino c’e’ una bomba, usa il tasto E, o Y del Joypad, per piazzarla e distruggere quel muro.”
- D7: in front of first mushroom

    - “I tuoi radar mi segnalano che qui sopra c’e’ molta attivita’… ti consiglio di prepararti!”
    - “Se qualche creatura ti volesse attaccare, usa il tasto sinistro del mouse, o X del Joypad, per attaccare.”
    - “Oppure puoi sempre ricorrere all’antica arte della fuga…”
    - “Quando sei in movimento, usa il tasto C, o B del Joypad, per eseguire una scivolata.”
- D8: in front of skeleton in the cave
    - “Bene, hai trovato una batteria! Con quella puoi recuperare un HP!”
    - “Non rilevo pericoli, raccoglila pure!!”
- D9: in front of life collectable
    - “Hai trovato una vita! Vabbe e’ inutile, tanto morirai miseramente…”

4.3. Level 2

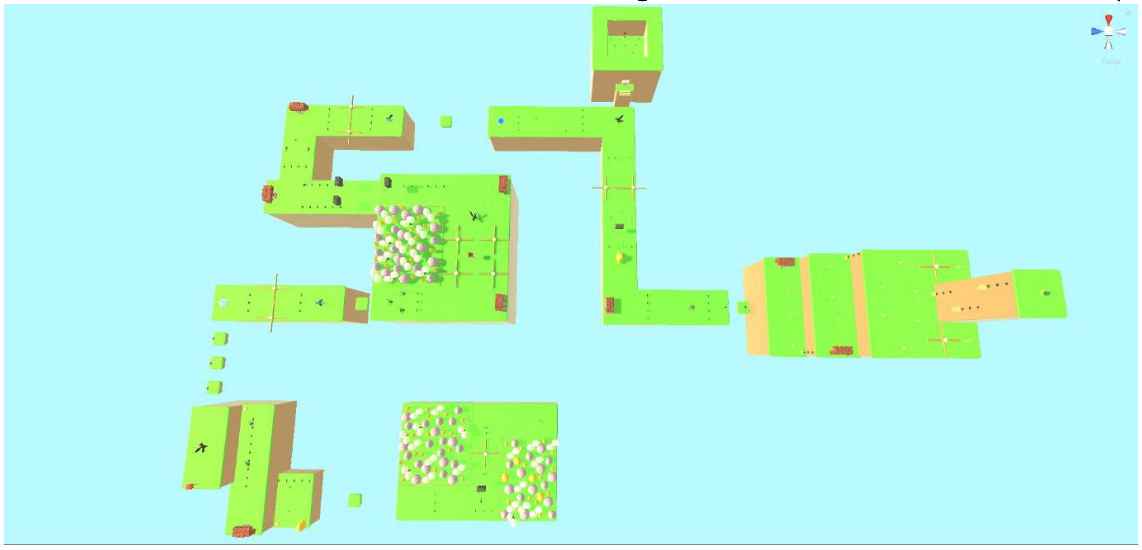
The level flow of level 2 is particular: in this level the player is chase by a boulder and he must run as fast as he can and reach the endpoint.



Dialogues:
- D10: when level 2 starts.
    - “Preparati a fare una bellissima e tranquillissima passeggiata… ;)”

4.4. Level 3

The flow of the level 3 is like level 1. The level is longer with more enemies and more trap.



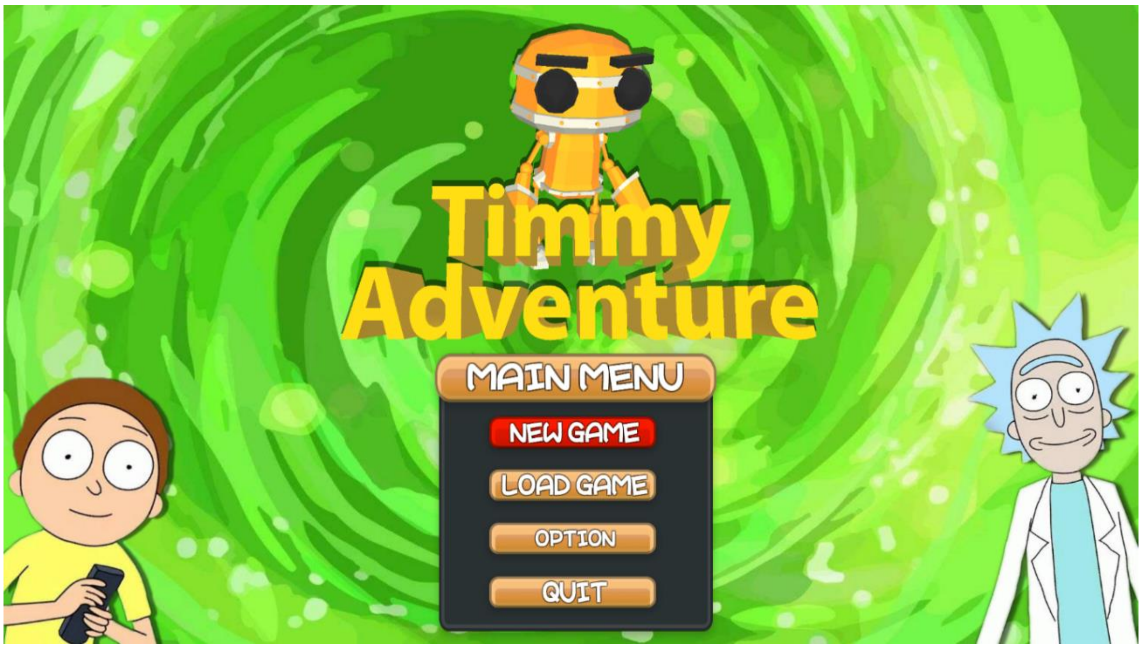
Dialogues:
There are not any dialogues in this level.

5. UI Design

The UI is composed by the following element:

- Main Menu
- Main Menu Options
- HUD
- Pause Screen
- Pause Screen option
- Dialogues
- Victory screen
- Game Over screen

5.1. Main Menu

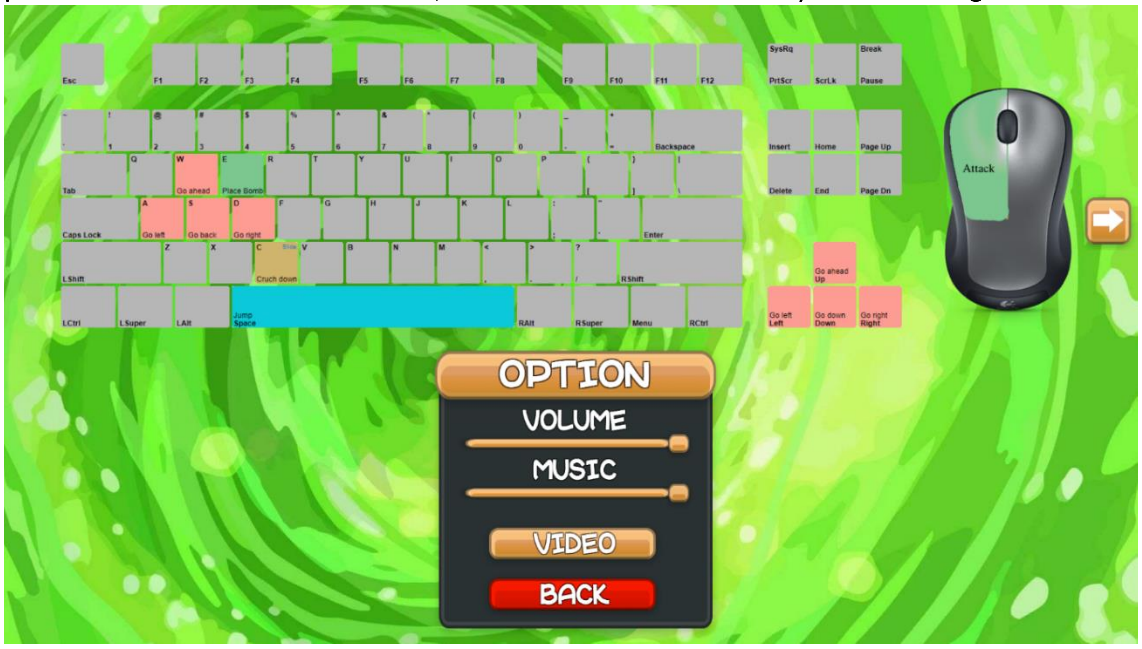


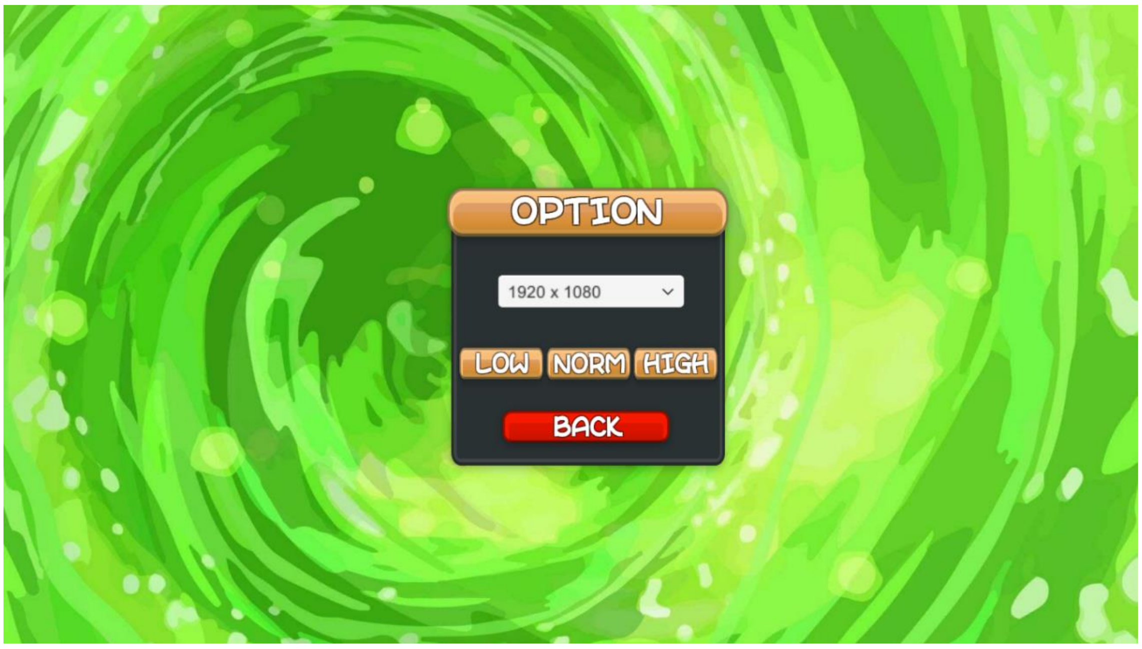
When the game is started, this is the first menu that will be shown to the player; It contains:

- **New game**: start a new game; new save files will be created;
- **Load game**: load the game from the save files with actual progress; if there is no save file this button cannot be pressed;
- **Option**: where player can audio and video settings;
- **Quit**: exit the game.

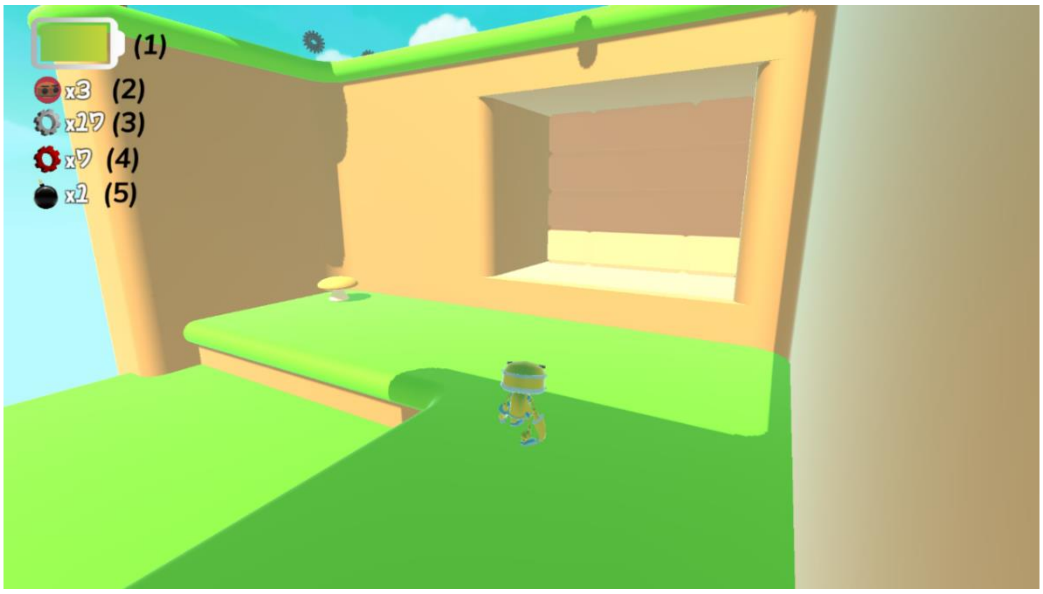
5.2. Main Menu Options

From the Main Menu we can access to a Main Menu Options screen where we can modify parameter for Volume and Music, check the control or modify video setting.





5.3. HUD

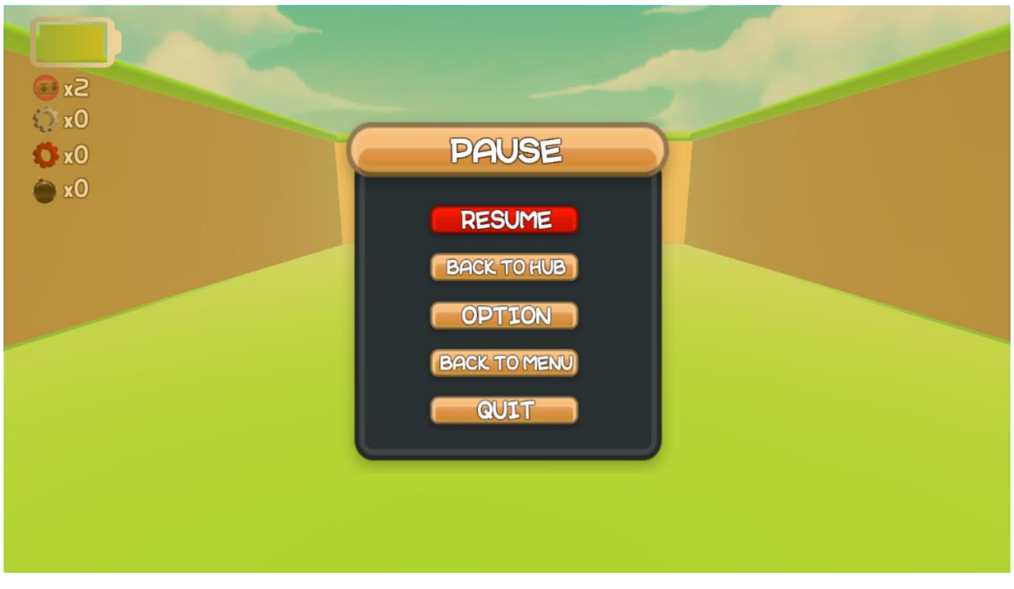


While playing the game on the interface will be shown:

1. Amount of health remaining with the battery;
2. Amount of life remaining;
3. Number of gears collected;
4. Number of bonus gears collected;
5. Number of bombs collected;

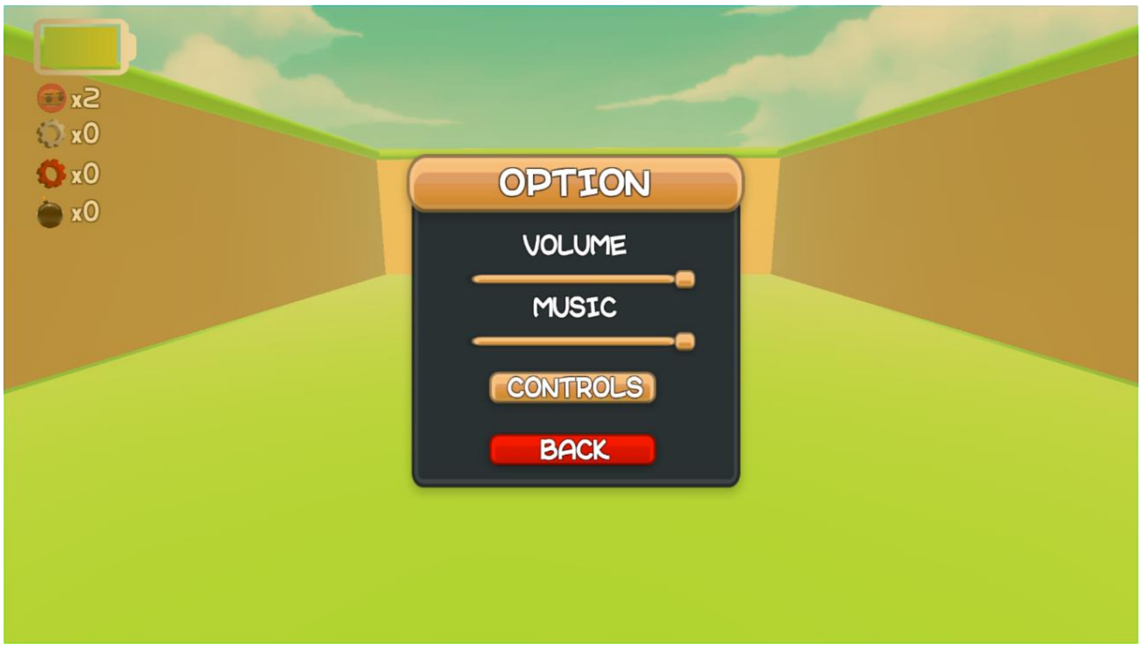
5.4. Pause screen

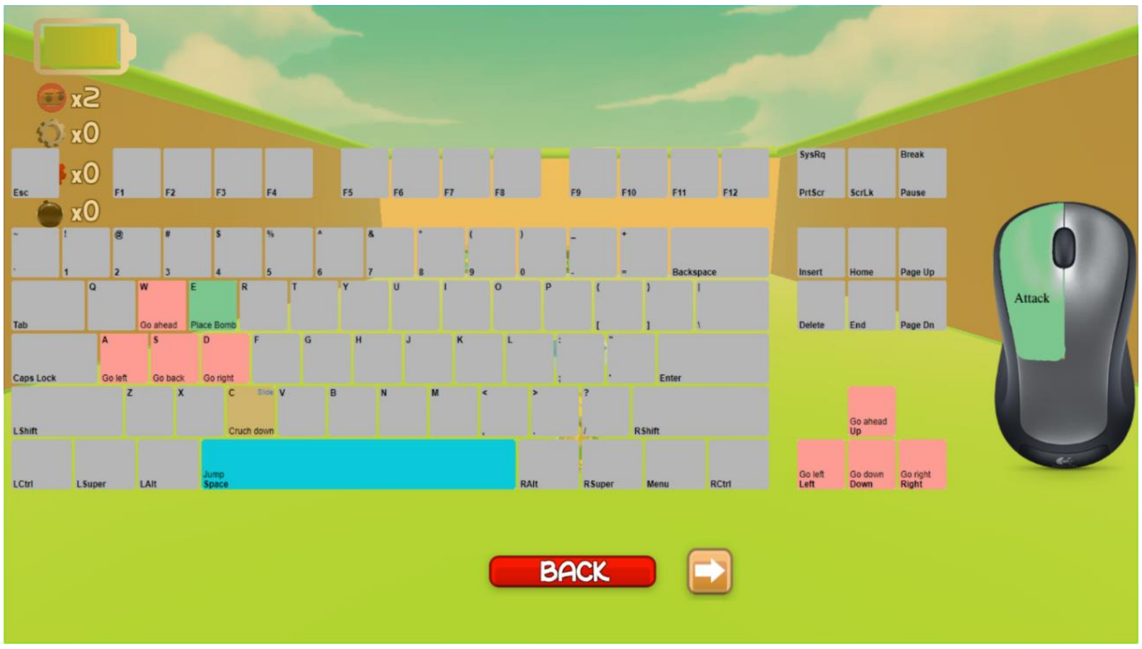
In the pause screen, the player can resume the game, go back to the HUB Level, go to options where audio parameters can be modified or control can be seen, or quit the game.



5.5. Pause Screen Option

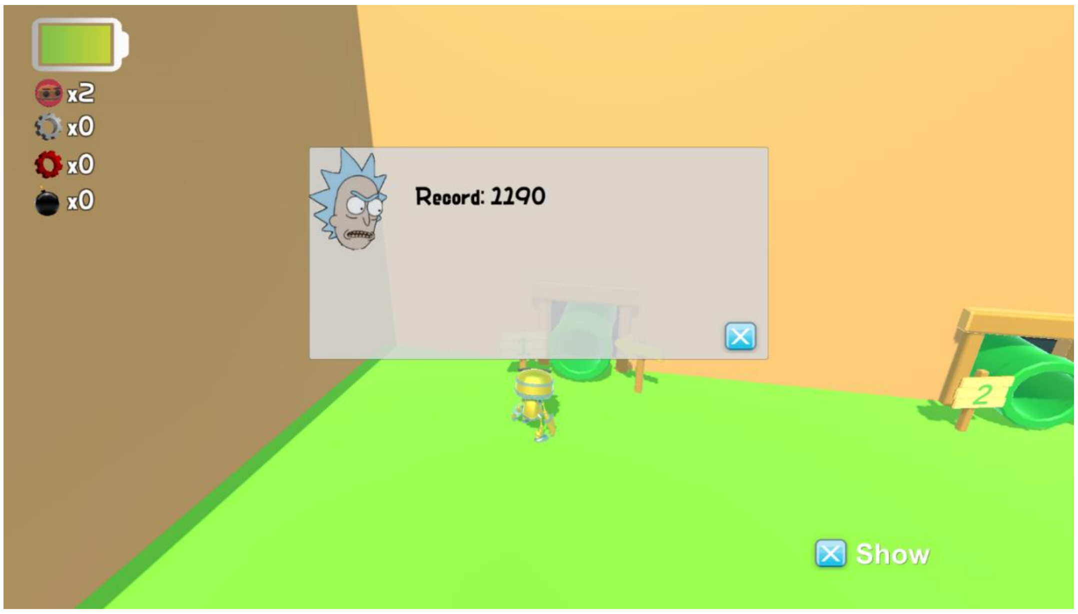
From the Pause screen we can reach the Pause Screen Option where we can modify the audio setting or go to controls and check the command used to play





5.6. Dialogues

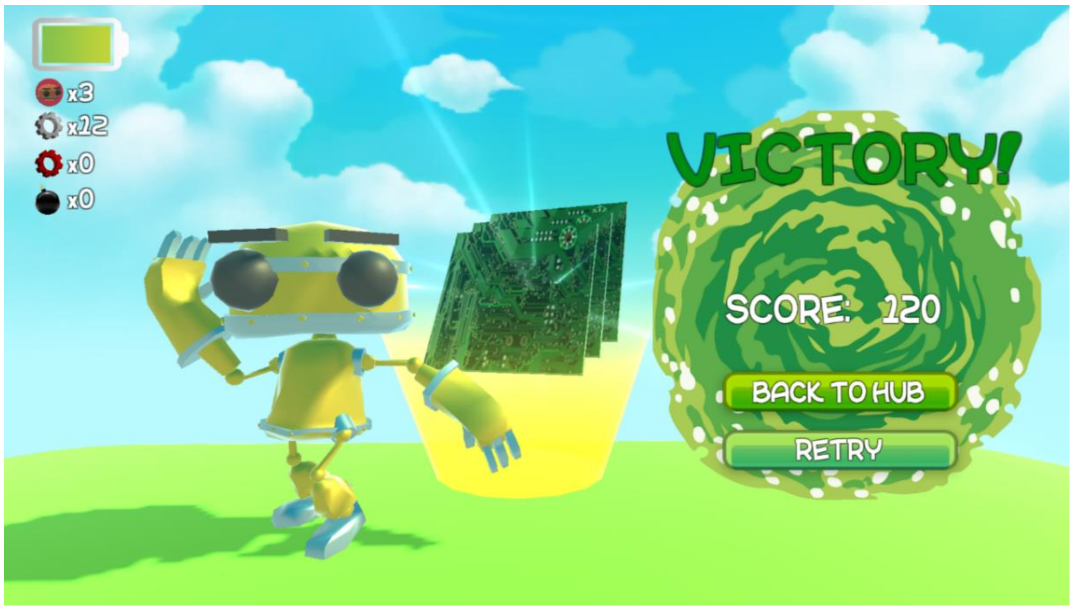
During the game, some dialogues may appear on the screen; the player can skip the dialogue using the button ENTER.



Example of dialogue

5.7. Victory Screen

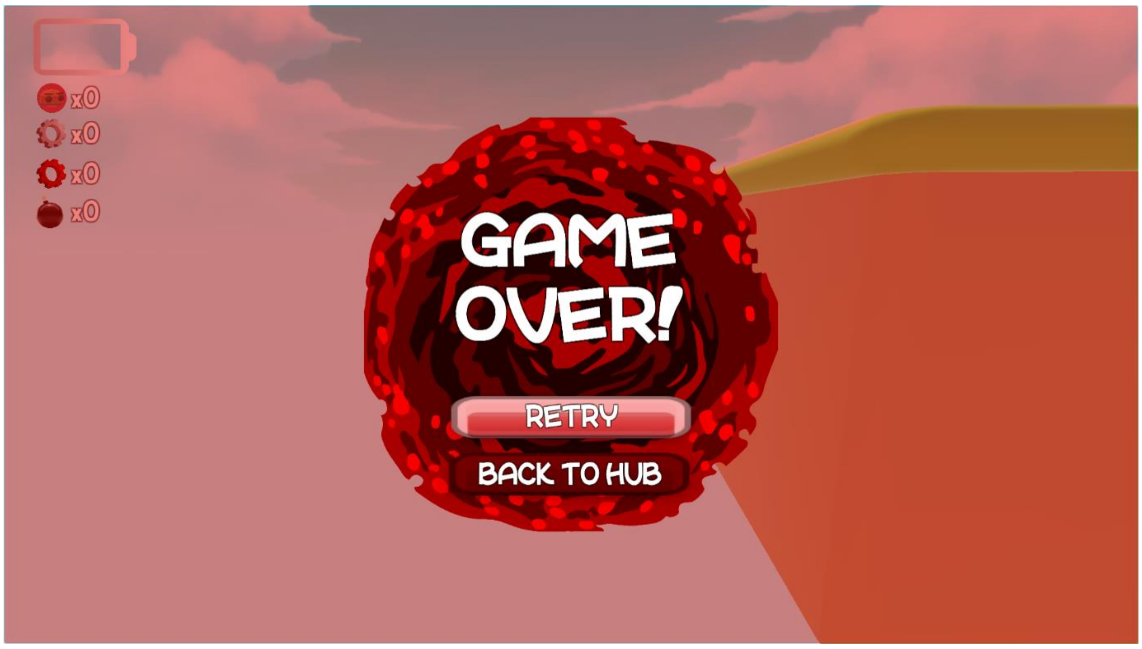
When the player reaches the final point of the level will be shown a victory screen with the scored achieved and the button to retry the level or go back to Level HUB.



Victory screen

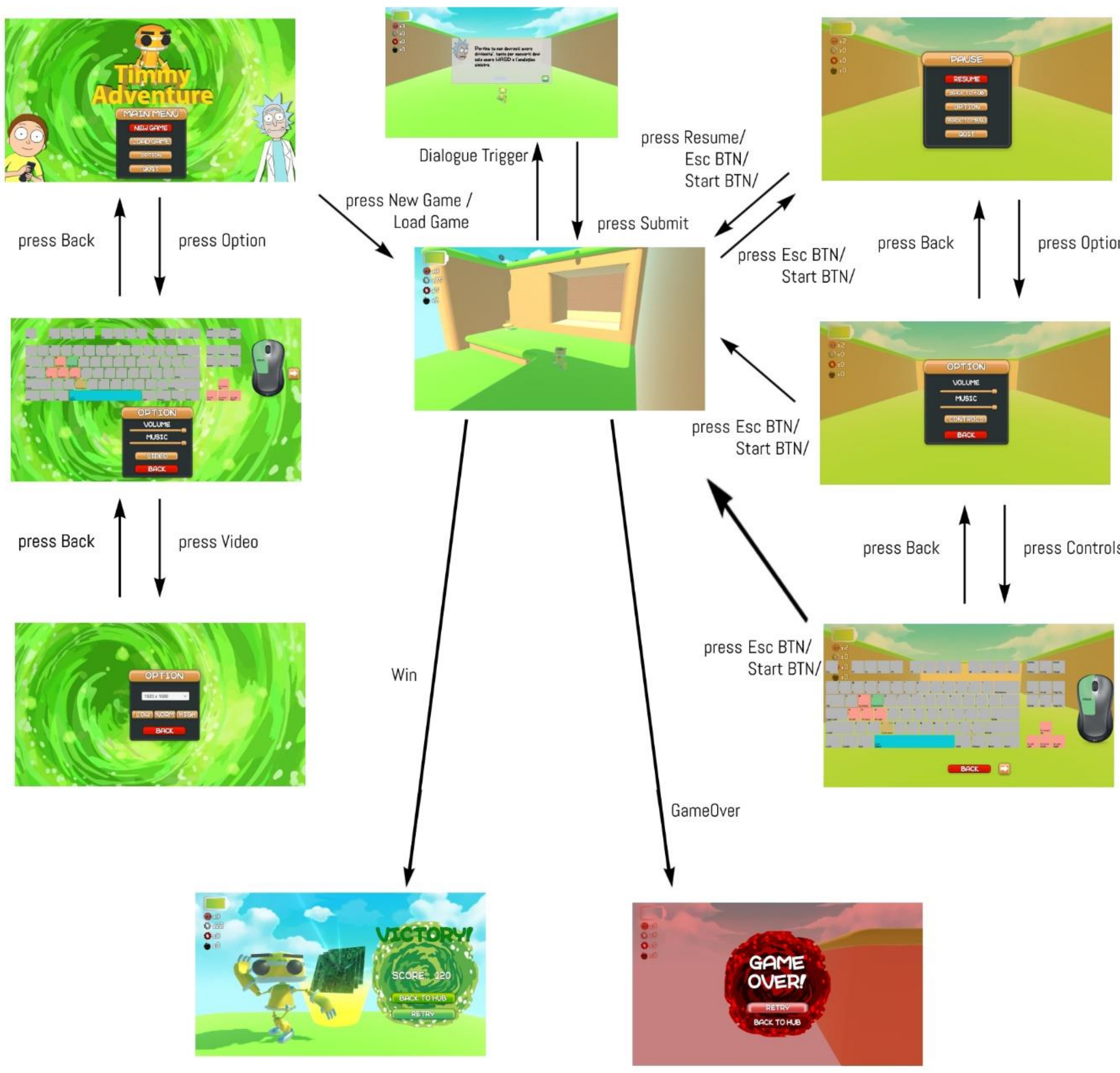
5.8. Game Over Screen

When the player loses all his life will be shown a game over screen with the options of retry the level or going back to Level HUB.



Game over screen

5.9. UI flowchart



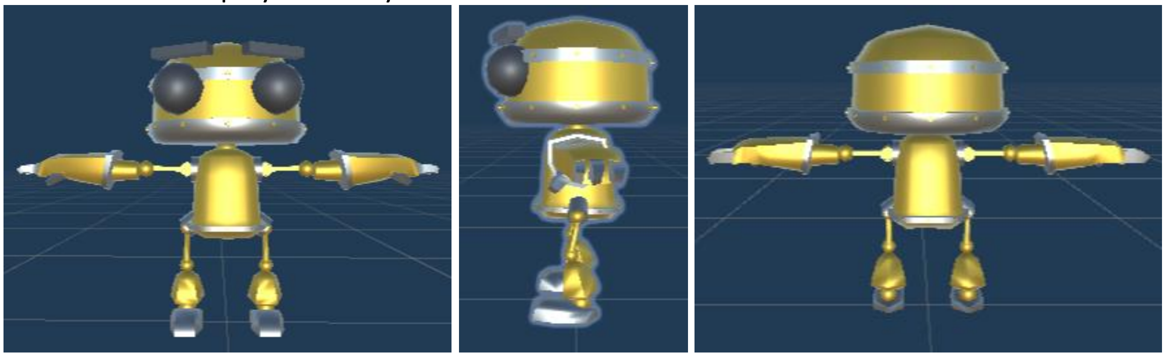
6. Art Assets

6.1. 3D models

The 3D models used for the project are all in .fbx format, so animations can be added if necessary. All the 3D models are found on internet, but some of them, for example Checkpoint or Endpoint, have been created using Blender.

6.1.1. Player

The model for our player Timmy is a robot available at the link.



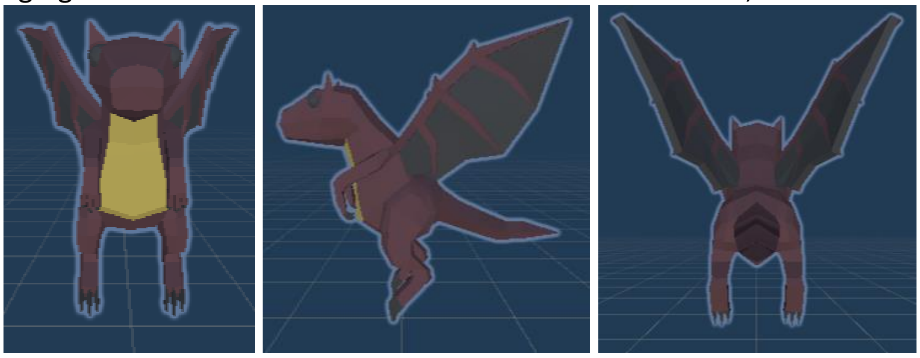
Timmy's model in T-Pose (front, side and back)

6.1.2. Enemies

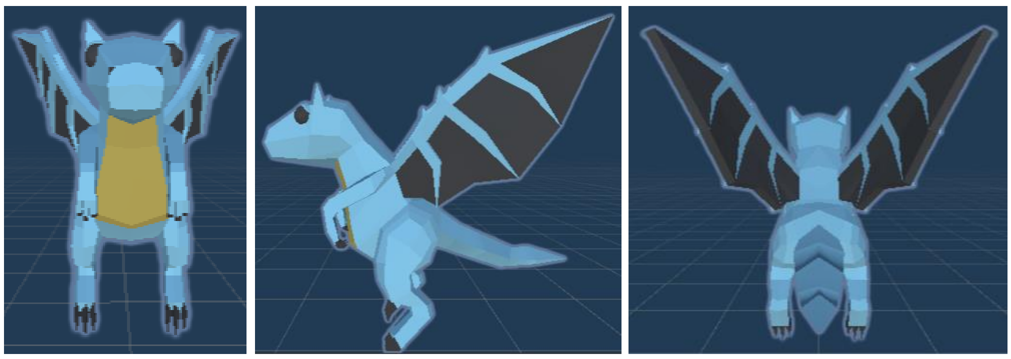
The enemies the player can meet while playing are the following:

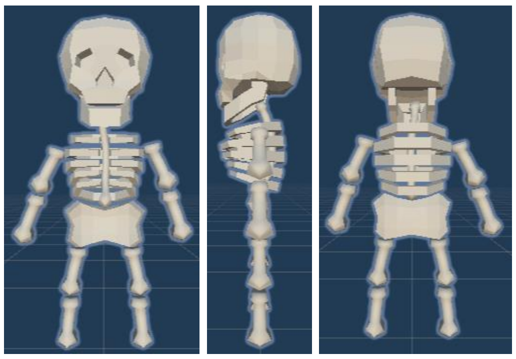
- Dragons and Ice Dragons;
- Skeletons;
- Wasp;
- Rat;
- Frog;
- Slime;
- Snake;
- Spider;
- Bat.

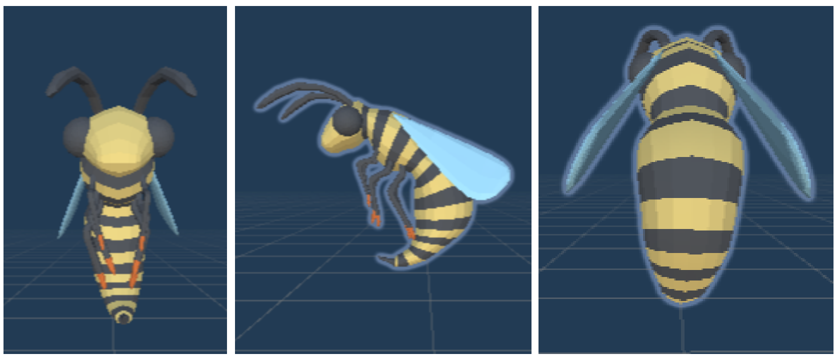
The following figures are the models of the enemies seen from front, side and back.

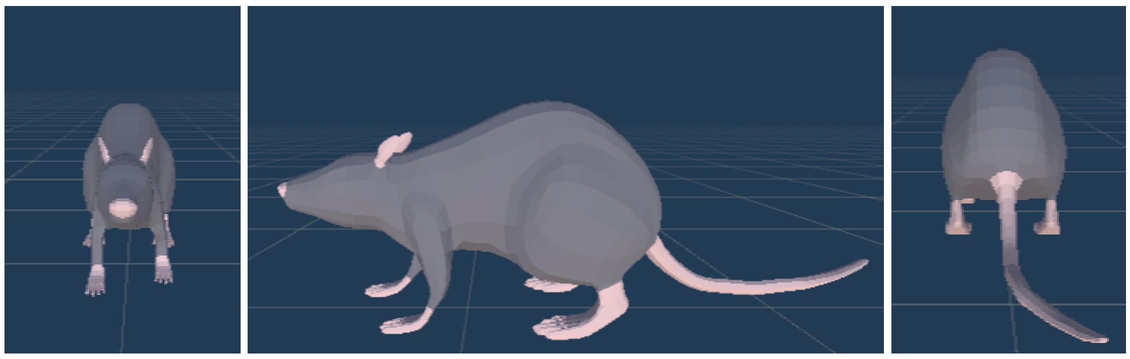


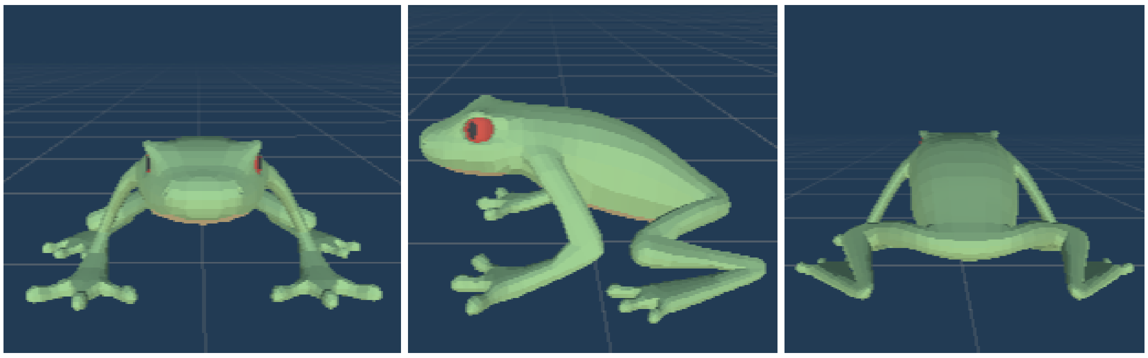
Dragons' model (front, side and back)

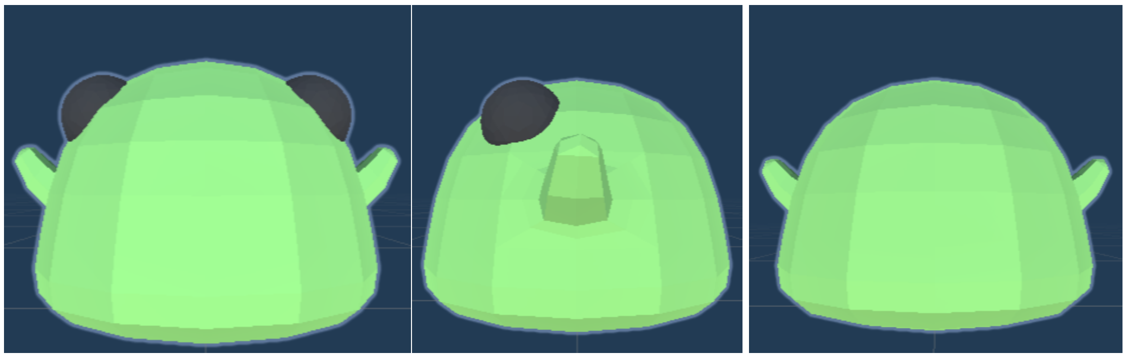

Ice Dragons' model (front, side and back)

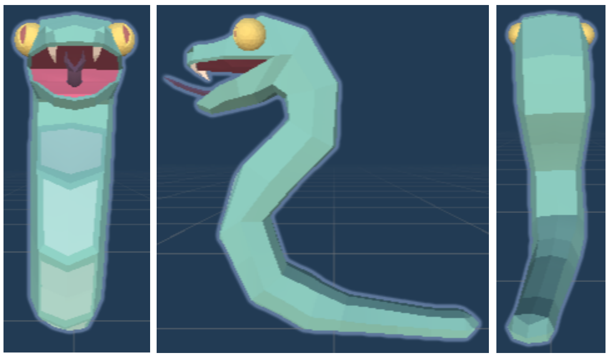

Skeleton' model (front, side and back)

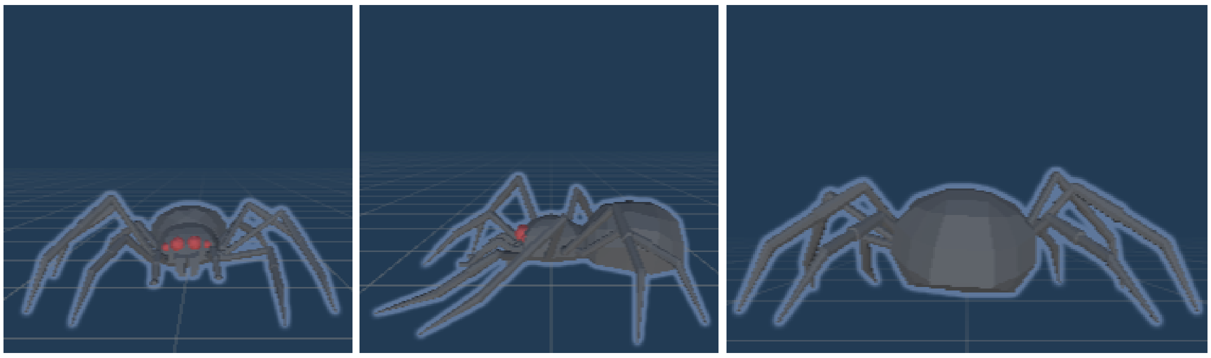

Wasp's model (front, side and back)

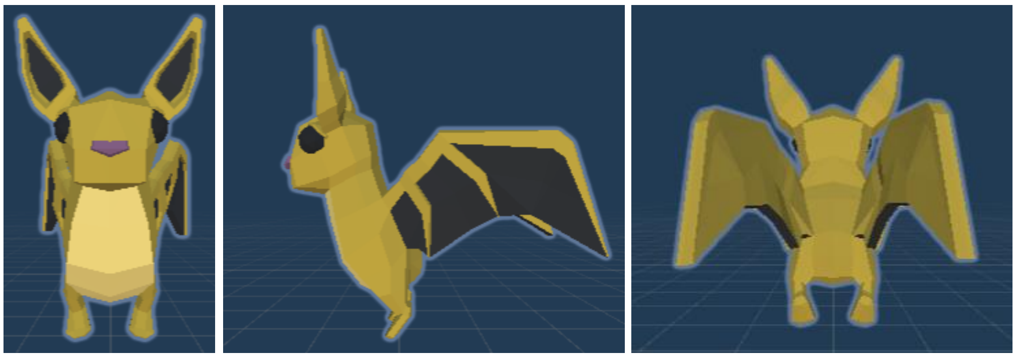

Rat's model (front, side and back)


Frog's model (front, side and back)


Slime's model (front, side and back)


Snake's model (front, side and back)


Spider's model (front, side and back)

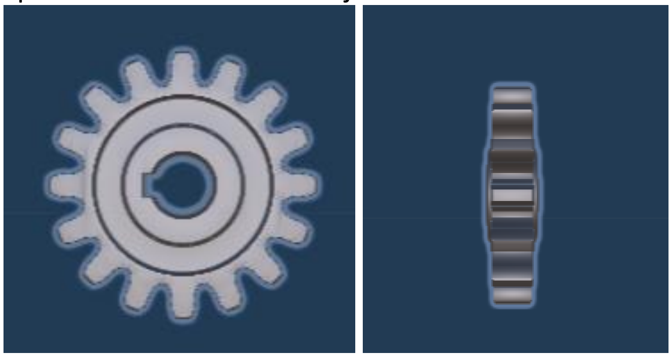

Bat's model (front, side and back)

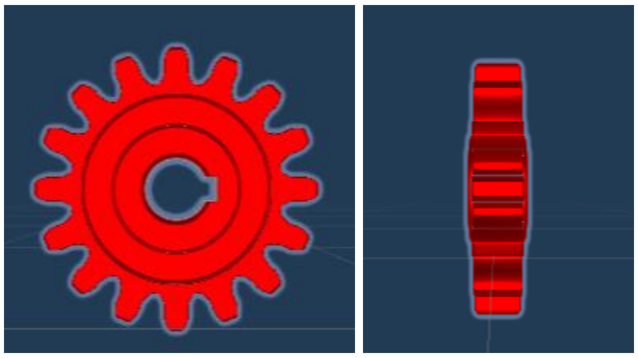
6.1.3. Collectables

The following are all 3D models related to collectables object:

- Gear;
- Gear Bonus;
- Lives (created manually);
- Battery;
- Bombs.

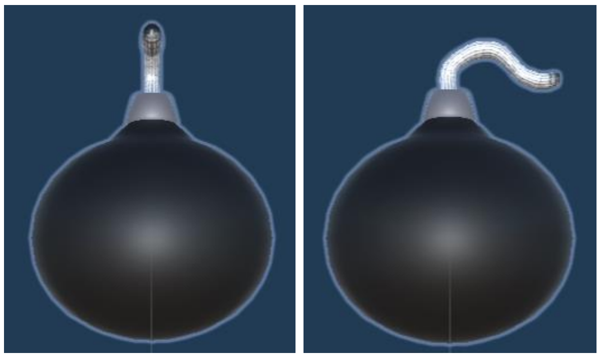
The following are the picture of collectables object.


Gear's model (front and side)


Bonus Gear's model (front and side)

Life’s model (front and side)

AA
ALKALINE
Energizer
HP Battery’s model (front and side)


Bomb’s model (front and side)

6.1.4. *Interactive object*

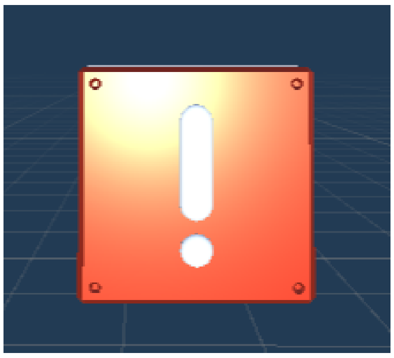
The following are the 3D models for related to Interactive Objects:

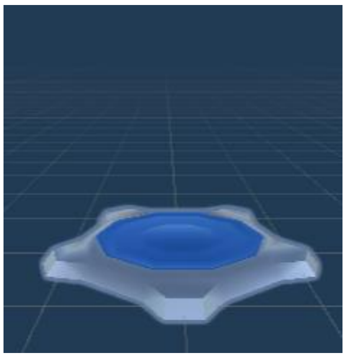
- Bonus Block;
- Checkpoint;

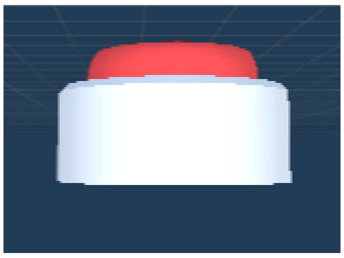
Platform Button;

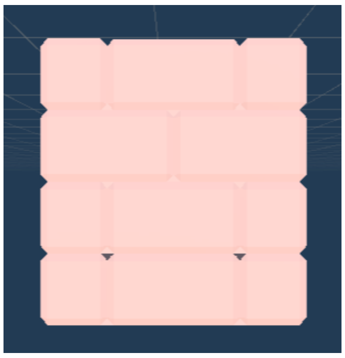
- Destroyable Wall (created manually);
- Hive (created manually);
- Endpoint (created manually).

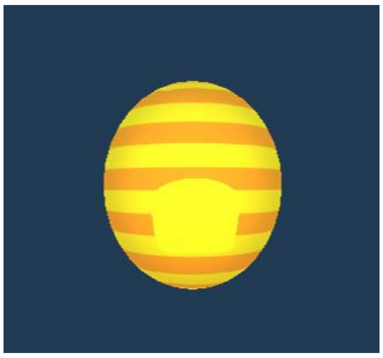
These are the figure of these objects.

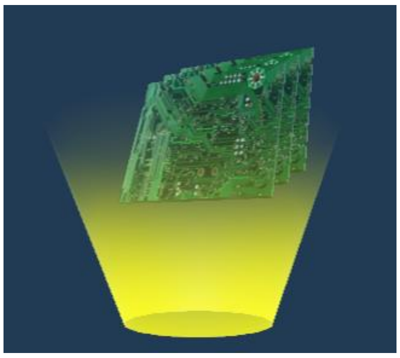

Bonus Block model


Checkpoint model


Platform Button model


Destroyable Wall model


Hive model

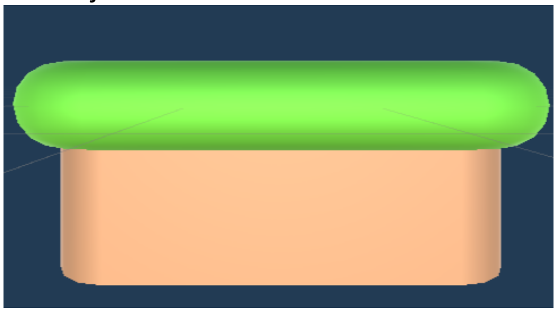

End point model

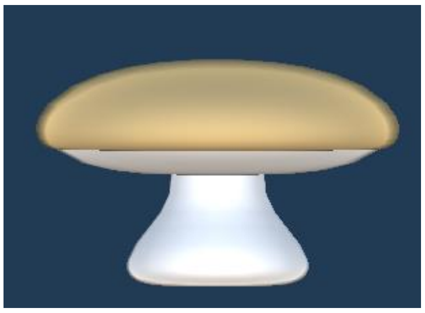
6.1.5. Environmental Object

The following are the 3D models for related to Environmental Objects:

- Moving Platform and Pop-up Platform (same 3D model);
- Trampoline, 3D model of the mushroom of MAST.

These are the figure of these objects.


Platform model (both Moving and Pop-up)


Trampoline model (Prop_Mushroom_2 in MAST)

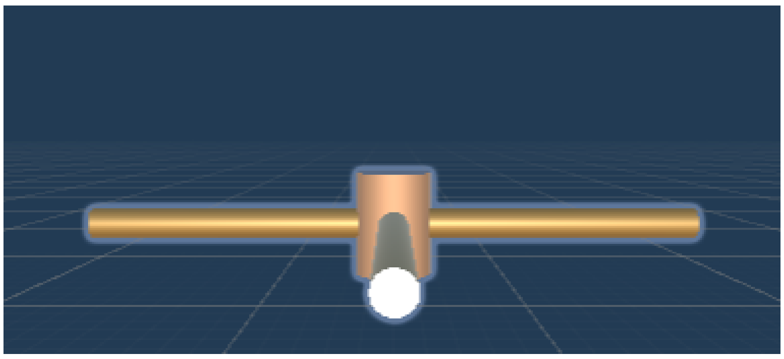
6.1.6. Obstacles

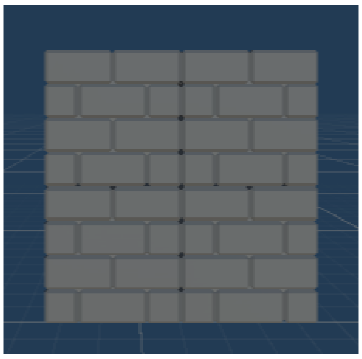
The following are the 3D models for related to Obstacles Objects:

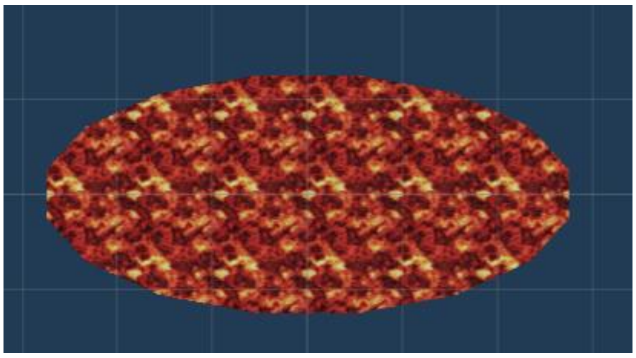
- Cannon;
- Rotating Log;
- Falling Block;
- Lava Pools;
- Boulder (created manually).

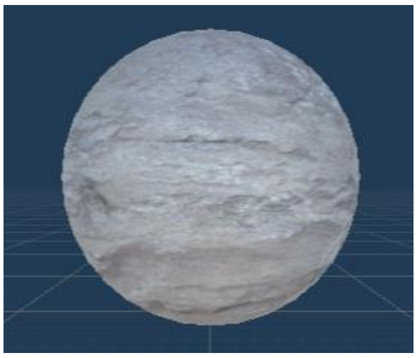
These are the figure of these objects.

Cannon model


Rotating Log model

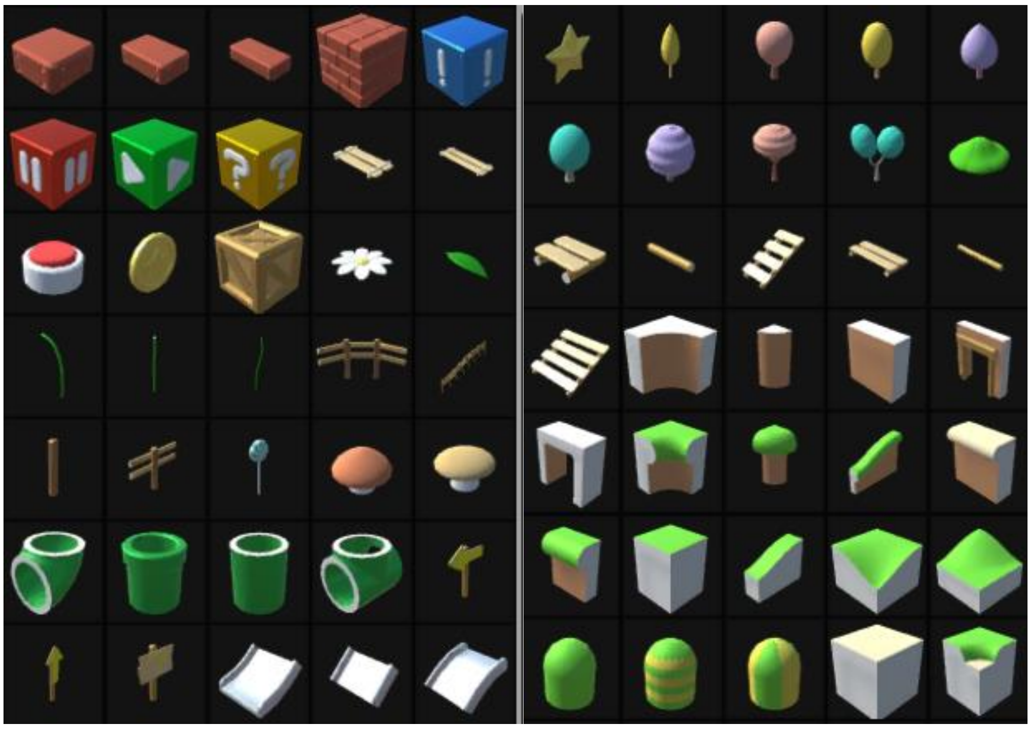

Falling Block model


Lava pool model


Boulder model

6.1.7. 3D models for creating levels

The 3D models used for the creation of the level are the combination of some of the single component offered by the tool MAST. The following figure show what model could be used for the creation of the level.



6.2. 2D sprites

The 2D sprites are in .png format and they are created using images found on internet and adding details. Below are described all the sprites used in some part of the project.

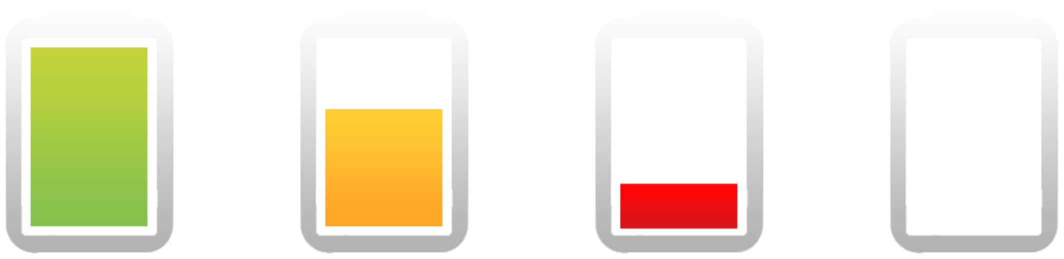
6.2.1. Logo

Timmy
Adventure

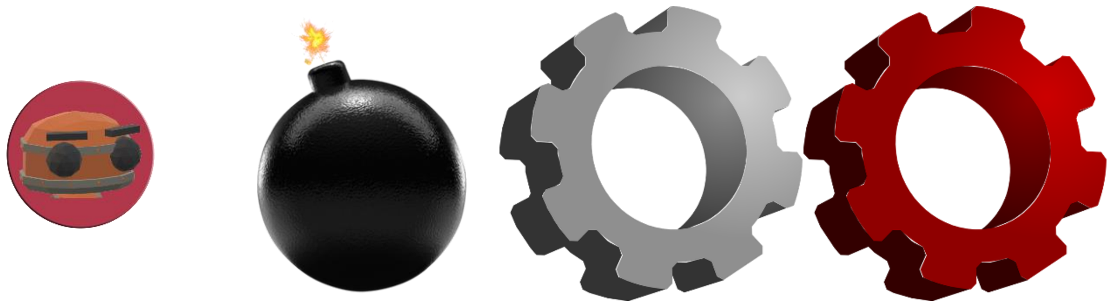
6.2.2. HUD

HUD is composed by the Player UI and the Dialogues.

- Player UI



Battery HP in all its states



Icons for player statistics

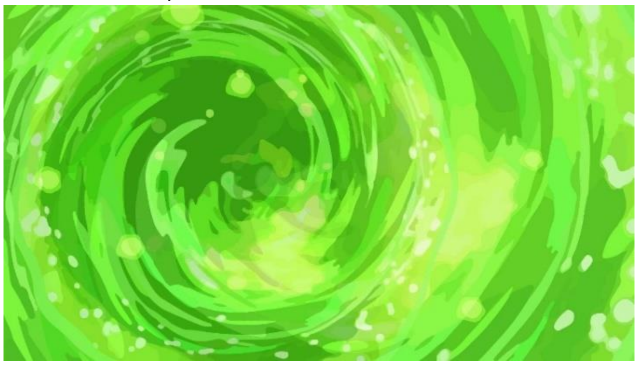
- Dialogue

Dialogue Box

6.2.3. Menu UI

Menu UIs are composed the by various images; the main background images are the following:

- Background for main menu;


Background menu

- Background for pause;

Background box, inside we can find buttons.

- Background for victory screen;

Victory screen background, inside we can find buttons.

- Background for game over screen;

Game over screen background, inside we can find buttons.

6.2.4. Skybox

The skybox used in all levels is the cubemap of the following pictures:

Skybox used for all the scenes.

6.3. Animations

The animations have been added to our 3D models to animate them with custom movement. Player animations have been downloaded from Mixamo. Enemies' animations were already present in their package. Only some animations about other object are discussed here because some of them are animated by script.

6.3.1. Player

The following are all the animation that the player can do during the game.

- **Idle**
  - Idle
  - Idle 1
  - Idle 2
- **Walking**
  - Walking Left
  - Walking Right
  - Walking Forward

    - Walking backward
- **Running**
    - Running Left
    - Running Right
    - Running Forward
    - Running backward
- **Jumping**
    - Jump start
    - Jump falling
    - Jump end

- **Attacks**
    - Attack punch left
    - Attack punch right
    - Attack air kick
- **Hitted**
- **Slide**
- **Crouch**
    - Idle to Crouoch
    - Walk crouched
    - Walk left crouched
    - Walk right crouched
    - Walk backward crouched
    - Crouch to Idle
- **Death**
    - Death 1
    - Death 2
    - Death mashed (created manually)
- **Victory**
    - Victory 1
    - Victory 2
    - Victory 3

6.3.2. Enemies

Frog, Rat, Snake and Spider have the same structure with the following animation:

- **Idle**
- **Walking**
- **Attack**
- **Death** (created manually for the Snake)
- **Despawn** (created manually)

Skeleton and slime have similar structure. Unlike the enemies above, they also have the following animations:

- **Inactive**
- **Spawn**

Wasp, Dragon and Ice Dragon have similar structure with the following animation:

- **Idle**

- **Attack**
- **Death**
- **Fall down** (created manually)
- **Despawn** (created manually)

Bat has similar structure to the enemies above, except for the animation "**Attack**".

6.3.3. Other animations

- **Boulder**

The boulder keeps rotating until it stops.

- **Endpoint**

The electronic device makes a little jump.

- **Collectables**

This object rotates on itself until it is collected.

- **Bonus block**

When this object is triggered, it will go up and then down to its original position.

- **Platform button**

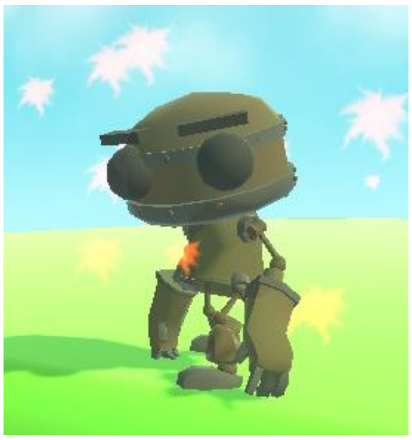
When this object is triggered, it will go down to look like a pressed button.

6.4. Visuals

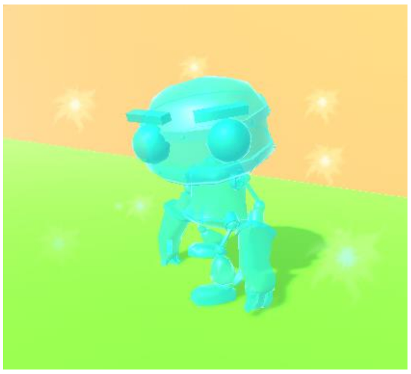
The particles used in this game are downloaded from the web or created manually in Unity.

Player

- Burned

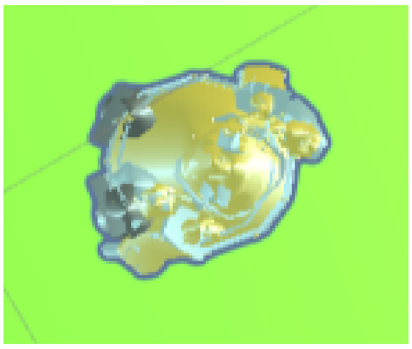


- Frozen

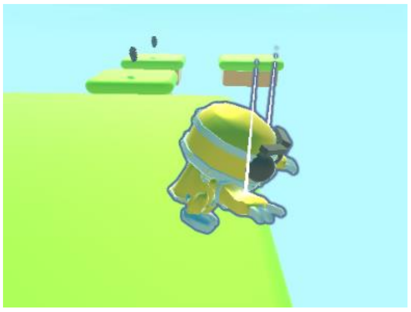


- Hitted

- Mashed

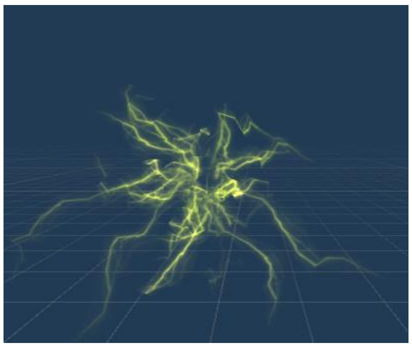


- Trail renderer

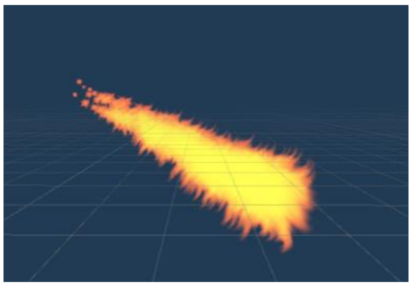


Enemies

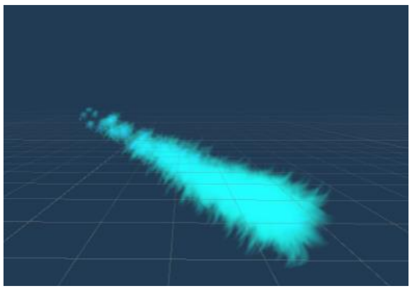
- Bat attack



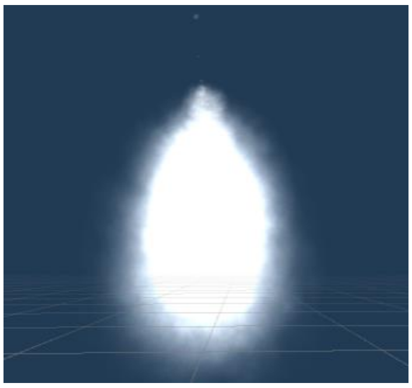
- Dragon fire spit



- Ice dragon spit

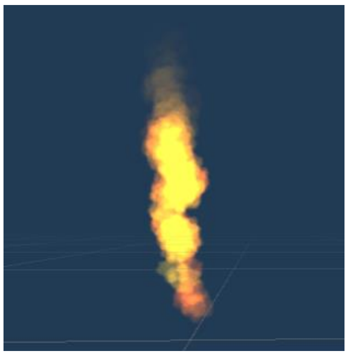


- Hive spawn smoke

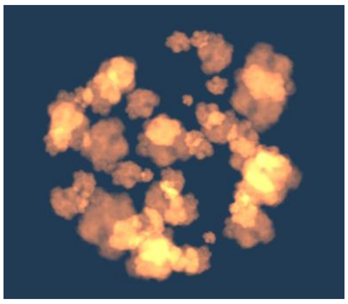


Interactive object effect

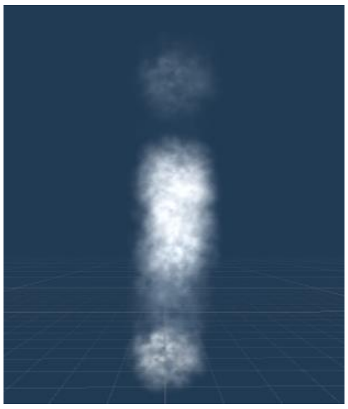
- Bomb muse fire



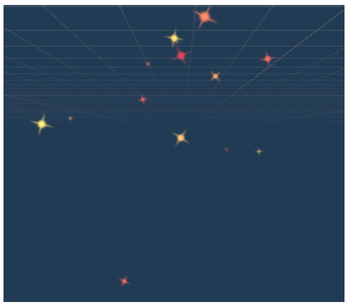
- Explosion for bomb and cannon ball



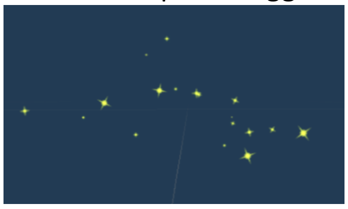
- Cannon smoke



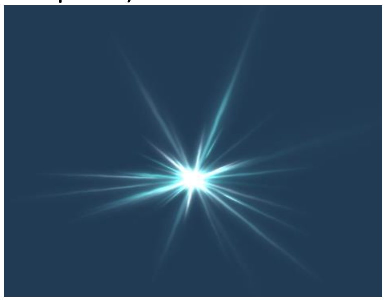
- Bonus block particle



- Event FX (collected an object or checkpoint triggered)



- Spark (for both hive and endpoint)



6.5. Audio

The audio clips chosen during the development of the game that can divided into two categories:

- Music
- SFX

6.5.1. Music

The background music used we can listen into the game are the following:

- **Main menu**: Dexter's Laboratory Main Theme (From "Dexter's Laboratory")
- **Level HUB**: Pokemon Picross - Map
- **Level 1**: Sonic Colours - Area - Sweet Mountain (Theme Song)
- **Level 2**: Kingdom Hearts 1.5 HD ReMIX OST - Deep Jungle
- **Level 3**: Spyro Reignited Trilogy Soundtrack -Stone Hill
- **Victory Screen**: “Snippet” found on Freesound
- **Game over screen**: Blonde Redhead - For the Damaged Coda

6.5.2. SFX

There is a long list of SFX used in the game. Below we will list a brief of what event can trigger an SFX clip.

- Player actions (walking, jumping, sliding, attack, …);
- Enemies’ actions (walking, attack, death, …);
- Obstacles;
- Interactive object;
- Interaction with collectible object;
- Interaction with menu screens.

The audio files used for SFX have been downloaded from web.

7. Implementation

In this section we will see in detail all the previous discussed aspect implemented in the game.

7.1. External Packages

To make easier the development of the game we have used some external packages. The packages installed and used for this game are described below with a description for each of them.

7.1.1. MAST

MAST (Modular Asset Staging Tool) is a custom editor for Unity, primarily intended for staging 3D modular assets, and is very useful for making scenes and levels that are based off of a grid. Instead of having to drag a prefab into the inspector, then manually move, rotate and flip it, **MAST** provides a simple interface that lets you do all the above before ever placing the item in your scene.

7.1.2. Cinemachine

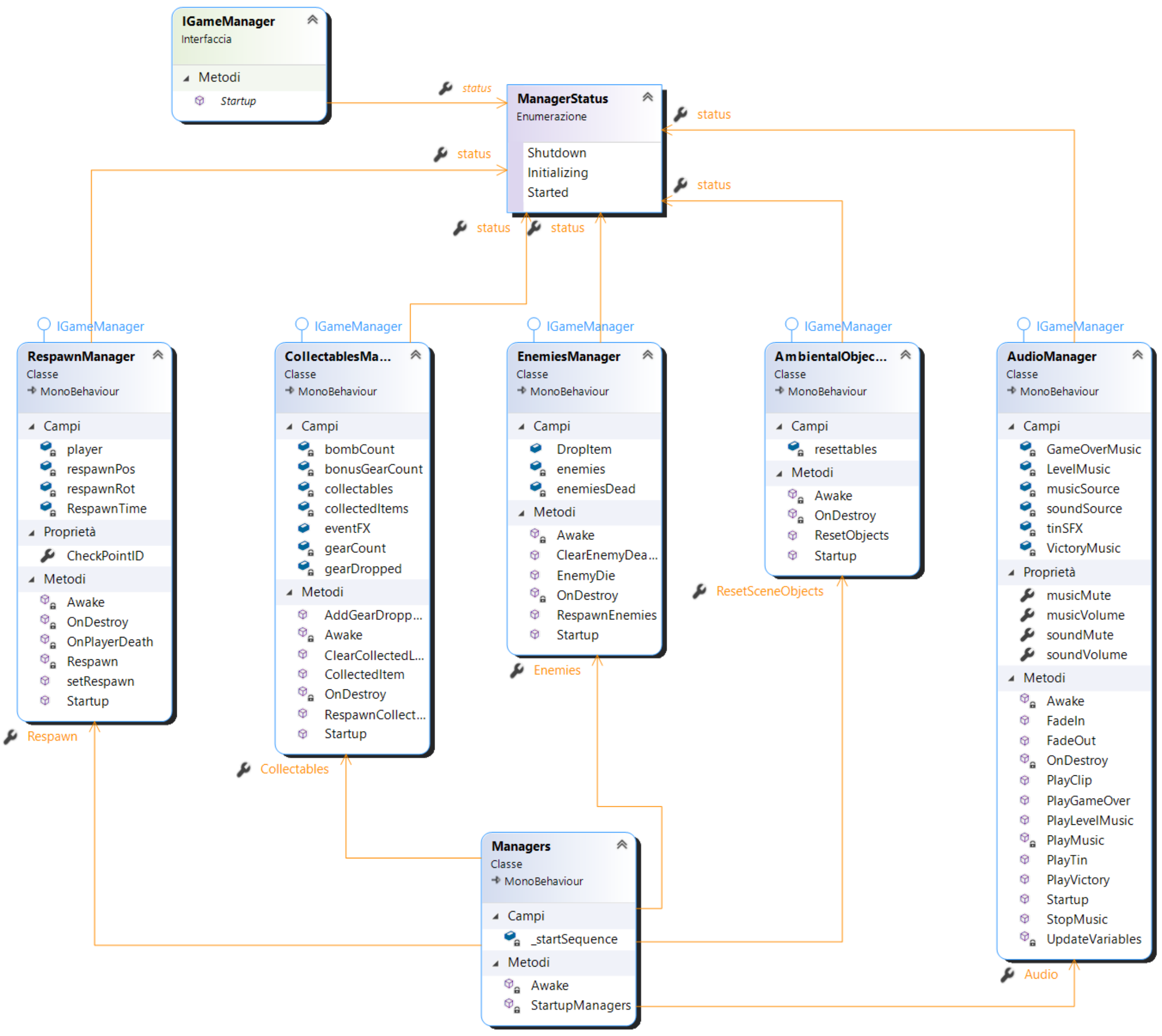
Cinemachine is a modular suite of camera tools for Unity which give AAA game quality controls for every camera in your project. It is an easy to install plugin which lets you add functionality to cameras or make new ones with amazing behaviors. **Cinemachine** has been designed to be the entire unified camera system in a project but can be used alongside the existing cameras as well.

7.1.3. Path Creator

An intuitive and lightweight editor for quickly creating smooth paths in the editor. Objects can be moved quickly along these paths, or they can be used as a guide for spawning objects and generating meshes. Paths can be rotated and scaled and work well in both 2D and 3D space. Paths can also be created entirely from code; just provide an array of waypoints and get a smooth path back.

7.2. Managers

7.2.1. *Managers Class Diagram*



7.2.2. *Managers description*

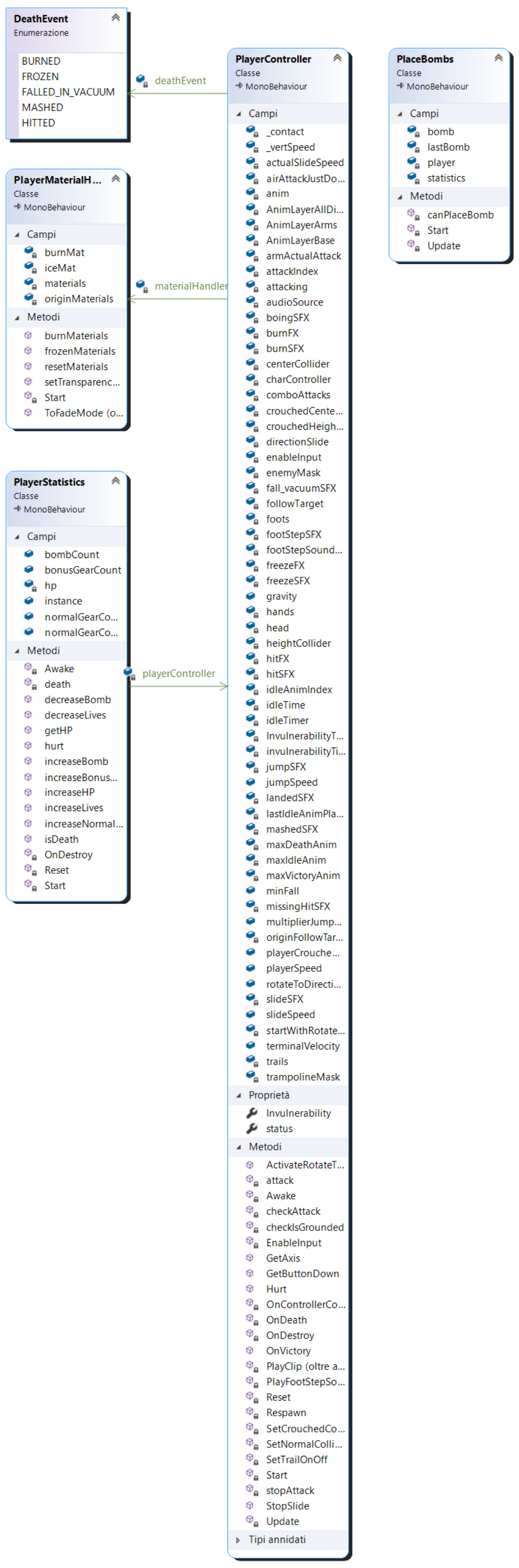
The **Managers** class takes care about startup all managers attached to it.
All managers implement IGameManager interface, the class ‘Managers’ execute startup of all of them and wait until all managers finish to initialize.
So, the managers initialized are:

- **Respawn Manager**: it takes care about where player must respawn after he die.
- **Collectables Manager**: it takes care about resetting gears, gear bonus and battery collectables after player die. In particular, it reset the collectables (and counters too) at the last checkpoint status. If no checkpoint is activated, it will reset all collectables.

- **Enemies Manager**: it takes care about resetting enemies and spawner after player die. In particular, it reset the enemies at the last checkpoint status. If no checkpoint is activated, it will reset all.
- **Ambiental Object Manager**: it takes care about resetting Ambiental objects such as breakable walls and boulders after player die.
- **Audio Manager**: it manages listener and music volume, it also permits to play some global sound effect such as 'Tin', played when player get collectables or reaches checkpoint.

7.3. Player

7.3.1. Player Class Diagram



7.3.2. Player description

On player are attached the following scripts:

- **PlayerController**: this script handles the user input to control player actions.
- **PlayerStatistics**: contains all player statistics like lives, HP and collectables counter, it has a singleton to access rapidly to the player gameobject.
- **PlayerMaterialHandler**: this script handles player material to make some visual effect.
- **PlaceBombs**: handles the action to place bombs.

Player Logic

The player can assume different status:

- **IDLE**: player can stay still or in movement, and can perform jump, crouch, attack, or slide actions. After 5 seconds of inactivity by user, player choose an idle animation to perform.
- **CROUCH**: player can stay still or in movement but in a crouched mode, performing a jump or an attack, or re-pressing 'Crouch' button, he will turn on IDLE status.
- **FALLING**: player is falling or just start a jump, here he can perform an air attack and, when he lands, he passes to IDLE status.
  When he lands, he checks the name layer of the surface touched.
  - If it has "Bouncy" layer, he re-jump higher;
  - If it has "Enemy" layer, he re-jump and hit the enemy under him.
- **SLIDE**: in this status the velocity is handled by a Mathf.Lerp to make a rapid movement, from this status he can jump, or he can wait the animation end to pass to IDLE status.
- **DEATH**: there are different type of death, defined in DeathEvent enumerator:
  - **BURNED**: when player is hitted by Fire Dragon spit
  - **FROZEN**: when player is hitted by Ice Dragon spit
  - **MASHED**: when player is mashed by Falling Block or Boulder
  - **HITTED**: when player is hitted by any kind of attack.
  - **FALLED_IN_VACUUM**: when player fall in vacuum.

  In BURNED and FROZEN, the player must be still to simulate that he is remaining locked, in MASHED and HITTED is performed only an animation.
  Instead, FALLED_IN_VACUUM is the only death type, which is completed handled in the status because, to simulate the animation, to player is constantly applied gravity to continue to fall in vacuum while the following target of the camera remain still there.

When player perform an attack, in base of the status, he activates a specific animation and the relative attack trigger. For example, when "Punch left" animation is performed, the hand left attack trigger will be activated until the animation finish.

When enemy hit player, he calls the PlayerStatistics.Hurt(DeathEvent, bool fatal) method, here there are various checks and if player effectively receive an attack, he calls playerController's Hurt method to animate player in base of type hit received.

After player receive a hit, he passes on an invulnerability mode for 3 seconds, in this time, over the animation or visual effect of hit, is called PlayerMaterialHandler to apply the invulnerability animation, so the player material switches smoothly between visible and invisible.

When player performs some quick animation like jump, slide or attack, the trials renderer is activated to make an effect of cleave air.

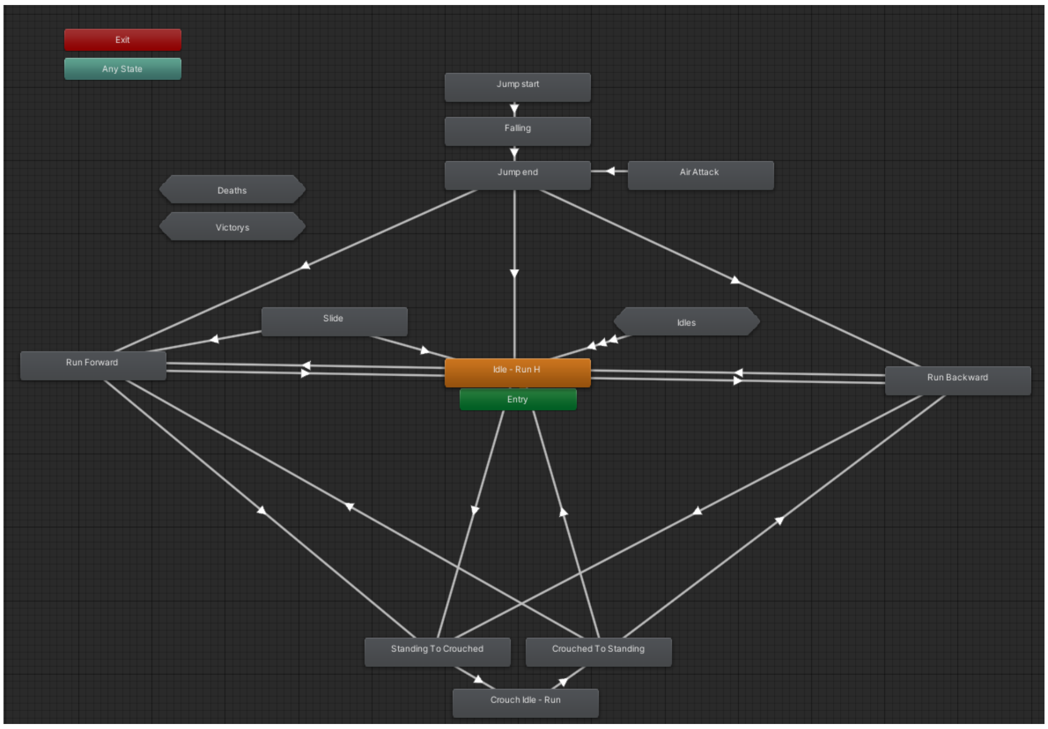
There are some areas in which camera is handled differently. In those cases, player is rotated towards the movement direction and not towards camera direction.

When player reaches the endpoint, he will choose randomly one of the three victory animations.

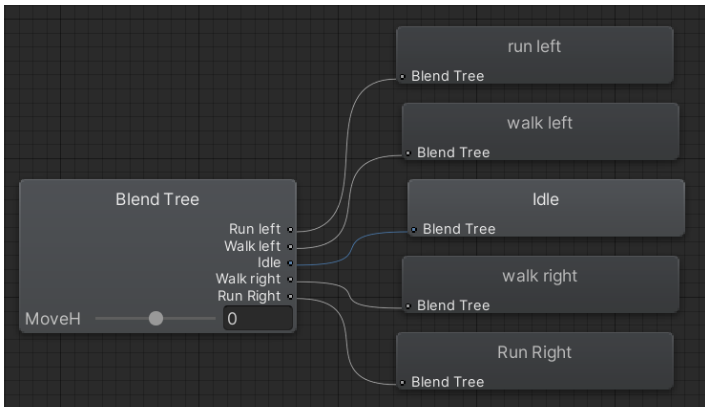
Another interesting method is the PlayFootStepSound, in which is checked if a foot position.y is under a certain height and then play footstep sound.

The player animator controller is composed by 3 layers:

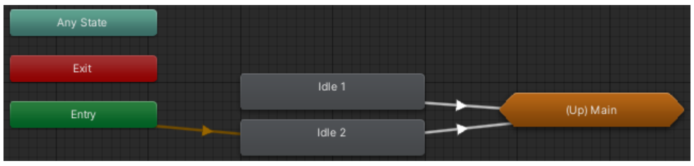
- **Layer 0**: contains the base animations of player.



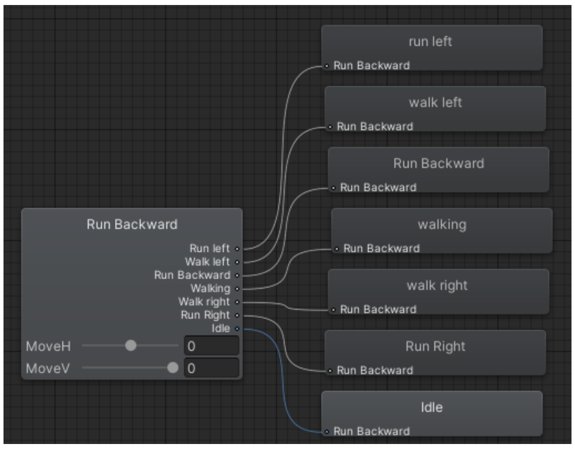
Layer 0



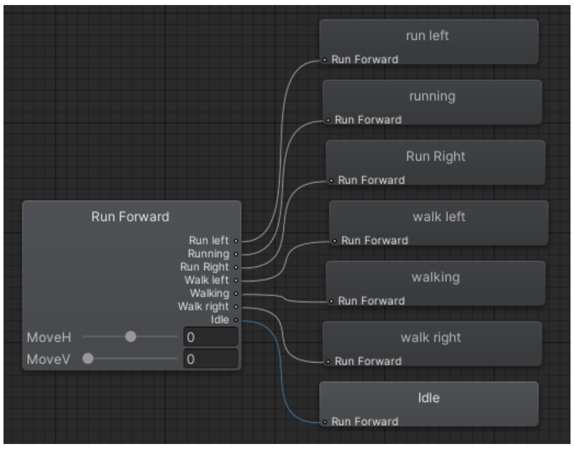
Layer 0: Idle – Run H



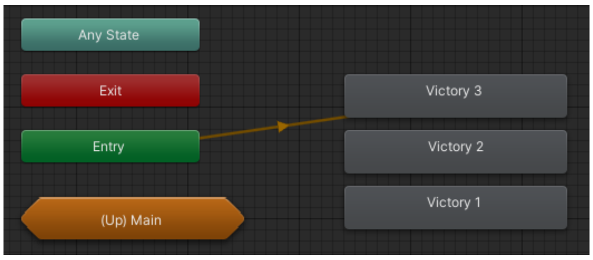
Layer 0: Idles



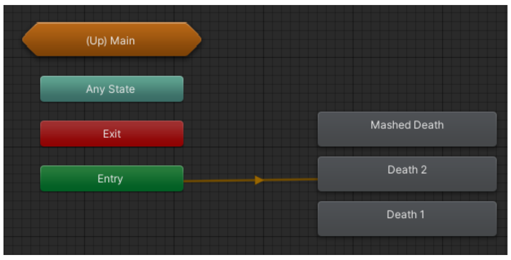
Layer 0: Run Backward



Layer 0: Run Forward



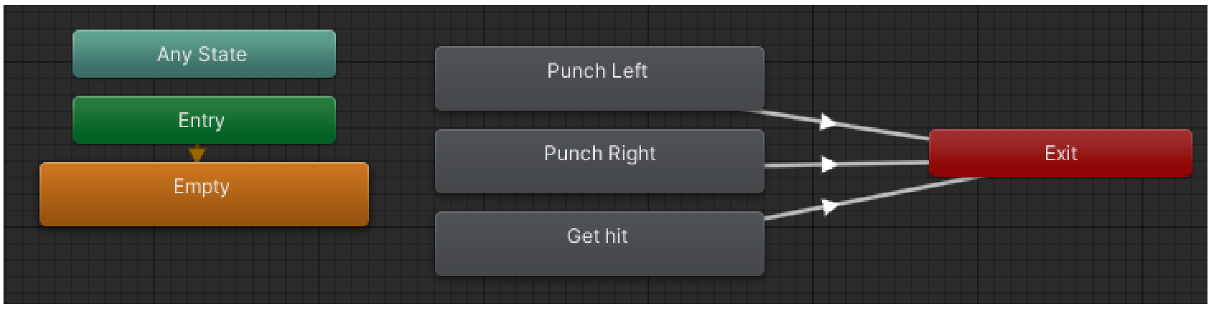
Layer 0: Victorys



Layer 0: Deaths

- **Layer 1**: is an override of layer 0, it is almost the same, change only that, when player rotate towards movement direction, not use more ‘Run left/right’ but use ever ‘Run forward animation’ for all directions. When player has to rotate towards movement direction, it set layer 1’s weight to 1, otherwise 0.

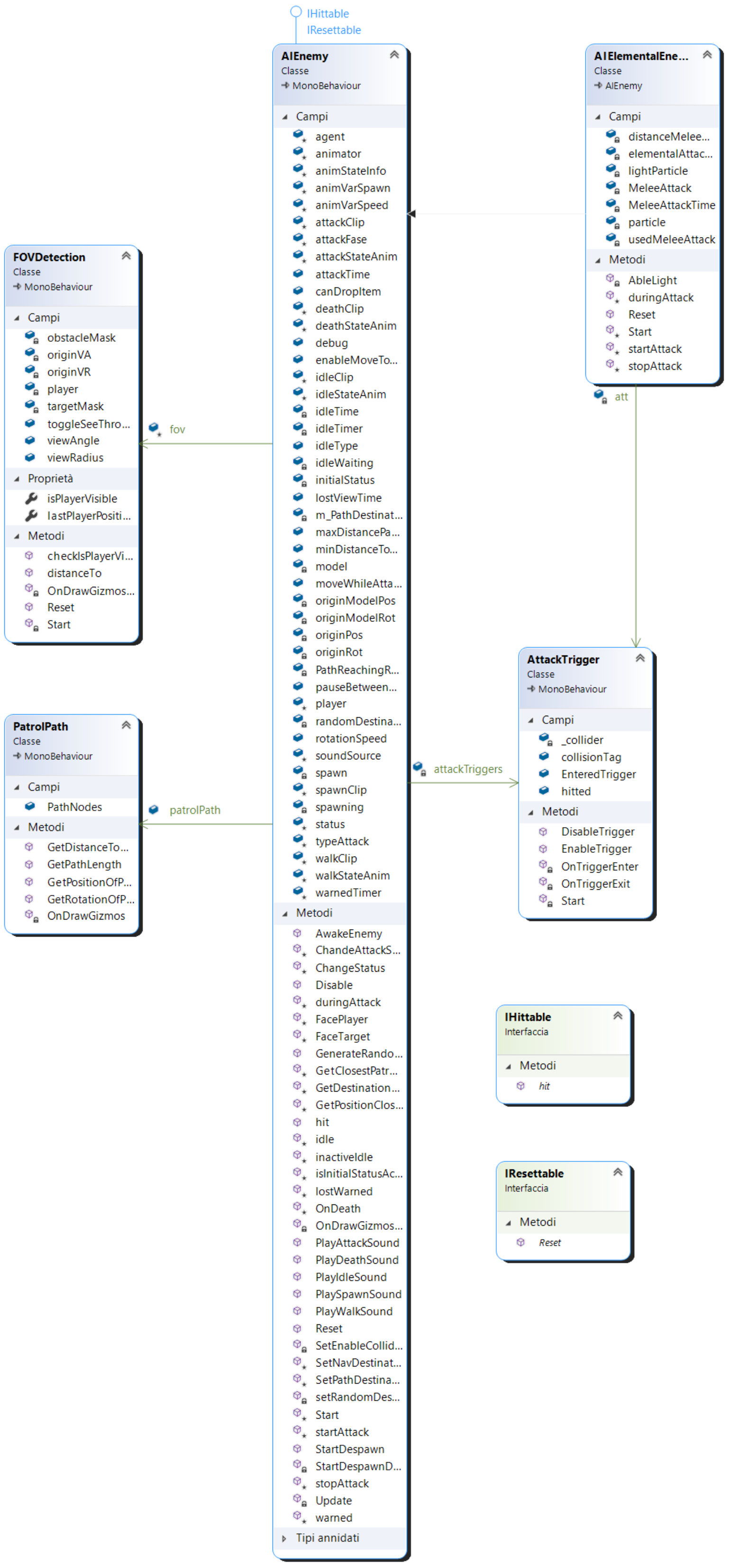
- **Layer 2**: contains animations that are mixed in ‘additive’ mode with the previous layers. It contains normal attacks (not air attack) and get hit animations.



Layer 2

7.4. Enemies

7.4.1. Enemies Class Diagram



7.4.2. Enemies Description

The AI of the enemies is handled by AIEnemy script and is extended for the elemental enemies by AIElementalEnemies. The enemy behavior is structured in different status:

- **INACTIVE**: Some enemies can spawn in INACTIVE status in which they have to wait to be triggered by player to pass in IDLE status.
- **IDLE**: here, they can either follow a ***patrol path***, ***walk random in an area***, or stay ***still*** in their spawn point, waiting for the player enter in their FOV and, when this occurs, enemy pass in WARNED status.
- **WARNED:** in this status, the enemy's aim is to attack player.
  This status is divided in other 3 status:
  - **AttackFase.NO**: in this phase, if ***enableMoveToPlayer*** is true, he can reach the player, otherwise he only rotates towards him. Moreover, the enemy cannot go far away from his spawn point or patrol path.
    If player is so close, enemy pass to ATTACK phase.
  - **AttackFase.ATTACK**: enemy is attacking player, if ***moveWhileAttack*** is true he can also move during this phase. When begin, he activates attack trigger and begin animation. During this phase if attack trigger touch player he hurt him. After time attack is finished, he disables attack trigger and he switches in PAUSE phase.
  - **AttackFase.PAUSE**: enemy wait a pause time and then, if the player is still close, he performs another attack (ATTACK); if the player is still in fov, he remains warned (NO), otherwise he switches in LOSTVIEW phase.
  - **AttackFase.LOSTVIEW**: here, enemy wait a lostview time and, during that, if player is again in fov, he come back in AttackFase.NO, otherwise, if the time is finish, he come back to IDLE status. In this case, enemy come back to spawnpoint, to closest patrolpath node, or go in a random point.
- **DEAD**: this status is not handled in update method because it doesn't do nothing.

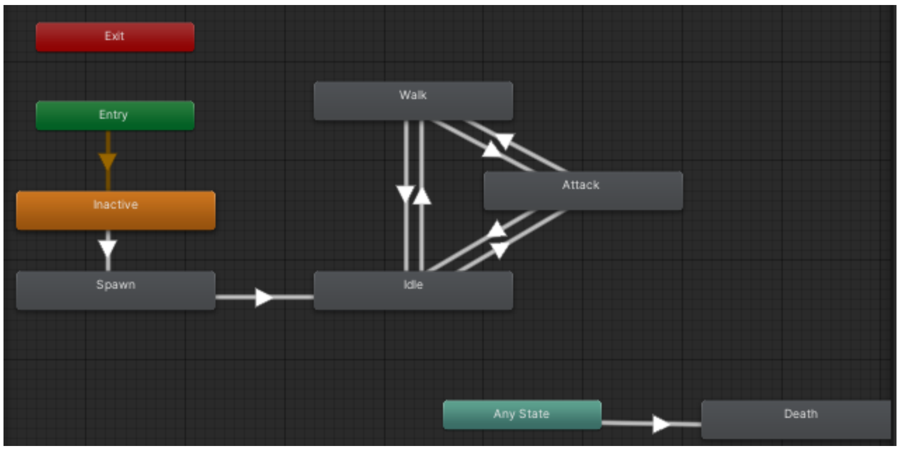
When player hit an enemy, he begins death animation and, if he can, he can drop a gear. When death animation end, after few seconds will start the Despawn animation which is the same for all enemies. After this animation, the gameobject will be disactivated until EnemiesManager reset it.

The FOV is handled by ***FOVDetection*** script which remembers the last visible position of the player.

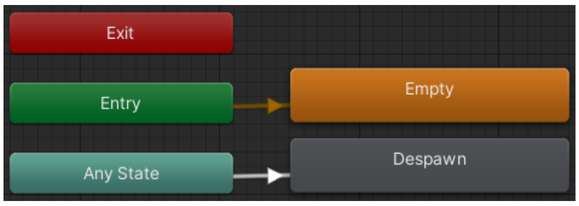
Elemental Enemies are enemies who overrides AIEnemy's virtual methods to use particles which, through ***ElementalSpit*** script, can hit player. In particular:

- Dragons use particle trigger and ElementalSpit script to hit player, otherwise, if player is so close, they perform a melee attack. They can do this because they have the variable ***MeleeAttack*** to true.
- Bat uses particles that are not triggered by player, so, in this case, he uses an attack trigger to detect player. He doesn't have a melee attack.

Every enemy have his animation controller, even if they are a lot similar to each other. The animation controller shown below belongs to Skeleton.



Skeleton: Layer 0



Skeleton: Layer 1

Other enemies have, more or less, the same structure: some does not have inactive and spawn animations, others use idle animation to walk too, as winged enemies, that, like bat, have not a melee attack.
The layer 1 above is the same for all enemies. It combines death animation with despawn ones.

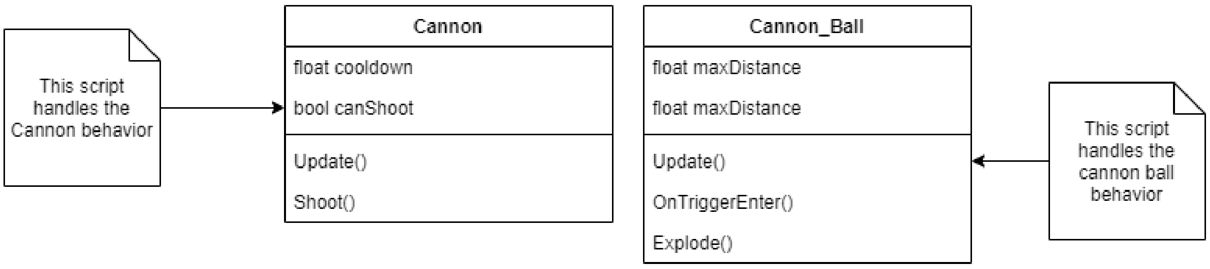
The enemy sound effects are played through animation events, forwarded in AIEnemy script by the ***ReceiveEventAnimation*** script.

7.5. Obstacles

In this section are described all obstacles. The script for the obstacles can both handle the behavior and the custom animation or handle the behavior and the animation created with animator. More information can be found inside of each script. Only main scripts are listed below.

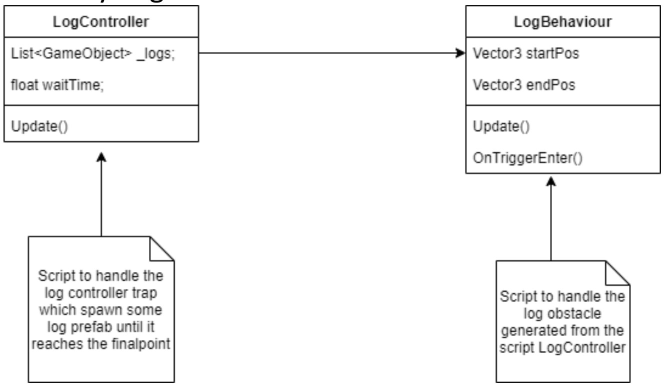
7.5.1. Cannon

The cannon behavior is handled by Cannon.cs. The behavior and animation of the cannon ball shooted is handled by Cannon_Ball.cs



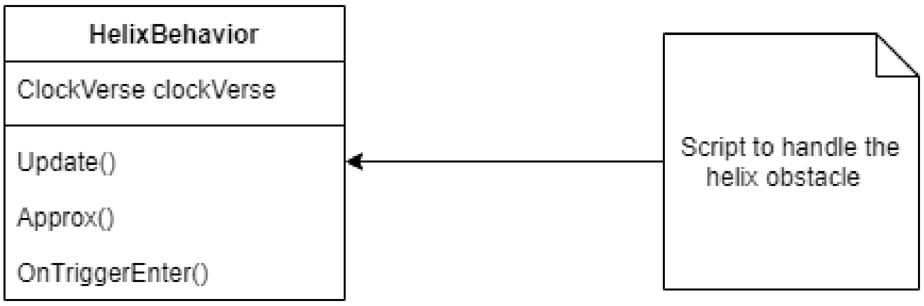
7.5.2. *Log waves*

The log waves behavior is handled by LogController.cs. The behavior and the animation of instantiated log is handled by LogBehaviour.cs.


LogController
List<GameObject> _logs;
float waitTime;
Update()
LogBehaviour
Vector3 startPos
Vector3 endPos
Update()
OnTriggerEnter()
Script to handle the log controller trap which spawn some log prefab until it reaches the finalpoint
Script to handle the log obstacle generated from the script LogController

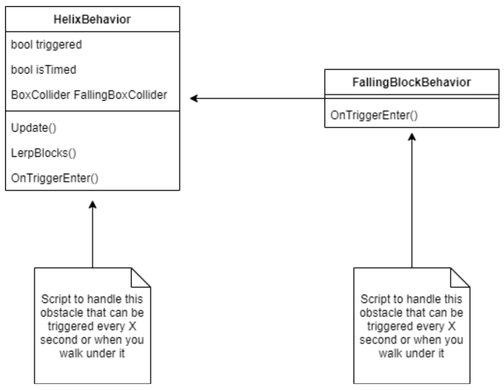
7.5.3. *Rotating logs*

The behavior and the animation of rotating log obstacle is handled by HelixBehaviour.cs



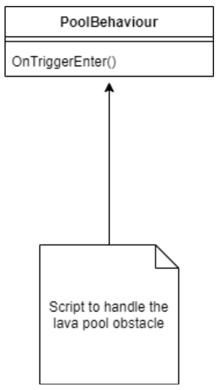
7.5.4. *Falling Blocks*

The behavior and the animation of Falling Blocks is handled by FallingBlocks.cs and FallingBlockBehaviour.cs


HelixBehavior
bool triggered
bool isTimed
BoxCollider FallingBoxCollider
Update()
LerpBlocks()
OnTriggerEnter()
FallingBlockBehavior
OnTriggerEnter()
Script to handle this obstacle that can be triggered every X second or when you walk under it
Script to handle this obstacle that can be triggered every X second or when you walk under it

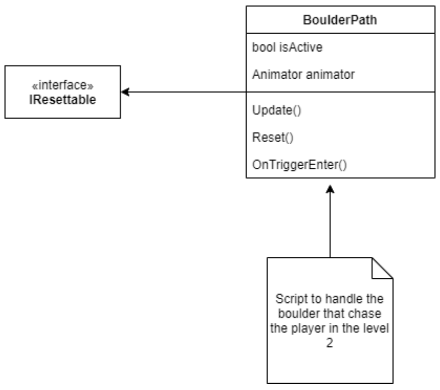
7.5.5. *Lava pools*

The behavior of Lava pools is handled by PoolBehaviour.cs

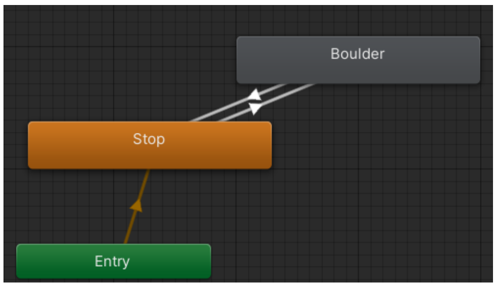


7.5.6. *Boulder*

The behavior of Boulder is handled by BoulderPath.cs



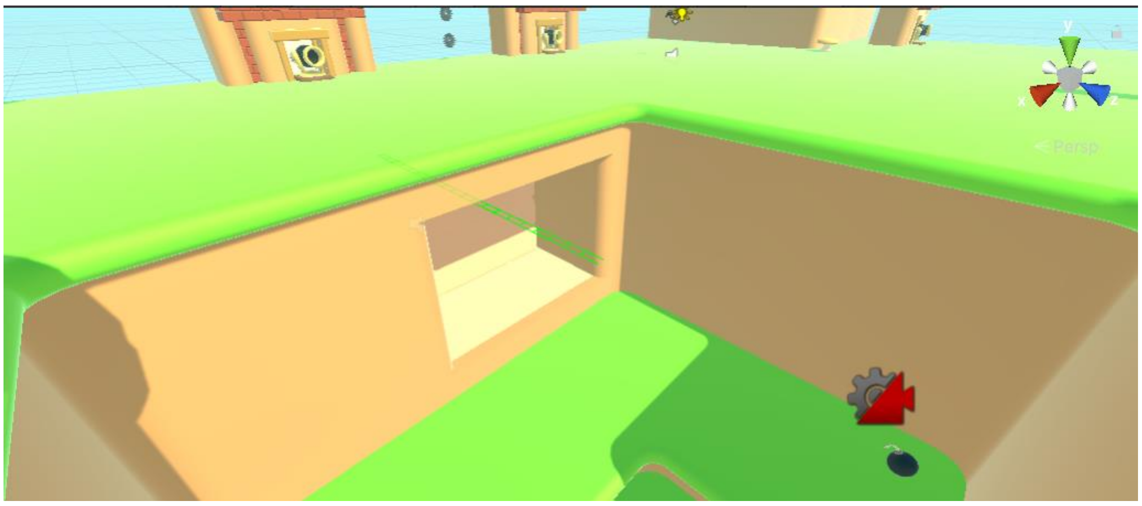
The animation of Boulder is handled with an animator. The following describe the state machine.



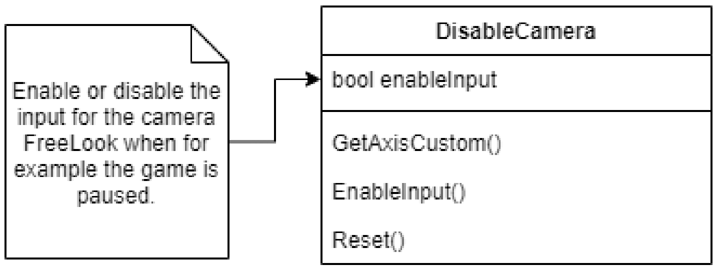
7.6. Camera

The camera is handled by ***Cinemachine***. In particular, the main virtual camera used is a ***Cinemachine's FreeLook*** component, which allows to have a good overview of the player and the surrounding area.
For some areas or levels, there is need to change camera to better view player without any kind of glitch.

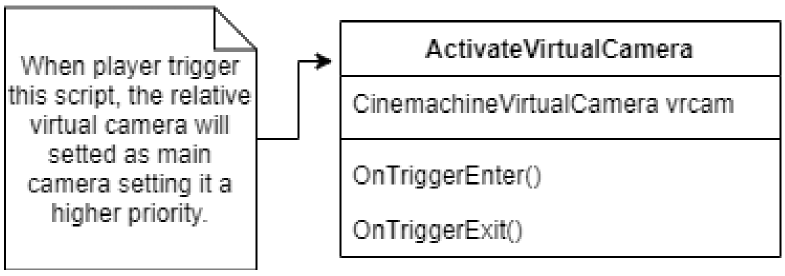
In level 1 and 2 is used a **DollyTrack**, which is a track where the virtual camera can flow to frame better the player at a certain distance. Lastly, in level 3 there are two simple virtual cameras.



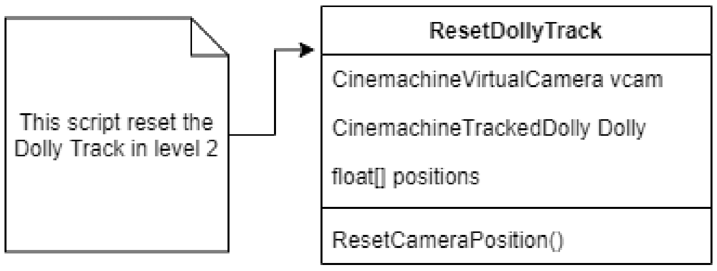
Relative to the camera managing, there are some scripts:
DisableCamera.cs


Enable or disable the input for the camera FreeLook when for example the game is paused.
DisableCamera
bool enableInput
GetAxisCustom()
EnableInput()
Reset()

ActivateVirtualCamera.cs

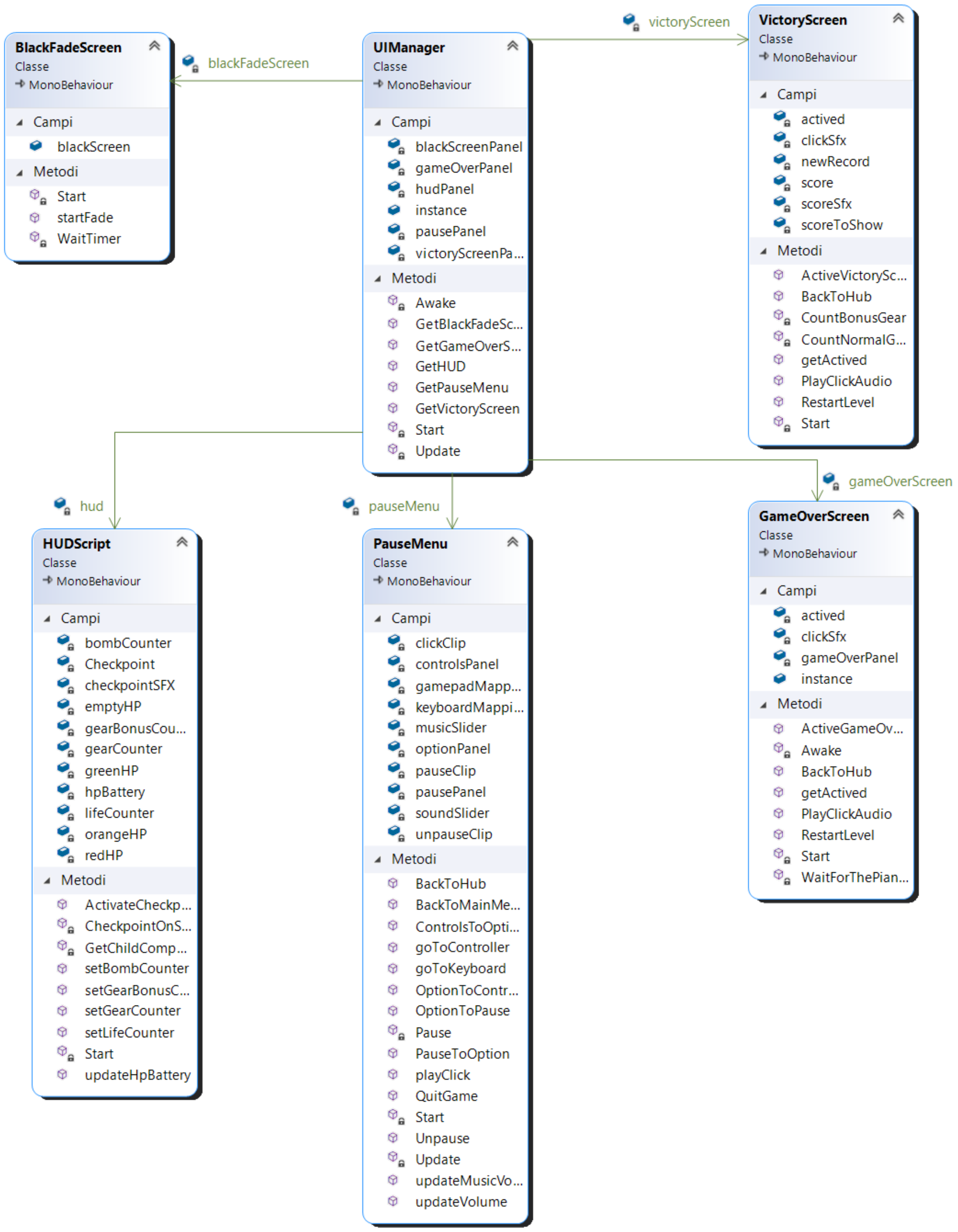

When player trigger this script, the relative virtual camera will setted as main camera setting it a higher priority.
ActivateVirtualCamera
CinemachineVirtualCamera vrcam
OnTriggerEnter()
OnTriggerExit()

ResetDollyTrack.cs


This script reset the Dolly Track in level 2
ResetDollyTrack
CinemachineVirtualCamera vcam
CinemachineTrackedDolly Dolly
float[] positions
ResetCameraPosition()

7.7. UI

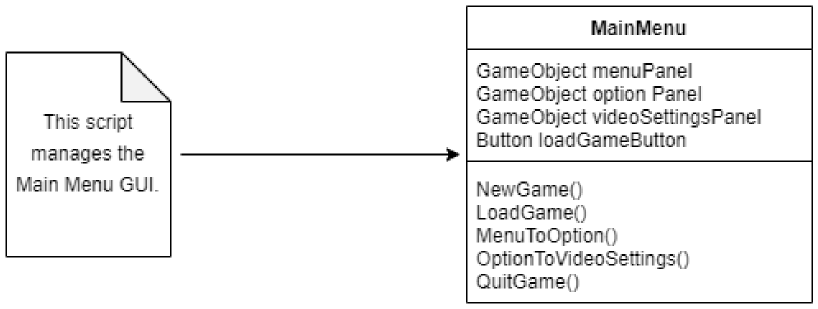
7.7.1. UI Diagram



7.7.2. Brief description of the classes

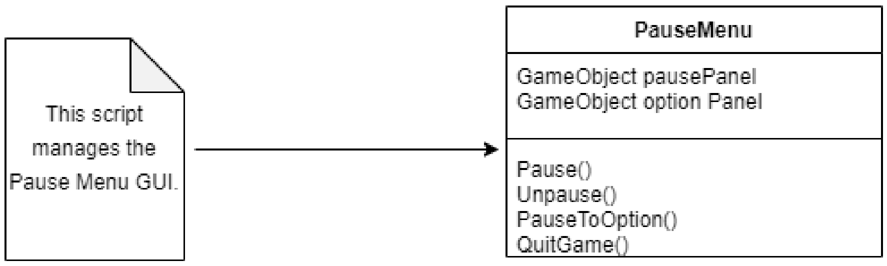
Main menu

The main menu is handled by MainMenu.cs



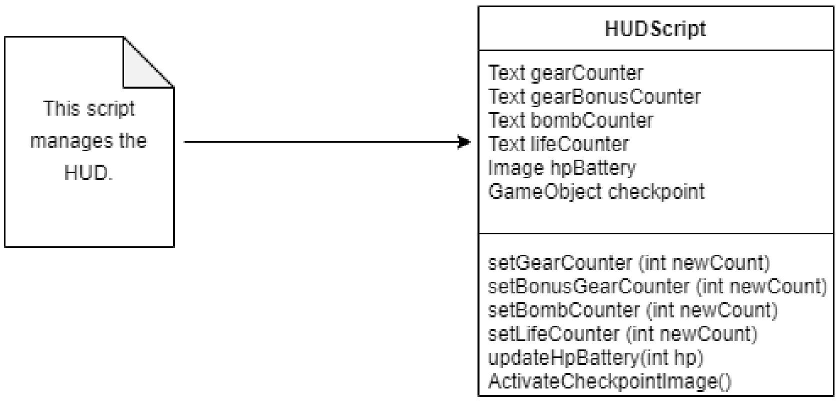
Pause Screen

The pause screen is handled by PauseMenu.cs



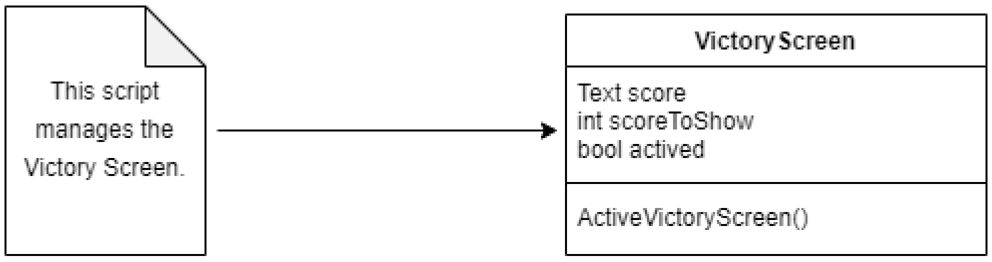
HUD

The HUD is handled by HUDScript.cs

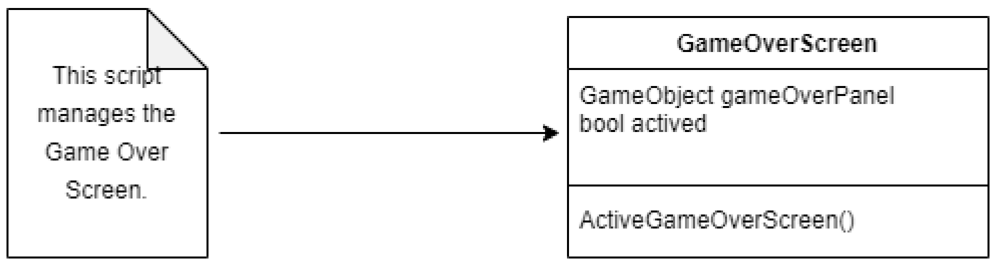

This script manages the HUD.
HUDScript
Text gearCounter
Text gearBonusCounter
Text bombCounter
Text lifeCounter
Image hpBattery
GameObject checkpoint
setGearCounter (int newCount)
setBonusGearCounter (int newCount)
setBombCounter (int newCount)
setLifeCounter (int newCount)
updateHpBattery(int hp)
ActivateCheckpointImage()

Victory Screen

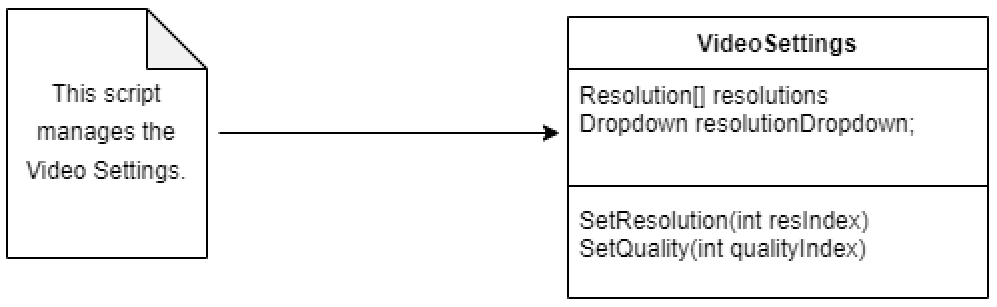
Victory Screen is handled by VictoryScreen.cs



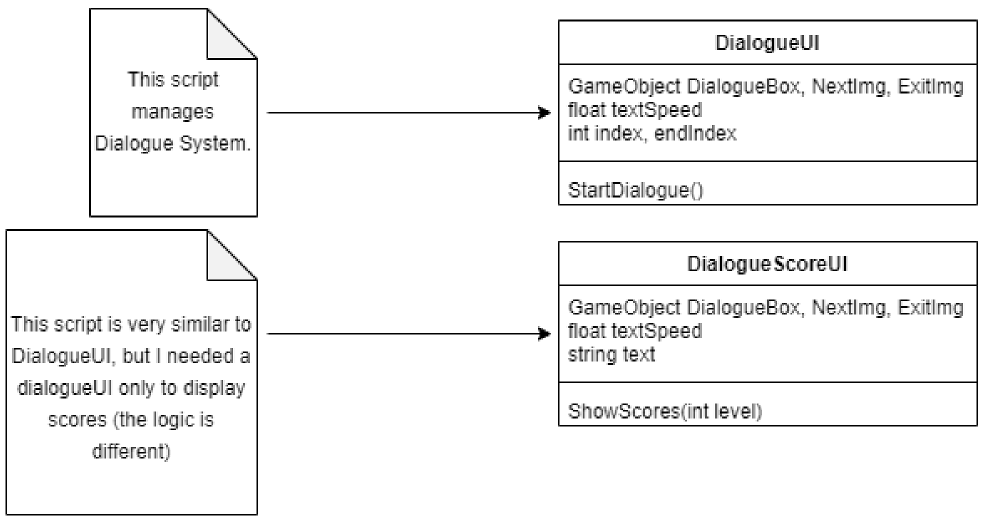
Game Over Screen
Game Over screen is handled by GameOverScreen.cs



Video Settings Screen
Video Settings section is handheld by VideoSettings.cs


This script manages the Video Settings.
VideoSettings
Resolution[] resolutions
Dropdown resolutionDropdown;
SetResolution(int resIndex)
SetQuality(int qualityIndex)

Dialogue Screen
The screens regarding the dialogue system are handheld by two different scripts: DialogueUI.cs and DialogueScoreUI.cs

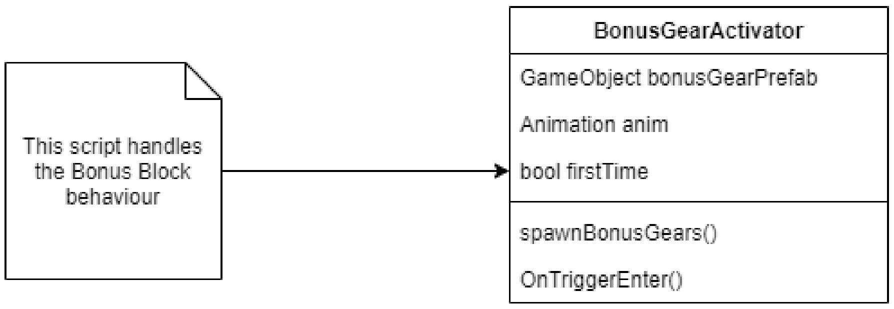

This script manages Dialogue System.
DialogueUI
GameObject DialogueBox, NextImg, ExitImg
float textSpeed
int index, endIndex
StartDialogue()
This script is very similar to DialogueUI, but I needed a dialogueUI only to display scores (the logic is different)
DialogueScoreUI
GameObject DialogueBox, NextImg, ExitImg
float textSpeed
string text
ShowScores(int level)

7.8. Others

7.8.1. Interactive objects

Bonus Block

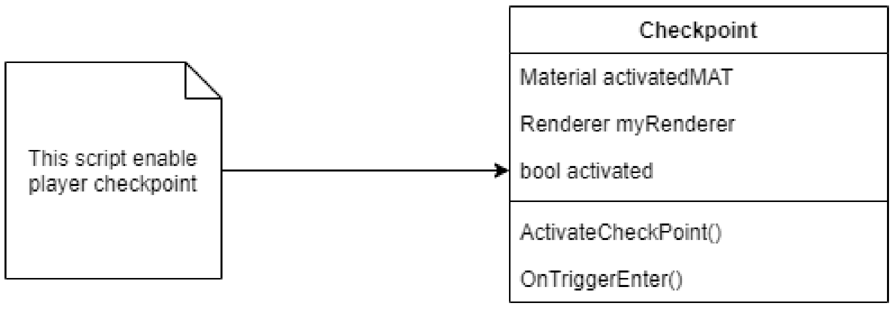
The behavior of the bonus block is handled by BonusGearActivator.cs



The animation of the bonus block has been created with the Animation component in Unity.

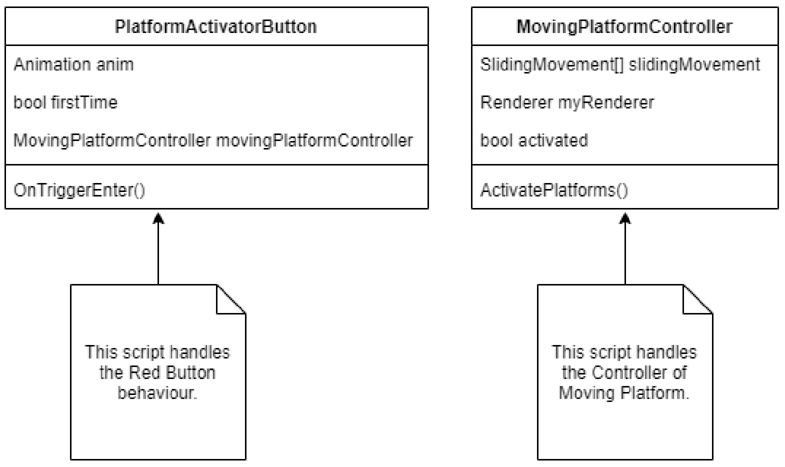
Checkpoint

The behavior of the checkpoint is handled by Checkpoint.cs



Platform Button

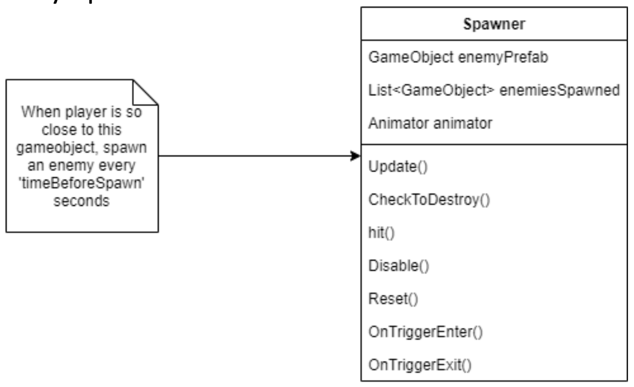
The behavior of the button is handled by both PlatformActivatorButton.cs and MovingPlatformController.cs.


PlatformActivatorButton
Animation anim
bool firstTime
MovingPlatformController movingPlatformController
OnTriggerEnter()
This script handles the Red Button behaviour.
MovingPlatformController
SlidingMovement[] slidingMovement
Renderer myRenderer
bool activated
ActivatePlatforms()
This script handles the Controller of Moving Platform.

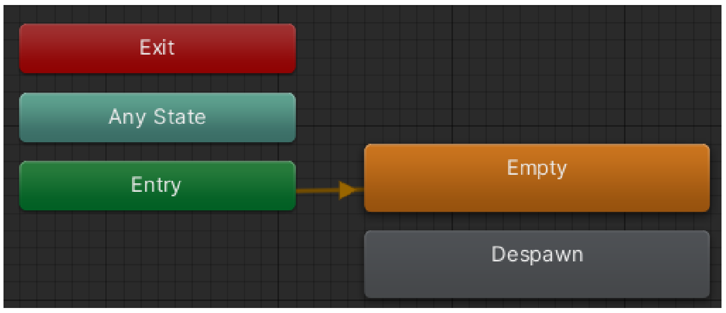
The animation of the button has been created with the Animation component in Unity.

Hive

The behavior is handled by Spawner.cs.

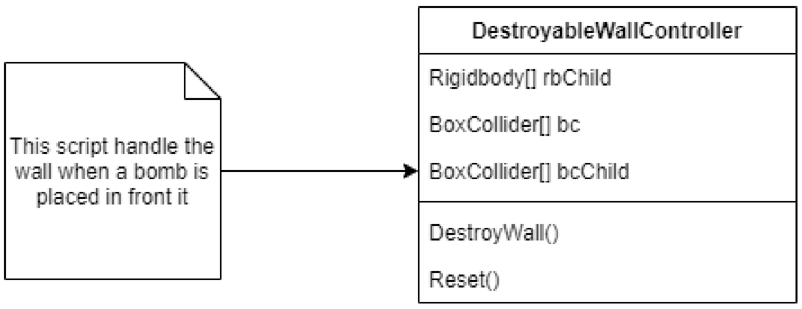


The animation is handled with animator with the sub states as follow.



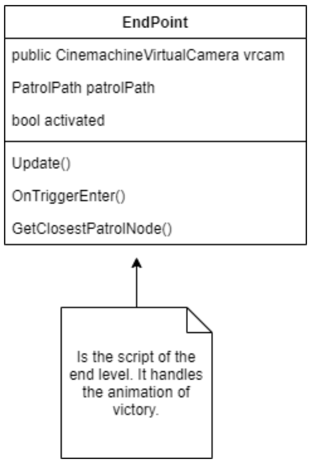
Destroyable Wall

The behavior is handled by DestroyableWallController.cs

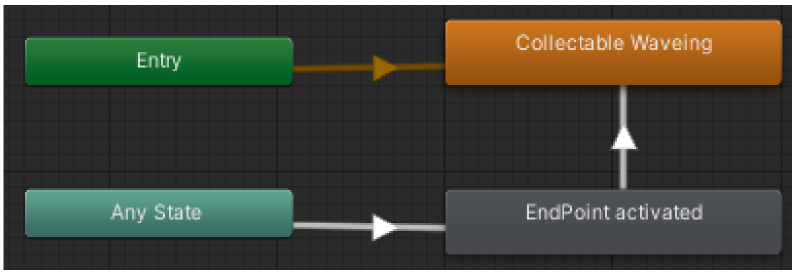

This script handle the wall when a bomb is placed in front it
DestroyableWallController
Rigidbody[] rbChild
BoxCollider[] bc
BoxCollider[] bcChild
DestroyWall()
Reset()

Endpoint

The endpoint behavior is handled by EndPoint.cs.


EndPoint
public CinemachineVirtualCamera vrcam
PatrolPath patrolPath
bool activated
Update()
OnTriggerEnter()
GetClosestPatrolNode()
Is the script of the end level. It handles the animation of victory.

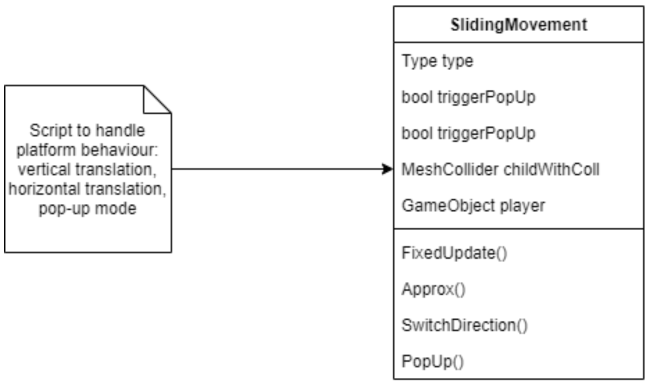
The animation of the electronic device is handled by Animator component in unity with the following state machine:



7.8.2. *Environmental object*

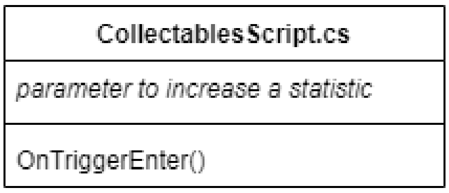
Platform

The behavior and the animation of platform is handled by SlidingMovement.cs


Script to handle platform behaviour: vertical translation, horizontal translation, pop-up mode
SlidingMovement
Type type
bool triggerPopUp
bool triggerPopUp
MeshCollider childWithColl
GameObject player
FixedUpdate()
Approx()
SwitchDirection()
PopUp()

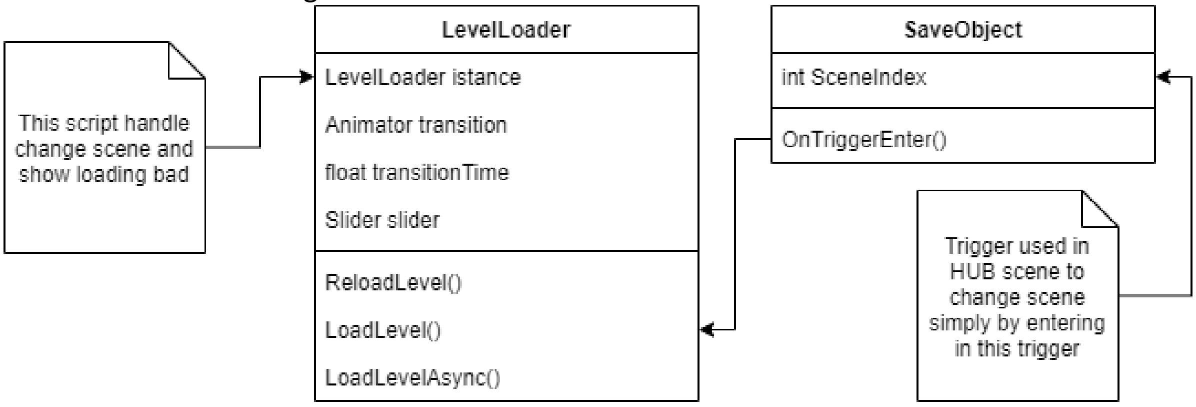
7.8.3. *Collectables*

The collectables gears, gear Bonus, HP Battery, Lives and bombs use the same structures for all the scripts. The following will be the structure used for the objects:



7.8.4. Scene Loader

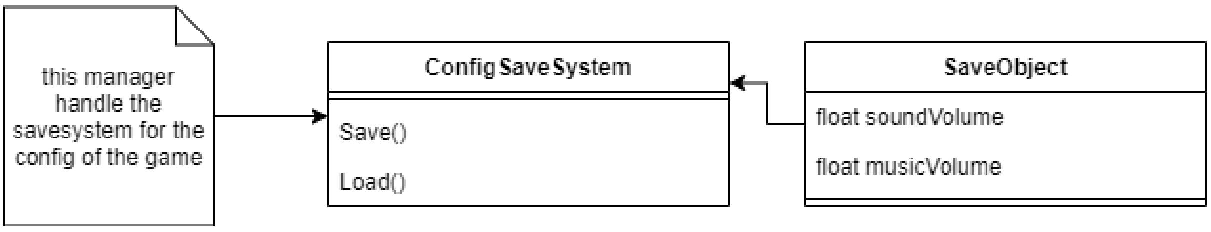
Whenever the ChangeSceneTrigger.cs is triggered, the LevelLoader.cs is called to have a transition with a loading bar between scenes.



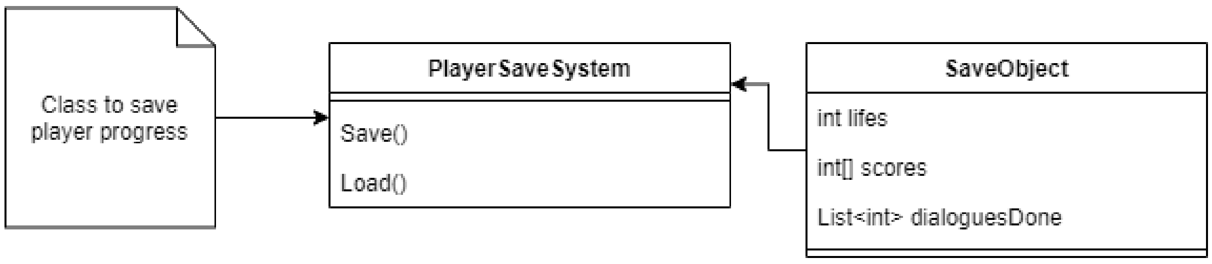
7.8.5. Save system

The save system allows you to save progress and setting made during the game.

Config settings
Whenever the player changes the volume or the fbx sound and then go back to previous screen, the changes are stored on a “config.txt” file in a JSON format. The config settings are handled by ConfigSaveSystem.cs



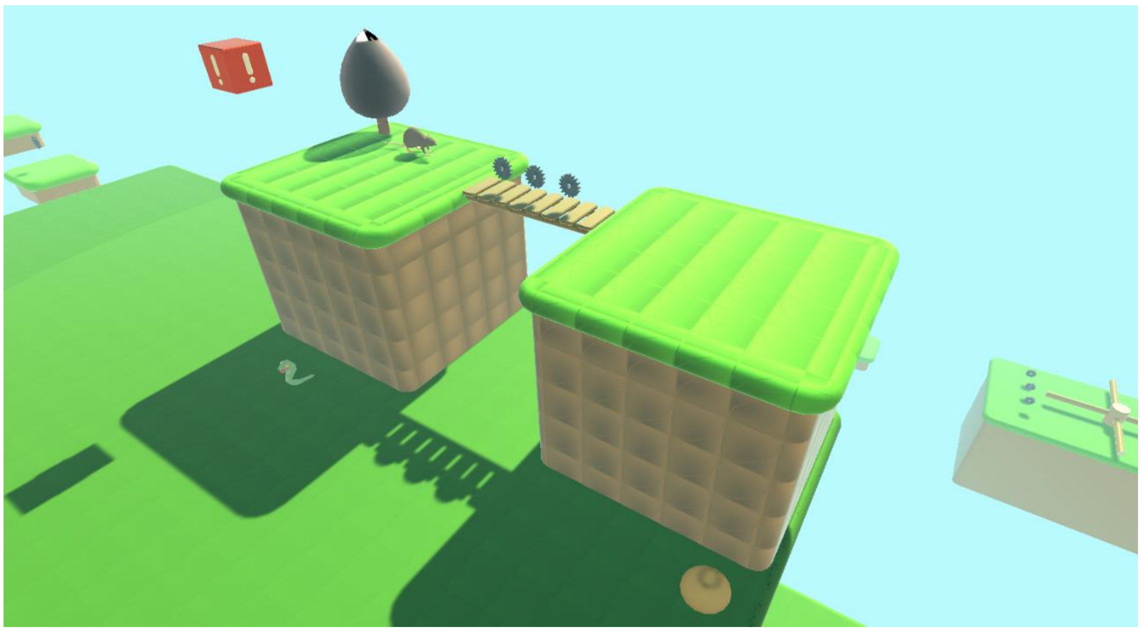
Player settings
The progresses made by the player are also stored in a “player.txt” file in JSON format. The progresses made by the player and store are number of lives collected, score done for each level and dialogues already met to not trigger them again. The behavior is handled by PlayerSaveSystem.cs



7.9. Performance

7.9.1. Lighting

In this game, the lightning is not baked because MAST prefabs are not UV mapped properly and this creates awful artifacts.

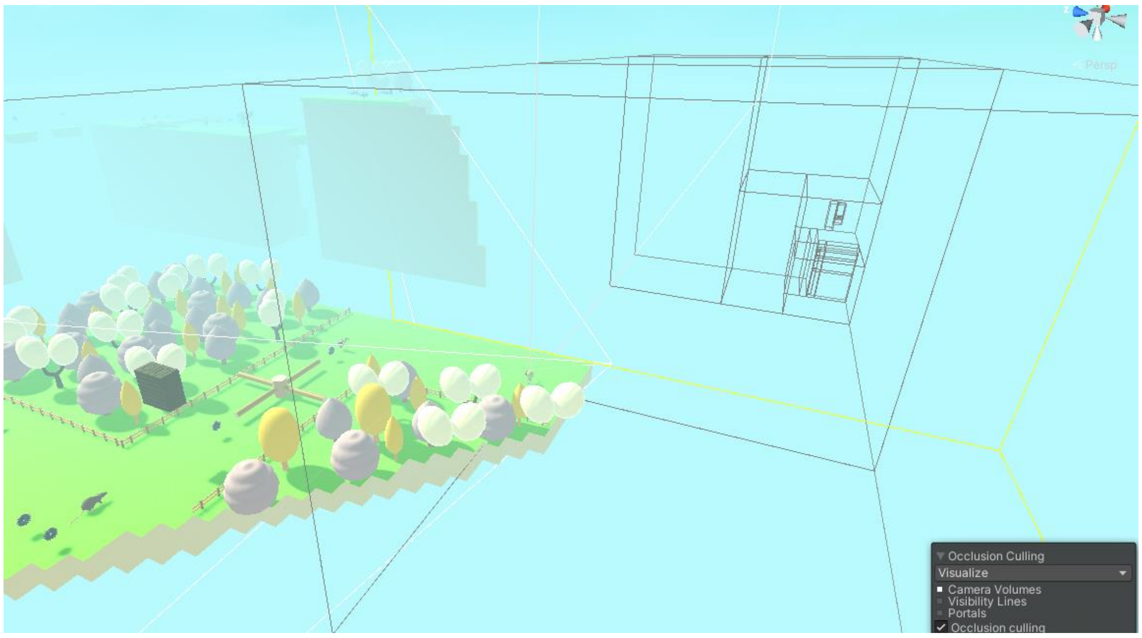


7.9.2. Fog

Fog is used to give a sense of distance to the player. In this way the player can clearly see object near to him but faded those in the distance.

7.9.3. Occlusion Culling

Occlusion Culling is a technique to increase performance that allows to not renderer the objects that are not in camera frustum. This process allowed us to increase a bit the performance, especially in level 3.



8. Future development

This game is presented as a demo for the project of the course of Virtual Reality of Professor Carmelo Macrì.
More improvements can be done on various aspect of the game.

First of all, the performance may be improved because MAST helped us a lot in the creation of the scenes, but it has lowered a lot the performance. In fact, the MAST prefabs are not UV mapped properly and, if the lighting is baked, it creates a lot of awful artifacts.
Moreover, it can be improved the look & feel to be more cartoonish by using **post-processing fx**, by using toony shaders, ambient occlusion, and more.

More mechanics regarding the player interaction with the environment or the diversification of the enemies can be added.
For example: an idea could be to implement a new level, that diversifies the gameplay, like an endless runner game or a stealth-like section, in order to implement stealth mechanics useful to avoid enemies. Also, more enemies could be added to have different enemies to interact with.
Another idea could be to implement frozen levels where the player has to be careful to not slide on ice.

Another improvement can be adding more dialogues with dubbing implemented also as cutscenes.

In this version of the game there are present some bugs that may happen during the gameplay and annoy the user. For example, some bugs are that the menu cannot be closed sometimes and the user is forced to shut down the game.