Department of Computer Engineering

Games and Multimedia – Games Programming

Teachers: Luís Monteiro, Ricardo Antunes and Roberto Ribeiro

Project Description – First, Second and Third Examination Periods – 2nd Project

<u>Important</u>

This assignment is independent from the first one and must be developed using a separated project. That means that if you want to develop and deliver both projects (1 and 2), you must upload 2 separated projects to Moodle.

Objectives

For this assignment it is intended to develop a platformer game using the **Unity** game engine and the **C#** programming language. During the development of this assignment, the best practices of programming should be applied.

Functionalities to Implement

All the necessary assets to develop the requested functionalities were provided by Unity's 2D Game Kit and are available for download in the **Moodle** platform. A project with the base map is available as well. **The work must be developed using the supplied project**.

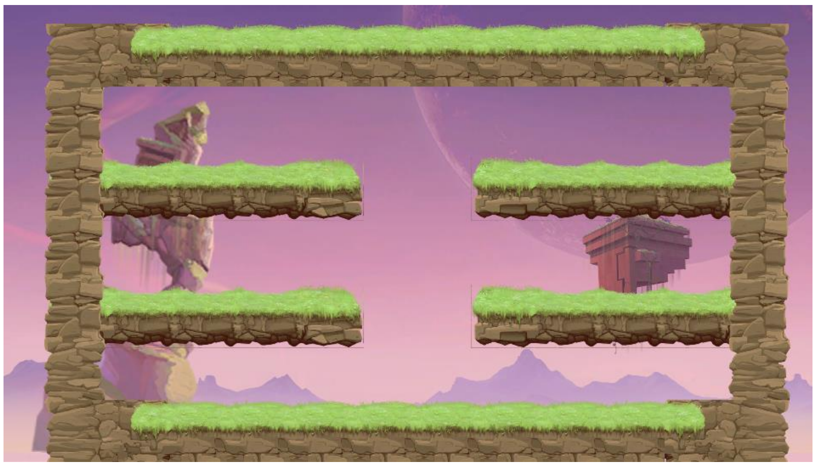
Figure 1. Representation of the playable area

1. Player

The visual representation of the **Player** should be constituted by all the animation states required by the project, available in the Gunner folder, which are: idle, walk, jump, attack, and death. For the jump animation of the **Player**, each group may use the most appropriate sprites from the available resources.

All the configurable values of the **Player** should be available in the **Inspector** (e.g., jump strength, walk speed, time intervals).

Relative to the locomotion, the **Player** should walk and face the direction of the horizontal input. When no input is given, the character should stay idle.

The jump should allow to get to higher platforms.

Relative to the attack, when the input is detected, the **Player** should shoot circular projectiles forward, but only if the character is on the ground and the weapon gauge is full. When the attack is executed, the gauge value is depleted.

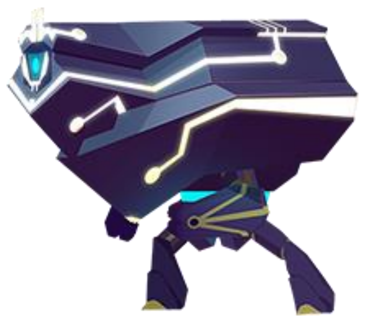

Figure 2. Representation of the Player during the attack

Projectiles can only collide with platforms and walls. When that happens, the projectile should be dismissed.

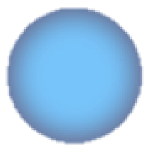

Figure 3. Representation of the projectile with a bluish tint color.

Relative to the life cycle, the **Player** should have health points (HP). When damaged, the HP value should be updated and if it reaches the 0 value, the **Player** should die, a “GAME OVER” message should be displayed for 3 seconds, and the game should return to the initial menu scene.

2. Acid Bubbles

From below the level map, there should be a spawning area that ejects floating bubbles vertically. The initial horizontal position of each bubble is defined randomly, according to the extent of the map.

When a bubble is ejected, it starts floating up until it is dismissed, for no longer making part of the playable and visible areas or explode when colliding with the **Player**. If this happens, the HP of the **Player** should decrease.

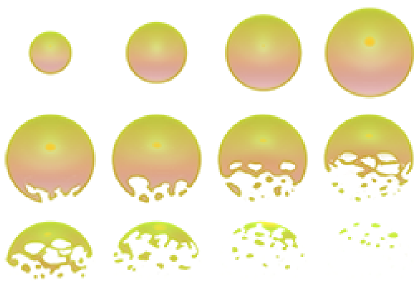
Figure 4. Representation of the acid bubbles

3. Object Pooling

Adapt the project to include a generic and scalable Object Pooling (OP) system, responsible for managing the projectiles of the **Player** and the acid bubbles. Before the level starts, the OP should instantiate an initial number of each reusable element and increase the available amount if required while in runtime.

4. UI

To represent the weapon gauge (blue) and the HP of the **Player** (green), add two UI bars to the upper left corner of the screen. Each group may use the most appropriate sprites to represent the bars from the available resources.

The game should begin with a start menu with buttons to start the game and quit. When the **start** button is pressed, the first playable level is loaded. The **quit** button should terminate the game execution when pressed.

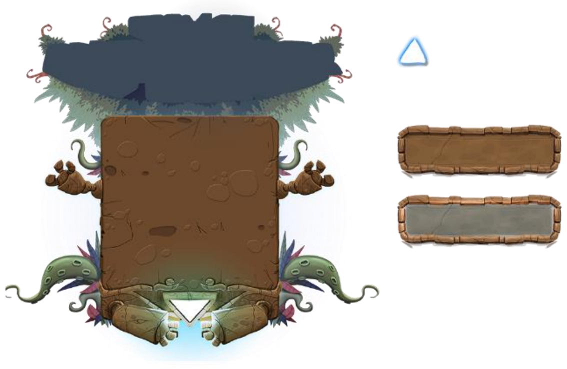
Figure 5. Representation of the initial menu assets

5. Portals

Scattered around the levels, there are portals that can teleport the **Player**. Each portal can be connected to an arbitrary number of different portals (i.e., 1 or more), creating a non-linear teleporting network.

In the beginning of the game, the game must guarantee that each portal discards itself from the available destinations, and that the portal remains inactive if it is not connected to any other portals. If a portal is connected to more than 1 destination, the **Player** destination is selected randomly from the available ones.

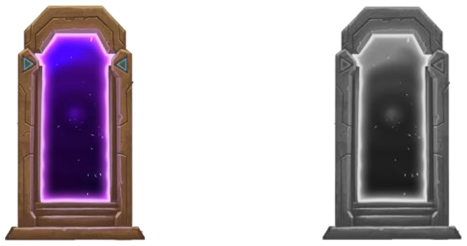

Figure 6. Representation of active (left) and inactive (right) portals

6. Scene Management

This project should be constituted by three scenes, being the first one the start menu. Besides, there should be two playable levels, distinguished by a different object arrangement (i.e., portals, contact areas, and box). When the second playable level is loaded, do not destroy the **Player** and other game elements that you consider helpful to maintain from the previous level.

The condition to transit to the next level or ending the game is using all the available portals, discarding the ones without defined destination (i.e., inactive). When the Player completes the last level, a “Congratulations” message should be displayed for 5 seconds, and the game should return to the initial menu.

7. Contact Areas

This feature must be implemented using inheritance.

Scattered around the levels, there are also **contact areas** that can affect the **Player**. At least one of each contact areas must be present in the level.

Every time the **Player** is affected, while inside the contact areas, the character should change its tint color with the color that represents each of the effects. When the player leaves the area, it should return to its default tint color.

The functionalities from the contact areas are:

Increase HP – Increases the **Player**’s HP over time.

Weapon Charge – Increases the **Player**’s weapon gauge over time.

Damage – Damages the **Player** over time.

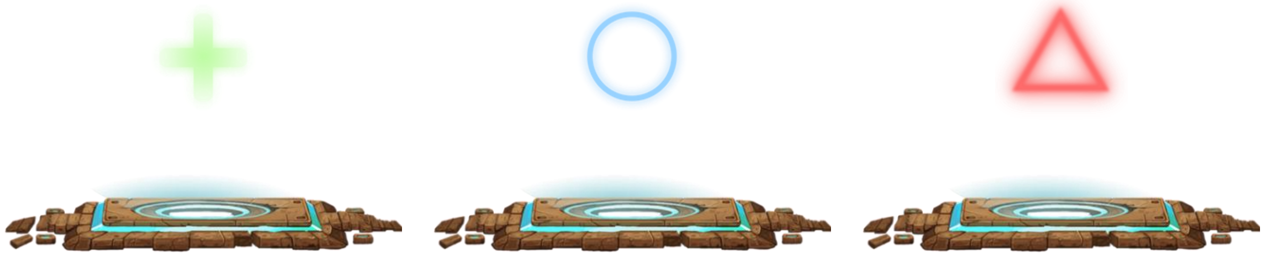

Figure 7. Representation of the contact areas that increase HP (left), weapon charge (center) and inflict damage (right)

8. Box

Inside each playable level there should exist one box that is activated when the player touches it. When activated, all active **portals** are deactivated (i.e., can't teleport the Player) and the **Player** can't shoot. The **Box** should deactivate itself after 10 seconds and all portals and the player should return to their previous behavior.

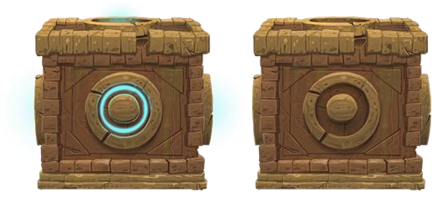

Figure 8. Representation of the active (left) and inactive (right) box

9. Report

Besides the project, you must also deliver a report, **in PDF format**, with the description of the implemented algorithms and an explanation for the taken decisions. The algorithms taught during the classes **should not** be described. The report should cite all the bibliographic references (books, websites, etc.) in which the code of the developed algorithms was based.

In the report, the list of the unfinished functionalities (partially or totally) should be **explicitly** mentioned as well. You should not include any source code in the report.

The report should follow the supplied template or, at least, its structure.

Criteria

The evaluation criteria for this assignment are the following.

Functionality	Score
Player	15%
Acid Bubbles	2.5%
Object Pooling	10%
UI	2.5%
Portals	10%
Scene Management	10%
Contact Areas	35%
Box	10%
Report	5%

Estimated date to publish the grades: 07/07/2021

Delivery Rules

The rules listed next must be respected for the project to be evaluated. Otherwise, **the work won't be accepted**.

1. Delivery deadline of the 2nd project: **19/06/2021.** The projects delivered after the deadline won't be accepted.

2. The project must be developed by a group of 2 students attending to the same practical shift (the projects developed by groups of more than 2 students won't be accepted).
3. The work should be developed using the provided project available in the **moodle** platform.
4. The developed work (Unity project + report) should be delivered through the **moodle** platform, using the link available for that purpose. **The project must not contain compilation errors or the implementation of non-requested features according to the assignment**.
5. **The report should be delivered in pdf format**.
6. All students must defend the delivered project so that they may be evaluated. The date of these discussions is indicated in the evaluation calendar of the course (These will take place in the week following the delivery date of the project).

7. The project must run in the version **2020.2** or **2020.3** of Unity's editor. Every project will be evaluated in this version of the editor as is.
8. The delivery of the project folder **must exclude** the folder "**Library**".
9. The name of the delivered file must be composed by the students' names (first name and surname) and numbers of students who carried out the project, respecting the following structure:

"**NameSurname1_Number1_ NameSurname2_Number2.(ZIP or RAR)**"

Third Examination Period

For the third examination period all the features from the second examination period are the same, plus the following must be implemented:

Audio Manager

Develop an **Audio Manager** responsible for the game background music. The Audio Manager should change the background music when a new level is loaded.

Hit Counter

Each **Spitter** must have a tracker above his head (UI Number), representing how many times a projectile from that **Spitter** has hit an **Enemy**.

Pause Menu

Develop an in-game pause menu that is activated when the user pressed a pause button. The options available in the menu should be: **Resume** and **Return to Initial Menu**. The **Resume** option makes the game unpausing, and the **Return to Initial Menu** option makes the game returning to the initial menu.

Box

Every time the box is activated it also makes a random number of existing **contact areas** to be deactivated. When the **Box** deactivates itself, it also makes the **contact areas** deactivated by the **Box** to return to their previous behavior.

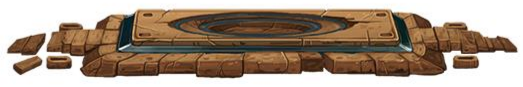

Figure 9. Representation of a deactivated contact area

Criteria

The evaluation criteria for this assignment are the following.

Functionality	Score
Player	5%
Acid Bubbles	2.5%
Object Pooling	5%
UI	2.5%
Portals	10%
Scene Management	5%
Contact Areas	30%
Box	15%
Audio Manager	5%
Hit Counter	10%
Pause Menu	5%
Report	5%

Estimated date to publish the grades: 30/07/2021

Delivery Rules

The rules listed next must be respected for the project to be evaluated. Otherwise, **the work won’t be accepted**.

1. Delivery deadline of the 2nd project: **13/07/2021.** The projects delivered after the deadline won’t be accepted.

2. The project must be developed by a group of 2 students attending to the same practical shift (the projects developed by groups of more than 2 students won’t be accepted).
3. The work should be developed using the provided project available in the **moodle** platform.
4. The developed work (Unity project + report) should be delivered through the **moodle** platform, using the link available for that purpose. **The project must not contain compilation errors or the implementation of non-requested features according to the assignment**.
5. **The report should be delivered in pdf format**.
6. All students must defend the delivered project so that they may be evaluated. The date of these discussions is indicated in the evaluation calendar of the course (These will take place in the week following the delivery date of the project).
7. The project must run in the version **2020.2** or **2020.3** of Unity’s editor. Every project will be evaluated in this version of the editor as is.
8. The delivery of the project folder **must exclude** the folder "**Library**".
9. The name of the delivered file must be composed by the students’ names (first name and surname) and numbers of students who carried out the project, respecting the following structure:

"**NameSurname1_Number1_ NameSurname2_Number2.(ZIP or RAR)**"