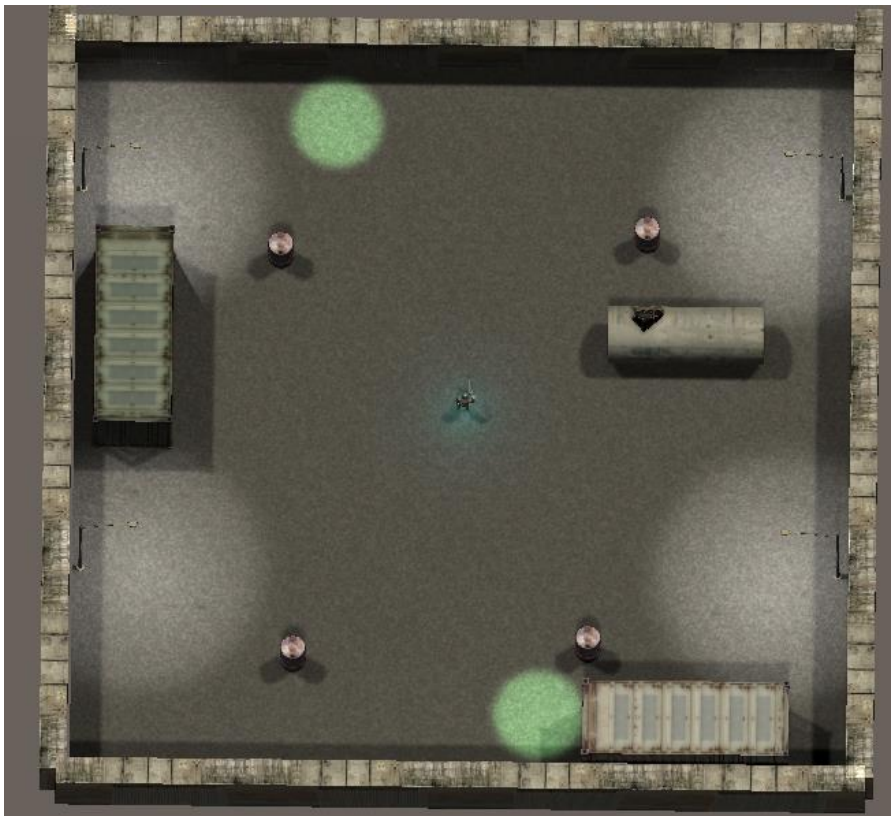
TÉCNICO LISBOA

3D Programming

2nd Semester 2016/2017



Project Report

Assignment 3

Group 3:

78303 Luís Espírito Santo

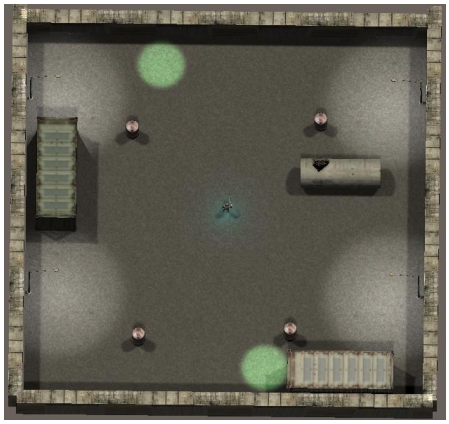
78414 Ricardo Silva

1 Introduction

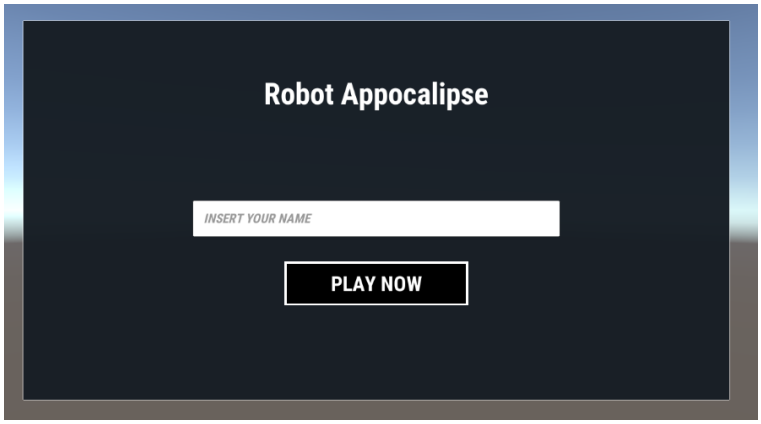
The objective of this assignment was to implement a simple game using Unity 3D game engine. We decided to implement a top view shooter where the player can run around using the ASWD keys or the arrows and can aim and shoot using the mouse. There are some enemy robots that try to attack the player and the objective of the game is to survive for as much time as possible. There are two types of enemies: a melee attack robot that just tries to chase the player and whenever it gets close enough causes damage to the player and a ranged enemy that can shoot bullets at the player. These enemies spawn randomly from the corners of the map. If you kill an enemy there is a small chance that an ammo box gets dropped so you can continue to play the game. If you run out of ammo an ammo box will eventually appear in the center of the map.

2 Game Area

To create the map of the game we used some prefabs downloaded from the asset store to build a square surrounded by four walls. The player and the enemies can walk inside these four walls. To decorate the scene and make it more interesting we used some shipping containers, oil barrel and a concrete pipe. Each object has its own textures (albedo, specular and normal map) to add realism to the scene. We also added four light posts and each street post light has a spot light pointing down and lighting the map.



Overview of the Map



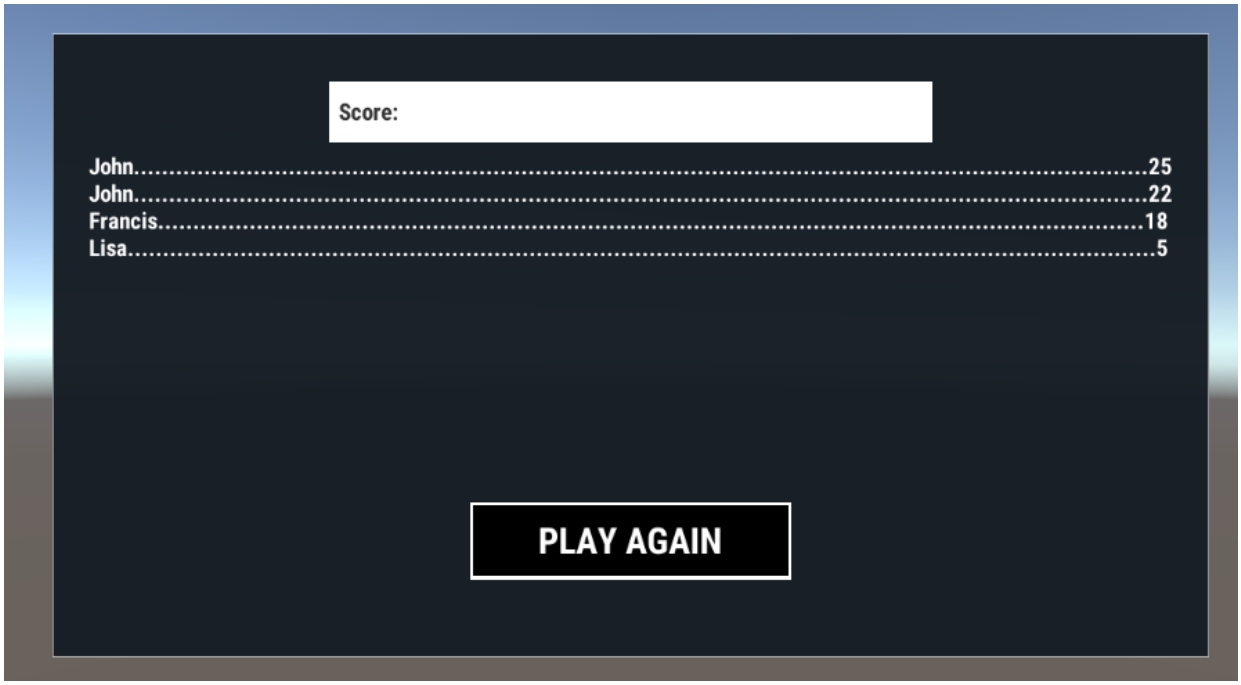
Start Screen

3 Start Screen

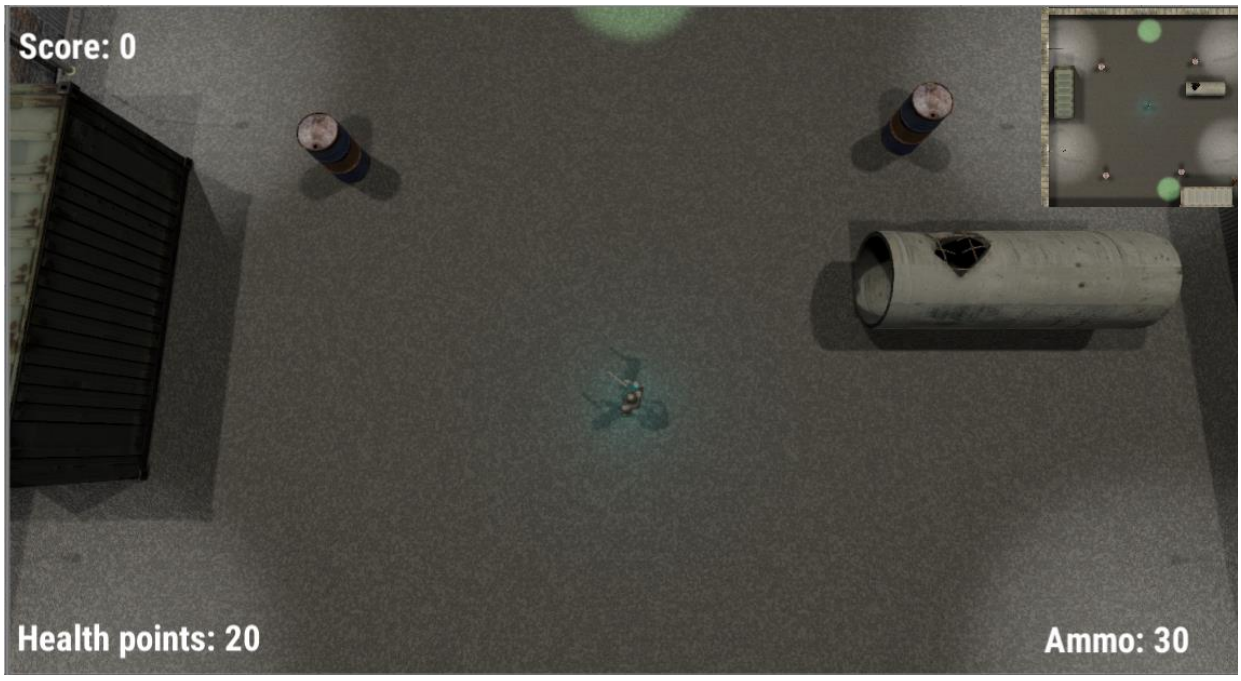
To make the starting screen of the game we used a dark image for the background and added three elements to the scene: Title of the game, input box where you can input your name and the play button. The input box receives your name so that it can later be used to fill the leaderboard. The play button takes you to the main scene where you can play the game after you click it.

4 End Screen

To make the end screen we added a text box where the current score is shown, a scoreboard that show the top scores and the players who achieved that score and a replay button that enables the player to restart the game



End Screen



HUD

5 Heads Up Display

We used a Heads-Up Display (HUD) to show the most important info to the user: remaining health points, number of bullets available, the current score and a mini-map. To implement the first three elements, we used simple text elements that get updated whenever the state of the game changes. For the mini-map we used an orthographic camera that is centered on the map and draws the mini-map in the right top corner of the screen.

6 Colisions

6.1 Character controller

In our game, the player can pick up ammo boxes that get dropped by the enemies. The collision is processed by `OnTriggerEnter()` inside the player controller script when the collider of the player and the collider of the boxed intersect each other.

6.2 Changing the Map

Whenever the player aproaches the bottom left lamp post the light goes out. When the player collides with the collider of the lamp post we use `OnTriggerEnter()` to disable the light and `OnTriggerExit()` to enable the light again. This creates a nice flickering effect

6.3 Autonomous Objects

We have two green light that move autonomously to simulate some sort of prison guard lights that is searching around for fugitives. To create this moving light, we used a spot light attached to a rigidbody and a collider and a bigger box where the light can move. While the light is inside the bigger box the light goes to the right and when it reaches de limit of the box it starts to go left until it reaches the limit and then it starts to go right again.

6.4 Static Enemies that shoot at the player

To implement this requisite, we decided that it would be more fun to add moving enemies that chase the player and shoot in his direction. To create these enemies, we first needed to create a navmesh so that the enemies could walk around the map and path find the player. After the enemies could chase the player we added a shooting mechanic so that spheres get shot from the weapon of the robot in the direction of the player. If the player gets hit he loses 1 health point.

7 Game Mechanics

The player can move around and shoot his weapon at the enemies.

Melee enemies have 1 health point and cause 1 damage per hit.

Ranged enemies have 5 health points and cause 1 damage per hit as well.

When the player catches an ammo box he gets 5 extra bullets.

When the player kills an enemy 1 point is added to the score.

The player loses the game whenever his health points reach 0.
Explosive barrels that explode after taking 3 hits. Explosion in a circle that knock offs the player and enemies back and causes 5 damage.

8 Visual Effects

8.1 Bump Mapping

All our scene objects use normal maps to do bump mapping. Bump mapping allows us to achieve a greater level of detail without the need to increase the number of polygons that make up an object. It works by changing the normals of the surface in different points and then the lighting calculations create the effect.

Without Bump Mapping

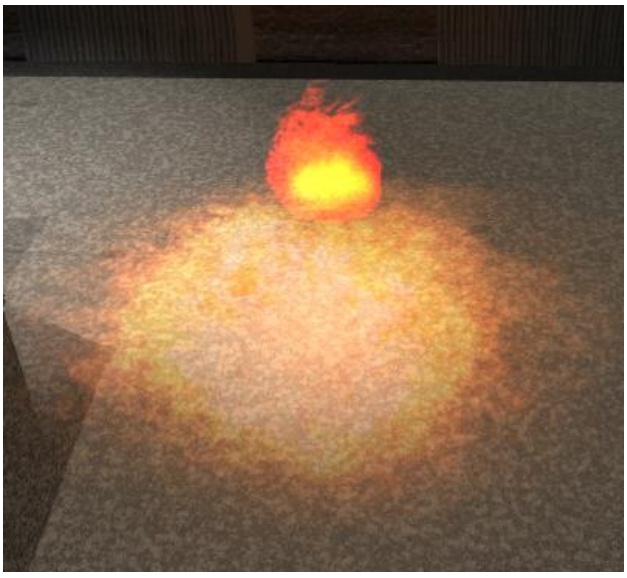
With Bump Mapping

8.2 Particle Systems

When the player shoots an explosive barrel, flames start to emerge from the top of the barrels. There are three levels of flame intensity according to how close the barrel is to explode.

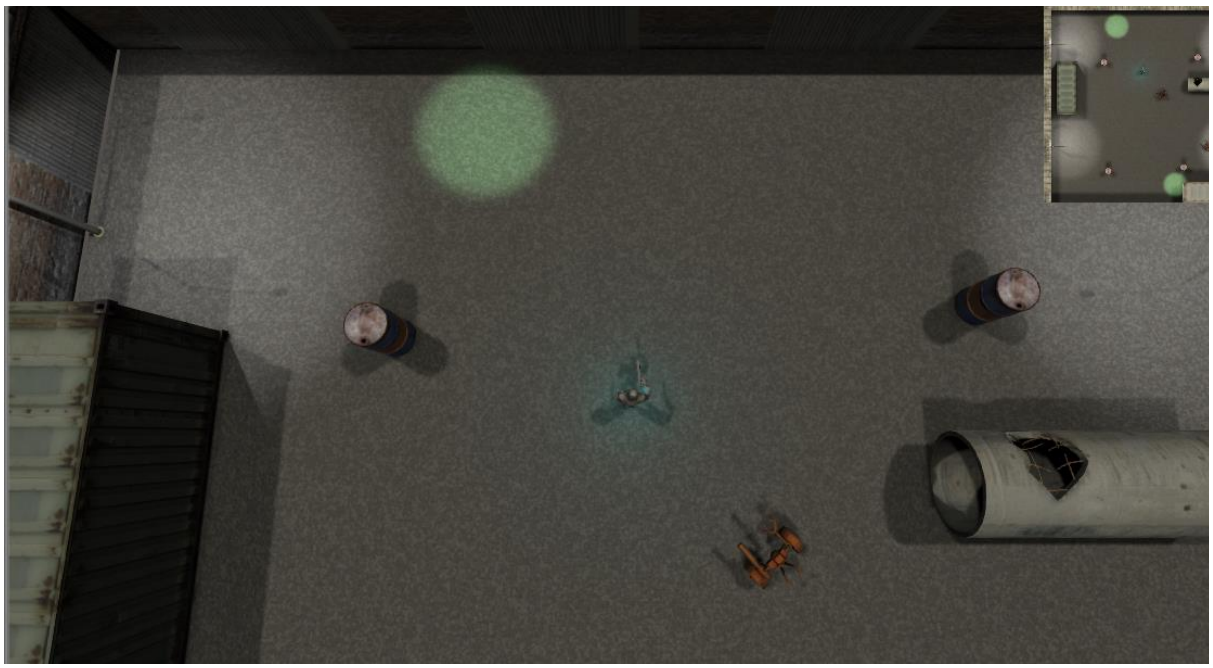

First level

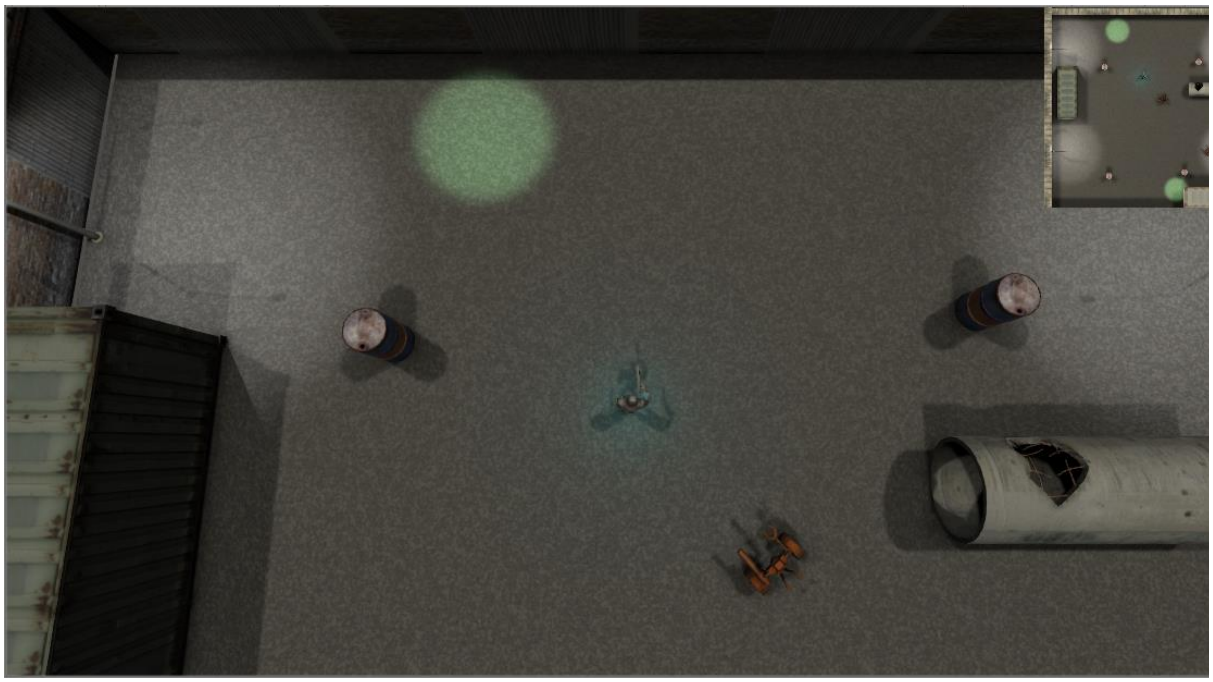

Second Level


Explosion

8.3 Ambient Occlusion + Light Maps

To achieve the ambient occlusion effect, we used the Unity built in Screen Space Ambient Occlusion script that attaches to the main camera of the scene. To use the light maps we selected all the static objects in the scene and baked a light map for them. This way all the lighting computations that involves these static objects can be avoided, since they never move. We have attached a GIF to better illustrate the changes since it is hard to see them with just still pictures.

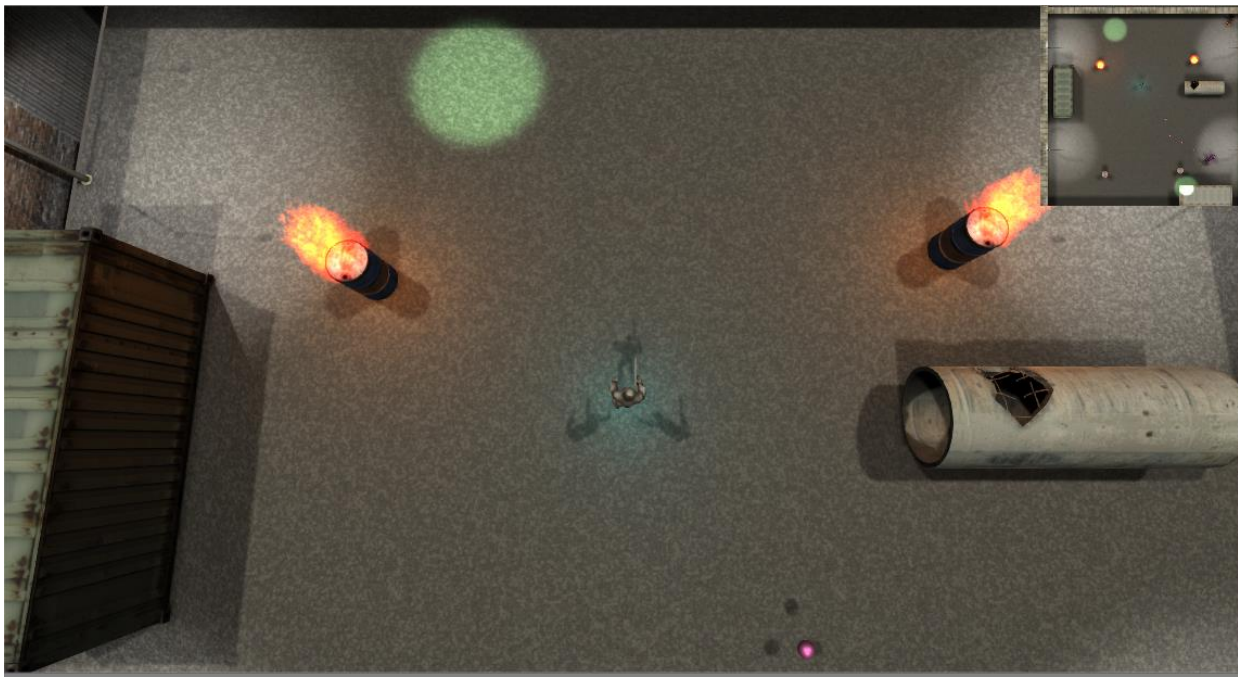

No ambient occlusion



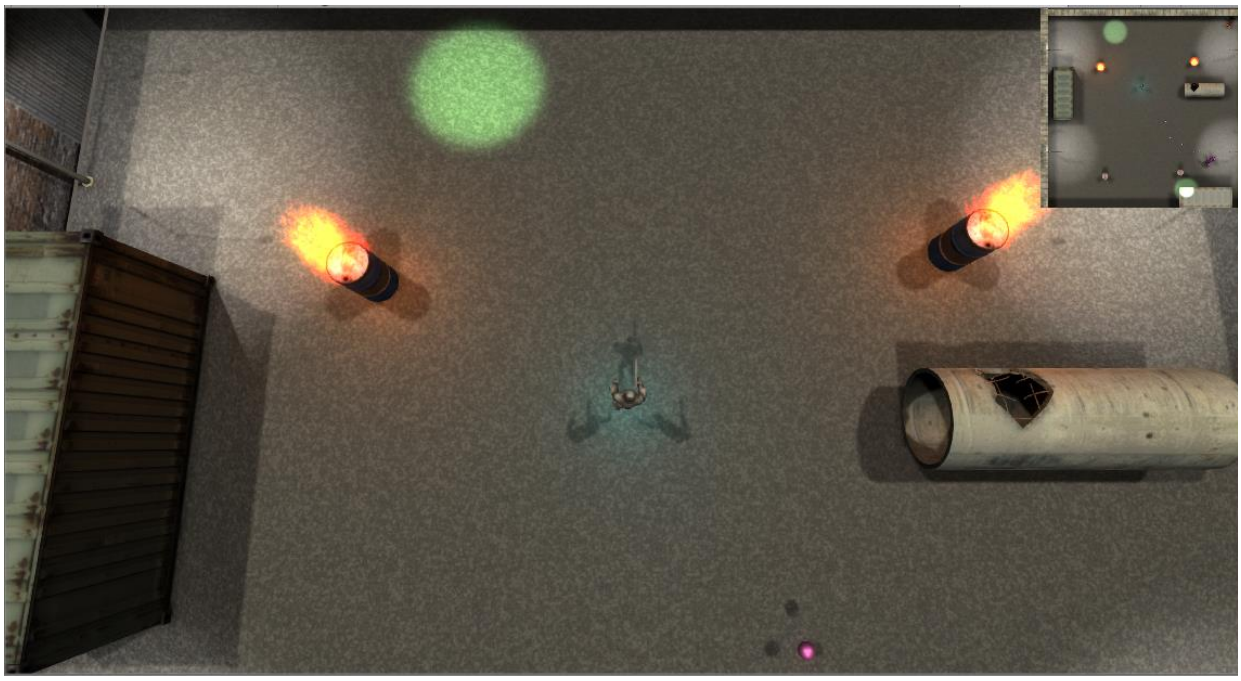
Ambient occlusion

8.4 HDR Bloom + Tone mapping

To achieve the bloom and tone mapping effects we used the built-in features of Unity and added both effect to the main camera of the scene. We then played with the parameters of both screens to try to get the best results possible. We have attached a GIF to better illustrate the changes since it is hard to see them with just still pictures.



Default



Bloom

Bloom + Tone mapping

9 Extras

To make our game more appealing we decide to add some animations to our characters. We use two different 3D models of robots obtained from the assets store, one more futuristic used for our player's character and another one looking rustier and older to represent our enemies. After that we animated our models with a multilateral movement animation. This animation is

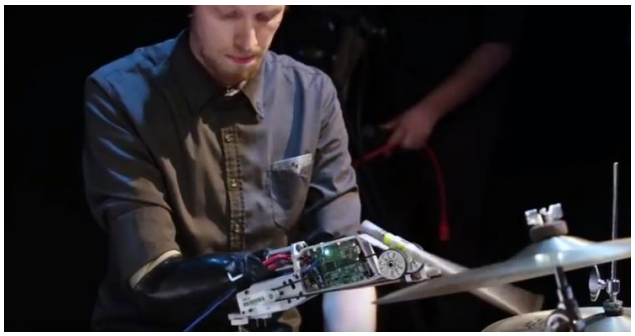
the same for every kind of movement. Also, we develop an idle animation clip for our main character to maintain a dynamic feedback and endow our model with a more living appearance. The coordination of the different animation was made using an Animation Controller object.

3D Model – Player's Character
https://www.assetstore.unity3d.com/en/#!/content/52064

3D Model – Enemies' Characters
https://www.assetstore.unity3d.com/en/#!/content/60200

Finally, we also decided to add some sound effects and background music, to create a more immersive game. For the background music, we use a live performance from Georgia Tech's *Robotic Musicians and Musical Cyborgs* that features Shimon, the robotic marimba player, and a drummer who wears an assistive robotic drumming prosthesis. We considered that both the origin of the soundtrack and the musical expressivity of it reflect the tension of our game concept: a Robot Apocalypse.

Shimon, the robotic marimba player
https://youtu.be/l9OUbqWHOSk



Jason Barnes, the cyborg drummer

Besides this background music, we added three different sound effects, all of them extracted from the *Freesound* website (https://www.freesound.org/): one explosion sound that is played when one barrel explodes; a futuristic lasergun shot sound effect for our main character's shots; and one bubble like sound for our shooter enemies. All of this sound resources were added using Audio Sources Components to our Game Objects.

10 Conclusion

We enjoyed completing this assignment because it enabled us to learn more about unity and improve our capacities to build a game from scratch. We didn't have as much time to complete this assignment as we would like due to excessive work from other courses but in the end, we managed to complete the assignment successfully and we think the end result is quite good.