3D Programming: Assignment 3 Report

André Fonseca, Samuel Gomes, Tiago Almeida

June 4, 2017

1 The Player Module

The player script includes the creation of an inventory and also its management. Aspects like the player sounds are also managed in this script.

1.1 Inventory

The inventory consists in a *List* of *GameObjects* which correspond to *Gun* objects. Those objects are first gathered in a *List* called *allGuns*. When a player picks a weapon, the correspondent game object is removed from *allGuns* and put on *inventory*. In other words, when the player picks all weapons there are no game objects in *allGuns*. When a player picks a weapon of the same type of one in the *inventory*, the bullets of the picked gun are added to the players' gun.

The player script always keeps track of the current gun (*currentGunIndex* which is 0 at start), because it saves such weapon's index on the inventory. When a gun change is needed, *currentGunIndex* is increased or decreased and another weapon is picked based on such index.

1.2 Moving the Player

The player movements and camera control was implemented by using a prefab called Fps Player Controller imported from *Unity* standard assets. This prefab contains scripts that make the player controls realistic. The player can move in all directions and jump, also it allows the adjustment of each of those movements' speed in the *Unity Editor*.

1.3 Player's life

The player's *life* is simply represented by an integer that starts at 100 (full health) and drops to 0 (dead). The life is reduced every time the method *injure* is called, unless the player has armor, which is instead decremented.

2 Weapons and Bullets

2.1 Weapons

Weapons are represented by *Gun* objects. They include attributes like their *numberOfBullets* (current number of bullets) and *maxNumberOfBullets*, the *numberOfPickupBullets* (the number of bullets that can be picked up from the gun), the gun type's attributes (*gunType*

and *isABulletsWeapon*) and the firingDelay (the delay between gun shots, for example machine guns have low delays and bazookas have high delays). The gun's bullet game object is also included (when the gun is a bullets weapon) in order to instantiate bullets everytime the gun fires.

2.2 Bullets

Bullets are represented by *Bullet* objects. They include attributes such as the *bulletSpeed* (the speed that the bullets can travel), a *massFactor* to efficiently simulate gravity through a vertical negative translation, the *bulletDamage* (the amount of damage the bullet injures the player and enemies) and *destroyable* (to indicate if the bullet is or not permanent, for example a laser is a permanent bullet). The destroyable bullets use the attribute *maxBulletTime*, which represents the max life span of the bullet object.

To simplify the bullets' collisions, a translation is applied to each bullet. That translation is mostly on the forward direction but, as mentioned before, it also serves to simulate gravity (mostly in bazooka bullets). The collisions are detected by using the *OnTriggerEnter* method relative to the bullets' colliders. When a collision is detected, a bullet checks the object it has collided to and triggers events accordingly. If it collides with an enemy or the player it injures such element. If it collides with a box, it calls a method which triggers the particle system associated with the box and destroys such object.

3 Particle Systems

Several types of particle effects were included in the gameplay, such as the bullets' smoke and explosion on impact, the explosions effects on enemies and guns or the torch flames on the walls.

3.1 Explosions

The explosions consist in burst that emanate instances of a special explosion sprite. The enemies' blood is represented by a *Unity 3D* default particle system with red start color. The boxes explosion it's similar to the blood but colored brown.

3.2 Bullets' Particle System

Two types of particle systems were implemented on the bullets. The explosions on impact consist simply in default *Unity 3D* particle systems with a color change over lifetime from yellow to red. However, the bazooka's bullets use the explosion particle system on impact.

3.3 Torch Flames

The torch flames were done by using a particle system and point light. The particle system use square particle sprites, and change the color over lifetime between yellow, red and gray. For the shape of the particles we customized a cone shape to make it look like a flame.

4 Graphical Effects

4.1 Post-Processing Effects and HDR

In order to apply the requested graphical effects like Ambient Occlusion, Bloom and Depth of Field (and also Motion Blur), a package called Post Processing Stack was imported, which includes a wide variety of shaders. The interface is straightforward. We just had to select and tweak the effects on *Unity 3D*'s inspector section.

Other effects like HDR were implemented on some materials' color. One example is the bullets' materials.

4.2 Lightning

The lightning model we used for our game was toon shading. We used a material shader included in *Unity* called Toon - Lit Outline.

4.3 Normal Mapping

In this game we implemented normal mapping in the rock model, the wood secret walls and floor and in the boss room brick walls. The objective of the inclusion of this technique was to improve the level of illumination detail on the most non regular surfaces. The normal maps were generated by a *Paint.net* plug-in called *normalMapPlus*. Then, those textures were exported to *Unity* as normal textures and attached to the respective materials.

5 User Interface

5.1 Start Screen

The start screen is a different *Unity* scene from the one where the game occurs. It has only two images and buttons. The logo image for the game was created using *Photoshop*. After pressing the play button, the main scene is loaded and the game starts.

5.2 Heads Up Display

The heads up display is divided in 4 parts: life, weapons, map and score. For both life and weapons, the information is retrieved from the player class and displayed on the screen. The map uses an additional camera in the game that is above the player and follows him as he moves. This camera ignores certain objects in order to be easier to see the player (ceiling, player weapons, etc.). The resulting image is saved in a texture that is then displayed on the screen. The score UI contains the score gained by the player after executing certain actions. The Score system informs the user interface whenever a new score is received. It is then shown on the screen with a quick animation. Since the animation takes some time and multiple scores can be received in such time span, a heap of scores is stored in order to show them. Multiple scores of the same name are merged with a multiplier number in front of them. This technique avoids receiving score messages for too long.

5.3 Score System

Score system is a small class with only the current score and a reference to the heads up display. However, it is different from other classes since it is persistent (not destroyed on

scene change). This happens because the score value must be read in the last scene: the end screen.

5.4 End Screen

The end screen is divided in two parts. The first is the new highscore and the second the leaderboard. When a player achieves a score high enough to belong to the leaderboard they start in the new highscore screen where they submit their name to be added to the leaderboard. After a small animation, they are transported to the leaderboard where their score is displayed. If no new highscore was achieved, only the leaderboard is displayed. The player can then quit the game or play again. When starting this scene, the leaderboard info is loaded from a file. If the loading fails, a default leaderboard with the developers' scores is shown. Whatever the case, the current leaderboard is saved to that same file location to be loaded the next time.

6 Action Feedback

An important part of games is to send feedback to the player whenever an action occurs. In this game, we gave feedback to player action through two means: animations and sound effects.

6.1 Animations

Animations are an integral part of giving feedback to the player. They were created using *Unity*'s built-in animator. However, to avoid the animated game objects to leave their position when animating, a hierarchical child had to be added and the animations only acted on the child. In the case of the boss, a second layer of animations was added for animations whose position depends on other animations. In total, three different objects were animated: The player for getting hit and dying, the enemies for getting hit and their multiple deaths and the boss for his movement, getting hit and death.

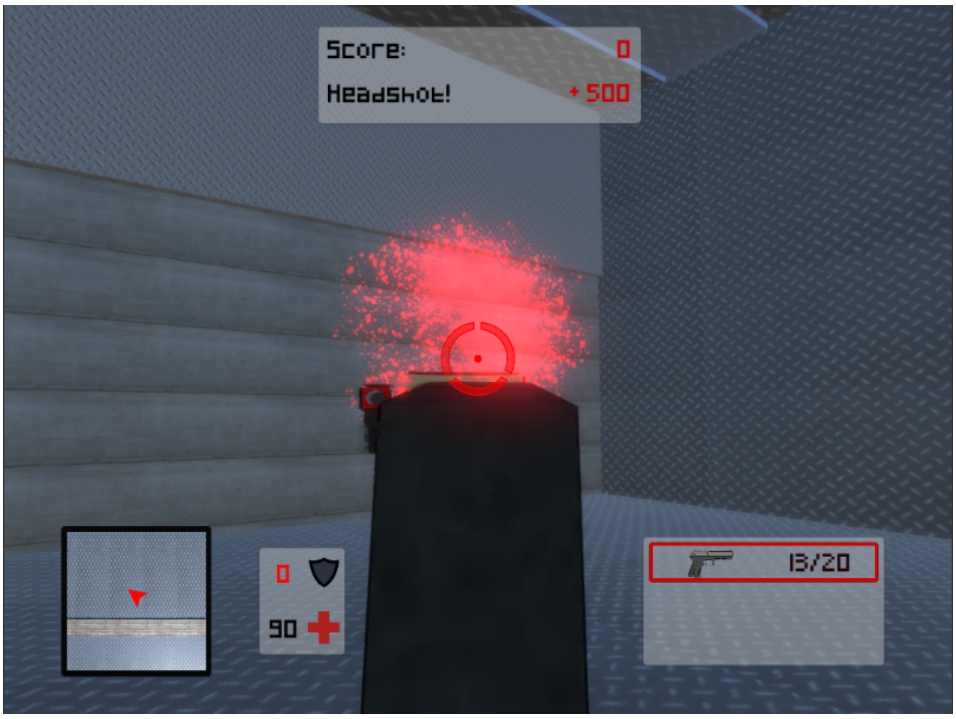
6.2 Sound Effects

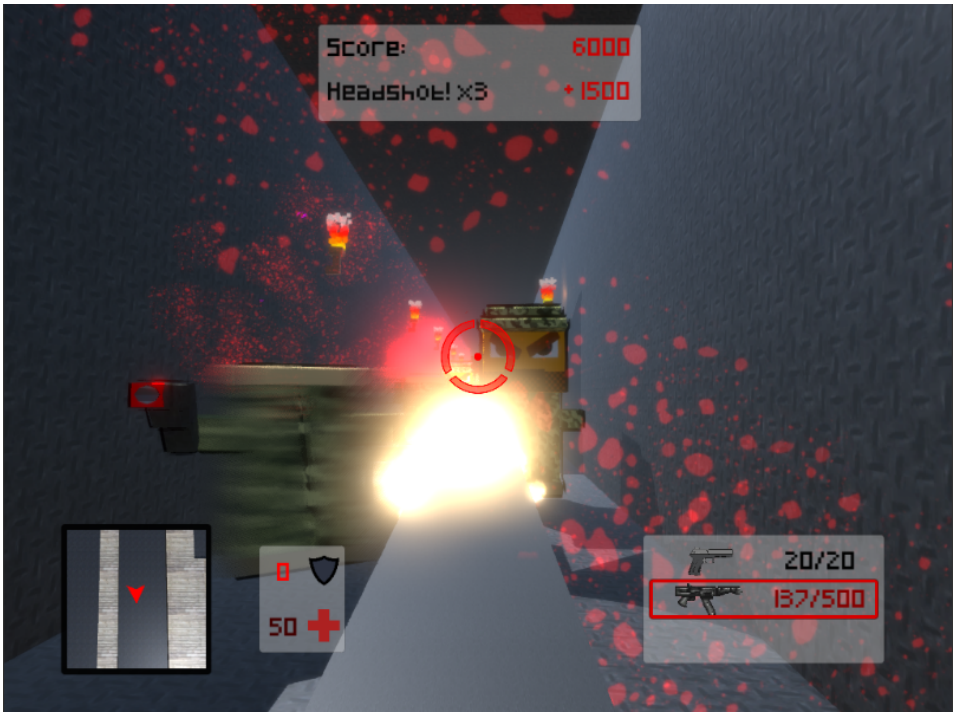
Another aspect of feedback to the player is given by the sound effects. To simplify the sound design, all sounds are 2D. From health pickups, to weapons firing or scoring points, dozens of effects were included in the game. Also, the main level plays music that changes on special occasions: The game starts with the level music, then once the boss room is reached the boss music plays and finally, after defeating him, the victory song is heard. Some code was also added to play the first two song that are divided in an intro followed by a looping score.

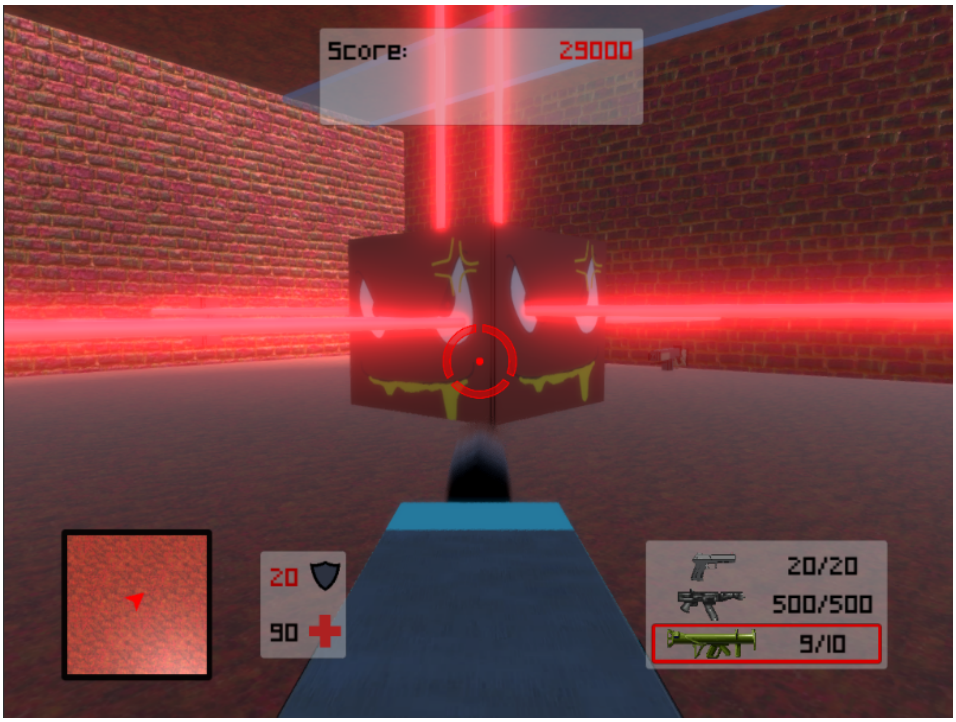
7 Level Design

All the meshes of the game were designed with *SketchUp*, a 3D modelling program. *SketchUp* was really easy to use and allowed us to export models with textures directly to *Unity*. Once we had all the components for the game, e.g. enemies, guns, we began designing our game. We designed the level in a way that rewards exploration and encourages the use of new guns, also finishing the game within a time interval grants more points. The level consists of 6 rooms, 1 initial room with the first gun, 3 rooms with enemies, 1 secret room with weapons, hp and armor and a boss room.

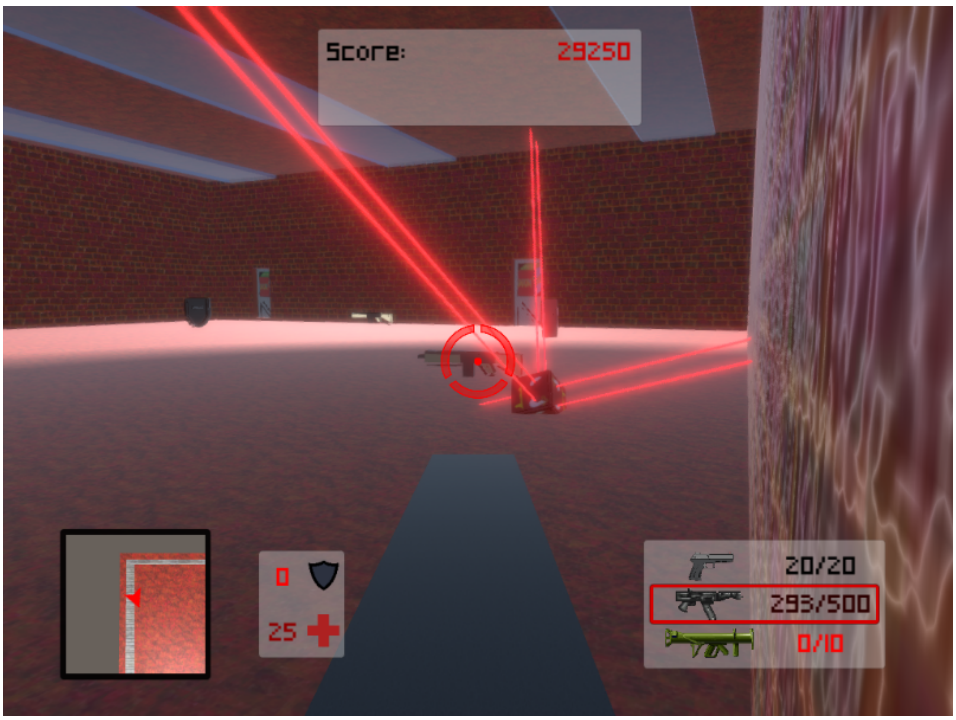
Appendix - Screenshots

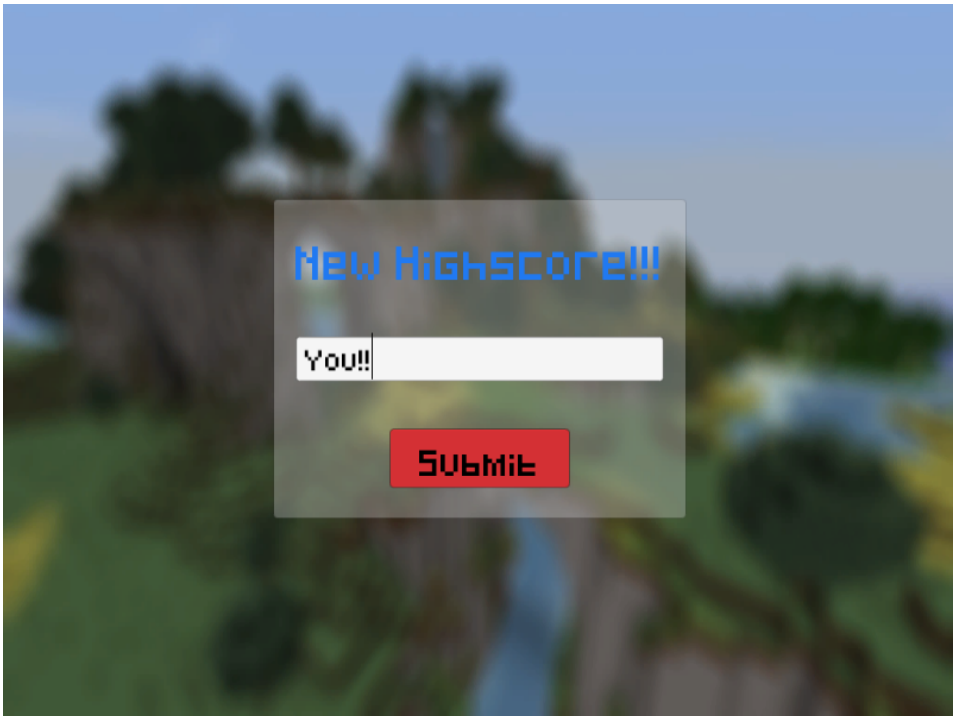





Score:
29000
20
90
20/20
500/500
9/10


Score:
29000
0
75
20/20
500/500
4/10


Score:
29250
0
25
20/20
293/500
0/10


New HighScore!!!
You!!
Submit


Top Scores!!!
Rank
Name
Score
1st
André Fonseca
79500
2nd
Samuel Gomes
77000
3rd
You!!
76500
4th
Tiago Almeida
75500
5th
Daniel Rodrigues
24500
Play Again
Quit Game