Virtual Reality

2018/2019 - Fall Semestre
MEIC-A / MEIC-T

Project 2 - VR Interaction

Group #	*11*
Student 1	*Ana Marta Mendes, 69799*
Student 2	*João Bernardo Alves, 82547*
Student 3	*Telma Barragão, 89241*

Link to project repository (private share): *https://github.com/telmabarragao/VR*

Indicate software versions
Unity 3D: *Unity 2017.1.1f1*
GVR SDK for Unity: *GoogleVRForUnity_1.170.0.unitypackage*
JDK: *jdk 1.8.0_131*
Target API level: *Android 7.0 Nougat – Level 24*

Main Goal: The main purpose of this project was to use a simple 3D VR scene as a base (in this case, the first project) and improve it by adding interaction elements, enabling the user to interact with the virtual scene by controlling the camera and using the gaze input.

Tasks:

The scene is the same as in the previous project, with the addition of a HUD containing the labyrinth map and the positions of the “minotaur” (in our case, a deadpool 3D model) and the player, which is updated every frame. There is also a sword object placed in the scene which is used to defeat the enemy. The Gvr prefabs such as GvrControllerMain, GvrEventSystem, GvrEditorEmulator and GvrReticlePointer had already been added in the previous project.

1. This task was achieved by creating a new orthographic camera facing downwards, standing at a distance high enough to see the scene from a top down view. The camera

was rendering to a texture called “Minimap texture” which in turn was added to the canvas that serves as the minimap.
Problems faced: when we hit play, the coordinates for the camera would change inexplicably.
Solution: In the debug option for the minimap camera, the value for “target eye” was 2 and had to be changed to 0.

2. For this task, we added the VRLookWalk script to the player object, to make him walk forward when lifting up the head 30º and backwards when lowering the head 30º.
   Problems faced: The walking animation didn’t seem very intuitive and fluid when testing.
   Solution: We changed the script by creating two bool variables to move forward and backward. We then created more conditions related to the angles for moving forward, backward and standing still.

3. In this task we had already added the GvrReticlePointer prefab to the main camera that belongs to the Player object. We also added a sword 3D model to act as the “spear” and placed it at the end of the maze. Following the lab indications, we added an empty game object called VRHand as a child of the GvrReticlePointer, and added the VRStare_and_Grab.cs script to the sword object, making the necessary input attributions. To activate the events, we added an EventTrigger to the sword object and the events “pointer enter” and “pointer exit” with the respective values.

4. Finally, we created box colliders for the enemy and sword game objects and a script for the sword object called DestroyDeadpool which would destroy the enemy game object when colliding with the sword.
   Problems faced: After grabbing the sword, we couldn’t make it collide with the enemy. After charging at him, nothing would happen. The collider seemed to be staying behind after we grabbed the sword.
   Solution: Since the sword had a child object with the renderer as well as the rigid body and we were grabbing the child object, it gave the illusion that we were grabbing the sword when we actually weren’t. To fix this, we changed the rigid body to the actual sword object (parent).