Virtual Reality Lab 1

Daniel Ariza Anton and Marcos Ilarraza Sarto (797720)

April 9, 2024

1 Introduction

The objective of this report is to explain which steps were taken in order to fulfill the different tasks that were asked for the third laboratory of Virtual Reality.

2 Addition of teleporting paradigm [MT-0]

For this task we adapted the code of the script *MovePlayer* to implement a teleporting paradigm when left clicking or pressing the *T* key. When it is pressed, the transform of the CameraPlayer is modified trough the function *Transform* from Unity's library with the forward multiplied by the value of *DistanceMultiplier* (which is five).

3 Addition of teleport marker [MT-1]

To mark the position in which the player will be teleported we have used a basic 3D cylinder figure from Unity and modified its aspect in order to give it the shape we wanted. On the first press, the marker is displayed on the position were the player would have been teleported with the previous implementation. With the second press, the position of the CameraPlayer is updated by assigning the X and Z values of the marker to its own, and the marker disappears afterwards. We have decided to do it this way because we can reuse the calculations made to display the target position instead of calculating the translation again.

We have implemented two versions of the teleport marker (toggled through the variable *moveAlongside*): one that **follows** player's camera (which also updates the new teleported position) and one that **doesn't follow** player's camera (which maintains the teleported position captured when the marker is displayed). The second version is the simplest one: With the first press it calculates where the player will be teleported and displays the marker on that position until the second press. As for the first one, we need to update the marker's position each frame in order to make it follow the camera. At first we just made the marker a child of the CameraPlayer, but it also made the marker move along the Y axis. It didn't affect the teleported position since we only used its X and Z coordinates, but it didn't look good. In the end we decided to recalculate the teleported position in each update and to assign it to the marker, which solved the problem of translation of the marker along the Y axis. Both teleporting modes markers can be seen in figure 1, and two videos that show how they work are attached.

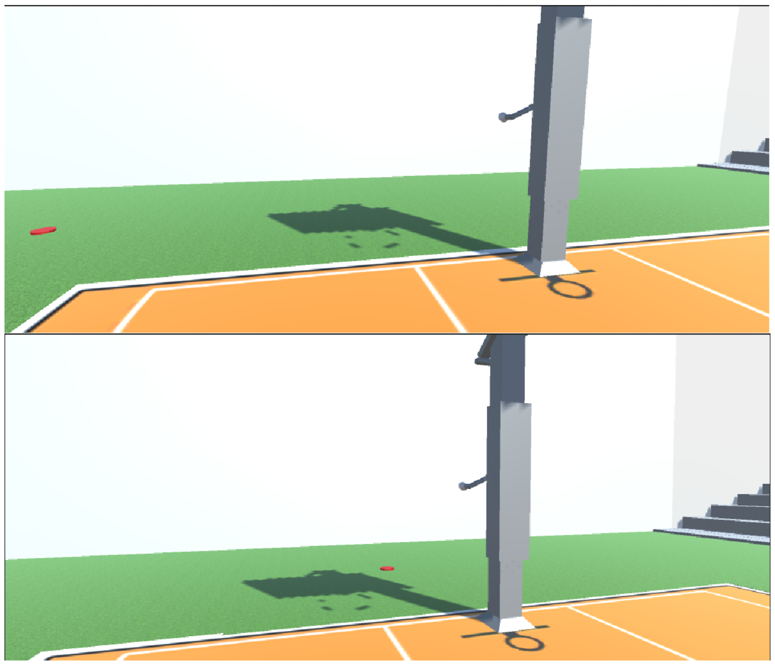

Figure 1: Vision of the target teleport marker following the camera of the player (bottom image) and without following it (top image).

4 Ideas for an HMD and accessories with tracking [MT-2]

If we had an HMD with tracking, we could implement a movement paradigm which would allow the player to move around the environment of the game through moving in the real world. If we had an accessory controller that also allowed tracking, we could also implement a way to move the ball in front of the player without the need to move their head. If our accessory also had the capacity to specify which button is being pressed, we could add functions such as *stop tracking the movement of the player* or *stop tracking the movement of the hands*. In that way, the player can change its position in the real world without moving in the game. This can be useful for moments in which the position of the accessory isn't calibrated to what is seen in the game and when the player wants to move forward but there is no more room for it in the real world. If the HMD had a camera towards the real life environment, we could also map it and model the game environment so we can "push" the player to walk in a safe direction instead of a blocked path/wall.

5 Insights over ***Half-life: Alyx*** [OT-0]

We have watched official gameplay videos ([3] and [2]) from *Valve* and we have detected two moving paradigms. The first one is the tracking of the player's movement in the real workd, which is useful for small distances such as look behind a corner. The second one is the teleporting through a marker on the game, which is useful to move alongside the map without the need of a huge space to play.

As for the interaction, we have seen that the game has "direct" interaction through the controllers by representing them with hands that interact with the rest of the elements of the game. They have also implemented an interaction through a point and drag mechanism, similar to the teleporting paradigm, to bring to the players hand objects in a mid distance. The objects within the game can also be used to interact with the elements of the scene, as using the gun to open a pot or throwing a pipeline to an enemy. We believe that the reason for needing both the drag and teleporting is the same: reduce the needed amount of space in the real world that the player may have to play.

Lastly, we searched for the list of moving paradigms it implements ([1]), and apart from the two mentioned before, it also implements two that make use of continuous movement (through a joystick or something else): one in the faced direction and another one in the direction pointed by your dominant hand.

6 About our implementation

The objective of this section is to explain some design decisions that were taken when implementing the basic functionality of the scripts, but that weren't directly related with any of the tasks of this lab.

6.1 Desktop implementation

We were asked to implement both the displacement of the player through the *WASD* keys and the rotation of the camera through the use of the mouse. For the rotation we copied the code used in lab 1 directly at the end of the *update* function. As for the displacement we took as a base the code provided in lab 2. To fix the movement only alongside the X and Z axes we temporally set the rotation around the Z and X axes to zero, so only the rotation on the Y axis is taken into account when applying the translation function is used. The rotations alongside the Z and X axes are recovered before calculating the new rotation of the player. It is visible in the script *MovePlayer* provided alongside this report.

6.2 HMD implementation

For the HMD script, called *Headset_MovePlayer* and provided alongside this report, we adapted the code from our original movement implementation for the joystick displacement instead of the method proposed by the script and it works well. We still removed the code for rotation and for the teleporting paradigm and switched the "Z" key press with the "any button" press.

References

[1] H. gabrielmosspdx, brianxbang. https://www.ign.com/wikis/half-life-alyx/how_to_move. *IGN*, 2020.

[2] Valve. https://www.youtube.com/watch?v=ltlotwkplgk. *Youtube*, 2020.

[3] Valve. https://www.youtube.com/watch?v=nfjtvmka54e. *Youtube*, 2020.