Esclab

Janine Mayer[*]
University of Applied Sciences Upper Austria

ABSTRACT

As part of the ”Virtual Reality” course, a small labyrinth game was created in cooperation with Axel Bauer in which the player has to find coins in a labyrinth without being killed by one of the monsters. The player is able to jump between several dimensions of the labyrinth and can move marked walls. The game was inspired by a challenge from the Japanese game show ”Takeshi's Castle” and the classic board game ”Labyrinth” (”Das verrückte Labyrinth”).

Index Terms: Human-centered computing—Virtual Reality—Game—Labyrinth

1 INTRODUCTION

Esclab is a labyrinth game that uses Virtual Reality (VR) to draw the player into the spell of a small ”crazy” labyrinth. For this game, the player only needs an HTC Vive, Lighthouses and two HTC Vive Controllers. The aim of the game is to collect all the coins scattered in the two parts of the labyrinth. You must not be caught by the small monsters hidden in the maze. These are small and easy to overlook, but can cause the player considerable damage.

Virtual reality makes it possible to immerse oneself completely in the game, with atmospheric light, music and sound effects providing the right mood for a short, exciting game.

The labyrinth game was designed and implemented in close co-operation and with a lot of consultation with Axel Bauer. During the implementation, I took care of the logic of the character, the scene change via portals, the selection and re-positioning of certain walls and the design and placement of the player's status bar. Axel took care of the game controller, the start and end scene, the various items the player can collect and the enemy (including pathfinding AI). This documentation deals primarily with the part of the project for which I was responsible.

The code for this project is available here on the public *”Esclab” Github repository* [4].

2 USER GUIDE

This VR game is designed to be played with an HTC Vive and HTC Vive Controller, where as Lighthouses are used to track the user in the configured room. A tracked free space of at least 3m x 4m is necessary to play this game safely. This game is not suitable for people with claustrophobia.

The game starts in an initial scene where the user is instructed to touch a cube to play the game. For this purpose, the controllers are already used and the user is prepared to use them actively in the game. From this start scene, the user is led directly into the labyrinth.

The labyrinth is divided into two parts that are connected via a portal. Here the user can collect coins by grabbing them, 10 of which are scattered throughout the entire labyrinth. The labyrinth has a fixed floor plan, although some marked walls can be moved by the user. These are highlighted when the user comes within a certain radius and can be grasped using the trigger of the right controller. The moving is controlled by mapping, so the user can move in the room and the selected wall moves with him. By clicking on the touch-pad of the right controller, the selected wall can also be rotated. The player is advised to move each of these walls to open up other parts of the labyrinth and find hidden treasures. The danger in the labyrinth is manifested by two dragons which sense the characters as soon as they enter within a certain radius and take up the chase. If the player is caught and attacked by the dragon, he loses health points. If there are no health points left, the game is lost. The user can fight off the dragon if he finds an ”Invincible” item and can thus disable the dragon. To heal damage, there is one healing item per labyrinth part. The current status of the health and the collected coins can be viewed by clicking the track-pad of the left controller. The health and coin status bar is mapped onto the controller at a slight distance, giving the feeling of a ”wearable”. The game is won when all 10 coins are collected.

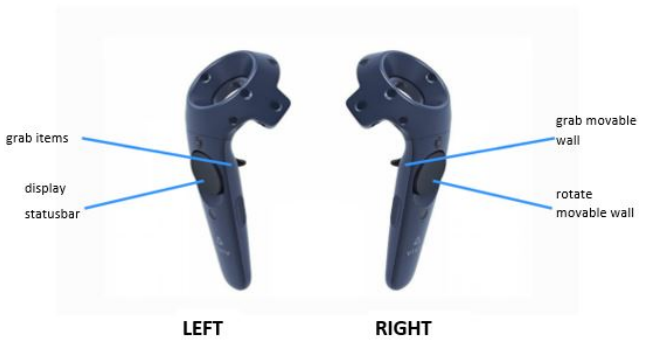

Figure 1: The left and right controller are used for different input options. The left hand can be used for interaction with items and as a display possibility for the status bar while the right hand allows the user interactions with movable walls.

At the end of the game (whether through the death of the player or the collection of all coins), the player is redirected to an ”end scene”, which gives the player the opportunity to restart the game.

3 DESIGN

This section deals with the game design, some of the used elements, interaction possibilities as well as the assets used.

The design of this game was iterated and refined several times. Initially, many ideas were collected on how to get the largest possible area for a labyrinth with a small tracked area. The idea of making better use of the available space came from the paper by Suma et al. [7] presented in the lecture. The dynamic change of space described in the paper would have been very interesting but also time consuming. After a short research, the topic of non-euclidean spaces also came up [1]. Due to the limited time available and in view of the presumed effort, we decided on the simple solution of a portal that connects two scenes.

The room layout is now composed of several scenes that are connected by teleport points and portals, and the point at which the player lands is predetermined.

The focus of the design was to provide the player with a great

[*]e-mail: janine.mayer@fh-ooe.at

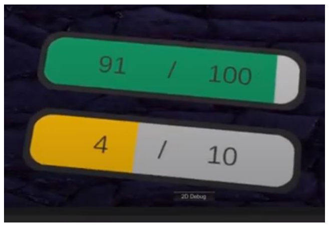

Figure 2: Health and coin bar are positioned above the left controller and can be queried whenever needed. Here a health bar with 91 of 100 health points and a coin bar with 4 of 10 coins can be seen.

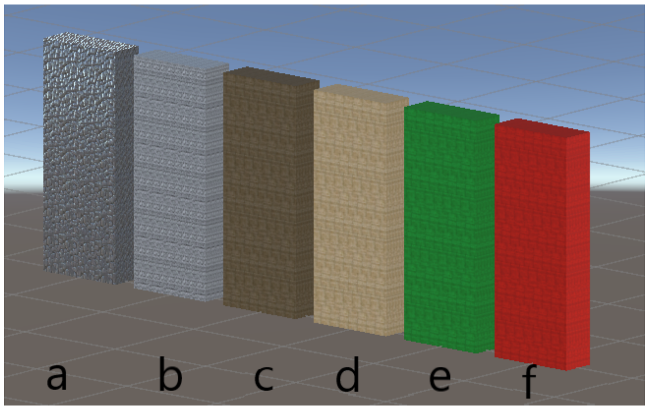

Figure 3: a, b) two normal walls, c) a movable wall seen from further away, d) a wall when within selection reach, e) a movable wall when selected and place-able, f) a movable wall when selected and not place-able.

gaming experience, which is why a lot of attention was paid to the interaction possibilities. The player has the possibility to move and rotate certain walls with the right controller to open up new paths in the labyrinth (see figure 1). The left controller can be used to interact with items (see Axel Bauer documentation). In this game, the status bar is not fixed continuously in the field of vision, as is the case with video games on monoscopic screens. To give the player a more real feeling, the status bar was attached at the level of the left controller, similar to a wearable. The status bar, consisting of a health bar and a bar that counts the collected coins, can then be displayed by clicking on the touch-pad of the left controller (see figure 1). In this way, necessary information can be requested when it is needed and does not clutter the player's field of view. As the health bar is not always displayed, a small animation has been added which starts pulsating in the field of view when the health value of the player has fallen below a certain value. In addition, there is an audio signal by means of a heartbeat that becomes louder the lower the player's health is.

Fade-in and fade-out animations were created for the transition between the levels when passing through the portal in order to make it easier for the user to change scenes. As already mentioned, in addition to the normal walls in the labyrinth, there are also selected ones that can be moved by the user (see figure 3). Movable walls have a different texture than normal walls, but they also light up when the player is within reach. When the player selects a wall, it glows green. The green texture remains as long as a wall can be placed. If the user wants to position is at an invalid position, e.g. in another wall, the colour changes to red, giving the player a visual signal that this action is not possible. If the player nevertheless decides to place the wall, it is placed at its starting point instead of being placed at the location marked in red.

3.1 Assets

Only a few assets were needed for the part of the game I created. A heartbeat [5] was added as a soundtrack, which was played when the player was injured. In addition to this sound, images of the portal [4] and the status bar [3] were downloaded from the internet. The wall textures [6] and the SteamVR Plugin [2] were provided from the Unity Asset Store.

4 Implementation Concept

This section outlines the implementation of the application and deals with the main components and the logic that runs in the background.

4.1 Technology

The game development platform Unity [1] (version 2021.2.3f1) has been used for the development of this game. For the usage of the HTC Vive the SteamVR plugin has been downloaded from the Unity Asset Store. It provided a default action set which was extended with actions for the selection and rotation of movable walls and the displaying of the status bar. A Steam Account is necessary for the usage of SteamVR and to adjust the controller bindings.

4.2 Classes

4.2.1 Character

The character class serves as the basis of the game and contains the health points and number of collected coins of the player. These values are taken over by means of PlayerPrefs when switching to a new scene. The entire logic of what happens when the player's health values deteriorate or improve is also located here. If the player's health points fall below a certain limit (the so-called painlevel), the player's field of vision begins to pulsate and, depending on the severity of the injury, a rather loud or quiet heartbeat is also emitted as an acoustic signal. Since there are only a few healing items in the game, the player also has the possibility to heal over time. However, this process is designed to be very slow. Status bar instances are passed with which the character class can directly update the contents of the coin bar and the health bar.

4.2.2 Portal

The portal class listens for a collision between the portal and the player. In case of such a collision a transition animation is played before the scene is changed. This happens by reading the current level from a level-array and selecting the following level to be loaded as next scene.

The gameobject component of the portal was implemented by means of a particle system that animates a transferred image in combination with a cube which tracks collisions (see figure 4).

4.2.3 Walls

Two classes are used for the movable walls. WallMover is a class that lies on a component of the player with which the direct interaction with those walls is controlled. The SteamVR actions are passed to the class and methods for selecting and rotating the walls are initialised. As long as a wall is selected, a co-routine is executed which enables the mapping between the player's right controller and the wall. This spatial navigation allows intuitive re-positioning of selected walls in a confined space. When the moved wall is set, the position is checked again. If it is invalid, the wall is moved to its initial position, otherwise it remains where it was re-positioned.

PositionChecker is the class that all movable walls have. Here it is checked whether there is an overlap between the moving wall and other objects in order to give the player visual feedback as to whether the wall may be placed (see wall e and f in figure 3).

[1]https://unity.com

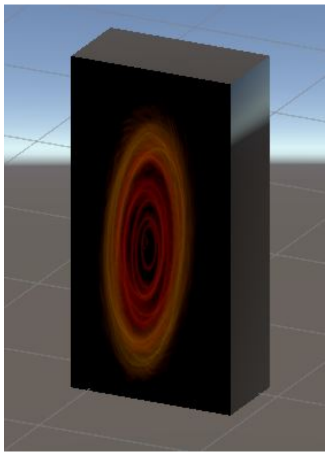

Figure 4: The portal consists of a cube gameobject which registers collisions and a particle system which renders the portal animation (seen in red and orange).

4.2.4 Status bar

The status bar is used to update the values of the (in this case) health bars and coin bars. There is a method that updates the values that is called directly in the character class. This class is placed on each of the value bars. The StatusbarPositioning class, in turn, takes care of positioning and displaying the bars using controller input. The game object of the left controller is passed to the class so that the position of the canvas in which the values are located can be aligned with its position.

5 Challenges

The implementation of this project brought many challenges. The biggest one turned out to be changing the scenes. When changing scenes, the controllers were no longer tracked, even though the SteamVR Player component was selected to be destroyed. The problem turned out to be that this happens when you use a self-defined action set for the interaction with the controllers. So we switched to extending the default action set of the Vive controller. Another challenge was developing without a headset, so Axel and I spent a lot of time in the MRS B room at the University of Applied Sciences Upper Austria.

6 Conclusion

The implementation of this virtual reality game was a great experience and in some situations a great challenge. On the one hand, it was great to deal with my own project and work on the implementation of my own ideas. On the other hand, the teamwork gave me the opportunity to work with a motivated visionary from the fields of media technology and design. Unfortunately, the use of the HTC Vive tied us to the hardware in Hagenberg, which also led to many long days in Hagenberg during the lockdown.

The final product of this course is a small but nice game which allows the user to dive into a labyrinth, explore dark corridors, escape from monsters and search for hidden coins.

Acknowledgments

Many thanks to my project partner Axel Bauer, who gave the game a special touch with his drive and approach to developing an innovative game concept.

References

[1] CodeParade. *Non-Euclidean Worlds Engine*. Youtube. 2018. URL: https://www.youtube.com/watch?v=kEB11PQ9Eo8.

[2] Valve Corporation. *SteamVR Plugin*. Unity Asset Store. 2021. URL: https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647.

[3] Bimzy Dev. *How to make a Health Bar in 5 minutes*. Youtube. 2021. URL: https://www.youtube.com/watch?v=FQNZwcd6FaY.

[4] John3. *red portal - portal 2 orange portal PNG image with transparent background*. TopPng. 2019. URL: https://toppng.com/free-image/red-portal-portal-2-orange-portal-PNG-free-PNG-Images_205988.

[5] newlocknew. *Heart beat.Strongly pounding.Blood flows in the veins(6lrs)*. FreeSound. 2021. URL: https://freesound.org/people/newlocknew/sounds/608243/.

[6] A dog's life software. *18 High Resolution Wall Textures*. Unity Asset Store. 2016. URL: https://assetstore.unity.com/packages/2d/textures-materials/brick/18-high-resolution-wall-textures-12567.

[7] Evan A Suma et al. "An approach to redirect walking by modifying virtual world geometry". In: *Workshop on Perceptual Illusions in Virtual Environments*. Citeseer. 2009, pp. 16–18.