Ecole Polytechnique Fédérale de Lausanne

CS-444 Virtual Reality
Lecturer: Dr. Ronan Boulic

VR - Redemption

Group 15
Zhendong Li (328874)
Yuxin Wang (338745)
Liangyong Yu (322546)
Mingchi Hou (321880)

May 2022

1 Synopsis

Our game, Redemption, is an escape room type of game. As soon as the game starts, players will find out that they are in a spooky haunted house, and the main goal for them is to find clues and keys to escape. An image illustrating the theme of the game is shown below:

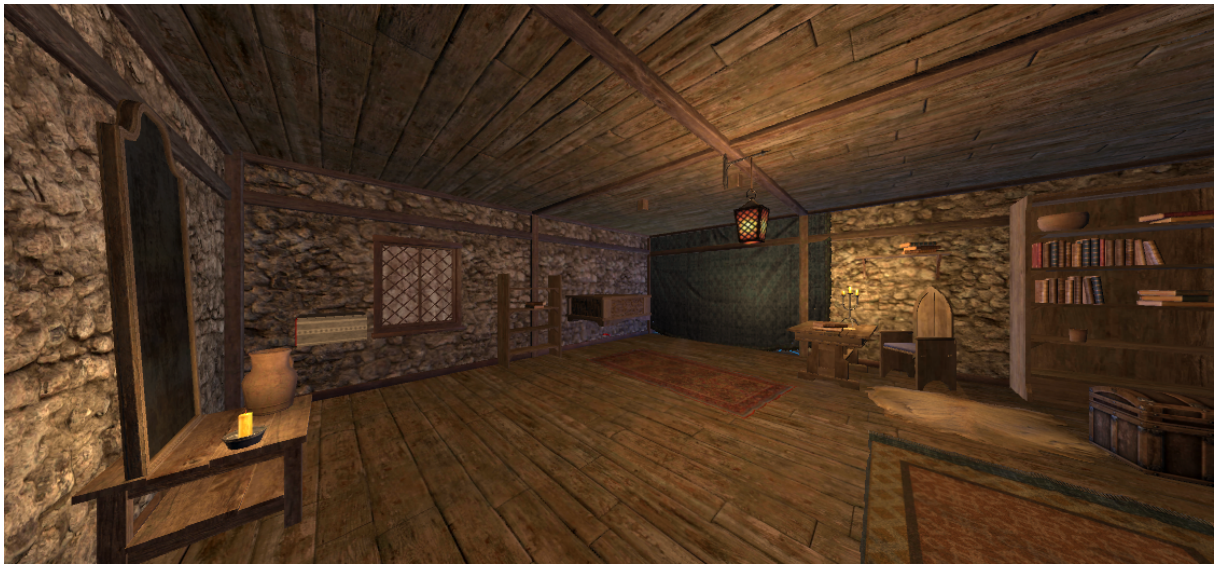

Figure 1.1: An illustration of the theme of Redemption

This game world is created in a way such that it is as realistic as possible, allowing players to have a good sense of presence. In auditory aspect, the background music, which is present during the whole game, enhances the horror atmosphere. We have also added sound effects to give players a more natural, and better experience. In visual aspect, we have included fire burning and smokes.

Furthermore, in collecting clues, players continue to uncover new tasks. In this process, players will keep their curiosity. Thinking and piecing together information to start a new task can also give players a great sense of achievement.

In addition, the difficulty of the game can be adapted due to the decisions of the player. If the player is not as confident or stucks in a certain room, then they can move around, and then pivotal objects will be highlighted. If desired, the player can then get more hints regarding how to interact with the object.

Last but not least, we have also implemented the Magic grab and Teleportation to prevent the player moving for a long distance to reach the object or a specific location.

2 Scenario and interactions

This game consists in total four rooms. Once the player finds the correct clue or key, the player can enter the next room. However, later room may use the clues from the previous room as well. In the end, when the player find the final key, in order to end the game, the player return to the first room and open the door.

2.1 Interactions

2.1.1 Required Interactions

Below is a list of all the required interactions, followed by detailed description of the schema:

Interaction	Controller commands
Grab	Get close to the object and press **A**
Grab inside the container	Get close to the object and press **B**
Magic Grab	First **Hand** triggers, then **Index** trigger, both hands work
Teleportation	Use Index and **A** or **X**
Throw	Release buttons pressed and throw the object with some force
Hit	Use right **Index** and **B** to destroy object

Table 2.1: List of all commands for interactions

- **Grab:** Not all the objects in the scenes can be grabbed. Instead only the objects that are set to rigid body, and ObjectAnchor can be interacted by the player. To perform this interaction, the player needs to be close enough to the center of the object and press A. The grasped object will be attached to the handle, which means the object moves corresponding to the handle. When A is released, the object will be detached from the handle, and the initial velocity of the detached object is the same as the velocity when the object leaves the handle.
- **Grab inside the Container:** If an object is inside a container, our grabbing method is difficult to distinguish the container and the object inside, because both of them are graspable. In order to fix this issue, we set another button B to grasp the object. Unlike button A, B omits all container objects. Additionally, we set a flag *isContainer* to distinguish containers and non-containers in scripts.
- **Magic Grab:** The scope of objects is the same as indicated above. You first need to point the ray at the desired object and then when this interaction is triggered, the location of the object will be updated to that of the respective hand anchor.
- **Teleportation:** "Ground" and "Carpet" objects are set with **Ground.cs** so that when a ray hits that ground, and then by pressing A or X depending on left or right controller, the player can be teleported to the location which the ray hits.
- **Throw:** Swing your arm which holds the object using some force at the same time when releasing the buttons, the object will fall due to gravity and the arm force.
- **Hit:** When pointing the ray at some hittable object, which is implemented by particle system then press B: the object will be destroyed and it will display some smoke effect.

2.1.2 Architecture of The Game

There are a large quantity of objects to interact in our scene, and many events to be triggered by the interactions of objects. We define two basic classes to represent objects and events. Abstract class **SignalObject** defines the basic structure of the object. This kind of objects, for example, a key, can trigger events, for example, open a box. **Event** is also an abstract class and defines the basic structure of events. All actual events derive from this basic class. **SignalObject** has an event list, and will trigger them when some user-defined conditions are satisfied.

There is an example: We create a class **CollisionSignalObject** which derives **SignalObject**, and attach this class to a key. And create a class **OpenBoxEvent**, and attach it to a door. In **CollisionSignalObject**, the list of events is **OpenBoxEvent**. We set the condition that if the key gets touched with a specific door, **OpenBoxEvent** of this door is triggered, which opens this door.

We also create a class **ObjectAnchor**. Any object which owns this class can be grasped. When an object is grasped, the object changes its parent to the handle instead of the scene, and disables its rigidbody; when the object is released, the object restores its original parent and enables its rigidbody.

2.1.3 Other implemented Interactions

Below is a list of other interactions implemented by us:

Interaction	Controller commands
with UI buttons	**right index** trigger
Get Hint	Get close and press **X**
View instructions	press **X** and **Y**
Audio	play automatically

Table 2.2: List of all commands for interactions

- **Press UI button:** we nearly just base on the functions provided by the UIHelper prefab in Unity.
- **Get Hint:** The objects close to the player will be highlighted in a shallower color (this might not be very obvious for some object) and by pressing X, some hint of the closest (this logic is similar to the **Grab**) object will be displayed.
- **See Instructions again:** By pressing X and Y together, one can review all the instructions of possible actions on a UI canvas which follows the camera.
- **Audio:** Audios contain the footstep, background sound, object fall sound, ray sound, door sound, box sound, stone gate sound and fire sound. They are triggered automatically by the action.

2.2 Scenarios

We will explain the details of our scenes in the following subsections. Note: the name of the object we mention is the same as the hint displayed on that object in the game.

2.2.1 Tutorial

The tutorial is built in a separate scene with a big UI canvas. In fact, we place the player right inside this separate scene when enters the game. If the player already knows the commands, he or she can skip this tutorial section and starts the game directly. There are several pages of the UI. Player is able to read through all by interacting with the UI buttons. By clicking the **START GAME** button, the player will be switched to the actual game scene. In addition, in case the player forgets the instructions during the game, by pressing X and Y together, one can review all the instructions again just like we mentioned in the interaction section.

2.2.2 Room 1

This room has multiple objects to collect and trigger. The player needs to collect a candle and use it to burn the green curtain (which is besides the bed). And a fire effect, which is implemented by particle system, will be displayed. Then the player can enter **Room 2**. However, there is still some work left in Room 1 in order to open some later room: (1) the player needs to destroy the bed, and find a key under the bed; (2) then the player can use the key to open a box which has the same color as the key; (3) then the player gets a battery for later use.

2.2.3 Room 2

There is a coded lock as a wall in this room. The player should use the long stick to touch the number. Then the number touched will be entered and displayed on the screen which is on the floor. Touching "C" clears the inputs and touching "E" checks if the input is the correct

password (which is 1234). If correct password, the wall goes downwards and the player can enter **Room 3**.

2.2.4 Room 3

There is a table which gives the hint "out of electricity". So the player needs to fill the battery from **Room 1** into the container on the table. Then the player can enter **Room4**.

2.2.5 Room 4

This room only has one task to do. A key is hidden inside the ball in this room. As the hint says it is "fragile", so the player can break it by crashing it onto the floor with some force. And then the player can pick up the key to the final door in **Room 1**!

3 Playtesting

While we were still developing the game, we asked friends to try out. We asked them to freely try to escape the room, and when they were stuck, we provided verbal instructions that later became the written hints. Overall, testers expressed excitement after solving the tasks and entering in the next room.

During the playtesting session inclass, we improved a lot on both the tutorial and the hint system based on the feedback from testers. For instance, we added the hint about how to enter the password in Room 2 with the rod lying on the floor due to the fact that none of the testers find the usage of the rod without being instructed to do so. We observed a general trend that most players reported cybersickness or have to remove the HMD for a while after 5 - 6 minutes of playing. One of the player suggest that we can choose to rotate around guardian so as to avoid moving after rotating with the joystick. Overall, the players thought our game is interesting and the ideas are cool. They did appreciate the fact that having some misleading extra objects add fun to this game.

4 Acknowledgements

We would like to give our special thanks to Dr. Ronan Boulic, and our Teaching Assistants, who gave us not just theoretical knowledge and technical support, but also ideas for a more playable and intriguing game. We are also grateful for our friends who tested our games during different stages of the development. Then we would like to express our gratitude for our fellow classmates, who provided valuable feedback and suggestion for debugging during the playtesting session. Last but not least, we thank our parents who support us for our study at EPFL.