École Polytechnique Fédérale de Lausanne

Virtual reality

CS-444

# Rocket Man

Project Report

*Authors* : Group 18

Fantini Elia
Lodetti Gianni
Lonneux Olivier

*Professor* :Boulic Ronan

May 25, 2022

# Contents


# 1 Introduction

The game *Rocket Man*, realized in the frame of the course *CS-444 Virtual Reality*, takes you into an apocalyptic world. The goal is to escape the earth thanks to a rocket that you will first need to repair and set up through various riddles. You can move by walking, teleporting, or by using the joystick of the controllers. To solve the riddles, you have to use various tools that you can pick up using your hands or through magnetic grabbing.

# 2 Synopsis

As the world is undergoing an apocalyptic extinction event, you find yourself within a spaceship/rocket with an opportunity to escape earth. A tablet in the room informs you that a nuclear bomb is coming towards your town. Earth is not a welcoming planet any more, therefore you need to escape aboard the spaceship. Unfortunately, it has some problems turning on. Through various riddles and using your logic and intuition, you will have to make it work. After you finish repairing it, the spaceship will finally escape earth. However, debris hit the rocket in space, and you need to repair the spaceship if you want to survive. You try to fix it, but you cannot do it and have to escape thanks to an emergency capsule. The game end as you get a glimpse of a new planet.

# 3 Scenario

You're in the machine room (first floor).There's a tablet on the table, on the back of it there's a code handwritten with a pen. A wall has stairs with a trap door on the top, but it is closed. There is also a trap door on the floor, closed by 2 screws. On the wall you can see some disconnected wires. There's also a pc on a big desk and wire hanging on its right, unplugged. If you plug it, three screens turn on: there is a schematic figure with the same colours of disconnected wires. If you realize that they are the same colours, you can a colour pattern that will tell you how to connect the wires. That turns a red button on the desk into green. If you try to press it you see that the traps does not open: the button has a golden frame and the very same frame is on a small statue on the sofa, putting it on the button will open the door.

You climb up to the living room (second floor). You see a safe in the room, but it is close with a screen on it. Under the bed there is a UV torch and pointing it at the screen of a tablet you see fingerprints doing a special sign. If you do the pattern with your finger, you unlock a box that falls from the ceiling: inside iy there is a hammer and a drill.

You return to the machine room and with the drill you just found you open the trap door on the floor by unscrewing the screws. Inside, there is a golden bolt you need to screw using a wrench. There is one in a safety box on the wall, covered by glass. You can use the hammer to break it, and with the wrench you can now screw the bolt in place. The engine then turns on , creating smoke and noise, and if you go back to the living room you can see that the control room's trap door opened.

You climb up to the control room (third floor), on the desk, there is a numeric pad with 3 numbers. The colour of the LED above the numbers change when you change the numbers. As you turn around, you see a combination of colours on a screen, and a series of ones and zeros below it. Once you changed the number until to get the three corresponding colour, a green light lit up above it. Next to the numeric pad there is a series of switch: by switching up the corresponding switch to '1' and keeping the others at '0', as in the screen behind you, the green light above it lit up as well. There is also a lever on the desk, blocked until you solved the two previous riddles. You can now move it and the rocket launches.

A big impact wakes you up, you are in the control room, somewhere in the space. In the machine room, there's a fire, using the fire extinguisher from the control room you can turn it down. However, the rocket cannot function any more. You need to escape. In the control room there is a painting, it is too high to pick up, but you can throw an object at it to make it fall. On the painting there is a photo of the living room with the painting on it hence you understand that you have to put the painting back into its place as it is in the photo. Done this, the tablet on your left will light up, telling you to touch it, then the wall in front of you will slide down, showing a door with a numeric pad. You put the code behind the tablet you found at the beginning and the door opens: entering the emergency capsule, you press a big red button. A launch countdown starts, and you see the new planet in front of you, getting closer and closer.

# 4 Interactions

## 4.1 Grab

### 4.1.1 Grabbable objects

Most of the objects in the environment can be grabbed with the hand using the index trigger. This was done by attaching the OVR grabbable script from the Oculus package to the objects.

### 4.1.2 Magnetic grabbing

The code for magnetic grabbing can be found in the "customGrab" script. We started by extending the hands of the player with a lineRenderer. The script then renders a line whenever a ray intersects with objects in the "Grabbable" layer, and that object becomes the "ItemInFocus". If the player squeezes the trigger, then we set the transform parent/position/rotation of the "ItemInFocus" to that of an entity that we stored in the public SnapPosition array with the same tag as the "ItemInFocus". (These entities with customPoses that we stored in the SnapPosition array can be found in the right/left HandAnchors of the OVRPlayerController)

## 4.2 Locomotion

### 4.2.1 Walking

Walking is the default walking with the joystick, already present in the OVRPlayerController script.

### 4.2.2 Teleporting

Teleportation is made by calculating the trajectory of a parable, taking some vertices of the curve and casting a line passing through them. If the line collides with an invisible floor belonging to the "Teleportable" layer, the line is rendered and shown to the player, with a marker on the ground where the line collides. Whenever the player releases the joystick, OVRPlayerController is moved to the position of the marker.

### 4.2.3 Climbing

To implement climbing we added scripts in Util/Climbing folder. The Hands script is given to to each hand, and handles collisions of the hands with objects containing the "ClimbPoint" tag. When the user presses the trigger it uses a reference to the OVRPlayerController script to notify it that the user is climbing by setting currentHand = "the hand that the user is using to climb". The OVRPlayerController was then modified to allow vertical movement, disable gravity and update the Controller.Move() with

the currentHand.Delta * sensitivity while currentHand != false. When the player releases the trigger currentHand is set to null in OVRPlayerController by calling the clearHand() function added to OVR-PlayerController script.

## 4.3 Riddles

### 4.3.1 Wires

The wires came with a package containing two scripts: CableComponent and CableParticles which are attached to the wires and take cares of the rendering and animation of the wires. Two additional scripts were written: AttachWirePlug and WiresRiddlesController. The first one is attached to the plug to handle the plugging of the wires. When a wire collide with a plug, it forces release the wire of the hand to plug it in by changing its position. After plugging in the wires, it informs WiresRiddlesController.

WiresRiddlesController is the controller of the riddles. It is used to turn on the screen with the solution of the riddles when plugging the power socket. It also checks if the wires are correctly plugged with the method for the riddle. It then calls on the ButtonTrap class to turn the button green and allow the trap to be opened.

### 4.3.2 Opening the trap door with the button

Two script are used for this interaction : ButtonTrap and TrapDoorController. The first one is attached to the button used to open the trap door. The button need to be enabled from WiresRiddlesController. Which turn it green. Using collider, the script checks if an object is colliding with the button and looks at the tag to see if it is a button presser (the statue). If yes, it plays a sound and open the trap door by calling on the TrapDoorController class. It also looks at collider exiting to close the trap door if a button presser was pressing on it and is not any more. The button is also moved downward and upward when pressed or unpressed. The trap door is an object from an external package which open and close through an animator. The TrapDoorController class changes a boolean in the animator controller to trigger a state change.

### 4.3.3 UV light and pattern lock

There is three scripts linked to this riddle. The UVLight class is attached to the UV lamp torch and update the game state when the light is grabbed for the first time. It was previously also used to turn the light on and off, but this interaction was removed (see section 6). The detection of the light is done with a capsule collider set on trigger which has the shape of the ray of light. The patternLock class then changes the material of the screen to make the Z appear when it detects this collision. There are 5 colliders on the screen which needs to be hit in the right order. The screenCollider class is attached to each one of them and is used to check if the collider are hit in the right order for the Z pattern drawing. If the pattern is drawn correctly, it calls on the patternLock class, which make the tool box fall. This is done by switching a non-grabbable, kinematic toolbox with a grabbable, non-kimematic one. This script is also used for the painting riddle.

### 4.3.4 Falling box

The box contains both the drills and the hammer. The class toolBox check for all objects entering or exiting the box using a collider on trigger. FreezeObjectsInside freeze all objects inside the box by making them kinematic.

### 4.3.5 Removing screws with the drill

The screwDriving class turns on the drill by rotating the insert, it adds sound and vibrations. It also detects if the insert is colliding with a screw and rotates it if the drill is turned on at the same time. After some rotation, the screw pops out of the ground and becomes non-kinematic. PlateRemoval continuously checks if the 2 screws have been removed and if so, it opens the trap door by changing the boolean of the animator.

#### 4.3.6 Breaking the glass with the hammer

The hammer is a grabbable object with the "hammer" tag attached to it. The breakable window comes from a package.The script BreakWindow triggers the script of breaking simulation taken from an external package only when an object with the "hammer" tag collides with it. This script also switches the non-grabbable wrench with a grabbable one, as with the toolbox.

#### 4.3.7 Turning the bolt with the wrench

The wrench class takes cares of detecting when the wrench touch the bolt with colliders and rotates the bolt when the wrench is rotated and on the bolt. When the desired angle is found, meaning the 2 red line are aligned, the bolt cannot be rotated any more and the trap door for the third floor opens. This is done with the RedTickColliding class, which uses colliders on the 2 red tick to check when they collide and calls on the wrench class. Such event instantiates a particle system object to simulate smoke.

#### 4.3.8 Numeric pad with colours

The ButtonTablette class delegates the 2 arrows button listener to the TabletteThirdFloor class, which increments and decrements the number displayed on the tablets according to which arrow has been pressed. It takes as input the name of the button gameObject from ButtonTablette to know which arrow has been pressed. TabletteChallengeThirdFloorController class takes care of changing the LED colours according to this number. It also checks if the riddles has been successfully complete. If so, it updates the game state and lit up the green light.

#### 4.3.9 Switches

The SwitchControl class animates the switch when they are hit by the hand. It informs the SwitchChallengeController class, which looks if all switches are on or off according to the correct pattern written on the wall. When the riddle is complete, it updates the game state and light the green light on.

#### 4.3.10 Lever

The JoyStickControl class checks if the lever collides with the hand and moves it to its final position if all previous riddles have been completed. StartRocketChallenge checks for previous riddles to activate this lever. When the lever is activated, it changes a public boolean which starts the launching animation. This animation moves the external objects (the city) down with a vibration effect and instantiates a particle effect to simulate a growing sense of speed. Finally, the GameManager loads the RocketInSpace scene.

#### 4.3.11 Extinguishing the fire

The fire extinguisher needs to be grabbed for the FireExtinguisher class to create particles which look like foam when it is activated by the index trigger. It uses particles trigger to check if the fire is being extinguished and stops it after some time. When this happens, the game state is updated and the fire and alarm sound effects stop.

#### 4.3.12 Painting

The screwForPainting class checks for collision with the painting: when it collides, it calls a function on the painting class which takes care of placing it at the right position. This class also makes the painting fall by unfreezing it when colliding with an object that was thrown. Then the patternLock class is called to change the pattern lock screen. It detects when this screen is touched and slide the wall down. If the fire wasn't extinguished, however, it displays a warning which tells the player to stop the fire.

#### 4.3.13 Keyboard and emergency capsule

The KeyboardButton class delegates the buttons' listener to the Keyboard class, which takes care of updating the display with the entered number, it also uses the gameObject name as an indicator of the button pressed. It also checks if the right number was entered to open the capsule door using the CapsuleDoor class. Then for the second door, it is the DoorOpener class which uses the door animation to open it when we press the corresponding button.

### 4.4 Main controllers and tutorial

The game logic is mainly handled by the individual riddles' scripts, since solving them gives access to areas or tools without which it would be impossible to progress. Nevertheless, a GameManager class saves the current game state to avoid any irregular progress in riddles and to handle the scene changing. The game state is also used by the SoundManager to play hints. In fact, in addition to the initial floating tablet showing an interactive presentation, explaining all main controls of the game, we recorded an audio hint for every game state the player can be in, so that he never gets stuck. The SoundManager handles the background music as well, and it collaborates with the AnimationManager to reproduce the two launches animations.

## 5 Aesthetics and immersion design

The game was thought to be played on a portable device such as the Oculus Quest, so we decided to use only low poly assets that could easily be rendered without a drop in the frames per second. That helped us also to find much more assets from the web with a compatible graphic style, creating an environment which is thematic consistent. In fact, the game uses nineteen different external assets' packages, but the whole scenes were made by placing manually every single prefab, recreating a rocket environment as we imagined it since the very beginning, without using any pre-made scene.
To improve immersion, not only we tried to make interactions as natural as possible, but we also provided each of them with a proper sound effect and some haptic feedback. The music in the background was chosen to be relaxing while the player is reasoning on riddles. A big part of our attention was put in the rocket and capsule launch animations, so that they could convey as much as possible the emotions and feelings of a real launch. For them, every sound and particle effect was carefully designed.

## 6 Play testing and feedbacks

The feedback we got from the play test was very valuable, as it allowed us to see what parts of the game were not very intuitive. For example, we noticed that the flashlight on the second floor was too hard to find, and people would forget they could interact with items with a button, so we decided to have the flashlight on by default instead of toggling it with a button.Moreover, many testers said that the tutorial was long and not intuitive due to the amount of text. Hence, we replaced most of the text with videos showing the interactions. Some people also forgot some parts of it, so we introduced an animation at the end of it that moves the tablet showing the videos on the wall, where it will always be available. We also had other minor comments, such as missing sounds indicating success or failure in some riddles, so players didn't know if they had successfully completed the riddle or not. Every comment or minor bug encountered has been noted down and solved. Other than that, every participant said it was a funny and immersive experience, without any kind of motion-sickness except for a player that felt slightly destabilized by the climbing.