University of Kassel

Graphical Simulation

Final Project

Ferdinand's Lost Money

Author:
Altar Tosun *(35605160)*

Lecturer:
Prof. Dr. Dieter Wloka

July 30, 2020

UNIKASSEL
VERSITÄT

Graphical Simulation Final Project
Ferdinand's Lost Money

Altar Tosun

Universität Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
altartosun06@gmail.com

1 Introduction

Graphical Simulation lecture's final project is to make a VR escape room game. My puzzle/escape room game is about to find the old man ,with Alzheimer's disease, Ferdinand's lost money in his apartment. He hid the money somewhere in the apartment but because of his disease, he forgot the place where he hid the money. In this game player will solve the puzzle step by step and in the end of the game player will find the hidden money.
Students had two choices between HTC Vive and Oculus Quest. I decided to use Oculus Quest in this project because, Oculus Quest is more innovative than HTC Vive. There is no need of any laser sensor around and still very accurate and headset is much lighter than Vive.
This was my first VR game development experiment. First of all, I watched and read some tutorials, while I was following the tutorial made a small VR game to explore and test the environment and then started to develop the escape room game.

2 Pre-Development

I started to learn how to make a VR game with the suggested YouTube videos. First thing was about setup of VR integrity to Unity. My Unity version is *2019.03.0f6* for this project because this was the suggested and stable version for VR games when I started to develop. Later on I activated the Virtual Reality Support setting under the Player Settings, XR Settings and selected the Oculus. This will make the game optimised for VR and Oculus environment. When I am done with the Unity settings, I imported the Oculus Integrity from Unity Asset Store. This brings default avatars, assets, scripts and models that will be very helpful later on. Finally I set the Unity Project Settings to Android because Oculus Quest's executable file as APK and set Texture Compression, on Build settings, as *ASTC* because this is what Oculus and Unity recommend.

Player Controller I used *OVRPlayerController* is a prefab that includes player controller scripts, camera, camera scripts and collider. I set the Tracking Origin Type to Floor Level on *OVRCameraRig* (an *OVRPlayerController*'s subsection) and set the camera rig option. Oculus Quest detects the floor and expects to draw a safe circle from the player. Setting the floor level mode, makes it use the floor level as a root for the players height in the game and will follow the player's move for example crouching. Player can walk with the joystick on Oculus Touch Controller. I wanted to add the "teleport" mode as well but I couldn't do it because the closure of the labs.

Fig. 1: Unity quality settings for the project

Player controller in the game is working just fine so it is time to add the hands. I wanted to use normal grabbing system, not the distance grab which makes the player able to pull objects to player's hand from a distance. I used *CustomHandRight* and *CustomHandLeft* ,comes with Oculus Integrity, in the game. I set their *Parent Transform* as Tracking Space of the camera rig from player controller. Thus hands

will follow the player. Hands had opaque texture but, I replaced it with the transparent one and its colour to orange.
I added some cubes to the scene, added rigid body and collider components but be able to grab these, I attached them the *OVRGrabbable* script. Thus player is able to grab and throw the cubes. Setting Collision Detection as Continuous Dynamic on the cubes rigid body component, makes their physics work better.

FPS and Timestamp At the beginning, FPS value of the game was low - around 20 FPS - and this caused headache and stomachache sometimes. FPS was low because Oculus Quest uses its onboard CPU so it is not powerful as PC's. I had to set quality settings to "Medium" or "Low" for Android devices(Oculus Quest uses Android built) and changed the Timestamp settings to 0.01388889. Oculus Quest's headsets screen has 72 Hz refresh rate so timestamp should be 1/72. I had to set the lights as well and finally I got around 45 FPS. It makes the game feel smooth and no more headache caused to player. I wanted to use post processing effects in the game but I could not use because it had a negative impact on FPS. Post processing is ready to use in the game but it is inactive. The other thing to improve FPS is baking the light maps. There was a baked light map already but It was too dark for VR so I changed and add some light sources and baked the lights again.

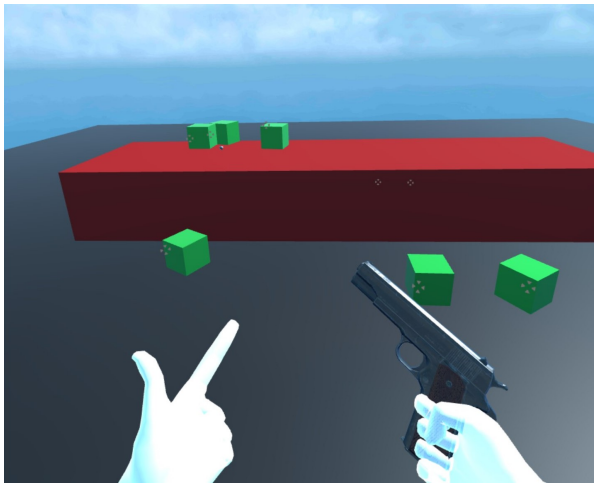

Fig. 2: My first VR game attempt

3 Studio Apartment

The Modern Studio Apartment is well designed, modelled and textured asset that given to us but there were some points that I wanted to change. For example, kitchen had no intractable points, player can not grab and interact with anything. I made up a new kitchen with the all free assets from Unity Asset Store. With the new kitchen, player can open the fridge, cabinets etc. by grabbing their handlers.
I realised, there is no bathroom in this house. I wanted to build a bathroom for this house but I had to change the whole Light Block at the outside of the house. I disabled the old one and created a new Light Block.
Everything in the room was using *Mesh Collider* and this caused some "passing through" problems. To solve that I changed every furnitures colliders to box collider and set them on size. Also walls had *Mesh Colliders* too so I added box collider component as well.
Later on created the bathroom with whole free assets from Unity Asset Store. Now, the apartment has a bathroom.

I added some new furnitures for free from Asset Store and made some colour changes, for example wall's colour and lampshade. I believe these changes made the room look on better way. I removed the reflection globe because it was a FPS consumer too.
Oculus Quest's executable file is APK so I had to change the build setting to Android. Baked lights map are removed because of this process. I had to create some new light sources in the room. On medium settings, as I mentioned above, light per pixel is one so I made these changes according to this.

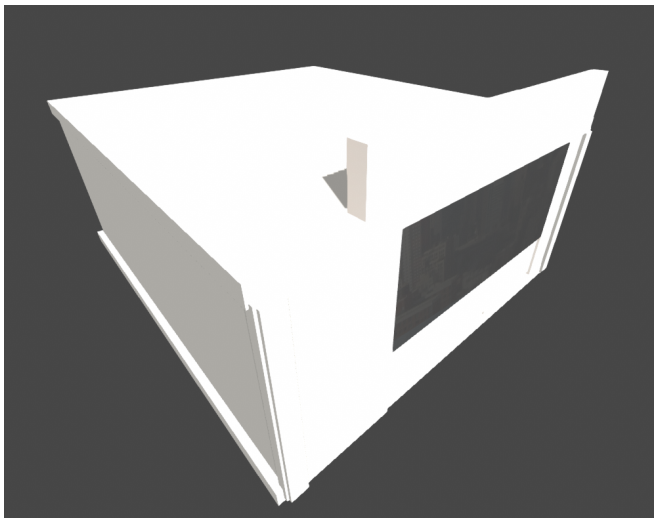

Fig. 3: New Light Block

Fig. 4: Bathroom

Fig. 5: New kitchen

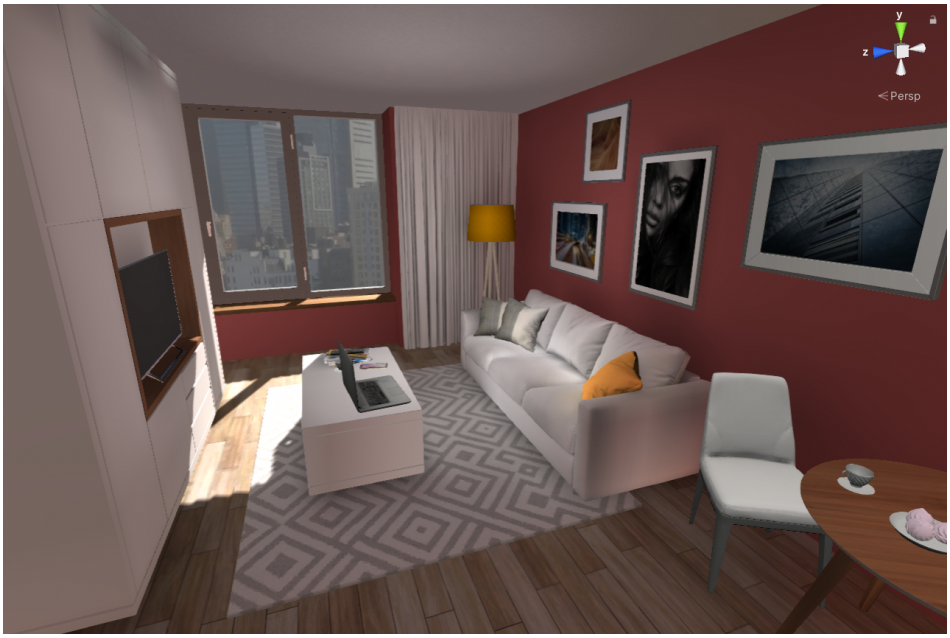

Fig. 6: Living Room

4 Grabbable Objects

To make objects grabbable it is needed a Collider and *OVRGrabbable* script which is comes with Oculus Integrity. It is usually simple as attach the script with the object and it is a grabbable object right now. It is possible to grab, throw and catch etc. but when it comes to open a door or a cabinet door, some things more needed.
To solve this problem at the beginning I created a cube called *Handler* and put this onto the door's handle and attached this with the door using Fixed Joint afterwards, I added the *OVRGrabbable* script to that. It looked just right but when I tried with VR, it did not go as I expected. Problem is when I pull the handler on X-axis door opens a bit but no more because I use force only in one direction, force on Z-axis is necessary as well to open the door. It is possible to open the door but It is not smooth and feels bad.
As a solution I created another cube as "Handler" and It is called "Grabbable Handler". These handlers will work together. Player will be able to hold the "Grabbable Handler" and move this, "Handler" will follow that and of course door will follow the "Handler" as well so door will open, this is the idea behind it. First of all, I created a script called "follow Physics", it is a one line code that make the object follow the other one's position. "Handler" has the "follow Physics" script and will follow the "Grabbable Handler". Now It should work just fine but, It did not again. The problem was when player releases the "Grabbable Handler", it goes back too fast and hits to door so it was closing backward. I should have to set the handler's velocity to zero for each frame.

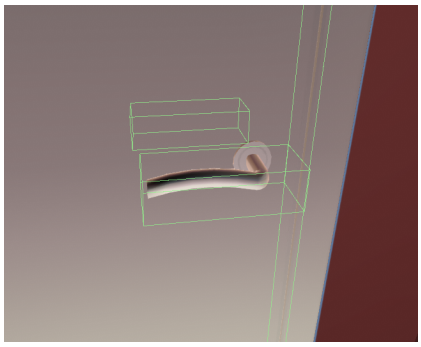

Fig. 7: Door grabbable handlers

I created another script called *Door Grabbable*. In this script, *DoorGrabbable* script inherits from *OVRGrabbable* script. I overrided *GrabEnd* and *Update* methods. In *GrabEnd* method what I do is set the Grabbable Handlers position to Handlers position on release because, they have to be the same position always. The second thing that I do in *GrabEnd* method is setting the Grabbable Handlers velocity to zero. In Update method what we do is if the player pulls the Grabbable Handler too much, this handler will go back to the Handlers position. The second thing is setting the velocity to zero for each frame.
These open-able or turnable objects have Hinge Joint component to give them a anchor point to turn around and it is possible to limit their moving angle.

5 Game Start Screen

On starting of the game there is a short story of the game and the player's goal is written down. Our character is in the game has Alzheimer's disease so he forgot the place where he hid his money. Before the player pushes to Start button in game, it is not possible to walk but player can turn his/her head and look around. I did this in a different way, I put obstacles around the player to limit its movement. Normally, I should do that as script but *OVRPlayerController* script did not let me add a another variable so I did this solution. This game start panel is optimized for VR. Canvas is using as Canvas render mode as Screen Space - Camera and Render camera is set to *CentreEyeAnchor* on *OVRPlayerController*. Thus, canvas follow the direction that player looks at. I did some of these settings without a testing phase.
The Start button is a world object, not an UI. Thus player can interact with this button by touching to the button.

Fig. 8: Start screen

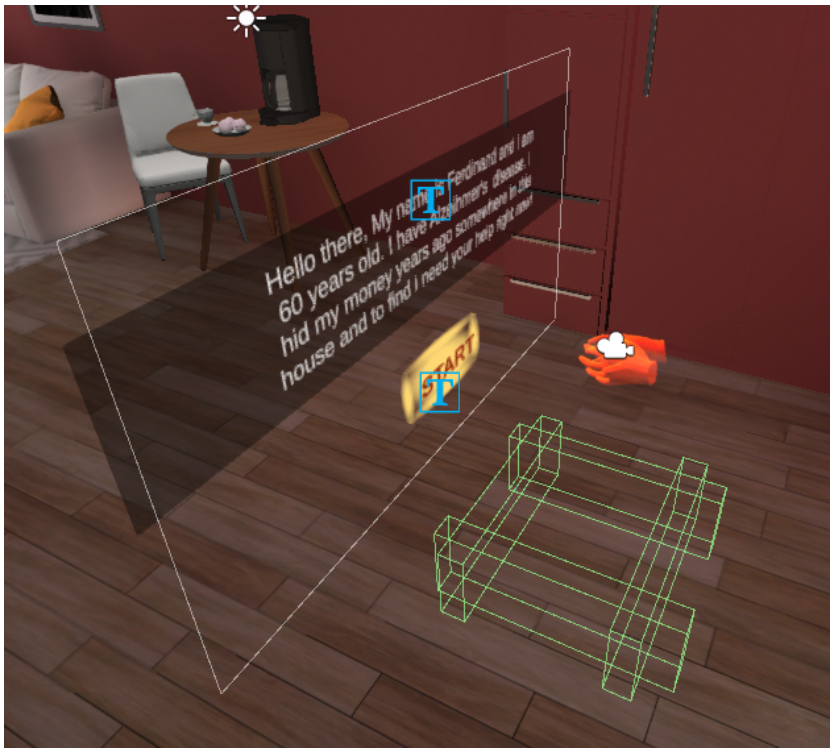

Fig. 9: Obstacles around the player and canvas includes text and start button

6 Puzzles

6.1 TV Puzzle

Player has to turn on the TV for the next puzzle step but there is no electricity because one of the fuse is not in place. The fuse is in the wooden box next to the bed, which can be found in the bedroom. Player grabs the key and place it to the wooden case. I check the item with *CompareTag* if it is the key or not. I made a turning and placing effect using with *Quarternion.Slerp* and *Euler*. I placed a invisible key to its last transform point and used the this as a*transform.position* information. I made the key *static* afterwards because I didn't want to make it able to remove from its place. When the key is placed, the box opens slowly by *Quarternion.Slerp* and *Euler*,110 degree, as well. The fuse will be inside of the box. Player can grab the box and bring it to the fuse box that beside the outdoor door. Fuse has the same smooth placing and moving affect as the key and wooden box door, I used the last point with the invisible game object as transform as well. This will set *true* the *fuseActivison*. Now, TV is able to open. Player can grab the remote controller and can turn on the TV by pressing the red power button with the index finger, while pressing the controllers *hand trigger*. If fuse placed to the fuse box TV will turn on, otherwise a hint will appear - this will be explained later on the paper -. Pressing the hand trigger on VR controller, enables the colliders on players hand in the game. Otherwise colliders on the hands are disabled.

Fig. 10: Opened wooden case and the fuse placed on to the fuse box

Fig. 11: TV is turned on

```
public class LostFuse : MonoBehaviour
{
    public Transform lastPoint;  LostFuse_LP
    public float speed;  "0.1"
    private OVRGrabbable _ovrGrabbable;
    private FuseActivation _fuseActivation;

    Event function
    void Start()
    {
        _fuseActivation = GameObject.Find("LostFusePlace").GetComponent<FuseActivation>();
        _ovrGrabbable = GetComponent<OVRGrabbable>();
    }
    Event function
    void Update()
    {
        if (_fuseActivation.isFuseFound)
        {
            _ovrGrabbable.enabled = false;
            gameObject.isStatic = true;
            float step = speed * Time.deltaTime;
            transform.position = Vector3.MoveTowards( current: transform.position,  target: lastPoint.position, step);
            transform.rotation = Quaternion.Slerp( a: transform.rotation,  b: Quaternion.Euler( x: 0,  y: 90,  z: 0),  t: step * 50);

        }
    }
}
```

Fig. 12: Fuse's script

6.2 Clock Puzzle

The wall clock does not work and we have a digital clock on the book shelf. Digital clock displays the time 12:30. When we set the wall clock to the same time a metal case will come through the wall. I put two colliders on the clock. One is on 12 and the other is on 6. When the hour hand triggers the collider on 12 o'clock and, the minute hand trigger the colliders on 6 o'clock, the picture on the wall falls down and the metal box that in the wall starts to move out. Both minute and hour hands are able to turn while grabbing. I set their velocity to zero per frame as the door because, otherwise these minute and hour hands were difficult to st, when player release the hand, it was still moving slowly. This is my favourite puzzle in this game because it is kinda difficult to solve and when you solve that, picture falls and a metal case comes out through the wall.

Playing Sound Effect in Update() While the metal case is moving through the wall, it makes sounds like rubbing. This sound effect is playing on *Update* method in the script. Playing sound on *Update* method is not the best choice, I assume because, Audio source is trying the sound on each frame. This causes such a bad echo in sound. After my experiments and research I found a solution. I created a variable that called *isPlaying* and set this *true*. Afterwards, I created a *if* statement and its condition is: *isPlaying* is *true* and *AudioSource* is not playing(*false*). In *if* statement I set the *isPlaying* variable false. Thus script execute the *if* statement only once, so echo problem is solved.

Fig. 13: Colliders on the wall clock

Fig. 14: Before and after the solution of the clock puzzle

```
Event function  More
void Start()
{
    AudioSource = GetComponent<AudioSource>();
    _detectHour = GameObject.Find("HourCol").GetComponent<ColDetectHour>();
    _detectMin = GameObject.Find("MinCol").GetComponent<ColDetectMin>();
    isPlaying = true;
}

Event function
void Update()
{
    MoveCheck();
}

Frequently called  1 usage
private void MoveCheck()
{
    if (_detectHour.hourOk && _detectMin.minOk)
    {
        float step = speed * Time.deltaTime;
        transform.position = Vector3.MoveTowards( current: transform.position,  target: lastPoint.position, step);

        if(isPlaying && !AudioSource.isPlaying)
        {
            AudioSource.PlayOneShot(MoveSound);
            isPlaying = false;
        }
    }
}
```

Fig. 15: The Metal Case's script

6.3 Laptop Puzzle

There is a laptop on top of the white table in the living room. It is Ferdinand's laptop. The metal case's password is in this laptop but there is a login password. This password displays on the TV's screen. Player can text that by using the laptops keyboard on numpad or the numbers on top of the letters. It can be difficult to touch the numbers correctly, I could not test that, It can be solved with adding a screen keyboard on the laptop. When logged in, player can see the desktop and in the middle of the screen there is an icon "Do Not Open!". This laptop has touch screen so player can touch to this icon and open. An authorizing screen appears, player needs to verify by the phone. Phone is placed on the coffee table in the kitchen. Player can verify and allow to authorizing by touching the green button on the phone's screen. Then player can see the metal case's password on the screen.

I made the screen changing on the laptop with flags on the each screens script. Each screen has its on script and when an action happens, screen will disable its own mesh renderer and will set the flag to true for the next screen, chain continues like that. For example, when the player enter the password correctly - four digit - and auto detects when four digit entered and check the password is correct or incorrect. If player entered wrong password then script clears the text box and waits for the correct password - screen will set its own mesh renderer disabled and set true the *passCheck* flag, desktop screen script will be enabled when the *passCheck* is set as true.

On the phone I set two different material. One is for the verification screen and the other is default screen. When the verification screen should be activated, script is changing the active material. On the laptop is used different objects mesh renderer but I wanted to try this method on the phone because corners of the phone screen are oval and this way of the solution is better.

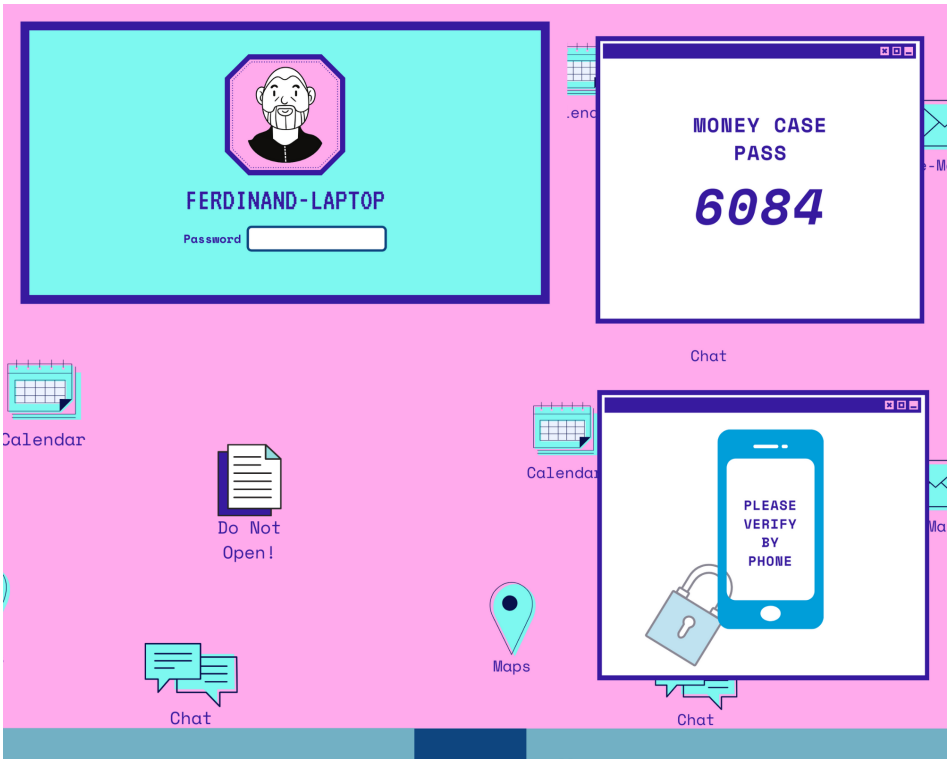

Fig. 16: Laptop's different screen shots

Fig. 17: Phone is screen when authorizing

```
public class PhoneScreen : MonoBehaviour
{
    public Material verifyScreen;  Serializable
    public Material defaultScreen;  Serializable
    private DoNotOpen _doNotOpen;
    private Material[] mats;
    private VerifyButton _phoneScreen;
    private MeshRenderer mr;

    Event function  asset usages  usages  overrides  ext methods
    void Start()
    {
        _doNotOpen = GameObject.Find("DoNotOpen").GetComponent<DoNotOpen>();
        _phoneScreen = GameObject.Find("VerifyButton").GetComponent<VerifyButton>();
        mr = GetComponent<MeshRenderer>();
        mats = mr.materials;
    }
    Event function  asset usages  usages  overrides  ext methods
    void Update()
    {
        if (_doNotOpen.isCicked)
        {
            mats[2] = verifyScreen;
            mr.materials = mats;
        }

        if (_phoneScreen.isVerified)
        {
            mats[2] = defaultScreen;
            mr.materials = mats;
        }
```

Fig. 18: Phone mesh renderer script

6.4 Money Case Puzzle

The mysterious metal case appeared when the player solved the clock puzzle. There is a digital locker on the box. Player can get the password by solving the laptop puzzle, this will be written on the laptop's screen. Player can push the buttons on the locker and enter the numbers, when player pushes the green button, the password checker script works and check the password if the correct password or not. If the password is incorrect player will see the "Incorrect" text on the digital lockers screen and play a "wrong" sound, if the password is correct then the case's door will open smoothly as I did on the other objects as well and play a "correct" sound effect. Pushing the red button makes the player able to delete the numbers if he/she pushes wrongly to the other number. There are two different key sounds I used in this project, they are selecting randomly by the script. Number buttons have the input value from *Serialized Field* on Unity. This made be able to use the same script on every number buttons on the digital locker.

Player can see the money inside the case when it is opened. Normally, when player touch the money a panel will appear and let the player select between "Keep chilling in the room" that will allow the player stay in the game and spend more time in the room or "Exit" that exterminate the game but I could not do this because of the labs closure.

Fig. 19: Money case wrong password and correct password

Fig. 20: Number keys script and on Unity settings

7 Game Walk-Through

It is possible to reach the end of the game in different ways. For example, it is possible to find the hidden metal case before finding the fuse but it is not possible to login to laptop before turning on the TV.

- First the player can go to the bathroom and take the small gold coloured key from small cabinet and then go to the small wooden box next to the bed and open it. There is a fuse in the box.
- Player grabs the fuse and then goes up to the fuse box on the wall and places it in. Now, the apartment has electricity.
- There is a remote controller on the white table that middle of the room. Click to red power button to turn on the TV.
- On the same white table there is a laptop. Turn this on. Player will see a "Sign in" screen. The password is written on the upper left corner of the TV screen. Password is: 0125. Player can type the password using laptops keyboard.
- Then, player can see the desktop screen. There is an icon called "Do not Open". Click to this but to open it is required a phone authentication.
- The phone is on the coffee table in the kitchen. Touch green button on phones screen and now we see another password on the laptop's screen.
- There is a wall clock in the kitchen and a digital clock on book shelf. Digital clock displays "12:30". Player needs to set the wall clock 12:30. To set this wall clock, player can grab hour and minute hands and rotate them. Then metal case will appear through the wall on the left side of player.
- To open this metal case, password is on the laptops screen: 6084. Push to numbers and open the case. Player will see the money inside.

8 Hint System

There are three different hints in the game. First one appears when the player opens the wooden box and founds the fuse. Second one appears when the player gets close enough to wall clock. I placed an invisible trigger collider in front of the wall clock, when the player triggers it, hint will be appeared. Third hint appears when the player needs to authorize by phone. Last hint appears when the player tries to turn on the TV before finding the fuse.
When the hint triggered, assigned script enables *Canvas* and *TextMeshPro - Text* component. Thus, hint gets active for five seconds which is controlled by a timer. Hint's canvas is optimised for VR with these settings: Canvas render mode is *Screen Space - Camera* and Render camera is set to *CentreEyeAnchor* on *OVRPlayerController*.

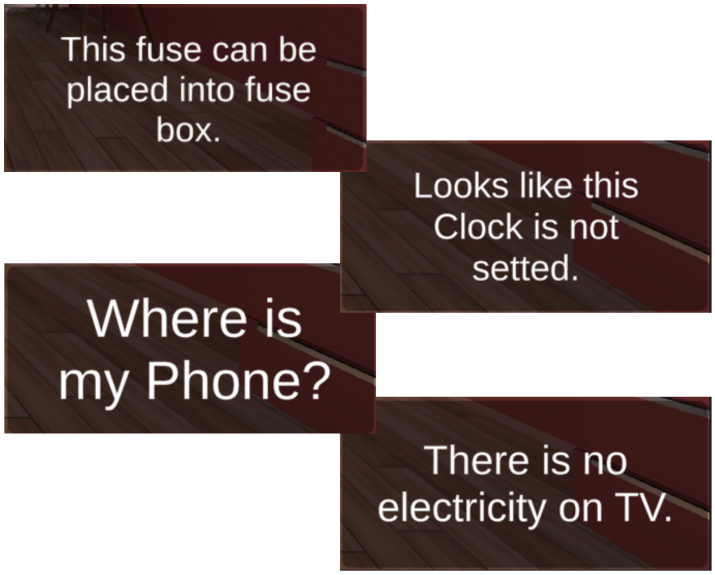

Fig. 21: Hints in game view

9 Future Work

Unfortunately, labs were closed because of the pandemic and it affected the project. I have done the most parts with the project before that because, I went to the lab every week for a month. It was easy to reserve place because, I was the only one who makes the project for Oculus Quest but the time was not enough, unfortunately.
I could not test the Start screen and the hint system because I have made them at home. It is possible retouch the grabbing system and make it more optimised and responsible. I could not make the ending game screen so when player founds the money nothing happens right now. I wish, I could make the end screen and make it possible to close the game after that.
I should change the fuse's model. For now, it is a blue rectangle as a prototype.
I can add the teleport mode for moving around, it can be positive for quality of life.

10 Executing the Game

10.1 Oculus Link

On Unity if you start the game in Unity engine, game just runs without build but on Oculus, Unity has to build the game and send the APK file to execute on Oculus. This process can take a long time depends on the game's size. As I mentioned before, Quest's CPU is not as strong as PC, some games that have good graphics, don't work smoothly. Oculus made a system called *Oculus Link* to help developers and gamers. With this system it is possible to play the VR game on Quest but game works on PC, uses PC's power. Thus games has more FPS and way faster to test the game on Unity but the game should be Windows build(exe). Quests charge cable or a Oculus Link cable is required to connect Quest to PC. Oculus Link cable has faster transmission speed compared to the charge cable therefore using the Link cable can be better for the graphic quality and input lag problem.

10.2 On Unity

Oculus Quest's output file is APK so It is not possible to double click and play. There is a 3rd party software about it but I don't recommend that because it is still on beta. The easiest way is connecting the Quest to the PC and start in Unity, in my opinion but even to play the game on this way, the Oculus account which is active on Quest should be a developer account otherwise it is not possible to play the unverified source game. There is one more way to play but it is only for one Oculus account. It is possible the push the game to accounts library but as I mentioned it is for only one account.
For now unfortunately, I don't now how the game works on VR. I even don't know does it still work but I tried to explain the game as much as I could. Puzzles and player controller were working fine when I played last time on Quest in lab.

11 Asset Sources

Phone's: `https://free3d.com/3d-model/iphonex-113534.html` accessed on 10.03.2020
Furniture's: Unfortunately this asset package is removed from Unity Asset Store
Coffee Machine's: `https://free3d.com/3d-model/coffee-machine-rigged-78507.html` accessed on 10.03.2020
Key's: `https://assetstore.unity.com/packages/3d/handpainted-keys-42044` accessed on 02.03.2020
Oculus Integration: `https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022` accessed on 07.02.2020
Fuse box's: `https://assetstore.unity.com/packages/3d/props/sockets-and-fuse-box-pack-148620` accessed on 04.03.2020
Clock's: `https://assetstore.unity.com/packages/3d/props/interior/clock-free-44164` accessed on 04.03.2020
Toilet door's: `https://assetstore.unity.com/packages/3d/props/interior/door-free-pack-aferar-148411` accessed on 25.03.2020
Laptop's: `https://free3d.com/3d-model/notebook-low-poly-version-57341.html` accessed on 21.03.2020
TV News's: `https://www.youtube.com/watch?v=P2sKgm8sfz0` accessed on 19.07.2020
Sound effects: `http://soundbible.com/free-sound-effects-1.html` and `https://freesound.org/`
Money stack's: `https://free3d.com/3d-model/stacks-of-money-52211.html` accessed on 19.02.2020
Fake window's: `https://forum.unity.com/threads/free-fake-window-shader.394971/` accessed on 12.03.2020