Main Design and development – VR Driver

Pablo Córcoles Molina

Escuela Superior de Ingeniería Informática
Universidad de Castilla-La Mancha
02006 Albacete, España
pablo.corcoles2@alu.uclm.es

Index

Index of Figures

1 Introduction

At first, the game was planned to be a VR Desktop game played in an HTC VIVE. But, due to the difficulty and the limited times for testing the headset, the development was shifted to a google cardboard one, played on a mobile phone.

The game title was “VR Rally” at first place, but then it was changed to “VR Driver”.

2 VR Rally HTC Vive Development

At first, the game was designed to be a racing game in a circuit. You should use the controllers to drive the car through the steering wheel and complete the circuit in less than the specific time.

Due to the short time we have to develop the game and the short periods of time we have with the HTC, I decided to shift to the google cardboard trying to maintain the main aspects of the game.

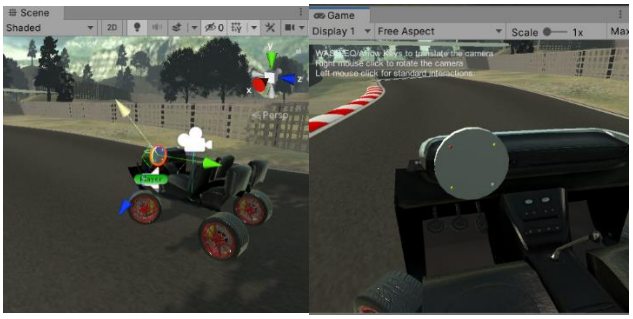


Figure 1 - First View of the HTC game

Anyway, I also upload the HTC game project and the last build made for it.

3 VR Driver Google VR Development

The name for the final game is “VR Driver”. I try to maintain the main aspects of the previous approach. Therefore, I also added another game mode and a menu.

Here. I will discuss the main aspects of this game.

The main objective of the game is to complete an objective in each game mode.

3.1 Google VR Cardboard Approach

The migration from the previous project was somehow easy but with a few errors. Finally, I import the package "GVR SDK for Unity v1.200.1" that works with the Unity 2019.3 version and resolve some problems with the Android SDK provided by Unity.

It is pretty simple to have Google VR working in your scene, you only have to drop to the scene menu some prefabs as "GvrEventSystem" (That works as the normal event system), "GvrEditorEmulator" and you have to attach to the camera a prefab called "GvrReticlePointer" (for the reticle in charge of selecting and clicking things) and a script called "GvrPointerPhysicsRaycaster" (in charge on determine at what distant the reticle can start working).

4 Design

4.1 Inputs and control of the car

The main functionality of the game was to track you head in order to move the car. I achieve this by saving the rotation of the axis of the camera and giving those values to the horizontal and vertical inputs of the car. Horizontal to drive to the sides, and vertical to increase or decrease speed. As I expect that this approach may have different behaviour, I decided to simplify the vertical input. If your head is up you will speed up, if the head is in the middle, you will maintain inertia, if the head is a bit down you will brake, and if your head even lower and you are stopped, you will go backwards.

Here is an example of this behaviour.

```
1 reference
private void GetInput()
{
    horizontalInput = camera.transform.localEulerAngles.y;
    if(camera.transform.localEulerAngles.x >180){
        if(!acelerationSound.isPlaying){
            acelerationSound.Play();
            reverseSound.Stop();
            brakeSound.Stop();
        }
        verticalInput = camera.transform.localEulerAngles.x;
        isBreaking=false;
    }
    if(camera.transform.localEulerAngles.x <5){
        verticalInput = 0;
        isBreaking=false;
    }
    if(camera.transform.localEulerAngles.x >10 && camera.transform.localEulerAngles.x<180)
    {
        if(!brakeSound.isPlaying){
            brakeSound.Play();
            reverseSound.Stop();
            acelerationSound.Stop();
        }
        isBreaking=true;
        verticalInput = -100;
    }
    if(camera.transform.localEulerAngles.x >20 && camera.transform.localEulerAngles.x<180){
        if(!reverseSound.isPlaying){
            //Debug.Log("sonido Reverso");
            reverseSound.Play();
            brakeSound.Stop();
            acelerationSound.Stop();
        }
        verticalInput = -200;
        isBreaking=false;
    }
```

Figure 2 - Inputs of the car

4.2 Design of the car and the camera

As the VR approach change, it was better to change the camera of the game. It was thought at first to be first person, but I changed to third person because I thought it was better for the motion of the car through the head and I also find a suitable 3d model.

The design of the car was given by a car controller script I found on the internet, and I used for inspiration.

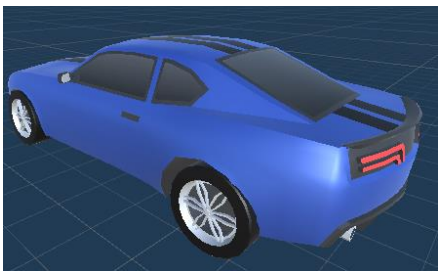


Figure 3 - Car 3D model

4.3 Design of the obstacles, upgrades, and others

Apart from the car, we have several other things to consider in each map. We have obstacles (red cubes), spheres that give you more time (green spheres), checkpoints (yellow rectangles), finish line, timer, and an arrow (that gives you the direction you have to follow).

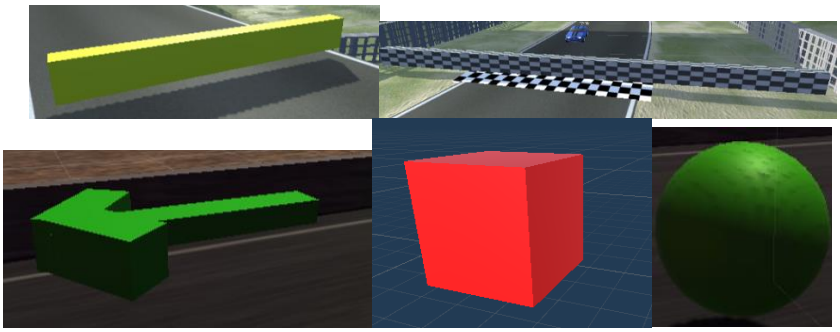
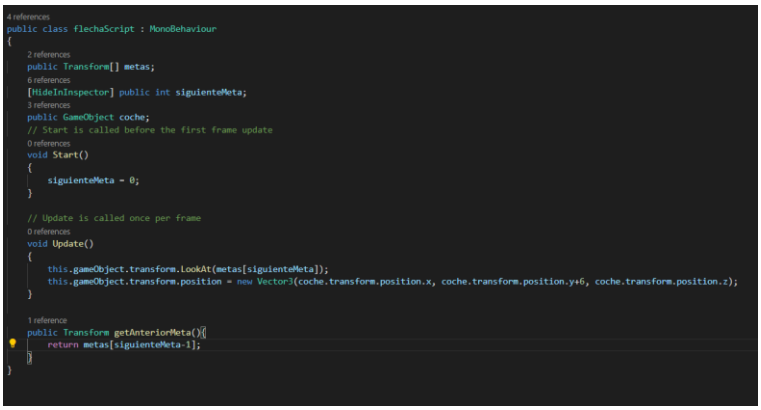


Figure 4 - Some of the designs

The trickiest part here was to do the checkpoints and make the arrow follows them. I achieve this making a list of the checkpoints and finish line of the current scene, and increasing the index each time you cross one, until you reach the final checkpoint, and the finish line appears (to avoid a player crossing the map by the middle to get sooner to the end)



```
4 references
public class flechaScript : MonoBehaviour
{
    2 references
    public Transform[] metas;
    6 references
    [HideInInspector] public int siguienteMeta;
    3 references
    public GameObject coche;
    // Start is called before the first frame update
    0 references
    void Start()
    {
        siguienteMeta = 0;
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        this.gameObject.transform.LookAt(metas[siguienteMeta]);
        this.gameObject.transform.position = new Vector3(coche.transform.position.x, coche.transform.position.y+6, coche.transform.position.z);
    }

    1 reference
    public Transform getAnteriorMeta(){
        return metas[siguienteMeta-1];
    }
}
```

Figure 5 - Script for the Arrow

```
    0 references
    void OnTriggerEnter(Collider other) {
        if(other.tag=="car"){
            var flechaGO = GameObject.Find("Flecha");
            if(flechaGO.GetComponent<flechaScript>().siguienteMeta==CheckPointId){
                flechaGO.GetComponent<flechaScript>().siguienteMeta+=1;
                checkpointSound.Play();
                this.GetComponent<Renderer>().enabled=false;
            }
            if(CheckPointId==5){
                Destroy(GameObject.Find("InitialTrigger"));
                metaTriggerGO.SetActive(true);
            }
        }
    }
}
```

Figure 6 - Trigger of each checkpoint

It is also interesting to mention that I created several spawners for the obstacles in order to make it a bit harder and interesting.

```
0 references
void Update(){
    transform.Translate(-Vector3.forward*speed*Time.deltaTime);
}
// Start is called before the first frame update
0 references
public void destruirse()
{
    Destroy(this.gameObject);
}

0 references
[redacted]

0 references
private IEnumerator OnTriggerEnter(Collider other) {
    Debug.Log("Trigger de "+this.gameObject.name+" con "+ other.gameObject.name);
    if(other.tag=="car"){
        other.gameObject.GetComponent<CarController>().obstaculo=true;
        this.gameObject.GetComponent<Renderer>().enabled=false;
        yield return new WaitForSeconds(segundosParado);
        other.gameObject.GetComponent<CarController>().obstaculo=false;
        crashSound.Play();
        destruirse();
    }
    if(other.tag=="barrera"){
        destruirse();
    }
}
```

Figure 7 - Obstacle Script

```
1 reference
public void Init(){
    InvokeRepeating("Spawn",firstdelay,delay);
}

// Update is called once per frame
0 references
void Update()
{
}

0 references
void Spawn(){
    GameObject p = Instantiate(obstacle, transform.position, transform.rotation);
}
```

Figure 8 - Spawner or Instantiation script

4.4 Design of the Game Modes and Objectives

I was looking to make a simple but also enjoyable VR game. So, I decided it will consist of two game modes or levels.

Highway to hell

This game mode was added when I shifted to the cardboard approach. It consists of a straight road where you must avoid the obstacles to get to the final line. In this case, the models used are all basic figures of Unity 3d objects including some materials and a skybox from the Internet to give it a better look.

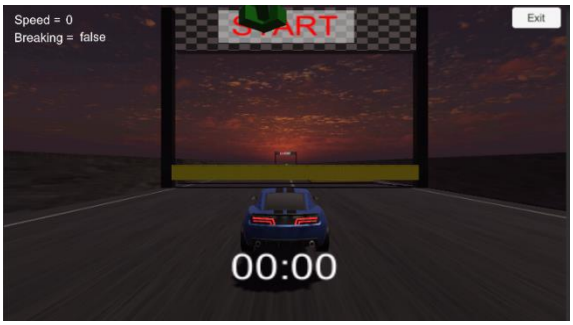


Figure 9 - Highway mode

Circuit

This game mode is inherited from the HTC version (but with a lot of modifications due to the poor progress of the previous version). You must complete a lap in the circuit crossing through all the checkpoints. This mode has a custom map from a free asset of the Unity store. It has a different skybox to give it another appearance. I also get rid of the obstacles because the main difficulty of this mode is to know how to drive properly to get to the final in time.

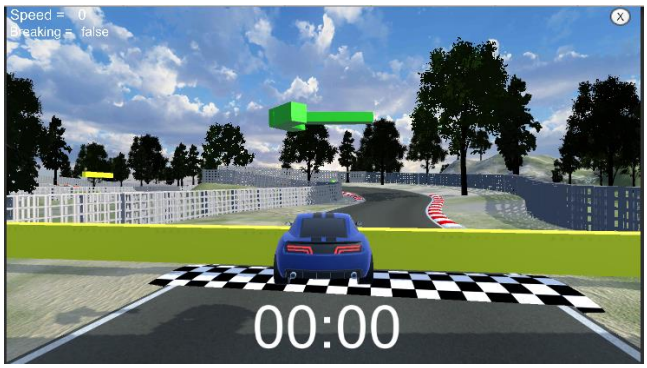


Figure 10 - Circuit mode

4.5 Design of the menus

I want the menus to be simple and that the user could easily navigate through them with the head and the view. You only have to look to the option and press the screen.

Figure 11 - Main Menu

Figure 12 - How To Play Menu

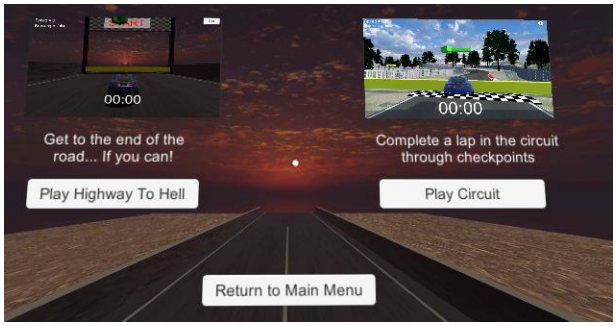


Figure 13 - Select Game Mode Menu

5 Music

I include several sounds and songs to make the game more professional and fun. The main menu has the Gran Turismo 4 Arcade Menu song which reminds me of playing racing games and the button sound is from that game.

Each game mode has a different song. The first has a song called “Gas gas gas – Manuel” popularized for the famous car anime Initial D as well as the other song “Dejavu” which it also appears in the anime.

The collision with the red obstacles also has a punch sound and the green spheres that gives you time has a different sound to differentiate from the rest. The checkpoints also have their own sound.

The car has sound when you are looking up and speeding up, and while breaking.

6 Conclusion

To sum up, I would like to have made this game for the HTC Vive headset, but It has been also very enriching to develop for the google cardboard. I think great games can be made using this technology. I think people underestimated the VR in mobile phones, because I think you can have a great experience.

There are more scripts in the project, I include in this document the scripts and images that seems to me more interesting to explain.

In overall, I think these developments have help us to understand more what a game development is and to know how difficult and enriching is to create a good game. Also, I really like the opportunity to gain experience in Unity 3D inside the computer degree because I would like to make at least a commercial game at some point (If I can).