Creating a Unity Game and the Hurdles to Overcome Various Problems

Chris Nutter

California State University Fullerton

***Abstract* -** Mention the abstract for the article. An abstract is a brief summary of a research article, thesis, review, conference proceeding or any in-depth analysis of a subject or discipline and is often used to help the reader quickly ascertain the paper's purpose. When used, an abstract always appears at the beginning of a manuscript, acting as the point-of-entry for any given scientific paper or patent application.

I. INTRODUCTION

The Unity engine is powerful in creating a decent 2D or 3D project of the creator's desire. It is also a decent way of building presets or prefabs for files created in whole projects. Unity itself uses C-Sharp as a coding language to handle physics, object entry, and more. The problem was Unity personally was the lack of knowledge of the program. Using various online tutorials and searching of documentation the ability to start understanding Unity became easier. The main drive was creating a Unity project for a VR landscape. This quickly, as shown, devolved into an ongoing project with various other projects devolving from it with the intention of generating usability in Unity.

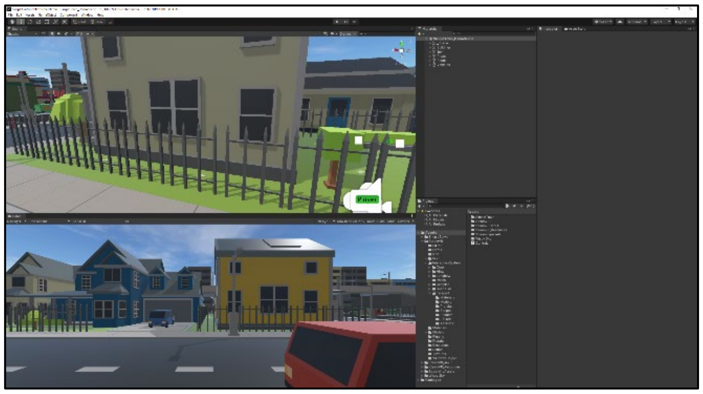


Figure 1 – The Basic Unity Layout

II. CHOOSING UNITY AND THE PROJECT DETAILS

Choosing Unity was a simple choice because of many factors. Unity allows the creation of multiple types of projects stemming from 2D based projects whether it be side-scrolling genre or fighting style games. 3D based projects are around the same difficulty however working with another axis and physical boundary could be daunting to newcomers.

A. Use of SteamVR Plugin

Using the SteamVR plugin in Unity was an obvious decision due to the feasibility and the value it gives. With the SteamVR plugin, the player, teleporting, and various other SteamVR requirements are already given including controls. Teleporting works immediately out the gate. The first obstacle to overcome was the layout and accessibility of the program. Once the SteamVR plugin was imported, the various options shown in *Figure 1* gave out the impression of a complicated task. Taking each step one-by-one helped solidify many problems that came to mind.

B. Deciding the Game Concept

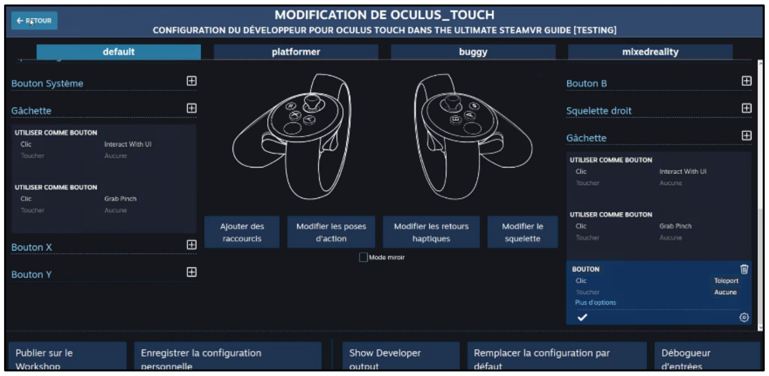
Next step after creating the Unity space is deciding on the game idea. 3D-platforming games are something of a past time since

Figure 2 – Base Environment Used from Asset Store

1996 with the release of *Super Mario 64*. Not many games were able to replicate how innovative the game became even today. Being the focus for my game, the decision to go with a 3D platformer was interesting due to the VR nature of development. VR is a decently different field to work with due to the intensity of certain situations. For instance, 3D platformers often require fast movement seen through variously quick inputs from user. This may require the real player to move their bodies in fast motions causing nausea. Working around this may be a problem. Movement of the character requires more fluid motion than teleporting requiring the coding of directional movement with the joypad (easy to overcome). All this comes down to the environment of the player and the abilities the player must work with. Choosing a basic cartoon layout could fit nicely with the vast amounts of real-world concepts in VR and the lack of high-fidelity in VR.

C. Personal Knowledge with New Formatting

C-sharp is something that needed to be learned in order to do basic physics-based operations. Luckily personal knowledge of C as well as YouTube videos were able to aid. Valem [1] of YouTube gave an appropriate introduction to SteamVR in Unity by giving a detailed way of working with Unity as well as creating basic movement options for the Player. While he didn't go to far in-depth, it helped bridge the gap between working with nothing and having a base to work with. Working more and more, C-sharp is


MODIFICATION DE OCULUS_TOUCH
CONFIGURATION DU DÉVELOPPEUR POUR OCULUS TOUCH DANS THE ULTIMATE STEAMVR GUIDE [TESTING]

Bouton Système
Gâchette
UTILISER COMME BOUTON
Clic Interact With UI
Toucher Aucune
UTILISER COMME BOUTON
Clic Grab Pinch
Toucher Aucune
Bouton X
Bouton Y

Bouton B
Squelette droit
Gâchette
UTILISER COMME BOUTON
Clic Interact With UI
Toucher Aucune
UTILISER COMME BOUTON
Clic Grab Pinch
Toucher Aucune
BOUTON
Clic Teleport
Toucher Aucune
Plus d'options

Figure 3 – Controller Layout Mapping

very similar to C is many ways which helped guide the way through many stumps. Working with many new functions created for Unity as well as Valve (creators of SteamVR and the plugin)

III. Process of Creating a Movement System

The tutorial mentioned earlier created by Valem [1] gives step-by-step the process of creating a basic movement system in SteamVR. Starting from nothing, we are first instructed to download the SteamVR plugin off Unity's Assets Store. The plugin is free allowing developers to get started on projects with very little fee being Unity itself. The tutorial later talks about the basics of what each object is. Player being the player in VR including the camera mounted to the head. How to create a place for the user to rest without falling infinitely through the map. Next, Valem [1] talks about running the project and how to view the controllers in action. Being able to notice yourself with your hands is crucial to understand how the Player works in the project. Valem goes on to give off a piece of C-sharp code demonstrating how to view the controllers in the game to show how to the controllers work in VR corresponding to your hands which is vital to mapping a controller layout that properly fits the game at hand. Following is how to map the controller to corresponding actions in SteamVR with the window in *Figure 3*. Mapping the controls is easy and requires mapping the pre-defined scripts to the buttons. Lastly was creating the basic walking system used. Valem [1] provided code with his project on how to walk around in Unity showcased in *Figure 4*. Slightly modified formatting, this gave the ability to walk the player around the base project field I imported. Something noticed was the assets imported did not contain collision field which meant the player could not walk around but instead would fall through the map. Fixing this was easy and required selecting all assets and adding Box Collider [3] provided within Unity's scripts. This allows collision to be added like how solid objects work in real life including the ability to allow the object to be picked up and moved around.

```
if(input.axis.magnitude > 0.1f) {
    Vector3 direction = Player.instance.hmdTransform.TransformDirection(new
    Vector3(input.axis.x, 0, input.axis.y));

    player.Move(speed * Time.deltaTime * Vector3.ProjectOnPlane(direction, Vector3.up)
    - new Vector3(0, 9.81f, 0) * Time.deltaTime);
}
```

Figure 4 – Code for Joystick Movement System

IV. Conclusion

After thoroughly watching and understanding the tutorial the basic project had been associated and creation has only just begun. Understanding the layout of Unity and the mechanics of the software allowed the project to reach new potential as a developer. Now that the project's beginnings are at its disposal, it was time to plan out the next steps and goals for the project. Creating a basic timeline for the ongoing project to go. Fine tuning the controls is the main priority as 3D platformers should strive to fluid controlling game as shown by games including *Super Mario 64* and *A Hat in Time*. Toying around the idea of creating a player model was possible however should not be main priority as its not required. The next step would be fixing the environment to allow more versatility. Moving cars, boats, swimming in water, etc. This is possible the hardest task to take on because of the workload however it should work well. The project is still ongoing and will be for the foreseeable future.

References

[1] Valem. *STEAM VR – The Ultimate VR Developer Guide – PART 1*. *STEAM VR – The Ultimate VR Developer Guide* – PART 1, 28 Oct. 2019, www.youtube.com/watch?v=5C6zr4Q5AlA.

[2] "SteamVR Unity Plugin." *SteamVR Unity Plugin*, Valve Corporation, valvesoftware.github.io/steamvr_unity_plugin/api/index.html.

[3] Technologies, Unity. "Unity User Manual (2019.3)." *Unity*, docs.unity3d.com/Manual/index.html.