COMP4801

Final Year Project

VR Gamification of Open World Games

Interim Report

Group: FYP23091

The University of Hong Kong

Department of Computer Science

Student: Zhi Weichen (3035448550)

Supervised by Dr. Choi Yi King

Table of Contents

Table of Figures

1. Introduction

This document serves as the interim report for the final year project *VR gamification of open world games*. We will first introduce the prospect of VR and gaming respectively by the industrial side as of 2023, then we'll re-address the advantage of combining VR and gaming to justify the practice of the project. After the background information, we try to describe the objective of the project in detail regarding the depth and scope, which is followed by the methodology of how this project is carried out and, the report finishes by disclosing the current progress and next steps of this project. A project timeline is also attached as the appendix.

2. Background

2.1. Dream of Virtual Reality Gaming

Every step forward of the XR industry ecstasies me of a virtual world which is could be similar to the world we live in, but with much more, if not infinite, possibilities all at my disposal. Gaming, among all the applications of VR, is definitely one of the best way to explore the feeling of immersion and fantasy at the same time. According to William J. et al (2017), VR games have been separating more of the cakes that their counterparts with respect to gamer satisfaction level.

2.2. Unity Engine in Gaming Development

Game engines nowadays provides the developers with flexible frameworks and handful of tools to save the latter from starting everything from scratch. With a proceeding reputation, Unity is a game engine which surpasses its counterparts in cross-platform support and graphical drag & drop programming.

As for the scope of this project, Unity serves as the foundation of the application. The *Universal Render Pipeline* is applied for better compatibility on mobile devices. The *Unity Input System* serves as the interface between hardware and callback functions.

3. Objective

To obtain the advantage of VR gaming combined together, this project aims to bring out a VR gaming demo where players can explore the virtual world with handful of interaction and immersion.

3.1. Player Controller

Most of the game needs a moving character, the first objective of the project is to implement a cross-platform player controller which supports multiple input sources. This player controller should be easy to expand for more interactive actions for the future virtual realization progress.

3.2. Facilitating Vibe of Fantasy and Immersion

The next objective is to find a way to enhance the prior advantage of VR gaming by enhancing the vibe of immersion and fantasy. This could be done by an exaggerated mimic to the real world we live in, so that the player can find themselves with both familiarity and fresh feeling.

Due to the limit of scope of the project, a low-poly art style will be applied to save the effect on unrelated art creation. Also, as an attempt of project management, the project is divided into different function modules to achieve the objectives from different perspectives.

4. Methodology

This project is built on the Unity game engine, with gaming logic written in C# scripts. As for the current progress of the project, this document introduces two methodologies applied in the development process: 1) The Unity Input System, and 2) Unity Shaders. More detailed discussion on the reasons to adopt the two is given in the 5. Progress part.

To control a gaming character, a controller script which defines the behavior of the character is required. The Unity Input System serves as the interface between the physical hardware, i.e. keyboard, and the controller script. The Unity Input System is adopted to replace the old Input Manager for its cross-platform hardware support.

As for the approach to bolster the vibe of fantasy and immersion, after some R&D on trending games of recent years, this project tries to mimic a real tiny world, however, by also applying a “macroscopic spherical bending” effect to the world of the game. Specifically, the similar cylindrical bending effect is adopted in the Animal Crossing (2020). The spherical effect to bolster the atmosphere of fantasy is originally implemented in a straightforward approach by constructing the world with spherical models, i.e., build the "planet" on an actual sphere and overload the physics such as gravity so that other objects sticks to the planet. However, it finally turns out to be infeasible due to much complicated physics performance and we turned to the approach of simulating spherical world with shaders. A more detailed discussion on this matter is given in the 5.2 Shader part.

Figure 1 A Cylindrical Bending Effect in Animal Crossing (2020)

For the future virtual realization part, the XR Interaction Toolkit by Unity will be adopted, However, this should be beyond the scope of this document and will be delivered in the final report.

5. Progress

After the progressive development in the first semester, this project has achieved part of the objectives as mentioned in *3. Objective.* Specifically, an input system for cross-platform (PC and console) is implemented for player control, and a curling shader is designed for a special spherical rendering effect.

5.1. Input System

The Unity Input System package was introduced in Unity 2019.1 to replace the old Unity Input Manager where the developer used to manually read different input parameters, i.e., keys and virtual axes, in their functional scripts.

As shown in Figure 2, the Input System listens to the Unity events invoked by Unity engine when the connected device sends signal. The Input System then translates the events into actions stored in the Input Asset component (Figure 3), then pass the actions to the custom player controller script that calls actual game logic functions.

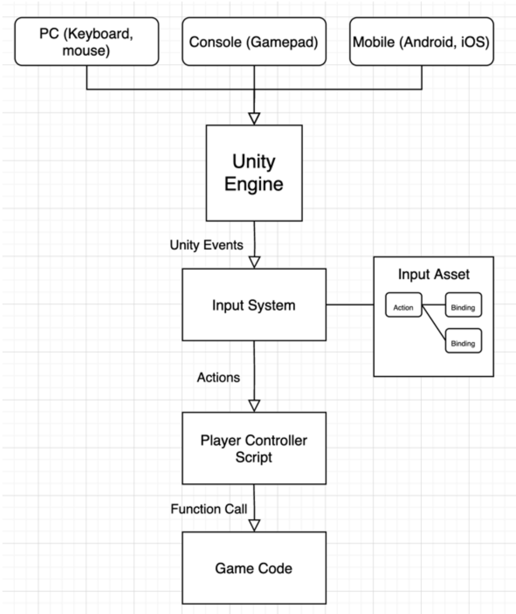

Figure 2 Input System Workflow

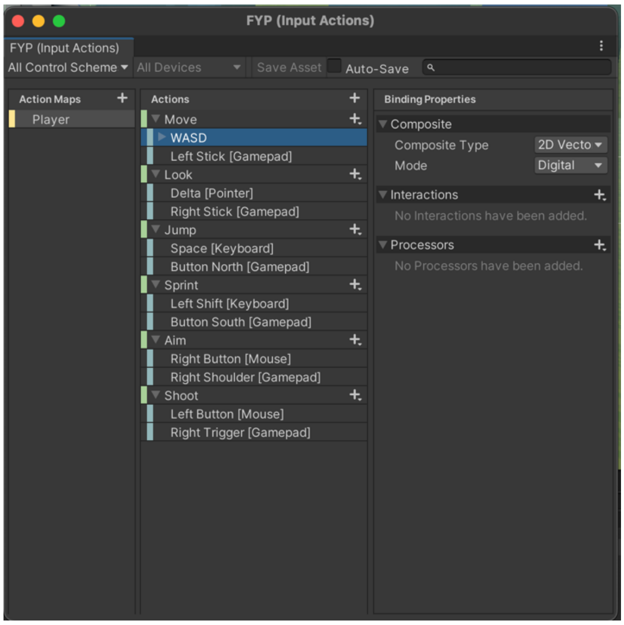

Figure 3 Input Action Binding

To achieve the objective of controlling a player, this project has bound a set of actions shown in Figure 3. The Input System therefore calls for the corresponding functions implemented in the controller script. A link to a complete demo of the controller actions is attached as Appendix II.

5.2. Curling Shader

The above section 4. Methodology mentioned the failed attempt of constructing a spherical world with actual spherical models. This part will introduce an alternative of the straightforward method, by utilizing the unity shader.

5.2.1. Unity Shader

Shaders are small components used for graphics rendering. Specifically, a Vertex Shader informs the GPU about the correct positions of all vertices when rendering, whereas a Fragment Shader is responsible for the calculation of the color of each pixel. By utilizing a shader, we can make an overall displacement to the position of every vertices of a model, and therefore achieve a distortion of the final rendered image.

5.2.2. Curling Shader

First, we try to simplify the objective of making a plane object spherical by ranking down and make it two-dimensional. When we try to curl a line to an arc, what we can observe from Figure 4, is that the extra "shift" of the camera-object distance can be approximately seen as positively proportional to the square of the horizontal distance "dist" between the camera and the object.

$$Shift = Dist^2 * someBendingFactor$$

That is, by telling the GPU to render all the vertices to a new position (viewing from the camera's vertical axis, i.e., y axis) which is now added by the shift, a line now looks like a curved arc.

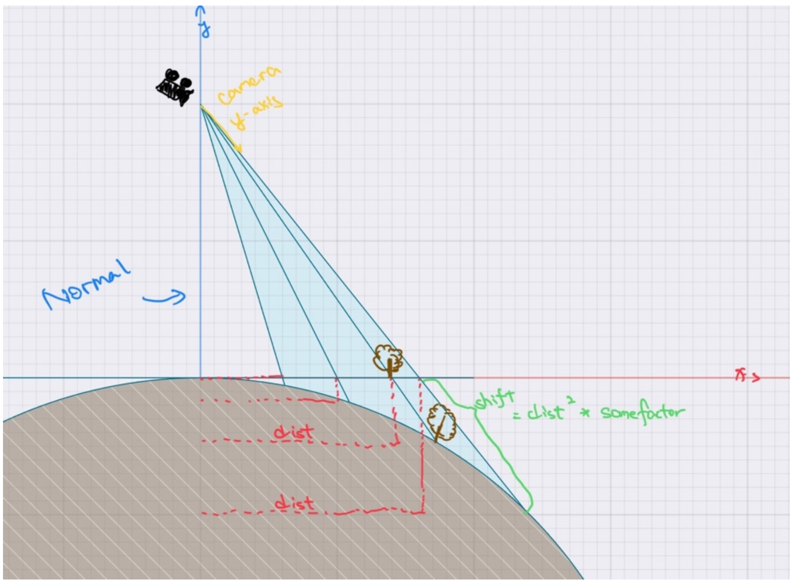

Figure 4 Two Dimensional Curling

5.2.3. Spherical Shader & Implementation

After we toggled the steps for a two-dimensional cylindrical bending effect, the key to spherical bending is straightforward. By adding an extra camera-object distance which is positively proportion to the sum of square of both horizontal axes (x & z axes as in Unity3d).

$$Shift = (distInXAxis^2 + distInZAxis^2) * someBendingFactor$$

By applying this shader, the original flat plane now looks like "rippling out" (Figure 5) as the new camera-object distance is equally increased for a set of vertices on the circle with same horizontal distance to the projection of the camera (original point).

The implementation of this spherically bending shader is completed graphically with a Shader Graph plugin, which can be seen as a workflow visualization of such a vertices displacement.

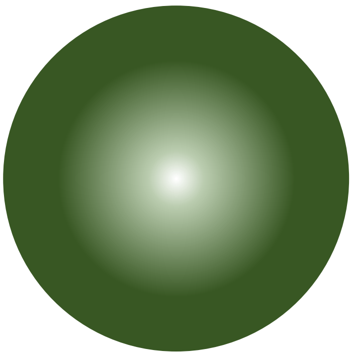

Figure 5 Visualization of Distance (shown in color spectrum) from Camera to a Plain W.R.T. Horizontal Distances

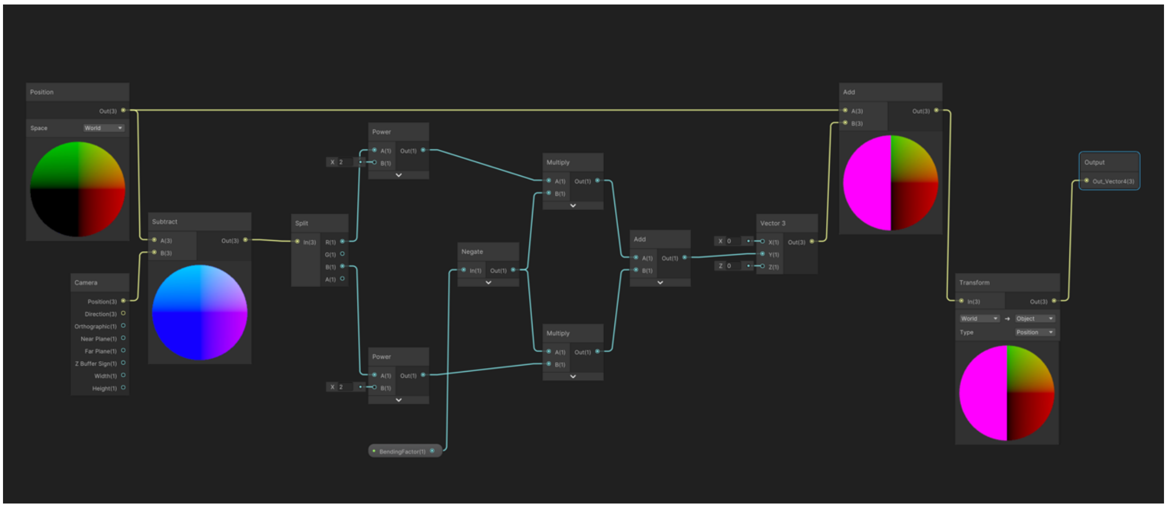

Figure 6 Visualization of Spherically Bending Shader Implemented with Shader Graph

5.2.4. General Effect

This section shows screenshots of a spherical world achieved by adding a curling shader to a plain world. The result is quite satisfying as shown in the following figures (Figure 7 & 8).

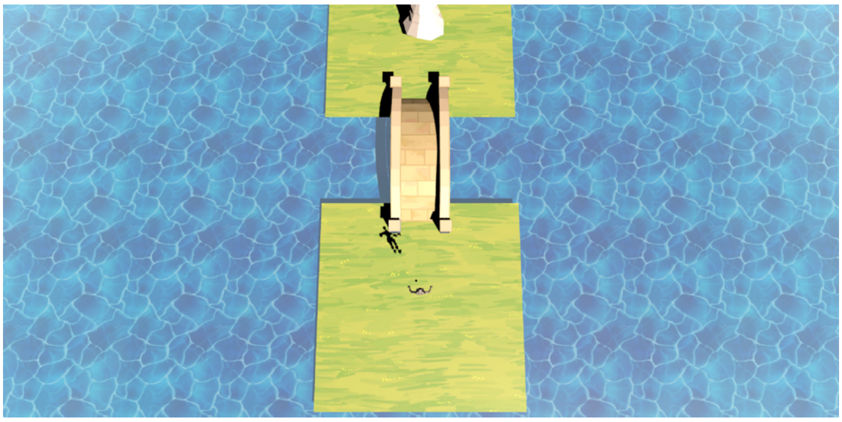

Figure 7 Top Down View of a Plain World

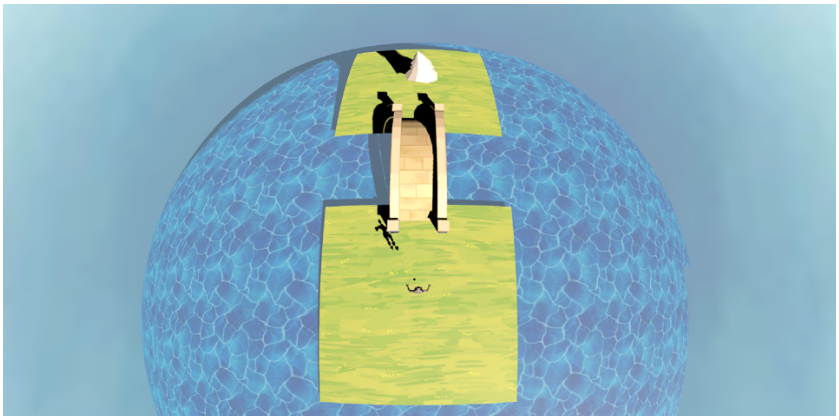

Figure 8 Same World but Applied with Curling Shader

6. Future Steps

In view of the current progress, the objectives remaining for the next semester is planned as follows.

6.1. Interactive Content

As for now, the project only reaches interaction by moving the character and shooting projectile to targets. As one major attraction point and objective of the project, this project should be able to deliver more interactive content such as more kinds of postures, interacting with environment or being able to get quests from NPCs. However, the project's scope should be able to reasonably deliver content that involves efforts in R&D. A more detailed content of what to be delivered regarding the interaction part should be further discussed with the supervisor.

6.2. Virtual Realization

This project, as stated in the topic, also aims to bring out a VR compatible version by the final presentation by mid-April. The schedule seems likely to be feasible since Unity also provides with a XR package that is out-of-the-box. Therefore, the main focus should still be on the interactive content and building an immersive environment to play in.

Appendix I: Player Controller Demo

The link below directs to the google drive folder with the demo of player's behavior such as moving, jumping and shooting targets.

https://drive.google.com/drive/folders/1bc1FNwrMA_yQNKJ_Pzmm1cqsSsSWInq9?usp=drive_link

Appendix II: Project Schedule

21 Jan 2024	Phase 2 Deliverables – Preliminary implementation, Detailed interim report
Mid Jan – Late Jan 2024	R&D – Confirming with supervisor on details to bolster the interaction of the project.
Feb - Mar 2024	Implement the interaction part of the project.
Late Mar 2024	Starting the virtual realization procedure.
15-19 Apr 2024	Final Presentation
23 Apr 2024	Phase 3 Deliverables – Finalized tested implementation, Final report
26 Apr 2024	Project exhibition

Reference

1. J. Jerald, P. Giokaris, D. Woodall, A. Hartholt, A. Chandak and S. Kuntz, "Developing virtual reality applications with Unity," 2014 IEEE Virtual Reality (VR), Minneapolis, MN, USA, 2014, pp. 1-3, doi: 10.1109/VR.2014.6802117.
2. *Learn game development with Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn*. (n.d.). Unity Learn. https://learn.unity.com
3. PlayStation. (2023, February 23). Resident Evil 4 - 3rd Trailer | PS5 & PS4 Games [Video]. YouTube. https://www.youtube.com/watch?v=bwxMrAy4z-s
4. *Most played VR games.* (n.d.). SteamDB. https://steamdb.info/charts/?tagid=21978
5. Nintendo of America. (2019, June 11). Animal Crossing: New Horizons - Nintendo Switch Trailer - Nintendo E3 2019 [Video]. YouTube. https://www.youtube.com/watch?v=_3YNL0OWio0
6. Shelstad, W. J., Smith, D. C., & Chaparro, B. S. (2017, September). Gaming on the Rift: How Virtual Reality Affects Game User Satisfaction. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *61*(1), 2072–2076. https://doi.org/10.1177/1541931213602001
7. Technologies, U. (n.d.-a). Unity - Manual: Input system. https://docs.unity3d.com/Manual/com.unity.inputsystem.html
8. Technologies, U. (n.d.-b). Unity - Manual: Shaders. https://docs.unity3d.com/Manual/Shaders.html