Department of Computer Science,

Faculty of Engineering,

The University of Hong Kong

COMP4801 Final Year Project

A First-Person VR Puzzle Game

Supervised by Dr. T. W. Chim

FYP18015 Individual Final Report

Wong Ka Wing (3035124269)

Date of submission: 14/04/2019

Abstract

Many industries have aggressively employed virtual reality technology for cooperating this unique virtual feature into their project in order to enhance the performance of their desired result. Typically the game industry has found a big impulse from this technology to its field, and VR gaming thus became a trendy experience for players all over the world. In the meantime, machine learning has also been aggregated into gaming too. Programmers are so eager to find possibilities that make this technology to be more mature to integrate with a new game idea. This paper are then subjected to the purpose of presenting the work around of "Dream Driver", a VR puzzle game which merges the concept of Reinforcement Learning and puzzles games, tries to present a virtual reality experience to players with a smart level-difficulty controller behind. In the following chapter, the project background, methodology and results will be examined in detail.

Acknowledgement

We would like to extend our sincere thanks to all individuals and organizations who have kindly assisted and offered help to this project.

We are highly indebted to our supervisor Dr. T. W. Chim for his regular supervision, sharing of his experience and ideas for this project. He has always conducted meetings with us and giving us useful advices and feedbacks.

We would like to give special thanks to FIGHT 4 DREAM LIMITED for their expert suggestion. They have aggressively provided us advices and answered our questions regarding game development.

Table of Contents

List of Figures

Abbreviations

VR	Virtual Reality
ML	Machine Learning
GDD	Game Design Document
SDK	Software Development Kit
VIU	Vive Input Utility
API	Application Programming Interface

1. Introduction

In this chapter, the background information of the project will be introduced by the evaluation of the status in different industries using VR. Following with the scope and objectives, the motivation of developing a VR puzzle game will also be addressed. Lastly, the outline of this paper is located at the end for reference.

1.1 Background

The rapid growth of VR technology has brought up a numerous of applications in different industries, like medical [1], logistics and business [2]. Among all, gaming is one of the field that has been substantially promoted because this technology absolutely brings the user experience to a next level [3]. Undoubtedly, VR games are being releasing onto the market. Game developers attempt aggressively to explore more innovative idea to fully utilize the experience that VR technology provides. Some work on various game types and some focus on the hardware support [4] which collapse into the same goal of enhancing the human senses in the virtual world.

At this stage in the VR game industry, the leading companies in the market like HTC, Samsung and Sony have published a similar model of VR components to all around the world (see Figure 1) [5][6][7]. These standard equipments are constituted by a headset, a pair of controller and a pair of base station. Providing with a set of these devices, players could have their visual and interactive illusion of being in the virtual world by wearing the headset and playing the object around with the controllers.

On the other hand, machine learning are coming in gaming too. For examples, Google has invented Emoji Scavenger Hunt taking the idea of supervised learning in 2018 [8]. Applying the neural network built behind, the system can do a image classification to match the input photo by user. Unity, a game development platform has also started to explore the possibilities from reinforcement learning in lately 2017 [9], which is also the time the ML toolkit for Unity that first officially available for developers. They proves the evolution of game.

Figure 1. *HTC Vive Headset, Controllers and Base Stations [5].*

1.2 Scope

Puzzle games are a popular game type on different platforms. When it comes to VR, game developers have suggested some great idea like virtual escape room [10]. However, lack of interaction related implementation turns out to be the common failure of VR puzzle games upon our observation and experience. Puzzle game should be solvable within a reasonable logic, any restrictions that does not allows players to demonstrate their common sense are a bad practice for a puzzle game design [11]. Especially for such a VR puzzle game that highlighting the importance of interaction, it becomes the best choice for a VR game project aiming to enhance the player's visual feeling and experience in the virtual world.

1.3 Objectives

The primary objective of our project is to develop a VR puzzle game in Unity and coding with C# languages. Players can play the game with a PC and a set of HTC Vive VR kit. In order to reinforce the virtual experience for players, the game aims to mitigate all the unnatural and weird feeling brings up by the unresponsive reaction for every object interaction. With the idea of open-ended puzzle integrated, players will have even more freedom to explore the virtual world and adopt the best way they find to pass a level in game. Supporting with the logic scripts implemented, this feature will probably provide another excitement to our players while they are being in the virtual world.

The secondary objective of developing the game is to try to integrate the Reinforcement Learning technique into the puzzle game using the ML agent toolkit provided by Unity, which we are aiming to develop an architecture that could be utilized the power from the neural network trained in a creative manner. In our project, we are going towards the direction that allows the neural network to find a best path for a solution to the puzzle provided in each level. Then a list of action generated by the model can be used to match with a list of actual player's choice, and compute for the performance of the player in that level, so to control the difficulty of next level procedurally.

However, this is nearly an impossible task to finish a complete version of the game with the quality assurance within a short period of time. As a result, the final deliverable will only be the first phase of the entire game. In which it leads to the final objective of this project, that is to get ready for the future implementation. To achieve this, we have to maintain our code its readability, modifiability and sustainability for adapting any possible structural change afterward.

1.4 Outline

This paper demonstrates the processes to build up a VR puzzle game from scratch, which includes the preparation work for setting up a VR ready environment, designing procedure for pointing out the details of workflow and implementation method for achieving the desired outcome and effect in the game.

The remainder of this report proceeds as follows. First, we introduce the methodology for implementing a VR game including the justification of software choices and the methodology for combining the idea of ML into the game. Followed with our final result obtained in this project, the difficulties encountered and the mitigate solutions have also been discussed. Lastly, we close with the conclusion of this paper.

2. Methodology

This chapter introduces two main tools for developing a VR game. The first one is game design document where it act as a implementation guideline for programmer to follow with. The second one is Unity, one of the most well known game development platform. Major implementation details will be discussed in this part.

2.1 Game Design Document

Before starting the project at once by digging into the coding part, a game design document (GDD) is employed to present the idea in a systematic manner after brainstorming [13]. The GDD generally includes the game identity, mechanics summary, features, interface and art style etc. These kind of keyword allows the developer to focus on the scope of the game. Some detailed GDD also provides several use cases to indicate every details coming from an idea. As the routine is explained step by step, developers can easily follow the instruction and implement the required logic accordingly. It boosts the efficiency of development process by making everything clear.

2.1.1 Design Principle

As being a beginner in game development, most of our work can be done are highly relying on the SDK provided by the publisher, especially for those newly released VR-related package where only a limited modification can be applied to avoid the faulty deconstruction of content structure, the design principle is then mostly depending on the practicality of the implementation. While the feasibility is being taken into account, the time cost is also a concern in the design process. Therefore, the scale of project has to be always under control. To conclude, the first priority of our design principle is to be practical.

For the general in-game ideas, the most efficient way for us is referring to the game ideas on the current market. As puzzle game is a typical game type, we have concluded three main design principles after conducting the research. They involves the common sense, solvable by brute force and relativity guarantee. The first one means the solution to the puzzle can always has a trivial answer like a natural phenomenon, the second one means the puzzle can always be solved within a reasonable trials and the last one means the flawless logic between puzzle should always be presented. And the game ideas are generated by these principles to ensure there is enjoyment provided to every player inside the game.

2.2 Unity Game Engine

Unity is a famous game development platform which provides a clear user interface and a set of simple commands and operations built-in [14]. The main reason for choosing Unity as the game engine is we have had experience to play around it in our previous game development course, we are more familiar with it. In addition, comparing to another game engine, Unity supports high level javascript and C# languages for development. These two languages are rather handy especially for beginners as everything can be defined like an English sentence [15]. The asset store associated along with Unity is also intrinsic to an efficient development. As a amount of high quality assets can be obtained free from charge on the asset store, we as an engineer can have no worry regarding the art stuff too.

2.2.1 Unity 3D Game Implementation

To develop a first-person VR game, we need a 3D environment and a 3D physics mechanics supported inside the game. In the following sub-chapter, hierarchy of a set of objects, commonly used game mechanics and visuals will be addressed in detail.

2.2.1.1 Hierarchy and Referencing of Objects

Unity Editor provides a handy interface in both the hierarchy tree and inspector for dragging and dropping object such as scripts and assets under a certain field to treat as a parameter or child of its parent (see Figure 2). Taking advantage from this, we can access to a specific object in the code by assigning a publicly accessible variable in the inspector with drag and drop or reference the object by calling the "GetComponent" function directly from a game object. It keeps the whole structure of the game hierarchy a sense of independent by means of not adopting any specific architectural pattern such as Model-View-Controller model to avoid any of constraint derived from it.

2.2.1.2 Commonly Used Methods

Trigger and collision detection are broadly used in the implementation. To apply the trigger detection, we add the component of collider to a desired object and set it to be a trigger. Similar for the collision detection, we add a rigidbody component on it, so the physics related calculation will be automatically performed in the game engine when two entity are collided together. After that, they are cheated as an event listener to respond to the status change with the collided object (see Figure 3). With such convenient listener provided, we implemented the action response logic to achieve our desired outcome in most of the cases.

Figure 2. *Inspector drag and drop*

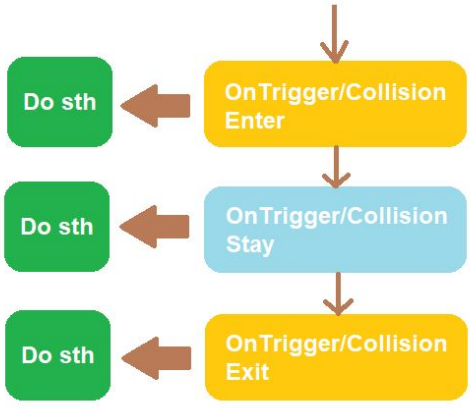

Figure 3. *Trigger and Collision State*

2.2.2 Unity Vive Input Utility

As we are adopting the HTC Vive VR kit as our hardware support, we need a dedicated SDK provided by the manufacturer to establish the connection between the user input and virtual environment. Vive Input Utility SDK package was available in Unity asset store serves as this purpose. There are script and APIs calls responding to the input of each hardware component. And it implies the APIs will be relied more frequently in the implementation especially for some of the game mechanics.

2.2.2.1 Hardware to Software Level Referencing

HTC Vive has recently released their exclusive SDK package, "Vive Input Utility" onto the Unity asset store. The SDK consists of the aforementioned testing scene, a detailed tutorial and guideline explaining the relationship and interaction method between the VR components and the virtual world. The APIs function call across different testing scripts provided are all well defined such that developers can match a basic level interaction with each of the hardware component such as the button press down action trigger (see Figure 4), and the script that defined all the properties of a grabbable should have as to be attached to an game object. These features are noticeable to be quite similar as an event listener too, which helps a lot with our implementation when we were getting more familiar with the code beforehand.

Class **ViveInput** under **HTC.UnityPlugin.Vive** provides a simpler API to achieve that.

```
using UnityEngine;
using HTC.UnityPlugin.Vive;

public class GetPressDown_ViveInput : MonoBehaviour
{
    private void Update()
    {
        // get trigger down
        if (ViveInput.GetPressDownEx(HandRole.RightHand, ControllerButton.Trigger))
        {
            // ...
        }
    }
}
```

Figure 4. *VIU Scripting [16].*

2.2.3 Unity ML-Agents Toolkit

To achieve the integration of machine learning technology and game development, the most feasible way for beginners is to look for plugins supported by the game development platform. Luckily we got our very first beta version of an open-source software which has just released from Unity, namely the ML-Agents [17]. This SDK helps developers and researchers to map games and simulations built in the Unity Editor into environments where intelligent agents can be trained using Deep Reinforcement Learning, Evolutionary Strategies, or other machine learning methods through a simple to use Python API. A set of example projects and baseline algorithms were also included in the SDK for users to get an intuition on how to apply those onto their projects. In such sense, we definitely need this for our project and the detailed implementation will be discussed in the following sub chapter.

2.2.3.1 Agent Observation

Observation in a deep learning process represents the status in an environment that the artificial intelligent need to know for deciding the next action. In the script provided in the ML-Agents SDK, we have to specify the observations to be added in each cycle. Combining the observations and previous knowledge gained from the reward function which will be discussed next, the agent should have sufficient information to determine a better decision for obtain a higher reward. Quoting the example from the official tutorial, it is just like to learn to balance a rotating ball on a platform (see Figure 5), you need to add the observation of the state of each relevant unit such as the rotation of the platform, the relative position of the ball, and the velocity of the ball [18]. In such case, the trained model can therefore theoretically perform well comparing to the absent of observations.

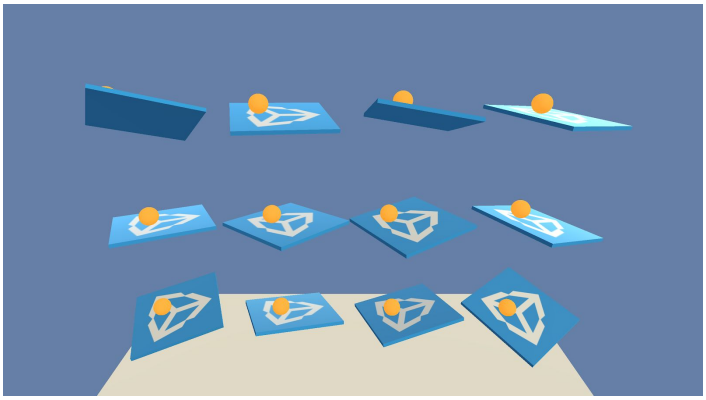

Figure 5. *Example Environment - 3D Ball [19].*

2.2.3.2 Agent Reward function

The reward in a reinforcement learning is an indicator to reflect the correctness of the action taken by the agent. The PPO reinforcement learning algorithm established by Unity works behind the scene, monitoring the performance and providing guideline to the agent by optimizing the cumulative reward earned from the decision made by the agent. As be mentioned in the previous paragraph, this algorithm should work with the observations. Taking the same example of 3D Ball, we should have the reward set to be negative in order to indicate as a punishment of a wrong decision making. For example, a negative reward can be given when the next action determined by the agent will lead to the ball fell down from the platform, vice versa. After a set of condition and constraints implemented in the reward function, the agent will learn the best action procedurally by the first sight of the observation in latter iteration cycle. This theorem applies exactly the same in our project.

2.2.3.3 Agent Action

A list of action can be defined in the agent action function, while each of the item in the list represents the index of a list of command. And there could be discrete or continuous vector action spaces. In a simpler case such as in a two dimension game environment, the action indicating the movement along x or y axis can be stored in a discrete format with only a Vector2 type parameter. In our project, an index of a list of command in discrete space will be more suitable in the sense that we want to allocate a true action for the agent but not a floating value.

2.2.3.1 Training Parameters

For the python training, we have not dig deep into the parameters that vary the efficiency. The most frequent changed values are the learning rate, batch and buffer size, gamma and the maximum steps the training takes. The learning rate is decreased if we found the training is full of fluctuation, and the reward does not increase consistently. And we are setting the batch and buffer size rather small as to lower the times of experience before the agent updates any decision criteria in order to fit our small discrete vector action space. For the gamma, it acts as a discount parameter to the future reward, and it is typically set close to 1 to just make the training result more stable in the very last training duration. And the maximum steps are setting to reasonably large to make sure the result are stable enough. (Since my project partner is responsible for the training, please find more detail in his paper).

3. Results

In this chapter, the progress and the final result for our project is presented. It includes the design progress of the game ideas, a successful hardware testing result, and the method we have adopted to help with version controlling. Lastly, the main features implemented in terms of puzzle game and the integration of machine learning algorithm will also be presented.

3.1 Progress Along the Project

As the preparation work are also important in game development, the progress we have made all along from the beginning will be discussed following for record. They includes the game design procedure, hardware testing and version controlling.

3.1.1 Game Design

After conducting research on the trending style of puzzle game available on the market, we have settled down with the topic and criteria of our game. We have finished the majority part of our GDD with writing down all the unique features, systems, art style and level design of our game. As developing a puzzle game requires a lot more innovative idea, each stage need to have its own characteristic and be a much more different one than another. These tasks are pending to be designed and added into our to-do list in Trello, an application that allows users to manage their projects and organize things in a easy, flexible and visual way. With the help of Trello, we can keep track of our process of development and achieve division of work easily. It will be a convenient tool when we come to the stage of implementation.

3.1.2 Hardware Testing

The HTC Vive VR components have been successfully set up and tested in our own place followed by the step by step instructions (see Figure 6 and 7). We have gone through the connection of the headset and computer, firmware update of the controller and room scale measuring performed by the two paired up base stations for motion tracking. We finished and passed also a built in tutorial associated in Steam VR and a Unity version testing scene to ensure the functionality of all the hardware.

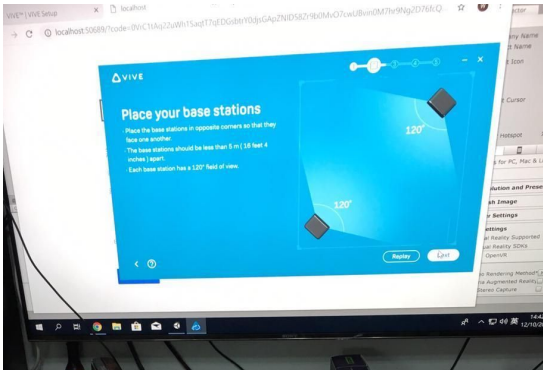

Figure 6. *Setting Up the Hardwares.*

Figure 7. *Hardware Testing.*

3.1.3 Version Control

Upon the study on the newly released official API, we started our very first project for testing purpose. The characterised summon feature in our game has been successfully implemented by working around mainly the trigger checking and the event handler to response to an event while the collision between the controller and the virtual object has been detected. This project progress has been located on Github, an online repository for developers to store their work and monitor the working schedule. Github also serves another function, which is to help different contributors to maintain a different version of code by cooperating with a git client software. We adopted Sourcetree for this purpose.

3.2 Final Results

In this Chapter, the final result presented in the deliverables will be discussed. They includes the feature implementation of the puzzle game and the part of applying the training result from the ML-agents.

3.2.1 Feature Implementation

In order to reinforce the uniqueness of Dream Driver, two features will be employed in Dream Driver. They are locomotion with gesture and Dream Driver. The remaining subsections introduce the detail of each feature.

3.2.1.1 Locomotion with Gesture

To further facilitate the future implementation, another testing scene has been built for evaluating different functionalities of the predefined logic scripts provided in the VR SDK. It includes the basic grabbable logic which once the script has been attached and applied on to an object, the object will react to the VR controller trigger button while the controller is hovering the object. And the follow up response of the object like sticky and unblockable state could also be controlled by other advanced script inheriting from the basic one. Based on these evaluations, the common grab and drag mechanism in most of the VR game could be easily achieved and applied into our puzzle game.

In addition, our originated implementation for the locomotions in VR has been successfully done. To achieve the simulation of floating in the air that people would possibly have their experience in the dream, we found out that ninja run mechanism (see Figure 8) is the most suitable case to accomplish this fantasy as it is a rather natural gesture for people to realise their forward movement without struggling with their hands or feet around the same position. And the work around is theoretically simple. Only the raycaster has been used to capture the direction pointing by the controller (see Figure 9). After that, the players will be translated forward according to the negative value of the ray casting direction.

Figure 8. *Ninja Run Locomotion.*

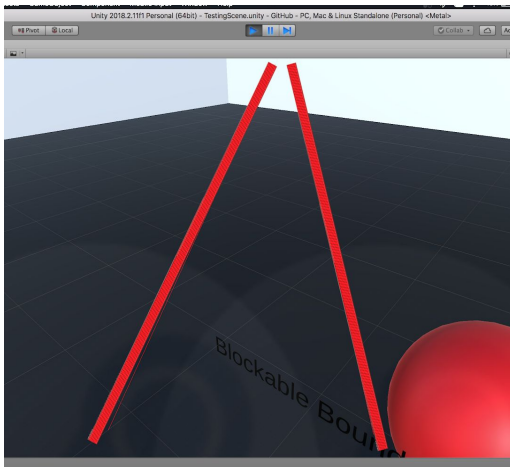

Figure 9. *Ray Casting From Controllers.*

3.2.1.2 Dream Driver

Dream Driver is one of our feature system. In the storyline, It acts as a tool for storing magical power and releasing sword skill by consuming magical power. Dream Driver's appearance similar to a watch. It is attached on the player's left hand throughout the whole game (see Figure 10). Before the player is able to release a sword skill, player needs to absorb magical power. In order to achieve this particular function, there are three main logics cooperating with each other. Among these three main logics, energy absorption, summon, and activating sword skill has been successfully implemented.

Figure 10. *Dream Driver.*

3.2.1.2.1 Energy Absorption

In order to activate a particular sword skill, the player has to first collect the corresponding colored energy by placing the colored energy cube inside the corresponding power station (see Figure 11), while different energy will have different ways to obtain. Next, Generator will be charged with energy and activated for absorption (see Figure 12). When the player get close enough to the Charged Generator and hold down the grip button of the left controller for several seconds, the energy will be successfully stored into the Driver and the animation of successfully absorption will be played (see Figure 13).

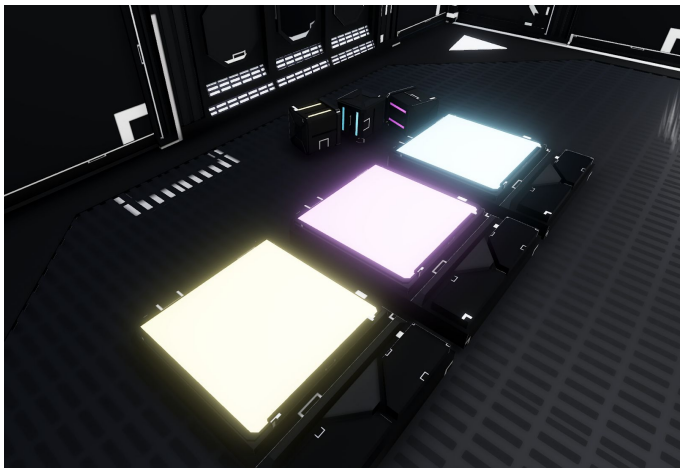

Figure 11. *Energy Cube and Power Station.*

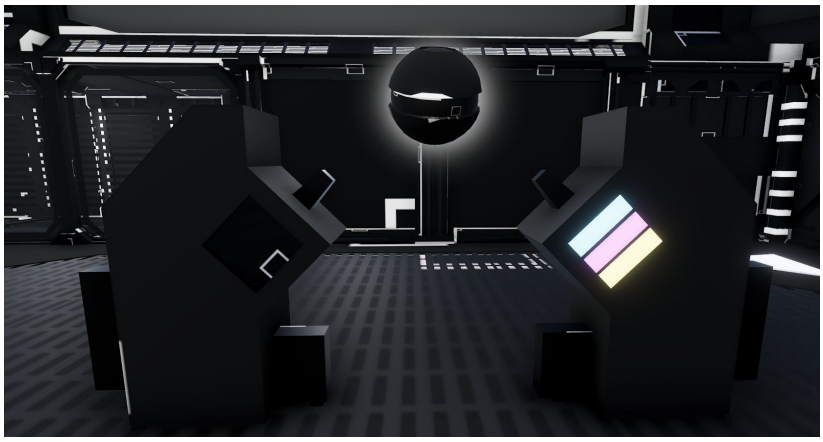

Figure 12. *Power Station with Absorbed Energy Order From Blue to Yellow.*

Figure 13. *Animation of Successfully Absorption.*

3.2.1.2.2 Summon and Sword Skill

After obtained the energy, the player will be able to summon a sword with emitting color as same as the color of absorbed energy (see Figure 14). The player can summon the sword by hold down the grip button of the right controller for several seconds. After summoning the sword, the system will activate the corresponding sword skill to release the seal of the room and escape out of here. While the seal of the room are a combination of three different colored power, player therefore has to first absorbed the corresponding color in order to get out of the stage.

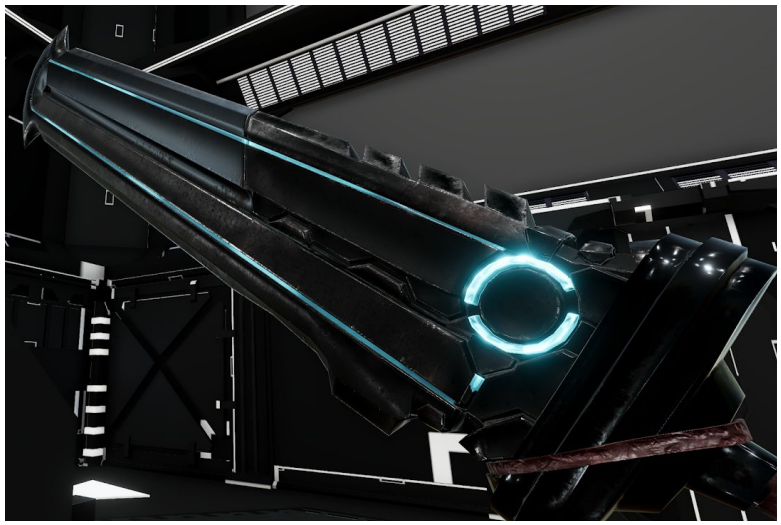

Figure 14. *Sword with Cyan Energy.*

3.2.2 Machine Learning

Integrating the machine learning algorithm, we trained a agent to observe the environment and react with a fastest solution path to solve the puzzle in each level. To demonstrate the idea of how well would be a trained agent compared to a human player. We first need to establish a puzzle game environment for it to familiar with. And this moves us to the stage design of the puzzle game. In the following, the preparation works and results all about machine learning in gaming will be discussed.

3.2.2.1 Level Design

Three features or we called level difficulties have been implemented in order to demonstrate the usage of the ML-Agents toolkit. The first difficulty would be rather simple which is to test how well a player can match a color cube to a corresponding color plate. While the second level of difficulty is to test how well a player can figure out the pattern to a specific dependence between two entities which demand only the observation of the change using the mechanism of rotation around the center basis. This is make use of a concept of parallel world such that the colored cube in two different world will have different position and if the player tries to move the cube along one direction, the other cube in the next world will be moved in another direction. And for the final difficulty is similar to the second, which is to test how sensitive is the player can observe for the environmental change from taking different combination of actions. And this is actually the phenomenon we want player to notice that while player is grabbing a special magic cube, other cubes will lose their gravity.

3.2.2.1 Agent Training

To facilitate the training, we have to configure the environment and reward to our agent inside the defined function provided along with the ML-Agents toolkit. As be mentioned in previous chapter, we need an observation collection, reward mechanism and a set of action provided to the agent. In this case, we add all of the useful information of a stage could have into the observation, such as how the level has been set up, the present of colored cube, parallel world and magic cube, etc. And for the reward function, we predefined the branch that the agent could be taken, and reward the agent when it find the a correct list of action to solve the puzzle, and punish otherwise. Lastly for the action, we took the similar strategy of predefined all possible action the agent can take in all of the level, and let the agent choose and update its action from the same pool.

3.2.2.2 Level Difficulty Control

Upon the agent was well trained, we run the agent behind the scene along with a real player's attempt. The agent would the compute for the best solution for a specific level that the player is playing. The list of action of the player will also be recorded in order to compare with the result bring up by the agent when the player can successfully pass a stage. In this sense, once we do the comparison between the player's performance and the theoretically optimal solution by the agent, we know how well is the player perform and thus to manage and generated a level with the difficulty that can be reflected by the result. And we can achieve a game that could procedurally increase the level difficulty according to the player performance applying the technique of machine learning.

3.2.3 Visuals and Art

The following will talk about the minimum work we have finished for the visuals and artistic work, mainly the artistic evaluation and the post-processing.

3.2.3.1 Artistic Evaluation

In terms of the artistic work, the art style of our game has confirmed to be low poly. And as long as the asset is suitable, we will stick to what can be acquired in the Unity asset store. Unless for the featuring item in the game, we may have need to spare some more time to prototype the 3D model by ourselves in other modelling software like blender. For our current progress regarding the artistic work, a basic experiment laboratory scene has temporary built (see Figure 15). It will be repeatedly used in every stage throughout the game as to convince the player to follow the fundamental storyline where they are going to solve puzzles level by level to practice their problem solving skill. Along with the scene building, the lighting of the environment, the technical scene transition, the door open-and-close animation in between scenes has also been implemented.

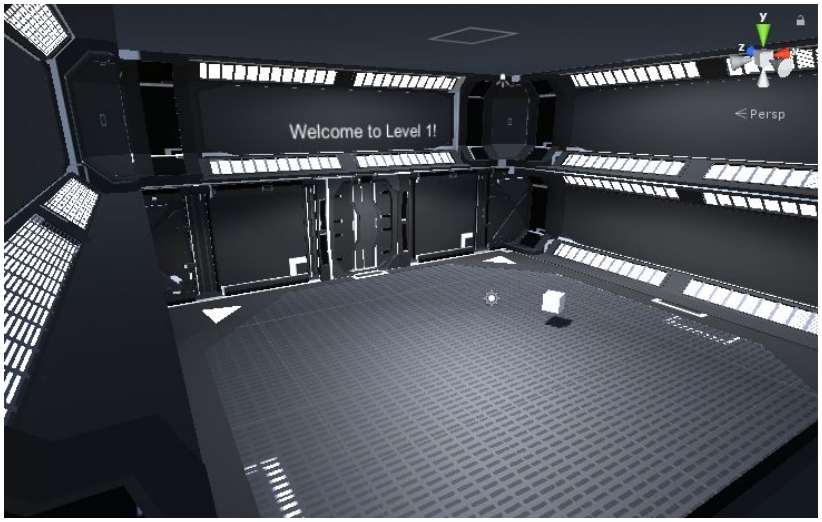

Figure 15. *Experiment laboratory scene in game.*

3.2.3.2 Post-processing

Post-processing is employed to enhance the quality of the visuals of Dream Driver's game environment. It is the process of applying full-screen filters and effects to a camera's image buffer before it is displayed to screen. It can drastically improve the visuals of the game environment with little setup time. We use the post-processing mainly for the color grading and the blooming of the brightness. It benefits Dream Driver because the pantone of our game is being so dark and dull. It acquires players' extra attention with the effect and helps to point out some keys and clues of the puzzle naturally (see Figure 16 and Figure 17).

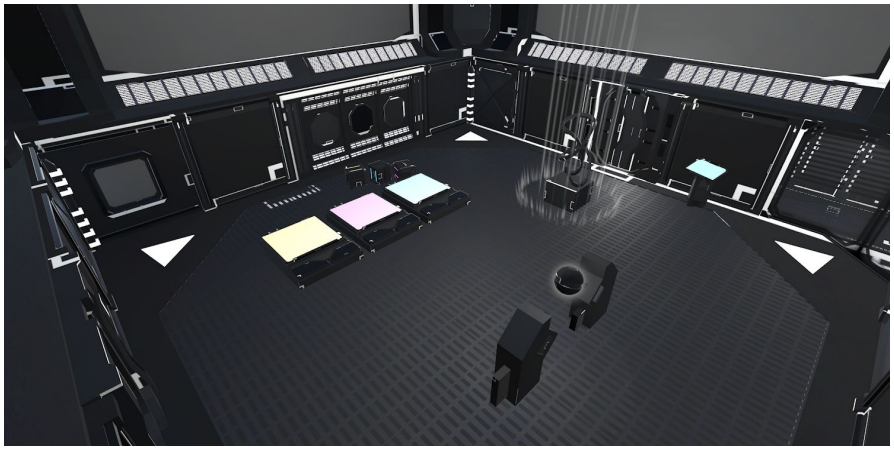

Figure 16. *Keys and Clues without Post-processing.*

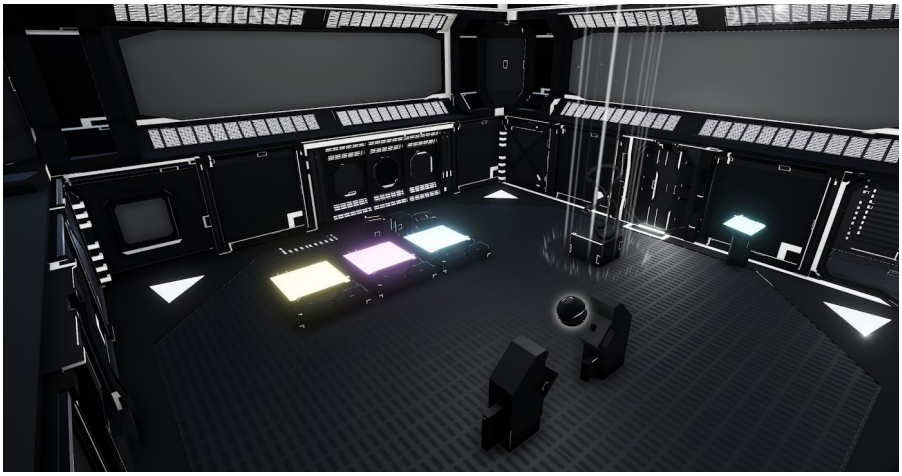

Figure 17. *Keys and Clues with Post-processing.*

4. Challenges Encountered

This chapter introduces the difficulties encountered while making design decision and dealing with the artistic work. The mitigation methods have also been addressed.

4.1 Design Decision

Gesture recognition is a typical example that we encountered as the major failure of our design decision. As most of our game idea is suggested by the actual experience of being inside a dream, the gesture recognition feature is adopted as a way to simulate the situation when one tries to summon something inside a dream. However, we foresee the implementation difficulties after the study of the API call. Without a further hardware support or fully implemented algorithm, it will be an expensive task to accomplish the recognition process which will probably affect the performance of the game. We decided to hold up the development of this feature until we get some spare time to work on. And the alternative way is to replace with a summon system to achieve this fantasy.

4.2 Artistic Work

A game is a combination of an innovative idea, implementation skill and artistic sense. Unfortunately as an engineer, we do not expert in art. We have to keep the scale of workload in respect of the artwork as small as possible. As a result, we decided to make use of what we could get from the asset store. The basic scene of every puzzle stages will therefore be the same and only alter slightly different by adding a reasonable amount of decorations to present the minimal visual enjoyment.

4.2 SDK Support

Since two main SDK we are using, the Vive Utility Input and ML-Agents toolkit are two young packages that has been released not far ago and updated versions are still keep releasing from the manufacturers. By this reason, we can only work on what has been given especially in the machine learning case, we cannot train our agent better until the ppo algorithm has enhanced by the Unity team. Luckily, it is still feasible to apply our knowledge from the machine learning class, train a agent and apply it on a simpler application level, and we think it is far more enough while we are only use the machine learning technique as a experimental basis.

5. Future Planning

In this chapter, three major steps to complete a full game will be discussed. They included all kind of system things and the enhancement of the machine learning application.

5.1 System Involved

After finishing up the entire game experience, the next thing to handle will be all kind of the game system, such as the save and load system for players to record his current progress of the game and user interface for players to interact with the game menu, etc. In the script, the function call that invoking sound effect should also be integrated into the game later in case to provide players extra enjoyment.

5.2 Machine Learning Application

As be mentioned in previous chapter, the application of the machine learning is not generally developed for all kind of the game, which is being only dedicated to our project now. We are attempting to develop an general architecture or system afterward to make the training of machine learning can be applicable for a various of environment and could be dynamically learned from the changed environment without providing predefined guideline by hand, so to facilitate all kind of games that subjected to the procedurally increment of level difficulty mechanism.

Conclusion

The final goal of this project is to develop a VR puzzle game named “Dream Driver” which utilize the machine learning technique to control the level difficulty procedurally. This paper introduces the basic background regarding the gaming industry and demonstrates the motivation of working on a new VR puzzle game from scratch. Following with the medology mentions to develop a game, this paper also states our progress mainly the preparation work have to be done before starting the implementation of the game and also the final results of the establishment of the game, which includes the documentation of GDD, hardware testing and technical studies, feature implementation and machine learning related works. We have also addressed with our mitigation method to the difficulties encountered to clarify our development decision.

For the future work, we will focusing on the implementation of the full game by the means of including all sorts of system things. We will also try more of the function given in the ML-Agents toolkit to explore more possibilities to integrate machine learning into game environment.

References

1. Laver, K. E., George, S., Thomas, S., Deutsch, J. E., & Crotty, M. (2015). Virtual reality for stroke rehabilitation. Cochrane database of systematic reviews, (2).
2. Mujber, T. S., Szecsi, T., & Hashmi, M. S. (2004). Virtual reality applications in manufacturing process simulation. Journal of materials processing technology, 155, 1834-1838.
3. Zyda, M. (2005). From visual simulation to virtual reality to games. Computer, 38(9), 25-32.
4. Lucas Matney. (2016, Nov 3). Inside Intels race to build a new reality. [Online]. Available: https://techcrunch.com/2016/11/03/inside-intels-race-to-build-a-new-reality-2/ [Accessed: Jan 16, 2019]
5. HTC Corporation (2018). Vive virtual reality system. [Online]. Available: https://www.vive.com/us/product/vive-virtual-reality-system/ [Accessed: Jan 16, 2019]
6. SAMSUNG ELECTRONICS CO., LTD. (2018). Samsung gear vr with controller. [Online]. Available: https://www.samsung.com/global/galaxy/gear-vr/ [Accessed: Jan 17, 2019]
7. Sony Interactive Entertainment Inc.(2018) PlayStation vr. [Online]. Available: https://asia.playstation.com/en-hk/psvr/ [Accessed: Jan 19, 2019]
8. Google. (2018) Emoji Scavenger Hunt. [Online] Available: https://emojiscavengerhunt.withgoogle.com/ [Accessed: Sep 21 ,2018]
9. Alessia Nigretti (2017, Dec 11). Using Machine Learning Agents Toolkit in a real game: a beginner's guide. [Online]. Available: https://blogs.unity3d.com/2017/12/11/using-machine-learning-agents-in-a-real-game-a-beginners-guide/ [Accessed: Oct 13, 2018]
10. VALVE Corporation. (2018) VR: Vacate the Room. [Online]. Available: https://store.steampowered.com/app/494810/VR_Vacate_the_Room/ [Accessed: Jan 17, 2019]

11. Smith, A. M., Andersen, E., Mateas, M., & Popović, Z. (2012, May). A case study of expressively constrainable level design automation tools for a puzzle game. In *Proceedings of the International Conference on the Foundations of Digital Games*(pp. 156-163). ACM.
12. FIGHT4DREAM LIMITED. (2016) FIGHT4DREAM. [Online]. Available: https://fight4dream.com/ [Accessed: Jan 20, 2019]
13. Rouse, R., & Illustrator-Ogden, S. (2000). *Game design theory and practice*. Wordware Publishing Inc..
14. Indraprastha, A., & Shinozaki, M. (2009). The investigation on using Unity3D game engine in urban design study. *Journal of ICT Research and Applications*, *3*(1), 1-18.
15. Hejlsberg, A., Wiltamuth, S., & Golde, P. (2006). *The C# programming language*. Adobe Press.
16. HTC Corporation. (2018). HTC VIVE Tracker (2018) Developer Guidelines. [Online]. Available: https://dl.vive.com/Tracker/Guideline/HTC_Vive_Tracker(2018)_Developer+Guidelines_v1.0.pdf. [Accessed: Jan 20, 2019]
17. Alessia Nigretti. (2017, Dec 8). Introducing ML-Agents Toolkit v0.2: Curriculum Learning, new environments, and more. [Online]. Available: https://blogs.unity3d.com/2017/12/08/introducing-ml-agents-v0-2-curriculum-learning-new-environments-and-more/ [Accessed: Oct 13, 2018]
18. Unity-Technologies. (2017). Learning-Environment-Design-Agents. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Design-Agents.md [Accessed: Jan 23, 2018]
19. Unity-Technologies. (2017). Learning-Environment-Examples. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Examples.md [Accessed: Jan 23, 2018]