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 Final Report

By: Wong Yu Hin (3035124324)

Another group member: Wong Ka Wing (3035124269)

Date of submission: 14/04/2019

Abstract

At present, an increasing number of games are developed as an open-ended puzzle game. They provides different methods for player to solve the puzzles in the game in order to reach different stages, stories or endings of the game. The underlying rationale is to the room for player to utilize his/her creativity throughout the whole game. Therefore, the gaming experience is reinforced.

This paper demonstrates the development procedures of Dream Driver, a new first-person VR puzzle game cooperating with the concept of dream with open-ended design. Players can achieve what they are thinking in our game like they had in their dream.

Currently, Dream Driver is a Personal Computer (PC) game using Unity on Window platform written in C# language which allows players to the character in virtual world by HTC Vive using VIVE Input Utility (VIU). Players can also interact with most of 3D models in stage and Summon Items with controllers. In later stage, the environment of every puzzle/stage, the interaction between Summon Items and objects, and the saving and loading system will be implemented. The graphical user interface (GUI) and the effects of actions will be further developed to reinforce the gaming experience.

Acknowledgement

We would like to extend our sincere thanks to the 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 constant supervision, provided comments and provided information in respect of this project. He has always conducted meetings with us and giving us advices and feedbacks.

Table of Contents

Abstract **1**

Acknowledgement **2**

Table of Contents **3**

List of Figures **5**

Abbreviations **6**

1. Introduction **7**

1.1 Objectives 8

1.2 Scope 9

1.3 Outline 10

2. Methodology **11**

2.1 Game Design Document 11

2.2 Unity Game Engine 11

3. Current Progress **12**

3.1 Game Design 12

3.2 Hardware Testing 12

3.3 Technical Study 13

3.4 Version Control 14

3.5 Feature Implementation 14

3.5.1 Locomotion with Gesture 14

3.5.2 Dream Driver 16

3.5.2.1 Energy Absorption 17

3.5.2.2 Summon 19

3.5.2.3 Sword Skill 20

3.5.3 Portal Feature 21

3.5.4 Auto Generating Puzzle System 22

3.6 Artistic Evaluation 24

3.7 Post-processing 25

3.8 Machine Learning 26

3.8.1 Reinforcement Learning 26

3.8.2 ML-Agents Toolkit 27

3.8.3 Application 31

4. Challenges Encountered **35**
4.1 Design Decision 35
4.2 Artistic Work 35

Conclusion **36**

References **37**

List of Figures

Figure 1	Setting Up the Hardwares	12
Figure 2	Hardware Testing	12
Figure 3	VIU Scripting	12
Figure 4	Ninja Run Locomotion	14
Figure 5	Ray Casting From Controllers	14
Figure 6	Dream Driver	15
Figure 7	Battery and Power Station	16
Figure 8	Power Station with Absorbed Energy Order From Blue to Yellow	17
Figure 9	Animation of Successfully Absorption	17
Figure 10	Sword with Cyan Energy	18
Figure 11	Sword Skill with Yellow Emitting Color	19
Figure 12	Portal Gate	20
Figure 13	Color Puzzle with Three Cyan Color	21
Figure 14	Experiment Laboratory Scene in Game	23
Figure 15	Keys and Clues without Post-processing	24
Figure 16	Keys and Clues with Post-processing	24
Figure 17	Reinforcement Learning	25
Figure 18	ML-Agents Toolkit	27
Figure 19	Dream Driver's ML-Agents Toolkit	29
Figure 20	Top with Low Learning Rate and Bottom with High Learning Rate	30
Figure 21	Training Details of Each Training Step	32
Figure 22	Summaries of Dream Driver's Training(Top) Closer Look(Bottom)	34

Abbreviations

VR	Virtual Reality
GDD	Game Design Document
SDK	Software Development Kit
VIU	Vive Input Utility
API	Application Programming Interface
PPO	Proximal Policy Optimization

1. Introduction

In the past, most games, especially puzzle games, are designed as close-ended games which having only solution and routine for everything. There is no room for accepting creative answers from the players. Now, we can observe that there is a tendency that games are developed as an open-ended puzzle game. They welcome different methods for solving the puzzles in the game from players in order to achieve different stages, stories or endings of the game. The underlying rationale is to provide the room for players to utilize their creativity throughout the whole game. The gaming experience is reinforced when comparing with puzzle games without open-ended design. Referencing two famous games, Human: Fall Flat [1] and Portal [2] are both designed as an open-ended puzzle game. Players are allowed to clear the same puzzle with different methods according to the playing style of the players. For example, brute force style method is to use the simplest way to reach the goal and creative style method is to employ different objects.

A substantial and growing number of applications of VR technology on different industries attributed to the rapid growth of VR technology. For example, medical [3], logistics and business [4]. Apart from other fields, gaming is one of the field that has been immensely influenced. The underlying reason is VR technology enhances the gaming experience of the player [5]. There are many VR games being developed and released onto the market everyday. Most of the game developers are trying to conceive more creative ideas to strength the gaming experience to a higher level that VR technology provides to player. To enhance senses of players in the virtual world, some game developers suggest using different game mode and some suggest improving the hardware support [6].

People always say that the best VR experience they have is dream [7]. The degree of freedom in the real world is minimal when comparing with that in the dream. People found that they have the similar experience in their dream and experiencing something unrealistic. This kind of physiological phenomenon is not completely being recreated and administered by anyone in spite of the magnificence of dreams. All the things will be completely different when it is

implemented onto the computer. Our game design is influenced by the abstract idea of dream. People are able to emulate and reassemble the experiences depends on the program we built. Players have the flexibility for utilizing their creativity attributed to our game complements the solving puzzle component.

1.1 Objectives

One of the goals of this project is to develop a first-person VR puzzle game that demonstrates the possibility to reinforce all human senses by utilizing VR technology, open-ended puzzle game design and abstraction of dream. Thus the gaming experience of players will be reinforced.

The second goal of this project is to develop a game that is prepared to put on the gaming market like Steam. This leads us to the third goal of this project, to develop a game with high extensibility that minimizes the efforts on maintenance and putting new features into the game. This project offers us opportunities to explore and collaborate with companies developing or disposed to possess game. A game with high extensibility provides rooms for accepting the requirements from the collaborative companies and manifest the readiness and sincerity to collaborate with companies.

1.2 Scope

This project develops a game with Unity game engine, C# language and HTC Vive. It provides a high-level gaming experience to the players by cooperating with VR technology, open-ended game design and the conception of dream. Thereby, human senses will be reinforced by utilizing VR technology.

Considering the concept of dream, we decided to build a First-Person VR puzzle game. When the game is introduced to be unrealistic and emphasize creativity, a puzzle game fits perfectly into the theme of dream. Given that a dream can be beyond our imagination, and everything is possible inside there, we can set the puzzle with even more varieties than those with restrictions. Besides, our puzzle always accepts more than one answer as an open-ended puzzle. We expect our players to explore possibilities themselves after several tutorial stages, in order to offer opportunities for them to demonstrate their creativities. As a result, we will keep our game challengeable and solvable every stage, to provide a brand-new puzzle game experience to our players.

This game with storyline can enhance the attractiveness of it to the players by integrating motivation. Storyline explains and endows the significance of keep playing this game, leaded by the instructions and achieve the goal. Thus storyline adopts a stance to provide motivation to players.

1.3 Outline

This final report introduces a new first-person VR game, named Dream Driver. Cooperating VR technology, open-ended game design, and the conception of dream, Dream Driver provides a superior enhancement on human sense when utilizing the VR technology. Players can interact with most of the objects in the game. Every stage contains different puzzles. Players are able to summon items in order to obtain extra artifacts.

The remainder of this final report proceeds as follows. First, we introduce methodology of Dream Driver, Next, progresses of Dream . Followed with the project's current progress and future planning. Lastly, we end up with a summary and references.

2. Methodology

To achieve the objectives of this project, game design document and Unity game Engine are employed for developing a VR game. The remaining subsections introduce the detail of the each tools.

2.1 Game Design Document

For this project, GDD is employed to centralize the ideas of developers and clarify the development milestones. GDD introduces the design of a game for organizing the work flow of a development team. It generally includes the game identity, features, interface and art style. The development team designs GDD for collaborating with developers. GDD acts as a guideline and instruction in the whole game development process. The developers can follow the routines stored in GDD easily as the instructions and logics are clearly presented. Therefore, it can enhance the efficiency of the development process.

2.2 Unity Game Engine

Unity game engine is used to develop Dream Driver. Unity provides asset store for developers downloading assets they need. Unity asset store provides many free and paid assets with many categories such as 3D animations, 3D characters, 2D GUI, Audio, Tutorials and AI. Unity is also commonly used by many gaming companies like FIGHT4DREAM LIMITED [8] and Magic Design Studio [9]. Thus, the readiness for collaborating with other companies is increased.

HTC Vive only support Windows Operating Systems [10]. Dream Driver will be developed as PC application on Windows platform. C# language will be used for the game implementation. Unity recommends developer to use C# instead of Javascript by removing Java selection from selecting script language since Unity 2017.2 beta [11]. C# also has better performance than Javascript [11].

3. Current Progress

Seven processes have been finished by now: Game Design, Hardware Testing, Technical Study, Version Control, Artistic Evaluation, Post-processing and Machine Learning. The remaining subsections introduce the detail of each process.

3.1 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.

3.2 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 1 and 2). 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.

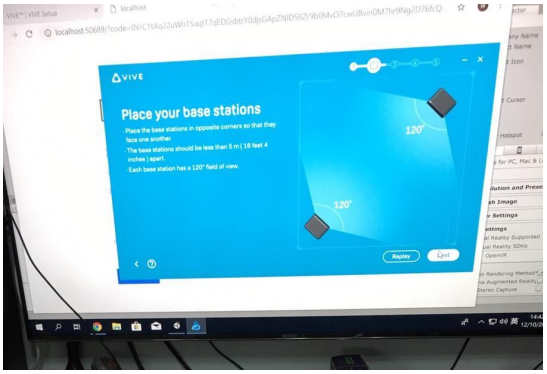

Figure 1. *Setting Up the Hardwares.*

Figure 2. *Hardware Testing.*

3.3 Technical Study

For this project, VIU is used for Dream Driver interacting with HTC Vive. VIU is an Unity plugin that allows developers to access Vive device status. It creates API for developers to use to reduce the time and effort on writing repeating code when managing Vive devices. The number of developers using VIU is minimal when comparing that using SteamVR Plugin [12]. Thus, there is not many tutorials and questions from the internet. We decided to use VIU for Dream Driver development because VIU is developed by HTC Corporation [13] while HTC Vive also is developed by HTC Corporation. From Figure 3, it shows that VIU shorten the length of implementation. This also leads Dream Driver to a game with high extensibility because VIU excluding lengthy API.

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 3. *VIU Scripting [13].*

3.4 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.

3.5 Feature Implementation

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

3.5.1 Locomotion with Gesture

Players can do the Ninja Run Locomotion to move (see Figure 4). When players pose like a running ninja, they walk forwards. In Dream Driver, the character model has a tracking box sticked on the back of it. The tracking box keeps tracking on the two controllers are entered it or not. This can be done by implementing VIU's Collider Event Handler, IColliderEventHoverEnterHandler and IColliderEventHoverExitHandler. The pointing direction of the controllers are opposite from the direction players want to move to when player doing Ninja Run Locomotion. Therefore, the moving direction can be calculated by the average opposite pointing direction of the two VR controllers. By firing ray from each controller (see Figure 5), the direction of the controller can be calculated by the position of the controller and

the first intersection point on object along the red ray. Walk function disables the translation on axis Y direction. Thus, the character cannot walk upward or downward acting like flying.

Figure 4. *Ninja Run Locomotion.*

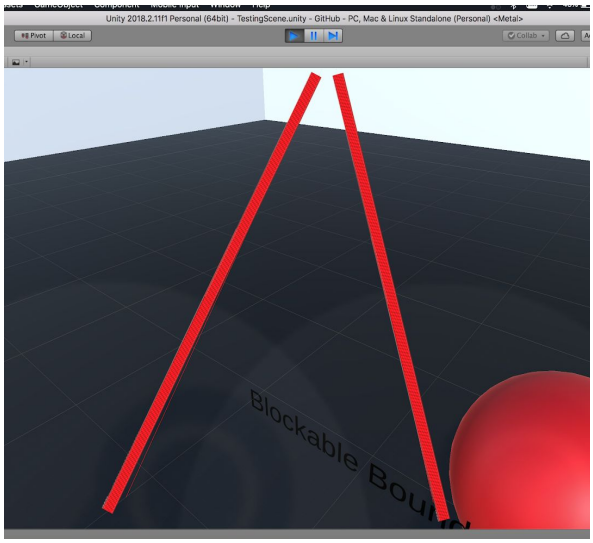

Figure 5. *Ray Casting From Controllers.*

3.5.2 Dream Driver

Dream Driver is one of our feature system. It is a tool for storing energy and consuming the energy to use sword skill. Dream Driver is a watch-like tool with six gems in circle and one at the center (see Figure 6). Before able to release a sword skill, player needs to absorb energy from the Generator. There are three main logics employed to achieve the above features. They are energy absorption, summon and sword skill. The following subsections will introduce the details of each logic.

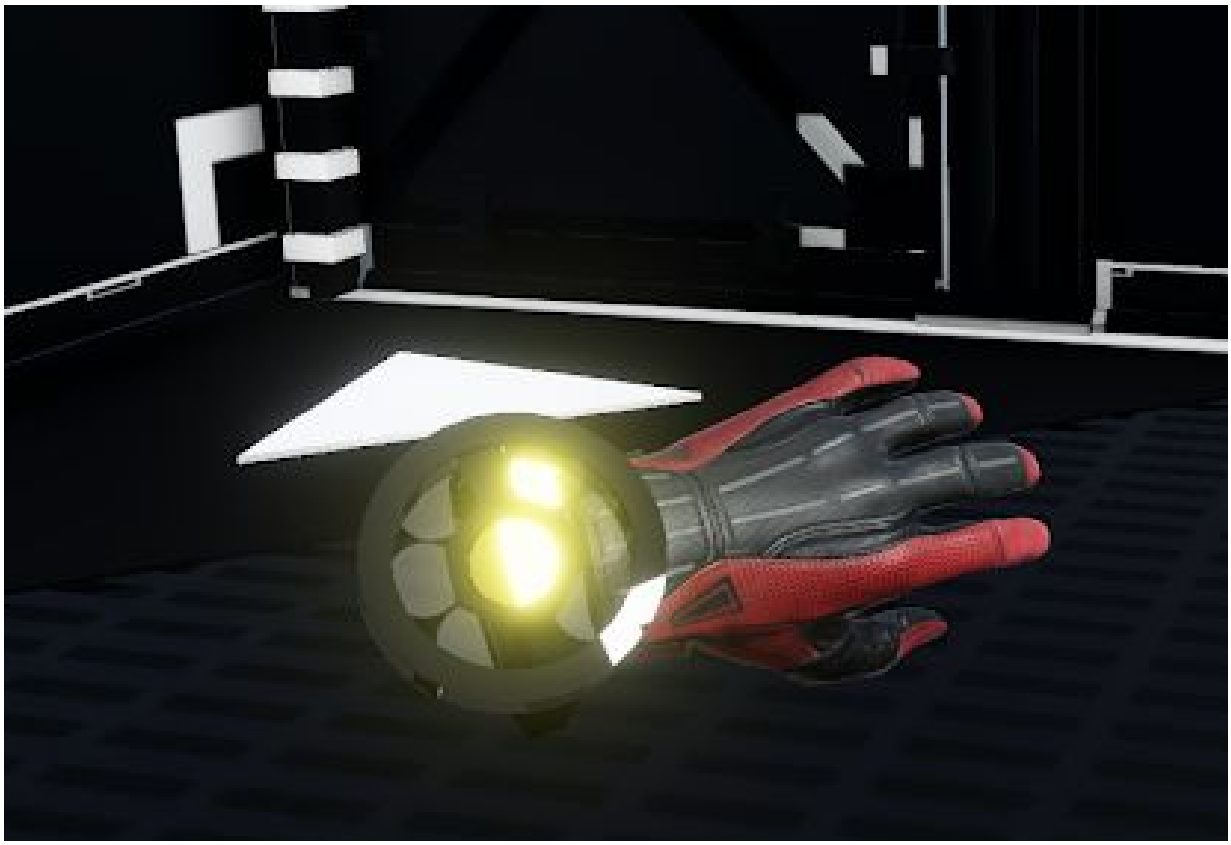

Figure 6. *Dream Driver.*

3.5.2.1 Energy Absorption

In order to solve the puzzle, the player is needed to collect the corresponding colored energy by placing the colored battery inside the power station (see Figure 7). Then, the Generator will contain energy and activate energy absorption (see Figure 8). The player can absorb the energy by pressing the left controller's grip button for several seconds when he/she is close enough to the charged Generator (see Figure 9).

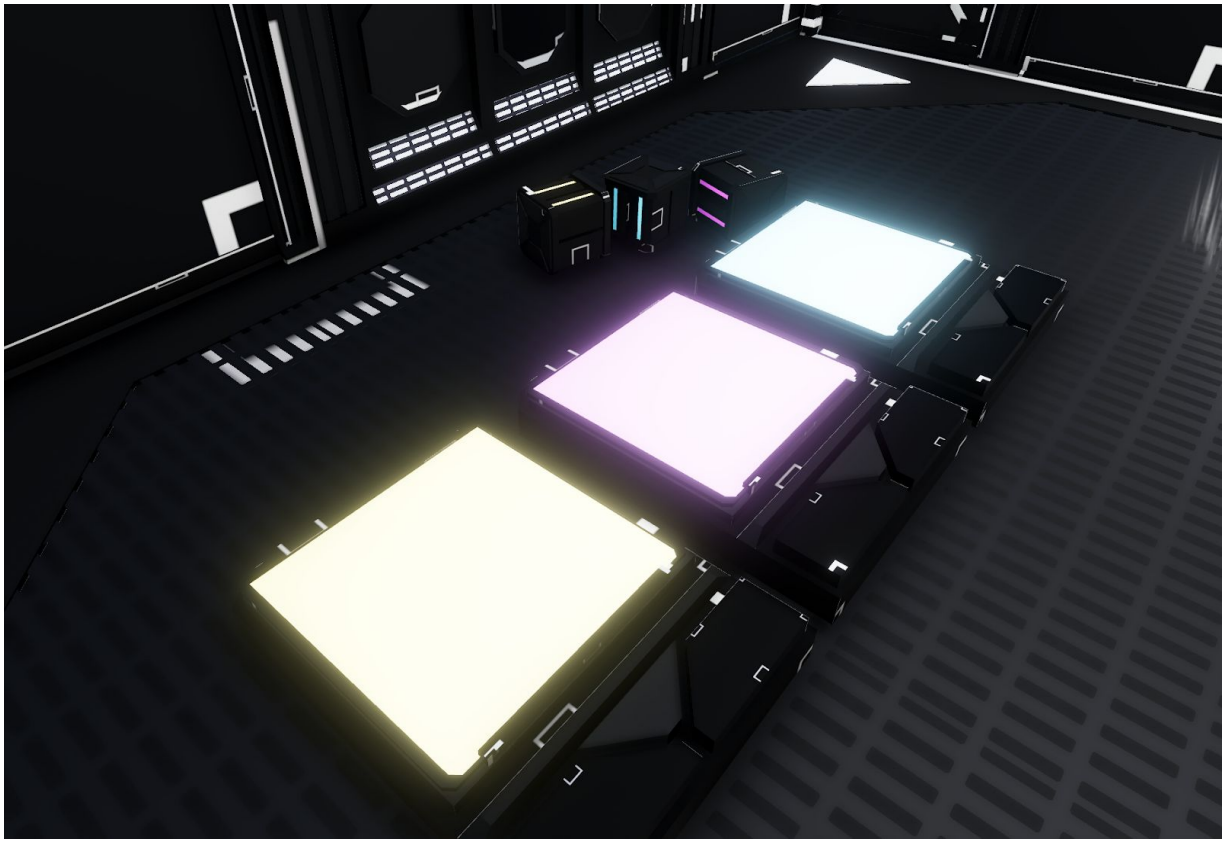

Figure 7. *Battery and Power Station.*

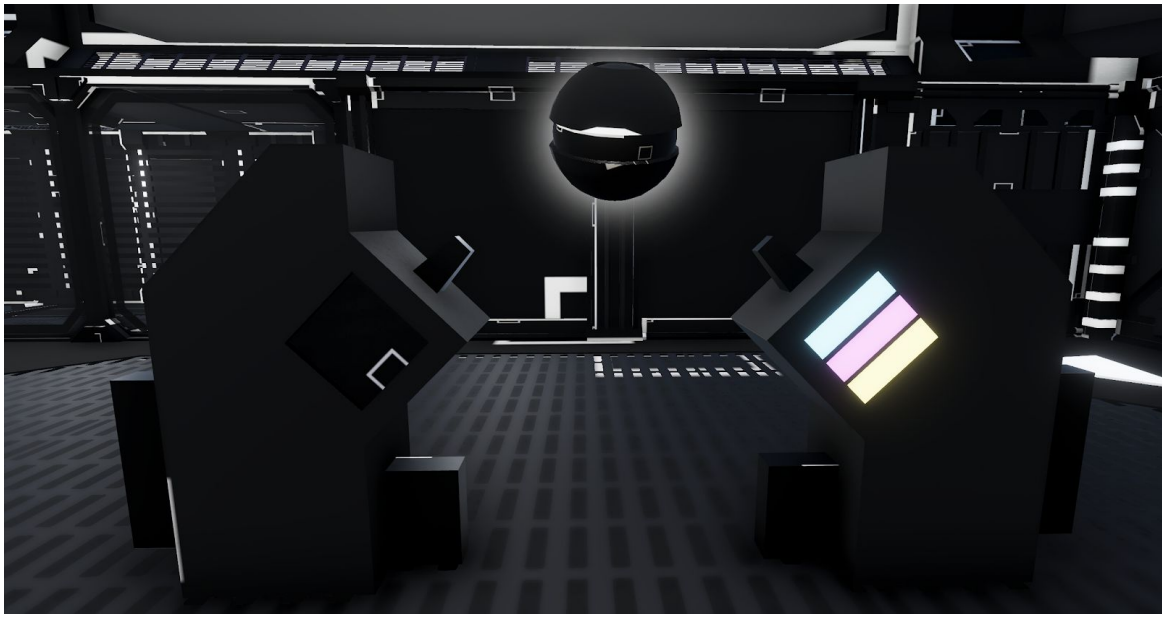

Figure 8. *Power Station with Absorbed Energy Order From Blue to Yellow.*

Figure 9. *Animation of Successfully Absorption.*

3.5.2.2 Summon

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 10). 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.

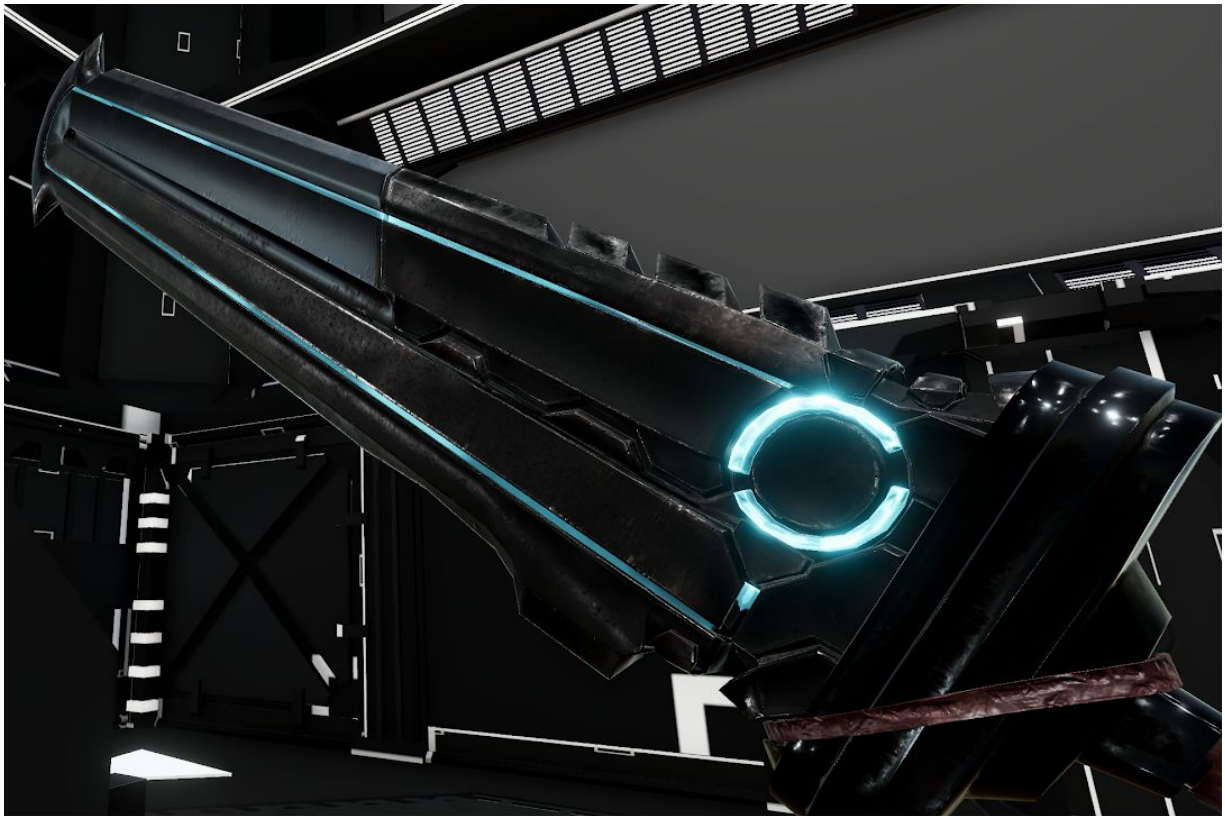

Figure 10. *Sword with Cyan Energy.*

3.5.2.3 Sword Skill

After summoned the sword, the player will be able to use the sword skill with the emitting color same as the emitting color of the sword. The player can hold down the trigger button of the right controller and move the controller like chropping the air. A sword skill will be activate and particles of the sword skill will go along the direction as same as the direction of the player's chropping motion (see Figure 11).

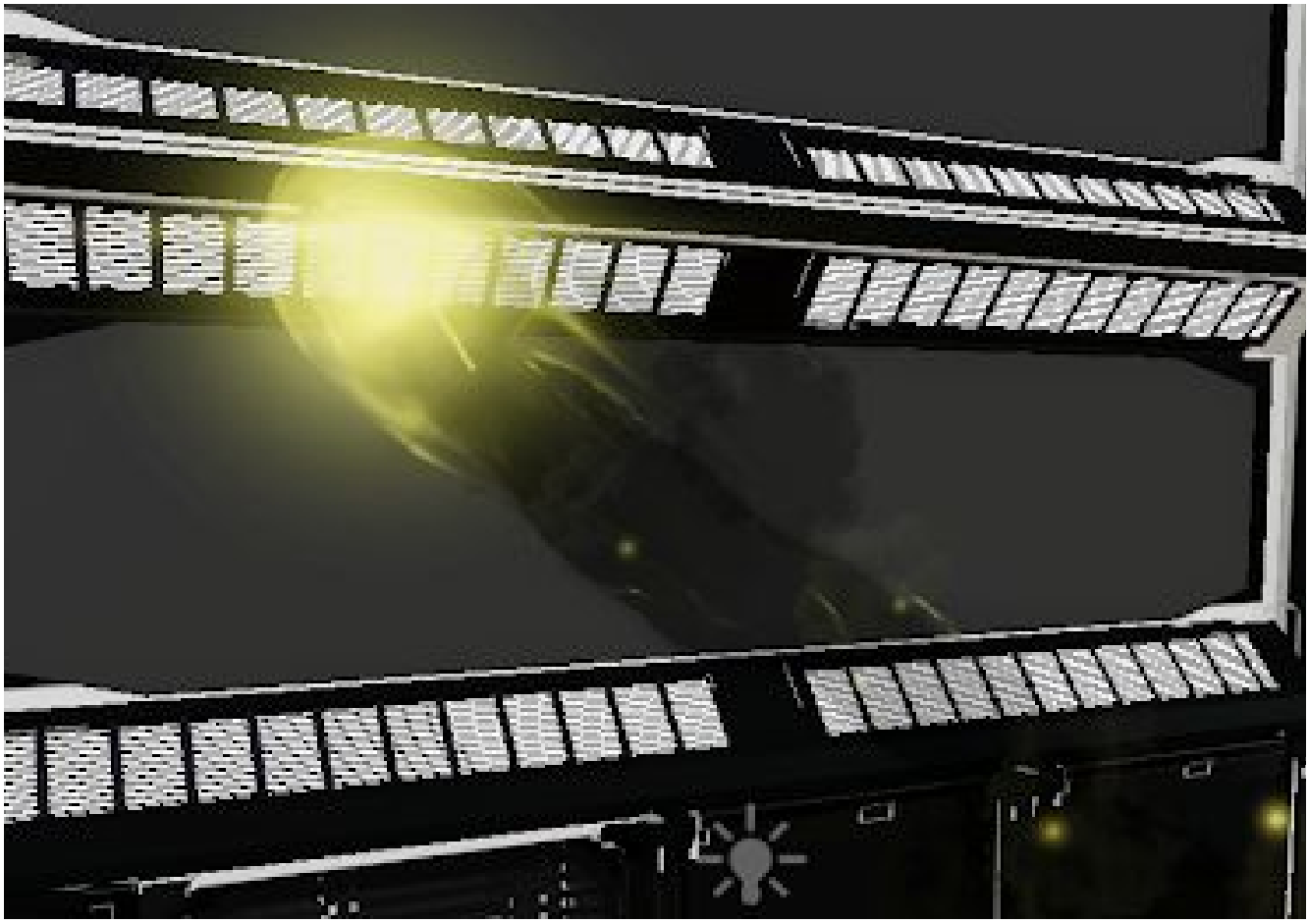

Figure 11. *Sword Skill with Yellow Emitting Color.*

3.5.3 Portal Feature

In order to provide an extra combination of Dream Driver's puzzle and difficulty. Dream Driver acquires parallel world idea when designing the puzzles. If there is a portal gate (see Figure 12) in the scene, it means that there is parallel world in that current scene. Therefore, the player cannot pick up the battery in the original world. In order to move the battery, the player needs to enter the parallel world through the portal gate. Moreover, parallel world does not allowed absorbing energy, break the blocking building and go into the exit. Thus, the player is required to move between the original world and parallel world in order to solve the puzzle.

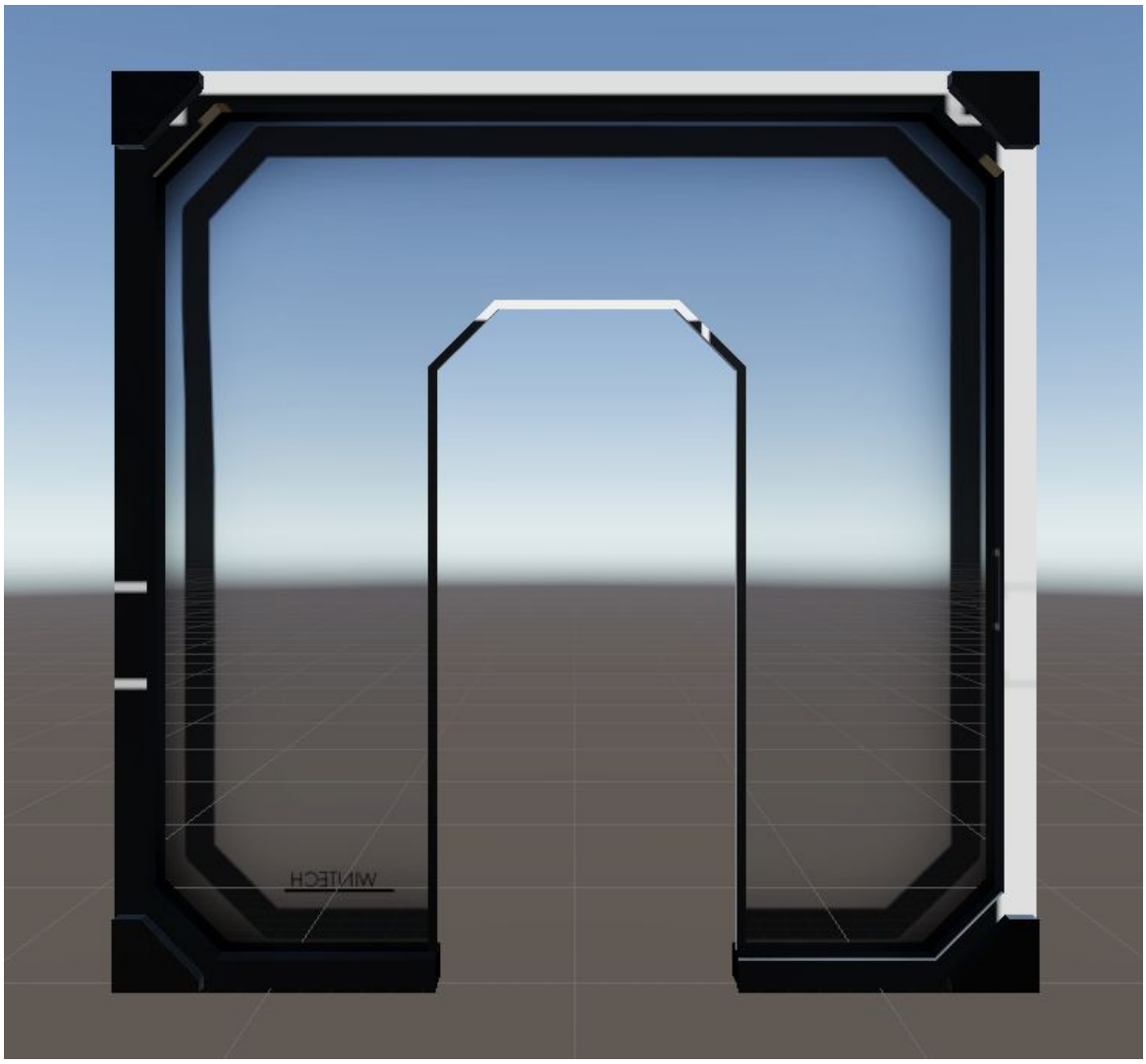

Figure 12. *Portal Gate.*

3.5.4 Auto Generating Puzzle System

To achieve extra visual enjoyment, Dream Driver will automatically generate a new puzzle for the player. Auto Generating Puzzle System provides four parameters for randomly generating. They are the position and the rotation of the batteries, the position and the rotation of the power stations, the position and the rotation of the portal gate, and the flapping angle of the whole scene. These four parameters can provide minimal visual enjoyment to the player when they keep playing Dream Driver. There is one more parameter for generating the puzzle, combination of the color puzzle. For each puzzle, there are three color required to absorb, summon and use sword skill in order to solve the whole puzzle (see Figure 13).

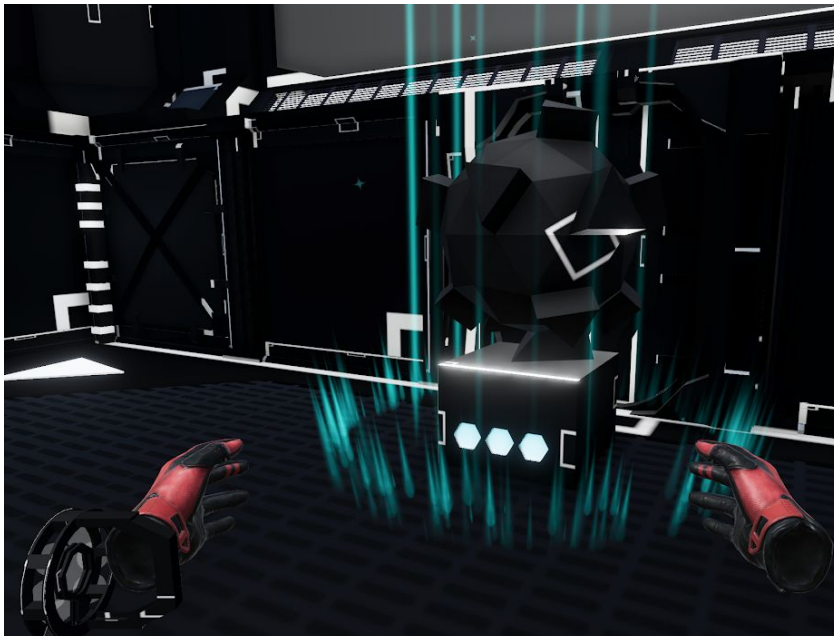

Figure 13. *Color Puzzle with Three Cyan Color.*

Moreover, there are seven different colors can be fit into the puzzle. They are Cyan, Magenta, Yellow, Red, Green, Blue and Black. Each color requires different batteries to solve. For example, Cyan needs Cyan battery, Blue needs Cyan and Magenta batteries, Black needs Cyan, Magenta and Yellow batteries. This color puzzle design is based on the CMYK color model. By calculation, the number of combination of the color puzzle is $7^3 = 343$. By adding portal feature, the number of combination of the puzzle of Dream Driver is 343 x 2 = 686. Auto Generating Puzzle System can greatly increase the visual enjoyment to the player by adding different small puzzles into the puzzle. Auto Generating Puzzle System generates puzzle based on the performance of the player by utilizing Machine Learning (see Section 3.8). The puzzle for the next generation will be easier when the performance of the player is likely not good. It means that the combination of the color puzzle will be simpler than previous generation. This feature can also maintain the balance of the gaming experience with the visual enjoyment. Generating a puzzle with the difficulty suitable for the player while also presenting the minimal visual enjoyment to the player.

3.6 Artistic Evaluation

Designing a game involves many things. For example, an innovative idea, implementation skills and artistic sense. Our team does not expert in art. Therefore, the basic scenes (see Figure 14) of every stages in Dream Driver are the same in order to minimize the workload on artwork. There is only slightly different between stages. In order to keep and present the minimal visual enjoyment, a reasonable amount of 3D objects will be added in each stage. The extra 3D objects are mainly the key items to solve the puzzle of that stage or the puzzle itself.

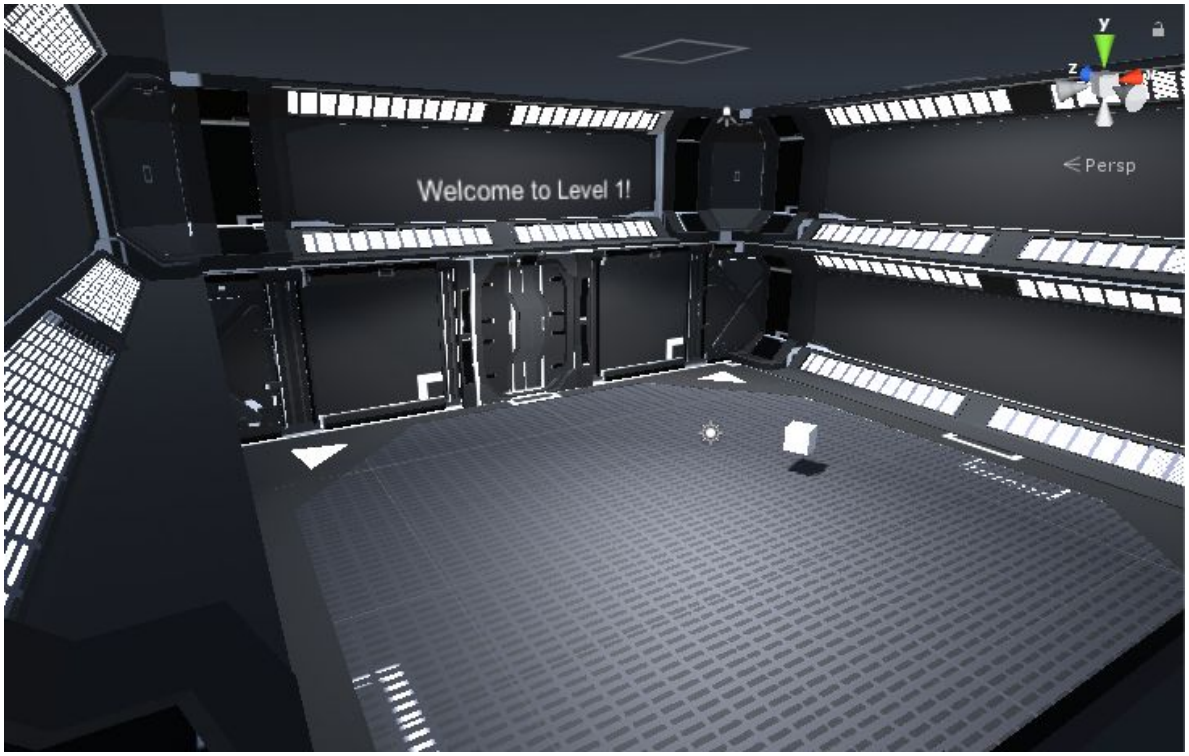

Figure 14. *Experiment Laboratory Scene in Game.*

3.7 Post-processing

Post-processing is employed to enhance the quality of the visuals of Dream Driver's game environment. Post-processing can apply filters and effects to the camera's image before show onto the screen. The color grading and blooming of the brightness are being used in Dream Driver. It can make Dream Driver's environment being dark and dull. At the same time, the keys and clues will draw extra attentions from the player because there are effects linking to them (see Figure 15 and Figure 16).

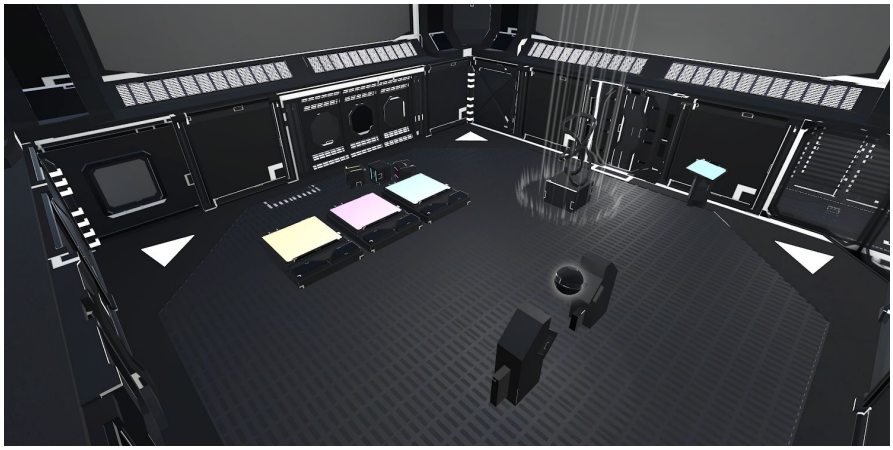

Figure 15. *Keys and Clues without Post-processing.*

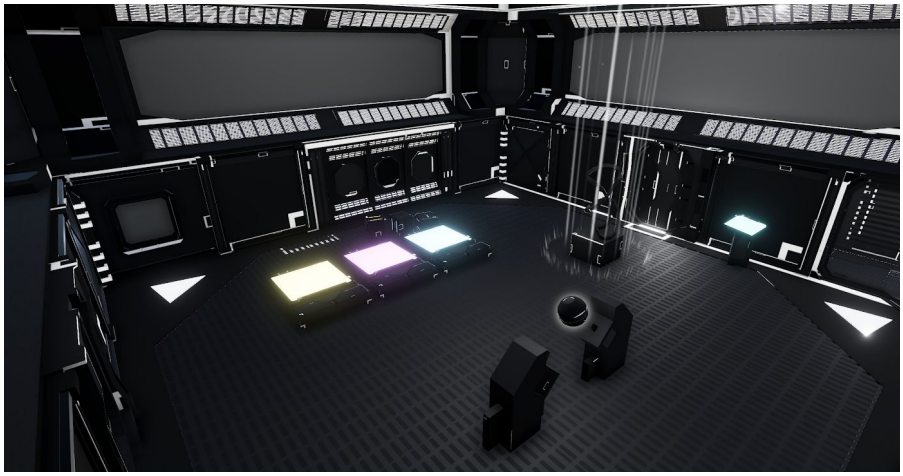

Figure 16. *Keys and Clues with Post-processing.*

3.8 Machine Learning

Machine Learning is employed to generate a model answer for every suitable puzzle in Dream Driver. Machine Learning is a branch of artificial intelligence which focuses on learning from big data. There are three different types of machine learning, unsupervised learning, supervised learning and reinforcement learning. In Dream Driver, Reinforcement learning is employed. Unity provides an open-source plugin, called Unity Machine Learning Agents Toolkit (ML-Agents Toolkit), which allows games and simulations to become the environment for training the system. The subsections below introduce reinforcement learning, ML-Agents Toolkit and how Dream Driver applies it.

3.8.1 Reinforcement Learning

Reinforcement learning is a system improves its performance based on the interaction with the environment. A system is also called agent, which is the machine that reinforcement learning aims to train. The system interacts with the environment by making an action to get the state and reward from the environment. Through the interaction with the environment, the system can learn a series of actions that maximizes the reward it gained from the environment via an exploratory trial-and-error approach (see Figure 17).

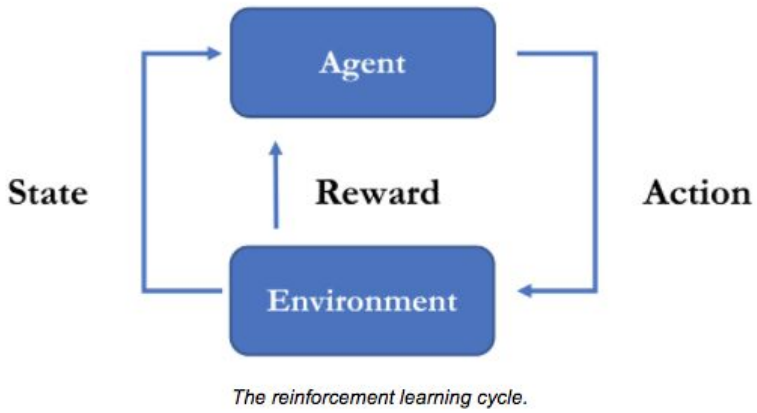

***Figure 17**. Reinforcement Learning [14].*

To achieve generating model answer for every puzzle in Dream Driver, reinforcement learning is the most suitable choice. Like the sections above, Dream Driver contains different puzzles for player to solve. Every puzzle is being generated automatically by the system based on the performance of the player. One of the method to define the performance of the player is to remember the actions the player took. Then generate the performance of the player by comparing the actions the player took and the actions the trained model took. From the definition of reinforcement learning, Dream Driver is needed a machine which can take a series of actions, is similar to the player, to solve the puzzle in Dream Driver. Thus, reinforcement learning is employed in Dream Driver.

3.8.2 ML-Agents Toolkit

The Unity Machine Learning Agents Toolkit, also called ML-Agents Toolkit, is an open-source Unity plugin which allows developer to train the system with games and simulation acting as the environment. The system can be trained using imitation, reinforcement learning, or other machine learning methods using Python API. ML-Agents Toolkit also provides implementations that ease the training process of the game. The trained system can be used in different areas, including controlling non-player characters' behavior.

ML-Agents contains three high-level components, Learning Environment, Python API and External Communicator. Learning Environment is the whole game environment, like Unity scene and all game characters. Python API contains all the machine learning algorithm for training. Python API is needed External Communicator to achieve communication between Unity and Python API because Python API is of part of Unity (see Figure 18).

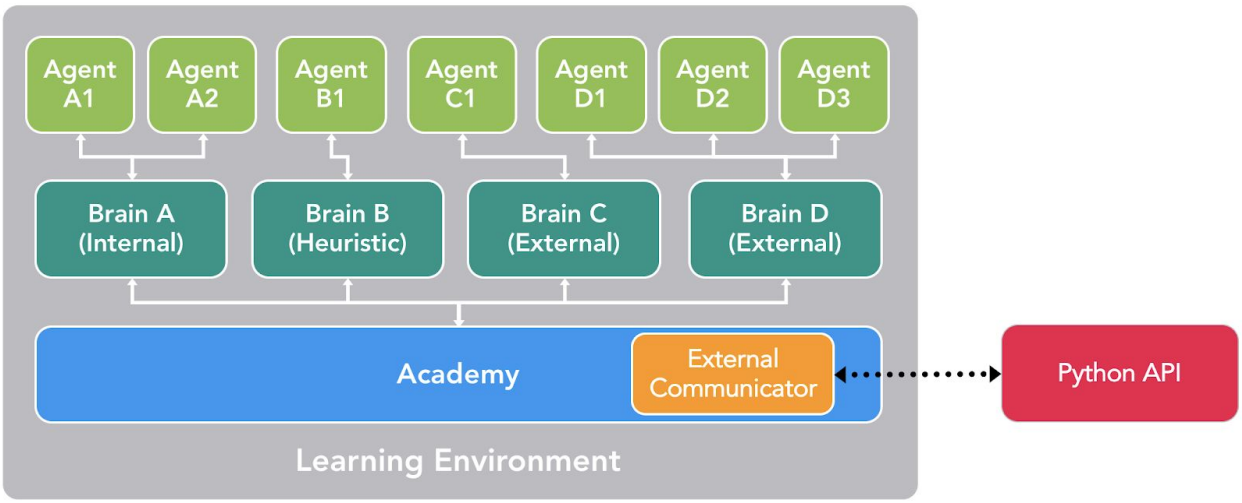

Figure 18. *ML-Agents Toolkit [15].*

Three additional components are employed in Learning Environment to organize the Unity scene, Agents, Brains and Academy. A Unity GameObject attaches an Agent to handle generating its observations, performing the actions received from the Brain and assigning a reward. One Agent is connected with one Brain. Brain is the component that receives the observations and the rewards from the Agent and return an action. Academy organizes the observation and decision making process. External Communicator is one part of the Academy (see Figure 18).

Each Learning environment always contains one global Academy and each character in the scene with one Agent. Also, each Agent must be connected with a Brain (see Figure 18).

For Dream Driver, the structure of the Learning Environment is simpler than the structure in Figure 18. Since Dream Driver is a single-player video game, there is only one Agent in the Dream Driver's Learning Environment which acting the player of Dream Driver (see Figure 19). In the training process, the Brain will conduct actions like the player and Agent returns the reward and state to the Brain. This process will be conducted many times until the training process finished.

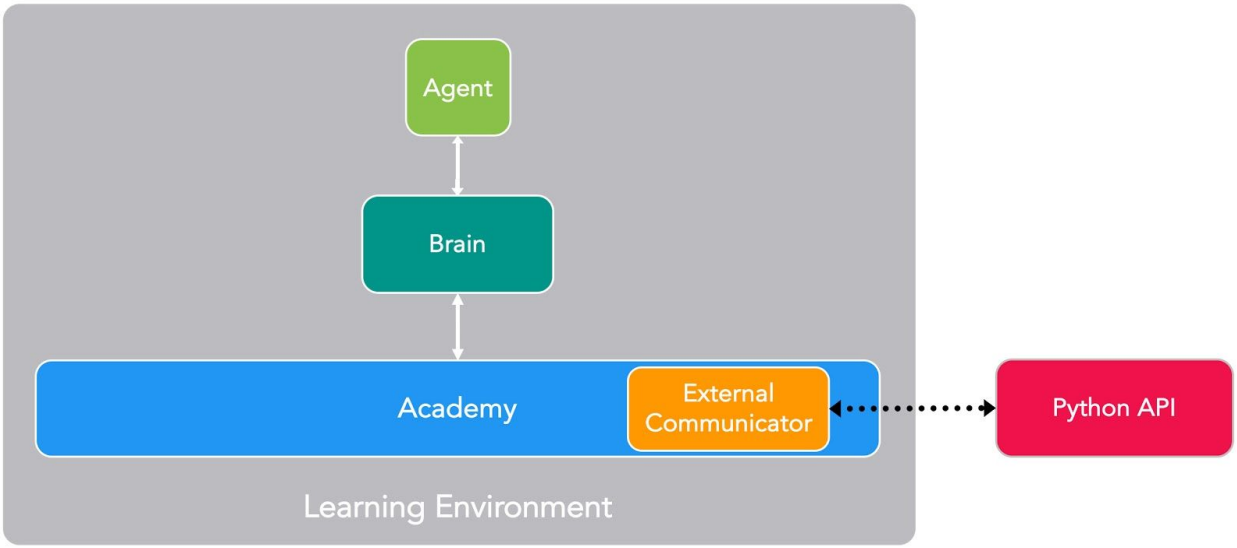

Figure 19. *Dream Driver's ML-Agents Toolkit [15].*

ML-Agents toolkit orchestrates the training using external Python training process. ML-Agents toolkit allows developer to modify the training config file which specifies the training method, the hyperparameters. There are 24 settings can be modified by the developer [16]. Dream Driver trains the system with Proximal Policy Optimization (PPO). PPO is a reinforcement learning technique which uses a neural network to generate the function that links the agent to the most likely action an agent can take in the given state of the environment. There are two important hyperparameters that greatly affect the learning result of the model, Learning Rate and Normalize. Learning Rate is the strength of each gradient descent update. For unstable training and stable rewarding system, the Learning Rate should be in low value. Higher Learning Rate increases the learning speed of the system while the risk that the system overshoot the global minimum (see Figure 20).

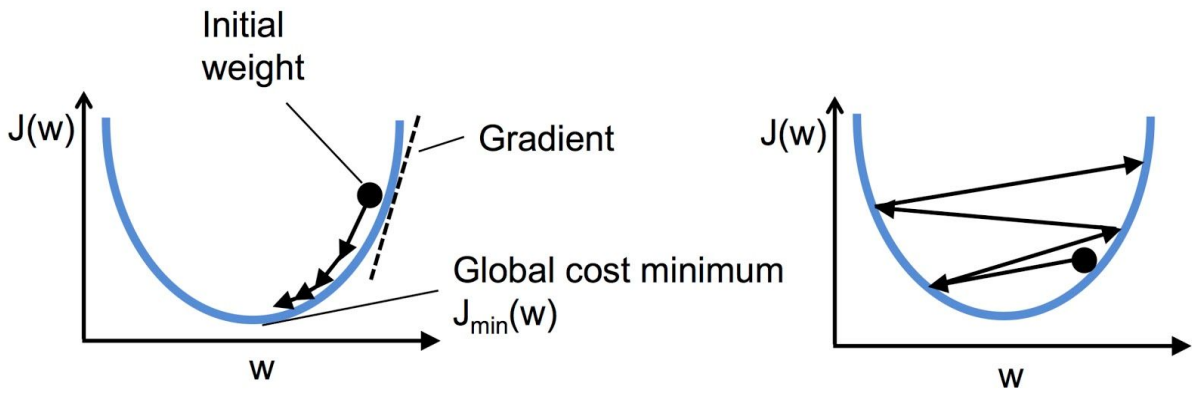

Figure 20. *Top with Low Learning Rate and Bottom with High Learning Rate [17].*

Normalize is representing whether normalization is applied to the observations. Normalization is one of the data preprocessing methods for feature scaling. It rescales the features to a range of [0,1]. The importance of normalize is to balance the contribution of all the features to the error. For example, the first feature is measured on a scale from 10 to 100 and the second feature os measured on a scale from 1 to 1,000,000. In this example, the second feature has a higher contribution to the error compared to the first feature.

3.8.3 Application

Dream Driver employes Machine Learning to generate the a model answer of every puzzle in it. Refer to the section SSS, Dream Driver generates a random puzzle automatically for player respect to the performance of the player. Dream Driver records every critical actions of the player, like player summons the sword, enters the portal gate. Before Dream Driver generates a new puzzle for the player, it compares the series of actions of the players and the series of actions of the system, which is the trained system using ML-Agents Toolkit. Therefore, the performance of the player for that puzzle can be calculated and used to decide the difficulties of the next puzzle.

The structure of Dream Driver's Learning Environment is shown in Figure 19. It contains one global Academy which contains External Communicator, one Agent links to one Brain. Dream Driver employes ML-Agents Toolkit with PPO training method to train the system. The training contains $1.0e^{7}$ steps with Learning Rate $1.0e^{-4}$ and applying normalization to the observations.

From the Figure 21, the training details of each training step is Agent gives one action from eight different choices. The environment will return all nine observations and rewards based on the current state of the environment. If the Agent choose a correct action at the current state, the environment returns 0 reward to the Agent. If the Agent solved the whole puzzle, the environment returns 1 reward to the Agent. If the Agent choose a wrong action, the environment returns -1 rewards. For example, the Agent choose to Go to the exit option while the color puzzle is not solved. The environment will return -1 reward to the Agent as publishment and encourage the Agent to not choosing the same action with the same observation.

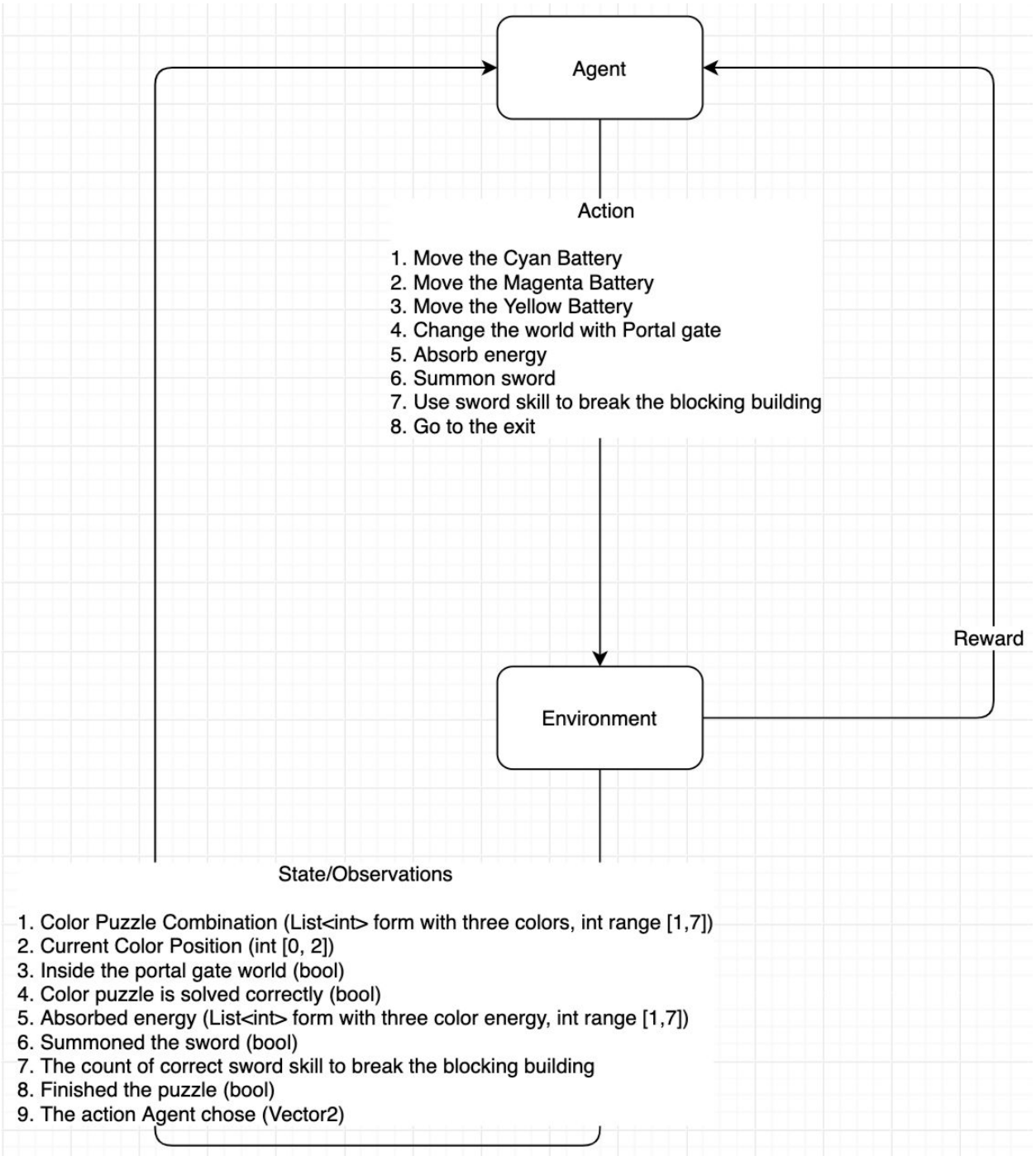

Figure 21. *Training Details of Each Training Step.*

To have a better learning result, the initial state of the training is not completely random. There is a List of Integer which contains all the combination of the color puzzle. For the color puzzle, there is always three color in it. Moreover, there are seven colors available. Therefore, the number of all the combination of the color puzzle is $7^3 = 343$. There is also one parameter which affects the initial state of Dream Driver, exist of the portal gate. If there is a portal gate, the player and Agent need to go into the parallel world in order to pick up the battery. Therefore, the total number of different combination of the initial state of Dream Driver is 686. In order to make sure each initial state of Dream Driver is existed in the training progress, simple random number cannot be ensured. Thus, the training process will shuffle all the combination of the initial state and train all the initial state before the next shuffle of all the combination of the initial state. In other word, the training process first shuffle all 686 initial states. Then finish all the 686 initial states' trainings. Next, shuffle all the 686 initial states and repeat the training process. If applying completely random initial state, there is chance that some initial states are not involved in the training progress which leads the system is not accurate.

ML-Agents Toolkit also provides summaries of each training using TensorBoard. The summaries provides the informations indicating that the training is an effective training or not. The Figure 22 is the summaries of Dream Driver's training. At the beginning of the training, the Agent choose an action randomly. Therefore, it chose many wrong actions which leads the cumulative reward is low at the beginning. After 1.0M steps, the Agent learnt that to have a series of action which have around 0 cumulative reward. It shows that the training process is effective and the system is learning from the previous rewards, observations and actions. If we have a closer look of the summaries, the performance of Dream Driver's system is actually frustrated. The reason behind is the system is fine tuning during each step of the training. There are 686 different initial states for Dream Driver. The system is finding the way to find the best function to choose the best action to that state.

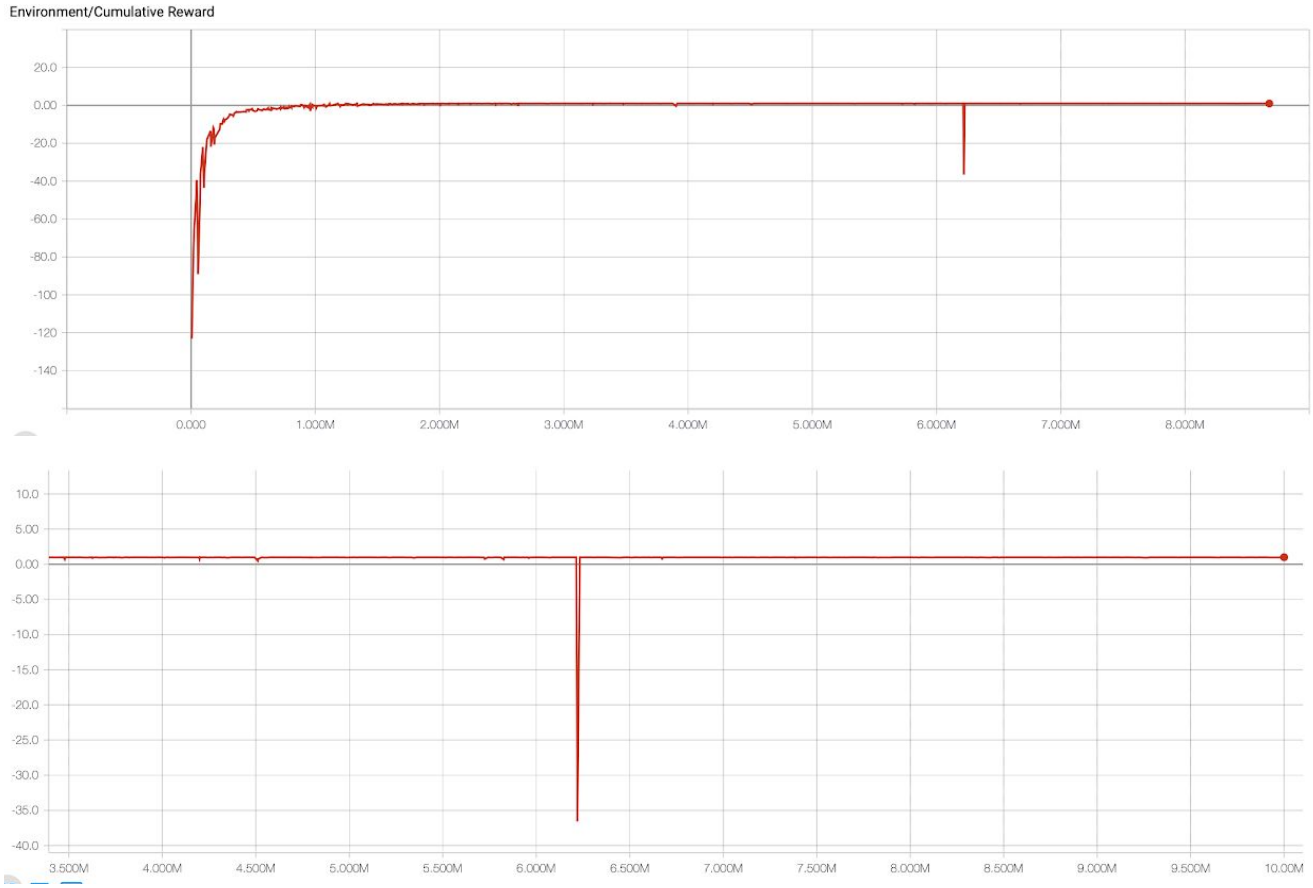

Figure 22. *Summaries of Dream Driver's Training(Top) Closer Look(Bottom).*

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.

Conclusion

This final report introduces a new PC game on Windows Platform which integrates with VR technology, open-ended game design and the conception of dream, Dream Driver. The goals of this project are to develop a game utilizes the experience of VR to recall player's creativity inside their dream.

All the features of Dream Driver have been implemented. The game can now support the players to move by gesture, doing three actions with Dream Driver, portal between places. Moreover, Dream Driver ables to automatically generate puzzle based on the performance of the player by utilizing the Machine Learning.

We will consult our supervisor, Dr. Chim, to seek for comments and advices. Enhancing Dream Driver and the gaming experience will be focused in the future work. Dream Driver is hoped to be fully developed so that it can be promoted to the gaming market in the future.

References

1. No Break Games. (2015). Human - No break games. [Online]. Available: http://www.nobrakesgames.com/human/. [Accessed: Apr 14, 2019].
2. The Orange Box. (2007). The Orange Box - Portal. [Online]. Available: http://orange.half-life2.com/portal.html. [Accessed: Apr 14, 2019].
3. Laver, K. E., George, S., Thomas, S., Deutsch, J. E., & Crotty, M. (2015). Virtual reality for stroke rehabilitation. Cochrane database of systematic reviews, (2).
4. Mujber, T. S., Szecsi, T., & Hashmi, M. S. (2004). Virtual reality applications in manufacturing process simulation. Journal of materials processing technology, 155, 1834-1838.
5. Zyda, M. (2005). From visual simulation to virtual reality to games. Computer, 38(9), 25-32.
6. Lucas Matney. (2016). 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: Apr 14, 2019]
7. Daniel Love. (2016). 5 Reasons Lucid Dreaming Is Better Than Virtual Reality By Daniel Love. [Online]. Available: https://www.world-of-lucid-dreaming.com/5-reasons-lucid-dreaming-is-better-than-virtual-reality.html. [Accessed: Apr 14, 2019].
8. FIGHT4DREAM LIMITED. (2016). FIGHT4DREAM. [Online]. Available: https://fight4dream.com/. [Accessed: Apr 14, 2019].
9. Unity Technologies. (2017). Unruly Heroes - Made With Unity. [Online]. Available: https://unity.com/madewith/unruly-heroes. [Accessed: Apr 14, 2019].
10. HTC Corporation. (2018). What are the system requirements? - Vive. [Online]. Available: https://www.vive.com/hk/support/vive/category_howto/what-are-the-system-requirements.html. [Accessed: Apr 14, 2019].

11. Richard Fine. (2017). UnityScript's long ride off into the sunset – Unity Blog. [Online]. Available: https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset/. [Accessed: Apr 14, 2019].
12. Unity Technologies. (2017). Vive - Asset Store - Unity Asset Store. [Online]. Available: https://assetstore.unity.com/lists/vive-14875. [Accessed: Apr 14, 2019].
13. GitHub, Inc. (2018). GitHub - ViveSoftware/ViveInputUtility-Unity: A toolkit that helps developing/prototyping VR apps. [Online]. Available: https://github.com/ViveSoftware/ViveInputUtility-Unity. [Accessed: Apr 14, 2019].
14. GitHub, Inc. (2018). ml-agents/Background-Machine-Learning.md at master · Unity-Technologies/ml-agents · GitHub. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Background-Machine-Learning.md. [Accessed: Apr 14, 2019].
15. GitHub, Inc. (2018). ml-agents/ML-Agents-Overview.md at master · Unity-Technologies/ml-agents · GitHub. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md. [Accessed: Apr 14, 2019].
16. GitHub, Inc. (2018). ml-agents/Training-ML-Agents.md at master · Unity-Technologies/ml-agents · GitHub. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-ML-Agents.md#training-config-file. [Accessed: Apr 14, 2019].
17. Pavel Surmenok. (2017). Estimating an Optimal Learning Rate For a Deep Neural Network. [Online]. Available: https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0. [Accessed: Apr 14, 2019].