Developing a Single-Player Sci-Fi First-Person Shooter in Unity3D

Keang Lyu

Faculty of Innovation Engineering, Macau University of Science and Technology, Avenida Wai Long, Taipa, Macau, China

creeperlv@ieee.org

Abstract. First-Person Shooter is currently one of most popular game genres. However, most of the shooters are realistic and multiplayer oriented. Besides, there's no single-player oriented FPS framework made for Unity3D which remains a small blank in Sci-Fi single player FPS game area in recent years. So, this paper developed a game named What Happened to Site-13? using Unity3D which is a realistic Sci-Fi FPS which is single-player oriented. The game contains a gamepad-oriented FPS Controller, Story-telling module, AI Module, an in-game map editor and scene with realistic sci-fi art style. Through tests with testers, to test if the game is immersive, and the combat loop is comfortable, the visuals meet expectations. After tests and validation, it was proved to achieve target art style, visual experience and combat loop, it also was proven to be immersive. Also, it can serve as a framework for FPS games after validation with a group of students.

Keywords: Unity3D; Single-Player Sci-Fi First-Person Shooter; A Realistic Sci-Fi FPS; Game Framework.

1. Introduction

First-Person Shooter is a type of games focused on guns and weapon-related combats in a perspective of first-person in 3D scene. Currently, in First-Person Shooter (FPS) game industry, most games are realistic or retro style according to which background they are based on, World War I, World War II, or modern wars. For those sci-fi FPS games, only a few of them are setting the background to the near future. The count of games based on SCP Foundation [1-2] theme is very small and they neither first-person but not a shooter game nor an FPS but have no single-player story.

Researching on FPS is trying to reach out to broader combat and visual experience which can potentially increase the attractiveness to players. In the meanwhile, current open-source FPS implementations for Unity are either designed for multiplayer or do not build for story-based single-player campaign such as: FPS Microgame [3] from Unity3D, Simple FPS Controller [4] and so on. Therefore, there is a blank to fill for open-source single-player, story-oriented FPS framework.

All in all, this paper aims on creating a single-player FPS game with a realistic sci-fi art style, providing a control scheme and combat loop optimized for gamepads. The also game aims on providing and evaluating new combat logic which is inspired by combat logic of Halo [5] with heavy modifications. In the meanwhile, the game also come with a map editor and a customizable PvE mode to enhance player's freedom. The game is based on the world of SCP Foundation, especially SCP-1730 [6]. This game is made with Unity3D, MakeHuman [7] and a lot of tools created to complete the game. In addition, this paper also aims on providing an FPS framework that future FPS game can be developed using work from this paper.

2. Related Work

Currently, there are some Sci-Fi oriented FPS games in the market, e.g.: Halo Series, Overwatch [8], Apex [9] etc. However, only a few of them have single-player story. Then only a few of them have the art style combining modern construction. Halo Infinite [10] is a game that combine realistic look and feel when it comes to human faction and far future Sci-Fi look when it comes to alien faction structures. However, Halo Infinite is not based on near future and the Sci-Fi art style is design to

match alien faction that they look obviously cannot be made by human. This remains a blank that allow this paper to implement a realistic human Sci-Fi art style.

Halo Infinite takes an implementation to Sci-Fi art style by making alien structures mega-structures that human cannot build, in the meanwhile, adding light bars to outlines builds up the sci-fi feel. Also, it made metal structures super-reflective that they are almost mirrors. Also, the design of Sci-Fi structures is very angular with alien-like metal stamps on them. Halo 2 and Halo 3 implemented Behavior Tree to make the AI behaviors more modular and flexible also maintain the complexity of behaviors to achieve a clean, scalable and customizable brain architecture. [11] [12] Behavior Tree is used to achieve similar goals in this paper.

For SCP Foundation related games, there are currently a few games such as SCP: Containment Breach [13] and SCP: Secret Laboratory [14]. But SCP: Containment Breach is technically not an FPS game: it utilizes first-person control scheme but with not shooting functionality. Also, SCP: Secret Laboratory is indeed an FPS game, but it is a pure multiplayer FPS game with no single-player experience until the paper is written.

3. Methods

3.1 Scene Design

The goal of the scene of the game is to present players a hybrid art style: combining modern facility and near future sci-fi research/military style. Thus, the office part of the scene is inspired by university hallways, then rooms and hallways for the high-end devices are inspired by modern semiconductor factories.

Several permanent practice effects were used to build up the sense of disaster in the scene. In the meanwhile, a shader was created to simulate the effect of a screen receiving incomplete signals. **Figure** 1 is the real-time screenshot for a scene that contains high-end devices. The device is designed to be large, and so the corresponding room is also large. Overall, the device is used to travel between alternative universes, in order to complete this function, it is assumed to consume large amount of computation power to calculate space and time coordinates and something else, so, a large amount server-like modules are added to the room and attached to the wall to make it fell more futuristic and sci-fi. Also, some texture details are added to make the overall room feel both realistic and sci-fi at the same time as shown in Figure 2 and Figure 3: dirty metal as main texture with angular, tiled triangle normal map and high reflectiveness.

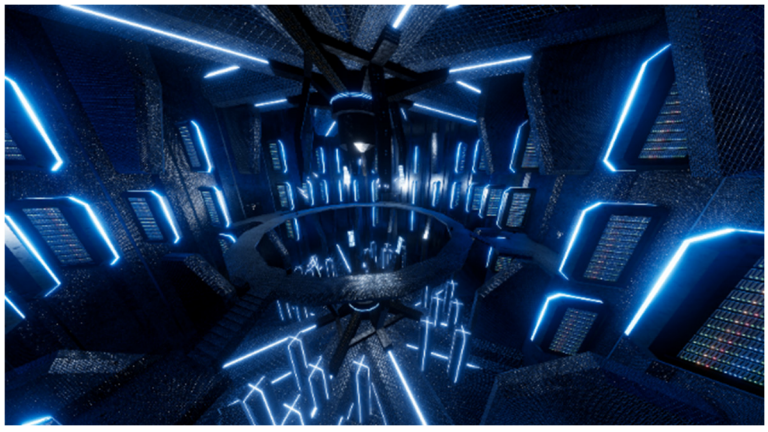

Figure 1. Sci-Fi Art Style

For most realistic part, the office zone of the scene is the most realistic style part. Figure 4 is the screenshot for a hallway inside a facility, to make the hallway feel realistic, pipes are added below the ceiling, the material used for wall is concrete. A few paint decorations were applied near the floor

to make it feel more like a facility with some text Site-13 on it. To improve realistic feelings, many modern props are added to the scene as shown in Figure 5

Figure 2. Diffuse Map for the base material.

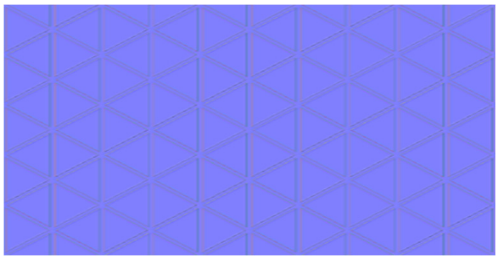

Figure 3. Normal Map for the base material.

Figure 4. Office Zone hallway

Figure 5. Many modern props

3.2 World Building

The world building of the game is purely done with ProBuilder, a tool shipped with Unity3D. This tool is easy to learn and use, building a room/hallway with this tool only needs almost one or two hours. In the meanwhile, using ProBuilder allows viewing the final visual presentation in game's renderer with post-processing.

In the meanwhile, to make the world feel more realistic, many modern props are made: a laptop based on real 2-in-1 laptop (shown in Figure 6); wall socket with unusual port; common TVs, common surveillance cameras (showed in Figure 7) and so on. To enhance visual reality, an OLED-Like shader is made to simulate LED screens as shown in Figure 8.

Figure 6. Laptop based on real world counterpart

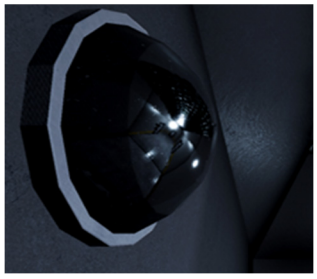

Figure 7. One of Common Surveillance Cameras

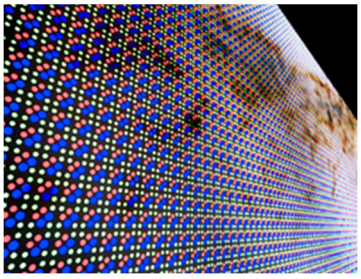

Figure 8. Example of OLED Shader

3.3 Weapon Modeling

Models of weapons in the game are purely made with ProBuilder. With ProBuilder, making a model for a weapon only takes up almost one or two days. The goal here is to make realistic and futuristic weapons with less faces possible to achieve high performance in a scenario which a lot of weapons will be drawn in screen (Figure 9-figure 10).

Figure 10 shown a combination of realistic and sci-fi design, it uses black as main body color like many modern rifles do. However, to make it more futuristic, the shape of front part of the body is designed to not be flat as a method to reduce the realistic feel. Also, introducing a glowing ammo counter make it has a high-tech feeling. Figure 9 shown a more realistic design, compare to the one in Figure 10 it has 2 more handles which is more like modern rifles. It also has many parts that modern rifles have like the rails on the top which is missing in the Figure 10 one.

Figure 9. Side and Oblique view of the rifle

Figure 10. Side and Oblique view of the rifle

3.4 Character Modeling

Models of human characters are made with MakeHuman. This tool allows non-professional artist to create high-quality human models. In fact, in this game's scenario, making all characters only used 5 hours. However, the armors are modeled using ProBuilder to save time.

Creating a armored solider need 2 steps: 1) make a model in MakeHuman as shown Figure 11; 2) make a armor with ProBuilder or choose an armor combination as shown in Figure 12.

Figure 11. A man with tech suit

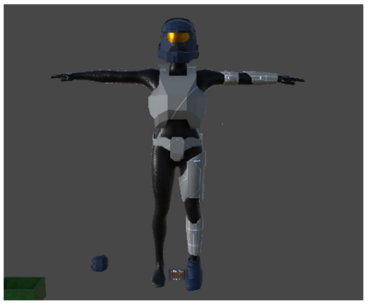

Figure 12. Armor made with ProBuilder

3.5 Textures

One goal of the game is that when it is ready, the textures can be seamlessly scale-up to fit the visual needs of 8K or even 16K gaming. So, a tool named Scalable Relative Image is created to generate textures according to given size.

For base textures like metal, plastic, the textures are from an asset website named AmbientCG which only provide assets licensed under CC0 License. With things above, this game can be easily converted to next-gen visual with only a few modifications to build configurations. Figure 13 shows a texture made with Scalable Relative Image; it shows the diffuse map which can be scaled up seamlessly without being blurry as regular pixel images do by adjusting the target resolution when building textures. The green wireframe in the figure is the UV reference image generated by ProBuilder in Unity.

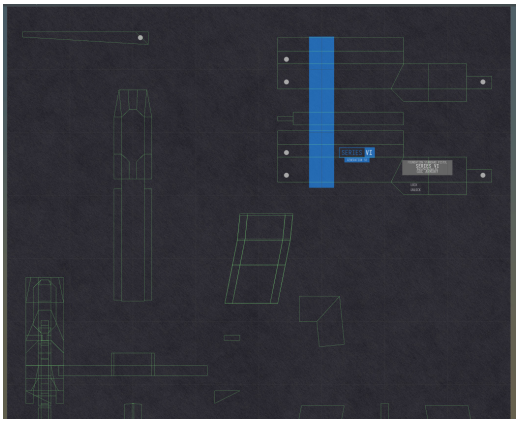

Figure 13. A texture made with Scalable Relative Image for a pistol

3.6 Story Scripting

Story scripting in this game consistent of two parts: script and director. The script is in fact a JSON file with a couple of objects whose classes are derived from EventBase class. The director is in fact a interpreter to the list of EventBase objects.

A tool named CampaignScript is created to create and edit scripts. This tool is completely independent from Unity3D and uses AvaloniaUI framework, making this tool completely cross-platform. The tool with Scriptable Director makes the scripting of the scene flexible, extendable, and more efficient. Making the events completely irrelevant to Unity3D makes it easy to be serialized to JSON, also allows future features such as allowing users to make their own levels.

3.7 AI Behaviors

The NPCs in this paper uses behavior trees to define behaviors which can increase modularity to the game [11]. A library named fluid-behavior-tree is used, which is licensed under the MIT License. However, in order to make the behaviors more customizable and not bind with C# code, a tool named BTNodeEditor is created. The tool allows developer to edit the behavior outside of Unity and even outside of C# code. With the tools and libraries, the NPC in this game is able to run on an ECS-Like system.

Figure 14 shows a behavior tree for one of enemies which is a worm. The tree describes its behaviors: when spotted enemy (which is player in this case), change the behavior mode to combat; when the enemy is in attack range and the behavior mode is combat, self-destruct (the death animation will cause damage in this case); when the enemy is out-of-sight, set the behavior mode to goal to continue goal given when spawning. Using the tool and behavior tree allow the behavior of a certain AI can be tweaked without directly touching the codes. It can even serve as a mod tool for players if it was made publicly accessible and add a loader to load external behavior tree.

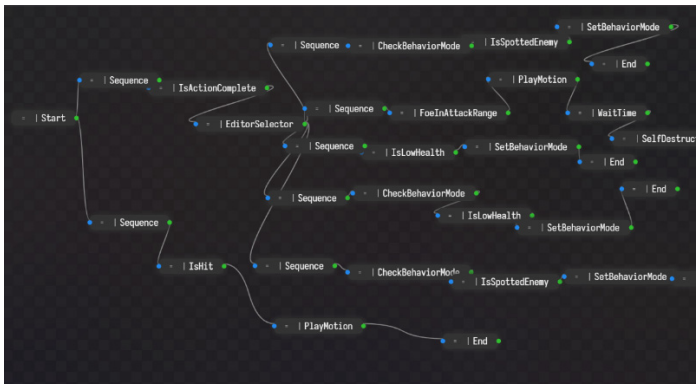

Figure 14. A behavior tree for one of enemies

3.8 UI Design

This paper made most user interface controls modular and replaceable. This paper aims on achieving a smooth user experience which means a heavy use of animations. Also, due to future development need, all UI parts are designed to be modular and made to a prefab which can be replaced easily. This means the art style of UI can be changed easily.

Figure 15 shows a main menu, which all buttons are a prefab which can be swapped easily. Figure 16 shows a level selector menu, the menu is generated from the definition to levels and can be easily modified with the embed inspector from Unity3D. The animation of changing menus is implemented by calculating the scale and alpha from time factor, so the menu is smooth when the game is running with high framerate.

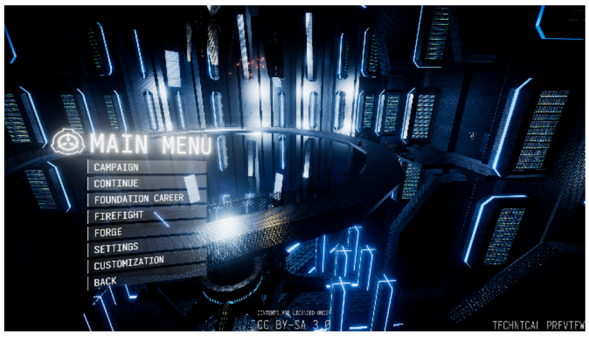

Figure 15 Main Menu

Figure 16 Level selection menu

Since UI is also a part of storytelling. According to a research, interfaces can be used to enhance storytelling. HUD is designed to be able be trigged by level script, and HUD changes as the story progresses. Figure 17 and Figure 18 shows the change of HUD as the level progresses.

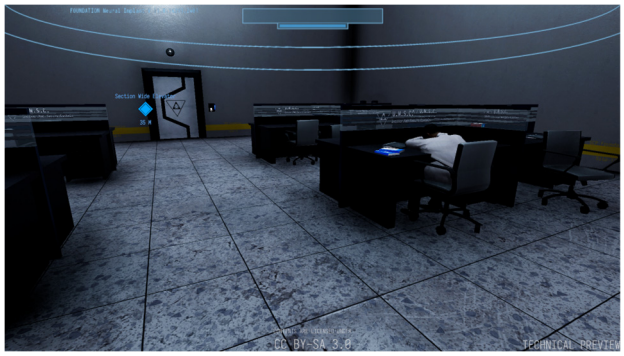

Figure 17. HUD in early level

Figure 18. HUD in later level

3.9 System Implementation

The combat loop is a modified version of golden triangle: silver quad, which adds equipment to the loop, player must deploy equipment before/during/after a combat to achieve high-efficient damage producing and low damage taking. With this theory, combat is separated into two parts: damage dealing and resource management more equally. In golden triangle, the only actual part of resource management is managing ammo and grenades, however, damage dealing consists of three parts: shooting, grenade explosion and melee attack. With equipment introduced, players must manage more resources: health and shields in some configurations.

To achieve the immersive experience in this game, a type of camera named Immersive Camera was created to implement no border transformation from FPS camera to cutscene camera. Also, a global HUD is implemented which is not strongly bind with player object, which means the HUD can be shown during the cutscene without the presence of the object of player. To allow this project

to be a framework for other FPS games, this game was designed to be modular. Most of the systems can be removed without crashing the game.

Most of the functions are made prefabs: HUD is a prefab, resource definitions per level are prefabs, systems used are prefabs and even input processor is also a prefab. This means each part can be independently modified and most of them can be carried to other projects. Performance is very important to FPS games, so, optimizations on codes are necessary. According to a research, heap allocations should be reduced as much as possible. Also, it is noted that some language features (e.g., LINQ) will also lead to temporary heap objects after compilation. So, LINQ APIs are avoided in codes and many global instances are used to avoid the time of querying components.

4. Results

The game provides a robust single-player gameplay. Eventually implemented a sci-fi scene with realistic visual art style. The gamepad-oriented gameplay is also implemented. In general, the game performs well: it can achieve around 45 FPS on a low-end laptop with Intel(R) Core (TM) i7-8650U CPU and a discrete GPU which is GTX 1060M in a resolution of 1080P. It meets the target framerate of Halo 3 and Halo Reach which was a popular FPS game on console in early 21st century.

On a regular medium-tier gaming PC with Inter (R) Core (TM) i7-11700 CPU and a RTX 3060 in a resolution, it can reach 600 FPS high, average 300 FPS and 100 FPS low which clearly meets the performance goal as shown in Figure 19, Figure 20, Figure 21 and Figure 22. The data is recorded via MSI Afterburner. From the figure we can spot 4 major performance drops, the first shot drop which reaches 0 FPS is the loading process of the level, during the benchmark process, the game was running on a hard disk which is relatively slow, the first spike is the loading of "loading scene" of the level which will load some scripts which will indicate what to load in the second stage. The second drop is the real loading process of the level, also, it was caused by the loading process it lasted longer than the first drop for the scene to load is significantly larger than the "loading scene" which will load a lot of 4k resources and high poly meshes. The third framerate drop is due to at that time, player is at a scene which contains 13 high-resolution reflection maps which needs to be generated during the runtime. The fourth drop was caused by going near the end of level where have 4 high-resolution reflection maps need to be generated during runtime.

All in all, performance drop can be generalized to 2 problems in this benchmark: i) Loading the level; ii) Generating reflection map during runtime.

For first problem, this behaviour is good for fewer framerates drop due to loading resource but will cause player to wait longer, however, since this is an FPS game, smoothness and framerate is more important to players rather than fast load time.

For second problem, this problem is due to the internal implementation of Reflection Probe from Unity3D, when the type is set to real time and refresh mode is set to On Awake, the generation will only happen when the probe box falls in camera viewport. Using real time type can reduce size of built binaries besides, however, it will cause performance drop, so in the future, some of the probes must be replaced with baked probes.

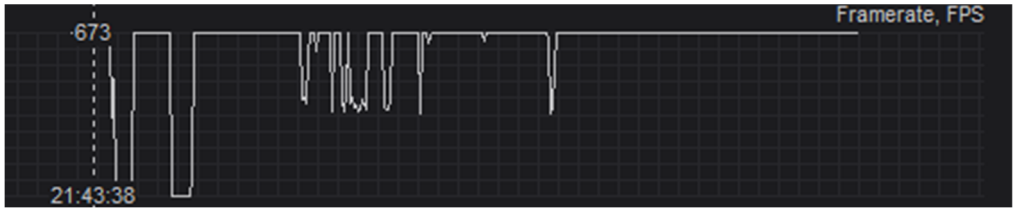

Figure 19. At start, the menu, reached 673 FPS, the zero framerate is caused by loading map from hard disk.

Figure 20. During the mid of a level, stable around 300 FPS. The instable is caused be massive high-quality plants with water.

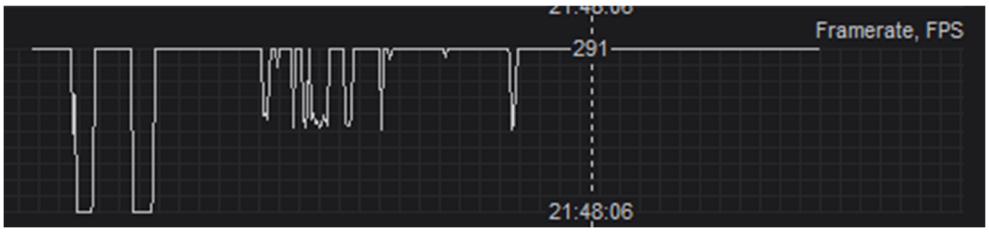

Figure 21. At the end of a level, in a reflective area heavily used reflection map, reaches 291 FPS.

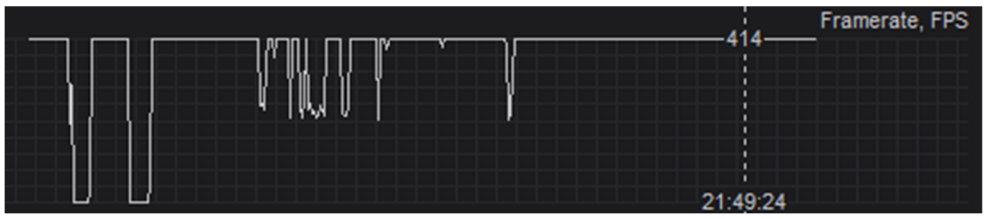

Figure 22. During a cut scene with massive scenery, the game reaches 400 FPS.

To verify the possibility of using the project as a framework, the code of the project is fully open sourced on GitHub licensed under the MIT License and invited a group of students to develop the game using this game's codebase which is reportedly successful. To evaluate the effect the game objectively, a copy of 0.2.90.0 version was sent to some players and invited them to perform an anonymous evaluation survey.

Table 1 shows one of collected feedback form from users which contains the score given by user and explanations to each grade.

Table 1. One of collected feedback form

Site-13 User Experience Feedback Form			
Experience Item	A	B	C
Combat Experience	✔		
Sci-Fi Visual	✔		
Realistic Visual		✔	
AI Responsiveness	✔		
Immersive Level	✔		

Table 2. Evaluate score from testers

Grade	Combat Experience	Sci-Fi Visual	Realistic Visual	AI Responsiveness	Immersive Level
A	4	5	2	4	3
B	1	0	2	1	2
C	0	0	1	0	0

For combat experience, 4 testers gave a grade of A, and 1 tester gave a grade of B, which means they agree that combat experience is enjoyable on their input device. This means that the design of combat loop is succeed. For Sci-Fi Visual, all testers gave a grade of A, which means they agree that Sci-Fi visual meets their expectation. Table 2 shows that the implementation of sci-fi visual is succeeded. For Realistic Visual, 2 testers gave a grade of A, 2 testers gave a grade of B, and one for C, which means they neutral about if the realistic visual meets their expectation. This means the implementation of realistic visual needs some improvement in the future.

For AI Responsiveness, 4 testers gave a grade of A, which means they agree that AI in the game that is responsive to player's behavior. This means the implementation of AI is succeeded. For Immersive Level, 3 testers gave a grade of A, and 2 testers gave a grade of B, which means 3 out of 5 of testers agree that the gameplay is immersive. This shows that the gameplay design is immersive to some and needs some improvement in the future.

5. Discussion

There are some threats of validity in this paper. The evaluation phase is based on a survey result from a few people near developer's location, therefore, there might be cognitive bias, the result cannot show the objective opinion across global players, the objectivity is limited to a culture area in China. However, the work still managed to deliver a gamepad-oriented FPS system. But with the limitation to the Unity UI system, there still need some modification to it to allow full mouse free operation in main menu. A lot of free tools are used such as ProBuilder and MakeHuman. During the development process, using ProBuilder and MakeHuman boosted up the speed of modeling as well as remains high quality.

All art assets made for this game are shared under the CC BY-SA 4.0 license which ensures everyone is able to benefit from the work. Last but not least, this game is highly modular, means the codebase of this game can actually be a framework for future Unity3D-based FPS games. Some students have already worked out 2 games with this framework.

6. Conclusion

To implement a realistic Sci-Fi single-player First-Person Shooter game, a game named What Happened to Site-13? was developed in this paper. The game successfully implemented the sci-fi style visual, and partially succeed on implementing the realistic art style. Testers agree that the visuals meet the expectation of Sci-Fi art style but neutral about the visual is realistic. The game also successfully implemented the gamepad-oriented control scheme in combat and corresponding combat loop. In the meanwhile, the immersive experience is also successfully implemented in this game. Also, it can serve as a base framework for other FPS games as some students used the framework to complete games as their schoolwork. The work is successful in many aspects, but some features are on horizon.

In the future, the game aims on implement a UI system that support mouse-free operations. Also, the customizable configuration of control scheme needs to be implemented. In the same time, a save system also need to be implemented for most of single-player games have this functionality. Hopefully, the project can be extended into an all-in-one FPS framework which contains both single-player and multiplayer at the same time.

References

[1] U. Tec, Unity Real-Time Development Platform |3D, 2D VR & AR Engine., 2022 U. Tec., 29 8.

[2] Li N Z J. A Novel FPSM Controller for DC-DC Switching Converters 2007, J. Elec. Tec., 5(2):136-140.

[3] Isla D. Handling Complexity in the Halo 2 AI. Pro. Gdc, 2005.

[4] Weber B G, Mateas M, Jhala A. Applying Goal-Driven Autonomy to StarCraft. 2010 AAAI Conf. Art. Intel. Int. Dig. Ent. 31.

[5] Ahin S, Kse B, Aran O T, et al. The Effects of Virtual Reality on Motor Functions and Daily Life Activities in Unilateral Spastic Cerebral Palsy: A Single-Blind Randomized Controlled Trial. 2019. G. Heal. J, 9(1).

[6] Wu J. Strategic Correlativity and Network Games. 2012, Eco. Mod., 102.

[7] Zimmermann M. Unbounded Lookahead in WMSO+U Games. 2015, Compute. Sci. 9139:412-425.

[8] Sosa G W. The Impact of a Video Game Intervention on the Cognitive Functioning, Self-Efficacy, Self-Esteem, and Video Game Attitudes of Older Adults. 2011, Dis. Th. Grad. 29.

[9] D. Isla, Building a Better Battle: HALO 3 AI Objectives, 2008, G. Dev. Conf. 324.

[10] Lee S, Willis B, Jr J, et al. Entertainment history museums in virtual worlds: video game and music preservation in second life, 2010, J. Conf. Dig. Lib., 631.

[11] Rigby C S, Przybylski A K. Virtual world and the learner hero: How today's video games can inform tomorrow's digital learning environments. 2009 T. Res. Edu., 7(2):214-223.

[12] Berns A, Gonzalez-Pardo A, Camacho D. Game-like language learning in 3-D virtual environments. 2013 Com. Edu., 60(1).

[13] Callaghan, Michael, J, et al. Using Game-Based Learning in Virtual Worlds to Teach Electronic and Electrical Engineering. 2013, Ind. Inf. 353.

[14] F. a. Ha. Fad. Nusrat, How Developers Optimize Virtual Reality Applications: A Study of Optimization Commits in Open-Source Unity Projects, 2021 Int. Conf. Soft. Eng., pp. 473-485.