Open World Survival Game

CSCI 5619 Final Project

Jiseok Moon
Computer Science
University of Minnesota
Minneapolis, Minnesota, USA
moon0222@umn.edu

ABSTRACT

In this project, the user will be able to experience an open world survival game with a high degree of freedom. Recently, many attempts are being made to implement popular games in Virtual Reality. The author also got his motivation from those attempts and created an open world in VR. The game will be in first-person view, which enables more immersion to the user. Just like in the real world, there will be days and nights when user stays and plays in the world. The user will feel hunger as time goes on, so the user will have to go on a hunt for food. There will be various animals such as pigs, chickens, and cows in the world. For hunting, the user will need a tool. Variety of tools can be manufactured with ingredient materials of nature. Woods can be earned after chopping down a tree, stones and be earned after mining a rock. By gathering them, the user can create a tool for more diversified materials. The user will be able to build their own house, make a gun, and dominate the whole world. The progress will be able to be paused and saved, so that the user can come back whenever they want.

CCS CONCEPTS

• Computing methodologies → Computer graphics → Graphics systems and interfaces → Virtual reality

KEYWORDS

Virtual reality, Open world, Survival game, Immersion

1 Introduction

It is well known fact that Virtual Reality is one of the technologies that take center stage now with its unpredictable potential. The advent of computers as a collection of technologies that enabled the study of virtual reality is not even a century old yet. However, the hypothesis about virtual reality emerged only a few decades after its advent, meaning that people had always pictured imaginary scenarios that might sound chimerical and dreamt to have a chance to experience them. As men's ability to think distinguishes them from animals, everyone has his or her own likes and dislikes. Therefore, we people want more entertainment that can spice up our lives, and are hungry for new "content". It can be a song that speaks to the one's heart, sports game that gives tension and excitement with various strategies, or novels, plays, and movies with a wide variety of genres.

But in all of those contents, game is considered one of the best contents in the society now. It may have some disadvantages such as an addictiveness and repeated exposure to violence, but it has a lot of advantages. Game is already widely used in stress reduction and communication. Also, a game itself has a commercial use as people are willing to pay for entertainment. Its superb accessibility enables people from all over the world to join and enjoy regardless of age or sex. Furthermore, people can meet up online as it's not hard at all for people to connect to the Internet whenever they want. The game industry all over the world is getting bigger and bigger, making hundreds of billion dollars only in the U.S. The annual income from this one industry exceeded the combined annual income of Hollywood and music industry.

Recently, many attempts are being made to implement the most popular games in VR. As VR gives the user more immersive experience, the implementation of popular games' VR version cannot fail. There is a game called "Muck" developed by Dani, a Norwegian indie game developer and also a YouTube streamer. It's a multiplayer survival game in a low-poly style which is quite similar to a game called "Minecraft." However, everything in Muck was developed by a single person, with unity, in only several weeks. This fact motivated me to challenge myself to create a game similar to them, but in VR.

2 Related Works

Figure 1: A video of Minecraft live show in 2021 from official Minecraft youtube channel. The games is in first-person and

there is a crosshair in the center of a user's screen. The player in the video is digging down the hole with diamond pickaxe[1].

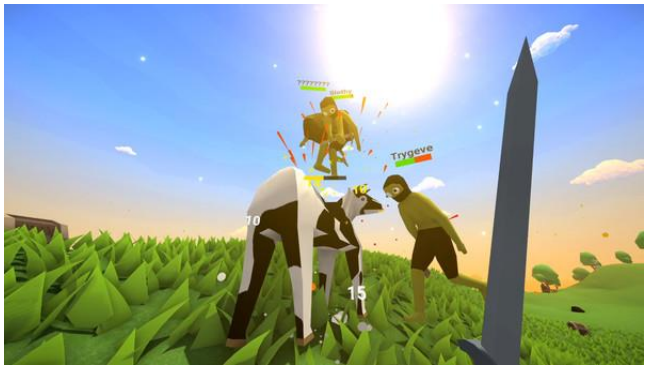


Figure 2: Figure shows the how the game "Muck" goes on. The user in the figure is holding a sword to hunt a cow with other players. Muck is also a first person perspective game[2].

As mentioned above, the author got a motivation from a game called Muck. It took the reference of Minecraft, shown in Figure 1, and made the goal for players to survive as long as they can by gathering ingredient materials and crafting various tools and weapons. There are diverse kinds of trees and minerals such as fir wood and oak wood for trees and iron, gold, and mithril for minerals. Figure 2 shows the user is hunting a wild cow with other players to get some meat. The user is holding a sword crafted by himself, and the green and red hit point bar shows the health of each object. The user's character itself is no exception as shown in Figure 3. It's also a roguelike game, which means that a user will lose all the progress when they die or reach the end of game.

Figure 3: Figure shows the actual game scene of Muck. The user can check the hp, level of hunger, and stamina at the bottom right, the holding amount of coin, time in the game, number of days passed, and company's health at center left, and quick item slots at the bottom left[2].

The source codes for Muck were decompiled by one of the users and were uploaded on GitHub, so the author could see how these were coded and take reference of it. Figure 4 shows the part of a list of decompiled codes of Muck.

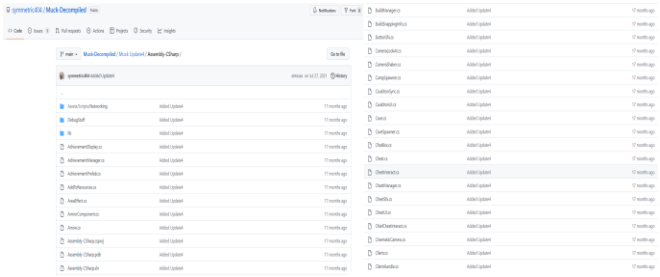


Figure 4: Figure shows the GitHub repository of one of the fans of Muck that has decompiled codes of Muck[3].

When planning the project, the author wanted to implement a gun in his project as a differentiator. There was a script for gun, which is a function that is not implemented in Muck. So, the author could get some idea of a gun in VR from the code and other examples[3]. Implementing a gun was not a hard task as expected, the author was able to have a gun that creates some effect on fire, reloads when a key entered, and deals a damage when fired on an animal.

Aside from the examples of Muck, there still were a lot of examples and tutorials of open world game in Unity.

Figure 5: Figure shows a tutorial video of implementation of character's status. The user can check his health and level of hunger and thirstiness[4].

Figure 5 shows the implementation of a character's status with hit point(HP), hunger and thirstiness, and inventory for obtained item. The author thinks status should be in a bar type as it has been a common choice for many other games. So, the author chose a bar-type status for the project.

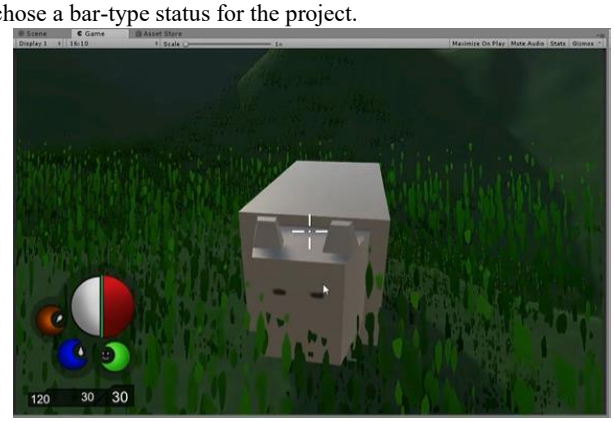


Figure 6: Figure is a screenshot from a tutorial video of implementation of status and wild animals. Bigger circle is for

health and defense. There is a thin bar between two semicircles for stamina. Smaller circles are for the character's level of hunger, thirstiness, and satisfaction[5].

As shown in Figure 2, securing supplies of sufficient food is of major importance. The author searched for implementation of wild animals and found out this low-poly pig in Figure 6, so he chose it to be a prey in his project. As well as the pig, the author used the assets provided with the tutorial. Also, there is a crosshair in the center of figure 6. It can be replaced with a raycast from the controller, but the author implemented the classic crosshair as it can present the information in a more linear way to the user.

Figure 7: Figure is a screenshot from a tutorial video of implementation of tools and their animation. The user in the video is implementing an axe so that it can get the lumber from trees[6].

Also, the author wanted to implement a few tools like axe, pickaxe as shown in Figure 7. The user will be able to swing the tool in his hand with a simple animation.

3 Accomplishments of The Project

The project was built on Unity 2021.3.4f1 version, with Windows platform. It is possible to switch the platform to Android and build on Oculus Quest 2 for VR mode, which is in progress. There is a sound manager that controls all the audio clips used in the game. All the sound effects and background music are saved in the object as a list.

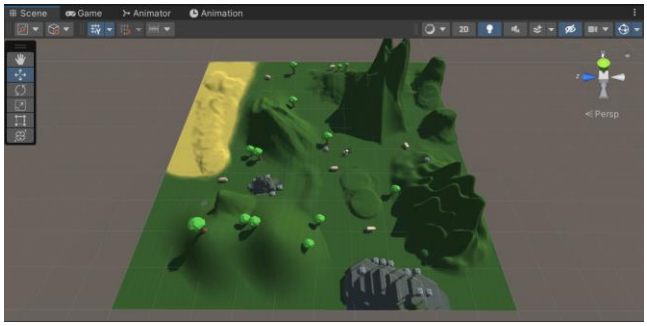


Figure 8: Figure shows the 3D terrain of the project. The terrain has rocks, trees, and pigs on a diverse geography.

With the built-in functions in Unity, a 3D terrain was successfully generated as shown in figure 8. Created mountains of different shape with terrain brushes. The part colored in yellow indicates that it is a desert area. Similarly, mountains colored in gray indicates a rockpile that will have more rocks than other area. Trees and rocks were added to the scene for the naturalness of terrain and they can be removed from the scene by a user for ingredient materials.

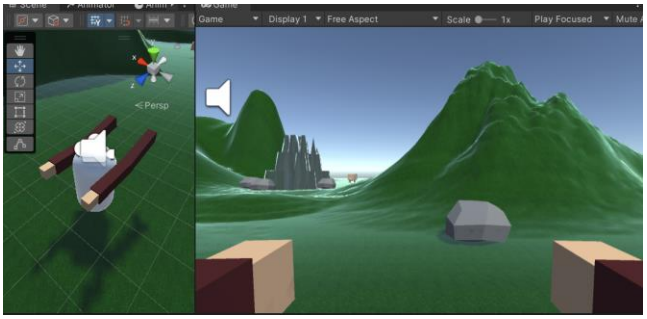
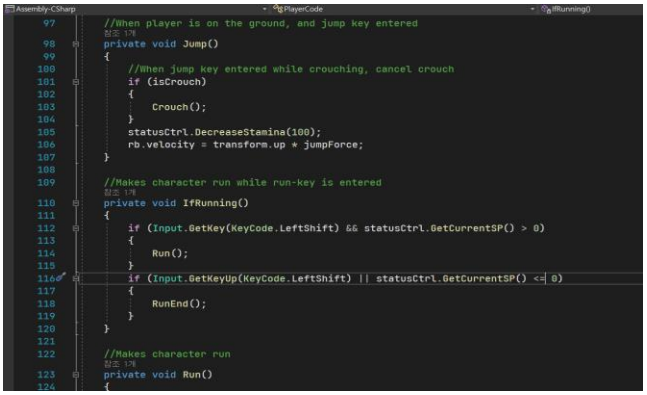


Figure 9: Figure shows the model of character in a scene on the left window, and the game view on the right.

For the user's avatar, a 3D object of capsule was chosen as shown in figure 9. The user can walk, run, crouch and jump with certain input. Figure 10 below shows the part of code for character's movement. Just like the "IfRunning" function and "Run" function, most of the functions for movement are separated into a few parts as each of them will require a specific condition to be executed. Walking speed will be the standard on deciding speeds for other actions. When the user tries to crouch, the speed will drop down significantly. For more reality, a coroutine for crouching was implemented. The coroutine will change the height of the main camera to go down a little bit when crouched. To prevent the character from jumping again before landing on the ground, the raycast function was used to check the status of player object. As the user will be able to see freely with HMD, the author created a function for camera rotation in case of running in Windows. The user will be able to see his hands as the right half of figure 9.



```
97      //When player is on the ground, and jump key entered
98      private void Jump()
99      {
100         //When jump key entered while crouching, cancel crouch
101         if (isCrouch)
102         {
103             Crouch();
104         }
105         statusCtrl.DecreaseStamina(100);
106         rb.velocity = transform.up * jumpForce;
107     }
108
109     //Makes character run while run-key is entered
110     private void IfRunning()
111     {
112         if (Input.GetKey(KeyCode.LeftShift) && statusCtrl.GetCurrentSP() > 0)
113         {
114             Run();
115         }
116         if (Input.GetKeyUp(KeyCode.LeftShift) || statusCtrl.GetCurrentSP() <= 0)
117         {
118             RunEnd();
119         }
120     }
121
122     //Makes character run
123     private void Run()
124     {
```

Figure 10: Figure shows the part of a code for the player. It deals with the movement, camera and character rotation.

Each action is divided into multiple functions as they require a specific condition to be run.

Tools. – Hand, axe, pickaxe, gun / interact possible / animation Canvas – status bar, sound manager, animals

Aside from bare hand, there are a few more tools implemented in the project. Figure 11, 12, and 13 shows an axe, a pickaxe, and a gun respectively.

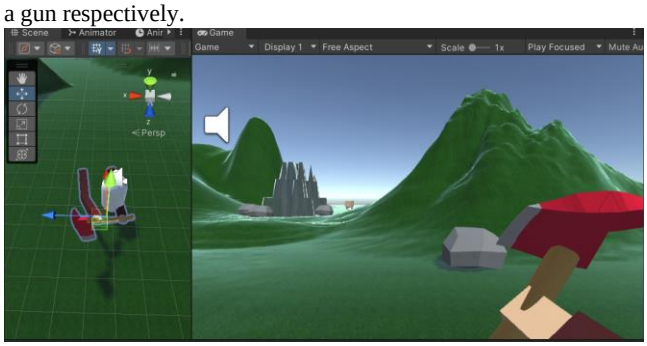


Figure 11: Figure shows an axe implemented in the game. The position of it is adjusted so that it can be seen more realistically as the right half of the figure.

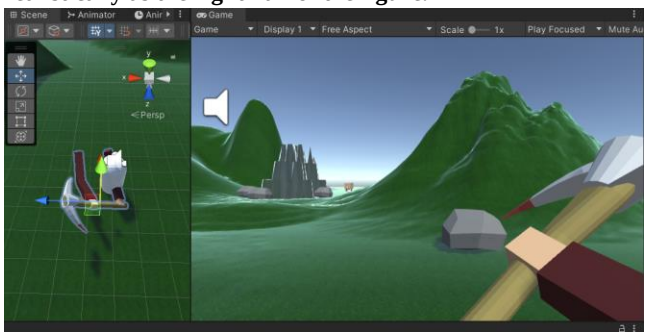


Figure 12: Figure shows a pickaxe implemented in the game.

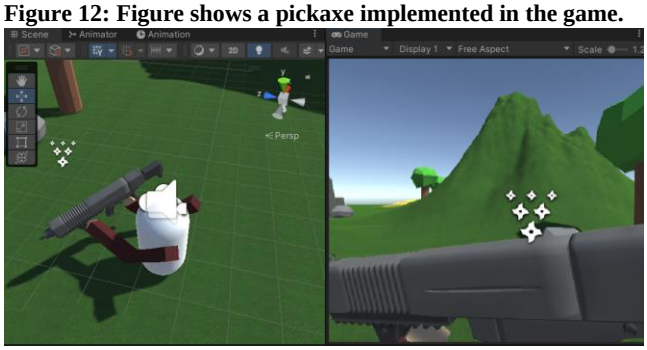


Figure 13: Figure shows a player running with a gun implemented in the game. The gun is brought to the front of the user character as it is on the animation of running while holding a gun.

The author added an animator to each of them, so they can play the corresponding one on every action. All of them in general has idle, walk, run, draw and equip, put back, and attack/swing animation. Figure 14 shows the animator of hand. Animators for other tools were generated by overriding the hand animator.

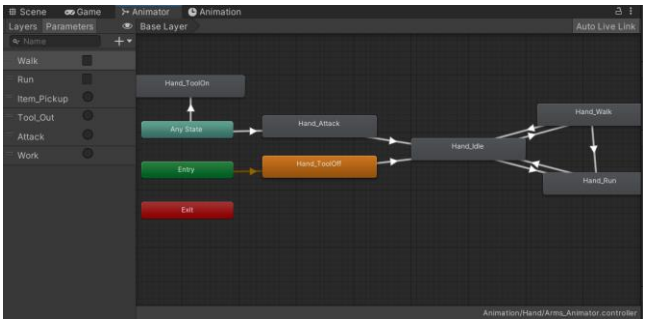


Figure 14: Figure shows the animation controller of the user's hand. ToolOn and ToolOff animations will be played when the user tries to hold a tool. Hands will stay idle with no input and start swinging up and down when the character starts to move.

When walking and running, the character will shake its arms to make it seem more realistic. For a gun, the user will not be able to shoot while running, as the animation will change to make the character hold it in front of him. Once they are generated in the game, they can be switched by pressing a button. If the game is run on Windows, number keys will help the user to change the tool: 1– hand, 2 – gun, 3 – axe, 4 – pickaxe. It was possible by creating a list and put all those tools except the gun into the list.

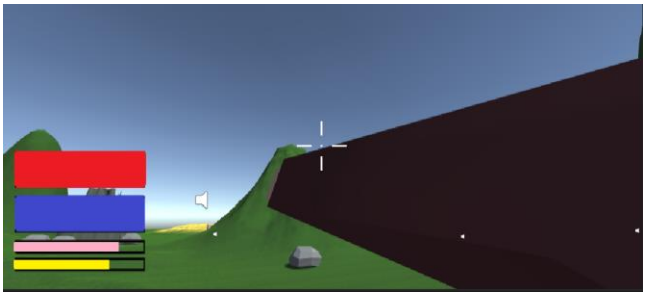


Figure 15: Figure shows the attack motion of the character. With an input, the animator in Figure 14 will start the Hand_Attack animation and this will deal a small damage to an game object that got hit.

An interaction between tools and environment materials are possible as well. Figure 15 is a screenshot of a game showing the user's hand punching in the direction that the character is facing. It deals the least damage to an animal and no damage to an environment material. Attack/Swing animation for all tools has some delay as it is impossible for a person to attack continuously in ridiculously fast speed.

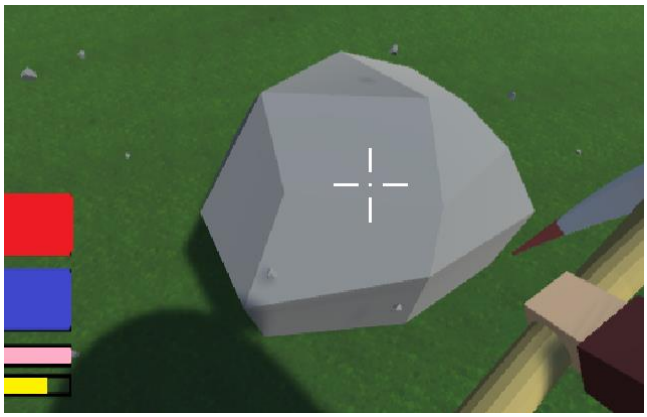


Figure 16:

Swinging a pickaxe on a rock will create some debris and play a sound effect from the sound manager on every swing as shown in figure 16. It will be scattered around the rock and then be removed after a few seconds by Destroy function in Unity. A rock's HP is set to 3 so that it requires the user to swing on it 3 times with a pickaxe in order to gather stone materials. It will create 0 to 5 stones randomly by setting the drop rate within a range: Mathf.Round(Random.Range(0, 5)).

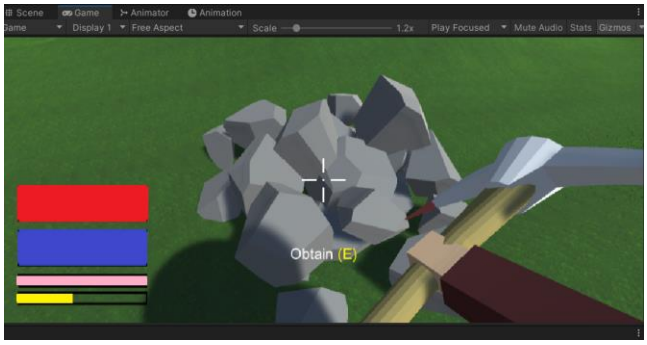


Figure 17

When the user swings a pickaxe three times, it will crash the rock into pieces and another sound effect will be played. It was successfully implemented by deactivating the rock and activating a rock with the same size and shape but already shattered into tens of pieces. By giving a box collider and rigidbody component to each piece, all the pieces will push each other. As shown in figure 17, the activated one will crash right away after the activation.

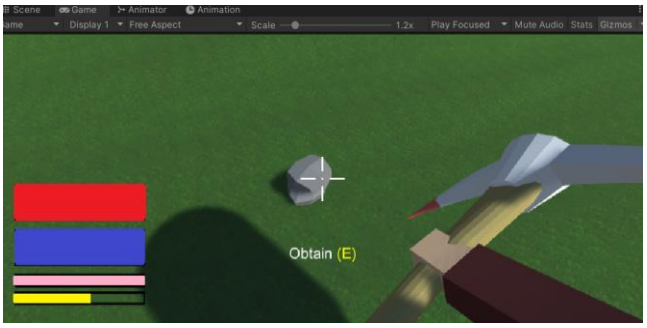


Figure 18

The debris will also be removed in a few seconds after crashing, leaving some stones on the same spot. This can be seen in figure 18. Within the range of 0 to 5, the user will get a random number of stones after mining.

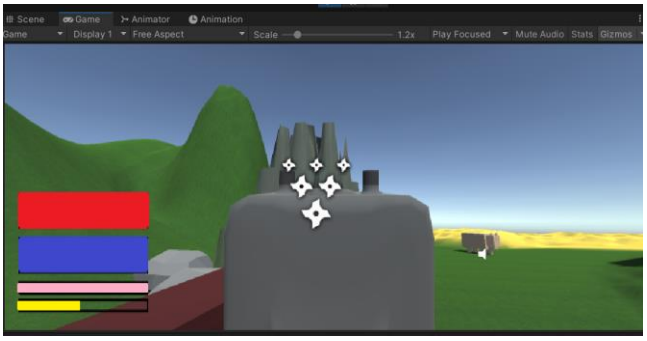


Figure 19

Among the tools, a gun has a couple more function than other tools. The reload function will start automatically after consuming all the bullets and the magazine becomes empty. It will set the trigger for reloading so that the animator will run the reloading animation. The function also calculates the number of bullets left. Also, the user can start reloading with an input whenever they want, as long as the magazine is not full and there still are some bullets left. Another one is the fine sight function. With an input, the gun comes to the front at eye level so that the user can take better aim on an object. It could be implemented naturally with the Vector3.Lerp when changing the position of a gun. The user can walk, crouch, and jump while on fine sight. However, the function prevents the user from running and fine sighting at the same time. The reload function also works but it cancels the fine sight for a moment to run the reload animation.

Figure 20

The user can shoot with a gun, but it doesn't mean that the gun is actually firing a physical bullet. Instead, the author chose the particle system in Unity's effect. The function uses a Raycast that starts from the position of camera to the front. The ray takes the gun's shooting range as length, so the ray is cast upon the game object only within in the range. Thus when fired, the function will create a hit effect on the surface of the object, letting the user know if he had hit the target or not. A couple of particle systems are generated at the muzzle of a gun for muzzle flash. A recoil was also implemented in the project. The gun recoils when fired but it will come back to the original position after it ends. The gun will also recoil when fine sighted but it will be slight.

As shown in figure 1 and 3, the game should show some information on the screen for a user. The author implemented a status bar on the bottom left of the screen. It can be seen in figure 20. The first red bar means the character's health, the hit point(HP). Blue bar in second row means the defense point(DP) that will decrease instead of health when damage received. Pink bar under the DP bar is for the character's stamina. The stamina is consumed when the user starts running or tries to jump. It will decrease like the last bar when consumed and be regenerated after about 5 seconds of break. When a user points a game object, a description for possible interaction pops up as shown in figure 18.

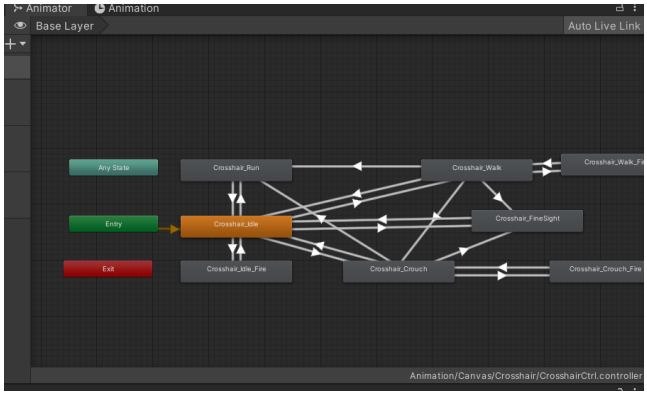


Figure 21

In figure 18, there is a crosshair at the center of a screen. It consist of a center dot and 4 lines. Figure 21 shows the animator for crosshair that changes the shape of it depending on the situation. Crosshair helps the user to get the information instinctively, so the author adjusted the accuracy depending on the player's movement. When the player moves, the crosshair will expand and the size of it becomes the impact area when the gun fired.

Figure 22

The game also has an inventory system that stores every item that the user obtained during the game, as shown in figure 22. It can be activated and deactivated whenever the user wants with a single input. The image of obtained item will show up in a circle slot with the holding amount of it in a smaller circle.

The author also implemented a wild animal system in the project. In figure 19 and 20, there is a pig in the scene. It has its own data such as hp and movement speed. In the game, it will repeat all the possible actions randomly as well as the animation of them. It will be a prey in this game that supplies the food to the user when hunted, as a common rule in an open world game shown in figure 2. As mentioned, every tool has its own damage. The pig will get damage when it gets hit by the user and die when its hp reaches 0 as depicted in figure 20.

4 Assessment of The Project

The author considers this in-progress project was successful in overall. The major expected milestones were the implementation of 3D terrain with various geography, status of a player, resource materials, tools, non-player objects and interactions between a player and them, days and nights, and a firearm. Starting with the first one, a 3D terrain with diverse geography was generated successfully and it has trees and rocks distributed randomly on it. However, the character falls through the terrain when the user tries to climb on the steep slope. The player's character can walk, run, crouch, and jump on the terrain without any errors. The user can use an axe, a pickaxe, and a gun with a smooth animation. With the implemented tools, the user can start hunting and gathering in the virtual world. While playing, the player can check the status of his character as it is reflected in real time. However, it requires a different way to show the status in VR, so the status in VR is unstable yet. There are pigs in the game with the code that makes them wander around and repeat actions randomly. With the tools created, the user can interact with them. The

implementation of all these milestones resulted in a 3D game with essential factors; tools and interactions described in the paper. Most of future works can be achieved without additional code scripts, but with the application of currently implemented works. As well as the status, however, all the functions and codes do not work in both platforms: Windows and Android, at the same time. At this writing, the game runs perfectly in Windows and it requires more modification of the codes to run on Android.

5 Limitations and Future Work

For limitations, the author wants to point out the time constraint and difficulty on developing the project in both Windows and Android platforms. The VR device used by the author is to be returned to the university as it is their equipment, so the project had to be implemented in Windows first for future work, then in Android. For future work, the implementation of day and night will be achieved to make the game more realistic. Further, the author will work on implementing more animals with multiple inheritance, moving the code for pigs to a new script, editing it, and making other animals to inherit that code. Implementing the construction with obtained ingredient materials is also a goal as the availability of crafting is one of the major factors that attracts people to an open world game. The author will add more items such as different types of minerals and woods which will increase the number of manufacturable items. Then, the author will put all these together and implement a game that can be run in both platforms.

ACKNOWLEDGMENTS

The author wants to express his deepest appreciation to Professor Victoria Interrante and Ph.D. candidate Yuxuan Huang for lectures and advices, Daniel William Sooman for motivation.

REFERENCES

[1] Patricia S. Abril and Robert Plant, 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan, 2007), 36-44. DOI: https://doi.org/10.1145/1188913.1188915.

[2] Sten Andler. 1979. Predicate path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226-236. DOI:https://doi.org/10.1145/567752.567774

[3] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[4] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY..