A REAL-TIME MULTIPLAYER FPS GAME USING 3D MODELING AND AI MACHINE LEARNING

John Zhang[1] and Yu Sun[2]

[1]Crean Lutheran High school, 12500 Sand Canyon Ave, Irvine, CA 92618
[2]California State Polytechnic University, Pomona,
CA, 91768, Irvine, CA 92620

ABSTRACT

AIs have been a key component in the gaming industry throughout its history. Developers have had multiple ways of creating new AI models that best suit their game in order to enhance the playing experience. However, with the increase in the popularity of online multiplayer games, AIs now must compete with the experience of playing with other people.To enhance AI behaviors to match that of a real player, the paper discusses the one solution for creating models that can be used for further AI research. Through utilizing some of the built-in features of Unity as well as Photon Network services, the game Maze Escape combines the multiplayer aspect of FPS games and some simple game AI models to allow them to be compared against each other in order to more easily recreate multiplayer experience using AI bots. Thus, this paper hopes to encourage developers to think about how AIs are not only used to enhance single player experiences, but it can also be used in multiplayer.

KEYWORDS

Multiplayer FPS Game, 3D Modeling, AI Machine Learning.

1. INTRODUCTION

As the U.S enters into the 21st century, the sources of entertainment ranging from music, sports, and movies have all been expanding in support of consumerism culture [4]. One of the major industries that have prospered in this time of competition are video game companies. Due to the global pandemic, more people have been spending their time indoors and subsequently video game industry giants such as Riot Games, Blizzard, and Epic games have all tried to produce new games for its audiences in order to maximize their profit [5]. Connected via gaming platforms, the video game industry is reaching its peak of prosperity generating explosive numbers of concurrent players and breaking historical records; the scariest thing being that this trend does not seem to be slowing down anytime soon.

One of the most popular categories of video games that have been played by millions of people around the world are “FPS"s, or First-Person-Shooter, which are popular among those who enjoy the combat experience of playing in the 3-dimensional world through the eyes of the character model [6][7]. While it is included in part of the sub-genre of shooter games, developers are free to shift away from the traditional path that focuses solely on combat with a set of firearms and specific objectives to achieve. They can incorporate their own ideas into the game. Throughout the development of “Maze Escape", the game explores the wide range of tools that are available

David C. Wyld et al. (Eds): SIPP, NLPCL, BIGML, SOEN, AISC, NCWMC, CCSIT - 2022
pp. 121-130, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121310

for developing AIs in video games in an attempt to select the best solutions for recreating multiplayer experiences.

Some of the existing AI techniques and systems that have been proposed include Navmesh for AI pathfinding, and reactive systems such as finite state machines and behavior trees [8]. Navmesh technique is constructing a mesh from which AI is able to determine the best path from getting from one point to another as well as providing a way to "see" the environment around the AI. Reactive systems allow the AI to interact with the player's decisions and take different actions to create different experiences for the player. Finite state machines (FSMs) achieve this by having different states the AI can be in to match different existing scenarios in the game while Behavior tree archives this by having a set of rules and conditions to guide the decision making process [10]. These systems have allowed developers to make more complex AI for their games and focus on crafting experiences that are more interactive. Developers have the ability to create AI experiences equivalent to that of other players. However, a bad version of AI can often lead to AI taking away the overall enjoyment of the game rather than adding to it. [Example of a game that has weak AI compared to their multiplayer system].Thus, developers need good tools to allow them to make good AI.

For instance, Unity provides a built-in nav-mesh system for pathfinding [9]. Pathfinding is the idea in which the player is able to determine the ideal traveling route to take during the navigational processes that takes the least amount of time between two points. The Unity Asset stores would be another case in point. It contains many reactive system tools like FSMs and Behavior Trees. For example, Candice AI has components for handling player detection and combat systems for the game AI. Another example would be Breadcrumbs AI . There are also non-character based AI like Procedural Level Generator, which creates the level around the player themselves.

Our goal is to create a game in which AI can match up to the experience of multiplayer. The current methods that are currently available in the game include Navmesh systems and way point systems. First, there are some good features of the Navmesh system which allows for pathfinding inside the level generator which prevents the player from walking straight through walls and objects in the environment. Second, some good features of the way point system is that it allows for the escaper bot to have an unpredictable route for reaching the end goal which would imitate that of a real player style of playing the game; filled with randomness and does not conform to a fixed way of playing the game. The way point systems generate different routes for the escaper AI bot to take to reach the end goal. When the game starts the way point system will randomly choose a path for the AI to take out of the many the developers had encoded into the system. Thus, we believe that using these simple tools we can at least get close to emulate multiplayer experience in a single player environment. Our system is a synthesis of the Navmesh system and the way point system allowing them to work together for pathfinding while other tools keep them separated.

In two application scenarios, we demonstrate how the above combination of techniques increases the accuracy or the imitation for the ability of the AI to mimic the multiplayer experience.

First, we demonstrate the usefulness of our approach by a comprehensive case study on the competitive AI compared to the competitive player in which the player has to play against one other opponent either AI or another player in a 1v1 game mode of Maze Escape in which one act as the "Defender" and the other as the "Escaper". Second, we try to analyze the result of our 1v1 competitive AI to a competitive player with a cooperative AI compared to a cooperative player. In this case, not only are we testing the player's experience playing against an opponent AI and seeing the differences between an actual person and the AI, we are also implementing a way for

the player to play with an ally in the game either AI or another player against two opponents in a 2v2 format. In scenarios, we can test our results through a questionnaire by asking the players' experiences playing the game with the AI versus that of an actual person. Therefore, by measuring the level of artificial intelligence imitation that has been achieved, we are able to verify whether or not the performance of each of the AIs, both the competitive version and cooperative, have been enhanced in order to emulate the real online multiplayer experience. Through the development of these complex, sophisticated and futuristic AIs, certainly, these goals are only one step further and will be explored later in the paper.

The rest of the structure of the paper is organized as follows: Section 2 gives the details on the challenges that we met during the experiment and designing the sample while Section 3 goes into the details of our solutions corresponding to the challenges that we mentioned in Section 2. Next, Section 4 presents the relevant details about the experiment we did which includes other similar experiments done towards this same topic, followed by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks as well as pointing out the future work of this project.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Deal with networking

To start off, the first challenge that we encountered has to deal with networking. For instance, in order to create the game, we need to first figure out which Unity Networking solution to use, which is difficult as creating custom networking scripts is much more difficult and challenging compared to using a pre-built solution that requires no coding and programming. Next was setting up the player. We had trouble determining the location of the spawn points and had issues with the player not spawning in the right locations. Also there were errors in the camera fov in which the character model's eyes will see the interior of the gun that blocks off its vision and also the controls as well when the players spawn and not be controlling its own character. There were also errors in the movement and shooting as well. Lastly, setting up the teams required the creation of the teams and in order to keep each side balanced and assign bots accordingly or the game would not start.

2.2. Deal with the navmesh system

Second challenge had to deal with the entirety of the navmesh system. First, we are required to find how to create the navmesh on top of a level. It was difficult to dynamically generate the navmesh during the game as when generating a new level, the nav mesh would not change and the problem with that is the inability to produce an accurate path that matches the level produced by the procedural maze generator that we implemented at the beginning of the game. We also looked for the Nav Mesh component package to make generating nav mesh easier as we kept the nav mesh as a single mesh to allow easy movement from one part of the level to another. We had trouble getting the bots to use the newly produced nav mesh as we couldn't figure out how to get the AI to "walk" with the player model and using NavMesh agent for walking as it required a separate animator component for AI movement to allow the bot to play the walking animation. Lastly, we had to program the code for the escaper bot's waypoint list and randomly choose one point to go to before heading to the end goal in which we designed to allow randomness for the path chosen just like that of a real player.

2.3. Figure the AI's behaviors

Lastly, figuring the AI's behaviors and goals to get it to emulate a real player was the most difficult among the rest. There was an issue with the escaper bots not moving to the end goal and also choosing the same path despite including the random way points that were designed to prevent fixed AI behavior from occurring. Next, there was also a movement issue with the defender bots as they were unresponsive and remained standing throughout the duration of the game and we had difficulty adding the "wandering" system to have them move around the level. The last two issues were for the escapers not trying to run away from the defender when it opened to its field of view and it decided to just run past it and lastly was with the bots not shooting when the opponent player was facing them as it entered its shooting radius.

3. SOLUTION

To begin, the overall system starts with the menu section in which includes the ability for the user to login with a name, then join a game room or create their own, choose their own team. After the initial process, the host of the game room will start the game once every user is ready. The host will also check if any kinds of bots in the game are added. Next, Maze Escape is a strategic multiplayer-FPS game in which players are deployed into a virtual 3D world and are divided into two teams, the Defenders and the Escapers. Because of its competitive nature, in order for both to achieve victory, they will need to satisfy their winning conditions. For instance, the Defender needs to either eliminate all Escapers or keep them occupied until the timer hits zero in order to win. On the other hand, the Escapers are able to either kill all Defenders or try to avoid them and escape the Maze to achieve victory.

We used Unity as the main Game Engine to develop this game and because the is a multiplayer FPS, we also used Photon to manage the multiplayer functionality specifically managing the network connections between the user's computer while synching all the changes such as player movement, health bar, damage output/input from one client's device to every users' screen thus enabling the multiplayer experience. After this, we are able to implement AI Navigation mesh and way point system for pathfinding and later adding shooting mechanics, animation for the AIs, and AI healthbar in order to recreate the experience of that of a real person.

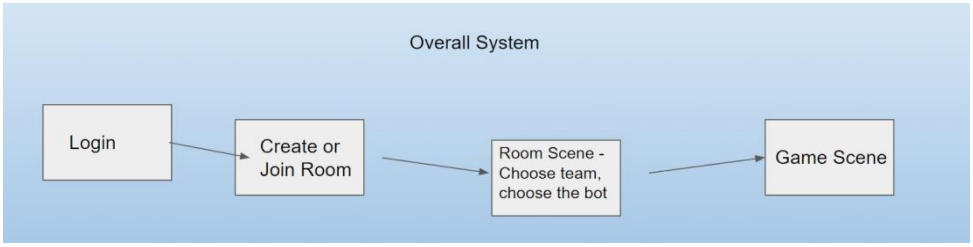

Figure 1. Overview of the system

To start off with, the first component that we implemented was Networking using the photon networking unity version 2.0 package. We built the subcomponents for the menu and the main level for the game. In the menu system, we implemented a sign in system that allows the user to join the game with a gamer tag, a room selection screen that allows the user to decide which active game to join or to create one themselves, and a game room scene in which displays the number of players, robots and their chosen roles. Furthermore, in the game level, we implemented networking by adding specific components to synchronize player position, health, and any animations that are playing. The game level has a component for handling the player and

bot spawning which has a list of points where each team can spawn in as well as a randomizer function to allow positions to be randomized. We also added a game manager system for synchronizing the time left for the round and the amount of players remaining. Based on the win/lose condition the game manager would track which team has won and which team has lost which is shown on the player's screen accordingly. In addition, we have added a script for handling the end goal for the escapers which is a simple trigger which will determine whether the escaper has entered the escaping zone and if true, it will display on the screen for all the players that the escapers have won.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;

public class NetworkedShooting : MonoBehaviourPun
{
    [SerializeField] Camera playerCamera;
    public float damage = 10;
    public KeyCode shootKey = KeyCode.Mouse0;
    public bool canFire = false;
    private void Start() {
        if(playerCamera == null)
        {
            playerCamera = Camera.main;
        }
    }
    public void Shoot(RaycastHit RCHit, int dmg=0)
    {
        if(!photonView.IsMine || playerCamera == null) return;
            RCHit.collider.GetComponent<NetworkedHealth>()?.TakeDamage(dmg==0?damage:dmg);
    }
}
```

Figure 2. Screenshot of code

Now moving on to the AI behaviors section, we started off with using Unity's built in navmesh system for pathfinding. There are two sides to the navmesh systems. First, the navmesh level component will generate all the pathways from one point in the level to another. Second, the navmesh agent, the AI pathfinder, which uses the generated mesh to determine the optimal path from one point in the level to another. With these two components, the escaper bots are able to find a path to reach the end goal because the navmesh agent will choose the optimal path which is usually the shortest path, the escaper bots tend to always choose the same path resulting in predictable behaviors. To remedy this issue, the waypoint system is administered to each escaper bot with a random waypoint that the AI has to go to before reaching the end goal. This also allowed the defender bots to randomly choose a waypoint to patrol that area. With the pathfinding system mostly updated, to make the AIs more like a player, we added the animation that is similar to the players animations as well a field of view system to allow the bots to react to other players within the level.

4. EXPERIMENT

4.1. Experiment 1

In order to verify that our solution can effectively solve problems at different levels and have good user feedback, we decided to select multiple experimental groups and comparison groups for several experiments. For the first experiment, we want to prove that our solution works stable and continuously, so we choose a group size of 100 different trials in 5 different kinds of target. The 5 different types of skin are cars, stationary enemy, moving enemy, animals, buildings. The goal of the first experiment is to verify if the AI auto targeting algorithm works good for different types of targets. Through sampling 5 groups of different targets. Result is collected by statistics if the AI could find the target correctly. Experiments have shown that almost all targets in different

types tested the right result. Moving enemy has the most correct rates, which means our user are works more better in aiming the moving enemy in the game. This experiment could explain that the data sets do have a obvious impact on the finding the targets and aim it, because the data we are using have a high rates of the moving enemy. The average correct rate of 5 different types of the aims shows below:

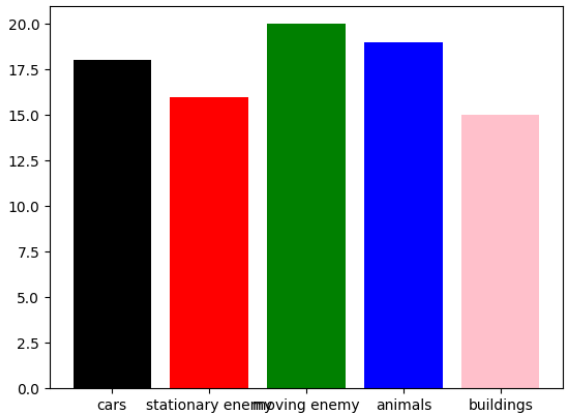

Figure 3. Result of experiment 1

A good user experience is as important as a good product. So a perfect solution should have excellent user experience feedback. In order to prove that our solution has the best user feedback, we specially designed a user experience questionnaire base on the US system usability questionnaire rules. We statistics the feedback result from 100 users, Track the user's data for 10 days play, let them explore freely on the functionality of the game. We divide those users into Five different groups. The first group of users ages from 10 - 20, the second group of users ages from 20 - 30, the third group of users ages from 30 - 40, the fourth group of users ages from 40 - 50, the fifth group of users ages from 50 - 60. The goal of the first experiment is to verify high feedback scores show high performance. We collect the feedback scores form these 5 different groups of users and analyze it. Experiments have shown that users who ages from 20 - 30 give the highest result feedback to our game. Which may because of the age between those range are more likely to play a shooting game and using the auto targeting scheme The experiment graph shows below:

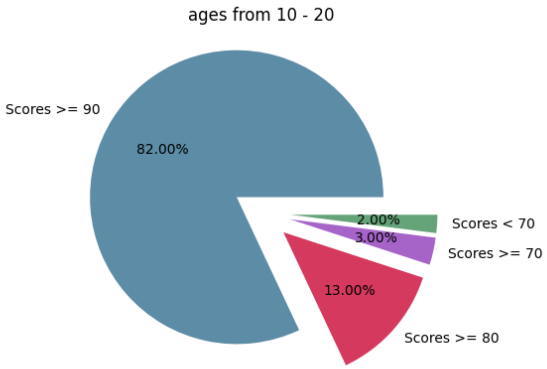

Figure 4. Result of age 10-20

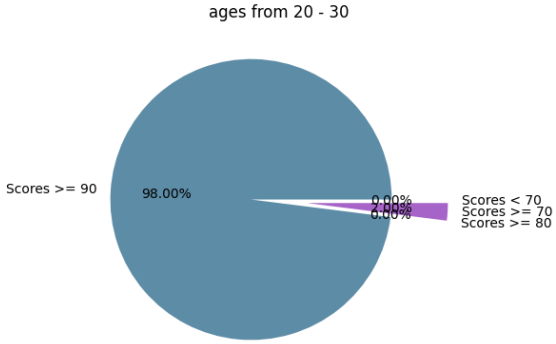

Figure 5. Result of age 20-30

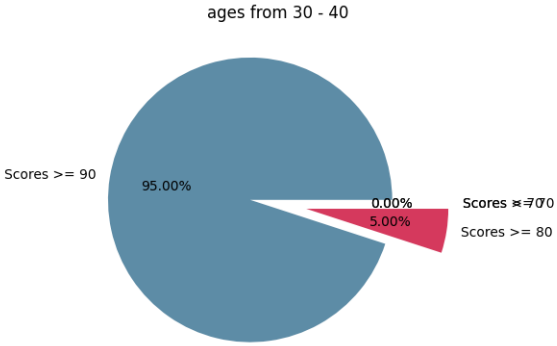

Figure 6. Result of age 30-40

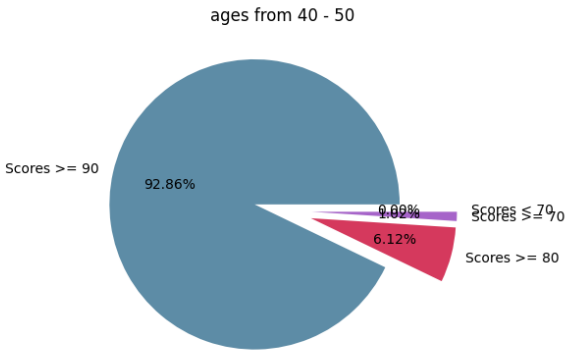

Figure 7. Result of age 40-50

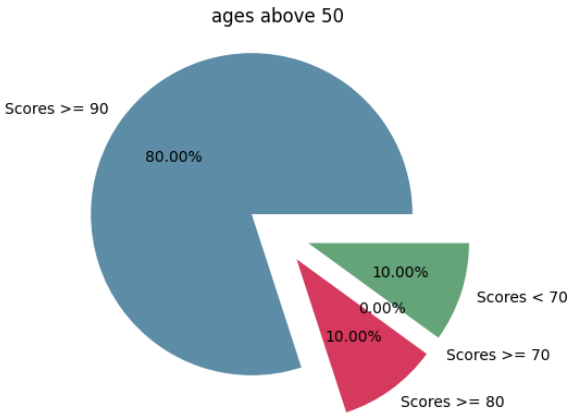

Figure 8. Result of age above 50

5. RELATED WORK

Omid Shafieid et al describes how games have contributed to the research of artificial intelligence [11]. When decomposing the technical nature of games, AIs can absorb the information in order to learn other complex tasks which is similar to our goal as we continue to pursue the enhancement of AI and super-human performance level to imitate that of a real player. For instance, this work uses graphical analysis to identify how each scenario is solved through a list of fixed sets of policies which are designed to encapsulate diversified interactions within the game which is extremely helpful as it helps generate creative game structures that can be used to train AIs.

Livingstone describes how the Turing's test can define the association of human behavior with ai through a series of interrogation questions and how successfully an AI is able to mimic human responses under certain circumstances [12]. This is similar to our works as we both try to explore the level in which the imitation of human-like responses can be generated by a machine. Some differences however is that the Turing test requires a human operator that engages in a series of questioning while our game's AI specifically focuses on the imitation of human-like behaviors.

Barata at el describes how it was difficult for a massive multiplayer strategy game that has a rapidly changing environment which requires the players to be active for a long time in order to encourage realism within the game [13]. They implemented a MMOTBSG as a replacement for human players right when they become inactive is a strength of their research as the AI plays the game for players with similar level of performance which is quite impressive. This also parallels our research as we seek to increase AI behavior to match that of a real player in order to emulate the multiplayer fps experience.

6. CONCLUSIONS

Maze Escape is a game that combines multiplayer with the AI experience. In order to enhance the multiplayer experience, we can replicate the behaviors from that of a real player onto the AI through using a list of methodologies which enables the AI to have human functions that is distinct from that of the AI experience. Maze Escape is excellent in determining the optimal solution for combining the multiplayer experience with the AI as it provides the option for the user to choose the AI to either be the opponent who enables the competitive aspect of the game or

choose the AI as a cooperative partner which enables the cooperative aspect of the game. All of which is essential for recreating the multiplayer experience as it fulfills its goal of having the AI imitate the human player. Maze Escape is unique as the escaper and defender AIs all have its own goal of reaching the winning condition and in order to achieve that, they will be able to mimic the responsibilities from that of a real player in order to satisfy the requirements for the game to end and ultimately for one side to achieve the ultimate victory. When developing AIs, others should consider the level in which the bots are able to provide an experience which will match that of a real player just as if the bots were human players not only for single player games but as well as multiplayer games.

Since the game is still in the developing phase, the AI behavior is limited to two main states as in the current AI it is only able to go to someplace and react if it sees an opponent which they will be programmed to either shoot or run away. The game itself is very simple because it only has one role, and one map level so it is not able to sustain the interplay of an environment that contains multiple variables/interactions between the game and the player.

In the future, the game will be adding an additional role system for players to choose their unique class division which contains special weapons and abilities that are designed just for that class. There will also be updates to maps, and animations specifically for different abilities based on player roles. With respect to the AI updates, there will be addition to AI states to allow more actions for the AI and different model types such as decision trees, behavior trees, and planner systems.

REFERENCES

[1] Cillessen, Antonius HN, and Peter EL Marks. "Conceptualizing and measuring popularity." Popularity in the peer system (2011): 25-56.

[2] Regensburger, Alois, et al. "Photon propagation in a discrete fiber network: An interplay of coherence and losses." Physical review letters 107.23 (2011): 233902.

[3] Corchado, J. M., and B. Lees. "Integration ai models." Workshop On Knowledge Discovery And Data Mining. Pml-Nerc, Plymouthlondon, UK. 1998.

[4] Hanif, Hanif, and Is Susanto. "Consumerism Culture Of Urban Communities Based On Islamic Economic Perspective." AGREGAT: Jurnal Ekonomi Dan Bisnis 4.1 (2020): 83-99.

[5] Mofijur, Md, et al. "Impact of COVID-19 on the social, economic, environmental and energy domains: Lessons learnt from a global pandemic." Sustainable production and consumption 26 (2021): 343-359.

[6] Jansz, Jeroen, and Martin Tanis. "Appeal of playing online first person shooter games." Cyberpsychology & behavior 10.1 (2007): 133-136.

[7] Cardamone, Luigi, et al. "Evolving interesting maps for a first person shooter." European Conference on the Applications of Evolutionary Computation. Springer, Berlin, Heidelberg, 2011.

[8] Brewer, Daniel. "Tactical pathfinding on a navmesh." Game AI Pro 360: Guide to Tactics and Strategy (2019): 25-32.

[9] Lester, Patrick. "A* pathfinding for beginners." online]. GameDev WebSite. http://www. gamedev. net/reference/articles/article2003. asp (Acesso em 08/02/2009) (2005).

[10] Brand, Daniel, and Pitro Zafiropulo. "On communicating finite-state machines." Journal of the ACM (JACM) 30.2 (1983): 323-342.

[11] Omidshafiei, Shayegan, et al. "Navigating the landscape of multiplayer games." Nature communications 11.1 (2020): 1-17.

[12] Livingstone, Daniel. "Turing's test and believable AI in games." Computers in Entertainment (CIE) 4.1 (2006): 6-es.

[13] Barata, Alexandre Miguel, Pedro Alexandre Santos, and Rui Prada. "AI for massive multiplayer online strategy games." Seventh Artificial Intelligence and Interactive Digital Entertainment Conference. 2011.

[14] Harrison, Fiona E., et al. "Spatial and nonspatial escape strategies in the Barnes maze." Learning & memory 13.6 (2006): 809-819.
[15] Cutumisu, Maria, and Duane Szafron. "An architecture for game behavior ai: Behavior multi-queues." Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Vol. 5. No. 1. 2009.