A thesis submitted in partial satisfaction of the requirements for the degree of Master of Computer Science and Engineering in the Graduate School of the University of Aizu

Using Reinforcement Learning to Create Human-Like Movement Patterns in Fighting Games

by

Ippo WAKANA

February 2024

The thesis titled

Using Reinforcement Learning to Create Human-Like Movement Patterns in Fighting Games

by

Ippo WAKANA

is reviewed and approved by:

February 2024

Chief referee

Senior Associate Professor

Maxim Mozgovoy Date:

Senior Associate Professor

John Blake Date:

Senior Associate Professor

Evgeny Pyshkin Date:

THE UNIVERSITY OF AIZU

Contents

Chapter 1 Introduction **1**

Chapter 2 Game: Universal Fighting Engine **2**
2.1 Game Rule 2
2.2 Move Type 3
2.3 Status 4
2.4 RandomAI 5

Chapter 3 Unity ML-Agents **6**
3.1 Structure of ML-Agents 6
3.2 ML-Agents Function 7

Chapter 4 Humal-Like Behavior **8**
4.1 Evaluation Method 8
4.2 Human-Like Behavior in fighting games 9

Chapter 5 Building the AI **10**
5.1 Difference between DQN and PPO 10
5.1.1 DQN 10
5.1.2 DQN 10
5.2 Learning Interval 10
5.3 Setting Reward 11

Chapter 6 Experiment **13**
6.1 Play-Video 13
6.2 Test Questions 14

Chapter 7 Result **15**

Chapter 8 Conclusion **17**

List of Figures

Figure 2.1: Ufe's battle screen "Character: Robot Kyle". 3
Figure 3.1: Structure of ML-Agents. 6
Figure6.1: Screenshot of the play video. 13

List of Tables

Table 2.1 Robot Kyle action 4
Table 2.2 Information obtained in debug mode. 4
Table 2.3 Random AI setting details 5
Table 3.1: Learning function. 7
Table 5.1: Value used for rewards. 11
Table 7.1: Answer to question 2 in video 1 15
Table 7.2: Answer to question 2 in video 2 15
Table 7.3: Answer to question 2 in video 3 16

Acknowledgement

I would like to express my gratitude to my supervisor, Professor Maxim Mozgovoy, for providing me with a great deal of advice, from deciding the theme to how to proceed with the research.

And we are very grateful to those who participated in the experiments of this research.

Abstract

Our research focused on creating an AI with "human-like" movements using a Unity-based fighting game called the Universal Fighting Engine. Fighting games allow the use of NPCs (non-player characters) for practice. However, NPC movements are mechanical and therefore boring to those who understand the game. Therefore, we thought that creating an AI that moves like a human would allow for more tense matches. The goal of this research is to create a human-like AI using reinforcement learning. To create the AI, we employed ML-Agents, an open source plugin, into our reinforcement learning environment.ML-Agents is a tool that makes reinforcement learning easy in Unity. The AI to be created take optimal actions based on rewards; the Turing test was used as a method to evaluate the AI. The Turing test is one in which a human observes an AI or human behavior and determines whether the behavior is AI or human. The results of the test showed that 47.7% of the judges judged the AI's play to be human. Therefore, it was considered that the AI was able to imitate human-like movements. This result suggests that the created AI can be practiced as a superior opponent to its NPC counterpart..

Related work

The key to this research is that it is “human-like”. Swiechowski listed three benefits of human-like AI[1] These are “getting an immersive feeling”, “being able to act as a substitute”, and “becoming a tester.” ”Immersive" refers to the fact that NPCs move like humans instead of mechanically, increasing the sense of realism and allowing the player to focus on the game. “Substitute” allows one person to participate in a game played by multiple people. “Testers” can be used as virtual testers who specialize in predicting interactions that real people would make.

When developing methods and technologies that accurately capture human play styles, “human-like” is an essential element..

Zhao and his team also said that dialogue is needed to increase immersion[2] They then used reinforcement learning techniques such as DQN to create AI. The game used is a “real-time multiplayer mobile game with a stochastic environment and continuous action" that does not have a titled title. This game is a team game, so players need to communicate with their allies. Therefore, it also plays the role of “Substitute” mentioned above. He also states that because he values ``immersion," the strength of the AI needs to be adjusted based on technical ability and play style.

Humanity is a vague concept, which makes it difficult to evaluate. A test called the BotPrize Challenge[3] was held from 2008 to 2014. In this challenge, you will perform a Turing test in a first-person shooter (FPS) environment. Actual human players were often judged below 50% as a result of this challenge. This shows the difficulty of the Turing test[4]. among them, Livingstone[5] describes the possibility of applying a variation of the Turing test to assess the human-likeness of computer-controlled characters. He talked about the reliability of the Turing test. When evaluating the behavior of game AI, the Turing test is a binary test, so it cannot be said to be very reliable. However, rather than just making a selection, he said that asking respondents for their reasons for their choice and their opinions on the AI's behavior will be important for future AI improvements.

Chapter 1

Introduction

AI has grown significantly in recent years and is permeating games and daily life. In terms of games, It is active in a wide range of fields, from board games such as Go and chess to fighting games, which are e-sports. These games are highly competitive and have professional players. However, many people who play are not professionals and are just playing for fun. I think there are three main ways to play these games.

1. Play a match with a friend.

2. Play a match online against someone you don't know.

3. Play a match with the NPCs implemented in the game.

The first is "playing against friends". This allows you to play while communicating with your friends. Therefore, if there is a difference in skill between you and your friend, you can reduce the gap by teaching your friend the skill or by taking it easy on him or her. In this way, the game can be played in a fun way. The second option is to "play a match online against someone you don't know." If your opponent has a large difference in ability, it will not be a match at all, and you will not be able to enjoy it. This will lead to it never being played again. However, if the opponent is similar in ability, it will often be a close battle, and you can enjoy this tense situation. Therefore, it is recommended that people with some level of skill play the game. The third option is "playing matches with NPCs implemented in the game." This leads to knowing the game. The purpose is to learn how to deal with the attacks of your opponent in the game, improve your skill level, etc., and you can enjoy it by feeling the improvement in your own level and gaining a sense of accomplishment. However, NPCs have many mechanical movements and are unnatural opponents. This makes it boring for players who understand the game.

AI has been integrated into games for some time. In the board game field, chess was surpassed at the professional level by IBM's Deep Blue in 1997[6] In Go, AlphaGo beat the professionals in 2016[7] These results have affected many different fields. Fighting games, like Go or chess, require only two participants and no teamwork, and the rules are very simple: you win if you can reduce your opponent's health to zero before your own health to zero. Although the rules of fighting games are simple, they are not turn-based like board games, and the opponent moves in real time. The player must respond to this by moving the character and performing a variety of attack actions. This requires proper reaction and anticipation. For this reason, fighting games have already created AI that surpasses that of professionals. However, they are often no match for humans because they determine their next actions from human input or respond at speeds that exceed human reflexes. Therefore, AI actions are clearly not human movements.

The goal of this research is not to create a strong AI. It is to create an AI that players can "enjoy." It has been proven that players enjoy playing against an AI that behaves like a human [8] Therefore, the AI must be at a level where humans can enjoy the game without unnatural movements.

In this research is to build an AI by determining rewards with DQN (Deep Q-Network) DQN is a type of reinforcement learning, in which rewards are given to AI actions and the AI learns which actions are optimal. Therefore, the AI seeks the behavior that yields the maximum reward. The result is an AI that performs the optimal behavior. The goal of this research is to make it "human-like" with DQN, which determines its behavior by reward without creating a teacher, rather than with a Markov decision process, etc. DQN will act to win to take the optimal action. Therefore, in this research is to create a "human-like" AI by adjusting the amount of learning and the rewards obtained during the game. The Turing test [6] was then used for testing. Also, the reason for using DQN is that we wanted to investigate whether it is possible to make human-like movements without learning from human movements.

Chapter 2

Game: Universal Fighting Engine

This research used the Universal Fighting Engine [7] This is a development tool for fighting games that can be used in Unity; Ufe is like fighting games that take place one-on-one, such as Street Fighter or Tekken. Ufe also supports 3D as well as 2D. (Fig. 3.1) In addition, there are various attacks and combos for each character, and these can be easily changed. The character and new techniques can be added and developed at will. And for this research, we focused on one character, "Robot Kyle".

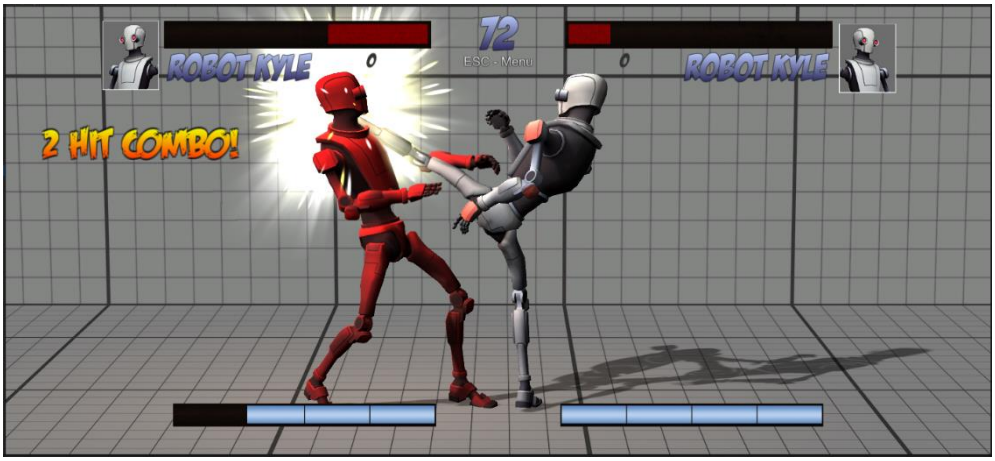

Figure 2.1: Ufe's battle screen "Character: Robot Kyle".

2.1 Game rule

In Ufe, each round lasts 99 seconds. The winner is determined after each round. A round can end in two ways: either the time runs out or the opponent's health is reduced to zero. If both players are still alive when time runs out, the player with the most health wins. In addition, if both players have the same remaining health when the time expires, the game is a draw, and the number of rounds won remains the same for both players. The player who gets the first set number of rounds wins the game.

2.2 Move Type

There are three types of action types.

Movement: Basic Move

Attacks: Action moves.

Moves that require multiple button presses: Special Moves.

The details of each type of move are given in Table 2.1.

	Move Name
Basic Move	Neutral, MoveForward, MoveBack, Jump_Vertical, Jump_Forward, Jump_Back, Crouch, Crouch_Back
Attack Move	Punch_Standing_Light, Punch_Standing_Heavy, Punch_Crouching_Light, Punch_Crouching_Heavy, Punch_Jumping_Light, Punch_Jumping_Heavy, Kick_Standing_Light, Kick_Standing_Heavy, Kick_Crouching_Light, Kick_Crouching_Heavy, Kick_Jumping_Light, Kick_Jumping_Heavy,
Special Move	Dash, Fireball_Light, Fireball_Heavy, Focus, Wall_Launcher

Table 2.1 Robot Kyle action

2.3 Status

Ufe has a debugger mode. When it is on, you can get a variety of information during a match. The information that can be obtained is listed in table 2.2 below.

	Means
Move Information	Name of current action
Position	Current position (x, y, z)
Life Points	Remaining life points
Gauge Points	Gauge for activating special moves
State	Current state. (Stand, Down, Crouch, NeutralJump)
Sub State	Current sub-state. (Resting, MovingForward, MovingBack, Blocking, Stunned)
Stun Time	Elapsed time from stun to return.
Combo Hits	Number of combo hits.
Combo Damage	Damage from combo.

Table 2.2 Information obtained in debug mode.

By displaying this information, it is possible to see briefly how close game is. Also, with Move Information and State turned on, developers can see what decisions the AI is making in the screen.

2.4 Random AI

Two types of AI exist in Ufe: FuzzyAI and RandomAI. Random AI was used in this research. Random AI sets weights based on distance to determine which buttons to press. The table below lists the features that can be set in the random AI and the performance of the random AI used.

	Means	Setting
Attack When Enemy is Down	Whether to attack when the enemy is down	False
Move When Enemy is Down	Whether to move when the enemy is down	True
Input Frequency (second)	Frequency of AI input per second.	0s
Distance Behaviors	Separate the actions to be performed for each distance.	-
Opponent Distance	Name by distance. (Value from 0 to 100)	Close (Between 0 and 30) Mid (Between 31 and 70) Far (Between 71 and 100)
Move Forward Probability	Priorities moving forward (Values from 0.0 to 1.0)	Close (0.5), Mid (0.7), Far (0.9)
Move Back Probability	Priorities moving back (Values from 0.0 to 1.0)	Close (0.5), Mid (0.4), Far (0.1)
Jump Probability	Priorities jump (Values from 0.0 to 1.0)	Close (0.6), Mid (0.5), Far (0.6)
Crouch Probability	Priorities crouch (Values from 0.0 to 1.0)	Close (0.5), Mid (0.5), Far (0.5)
Attack Probability	Priorities attack (Values from 0.0 to 1.0)	Close (0.9), Mid (0.2), Far (0.1)

Table 2.3 Random AI setting details

The above settings indicate that the Random AI is a very aggressive AI that prefers close combat. To make it more aggressive, the "Input Frequency (second)" is set to 0 seconds, so that it always presses some button. Long range attacks exist in Ufe. An AI that continuously launches them is not a fun opponent to fight with, even if it has a high win rate. This is not the purpose of this research, so to make the AI more approachable, we set it to move closer when it is in the "Far" position. This Random AI was used as an opponent for the DQNAI created this time. You play against this random AI many times and the AI learns its behavior.

Chapter 3

Unity ML-Agents

Unity Machine Learning Agents (ML-Agents) [8] is an open-source framework for building a "machine learning" learning environment in Unity. It allows us to optimize the behavior of our characters by letting them perform reinforcement learning. Also, ML-Agents also provides tools to enable reinforcement learning called Proximal Policy Optimization (PPO)[9] and Soft Actor-Critic (SAC). Therefore, simply adding the Brain function and Reward in the code on the Unity side of the code allows for reinforcement learning of each. ML-Agents requires the placement of three part in a Unity scene: Agents, Brain, and Academy. The Agent part is needed to define the states, actions, and rules for learning. Reward settings are also included in this part. The Brain part will be the part that thinks about the action and endorses and controls the action to Agent. The Academy part manages the learning environment, including the speed of the learning steps and the amount of learning per episode. For this research, an AI trained with PPO with the same reward settings was created for comparison.

3.1 Structure of ML-Agents

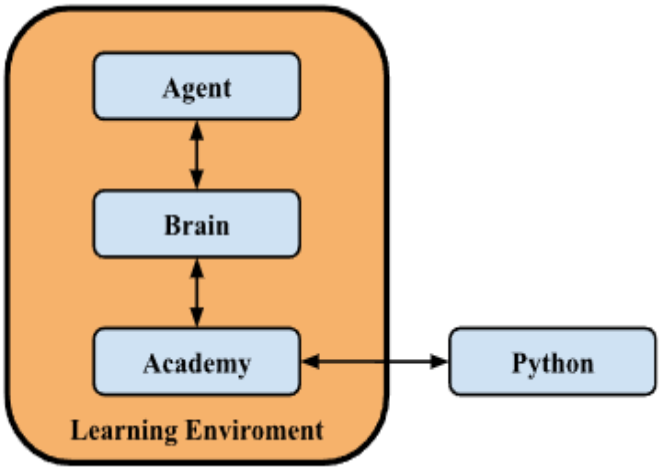

Figure 3.1: Structure of ML-Agents

Figure 3.1 shows the structure of ML-Agents, each of which has three parts in the Learning Environment on the Unity side. The Agent and Academy parts are linked to the Brain, and the Academy part is linked to Python.

3.2 ML-Agents Functions

There are several functions that are required to train the AI. The functions performed by those functions are listed in the table below.

	Mean
OnEpisodeBegin	Function executed at the start of an episode.
CollectObservation	Check Agent status
OnActionReceived	Receive and execute actions from the Python side
SetReward	Set the reward to the value received.
AddReward	Add the received value to the current reward
EndEpisode	Episode ends

Table 3.1: Learning function.

The above are the functions mainly used on the Unity side to implement ML-Agents. OnEpisodeBegin is a function that is executed every time an episode starts. Therefore, it is used when initializing the character's status such as position and health. CollectObservation is a function that observes the state of the agent. Therefore, it has the role of observing and saving actions such as executed actions and current position. OnActionRecieved is a function to execute the determined action, move it, and get the reward. AddReward and SetReward are performed in this. Finally, if the conditions are met, EndEpisode is executed and the episode ends. AddReward and SetReward are both reward functions, but there are distinct differences. AddReward adds the reward value during this episode. SetReward sets the reward value for this episode to the received value. So if AddReward is done before SetReward, the AddReward function becomes meaningless. Incorporate these functions into Ufe to create the Unity side of AI.

Chapter 4

Human-Like Behavior

The most important aspect of this research is human-like behavior. The reason why human-like behavior is important is that humans enjoy games with NPCs that act like humans more than NPCs that act like machines [8] Furthermore, there are three advantages of AI with human-like behavior [1]

1. Increases immersion in the game.

2. One person can participate in a multiplayer game.

3. AI can become a tester that predicts the movements of actual human players.

The first is that players will concentrate more on the game in order to defeat an opponent who offers various strategies like a human being, rather than an opponent who only performs mechanical predetermined moves. Therefore, this leads to an increase in the playing time of the game. Second, when multiplayer is needed, if there are not enough players, one can join instead. It is also possible to imitate the play in place of a player who is disconnected from the game in the middle of the game. Third, when you need to counter a particular player's moves, you can use a human-like AI to imitate and practice the player's moves. In addition, while the goal of this research is to create an AI that moves in a "human-like" manner, it is also important for humans to be able to enjoy the game at the same time. Therefore, a "human-like" AI can entertain humans more than a normal NPC as an opponent. Below we discuss what we mean by "human-like".

4.1 Evaluation Method

The Turing test[6] is a typical method of evaluating human-like AI. In the Turing test, the judges determine whether the player currently moving is an AI or a human. Therefore, it is not judged only by the values obtained, such as similarity or game score. The Turing test we will perform in this research is slightly different. Videos of the matches are viewed by a human judge. The judges will then determine whether the target player is being operated by a human or an AI. The opponents in the videos will be unified to the RandomAI introduced in Chapter 2. In other words, the match will be composed of RandomAI vs. human or RandomAI vs. created AI.

4.2 Human-like behavior in fighting games

The humanness of AI changes from game to game and scene to scene. In this research, we examined human-like behavior in fighting games. To begin, we interviewed fighting gamers. The reason for this was that since the Turing test was used to test this research, we thought it would be best to get input from people who actually play the games. The target audience for these interviews was people with extensive fighting game experience. Specifically, we limited the interviews to those who have been playing fighting games for at least one year. There are two "human-like behavior" that I had in mind before the interview. The first is feint, which is to intentionally make a mistake to induce an opponent to attack and counter. Therefore, we thought that an AI that feints would be more human-like. The second is impatience. Humans become impatient in a pinch. Therefore, if their physical strength decreases, their movements become a little unnatural. We considered this point to be human-like.

Several similar responses emerged from the interviews. Those are the following three.

1. The reflexes were not too fast.

2. Change the tendency of the technique depending on the distance.

3. People can make moves that work to their disadvantage.

The first is that it is very unnatural to perform actions at a reaction rate that is clearly more than that of a human being, and therefore less human-like. Second, people do not keep attacking at arm's length, so if it makes such a move, people will know immediately that it is an AI. Finally, one can always make mistakes. It is impossible to keep every move perfect. Thus, even very good play becomes suspect. Based on the above, the following two points are defined as "Human-Like" in this research.

- **Sometimes attacks that seem to hit the target.**
- **The type of attack varies depending on the distance.**

Chapter 5

Building the AI

Because this research used reinforcement learning, the reward value was set at the beginning. Then began building the AI. DQN was used for reinforcement learning. As a comparison, we created an AI trained using the reinforcement learning PPO that exists in ML-Agents without changing the reward settings. Each AI created in this research was trained for 100000 steps, with 60 frames as one training step.

5.1 Difference between DQN and PPO

Both DQN and PPO are reinforcement learning and therefore learn based on reward values. However, a clear difference exists: DQN is used when visual information is important, and the agent is aware of the current situation and chooses the best course of action. In contrast, PPO selects the optimal action based on data collected in the past.

5.1.1 DQN

DQN has state s and action a. It then chooses the action a that maximizes Q(s, a) and receives a reward. Therefore, the following states are considered in advance. The changed state is then saved and the action is selected again. This is repeated to learn. The learning method saves the state and action for each set step, and the reward is given.

5.1.2 PPO

Like DQN, PPO also has a state s and an action a. However, PPO does not refer to the next state, but learns by selecting the optimal action based on the past collected data. The learning method is the same as in DQN, and it learns step by step.

5.2 Learning Interval

Ufe is a 60fps game. Therefore, a frame buffer is created and set up to store actions every 60 frames and reward the player according to the actions performed in the frame buffer. Also give a reward value based on the win or loss at the end of the game. Below is the code for processing frames. Once the frame buffer exists, it loops through a for statement and calculates the damage. Damage received and damage inflicted affect the reward value, respectively. Their formulas are described in 5.3.

```
// RLAgent.cs
if(frameBuffer.Num > 0) {
	for (int i = 0; i < frameBuffer.Num; i++) {
		if (frameBuffer[i].frame == lastFrame) break;
		if(frameBuffer[i].blocking) {
			Damage A(Opponent Blocked)
		} else {
			Damage B
		}
			Damage C
	}
	lastFrame = frameBuffer[0].frame
}
Reward Calculation
```

5.3 Setting Reward

Set the reward value, which is the most important part of creating a human-like AI through reinforcement learning. In order to reproduce the human-like characteristics described in Chapter 4, we determined the reward values as follows.

Name	Value	Mean
Win	10	Agent win
Lose	-10	Agent loses
Damage Reward	Variable	Calculate from Damage A B C values
Damage A (AI Blocked)	Variable	Damage sustained
Damage B	Variable	Damage sustained
Damage C	Variable	Damage inflicted

Table 5.1: Value used for rewards.

The rewards for winning and losing were originally planned to be 1, -1, but as noted above, the use of giving rewards every 60 frames meant that 1, -1 did not have much of an impact. Therefore, it was necessary to increase each of the reward values to 10, -10 respectively.

1. **Win**
$$r = 10$$

2. **Lose**
$$r = -10$$

3. **Damage A (AI Blocked): Damage received if AI blocks**
$$a = \frac{damageTaken}{maxDamage}$$

4. **Damage B: Damage received without AI blocking**

$$b = \frac{damageTaken}{maxDamage}$$

5. **Damage C: Damage received from AI**

$$c = \frac{damageRecieved}{maxDamage}$$

6. **Damage Reward: Damage inflicted by the AI on the opponent.**

$$r = c - (\frac{a}{2} + b)$$

As a prerequisite, the AI made a winning stand. In addition, the Damage Reward makes them move in such a way that they do not take too much damage. In addition, in order to achieve the "Human-Like" described in Chapter 4, the calculation is now done frame by frame. The reason for this is that close-range attacks are faster, so multiple attacks can be launched within 60 frames. However, if the attacker is only at close range, he will be hit by more bullets. Therefore, in order to avoid taking damage, they will

also perform long-range attacks with slower attack speeds. Therefore, we set up the system so that there would be more variations of attack types. Because people try to win when they play. Therefore, we felt that rewards for winning or losing were necessary. However, since DQN operates to take the maximum reward, it learns mechanical movements because it gradually moves to get the reward efficiently instead of human-like movements as it continues learning. Therefore, we wondered if it would be possible to retain a human-like quality by adjusting the number of times it learns.

Chapter 6

Experiment

The experiment was used the Turing test[6] The Turing test is a test in which a human judge is asked to look at an AI or human movement and decide which it is. We chose this test because it is a human who plays the game. It is of utmost importance that humans actually enjoy the game when they play it. Therefore, we thought that the jury to judge whether the game is AI or human needs to be human. Match videos were taken to conduct this test. The match was played for one round. Each of the three players, DQNAI, PPOAI, and a human, played against RandomAI. The judges watched videos of each match and answered the following questions.

6.1 Play Video

There are three play videos in total. Video 1 is a match played by an AI trained by DQN. Video 2 is a game played by a human. Video 1 is a game of AI trained by PPO.

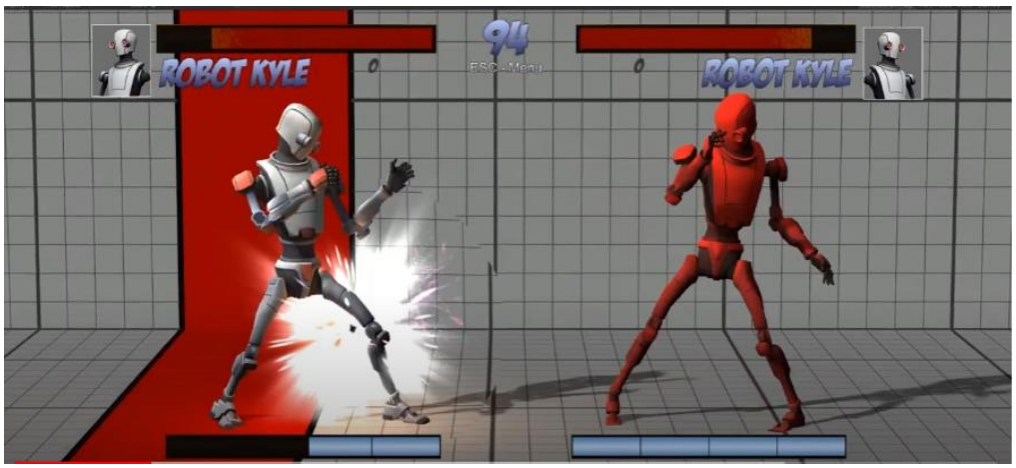

Figure6.1: Screenshot of the play video.

As shown in Figure 6.1, the test player starts on the right side. Its red color makes it easy to distinguish it from your opponents. Also, since the characters are the same, it is easy to understand the difference in movement between them and the opponent.

6.2 Test Questions

In the test, participants were asked to watch a video of the play and answer the following two questions. The play video is a one round match, and the opponent is a RandomAI within Ufe with the same character. Since the same characters fight each other, it may be difficult to tell them apart. That's why I set the character on the operating (AI or human) side in red. The test asked questions of both beginners and experienced person.

1. **Is red robot character controlled by**
   Choice: AI
   Human
2. **Watch video and select the item that applies to (Human or AI)**

   Choice: Easy to see through.

   I understand, but it was similar.

   It was pretty similar and difficult.

   I did not understand.

The second question asks how confident the respondent is that it is an AI, in order to see if there were any actions present that would help the respondent discern the answer. At the end of the questions, there was a comment box that could be filled in voluntarily, and the participants were asked to write their own thoughts.

Chapter 7

Result

As described in Chapter 6, the test was conducted by the judges, who watched videos of DQNAI, PPOAI, and a human each playing against RandomAI and answer questions.Video 1 shows DQNAI vs. RandomAI, Video 2 shows human vs. RandomAI, and Video 3 shows PPOAI vs. RandomAI. And 65 people cooperated with the questionnaire. The results of the experiment are as follows.

Video 1(DQN vs RandomAI)

Choice	Percent (%)
Easy	18.5
Similar	23.1
Pretty Similar	26.2
Didn’t understand	32.3

Table 7.2: Answer to question 2 in video 1

Question 1 in Video 1 was answered by **47.7% of respondents as human.** The fact that about half of the respondents recognized it as human suggests that DQN's AI can move like a human. “Didn’t understand" was the most common response to Question 2 in Video 1. This makes it difficult to determine whether they were actually human-like. However, the next most common response was "Pretty Similar”. Therefore, it can be assumed that the respondents were taking many actions that were difficult to judge.

Video 2(Human vs RandomAI)

Choice	Percent (%)
Easy	23.1
Similar	24.6
Pretty Similar	18.5
Didn’t understand	33.8

Table 7.2: Answer to question 2 in video 2

Question 1 in Video 2 was answered by **61.5% of respondents as human**. More than half of the respondents identified the players as human. Furthermore, "Similar" was the second most common response after "Didn't understand" in Question 2, but the percentage was very close to that of "Easy". This suggests that players understand the part of unconsciously judging "human-like movements.

Video 3(PPOAI vs RandomAI)

Choice	Percent (%)
Easy	16.9
Similar	23.1
Pretty Similar	26.2
Didn't understand	33.8

Table 7.3: Answer to question 2 in video 3

Question 1 in Video 3 was answered by **43.1% of respondents as human**. This shows that they were judged to be more AI than Video 1 (DQN). However, about 40% of the respondents also judged it to be human, suggesting that it was moving like a human. The value for question 2 was almost the same as that of the DQN AI, suggesting that respondents understood the difference between "human-like" and "unnatural" movements.

As a result, the judges were able to distinguish between human-played matches. However, about half of the judges were incorrect about DQNAI. In addition, PPOAI, which was used for comparison, was incorrect by about 40% of the judges, but the results were not very different from DQNAI. This suggests that we were able to create an AI that performs human-like movements. In addition to these results, several comments were solicited. One of these comments was particularly interesting.

"It was good that they were all aggressive, but I think you should be aware of starting the movement from the sidestep."

"It was AI-like to suddenly swing away from close-range moves while at a distance, even though they were deciding difficult-looking combo moves with great precision."

These factors required us to be careful not only in our attacks but also in the way we moved. In addition, the fact that the AI would make an obvious mistake after a good action was "AI-like" in its movements. Therefore, our feints could have had the opposite effect.

Chapter 8

Conclusion

In this research, DQN, a reinforcement learning method, was used to create an AI that performs "human-like" movements so that humans can enjoy playing alone. Reinforcement learning was chosen because of the question of whether it is possible for an AI to perform human-like movements from scratch, rather than imitating them through supervised learning or other methods. The difficult part of this task was to understand what movements were human-like. Therefore, we solicited opinions on "human-like movements" in fighting games from papers and people with fighting game experience. Then, we determined the "human-like movements" in fighting games. Next, we used Unity's Ufe as the environment for creating AI and introduced ML-Agent for reinforcement learning. Then, we created two types of AI, DQN and PPO. The strength of the AI created here is not important." It is important that the AI performs "human-like movements.

The created AI was then examined by the Turing test to see if it was capable of human-like movements. As a result, it was able to fool about half of the respondents. Moreover, more than half of the respondents correctly answered that the video of a human operating the AI was human, suggesting that humans subconsciously understand "Human-Like Behavior" within themselves. However, we must consider that this result is not everything. The reason is that it is difficult for humans to evaluate a large number of videos in the first place. In addition, if the playing style changes from video to video, the criteria for judgment may shift. Therefore, it was necessary to limit the test to a small number of videos.

We believe that creating an AI that acts like a human will lead to a better understanding of how humans often behave and how their behavior changes. We believe that this will be useful not only for this fighting game, but also for MOBA games with a high level of strategy and will contribute to the future growth of the E-sports field.

References

[1] M. Swiechowski, “Game AI competitions: Motivation for the imitation game-playing competition,” in 2020 15th Conference on Computer Science and Information Systems (FedCSIS). IEEE, 2020, pp. 155–160.

[2] Y. Zhao, I. Borovikov, F. de M Silvam, A. Beirami, J. Rupert, C. Somers, J. Harder, J. Kolen, J. Pinto, R. Pourabolghasem, J. Pestrak, H. Chaput, M. Sardari, L. Lin, S. Narravula, N. Aghdaie, K. Zaman, “Winning Isn’t Everything: Enhancing Game Development with Intelligent Agents” IEEE Transactions on Games (Volume: 12, Issue: 2, June 2020), pp. 199–212.

[3] P. Hingston, “A turing test for computer game bots,” IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 3, pp. 169–186, 2009.

[4] A. M. Turing, “Computing Machinery and Intelligence, ” Mind, vol. 59, no. 236, pp. 433–460, 1950.

[5] D. Livingstone, “Turing’s test and believable AI in games,” Computers in Entertainment (CIE), vol. 4, no. 1, pp. 6–es, 2006.

[6] F. Hsu, “IBM’s deep blue chess grandmaster chips,” IEEE micro, vol. 19, no. 2, pp. 70–81, 1999.

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., “Mastering the game of Go with deep neural networks and tree search,” nature, vol. 529, no. 7587, pp. 484–489, 2016.

[8] I. Umarov and M. Mozgovoy, “Believable and effective AI agents in virtual worlds: Current state and future perspectives,” International Journal of Gaming and Computer-Mediated Simulations (IJGCMS), vol. 4, no. 2, pp. 37–59, 2012.

[9] “Universal Fighting Engine” [Online] Available: http://www.ufe3d.com/

[10] “Unity ML-Agents” [Online] Available: https://unity.com/ja/products/machine-learning-agents

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, “Proximal Policy Optimization Algorithms” [Online] Available: https://arxiv.org/abs/1707.06347v2