Catalogue

1 Introduction

1.1 Introduction to the domain

1.1.1 AI in games

AI, also called machine intelligence, has been an area with various branches and increasingly important in the world after decades of development. Being the foundation in video games, artificial intelligence (AI) is always used to generate responsive, adaptive or intelligent behaviors primarily in non-player characters in human-like intelligence. On one hand, AI is an essential part of games, having been an integral part of video games, from the hardcoded simple Pac-Man to the AlphaGo and other products of Deepmind and OpenAI which were based on cutting-edge big data and deep learning technology. On the other hand, the game has been a different (and sometimes better) platform to research AI, for the result of the experiment is easy to understand, and close to our daily life, and cost less.

Though the mainstream game category appearing 30 years ago, the core of games did not change so much, and it is still worthwhile to dig into the game AI for more creation and the research of the interaction between humans and AI, i.e. making the AI in games more believable. Unlike the past, when game AI was mainly hard-coded or implemented by simple status machine and tree search was the most frequently used algorithm, machine learning technology is one of the most popular techniques to develop highly qualified AI currently. For the tree search algorithm, in games like chess, designers can implement AI-based on Game Tree to find the optimal solution. However, the cost of searching the solution is in direct proportion to the complexity of the original question. As simply searching for the best solution by some Tree Search algorithm is not always effective, the combination of searching and minmax algorithm and heuristic searching can be more achievable for simple chess-like games i.e. the Deep Blue was designed on chess custom chip with alpha-beta pruning algorithm and bruteforce searching. But this is not always working when developing AI for more complex games, like the board game Go where the complexity is greater than chess exponentially, as is shown in Table 1-1. (腾讯游戏学院（2019），游戏 AI：AI 的游戏还是游戏的未来，https://www.gcores.com/articles/113763#nopop_2iqn5)

Game	State-space complexity	Tree-search complexity
Tic-Tac-Toe	10^3	10^5
5-In-Row	10^{28}	10^{58}
Checkers	10^{30}	10^{54}
Chess	10^{47}	10^{123}
Go	10^{171}	10^{360}

Table 1-1 The complexity of state space and tree-search of different games

(Reference from https://www.gcores.com/articles/113763#nopop_2iqn5)

Aforementioned methods have some drawbacks:

a) Limited to the computing power, AI cannot give out prognosis according to the current situation in Go (or more complex games).

b) The competition in a partial environment should not totally based on some pattern algorithms and pure searching. The solution should take the intuitive decision as supplements.

c) AI should be available to break mindset, for creating a new behavior pattern.

In recent 10 years, deep learning began to play a critical role in AI development with the help of big data and the decreasing cost of computing. AI trained by deep learning, like AlphaGo and OpenAI Five for Dota2 is proven as competitive as professional players. This is the route of game AI development, from being based on rules and search to exploring by itself, or more intuitively.

1.1.2 Game Engine

In the past, the game machine manufacturer only provided the basic hardware information of their machines, on which the game engineers developed the game. However, as the game is becoming more complex, it is much harder for some small game companies to develop a game, so some game companies began to use ‘middleware’ working on multiple OS, by which the developers can simplify their procedure of development (i.e. YEBIS 2 of Silicon Studio). As the workload of developing a game has become even heavier, the urgent need for a software which can take over most of the developing tasks. And game engines are now the most popular tools among game manufacturers. And one of the best current game engines is Unity3D, a game engine developed by Unity Technologies, aiming at developing middle and small-sized games and makes it easier to construct 3D video games and real-time 3D interactive games.

1.2 Motivations

As the AI technology is widely used and is likely to be the most important increasing point in the future, it will no doubt influence the world greatly. Games, one of the most popular ways to release pressure, which is also becoming more prevalent, can make good use of the cutting-edge AI technology.

Highly performed AI is the key point of almost all kinds of game development, such as Go, Texas Hold’em poker, Dota or even the simplest link game. But ‘highly performed’ here is not only to be agile, able thinking deeply, or just beating the opponent every time. Though many researchers are aiming at building an always-winner, however, believability is always the most essential point of game AI. The believability here is not to pass a conventional Turing Test, but just to fool players or observers that the game-AI-agent is a human.

So, in this project, two game demos are implemented and send to people to see whether they can distinguish AI from humans.

1.3 Research questions and hypothesis

1.3.1 Research questions

Given several videos of playing games, interrogators were asked to figure out which player is AI or human player, and if possible, the reason why they draw their conclusions.

1.3.2 Hypothesis

Develop two game demos, assign AI agents to each player, train the agents by reinforce learning, and achieve human-like game behavior.

Targets:

- In the first demo the AI recognized rate lower than 40%.
- In the second demo, the recognized rate should be lower than 20%.

The factors involved in proving the hypothesis:

1) Choosing the game engine-Unity3D, and learning how to build a game.
2) Choosing the script language-C#, and learning the language.
3) Choosing the environment to train AI players-ML Agents (Unity Machine Learning Agents), understand the architecture and the algorithm.
4) Time management.

1.3.3 Some supplement of hypothesis

1) As the experience of interrogators can somehow influence their judgment, game demos in this research are designed highly understandable.

2) The game genre also influences the accuracy, because some interrogators may find whether the entity being tested is an AI or not by some other aspect like the similarity of the players' strategy in a chess game or the mood of the NPC. More importantly, sometimes the game environment will be too complex to conclude, the

testers' attention can be easily attracted by the dizzying environment. To solely test whether the AI can be recognized, and shorten the testing time, the game demos in this research are simple physical-stimulation games rather than other types like strategy, RPG, etc.

Accordingly, the opinions and experience of testers are recorded and documented as an essential part of the process of improving the quality of game AI.

1.4 Overview of methodology

The methodological approaches are:

1) Review related works to develop game demos, PPO algorithms.
2) Develop the game demos with the game engine.
3) Train AI players.
4) Design questionnaires for testers.
5) Send the demos and questionnaires to more than 20 interrogators.
6) Collect the data and analyze it.
7) Conclude, verify the hypothesis.

The following data would be necessary:

1) The recognized rate of AI in each case.
2) How long the tester played games.
3) The reason why the interrogator thinks it is the AI or human.

The questionnaire is the main method to collect data in this research. The recognition of the AI player is an all-or-nothing test: the game AI will either be recognized or reckoned as a human. So, a multiple choices question like 'which do you think is an AI player, red or blue?' is fine for that.

The questionnaire is sent to the participants online, together with the game demos, and videos of them. There are some subjective items like ‘why you think the blue one is the human?’ to elaborate on the reason why the participants draw such a conclusion. Because in some cases, the interrogator might have found the difference between AI and human, but due to the lack of experience of the game, they would choose the wrong side, which is a paradox-like conclusion, it is necessary to consider those as well. Similarly, some participants may randomly point out the right answer, so it is reasonable to have an explanation of their conclusion. The bilingual questionnaire is attached in Appendix I.

2 Literature Review

This part is a review of the scholar sources i.e. books, articles and theses related to game design and training methods of game AI.

2.1 Machine learning

“It is a brunch of AI research focuses on learning patterns from data.”(Juliani et al. Unity: A General Platform – Background Machine Learning, 2018, para. 2). There are three different kinds of ML algorithms, supervised learning, unsupervised learning and reinforcement learning.

In this project, the reinforcement learning method is adopted which is indicated by Juliani et al. (2018) the process of learning a sequential decision making associated with controlling robots. (Unity: A General Platform – Background Machine Learning, 2018, para. 8). The robot may observe the environment, and continuously make decisions based on the environment. To a more specific aspect, the robot can learn a policy which is the mapping relationship between observations and actions. The observation means measurements of the environment made by the robot. To make some specific decisions deeper in mind of the robot, a reward signal is the keystone. If the robot has done the right thing, it will be rewarded for completing the task. And the robot will learn to achieve the target by a large positive reward.

2.2 Reinforcement Learning in Unity

According to Juliani et al. (2018), Reinforcement Learning is an artificial intelligence technique that trains agents to perform tasks by rewarding desirable behavior. The agent explores the context, observes and chooses an action in reinforcement learning. It acquires positive or negative reward according to the action, by which its behavior pattern can be optimized. The reinforcement learning process is implemented in TensorFlow in a Python process which communicates with a running Unity application over a socket. (Unity: A General Platform – Background TensorFlow, para. 1,2,3).

The architecture of ML-Agents and communication structure is shown in figure 2-1:

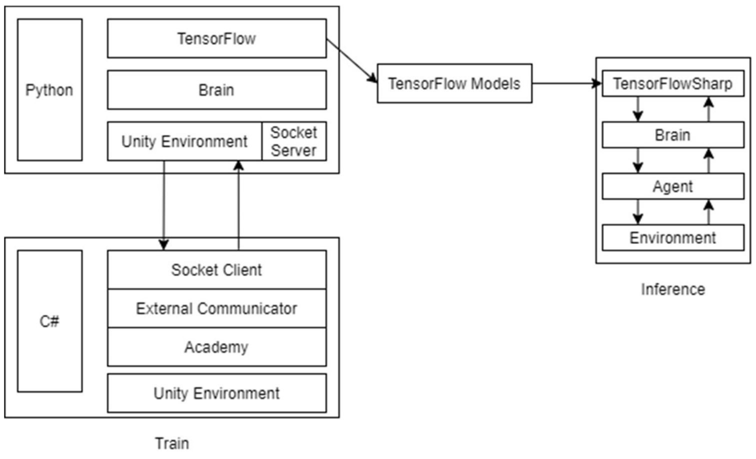


Figure 2-1 The architecture of ML-Agents and communication structure

The python side is the Socket Server, while the Unity3D (whose scripts are mainly in C#) is the Socket Client. The trained model is saved as bytes by TensorFlow, and the TensorFlowSharp is used to read the trained model at the inference stage, to interact with the environment as the brain in Unity.

Another perspective, ML-Agents has three components:

1) Learning environment: the scenes and all roles in the game.
2) Python API: includes all machine learning algorithms for training some behavior or policy. It is not a part of

Unity but communicates with it with the external communicator.

3) External Communicator: integrated in Unity, communicates with Python API.

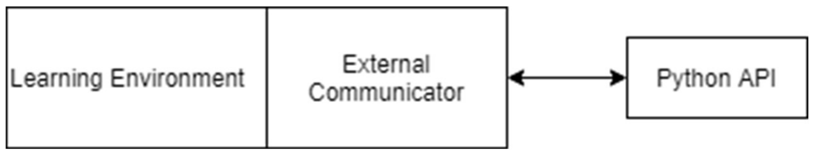


Figure 2-2

There are three components of the Learning Environment as well.

1) Agent: It is attached to a Unity GameObject (any role in games), generates its observing data, executes its action, and acquires rewards (either positive or negative). Each Agent is combined with a Brain(but multiple agents can also be combined with one brain as long as their behavior pattern is similar), it is the shell between the environment and the brain (kernel), which observes environment status and obey the order of the brain.
2) Brain: It encapsulates the decision logic of the agent and stores the policy of the agent where the agent should comply according to the situation. More specifically, the brain is the component that receives the observing and rewarding data and returns an action.
3) Academy: It controls agents during the process of observing and making decisions and maintains the environment parameters, such as render quality. And it is where the external communicator is located.

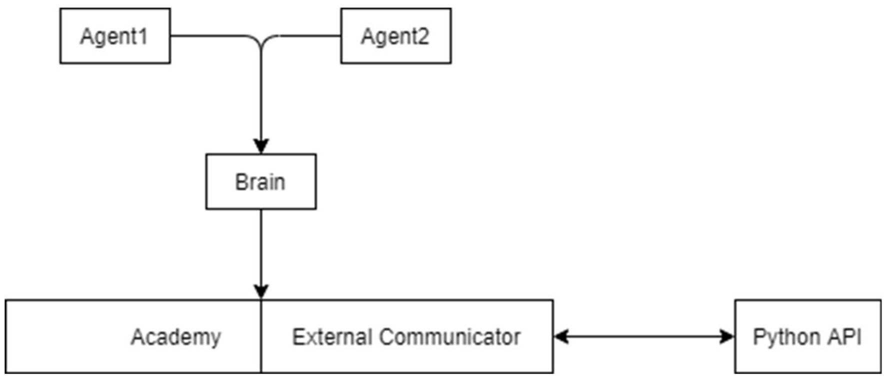


Figure 2-3

2.3 Tensorflow

According to Juliani et al. (2018), many of the algorithms in ML-Agents of Unity3D use some form of deep learning. Precisely, the ML-Agents toolkits are mainly based on the open-source library Tensorflow. It uses data flow graphs to compute and facilitates training and inference on CPUs and GPUs. After training with ML-Agents, the TensorFlow model is generated which then can be assigned to the agent. (Unity: A General Platform – Background TensorFlow, para. 1,2,3).

2.4 PPO Algorithm

"ML-Agents implemented the reinforcement learning algorithm called Proximal Policy Optimization (PPO), which performs better than other state-of-the-art approaches and is much simpler to implement and tune." (Juliani et al. 2018). It is also the default reinforcement learning algorithm at OpenAI because of its ease of use and good performance.

While policy gradient methods making it challenging to get a good result as its sensitiveness of the choice of stepsize, nor too small (slowing down the program), or too large (overwhelming the signal by noise), the PPO, ACER, and TRPO eliminates these flaws. However, ACER is more complicated than PPO where it needs more code to do off-policy corrections and replay buffer and sadly just a little bit better than PPO with all those done, and TRPO can be hardly compatible with the algorithms that share parameters between different policies. So, PPO is the best in this case with easy implementation of the cost function, run gradient descent and getting satisfying results with less hyperparameter tuning. (John Schulman et al. (2017) Proximal Policy Optimization).

2.5 Turing Test and Believable AI in Games

This paragraph is mainly based on Livingstone, D. (2006). Turing's test and believable AI in games.

This research project is actually a Turing test where the interrogator watches the video of the game segment and determine whether the player is an AI or human. “Some researchers think about building something of substance rather than a façade…But for game developers, the façade is what counts: it provides a simulation of intelligence to characters in a game world. Believability is more important than truth.” (Livingstone, D. (2006). Turing's test and believable AI in games. Computers in Entertainment, p. 2) Daniel Livingstone (2006) discussed what believable is and the means of testing the believability of AI, which depends strongly on the game genre and the role of AI. He took two games as examples, and the first example is an FPS (first person shooter) game, where the interrogator was asked to answer which of the avatars are AI. This was keeping more pattern in Turing’s Imitation Game where the machine played some role in human tasks. Then the second game was a strategy game, but Daniel was asking should the AI here be tested in the similar way as in the first game. In the strategy game, the AI is supposed to control multiple roles and to say whether an individual role is behaving like human makes no sense. So, in the cases where AI play different roles, like Player AI, NPC AI etc., the believability of AI should be measure separately based on different standard.

It is subjective to determine whether an AI player is believable and depends on the interrogator’s observation. And this leads to the use of questionnaires, designed to get the perception of the observers. Also if the interrogator is asked to observe the AI behavior by playing the game himself, he would be less sensitive because of the deep involvement into the game and may give some different results than a pure observer. However this also depends on the game genre, as in strategy games the interrogator was always given enough time to observe and to draw a conclusion based on the strategy the AI took is not as hard as in a FPS game.(Laird and Duchi (2000) Creating human-like synthetic characters with multiple skill levels: A case study using the Soar Quakebot.)

According to Daniel Livingstone, the AI believability criteria is shown in the table:

	AI should…
Plan	▪ demonstrate some degree of strategic/tactical planning ▪ be able to coordinate actions with player/other AI ▪ not repeatedly attempt a previous, failed, plan or action
Act	▪ act with human-like reaction times and abilities
React	▪ react to players' presence and actions appropriately ▪ react to changes in their local environment ▪ react to presence of foes and allies
Notable exceptions	▪ Might not apply where design/plot calls for impulsive or stupid characters, nor for animals ▪ Might not apply where design/plot calls for characters with significantly superior or inferior abilities ▪ Might not apply where game-design/plot call for characters with limited awareness

Table 2-1 AI believability criteria

Referenced from *Livingstone, D. (2006). Turing's test and believable AI in games.*

Though Daniel said "It is necessary to recognize that people who play few games have such limited experience of game AI that we cannot expect them to sensibly evaluate different versions of an AI…Accordingly, we argue that the opinions and experiences of players need to be recorded and documented as an essential part of the process of improving the quality of game AI." The game demos in this project are designed to be physic simulation demo, to minimize the influence of game experience of the interrogator, and whether the experience impacts the result will be tested in the questionnaire by asking related questions.

In this project the Act, React and Notable exceptions criteria will be taken into the design of the questionnaire because for the physic simulation game demo in this project the Plan does not make sense.

3 Design and Implementation

3.1 Designing a simple rolling ball game demo – the first demo

This is a simple ball-chasing-target demo, and it is the basic implementation of the second demo in chasing the target part.

3.1.1 Rules

The rules of this demo were identified as:

a) The player controls a rolling ball (the white ball is shown in figure 3-1) on a boundaryless plain (green) to chasing the target ball randomly generated.

b) Once the player touches the target (the red ball is shown in figure 3-1), a new target appears at a random position on the plain.

c) If the rolling ball falls off from the plain, the player fails.

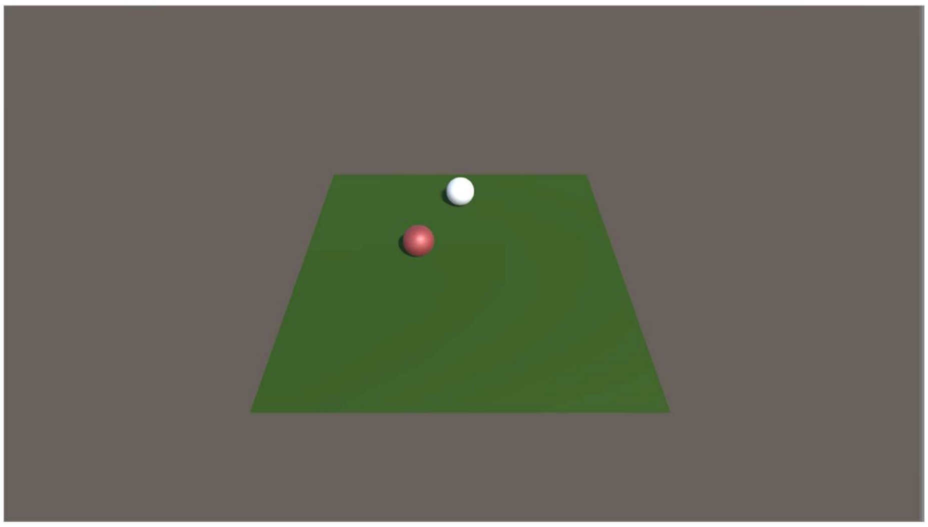


Figure 3-1 The game scene of the game demo

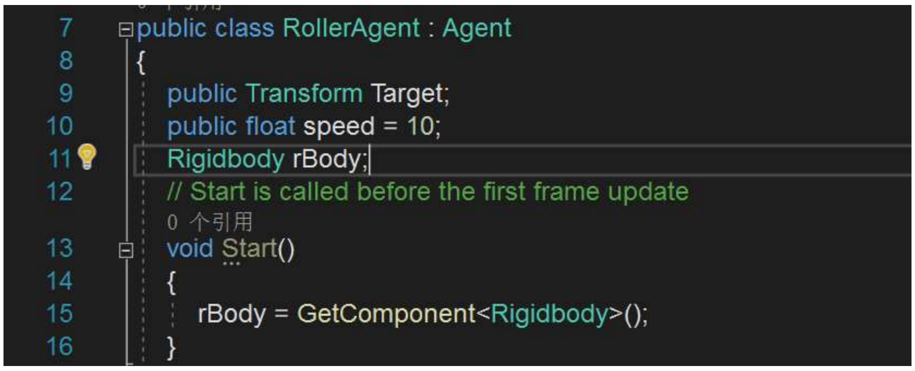
3.1.2 Implementing the rules with code

The RollerAgent extends the Agent class.

To begin with, the RollerAgent has three fields,

- Transform Target: the necessary component of each game object, maintaining the position, rotation, zoom, etc. of the game object. It is identified as public so that it can be assigned with some real target in Unity3D.
- float speed: the speed of the agent, set public to adjust in Unity3D.
- Rigidbody rBody: the Rigidbody reference which refers to the player ball.

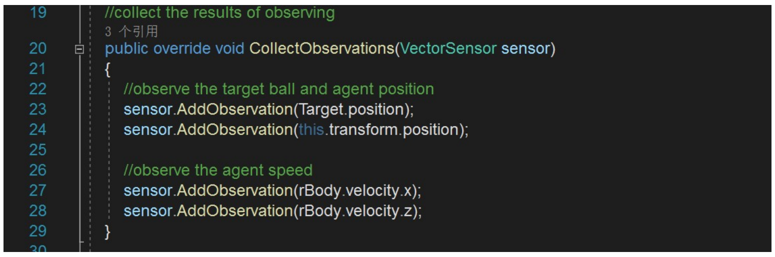
Start() method gets the rigid body of the player ball which holds this script. It is a default method of Agent which is called before the first frame is updated.



```
public class RollerAgent : Agent
{
    public Transform Target;
    public float speed = 10;
    Rigidbody rBody;
    // Start is called before the first frame update
    0 个引用
    void Start()
    {
        rBody = GetComponent<Rigidbody>();
    }
```

Figure 3-2 Fields and Start()

Method CollectionObservations (VectorSensor sensor) is used to collect the results of observing, taking a VectorSensor object as the parameter. The sensor observes the positions of the target ball and the agent, also, it observes the velocity of the x-axis and the z-axis of the player’s rigidbody. In this observation, the agent needs to observe two 3D positions and two 1D velocity, which leads to 8 floats space per observation unit.

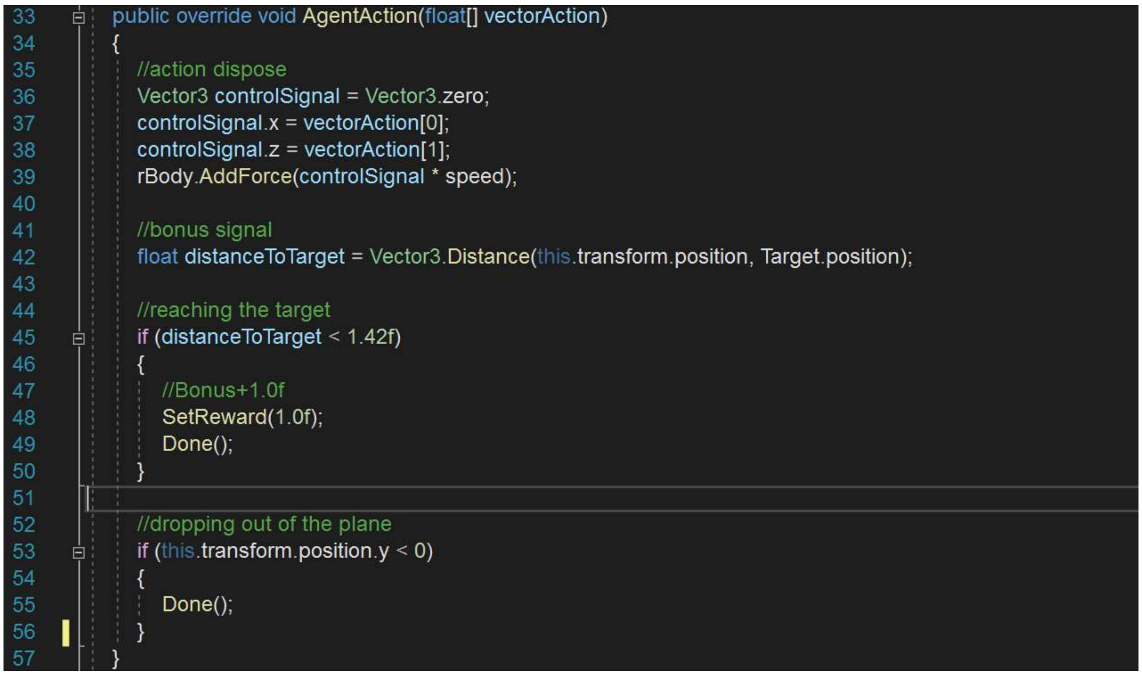


```
//collect the results of observing
3 个引用
public override void CollectObservations(VectorSensor sensor)
{
    //observe the target ball and agent position
    sensor.AddObservation(Target.position);
    sensor.AddObservation(this.transform.position);

    //observe the agent speed
    sensor.AddObservation(rBody.velocity.x);
    sensor.AddObservation(rBody.velocity.z);
}
```

Figure 3-3 CollectObservations (VectorSensor sensor)

Method AgentAction (float [] vectorAction) is in charge of the behavior of the agent. In this method, two float variables are assigned to Vector3 controlSignal to generate a vector quantity to move the player ball by calling AddForce(controlSignal*speed) with the rigidbody of the player.

The float variable ‘distanceToTarget’ is created to refer to the distance between the player and the target. If the player touches the target (distanceToTarget<1.42f), reward the agent with bonus +1 and call Done (), or if the player drops from the plain (y<0), also call Done().

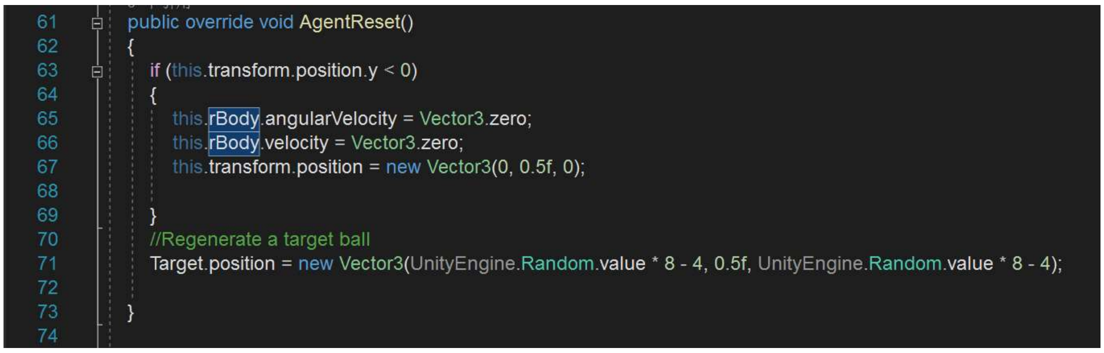


```
public override void AgentAction(float[] vectorAction)
{
    //action dispose
    Vector3 controlSignal = Vector3.zero;
    controlSignal.x = vectorAction[0];
    controlSignal.z = vectorAction[1];
    rBody.AddForce(controlSignal * speed);

    //bonus signal
    float distanceToTarget = Vector3.Distance(this.transform.position, Target.position);

    //reaching the target
    if (distanceToTarget < 1.42f)
    {
        //Bonus+1.0f
        SetReward(1.0f);
        Done();
    }

    //dropping out of the plane
    if (this.transform.position.y < 0)
    {
        Done();
    }
}
```

Figure 3-4 AgentAction(float[] vectorAction)

This method is called after Done (), resets the environment and prepares the next session. When Done () is called, it means the end of one session. The machine learning model evaluates the reward of the whole session and reset the situation.

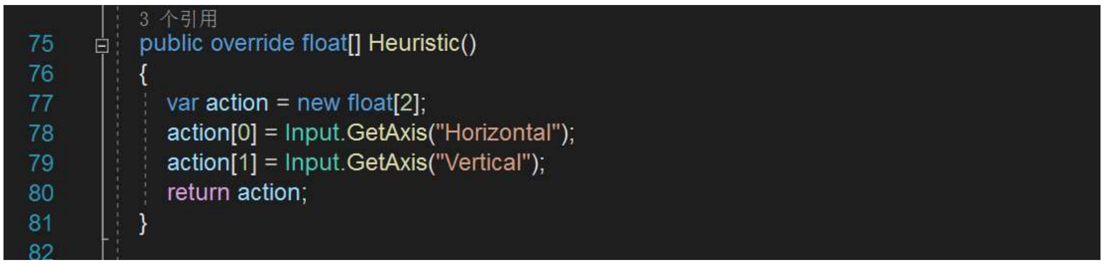
During training, when the player falls off the plain (this.transform.position.y <0), the velocity and angular velocity of the player are set to zero, and the player is repositioned in the original point. And in a new session, the target can appear at any position on the plain.



```
public override void AgentReset()
{
    if (this.transform.position.y < 0)
    {
        this.rBody.angularVelocity = Vector3.zero;
        this.rBody.velocity = Vector3.zero;
        this.transform.position = new Vector3(0, 0.5f, 0);

    }
    //Regenerate a target ball
    Target.position = new Vector3(UnityEngine.Random.value * 8 - 4, 0.5f, UnityEngine.Random.value * 8 - 4);

}
```

Figure3-5 AgentReset()

This is the Heuristic method, can be used in imitation learning and testing the control script. It takes two inputs from the keyboard, then returns the action array.



```
3 个引用
public override float[] Heuristic()
{
    var action = new float[2];
    action[0] = Input.GetAxis("Horizontal");
    action[1] = Input.GetAxis("Vertical");
    return action;
}
```

Figure 3-6 Heuristic()

3.2 Designing a simple balance ball demo – the second demo

3.2.1 Rules

The rules of this balance ball demo were identified as:

a) The balance ball (the billiards shown in figure 3-7) is assumed to move from the start point to the cannon.

b) It has to stay on boards (shown in figure 3-7), if it falls off, it fails.

c) Once the balance ball touches the cannon (shown in figure 3-7), it is launched by the cannon and blow off the wall.

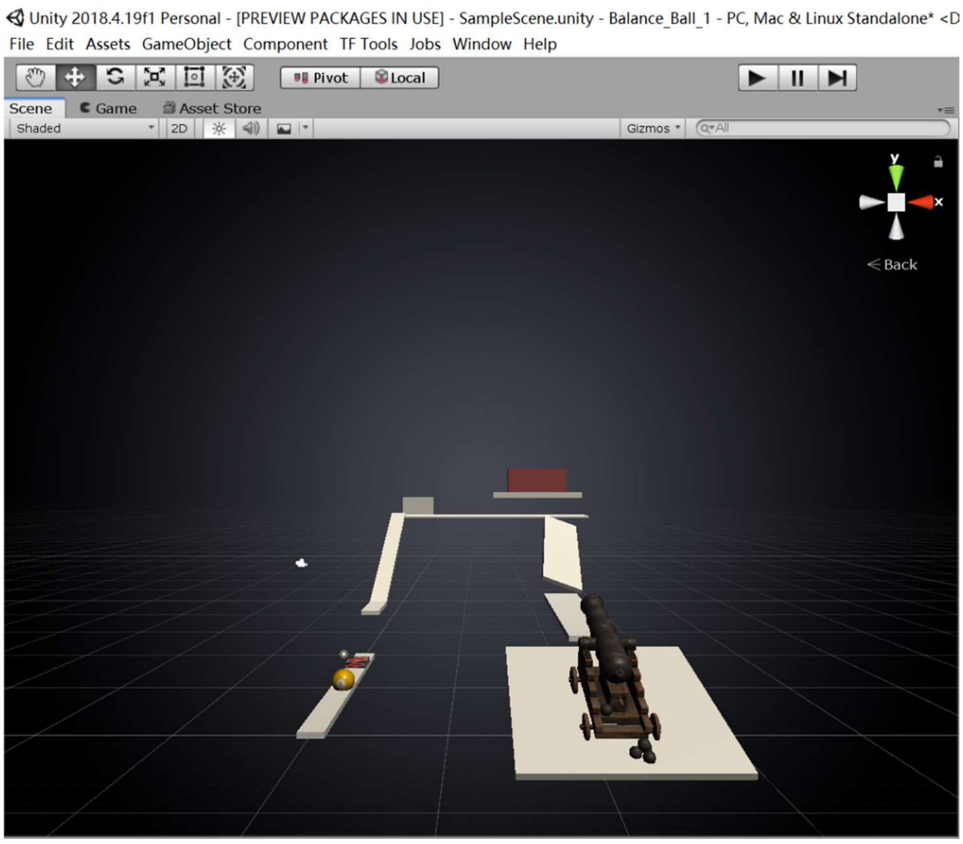

Unity 2018.4.19f1 Personal - [PREVIEW PACKAGES IN USE] - SampleScene.unity - Balance_Ball_1 - PC, Mac & Linux Standalone* <D
File Edit Assets GameObject Component TF Tools Jobs Window Help
Pivot
Local
Scene
Game
Asset Store
Shaded
2D
Gizmos
All
y
x
Back

Figure 3-7 The game scene of demo_2

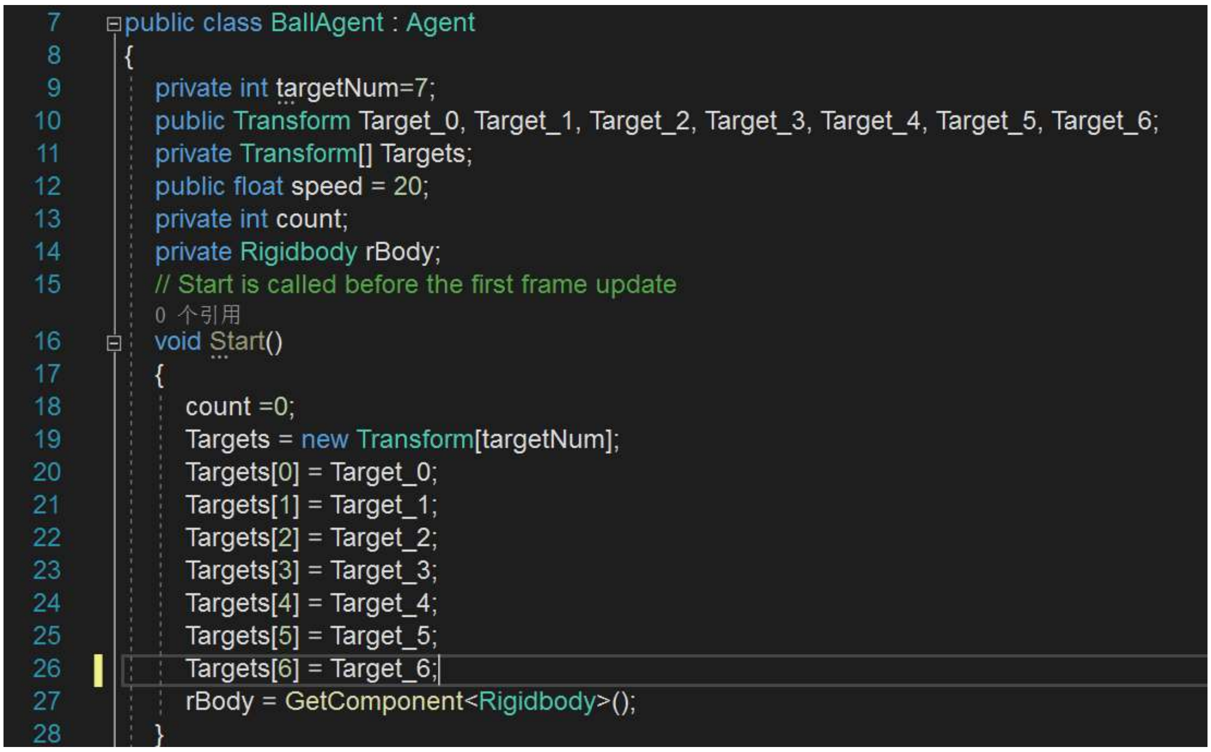
3.2.2 Implementing the rules with code

The BallAgent inherits from Agent, and it has the following fields:

targetNum: is the number of targets, because of the complexity of the map, several reward targets are set for the agent observing the environment.

- Transform Target_0 ~ Target_6: The transform of the 7 targets.
- Transform[] Targets: the array of Transform.
- Speed: the speed of the agent.
- Count: counter of how many targets have been touched.
- Rigidbody rBody: The reference of Rigidbody of the agent.

In the Start() method, the ‘count’ is initialized as 0, and the array of Transform and the rBody are all initialized.

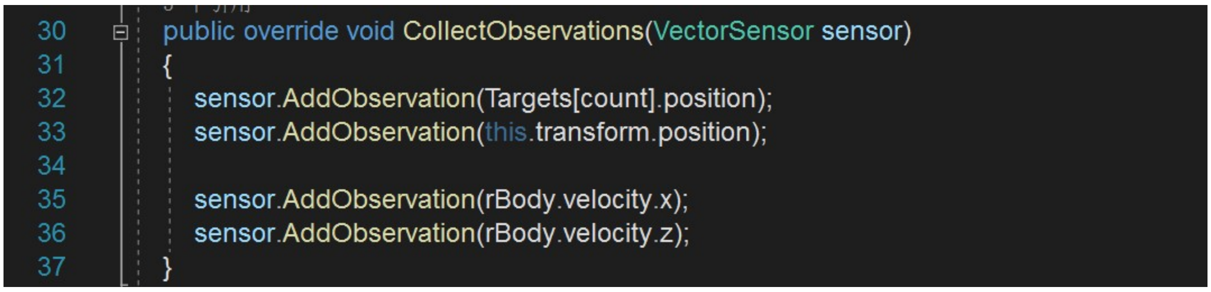


```
public class BallAgent : Agent
{
    private int targetNum=7;
    public Transform Target_0, Target_1, Target_2, Target_3, Target_4, Target_5, Target_6;
    private Transform[] Targets;
    public float speed = 20;
    private int count;
    private Rigidbody rBody;
    // Start is called before the first frame update
    0 个引用
    void Start()
    {
        count =0;
        Targets = new Transform[targetNum];
        Targets[0] = Target_0;
        Targets[1] = Target_1;
        Targets[2] = Target_2;
        Targets[3] = Target_3;
        Targets[4] = Target_4;
        Targets[5] = Target_5;
        Targets[6] = Target_6;
        rBody = GetComponent<Rigidbody>();
    }
```

Figure 3-8 The fields and Start() of demo_2

The CollectObservations (VectorSensor sensor) method:

Same as demo_1, it observes the position of Targets and the agent, the velocity of x and z.

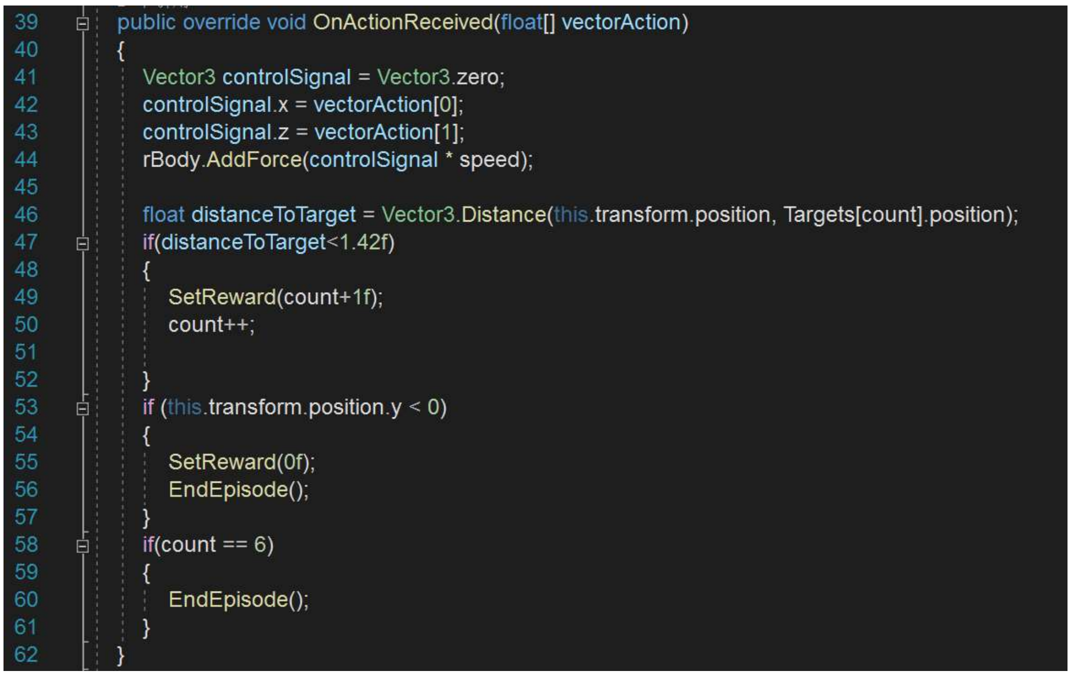


```
    public override void CollectObservations(VectorSensor sensor)
    {
        sensor.AddObservation(Targets[count].position);
        sensor.AddObservation(this.transform.position);

        sensor.AddObservation(rBody.velocity.x);
        sensor.AddObservation(rBody.velocity.z);
    }
```

Figure 3-9 The CollectObservations(VectorSensor sensor) method demo_2

In OnActionReceived (float [] vectorAction), similar to demo_1, two controlSignals on x-axis and z-axis are assigned with the float [] elements, and add force to the rigid body of the agent.

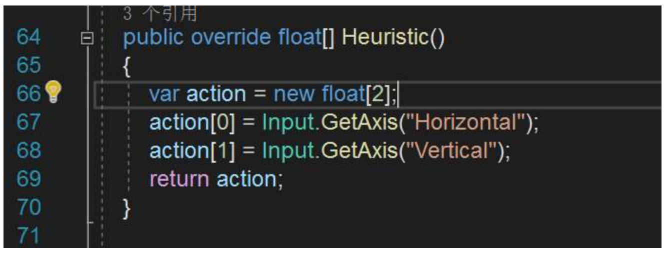
Also, the distance between the agent and the current target is the signal of rewarding. Once the agent touches the current target it is rewarded, while it falls off the plain, it gets no reward and the episode is ended. And when the count reaches 6, the episode is also ended.



```csharp
public override void OnActionReceived(float[] vectorAction)
{
    Vector3 controlSignal = Vector3.zero;
    controlSignal.x = vectorAction[0];
    controlSignal.z = vectorAction[1];
    rBody.AddForce(controlSignal * speed);

    float distanceToTarget = Vector3.Distance(this.transform.position, Targets[count].position);
    if(distanceToTarget<1.42f)
    {
        SetReward(count+1f);
        count++;

    }
    if (this.transform.position.y < 0)
    {
        SetReward(0f);
        EndEpisode();
    }
    if(count == 6)
    {
        EndEpisode();
    }
}
```

Figure 3-10 The OnActionReceived(float[] vectorAction) method demo_2

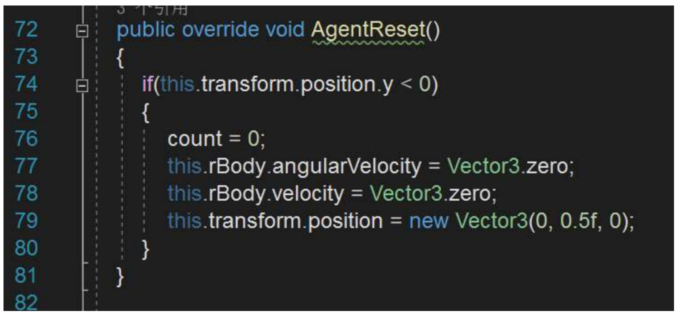
This is the Heuristic method, can be used in imitation learning and testing the control script. It takes two inputs from the keyboard, then returns the action array.



```csharp
3 个引用
public override float[] Heuristic()
{
    var action = new float[2];
    action[0] = Input.GetAxis("Horizontal");
    action[1] = Input.GetAxis("Vertical");
    return action;
}
```

Figure 3-10 The Heuristic() method demo_2

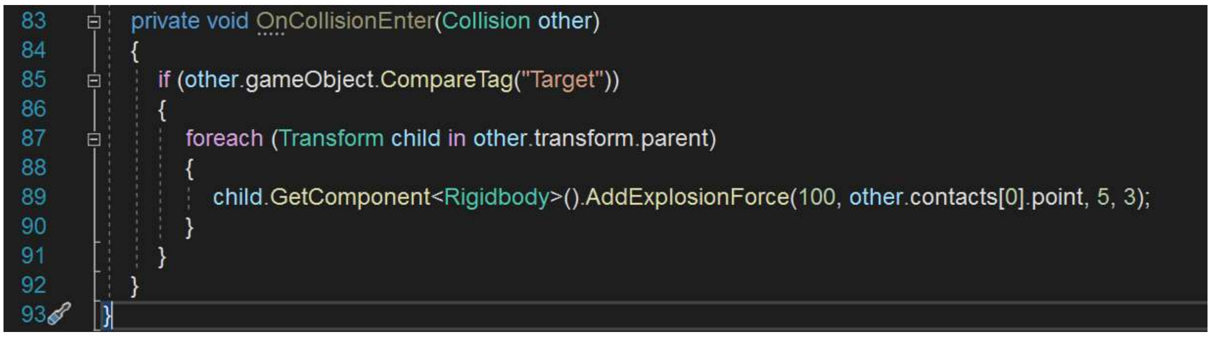
The AgentReset() method, reset the player if the episode ends.



```csharp
public override void AgentReset()
{
    if(this.transform.position.y < 0)
    {
        count = 0;
        this.rBody.angularVelocity = Vector3.zero;
        this.rBody.velocity = Vector3.zero;
        this.transform.position = new Vector3(0, 0.5f, 0);
    }
}
```

Figure 3-11 The AgentReset() method demo_2

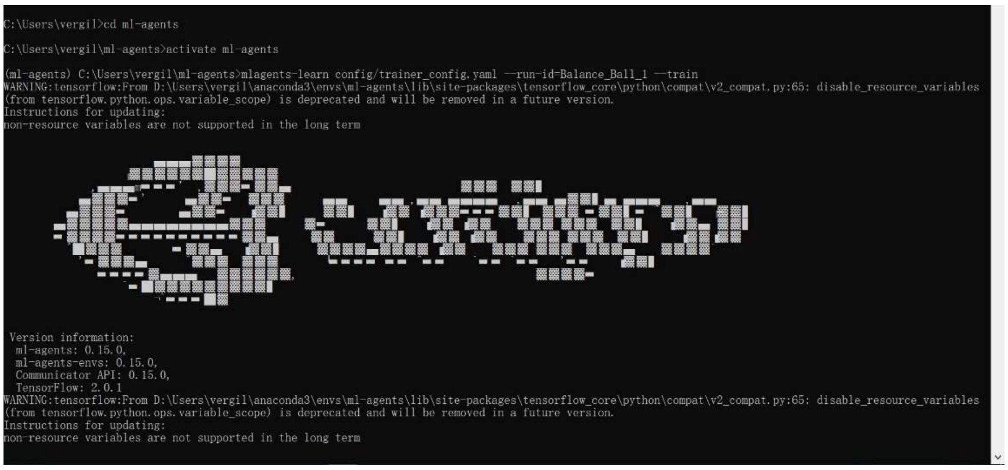
This OnCollisionEnter (Collision other) method makes sure that when the player touches the cannon, it will be launched out to the wall. By identifying the tag of the Collision (Target in this case) the Collision Object's subObjects will be added an explosion force centered at the contact point.



```
private void OnCollisionEnter(Collision other)
{
    if (other.gameObject.CompareTag("Target"))
    {
        foreach (Transform child in other.transform.parent)
        {
            child.GetComponent<Rigidbody>().AddExplosionForce(100, other.contacts[0].point, 5, 3);
        }
    }
}
}
```

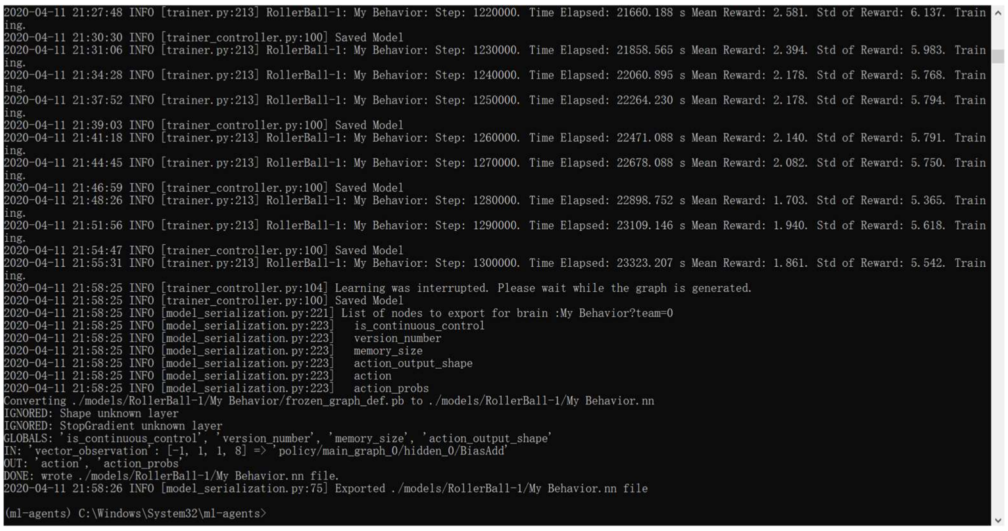
Figure 3-12 The OnCollisionEnter(Collision other) method demo_2\

The following two figures are the screenshot of training the AI agent.



```
C:\Users\vergil>cd ml-agents

C:\Users\vergil\ml-agents>activate ml-agents

(ml-agents) C:\Users\vergil\ml-agents>mlagents-learn config/trainer_config.yaml --run-id=Balance_Ball_1 --train
WARNING:tensorflow:From D:\Users\vergil\anaconda3\envs\ml-agents\lib\site-packages\tensorflow_core\python\compat\v2_compat.py:65: disable_resource_variables
(from tensorflow.python.ops.variable_scope) is deprecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term

 Version information:
  ml-agents: 0.15.0,
  ml-agents-envs: 0.15.0,
  Communicator API: 0.15.0,
  TensorFlow: 2.0.1
WARNING:tensorflow:From D:\Users\vergil\anaconda3\envs\ml-agents\lib\site-packages\tensorflow_core\python\compat\v2_compat.py:65: disable_resource_variables
(from tensorflow.python.ops.variable_scope) is deprecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

Figure 3-13



```
2020-04-11 21:27:48 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1220000. Time Elapsed: 21660.188 s Mean Reward: 2.581. Std of Reward: 6.137. Train
ing.
2020-04-11 21:30:30 INFO [trainer_controller.py:100] Saved Model
2020-04-11 21:31:06 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1230000. Time Elapsed: 21858.565 s Mean Reward: 2.394. Std of Reward: 5.983. Train
ing.
2020-04-11 21:34:28 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1240000. Time Elapsed: 22060.895 s Mean Reward: 2.178. Std of Reward: 5.768. Train
ing.
2020-04-11 21:37:52 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1250000. Time Elapsed: 22264.230 s Mean Reward: 2.178. Std of Reward: 5.794. Train
ing.
2020-04-11 21:39:03 INFO [trainer_controller.py:100] Saved Model
2020-04-11 21:41:18 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1260000. Time Elapsed: 22471.088 s Mean Reward: 2.140. Std of Reward: 5.791. Train
ing.
2020-04-11 21:44:45 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1270000. Time Elapsed: 22678.088 s Mean Reward: 2.082. Std of Reward: 5.750. Train
ing.
2020-04-11 21:46:59 INFO [trainer_controller.py:100] Saved Model
2020-04-11 21:48:26 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1280000. Time Elapsed: 22898.752 s Mean Reward: 1.703. Std of Reward: 5.365. Train
ing.
2020-04-11 21:51:56 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1290000. Time Elapsed: 23109.146 s Mean Reward: 1.940. Std of Reward: 5.618. Train
ing.
2020-04-11 21:54:47 INFO [trainer_controller.py:100] Saved Model
2020-04-11 21:55:31 INFO [trainer.py:213] RollerBall-1: My Behavior: Step: 1300000. Time Elapsed: 23323.207 s Mean Reward: 1.861. Std of Reward: 5.542. Train
ing.
2020-04-11 21:58:25 INFO [trainer_controller.py:104] Learning was interrupted. Please wait while the graph is generated.
2020-04-11 21:58:25 INFO [trainer_controller.py:100] Saved Model
2020-04-11 21:58:25 INFO [model_serialization.py:221] List of nodes to export for brain :My Behavior?team=0
2020-04-11 21:58:25 INFO [model_serialization.py:223]   is_continuous_control
2020-04-11 21:58:25 INFO [model_serialization.py:223]   version_number
2020-04-11 21:58:25 INFO [model_serialization.py:223]   memory_size
2020-04-11 21:58:25 INFO [model_serialization.py:223]   action_output_shape
2020-04-11 21:58:25 INFO [model_serialization.py:223]   action
2020-04-11 21:58:25 INFO [model_serialization.py:223]   action_probs
Converting ./models/RollerBall-1/My Behavior/frozen_graph_def.pb to ./models/RollerBall-1/My Behavior.nn
IGNORED: Shape unknown layer
IGNORED: StopGradient unknown layer
GLOBALS: 'is_continuous_control', 'version_number', 'memory_size', 'action_output_shape'
IN: 'vector_observation': [-1, 1, 1, 8] => 'policy/main_graph_0/hidden_0/BiasAdd'
OUT: 'action', 'action_probs'
DONE: wrote ./models/RollerBall-1/My Behavior.nn file.
2020-04-11 21:58:26 INFO [model_serialization.py:75] Exported ./models/RollerBall-1/My Behavior.nn file

(ml-agents) C:\Windows\System32\ml-agents>
```

Figure 3-14

4 Experiments

4.1 Description

Based on the game demos, over 20 testers were invited to finish the questionnaire. Together with the questionnaire, two sets of videos were sent to the testers as well, the questionnaire was designed to ask the testers which of the player in the videos are AI players or there was no AI player at all. Their gender, game experience was asked for further analysis. See the questionnaire and the screenshot of the two set videos in appendix I and II.

4.2 Results

35 questionnaires were sent and received 32.

	1	2: D	3	4	5: A	6	7
Tester_1	A	C	B	D	D	A	In the first set of videos, the assumed AI only moved along a fixed route. For the second case, it was a shot in the dark.
Tester_2	B	ABCD	C	C	AB CD	D	Based on my game experience, the excellence of those players means that maybe they are all AI players.
Tester_3	A	E	C	D	E	B	AI should have some regular pattern, but the players in both videos did not have.
Tester_4	A	D	B	E	C	B	In the first set of videos, the yellow, blue and white spent some time to respond, while the black ball did not need any response time and could always pick the shortest route. In the second set of videos, the yellow and black ball seemed to know the shortest path, and the blue ball was under control. However, the red ball is always moving along a straight line, so the red ball is AI
Tester_5	B	D	B	B	E	B	All players do not move along a perfect line
Tester_6	A	D	A	C	D	B	In the first set of videos, the black ball did not need any responding time. In the second set of videos, only the assumed AI player blew off the wall.
Tester_7	A	A	B	C	D	B	In the first set of videos, the AI moved smoother than the others. In the second set of videos, it was the most stable AI, and it blows off all the bricks

Tester_8	B	D	B	B	D	B	1)The first game: the black one reacted much faster; all other balls swing largely when they were turning around. 2) The second game: the black ball moved more straightly than others. And it hit the center of the brick wall. The impact force was relatively bigger.
Tester_9	A	BC	A	A	C	A	Did not understand the first game, the second game the red one hit the target
Tester_10	B	E	B	A	E	B	Did not understand
Tester_11	A	D	A	E	E	A	Based on the movement pattern
Tester_12	B	B	B	C	A	B	Based on the movement pattern
Tester_13	B	BD	B	B	E	B	Based on the movement pattern and smoothness accuracy
Tester_14	A	C	C	C	E	B	AI behaved more simply, and human was more complex.
Tester_15	B	CD	B	D	E	B	The white ball was fast, accurate so it was AI. The yellow ball moved too randomly and too slow. The black ball was the same as the white ball. The blue ball moves too randomly. The second case: all the balls are too random, so it is human.
Tester_16	B	A	A	E	A	C	1. The player did not know how to turn around sharply. 2.The yellow ball nearly fell off the plain.
Tester_17	A	BD	B	E	CD	B	The pattern
Tester_18	A	B	B	A	B	B	
Tester_19	B	C	C	C	A	B	
Tester_20	A	C	A	E	A	B	
Tester_21	B	C	B	B	CD	A	The red ball could modify the angle and direction of moving to adjust its pattern.
Tester_22	B	D	B	D	B	A	The AI should be more accurate than human.
Tester_23	B	CD	A	C	AC	B	AI should be more professional than a human player, it should be faster and have shorter responsive time
Tester_24	B	C	B	A	A	A	AI should be faster than a human.
Tester_25	A	AD	A	D	E	A	I was sure that in the first set of videos there were one or two AI players. But for the second set of the video, I think there is no AI player because every player seemed flexible while AI players should be clumsy.
Tester_26	A	BD	B	D	BC	A	In the first set of videos, the AI players should change the direction sharply as the target randomly was generated.
Tester_27	A	D	B	C	B	D	The three players in the first set of videos had some responsive time. But the AI player moved more smoothly. And I assumed that the control of AI should be better than humans, while humans reacted faster than AI. Based on repetitive watching the video, I think B is the right answer for the 5th question.
Tester_28	B	C	B	C	D	B	For chasing the target, if it was I who played the game, I will directly rush into the target ball, but some players were just turning around, which I think is hard for humans.

							In the second set of videos, the black ball moves very smoothly, but the red ball and the yellow ball's movement were strange.
Tester_29	D	E	B	A	C	B	The player I chose in 1st set of videos behaved too passively. In 2nd set of videos, I found the red ball moved very quickly and flexibly.
Tester_30	A	D	B	D	A	B	
Tester_31	D	E	B	B	C	A	Every video has a red ball. So it must be the AI
Tester_32	B	D	B	A	D	B	The pattern of the black ball in demo-1 is different from other balls. And the black ball is more professional and stable than others, so in the second set of videos, it is the AI

Table 4-1 The result of questionnaire feedback

4.3 Analysis

4.3.1 Basic information

After analysis, among the 32 returns of questionnaire, except 7 invalid answers (Tester_9, Tester_10, Tester_18, Tester_19, Tester_20, Tester_30, Tester_31) which have no explanation of their choices or just cannot understand the test, 25 valid answers were left.

For these testers who returns valid feedbacks, 11 of them were male (44%), 13 of them were female(52%).

5 of them have played a similar game (20%), while 18 of them(72%) have played electronic games for more than 1 year.

Testers who recognized the AI in the first demo were Tester_2, Tester_4, Tester_5, Tester_6, Tester_8, Tester_11, Tester_13, Tester_15, Tester_17, Tester_22, Tester_23, Tester_25, Tester_26, Tester_27, Tester_32. (15 people)

Testers who recognized the AI in the second demo were Tester_2, Tester_12, Tester_16, Tester_23, Tester_24.

Among the results, roughly, the recognized rate of Demo_1 is 15/25 = 60%. The recognized rate of Demo_2 is 5/25 =20%. The tester who recognized both AI players in two demos were Tester_2 and Tester_19. The first demo's recognized rate is higher than the hypothesis, while the demo_2's is the same as the hypothesis.

Though 5 testers found the AI in demo_2 but only three of them give out the very correct answer ‘A’, as the other two of them gave a multiple answers including ‘A’. And for demo_1, only 8 of the 15 testers accurately recognized only ‘D’ was the AI player. The rate of recognition is fine to this perspective, for the ‘accurate recognizing rate’ is 8/25 (32%) for demo_1 and 3/25 (12%) for demo_2.

To draw the conclusion, some further analysis is necessary.

4.3.2 Arguments of valid feedback

And arguments why they drew their conclusions is in Table 4-2 and are separately shown in fig 4-1.

Reason	Demo_1	Demo_2
Fixed route of the player	T1,T3,T11,T12,T13,T15,T17,T32(8)	T3,T11,T12,T13,T15,T17,T28(7)
Good control of the player	T2,T5,T6,T7,T8,T13,T14,T16,T21,T22,T26,T28(12)	T2,T4,T5,T6,T7,T8,T13,T14,T16,T21,T22,T25,T27,T29,T32(15)
Responding time of the player	T4,T6,T23,T24,T27(5)	T23,T24(2)
By intuition	T25,T29(2)	T1(1)

Table 4-2

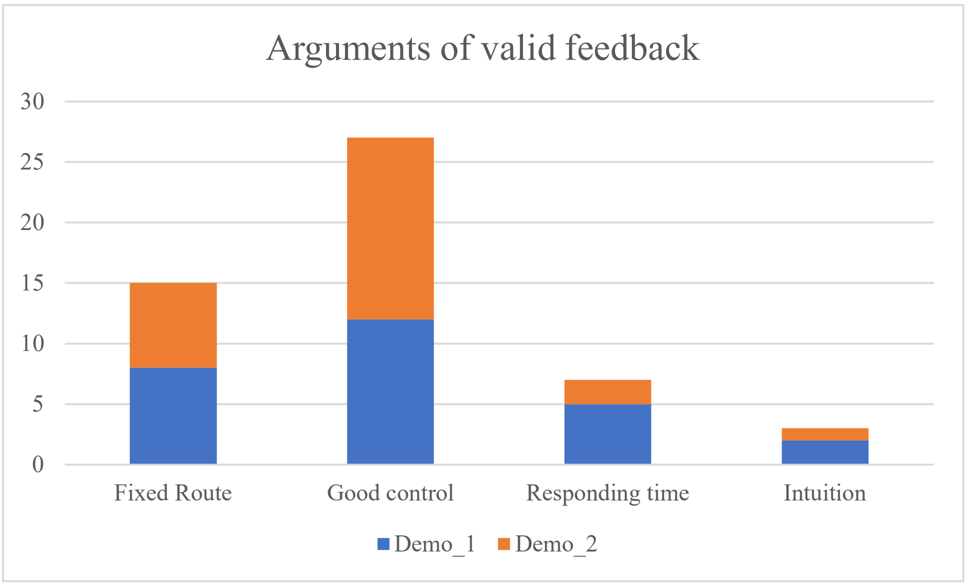


Fig. 4-1 Arguments of valid feedback

According to the diagrams, the reason why the tester drew such conclusions (recognize the AI or not) are very similar in each demo. Nearly half of them thought that good control is the basic characteristic of an AI player. While the second highest opinion is that an AI player should move based on some fixed route. The third reason is the responding time, whereas it is harder to recognize an AI player in a relatively complex environment(demo_2). While some of them even get their answer out of intuition, it makes sense to some degree, as it leads to some judgment based on sub-consciousness.

4.3.3 Arguments of testers who successfully recognized AI players

Reason	Demo_1	Demo_2
Fixed route of the player	T11,T13,T15,T17,T32 (5)	T12(1)
Good control of the player	T2,T5,T8,T22,T26 (5)	T2,T16(2)
Responding time of the player	T4,T6,T23,T27 (4)	T23,T24(2)
By intuition	T25 (1)	

Table 4-3

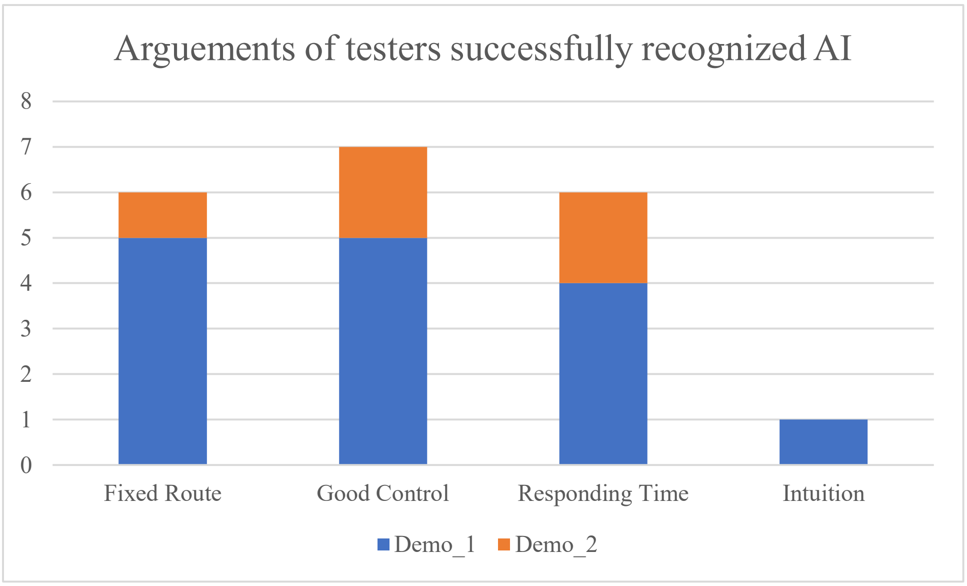


Fig. 4-2 Arguments of testers (successfully recognize)

In the successful recognized cases, the ‘good control’ and ‘responding time’ are still an important character in testers’ minds, while only intuition seemed not enough for recognizing AI players in a relatively complex environment. And fixed routes can hardly be the judgment criteria of an AI player in a relatively complex environment.

4.3.4 The statistic of game experience of the testers who gave the right answer

According to figure 4-3 and figure 4-4, most of the testers who recognized the AI player have more than one year game experience, whereas, in both demo_1 and demo_2, two testers does not have any game experience but still gave out the right answer. They are shown in table 4-4:

AI should be faster than human.	Tester_24
The pattern of the black ball in demo-1 is different from other balls. And the black ball is more professional and stable than others, so in the second video, it is the AI	Tester_32

Table 4-4

Tester 24 found the AI in demo_2 while tester 32 found the AI in demo_1. Their conclusions were drawn because of some stereotype of AI, which is 'AI should be faster and more professional than human player'. Though how professional the AI performs is decided by the training time, in this case, together with the diagrams(as the game experience increase, the recognizing rate did not increase as well) can indicate the recognition rate is not always related to the game experience, testers seem to draw their conclusion based on their intuition or some reason they thought made sense.

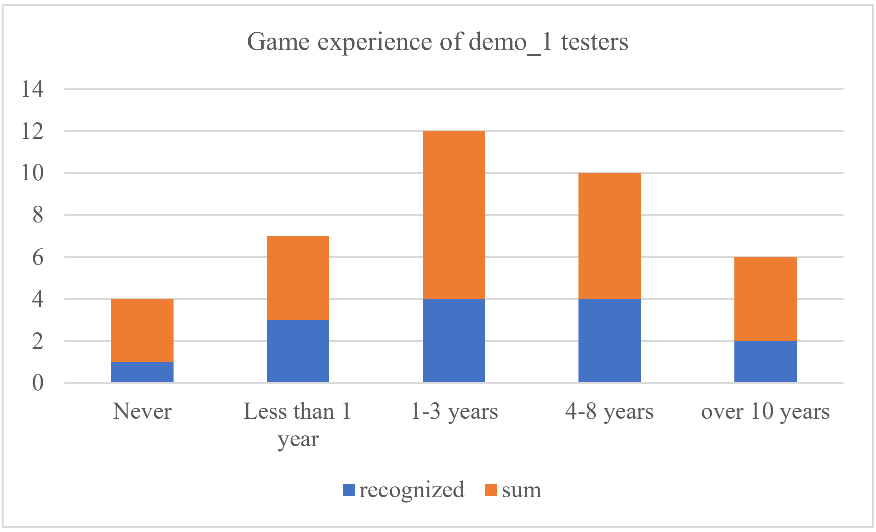


Fig. 4-3 The gaming year range of demo_1 tester (successfully recognized)

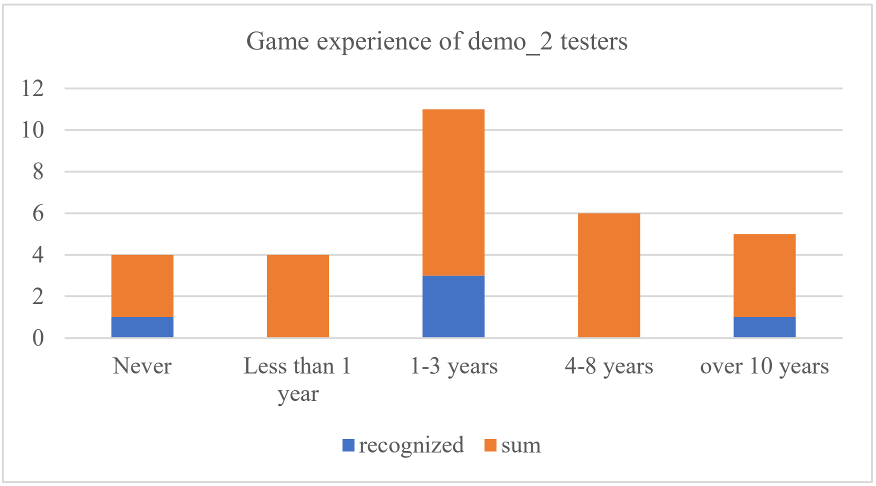


Fig. 4-4 The gaming year range of demo_2 tester (successfully recognized)

5 Conclusion and future work

5.1 Conclusion

For the demo_1 the recognizing rate is 20% higher than expected 40%, while in demo_2 only 20%, just fit the hypothesis. But only 32% testers of demo_1 and 12% testers of demo_2 recognized the AI player. The game experience is not related to the accuracy of recognition in this case. And the responding time and good control are two main aspects where testers believe they can recognize the AI players.

5.2 Future work

Based on the research results of this project, I assume that the difference of recognition rate is caused by the different complexity of the environment. Also, I think the interaction between player and AI will also influence the recognition rate. I am planning to build a highly immersed game where the tester can play with their opponent, and add in an AI player at any random time, then ask the tester can he recognize that the player was substituted by an AI player. The hypothesis for this can be the recognition rate should be much lower than the cases in this project, and it would be inversely proportional to the complexity of the game, the time given to the tester to react and how deep they are into the game environment.

References

腾讯游戏学院. (2019). 游戏 AI: AI 的游戏还是游戏的未来. *https://www.gcores.com/articles/113763#nopop_2iqn5*

Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M.&Lange, D. (2018). Unity: A General Platform for Intelligent Agents. *arXiv preprint arXiv:1809.02627.* https://github.com/Unity-Technologies/ml-agents.

John Schulman, Oleg Klimov, Filip Wolski, Prafulla Dhariwal&Alec Radford. (2017). Proximal Policy Optimization. *https://openai.com/blog/openai-baselines-ppo/*

Karangiya, H. C. M. (2019). "Training an Agent to Play a Game using Reinforcement Learning."

Livingstone, D. (2006). Turing's test and believable AI in games. Computers in Entertainment (CIE), 4(1), 6-es.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Ostrovski, G. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

Renz, J., Miikkulainen, R., Sturtevant, N. R., & Winands, M. H. (2016). Guest Editorial: Physics-Based Simulation Games. IEEE Transactions on Computational Intelligence and AI in Games, 8(2), 101-103.

Yang, J., Liu, L., Zhang, Q., & Liu, C. (2019). Research on Autonomous Navigation Control of Unmanned Ship Based on Unity3D. Paper presented at the 2019 5th International Conference on Control, Automation and Robotics (ICCAR).

Yannakakis, G. N. (2012). Game AI revisited. Proceedings of the 9th conference on Computing Frontiers.

Immersive Limit. (2019, July 31st). Unity ML Agents Tutorial - Penguins 🐧 (FULL WALKTHROUGH) [Video File]. Retrieved from *https://www.youtube.com/watch?v=axF_nHHchFQ&feature=youtu.be*

Unity. (2015, April 17th). Unity 5 - Roll a Ball [Video File].
Retrieved from: *https://learn.unity.com/project/roll-a-ball-tutorial?language=en*

LAIRD, J. E. AND DUCHI, J. C. 2000. Creating human-like synthetic characters with multiple skill levels: A case study using the Soar Quakebot. *In Proceedings of the AAAI 2000 Fall Symposium: Simulating Human Agents* (North Falmouth, MA, Nov. 3-5, 2000).

Appendix I

AI-in-games Recognition Questionnaire

游戏 AI 认知调查问卷

This questionnaire is open to all testers of the two game demos.

Please note: All responses will be treated as confidential.

本问卷对所有尝试过两个游戏小样的测试者开放

请放心:所有的回答都会进行保密处理

Introduction: Before filling in this questionnaire, you should have watched several videos of the two game demos, where the player can be either AI trained by pure reinforcement learning, AI trained by both reinforcement learning and imitation learning, raw human player , or veteran human player.

Please note that in the video you have seen, there are not always human players or AI players. In order to make sure that the AI players work fine, we need as much as the feedback from our testers. In the questionnaire, you will be asked about your game experience, which is player is AI or is there an AI player etc..

简介：在填写本问卷之前，你应该已经看过了这两个游戏小样的视频，其中的玩家可能是通过纯粹的加强学习训练出来的 AI 或者是通过加强学习和模仿学习共同训练出来的 AI， 新手人类玩家， 熟练的人类玩家。 请注意，在你看到的视频中，玩家有可能都是 AI 或者都是人类，人类和 AI 的数量是随机分配的。

为了确保我们的游戏 AI 工作良好，我们需要尽可能多的参与测试者的反馈。在本测试中，你可能会被问到关于你的游戏经验，以及哪个玩家是 AI 或者有没有 AI 的问题。

Q1: How do you describe your gender?

问题 1：你怎么描述你的性别

A. Male B. Female C. Transgender D. I do not want to say

A. 男 B. 女 C. 跨性别者 D. 我不想说

Q2: Which of the first set of videos, in your opinion, is the AI player?

(there may be more than one correct answer)

问题 2：在第一个系列的视频中，根据你的观点，哪个玩家是 AI？(多选)

A. Yellow B. Blue C. White D. Black E. There is no AI player at all

A. 黄色的 B. 蓝色的 C. 白色的 D. 黑色的 E. 根本就没有 AI 玩家

Q3: Did you ever play such games?

问题 3： 你曾经玩过类似的游戏吗？

A. Yes B. No C. I am not sure

A. 玩过 B. 没玩过 C. 我不确定

Q4: How long have you played electronic games?

问题 4：你玩电子游戏多久了？

A. Never B. less than 1 year C. 1-3 years D. 4-8 years E. over 10 years

A. 从来没玩过 B. 不到一年 C. 一到三年 D. 四到八年 E. 玩了十多年了

Q5: Which of the second set of videos, in your opinion, is the AI player? (there may be more than one correct answer)

问题 5： 在第二个系列视频中，你觉得哪个是 AI 玩家？(多选)

A. Yellow B. Blue C. Red D. Black E. There is no AI player at all

A. 黄色的 B. 蓝色的 C. 红色的 D. 黑色的 E.根本就没有 AI 玩家

Q6: How many times did you watch the videos to draw your conclusion?

问题 6： 你看了多少遍视频才得出的结论？

A. 1-2 times B. 3-5 times C. 6-9 times D. over 10 times

A. 1 到 2 遍 B. 3-5 遍 C. 6-9 遍 D. 10 多遍

Q7: How did you draw your conclusion? Could you please give out your reason in detail?

问题 7： 请问你怎么得出的结论，可以详细叙述一下你的理由吗？

Appendix II

More related materials of this project

The first set of testing videos.

The second set of testing videos.