University of Portsmouth

School of Creative Technologies

The effectiveness of learning artificial intelligence in video games.

By

Kieran Mark Grist

814853

Supervisor: Jahangir Uddin

CT6PRO

Project type: Combined

Abstract

The current technology and hardware available to developers has contributed to creating Artificial Intelligence which learns known as Neurological networks that can learn from their actions and input. This technology is currently not in main releases and this project aims to assess and compare a learning agent to an Artificial intelligence which uses standard industry techniques.

This project looks at and assesses the effectiveness of Neurological Network within video games. Starting off with looking at past projects and what experts say. The project creates a video game artefact which has two agents. One agent is using standard Artificial Intelligence techniques for video games and the other is a Learning Agent which uses Neurological networks to learn. The project then tests and analyses this project and finds that both techniques are viable for video games but learning agents are too early in development to be in modern games.

The Project then looks at what Learning agents can now be used for after the success of the project and discusses and reviews if they are effective for the video game industry. The project concludes that Learning agents could be the future of the industry if more developers use them in their video games.

Table of Contents

Table of Figures

Chapter 1: Introduction

Artificial intelligence as a method of controlling characters has been around since the days of Pong. Technology and hardware for the intelligence has been rapidly growing creating new techniques, software and games. Neurological networks are an imitation of our own brain and the network that produces it and it is a close attempt to simulate an intelligence that can learn from its actions.

AD-HOC Artificial intelligence is the traditional implementation of agents in video games. AD-HOC means for a purpose and all the systems created and implemented in each agent are there to increase the intelligence of the agent and aim to simulate real life.

Neurological networks or Learning agents is an upcoming technique used to imitate how human beings "learn" how to complete actions, this is done using numerous algorithms and techniques to teach the agent how to perform tasks without any programming needed for that task.

At a basic level both systems aim to produce realistic agents which can immerse the player in the world created. The most substantial difference is how they both complete actions, the standard ad-hoc agent will complete an action if its "brain" has told to by programmed techniques, such as a behaviour tree or goal driven behaviour it will only do actions by what has been programmed.

The Learning agent will complete an action based on what it has learnt to do in its training phase and will complete actions based on the rewards it has gotten. The learning agent has a brain telling it what to do and not what to do and actively learns how to be better with training. Learning agents rely more on training and time spent perfecting how they learn whereas the ad-hoc relies more on intensive programming and implementing techniques to make it realistic.

This project is being conducted to better understand learning agents, how they can be used and what applications they have in video games. Learning agents can improve the realism of video games and can push the boundaries of what is possible for video games.

This project creates the video game Survive. Resist. Evade. Extract (S.E.R.E) a United Kingdom Special forces selection program where one force (BLUFOR) must evade a hunting force (OPFOR) and get to an extraction point. BLUFOR must avoid OPFOR at all costs or they will be captured. Both agents will be OPFOR which is the hunting force and BLUFOR will be an agent set to go to the extraction point.

The project will compare the two agents in the video game, analysing how the two agents performed their actions and the current flaws they have. The project primary interest is with Learning agents and how effective they are currently.

Chapter 2: Literature Review

2.1 Introduction into Artificial Intelligence in video games

Artificial Intelligence is a collection of tools and algorithms that attempt to imitate intelligence Gordon, B. M. (2011). Artificial intelligence is used in video games to create non-player characters (NPCS) which interact with the world and player(s). In Real Time Strategy games such as Planetary Annihilation (Star Theory Games, 2015) the NPCs use Artificial intelligence to battle each other in a global battle for dominance, in these styles of games the player will control where the units go but battles are usually automated.

Video game, artificial intelligence uses two main techniques, goal-driven behaviour and behaviour trees. These intelligence algorithms are designed to imitate realistic patterns set out by the programmers. (Buckland, 2011)

In most video games artificial intelligence is meeting gamers standards with some recent modern games creating challenging AI for players to combat in their objectives in the game. Such games like

Heat signature use techniques like procedural generation and goal driven behaviour to create a dynamic and challenging game.

However, the AI can only adapt to a certain calibre of player level, and if players try new and untried tactics the AI may not be able to respond, often is the case that players find the AI weakness's and exploit that be it game mechanics or bugs within the intelligence of the NPC.

2.2 Introduction to Neural Networks in video games

Neural networks are the simulation of a human brain using artificial intelligence, these networks can essentially be trained and learn from their actions and can recognise patterns from data inputted into them and select certain strategies. They can also adapt more easily to player personalities and can model the player behaviour and experience. (Yannakakis & Togelius, 2018)

Neural Networks or machine learning is currently used in most fields of research to answer question which would commonly take humans years of work. Neural Networks however are relatively new to video games and hence have less research and techniques to implement them into video games.

This is largely due to the performance requirements of these artificial intelligence compared to AD-HOC techniques.

There is the issue of balancing a neurological network. Balancing issues can arise from an intelligence, new players could have a hard time adapting to a smart learning computer, players can easily find a way to abuse the training data. These are just some of the issues from a development standpoint (Boser, Sackinger, Bromley, LeCun, & Jackel, 1992)

Another reason is simply that AD-HOC AI is easier to implement, neurological networks require training and data to be built and that comes with its own issues (Yannakakis & Togelius, 2018). Simple games like Pac-man (Namco, 1980) would have little benefit for having a learning agent in the game, these style of games are not intended to be realistic or imitate real life and thus a learning agent which is more intended for realistic games would not be a good suit for this game. Another reason is the time and resources spent on a learning agent, for more simple games a state machine is smaller and easier to implement then a learning agent and the developers already know what to expect from a state machine and do not need to worry about the learning agent not training correctly.

However Learning agents do not always have to be realistic and simple games could benefit from an agent that learns, even though the time spent teaching the agent might increase game development time it could lead to more features and interaction from the agents and increase the challenge towards the players.

For the purpose of this study the effectiveness of neurological networks for small simple games is void this is due to the obvious ease that an AD-HOC AI can be created for these games within a much quicker timer then a neural network. The study is focusing more on larger games with more complex inputs as these are the games that most modern developers are creating and it would be a better test to put a learning AI in a game with multiple data inputs and unpredictable variables to truly test the system.

2.3 Neural Networks current capabilities within video games

Neurological networks do exist in video games. Forza Horizon 4 (Turn 10 Studios, Playground Games, 2018) uses them for its driver profiles to adapt dynamically to players techniques and create more dynamic AI for their game (Forza Support, 2019). Alpha star is a modded ai system for Star craft 2

(Blizzard Entertainment, 2010), Alpha star (DeepMind, 2019) is a Deep Mind AI that learns from its matches and training data and is currently battling the professionals of the game.

Neurological networks are very rare as of 2019 in the video game industry. With the only two examples being those above. However, their capabilities for games is evident. Machine learning AI has picked up games such as chess and are very effective competitors at the game (IBM, 1997). Using this knowledge Learning AIs on a base level should be effective at strategy games, specifically playing the commander of the forces. This is because they have all the information and learn from their mistakes often completing actions in a logical order and taking routes that players would not consider. Most strategy games have some computer opponents anyway which are adequate at the game so the systems should work with a learning AI behind them, as Alpha Star (DeepMind, 2019) proves within Star Craft 2 (Blizzard Entertainment, 2010).

Neurological networks are evidently capable of being the commander, which is one component of the game, however the true test of their effectiveness would be to look at agents. Agents must process information every frame and decide their actions using ad-hoc techniques which have a set outcome, I.E. Agent sees enemy, agent shoots enemy, enemy dies, agent carries on. Those steps have been programmed out for the ad-hoc AI, but a learning AI would have to learn how to do all that (Yannakakis & Togelius, 2018). In theory the most effective way to create the learning AI would have a hybrid of the two systems. The learning AI chooses what actions it conducts, and the AD-hoc programming will complete that action.

2.4 Main Differences between AI and Neural Networks

The biggest and most notable draw back of a neural network is the time and resources needed for it. Hypothetically it will take one day just to train it to walk around the map, whereas the AD-HOC has already got all those systems created for it. Simply put the ad-hoc ai is cheaper and more effective to have in a video game, the game can have more agents. Training the AI requires data and input as well as time to fix any learning issues. (Yannakakis & Togelius, 2018)

Continuing this learning AIS still need to be researched for video games, they are still a relatively new technology. AD-HOC systems already have the performance, research and industry standards. Learning AI is currently only in one game in a relatively small way and it was required for them to have a server farm to run that system. (Buckland, 2011)

However, the main advantage of neural networks is the realistic nature of them, while they require allot of training behind them, they are far more complex and capable of completing tasks that ad-hoc are not due to limitations. Neural networks can expand as much as the developers allow it to and in theory could be as competent as the players in the video games.

There could be a chance that these agents are used in multiplayers games to fill in the numbers that way lower player count does not impact the players enjoyment on the game. Learning agents would be a good fit for Massive Multiplayer Online (MMO) games as they rely on a consistent player rate. If a player connects to an empty server, they will not have much insensitive to carry on playing, the same applies to normal multiplayer games as well. Low player counts leads to people not wanting to join a server and play as the gameplay is suited towards a full server, most multiplayer games have the issue with retaining players especially older games. Most players do not want to be alone on a server as there is no interaction, in First Person Shooter (FPS) the game is reliant on a player count. #

A learning agent can learn how to speak and act like a player and have different personalities and voices to keep players interested in games even if the servers are not full with players, these agents interact with the world in a way which the player does not suspect that they are AI. The agents can improve games chances of being popular and staying alive for longer and give players a chance to create servers for friends and still fill in roles.

2.5 Current Issues with Artificial intelligence in modern games

Artificial intelligence in games can be complex beasts which can be hard to maintain and often have bugs, even triple A titles like outer wilds had bugs with the AI getting stuck or shooting at targets that the player wouldn't otherwise know existed. Most issues with the current AI systems are the AI can get stuck within doors, or walls with small gaps, which players can get through but the AI struggles to. The AI can also have supernatural ability to see the player, even when they would not be spotted by another player.

The biggest issue with Artificial Intelligence is within its ability to adapt for players mistakes or personalities. The players can do unknown actions and the AI would either not respond or bug out and get stuck in loops.

In video games like Arma 3 (Bohemia Interactive, 2013) the Artificial intelligence has a reputation for having god like eyesight and can see through objects. The AI also detects people from behind even when they are making no noise, this often makes stealth in the game difficult without modifications improving the AI.

AD-HOC also has a smaller issue with only being able to perform set tasks. If the AI needs to perform a new task or do something different it will need to be told to do that. This results in the AI only being as dynamic as it is programmed to be, while it can have responses to stimuli and actions upon that, if it hasn't been programmed to complete that task or respond to that set of inputs it won't do anything.

2.6 Industry Problems with Neural networks

Neural networks are rarely incorporated within the video game industry due to problems with the technology. The first and most substantial problem is resources needed to create a neural network currently there are multiple methods to creating a neural network (Yannakakis & Togelius, 2018), this will lead to unnecessary developer research into what method they need to use. Adding to this problem with resources is the time and data needed to teach the learning AI, hypothetically each node takes 500ms of a CPU and requires 1 day to train the data for that node, the more nodes added the bigger and more expensive the AI gets on the CPU. (Yannakakis & Togelius, 2018)

The second problem with the method is the current fact that video games can incorporate dynamic AI which can be realistic and immersive without the need for learning algorithms. Games like Heat Signature (Suspicious Developments, 2017) and Outer Worlds (Obsidian Entertainment, Virtuos, 2019) already have dynamic and realistic AI responses with little issues. Developers can create gold standard AI without the amount of time it takes to create the learning AI and with the added benefit of industry standards, documentation and resources on creating a better AI. AD-HOC AI within the industry is preferable as it is easier to implement and does not require as many resources to create it.

Even though there are processes like reinforced learning a minor issue with neural networks is not being able to guarantee what a learning AI profile does if it gets shot at there is no guaranteed way it will seek cover if its brain decides to run at the player and stab them. This can lead to issues with the AI, while this does create dynamic gameplay, such as unpredictable behaviour and often doing the wrong action at the wrong time.

Chapter 3: Methodology

3.1 Time Management

To manage the time the Kanban method was chosen. Kanban is a method of time management for just in time manufacturing. The work is put on a board which tracks tasks needed for the artefact and essay in the stages of the development cycle. Stages are not set in stone. This is effective for the project as a certain task may suddenly need to be developed or an idea was no longer viable.

As Kanban is less structured then other methodologies there are no deadlines, working to deadlines has always been more effective in development cycles. This avoids the pushing stuff tasks aside because they do not have a deadline and leaving it to last minute.

To manage time more effective tight week development cycle was chosen, each week it would be assed if a system has taken too long, if it has it gets pushed to the stretch goal region that way, core systems can be focused on. Appendix 5.

The artefact will need to be complete within 3 months to gather primary research, this requires having an altered Kanban system and run the project like an extended game jam having prototype systems instead of clean created code.

3.2 Analysing the artefact

To analyse the artefact two areas where chosen: Primary research of how a group of players interact with the game and primary research of how effective the learning AI is.

For the group it will be a focus group of 20 players, from any background possible, the focus group will complete a blind test on two games. A blind test was chosen as it is a more effective way of gathering data (Haverford College, 2015). The blind variable for these tests will be which game has the learning AI and which game has the AD-HOC. The players will get a chance to play both games for 30 minutes, the only difference will be the AI. After each session with the game the players will be given a survey to fill out asking them to quantitate and qualitative questions to gather data. After this it will be revealed as to which AI was the learning and they will have one final game with the learning ai, after this taking a final survey.

To measure how effective the development of a learning AI was two areas will be analysed. The performance of the learning AI and the development process of creating a learning AI. The performance will look at CPU, GPU, RAM AND DISK usage using unities profiler in test cases with both the learning AI and the AD-HOC AI seeing if the learning AI takes to many resources. The next is to look back at the development through GitHub pushes, project diary and notes as to how the process of creating a game for a learning AI went.

3.3 Software

Due to past knowledge and its ease of access to prototypes Unity (Unity Technologies, 2019) will be used for the project. The ad-hoc AI is going to be created using pre-created AI systems from either Unity (Unity Technologies, 2019) or online resources. Unity (Unity Technologies, 2019) is effective at creating the systems needed due to the nature of the component system it allows easy creation of the world and systems needed for the artefact and also allows the creation of more dynamic gameplay in a quicker nature then unreal. It is also easier to develop for and to debug in making ideal for a project of 3 months.

Chapter 4: Creating the AI Systems

4.1 S.E.R. E

S.E.R.E is a game based on the United Kingdom Special Forces selection process; the game aims to simulate this process of evasion. The learning AI will be trained to be the people searching for the players. The AI will get input from dogs, helicopters, thermal cameras and more to help them in their search. The learning AI will learn and use searching patterns to find the players within an area and their objective is to capture the players. The players must reach an extraction point without being caught by the AI. The world is procedurally generated which adds challenge both to the AI and players, the world is roughly 10km squared and can have various biomes and areas in it including jungle, woodland, rivers, desert, urban, fields and more. Appendix 1 Design, Appendix 2 Pitch Document

4.2 Designing systems for a game with a Neurological network

Designing gameplay mechanics and systems for a game intended to have a Neurological agent/Learning agent instead of an AD-HOC agent is a different process.

The first process is to look at what the AI needs to learn from otherwise known as the stimulus. This is what it is going to see and hear in a video game.

Unity's Machine Learning agents (Unity Technologies, 2020), use a sensor systems which goes of the tagging system within unit. The system allows developers to add and remove tags that the sensors are checking for. This is the visual stimulus for the game. Now knowing what the Learning Agent sensors are detecting we can now create 'tags' for objects we want the AI to be noticing.

In S.E.R.E the Learning agent needs to be able to detect the extraction point, soldier and any walls/obstacles. That is 3 tags which the sensors can look out for.

The second process it to look at how the Learning agent completes its actions. Unity's Machine Learning agents (Unity Technologies, 2020) uses a vector action system. This works by having a list of 'branches' these branches would have a size relating to the number of actions they can execute. An example being a branch size of 2 can be used for "on/off" actions, so if the number is 0 it executes the action. For S.E.R.E the Agent needs to be able to move, turn and shoot which is 3 separate branches.

The third process is the Learning itself, Unity's Machine Learning agents (Unity Technologies, 2020) commonly uses python training using a configuration for the trainer and a curriculum if the developers want the agent to face harder parameters. The trainer configuration is best left to default settings to reduce errors. With Unity's Machine Learning agents (Unity Technologies, 2020) in code you can add rewards both positive and negative to help teach the AI what is good and bad, this helps the Neurological Network learn. The curriculum is used to set a goal for the Agent to reach, when it reaches that goal it switches to the next set of parameters in the array. (Immersive Limit, 2020)

It is up to the developer how they handle these processes and it will be different for each project, there may even be more design consideration for the impact of having a learning agent such as player interaction, ensuring they are completing tasks and balance. All of which should be discussed in the prototype/design phase.

4.3 Building the AD HOC AI

The AD-HOC AI was implemented with a behaviour tree. This is due to behaviour tree node system which allows the developer to create nodes for systems like search and "plug and play" them within the AI. The AD-HOC AI at this stage is a basic behaviour tree, which can be improved with more development but as the focus of the project is Learning Agents the implementation is adequate at this stage.

The agent works by having a sphere detection system, the closets enemy in that sphere to the agent is sent to the "brain" which is a class that stores all the important information for the agent. The agent has 3 behaviour trees it can select from: Search, Hunt, Combat. The agents root node is a selector of these 3 trees. The selector runs the Behaviour tree selectors if they return true the Behaviour tree will run.

The Combat node checks if an enemy exists in the brain. If one does exist it will start shooting at that enemy. It does this by getting the agents position and creates a direction for the agent to shoot at, with more development the agent could predict where the enemy was going and shoot where it is going not where it is.

The hunt Node checks if the enemy did exist but the agent has lost eyes, if it has it will enter the hunt phase using the last known data from the combat phase to try and predict where the enemy just went.

The Search node executives if the previous two nodes have not run. It will create a random search pattern which the agent will follow to try and find the enemy. Once the agent has found the enemy the search pattern is destroyed, and the agent will generate a new one if they enter back into the search node.

After these 3 nodes have executed it will they enter the Move node which moves it along the waypoints that previous nodes have created. The move node will always be executed by the ad-hoc agents, so they are always entering a node.

4.4 Building the Learning AI

The Learning AI needs to learn 3 techniques, how to move, how to turn and how to shoot. It does this using a reward learning system meaning it will complete an action and if that was a good or bad action it will get a reward/points. Some actions are neutral and do not give rewards. (Immersive Limit, 2020)

The first step is to create the action system (Immersive Limit, 2020). The action system works by having numbers in a branch/array relating to an action. The Project had 3 branches, branch 1 is a "toggle state", meaning if its 0 or false the AI stays, if it is 1 or true the AI moves at full speed. Branch two is the turn state needing 3 inputs/ numbers: 0 do not turn, 1 turn in a negative rotation, 2 turn in a positive rotation. Branch 3 is again a toggle state 0 is shoot and 1 is do not shoot

Throughout the project there are rewards for the AI to learn what are good actions and bad. It gains a reward if the bullet hits an enemy, this reward is to Incentivise the agent into wanting to shoot at the enemy ensuring they attempt to kill targets. It gains another reward if it kills the enemy added to the bullet hit rewarded this should ensure the learning agent wants to kill the enemy force.

It loses rewards if an AI gets to the extraction point this is to try and get the agent to guard or stay near the extraction point. If the bullet hits the wall or floor to adds a negative reward, this makes sure the agent is not shooting at random and it is trying to shoot the enemy. The last negative reward is for the agent falling of the map this is just in case the agent finds a gap and to try and avoid that gap.

The above reward systems ensure that the AI fires at the enemy and tries to avoid shooting at walks, this makes the agent relatively accurate. As the agent was taught using infinite ammo there is room to add negative rewards for reloading so it tries to conserve ammo and even negative rewards for wasting all their ammo.

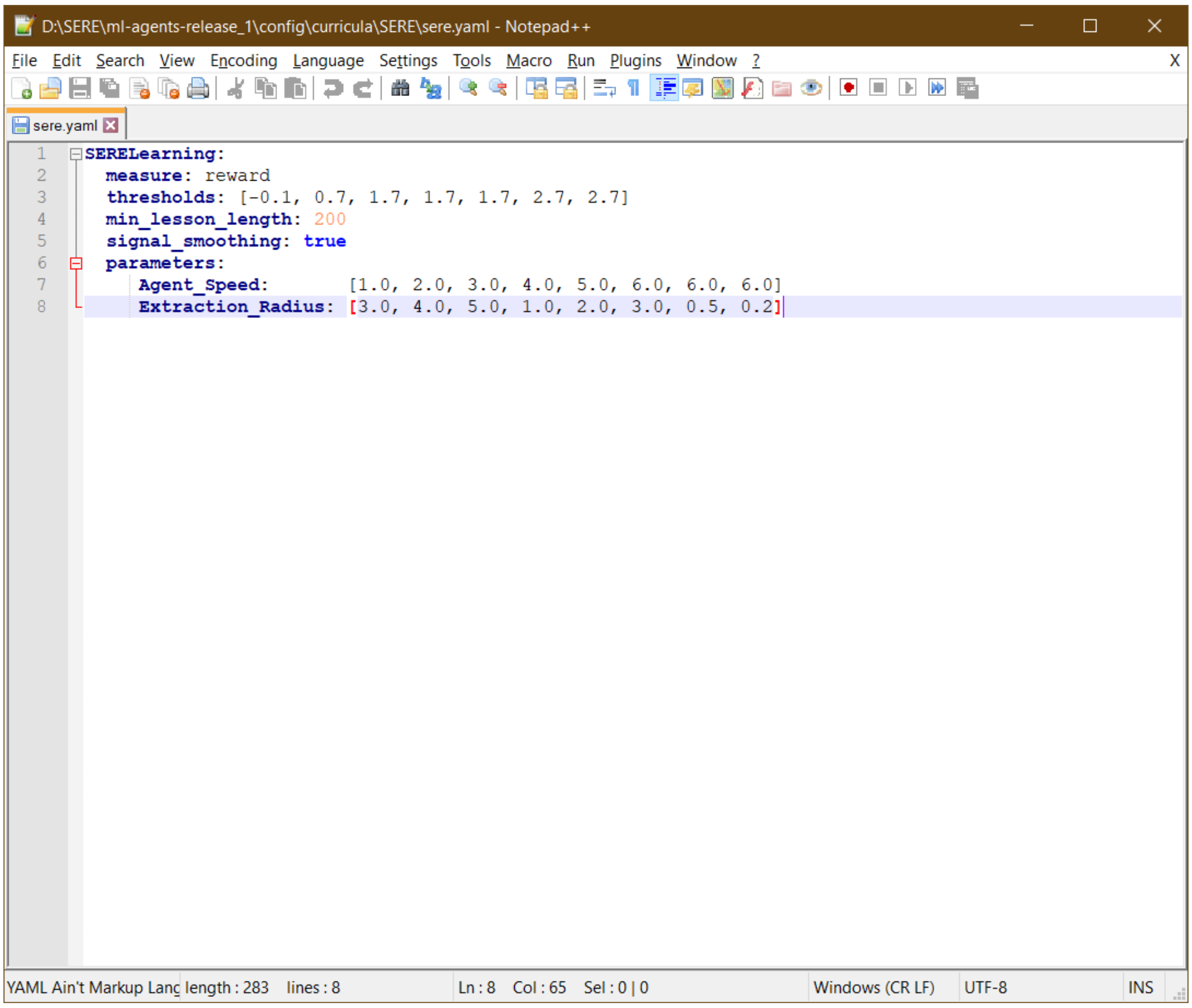
With the input system in the project there could be room to have weapon switching, reloading and even firing grenade launchers or throwing grenades.

The sensors system works by using ray casts to detect what it can see based on the settings the developers set. Within the project it has been set to detect the enemy, extraction point and the wall and a relatively realistic view distance and field of view.

4.5 Training the AI

To train the AI the project needs the Unity's Machine Learning latest stable release set up able to train with python (Unity Technologies, 2020) (Immersive Limit, 2020) (Siedler, 2020). Once this is set up the developers can run the mlagents-learn command which is the main command when training the brains for the Learning Agent. When you run this command, you can set the curriculum which allows the developers to set learning parameters for the curriculum.

The main setting is the threshold parameter, the agent has an average score and when it meets this threshold it changes the lesson to the next parameters. The image bellow shows the parameters used within the project when the agent reaches an average of -0.1 it will then switch the speed to 2 and radius to 4.


D:\SERE\ml-agents-release_1\config\curricula\SERE\sere.yaml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
sere.yaml
YAML Ain't Markup Lanc length : 283 lines : 8 Ln : 8 Col : 65 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

```yaml
SERELearning:
  measure: reward
  thresholds: [-0.1, 0.7, 1.7, 1.7, 1.7, 2.7, 2.7]
  min_lesson_length: 200
  signal_smoothing: true
  parameters:
    Agent_Speed:        [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 6.0, 6.0]
    Extraction_Radius: [3.0, 4.0, 5.0, 1.0, 2.0, 3.0, 0.5, 0.2]
```

Figure 1 - Learning Agent Curricula Settings

The developers found allot of issues with this curriculum method. Certain parameters would cause unity to freeze with no error in the console or log. Often it was a case of trial and error and trying to find the right settings without breaking unity. This freezing was frustrating and seemed to have no real fix or cause.

The reward system while effective did have its issues, brain 1 was punished too much for missing shots and it meant that the AI would no longer want to try and shoot the enemy. Prototype brains for brain 2 had the opposite issue where they were rewarded too much for shooting the enemy and they were still being inaccurate. With the training system it is finding the right balance of teaching the AI. This project and Penguin Machine Learning (Immersive Limit, 2020) both use the reward system which seems to give the best response during training, the balance of training in the reward system allows you to control what rewards you give and thus allows the creation of advanced teaching methods. Teaching learning agents can be done in several ways and the exact way of doing it will depend on projects, but the reward system will likely be the most common technique used as it is the easiest to implement.

The project created two brain, brain 1 which was an early prototype and brain 2 which is the release brain with the fully trained prototype. There is room to create more rooms with improvements and fix issues within the project but as a prototype the brains are satisfactory.

4.6 Testing the AD-HOC AI

To test the ad-hoc AI the developers need to make sure that the AI can do 2 things Search for the enemy and engage the enemy. These are the two major ad-hoc nodes that need to work in order to have a fair comparison between the two systems.

Searching the enemy can be tested by having the agent go around the map and shooting one enemy, this would be a successful search pattern and the ad-hoc behaviour tree can switch between the nodes. If it engages the enemy the combat node is also working so one test can show that the two main systems are working.

The test was completed by running a scene with the AD-HOC area on its own with nothing else in the scene to make sure that nothing was interfering it. The table below is the results from that test.

Test Number	What is being tested	Expected result	Result
1	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
2	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
3	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
4	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
5	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
6	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
7	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
8	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
9	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed
10	AI Searching and finding one or more enemies	Enemy found and shot at	Enemy found and killed

Figure 2 - Ad-hoc AI Testing table

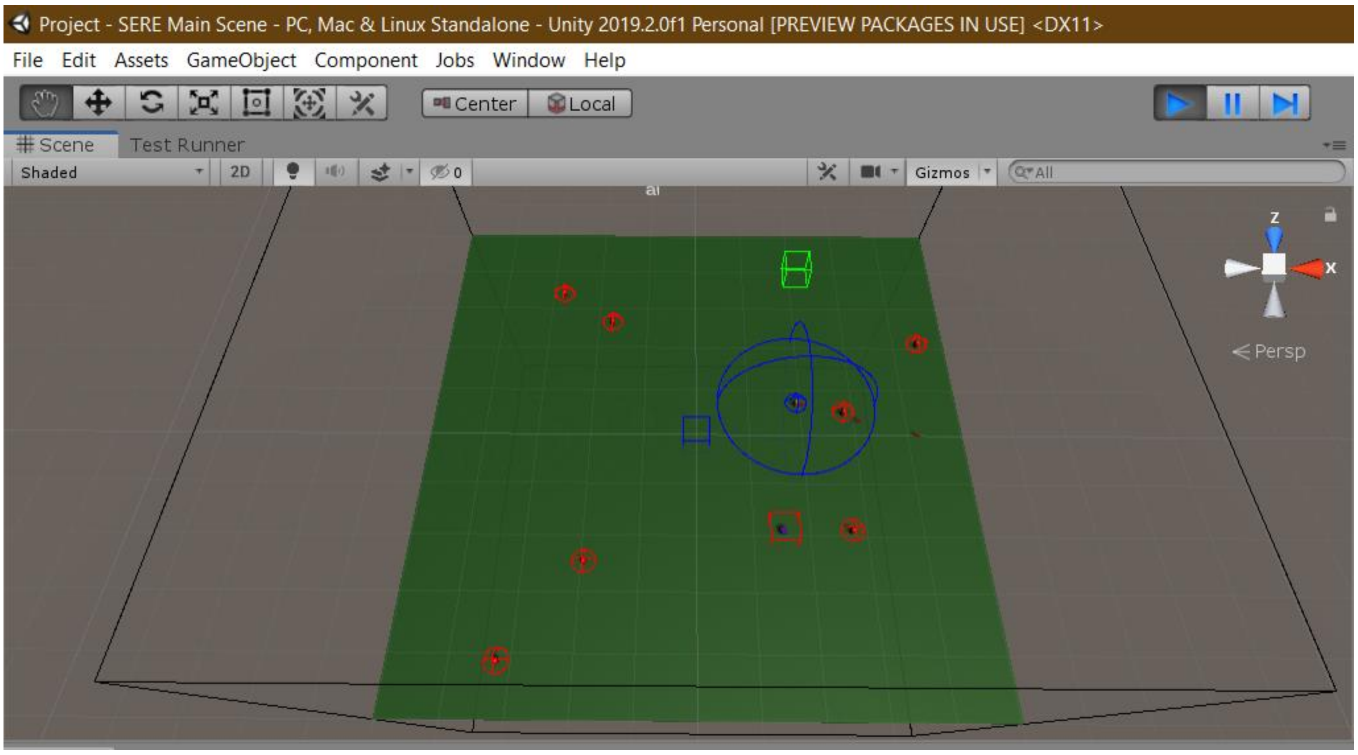

Project - SERE Main Scene - PC, Mac & Linux Standalone - Unity 2019.2.0f1 Personal [PREVIEW PACKAGES IN USE] <DX11>
File Edit Assets GameObject Component Jobs Window Help
Center
Local
Scene
Test Runner
Shaded
2D
0
Gizmos
All
ai
Z
X
Persp

Figure 3- Ad-hoc Testing image 1 – Shooting at target

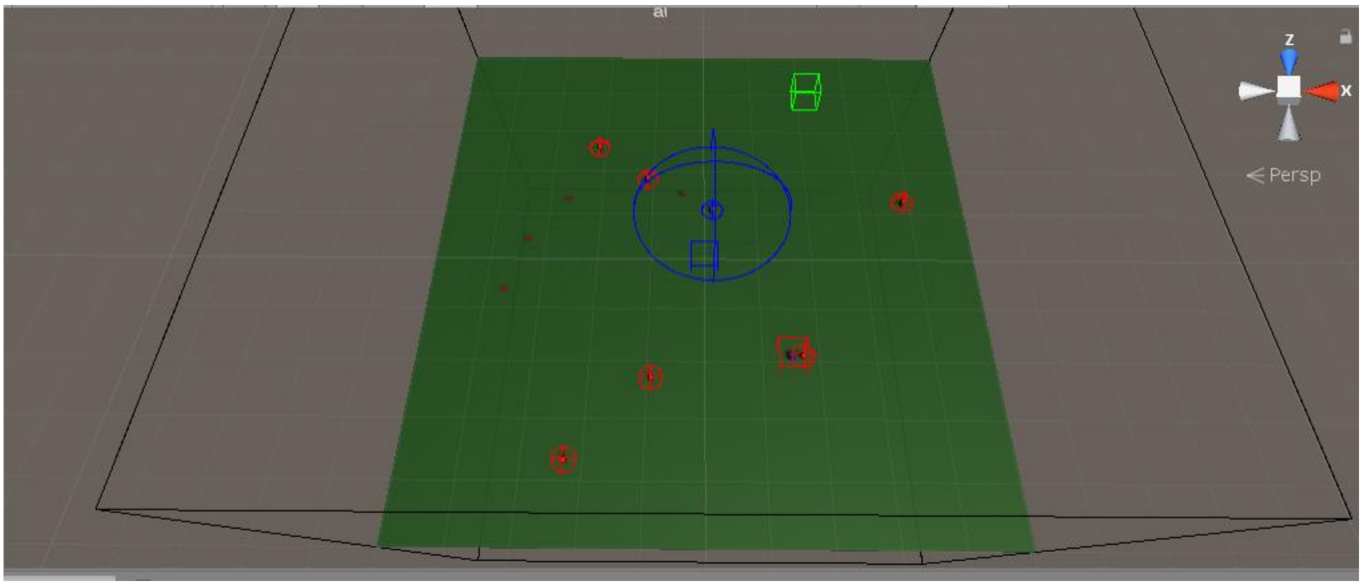


Figure 4 - Ad-hoc Testing Image 2 – Shooting at target

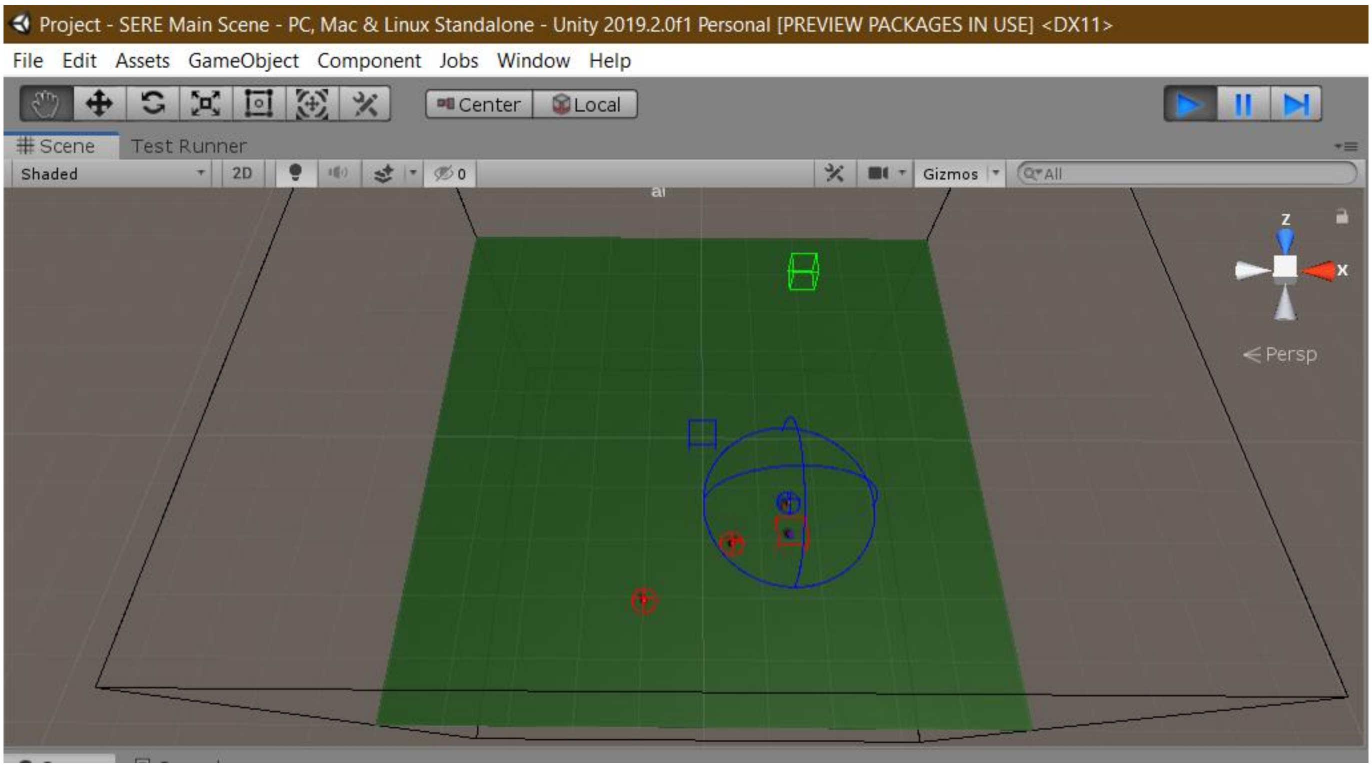

Project - SERE Main Scene - PC, Mac & Linux Standalone - Unity 2019.2.0f1 Personal [PREVIEW PACKAGES IN USE] <DX11>
File Edit Assets GameObject Component Jobs Window Help
Center
Local
Scene
Test Runner
Shaded
2D
Gizmos
All
ai
z
x
Persp

Figure 5 - AD-HOC Testing Image 3 - Patrolling

The results and picture above show that the AI is successfully finding and killing the enemy targets.

The last stage of testing is balance, as the AD-HOC ai does not use a perception system the developers had to ensure that the sphere radius was not to powerful and detecting everything, to achieve this the sight range was set to the AI could see people on their patrol path but not see people across the map. Other things to balance would be the AI shooting, the infinite ammo and the patrol pattern.

The test was a success and the AD-HOC AI has little to no issues, the biggest issue it has is the set patrol pattern, but this could be fixed with steering behaviours or other techniques. Other than this one issue the AD-HOC is effective at finding most of the enemy on the map before they escape.

4.7 Testing the Learning AI

To test the Learning AI there are 3 tests that need to be completed, the first test is to see if the agent moves around and finds the enemy. The second is shooting the enemy and killing them. The third test is to guard or stay near the extraction point. The agent does not have much reward for staying near the extraction point so the final test should fail as the brain has not be trained to stay near it.

Testing the agent moving around and finding the enemy is the hardest test to complete scientifically, as the agent is moving around in random directions so is it “searching” or “stumbling” across the enemy, this is hard to determine due to the random nature of learning agents. However apart of AI development is emergent behaviour where AIS can complete actions that programmers did not intend to be in the game. However, for this test a successful out-come will be the enemy being found.

Testing the agent killing the enemy is a simpler test, the programmers just need the agent to find the enemy and then shoot at them. If the enemy is killed by the agent the test has been a success, not killing the enemy is considered a failure as S.E.R.E's hunting force is meant to find and neutralise the enemy force.

The final test will be to evaluate how long the agent is staying near the extraction point. As the agent has no learning objectives from the extraction point it should have no reason to stay near the extraction point for any period, unless there is already enemy's nearby it

Test Number	Has the agent found the enemy?	Has the agent Killed the enemy?	Is the agent staying near the Extraction point?
1	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
2	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
3	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
4	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
5	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
6	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
7	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
8	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
9	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point
10	The agent found the enemy	The agent killed the enemy	The Agent is not staying near the extraction point

Figure 6 - Learning AI Testing Table

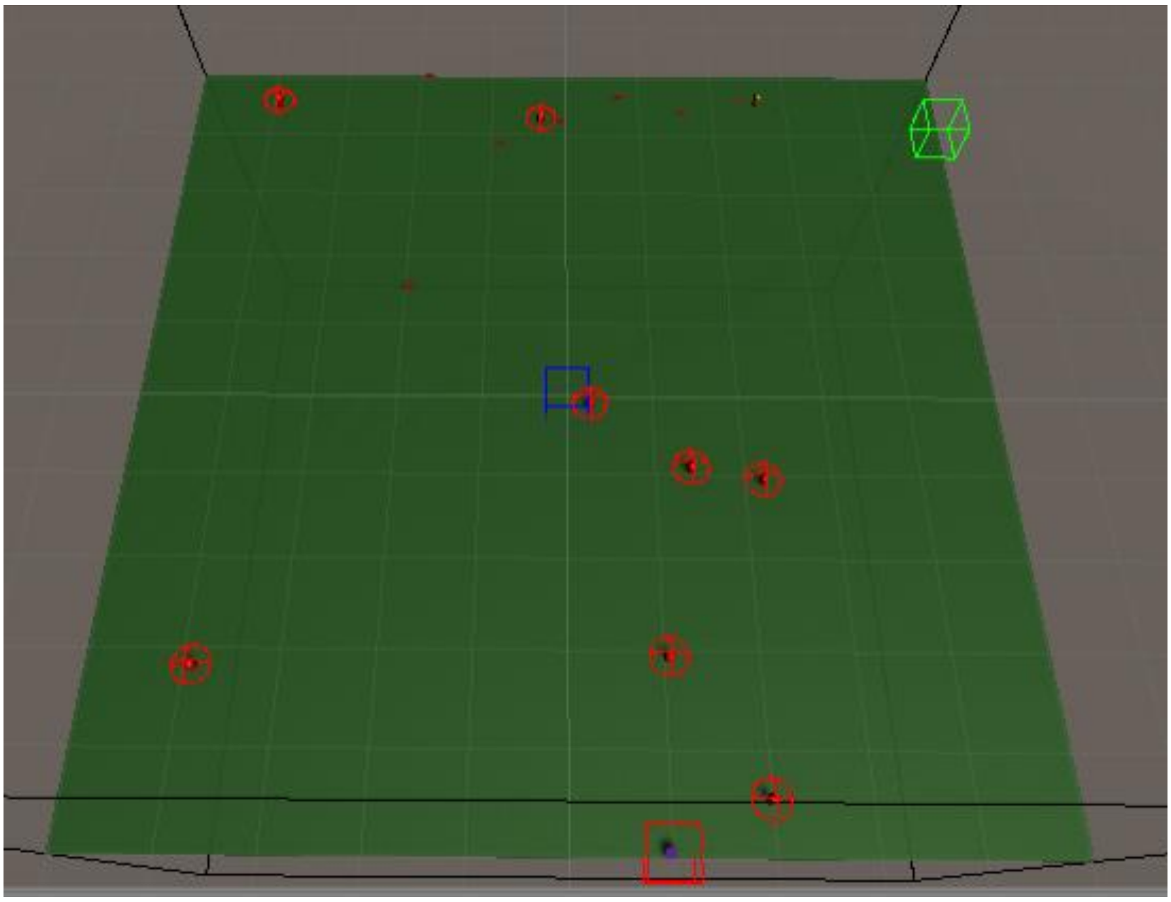


Figure 7 - Learning Agent shooting at targets

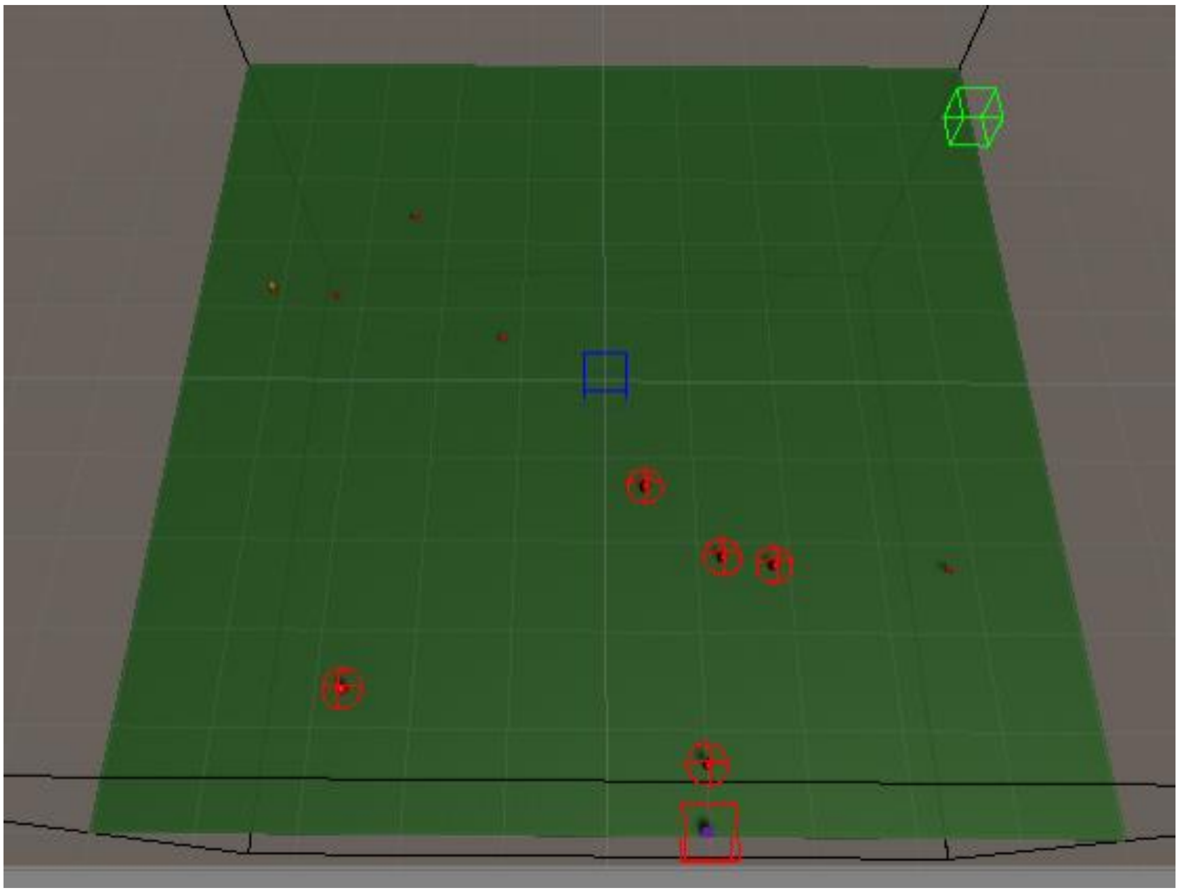


Figure 8 - Learning agent shooting at nothing

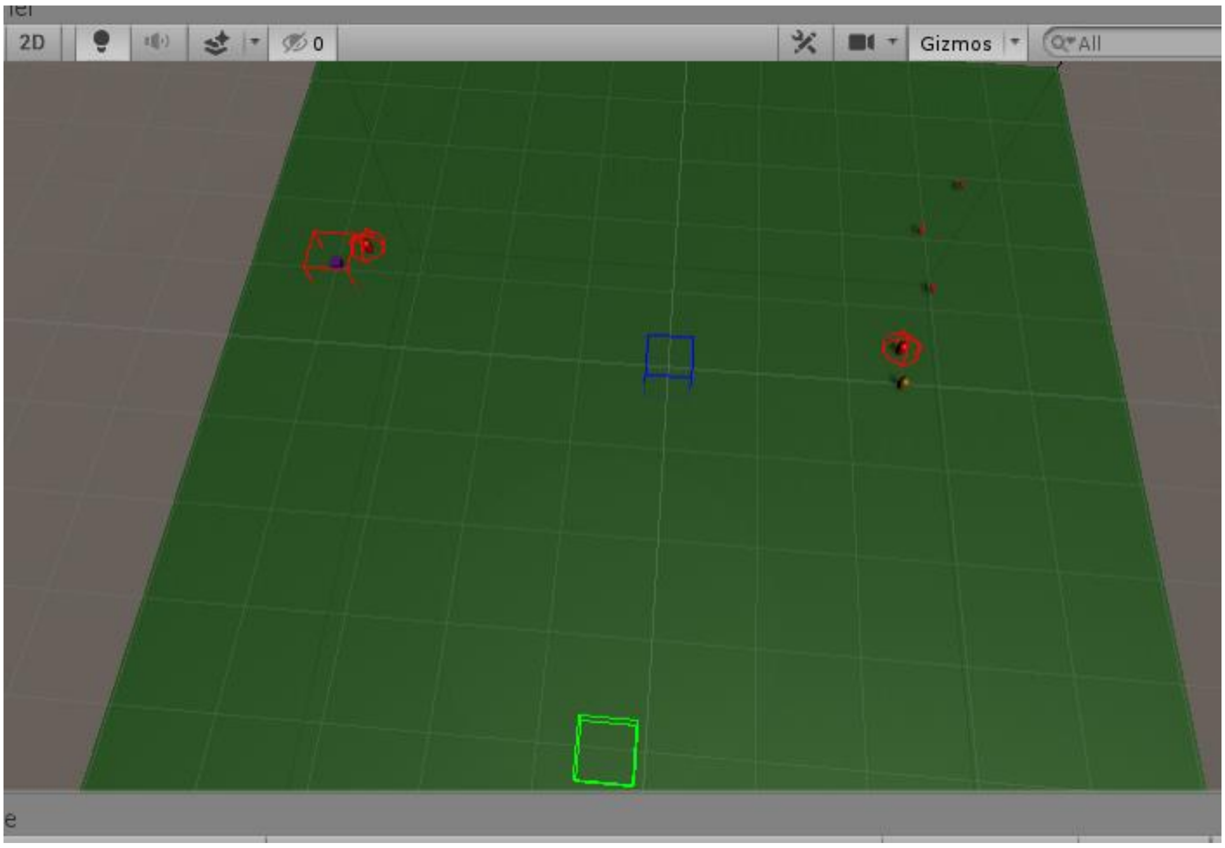


Figure 9 - Learning Agent shooting at a target

The table and images above show that the agents learning has successfully taught it how to shoot and kill the enemy evaders. The learning AI found most if not all enemies during the tests and often stayed away from the extraction point due to it not learning its importance.

Balancing a learning agent is all about the rewards for its actions. The balance is about what the reward and amount is for what action, if the agent got 10 points for hitting the enemy it could be a case that the agent is far to accurate and is laser point but the opposite could happen where too little of a reward can lead to missing all the enemy.

4.8 Initial analysis of the effectiveness

The Learning agent base goal was to be able to find an enemy game object and kill it. The agent has met this primary objective and thus is successful within its actions, however in a video game the agent needs to be able to do more than simply find and kill one object.

An initial analysis of the Learning agent shows it can find an enemy game object which is moving at different speeds towards a location which changes each round suggesting that the AI can adapt and move.

The Learning Agent is effective at finding the enemy however often it misses the agent and wastes allot of bullets trying to hit the enemy. This could be fixed by adding the magazine system back in or having a higher negative reward every time it misses however it could lead to the agent not wanting to shoot the enemy.

The Learning agent is currently very childlike it often just wanders around randomly often finding the enemy at different points along its walk and killing them, it does not make allot of mistakes but the training needs to be refined as the AI is not exact in its action. For a prototype, the Learning agent is satisfactory even though it could be improved.

The most complex tasks designed for this AI was the ability to shoot accurately at a moving target, as this was a prototype the learning AI would probably not accurately learn how to track and shoot at a fast moving target so the agents speed was kept slow.

Even with this the agent would often waste allot of bullets which shows it is not very effective at saving ammo and it takes the tactic of “spraying and praying”, which for this agent is not good as the game is designed to have limited ammo.

Chapter 5: Results & Findings

5.1 Ad-hoc AI

The ad-hoc AI for this project was intended to be basic and just a comparison for an early prototype of a learning agent. The ad-hoc agent was very effective with the search pattern and finding the enemy often killing many or all the enemy targets. The search pattern it created while random often found the enemy agents while patrolling.

The ad-hoc would find an enemy 10/10 in each of the tests completed, this is extremely efficient in its role to find an enemy target in certain area without receiving knowing where it was. However even though it found enemy, it would not hunt them if it lost eyes on them. This led to enemy escaping towards the extraction point meaning the ad-hoc agent failed to stop them all.

As the agent was on its own escapes where going to be expected when a 1 v 8 situation is created as the agent will likely not be able to find all the enemy in time to stop the extraction, some enemy would spawn close to the extraction point anyway meaning the agent would not have sufficient time to look for the enemy.

With more development the agent could have a more developed behaviour tree handling more states and scenarios including larger maps and guarding the extraction point.

Overall, the agent was successful at finding one or more enemy agents but would fail in chasing them if they moved out of its line of sight. The agent could handle multiple targets at one due to it only shooting at the closest target. If the agent were moving too fast the agent would miss the shots allot as there is no code for prediction of movement.

5.2 Learning agent

The learning agent for this project was intended to be a basic prototype of a Neural network trained to find and kill enemy targets on a map before they get to an extraction point. The Agent was surprisingly effective at doing this often killing all the enemy.

The learning agent would find the enemy 10/10 times in each test completed, in most of those tests the agent would kill all the enemy. It would attempt to chase enemies it lost eyes on but if it saw an enemy most of the time the enemy would be destroyed.

With the learning agent being alone escapes where to be expected as there was one agent going against eight enemies, the enemy's position was random and could be close to or next to the extraction point meaning escapes where to be expected.

With further development and learning the agent could become better at hunting the enemy and not being stuck in the corner of the map.

The Learning agents' biggest issue was if it got "stuck" in one corner of the map wandering around without finding an agent, it would often do this when there was one enemy left on the map it could not find.

Another issue of the agent was it sometimes looked and acted cartoonish and you could tell it was a learning agent by the way it moved, if a game was being made for more realistic nature this could throw the player out of the experience.

5.3 Comparing the two AI

A comparison of the two agents starts at acknowledging both agents have done the jobs they were programmed to complete. The first advantage of the learning agent is it could kill the enemy if they were moving, this is with the acknowledgement that it could waste allot of ammo doing so, the ad-hoc agent would often miss the enemy when they were shooting and this lead to escapes.

The second advantage of the Learning AI is it would not be stuck in a patrol pattern, while developers could change the pattern and use steering behaviours to fix this, the ad-hoc agent would not find enemies if they were not in its pattern, however the learning agent would as it was moving around the map trying to look for the enemy.

The first advantage of the Ad-hoc AI was it could better find one enemy on its own. As the agent was in a search pattern it was not reliant on random steering behaviours and would not get stuck in a corner.

The second advantage of the ad-hoc AI was it was better at not shooting blindly and randomly, it would only shoot when it was seeing an enemy and it would only shoot at that enemy. The Learning agent would shoot randomly at times even though its training told it not to.

Both agents could be further developed for the project however an initial analysis of the results and artefact show the ad-hoc agent would likely be better in the long run for this project. This does not mean the learning agent was ineffective, the learning agent is just to new of a technology with issues and balancing for a main release to be considering it.

5.4 How much did the AI learn

The learning agents, learning objectives where to search for an enemy target, shoot and kill the enemy target, not shoot at walls/nothing and to handle a moving target.

The learning agent learnt how to do most of these learning objectives, the finding and killing the agent was taught in one of the first prototypes and it would often achieve this learning objective in the outset.

The learning agent is still struggling to learn to shoot properly and not waste ammo, this is a learning objective which it has not met and it will often waste ammo bringing the rewards it has down to -100 or 200 most of the time due to how many bullets are wasted.

With this the objective of not shooting walls is currently a failure in its learning, even with the walls adding a negative reward each time they are hit, if the agent it's an enemy it gets a positive reward and wants to shoot more. So, the agent sort of gets stuck in a loop of hitting walls but sometimes hitting the enemy. It can be accurate but would sometimes waste ammo randomly.

The last objective with moving targets is currently a success as it is able to shoot a moving target relatively accurately, it has not been tested at higher speeds then the training so it may not be able to hit vehicle targets.

5.5 Application of Learning AI in video games

From the results and artefact of the project, the application of learning agents in video games can be vast and diverse. The main application would be for Real time strategy game as a commander since the agents can learn strategies and what is the right and wrong move. The agents can also be put in most modern game genres with application in single player games but also Player vs environment games.

Learning agents do not have to be a Non playable character, they could be the system behind making puzzles for players to challenge themselves or used in some sort of generation or creation system for the game, the agents could learn how to make 3D worlds for the player to explore or even start creating basic assets for video games based on images.

Translation with learning agents would also be a big application and help, with a learning agent learning and understanding most languages, if the agent could translate what someone was saying it could better help players communicate to each other as well as have real world applications.

Learning agents could be used for all of this but the main issue is training and data, for the learning agent created for the project it took a day to get it to a basic state for a prototype, the agent was only learning how to do 3 things where 2 where relatively basic for it to learn and act upon. With each feature/learning objective for the agent leads to more complex training and further development for the project.

Chapter 6: Discussion

From creating the Learning agent, the developers will create two additional projects that will be completed outside of this one, a Racing agent that learns how to be the fastest along real and procedurally generated racetrack. ATC agent which handles an airport with differing traffic and challenges. The purpose of these two future projects is to better analyse and assess the effectiveness of learning agents for different style of video games.

Learning agents is too early for modern games. During the testing phase the learning agent would often shoot at walls and miss the enemy. The learning agent was very early and almost childlike in its movements and actions as mentioned in the literature review learning agents are a new technology and the testing supports the analysis that they still need further testing and development to be a full game agent. The learning agent has issues and kinks which make it fine for a prototype of proof of concept however for a full release the learning agent would have to many issues for it to work correctly and improve the game.

From creating the artefact it was discovered that the teaching and training phase was allot easier than initially theorised, the resources needed to test and train an agent where not as high as first thought and the first working version of the agent was created in a day. Although the agent was a very simple in its learning the teaching took 2 hours on average which is much lower than suspected.

The artefact also discovered that learning agents are quicker to create in the sense you do not need to code the features instead program the reward system and how it will learn, this does reduce the time to create the agent but getting it to learn correctly would be an increased step in the development cycle.

As suspected in the literature review Ad-hoc agents have more techniques and are easier to implement currently making it a more viable agent at this point, with years of experience agents using ad-hoc can appear to be human like and make their own decisions without learning. Once a feature is created and has no bugs that is it with an ad-hoc agent whereas a learning agent new features could lead to more problems teaching the agent to deal with multiple things at one time. Ad-hoc is generally easier to create agents for and more developers have experience using them then learning agents.

The artefact discovered that it is very possible to create learning agents within video games and they are effective at what they do, the biggest drawback from them is the time to fully create and train them and the resources needed to replace an already working AD-OHC agent. Learning agents could be the future of agents within future video games with more development and developers taking the chance to implement this need system.

The artefact also discovered that most programmers could easily create learning agents from home, once past the setup phase the creation of the learning agents is easy with the only issues being where and when to reward the agents for their actions and creating a correct curriculum. Unlike the ad-hoc agents needing programming for all the features Learning agents will learn the features based on rewards and could even perform actions not suspected of them. The impression that it will take too much time and resources to create the learning agents at least for Unity Machine Learning agents (Unity Technologies, 2020) is unfounded as the possibility to create advanced agents is there for developers wishing to implement them. The biggest issue facing learning agents is getting them train correctly and not perform negative actions but with more development and time this issue could easily be fixed.

For learning agents to be a new feature in a video game a better SDK will need to be created as the current Unity Machine Learning (Unity Technologies, 2020) SDK needs allot of setup and developers have run into issues using python and creating the machine learning agent. A better SDK would involve an all in one package that can learn, edit settings and create a better environment for learning agents.

Upon reflection the AD-HOC had too much time in the development phase. Spending at least 1 month creating it and it being the first thing created it left little room to create the Learning agent and prototype and fix issues. Luckily, the learning agent came out with little issues within a week, but more time should have been spent on the Learning Agent.

Ideally this project should have had a bigger team and more resources to create a more advanced prototype for both the AD-HOC agent and Learning agent, this would hopefully create a better comparison of the two agents with more experienced teams creating them and having assets like trees and helicopters for the project to be more complex and increase the learning challenge for the agents. With these resources would create a better project to analysis the effectiveness and perhaps fix the issues with the childlike nature of the learning agents.

The project needed more time on learning settings, with the unity freezing randomly with the trainer settings which is still an unknown bug which was fixed by just having default settings from (Immersive Limit, 2020). More time should lead to a better understanding with what you can do with the learning settings and how it can change the behaviour of the agent it has taught.

With the knowledge gained on learning agents more time needs to be spent planning how they are going to learn and exactly what they need to know. This time will better understand how the features are going to reward the agent for its actions and how and when the agent will learn actions.

The project was a success but better time management and less time spent on the ad-hoc agent would lead to the project being completed earlier and more time spent on better techniques for the learning agent and better understanding how to use it. The biggest issue was planning and following the plans, allot of things changed very quickly leading too little to no planning and just creating the project.

From creating this project the developers theorise that the time needed to create a full learning agent is 1 to 2 years, with the current crisis and the development cycle of most games any game in mid to late development will either come out late 2020, or mid-2021, with games in early development/access unless the developers already are implementing learning agents which is unlikely these games should be finished by late 2021 to 2023 depending on issues and size of the game meaning most games in the near future will have AD-HOC agents and learning agents should start appearing 2022 to 2023 depending on technology and SDKS.

This project also discovers that learning agents could be a gimmick or not the future if they are not developed correctly. This is due to the nature of learning agents being random and sometimes uncontrollable in their actions, learning agents could just be seen as a "cool" feature developers can push for but do not need in their video games as ad-hoc is already at the point of nearly successfully creating realistic AI.

Learning agents should be the future of AI if the development and software carries on the way it is and developers start using it in video games, most players would not notice the difference between the two agents except from little differences in how they behave but even then there is not a big noticeable difference visually between the two.

The study was not conducted due to Covid-19, with the test being blind there was no way I could accurately monitor and ensure people were playing the game in the intended order and not cheating. The decision was taking early to abandon the group study and focus on the agents chasing targets instead of players so the project could still be created and tested.

Chapter 7: Conclusion

In conclusion, this project demonstrates that a learning agent is effective and could be the future of AI within video games. AD-HOC will likely be the future for the next 2 or 3 years of games releasing as most games coming out in this time are already in development with ad-hoc agents.

Learning agents are particularly suited to being used as commanders for strategy games due to them having a rule set with parameters which the agent can easily learn from due to the limitation of what units can do in strategy games. Strategy games are likely to be the first games to involve Learning agents due to the above reasons.

Learning agents can be used for a multitude of different titles other than strategy if the developers wanted to implement them, the main drawback is the time needed for training and creation of the agent for the game. Ad-hoc is simpler to create and most engines have in built methods for these agent's creation.

Ad-hoc agents are currently still very effective at portraying intelligence within video games. The techniques can simulate agents that can go against most players which is all most developers need. The push of multiplayer only games could be a factor into learning agents not being the future of AI within video games. Most multiplayer games are not in need of AI and if they do, they will use NPCS with AD-HOC techniques for shops and other basic interactions with the world. well intelligence platforms may be less needed and learning agents could be a gimmick in some niche games.

Overall, the project was a success and showed that learning agents are possible within video games, and all developers need is a push to be using the learning agents within their video games. Learning agents could well be the future of AI in video games or could be forgotten in the next year it all depends on how well the networks can work with players.

It is the recommendation of the developers for future work to investigate the following topics:

- The correct teaching technique for learning agents
- The application of learning agents in a multiplayer only game
- The application of learning agents in a single player game
- The application of learning agents for world building/procedural generation

References

Bohemia Interactive. (2013, September 12). ARMA 3. *ARMA 3*. Prague, Czechia, Czech Republic: Bohemia Interactive.

Berges, V.-P., & Meuleau, N. (2019). Unity AI and Machine Learning Tools for Behavior Creation. *Unity AI and Machine Learning Tools for Behavior Creation*. GDC. Retrieved from https://www.gdcvault.com/play/1026172/Unity-AI-and-Machine-Learning

Biran, Y., Finch, C., Khim, S., O'Brien, R., & Subramanian, S. (2019). Getting Started with Machine Learning and Artificial Intelligence. *Getting Started with Machine Learning and Artificial Intelligence*. GDC. Retrieved from https://www.gdcvault.com/play/1026138/Getting-Started-with-Machine-Learning

Blizzard Entertainment. (2010, July 27). StarCraft II: Wings of Liberty. *StarCraft II: Wings of Liberty*. Blizzard Entertainment.

Boser, B. E., Sackinger, E., Bromley, J., LeCun, Y., & Jackel, L. D. (1992, Febuary). *Hardware Requirements for neural network pattern classifiers*. Retrieved from AT&T Bell Laboratories: http://yann.lecun.com/exdb/publis/pdf/boser-92a.pdf

Briskman, D. (2018). Reinforced Machine Learning: Leveraging the Power of Intelligence for Gaming . GDC. Retrieved from https://www.gdcvault.com/browse/gdc-18/play/1024835

Buckland, M. (2011). *Programming Game AI By Example.* Sudbury: Jones and Bartlett Publishers, Inc.

Catanzaro, B. (2017). Deep Learning for Game Developers. *Deep Learning for Game Developers*. GDC. Retrieved from https://www.gdcvault.com/play/1024260/Deep-Learning-for-Game-Developers

DeepMind. (2019, January 24). *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*. Retrieved from Deep Mind: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

Forza Support. (2019). *Drivatars*. Retrieved from Forza Support: https://support.forzamotorsport.net/hc/en-us/articles/360005302934-Drivatars

Gordon, B. M. (2011). *Artificial intelligence approaches, tools and applications.* New York: Nova Science Publications, Inc.

Hammoudi, S., Okumura, J., & Tanaka, I. (2019). Applying AI in Games with DeNA. GDC. Retrieved from https://www.gdcvault.com/play/1026157/Applying-AI-in-Games-with

Haverford College. (2015). *Science Exposed*. Retrieved from Psych 2015: http://ds-wordpress.haverford.edu/psych2015/projects/chapter/blind-testing/

IBM. (1997, May 11). Deep Blue. *Deep Blue*. IBM.

Immersive Limit. (2020, Febuary 20). *Reinforcement Learning Penguins*. Retrieved from Immersive Limit: https://www.immersivelimit.com/tutorials/reinforcement-learning-penguins-part-1-unity-ml-agents

Luo, J. J. (2019, May 22). *An Exploration of Neural Networks Playing Video Games*. Retrieved from Towards Data Science: https://towardsdatascience.com/an-exploration-of-neural-networks-playing-video-games-3910dcee8e4a

Macri, D. (2017, November 19). *An Introduction to Neural Networks With an Application to Games*. Retrieved from Intel Developer Zone: https://software.intel.com/en-us/articles/an-introduction-to-neural-networks-with-an-application-to-games

Namco. (1980, May 22). Pac-man. *Pac-man*. Namco.

Nicholson, C. (n.d.). *A Beginner's Guide to Neural Networks and Deep Learning*. Retrieved from Pathmind: https://pathmind.com/wiki/neural-network#apllyingLossFunction

Nvidia. (n.d.). *Deep Learning*. Retrieved from nvidia Developer: https://developer.nvidia.com/deep-learning

Obsidian Entertainment, Virtuos. (2019, October 25). The Outer Worlds. *The Outer Worlds*. Private Division, Take-Two Interactive.

Pomerleau, D. A. (1991, March). *Efficient Training of Artificial Neural Networks for Autonomous Navigation.* MITP.

Rabin, S. (2017). *Game AI Pro 3: Collected Wisdom of Game AI Professionals.* Boca Raton: Taylor & Francis.

Reynolds. (2000, June 2). *Research*. Retrieved from Reynolds: https://www.red3d.com/research.html

Siedler, P. D. (2020, March 19). *How to install Unity ML Agents 0.14.1.* Retrieved from Youtube: https://www.youtube.com/watch?v=2wc5gttJ9Oc

SINGH, H. (2019, February 24). *Everything you Need to Know About Hardware Requirements for Machine Learning*. Retrieved from EInfochips: https://www.einfochips.com/blog/everything-you-need-to-know-about-hardware-requirements-for-machine-learning/

Skinner, G., & Walmsley, T. (2019). *Artificial Intelligence and Deep Learning in Video Games A Brief Review.* IEEE.

Star Theory Games. (2015, August 18). Planetary Annihilation Titans. *Planetary Annihilation Titans*. Kirkland, Washington, United States: Planetary Annihilation Inc.

Suspicious Developments. (2017, September 21). Heat Signature. *Heat Signature*. Suspicious Developments.

The AlphaStar Team. (2019, October 30). *AlphaStar: Grandmaster level in StarCraft II using multi-agent reinforcement learning*. Retrieved from Deep Mind: https://www.deepmind.com/blog/article/AlphaStar-Grandmaster-level-in-StarCraft-II-using-multi-agent-reinforcement-learning

Turn 10 Studios, Playground Games. (2018, September 28). Forza Horizon 4. *Forza Horizon 4*. Redmond, Washington, United States of America: Xbox Studios.

Unity. (2018). *Get to grips on the latest AI power in Unity*. Retrieved from Unity: https://unity3d.com/how-to/unity-machine-learning-agents

Unity Technologies. (2019, December 11). Unity. *Unity*. Unity Technologies.

Unity Technologies. (2020, April 30). *Unity Machine Learning Agents Toolkit* . Retrieved from Github: https://github.com/Unity-Technologies/ml-agents

Yannakakis, G. N., & Togelius, J. (2018). *Artificial intelligence and games.* Cham, Switzerland: Springer.

Appendences

1 – Survive, Evade, Resist, Extract Design idea

S.E.R.E

Quick Fire

- S.E.R.E is a game based on the United Kingdom Special Forces selection process; the game aims to simulate this process of evading a force hunting for you.
- The game puts you in control of the hunted, with your fellow players you must escape to a point on your map.
- 5-minute head start.
- Players leave behind scent smells for dog and thermal footprints as well as other methods of tracking them down including audio
- S.E.R.E has two games as it is designed to be tested, the games are called game 1 and game 2 so not to give away which AI system is controlling them
- The players will have 30 minutes with each game.
- The players are given a weapon to defend themselves with, However the weapon is loud and will attract more attention.
- The player can win in 2 ways, kill all the enemies or reach the extraction point.

Brief:

· S.E.R.E is a game based on the United Kingdom Special Forces selection process; the game aims to simulate this process of evasion. The learning AI will be trained to be the people searching for the players. The AI will get input from dogs, helicopters, thermal cameras and more to help them in their search. The learning AI will learn and use searching patterns to find the players within an area and their objective is to capture the players. The players must reach an extraction point without being caught by the AI. The world is procedurally generated which adds challenge both to the AI and players, the world is roughly 10km squared and can have various biomes and areas in it including jungle, woodland, rivers, desert, urban, fields and more.

Units

1. Rifleman
2. Tracker
3. Dog
4. Helicopter
5. Drone
6. Team Leader
7. Thermal Camera Operator
8. Sniper
9. Squad Leader

10. Platoon Leader
11. Company Commander

AD-HOC AI (Game 1)

The AD-HOC AI has the same information given to them as the learning AI. The only difference between the two systems is the AD-HOC ai are using a goal driven behaviour tree system to choose their actions. The AIS in this system always passes the information up the chain of command.

Learning AI (Game 2)

- The learning AIS information is received through a system called filter. The Agent knows what they see, and what their "senses" perceive, and they call out any important information to the person leading their team. This team leader through their learning decided if that information needs to be passed up the chain of command or not. If they call it out on the radio, every other team leader knows this information, team leaders then choose if they filter that information back down through their learning. This is a basic way of explaining it
    - Ok so the teams are organised how the military would have them, as I am simulating this I have given the learning AI profiles based on their rank, the higher their rank the more leadership weighed their decisions are.
    - So you have a team, squad, platoon. The company commander is the stop point of all commands. He is the most sophisticated learning AI as he is making the decisions of where the platoons are going and through game systems receives the most information.
    - Each agent in this will be learning agents, this will test how effective the process is.
- There is learning agent profiles, a method I created to simulate dynamic behaviour while still being predictable in the way I intend it to be
    - Rifleman
    - Dog
    - Helicopter
    - Thermal Camera Operator
    - Sniper
    - Team Leader
    - Squad Leader
    - Platoon Leader
    - Company Commander

2 - S.E.R.E Pitch Document

S.E.R.E: Survive, Evade, Resist, Extract. Game Design Document

High Concept

Players must escape from a hunter force and reach an extraction point, there are multiple ways to escape or be captured. Players can choose to fight and kill all the hunters or escape without a trace.

Unique Selling Points

1. Learning AI
2. Slow Gun play
3. Changing level design
4. Focused game play

Game Overview

In S.E.R.E you play as an armed forces member trying to get into the United Kingdom Special Forces. You are at the final stage and are being challenged with the final S.E.R.E trial. Your objective is to escape the hunters and use any skills, equipment or terrain to your advantage and learn how to avoid an enemy force actively hunting you. The game is open world with a border at the edge of the world to stop the players and ai from accidently leaving the area. The game is primarily stealth, but players can run and gun.

Game Objective

The primary objective of the game is to survive and escape without the enemy force knowing you were there. The players must reach a given extraction usually point a reasonable distance away to add the challenge.

Game Genre

Stealth Action - Adventure with open world elements

Target Audience

The target audience of S.E.R.E is 17+ with it aiming towards audiences who like stealth games and wish to be challenged.

Controls

W: Go Forwards
A: Go Back
S: Go Left
D: Go Right
Mouse: Aim
Left Click: Shoot
Right click: Scope in/zoom
Space: Jump/climb
Shift: Sprint
Z Prone/Stand up
X Crouch/Stand up
Control: Ultra Slow Movement
M: Map

Gameplay mechanics

Learning AI

The learning AI system will be a hybrid system custom made for the game. AD-HOC will be the body and complete the actions that the learning AI brain chooses to do, this way the learning AI will not have to learn steering behaviours. The brain will have information stored with what it knows acting like an active memory which includes details told to them by squad members and what they have seen. They take this information and choose an action based on the desired strategy. Each learning AI will be a profile assigned to a role that way it only trains and learns what it needs to know. Each AI will be slightly different in how it performs an action, but they should all roughly perform in the same light. The learning AI will also learn what the players do and start to adapt against them.

Senses

The AI will have simulated senses with being able to hear, see and smell. Some roles like dogs have a heightened sense of smell and can track the players using this sense. Each role has different sense which tell its brain information in that area and then pass up that information.

Filter (Communications)

The AI uses a system which I am calling filter, essentially the AI chooses to pass information up or down the chain of command. Each role has a radio they are on and can hear for example squad leaders are on their platoon radio and team radio and can hear and pass information on both. This system is used to allow the player to break up the radio communication and talking and also to make sure that the AIs are not always all knowing of each other's action they have to choose to pass this information to each other. The short way it works is an AI senses something and then it triggers a response and if the brain believes it should communicate that up to its team leader they will.
Example of how this works:

1. Dog smells a player more clearly and hears a branch break
2. Dog decides to bark as its brain thinks the player is close
3. Handler hears the dog bark and knows that this means a player is nearby
4. Handler tells their team leader that their dog may have found something
5. Team leader passes onto their squad leader
6. Squad leader calls for their squad to regroup on the handlers position and perform a sweep of the area.
7. Squad leader informs their platoon leader of this action.
8. Platoon leader calls for the other 2 squads to quickly move to this location and start sweeping in a pattern 500 meters away
9. Platoon leader tells their company commander of a player being near their platoon and tells the company commander the location
10. Company commander will inform all units of the possible location of the player and then their brains will make decisions on what to do

Sound

The game will have basic sounds like branch breaking, dogs barking, helicopter engines, radios and ai "talking" to each other and gunshots that way the player and AI can all hear these sounds. Sound quality does not need to be paramount and can be done by getting sample sounds.

Basic Combat

The game will have basic projectile combat where shoot a small ball at the enemy and they shoot at you with the same the small ball deals 20 damage no matter where it hits. There are no "guns" or weapons in the inventory instead everyone has the ability to shoot at a set fire rate and all have the same weapon.

Procedural map

The map will be procedural generated in a 10KM by 10KM system, first a map theme is chosen from Desert, Jungle, Forest or marshland. This choice gives the height map generator parameters to generate a random terrain for the players to walk across. Then the features like rivers, trees, rocks and more are placed. A rough texture and map are then created.

AI Roles/Profiles

Each AI will have a role which then gives them their brain. The roles or profiles are just another name for the AIS brain which chooses actions based on their role, so a helicopter would only have actions and choices based on what a helicopter can do so they won't need to be able to fire or have any code for shooting at players . Roles will change what the AI is able to do. The current planned roles are

- Helicopter
- Dog
- Rifleman
- Thermal Camera Operator

- Team Leader / Squad Leader
- Platoon Commander
- Company Commander

Game flow

The game is fully single player with the learning AI already being trained and having the necessary data for it to run, it can still learn from its actions and create new responses towards the players actions. The game is open world and has the ability for the players to hide from the hunting force and use the terrain to their advantage. The game's progress is linked to how competent the player is and in theory the player may never reach the extraction point but also never be found.

Technology

The game will use unity due to how quick and easy it is to create a prototype game and the ease of systems in unity, with the added benefit of unity already has open source machine learning agents I can learn and adapt for the project. Other technology includes any AD-HOC system I need for the game to work.

3 – Gifs of the project

Gif 1 Learning AI vs AD-HOC:

On the left is the Learning AI, right is AD-HOC AI. This gif shows a comparison between the two AI systems

https://gyazo.com/9cbd0c14afb5ffeeb011024b5a1f23b9

Gif 2 Brain 2 Complete:

https://gyazo.com/28bdd227cb71ff4e7684cf67d14f162c

Gif 3 Brain 2 Learning:

https://gyazo.com/a2efecdcb4334c6041dbdc62517bb454

4 – Curricula

```
S.E.R.E Learning:

  measure: reward

  thresholds: [-0.1, 0.7, 1.7, 1.7, 1.7, 2.7, 2.7]

  min_lesson_length: 200

  signal_smoothing: true

  parameters:

    Agent_Speed: [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 6.0, 6.0]

    Extraction_Radius: [3.0, 4.0, 5.0, 1.0, 2.0, 3.0, 0.5, 0.2]
```

5 – Gannt Chart

https://docs.google.com/spreadsheets/d/1A0FdfkHbrf_67U4EqEaqa2PIua8K7OpnN5qiZ-Xku_g/edit?usp=sharing