NEWCASTLE UNIVERSITY

**Investigating AI approaches and how they challenge resolutions: State AI vs Machine Learning**

**Chloe Sunter | 180273436**

MComp Computer Science

May 2022

Supervisor: Dr Giacomo Bergami

Word Count: 1234

Abstract

Declaration

“I declare that this dissertation represents my own work except where otherwise stated.”

Acknowledgements

Firstly, I would like to thank my supervisor, Dr Giacomo Bergami, for his guidance and support through each stage of the process.

Dr Giacomo Bergami

Dr Rich Davison???

Dr Graham Morgan

Dr Gary Ushaw

Contents

[Abstract 2](#_Toc103023231)

[Declaration 2](#_Toc103023232)

[Acknowledgements 3](#_Toc103023233)

[1 - Introduction 6](#_Toc103023234)

[1.1 - Dissertation Structure 6](#_Toc103023235)

[1.2 - Definition of Problem 6](#_Toc103023236)

[1.3 - Hypothesis 6](#_Toc103023237)

[2 - Aims & Objectives 7](#_Toc103023238)

[3 - Background Review 8](#_Toc103023239)

[3.1 - Hardpoint 8](#_Toc103023240)

[3.1.1 - Rules 8](#_Toc103023241)

[3.1.2 - Examples from the real world 9](#_Toc103023242)

[3.2 - AI 9](#_Toc103023243)

[Overview 9](#_Toc103023244)

[State AI 9](#_Toc103023245)

[Finite state, heiracly finite state, probalistic finite 9](#_Toc103023246)

[Machine Learning 9](#_Toc103023247)

[Algorithms, compare etc, Deep Reinforcement, Imitation Learning 9](#_Toc103023248)

[3.3 - Tools & Technologies 9](#_Toc103023249)

[Software 9](#_Toc103023250)

[Unity, compare other options perhaps. 9](#_Toc103023251)

[Mlagents, do a lot of research into this. What alorithms it uses etc 9](#_Toc103023252)

[Hardware 9](#_Toc103023253)

[Kind of like system requirements. Obvs tho, I don’t match those. Maybe try to find some graphs /studies that show how better equipment can affect the training times etc 9](#_Toc103023254)

[3.4 - Similar Projects 9](#_Toc103023255)

[4 - Project Development 10](#_Toc103023256)

[4.1 - System Setup 10](#_Toc103023257)

[4.1.1 - Hardware 10](#_Toc103023258)

[4.1.2 - Version Control 10](#_Toc103023259)

[4.1.3 - Software 10](#_Toc103023260)

[4.2 - Design 11](#_Toc103023261)

[4.2.1 - Map Design 11](#_Toc103023262)

[4.2.2 - Gameplay Rules 11](#_Toc103023263)

[4.2.3 – Player Design 11](#_Toc103023264)

[4.2.4 - State Machine 11](#_Toc103023265)

[4.2.5 - Deep Reinforcement setup? 20](#_Toc103023266)

[4.3 - Implementation 20](#_Toc103023267)

[4.4 - Testing 20](#_Toc103023268)

[5 - Results & Evaluation 21](#_Toc103023269)

[6 - Conclusion 22](#_Toc103023270)

[7 - References 23](#_Toc103023271)

[8 - Appendices 24](#_Toc103023272)

1 - Introduction

1.1 - Dissertation Structure

1.2 - Definition of Problem

1.3 - Hypothesis

2 - Aims & Objectives

3 - Background Review

3.1 - Hardpoint

Hardpoint is a game style that is similar to Capture the Flag and has been implemented as a main game mode in many popular online multiplayer games. Whilst this style of game is commonly known as “Hardpoint”, its name varies across different franchises. Other names include: “Stronghold” (Halo), “Headquarters (Call of Duty), “Hardpoint Domination” (Titanfall) to name a few. Each franchise has modified the game to fit their own style, however the foundation stays the same.

3.1.1 - Rules

Overview

Two teams play against each other on a map that has designated areas (hardpoints) that reward points when players enter them. The minimum amount of hardpoints is 3, with one of these hardpoints being equal distance from both team’s respawn areas. The aim of the game is to beat the opposing team by reaching the total score before them (for example, 100points). Points can be earnt by capturing the hardpoint from the other team and for defending the hardpoint from the opposing team.

Capture

A player can only capture a hardpoint that is either owned by the enemy or has yet to be captured by anyone for the first time in a game. To capture it, a minimum of one player must enter the hardpoint and stay within the area for a small period of time (for example, 3seconds). If an enemy enters the area during this time, the hardpoint will become congested. Once captured, the team gains a large portion of points (for example, +15points).

Defend

A player can only defend a hardpoint that is owned by its own team. To defend it, a minimum of one player must enter and stay within the area for an extended period of time (for example, 10seconds). If an enemy enters the area during this time, the hardpoint will become congested. Once defended, the team gains a small portion of points (for example, +5points). This will repeat for as long as a player is within the hardpoint and it belongs to said player’s team.

Congested

If players from both teams are present in the hardpoint at the same time, it becomes “congested” and no points are rewarded whilst the hardpoint is in this state. To exit this state, a team must remove all enemy players from the area. Once complete, the hardpoint will go back to the state it was beforehand and normal gameplay continues.

Kills

Players can kill other players, however, this does not affect the score. Respawns are enabled.

3.1.2 - Examples from the real world

3.2 - AI

Overview

State AI

Finite state, heiracly finite state, probalistic finite

Machine Learning

Algorithms, compare etc, Deep Reinforcement, Imitation Learning

3.3 - Tools & Technologies

Software

Unity, compare other options perhaps.

Mlagents, do a lot of research into this. What alorithms it uses etc

Hardware

Kind of like system requirements. Obvs tho, I don’t match those. Maybe try to find some graphs /studies that show how better equipment can affect the training times etc

Please see <> for specific versions used in this project.

3.4 - Similar Projects

4 - Project Development

4.1 - System Setup

asfasfd

4.1.1 - Hardware

|  |  |
| --- | --- |
| Device | Laptop |
| Processor | Intel® Core™ i5-7200U CPU @ 2.50GHz, 2713 MHz |
| Core(s) | 2 |
| Logical Processor(s) | 4 |
| Operating System | Microsoft Windows 10 Home |
| Memory *(RAM)* | 8GB |
| Graphics Processing Unit *(GPU)* | Intel® HD Graphics 620 |

Table X -

4.1.2 - Version Control

|  |  |
| --- | --- |
| Git | version.2.21.0.windows.1 |
| Github Desktop |  |
| Gitbash |  |

Table X -

4.1.3 - Software

Editors

|  |  |
| --- | --- |
| Unity | version.2021.1.21f1 |
| Vs Code | version.1.67.0 |
| Microsoft Excel | version.2204 |

Table X -

Languages

|  |  |
| --- | --- |
| Python | version.3.9.9 |
| C# | n/a |
|  |  |

Table X -

Toolkits, Frameworks, Libraries & Packages

|  |  |
| --- | --- |
| ML-Agents Toolkit | release.19 |
| com.unity.ml-agents | version.2.2.1-exp.1 |
| com.unity.ml-agents.extensions | version.0.6.1-preview |
| ml-agents | version.0.28.0 |
| ml-agents-envs | version.0.28.0 |
| gym-unity | version.0.28.0 |
| Communicator | version.1.5.0 |
| Probuilder |  |
| Tensorflow | version.2.8.0 |
| Tensorboard | version.2.8.0 |
| Torch | version.1.7.1+cu110 |
| Pip | version.22.0.4 |
|  |  |

Table X -

*Please note: This is not a full list of all packages included. It only includes the most prominent ones which may be mentioned throughout this paper.*

4.2 - Design

4.2.1 - Map Design

4.2.2 - Gameplay Rules

I have implemented the basic version of the Hardpoint game mode outlined in <>. Below is a list of parameters that I have used to run the game. These will stay the same for all playthroughs so as to not affect the result comparisons.

Total number of teams: 2

Total number of players per team: 4

Total number of Hardpoints: 3

Score needed to win: 100

Time (seconds) it takes to capture: 3

Time (seconds) it takes to defend: 10

Points earnt for capturing: 15

Points earnt for defending: 5

Points earnt for killing an enemy: 0

Respawns enabled: Yes

4.2.3 – Agent/Player Design

4.2.4 - State Machine

The State AI agents will be implemented using a Probabilistic Finite State Machine. This section will explore each state and the transitions to and from it, along with the probabilities for each weighted decision. In the following figures, the representations are as follows:

|  |  |  |
| --- | --- | --- |
| Icon  Description automatically generated | Oval | Start/Stop Terminator |
| Shape  Description automatically generated with medium confidence | Arrow | Transition (directional) |
| Rectangle  Description automatically generated with medium confidence | Square/Rectangle | State |
| Shape, icon, rectangle  Description automatically generated with medium confidence | Parallelogram | Data - Probabilities for any weighted decision |
| A picture containing clipart  Description automatically generated | Diamond | Decision |

Graphical user interface, application

Description automatically generated

Figure 1 -

Figure <> above is a simplified representation of the finite state machine to be used. It features five states (Search For Hardpoint, Move, Patrol, Attack, Search For Enemy) that are interlinked with each other and transitions between states must follow the flow shown. A further two states (Respawn, Flee) have also been added and these can be entered at any stage regardless of the current state. For some transitions between states it requires a decision to be made and this decision is weighted with the aim to make the State AI agent less predictable.

States

Respawn

During this state, the agent will respawn. This includes resetting the agent’s data and moving it’s position back to the respawn area. Respawning will include a delay of 5 seconds.

Flee

Whilst in this state, the agent will turn and run to a point a that is a certain distance away, for example, 10 metres. The direction that it will run in will depend on whether it can see any enemies or not. If it can see an enemy, it will run the in opposite direction to that enemy. Otherwise, it will run forward, with the logic that if needs to flee because it has been hit but cannot see any enemies, then the enemy must be behind it. The aim of this state is to allow the agent to get out of the field of vision of an enemy.

Search for Hardpoint

Within this state, the agent will decide which hardpoint it would like to move to. This decision will be weighted based on the current state of the hardpoints on the map, and the distance the agent is from the hardpoint. <>

Move

Whilst the agent is in this state, it will gradually move towards the hardpoint is has chosen, using the shortest path, and avoiding obstacles.

Patrol

This state will occur only when the agent is within a hardpoint. To collect points as part of the game rules outlines in section <>, it will need to stay within the hardpoint. However, staying still will make it an easy target for enemies, therefore it should move small distances within the area. The direction and distance the agent will move whilst patrolling should be randomised.

Attack

This state will include the agent rotating to face an enemy and then firing a projectile at it.

Search For Enemy

The agent may be getting attacked from behind and therefore is unable to see the enemy but will know it is being hit. Therefore, during this state, the agent will rotate on the spot until it can see an enemy.

Transitions

Please note, the transition between states into the Search for Enemy and Attack states has been explained below Figure <>. This will appear on multiple diagrams from then on, so please refer to Figure <>.

Shape

Description automatically generatedA picture containing diagram

Description automatically generatedShape

Description automatically generated with medium confidenceFigure X - Figure X - Figure X -

Both the Respawn and Flee states can be entered at any point during the game regardless of what state the agent was previously in. This differs from the other states, for which their transitions are limited and must include the same two states at all times.

As seen in Figure <> the transition from any state to Respawn will happen when the player is marked as dead and the transition from any state to Flee will happen when the agent has decided to run away. The decision to flee will be a result of a weighted decision check during many of the other transitions. Please see Figures <>,<>,<>,<> for further clarification.

The Flee state involves movement of the agent, and therefore a check to make sure the agent hasn’t gotten stuck is included. In the case that they have, it will invoke a transition to the Search For Hardpoint state to force the agent to change it’s behaviour. Likewise, once the agent has finished fleeing, they will now need to re-evaluate their surroundings, aka the hardpoints and the distance to them. Therefore, they will enter the Search For Hardpoint state to do this. This can be seen in Figure <>.

Whilst an agent is within a Respawn state, they will stay within the state until 5 seconds has passed. At this point, a transition will occur and the agent will now enter the Search For Hardpoint state to start engaging in the game again. This can be seen represented in Figure <>.

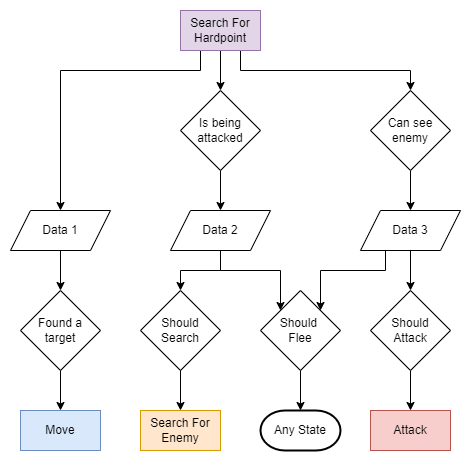


Figure X -

An agent in the Search for Hardpoint state can move onto three other states, Move, Search For Enemy, and Attack as shown in Figure <> above. During the Search for Hardpoint state, it will have decided on a hardpoint to head towards, therefore it can transition to the Move state almost instantly and this is the default transition. This decision is represented by Data1 in the figure above. However, this can be interrupted by the transition to Search for Enemy and Attack which will occur due to weighted decision checks, represented by Data2 and Data3 on the figure. This decision check will occur when triggered by a collision detection on the agents enemy detection component. For example, if the agent sees an enemy, it will run the decision check in Data3. If the agent is taking damage, it will run the decision check in Data2. See section <> for details surrounding the percentages used for the decision in Data1, Data2, and Data3.

The reason for not including the states Search for Enemy and Attack in the Any State section is because there could be no interruption of the Respawn state allowed and there would also be the chance that the Search For Enemy state could be triggered over and over again, causing the agent to get stuck in a loop. Manually selecting which states could move to which states helps to reduce the chances of unexpected behaviour.

Shape

Description automatically generated

Figure X –

An agent can move from the Move state to four other states as shown in Figure <>. There is the chance that an agent may get stuck on a wall due to collision shapes or the destination chosen may no longer exist. This would cause the agent to stay in the same place and never transition out of the Move state. Therefore, I will include a check that will allow the agent to reset back to the original state if it stays still for too long. As long as this doesn’t occur, then the agent will head towards the target and once it has reached it, it will then transition into the Patrol state. Please see the note at the beginning of section <> for further explanation of the transition to Search For Enemy and Attack states.

Shape

Description automatically generated

Figure 2

An agent can move from the Patrol state to four other states as shown in Figure <>. There is the chance that an agent may get stuck on a wall due to collision shapes or the destination chosen may no longer exist. This would cause the agent to stay in the same place and increase the risk of never transitioning out of the Patrol state. Therefore, I will include a check that will allow the agent to go back to moving towards the target it has chosen previously. Alternatively, the agent will stay within the hardpoint and will collect some points. Upon collecting points, the agent will go back to the Search For Hardpoint state to decide on whether it will move to another hardpoint, or stay where it is. Please see the note at the beginning of section <> for further explanation of the transition to Search For Enemy and Attack states.

Shape

Description automatically generated Shape

Description automatically generated

Figure X - Figure X –

An agent can only attack if it can see an enemy to attack, therefore, it will transition back to the Move state if there are no enemies in sight. This way, it can continue to move towards a hardpoint. This process can be seen represented in Figure <>.

The aim of the Search for Enemy state is to allow the agent to rotate and try to find an enemy. In the case that this doesn’t happen, the state then changes to the Move state. This is a safety transition to stop the agent from being stuck rotating forever. Alternatively, if it can see an enemy, then it will use the weighted decision checker in Data3 to decide whether it wants to attack the enemy or flee.

Probabilities & Weighted Decisions

Data1

This will be represented by a method that will take into account the distance between the agent to each hardpoint and the state/owner of each hardpoint. This information will help form the decision for the agent’s next move. For example, if the agent’s team hasn’t captured any hardpoints yet, then the most logical decision would be to head to the nearest hardpoint. However, just moving to the nearest hardpoint all the time is predictable and all agents on the team would move as a group if starting from the same location. This causes issues. Therefore, each decision is weighted and there is always the chance that agent will do the least logical thing to keep it interesting and more ‘real’.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Situation | Probability (%) of each decision occurring | | | | | |
|  | Capture Nearest | Capture Second Nearest | Capture Furthest Away | Defend Nearest | Defend Second Nearest | Defend Furthest Away |
| If the nearest hardpoint has not been captured by the agent’s team | 80 | 15 | 5 | n/a | n/a | n/a |
| If the nearest hardpoint has been captured by the agent’s team but the other two are uncaptured | n/a | 85 | 10 | 5 | 0 | 0 |
| If the nearest hardpoint has been captured by the agent’s team and if the second nearest hardpoint has not been captured | n/a | 70 | n/a | 20 | n/a | 10 |
| If the nearest hardpoint has been captured by the agent’s team and if the second nearest hardpoint has also been captured by your team | n/a | n/a | 60 | 25 | 15 | n/a |
| If all hardpoint have been captured by the agent’s team | n/a | n/a | n/a | 60 | 30 | 0 |
|  |  |  |  |  |  |  |

Table X -

Data2

This will be represented by a method that will take into account the distance between the agent and the hardpoint it is moving towards when make the decision as to whether to look for the enemy attacking the agent, or whether to flee. The reasoning behind this is to help reduce the chances of the agent fleeing whilst inside the hardpoint as this is not a logical solution and would not occur often if humans were playing the game.

|  |  |  |
| --- | --- | --- |
| Situation | Probability (%) of each decision occurring | |
|  | Enter Search for Enemy state | Enter Flee state |
| If the agent is being attacked and it cannot see any enemies and it is more than 5m away from the hardpoint it was heading towards | 90 | 10 |
| If the agent is being attacked and it cannot see any enemies, but it is closer than 5m from the hardpoint it was heading towards | 70 | 30 |

Table X -

Data3

This will be represented by a method that will take into account how many enemies the agent can see, how close the enemies are to the agent, and how close the agent is to the hardpoint. This will allow logic such as fleeing when surrounded by multiple enemies, or taking the chance and fighting when there is only one enemy and the agent is so close to its goal.

|  |  |  |  |
| --- | --- | --- | --- |
| Situation | Probability (%) of each decision occurring | | |
|  | Continue in current state | Enter Attack state | Enter Flee State |
| If there is 1 enemy within 5m of the agent and the agent is less than 5m from the hardpoint | n/a | 99 | 1 |
| If there is 1 enemy within 5m of the agent and the agent is more than 5m from the hardpoint | n/a | 95 | 5 |
| If there is 2 enemies within 5m of the agent and the agent is less than 5m from the hardpoint | n/a | 98 | 2 |
| If there is 2 enemies within 5m of the agent and the agent is more than 5m from the hardpoint | n/a | 95 | 5 |
| If there is 3 enemies within 5m of the agent and the agent is less than 5m from the hardpoint | n/a | 95 | 5 |
| If there is 3 enemies within 5m of the agent and the agent is more than 5m from the hardpoint | n/a | 80 | 20 |
| If there is more than 3 enemies within 5m of the agent and the agent is less than 5m from the hardpoint | n/a | 90 | 10 |
| If there is more than 3 enemies within 5m of the agent and the agent is more than 5m from the hardpoint | n/a | 50 | 50 |
| If there is 1 enemy within 10m of the agent and the agent is less than 5m from the hardpoint | 32 | 67 | 1 |
| If there is 1 enemy within 10m of the agent and the agent is more than 5m from the hardpoint | 30 | 65 | 5 |
| If there is 2 enemies within 10m of the agent and the agent is less than 5m from the hardpoint | 46 | 52 | 2 |
| If there is 2 enemies within 10m of the agent and the agent is more than 5m from the hardpoint | 45 | 50 | 5 |
| If there is 3 enemies within 10m of the agent and the agent is less than 5m from the hardpoint | 47 | 50 | 3 |
| If there is 3 enemies within 10m of the agent and the agent is more than 5m from the hardpoint | 40 | 50 | 10 |
| If there is more than 3 enemies within 10m of the agent and the agent is less than 5m from the hardpoint | 66 | 30 | 4 |
| If there is more than 3 enemies within 10m of the agent and the agent is more than 5m from the hardpoint | 50 | 30 | 20 |

Table X -

4.2.5 - Deep Reinforcement setup?

4.3 - Implementation

4.4 - Testing

5 - Results & Evaluation

6 - Conclusion

7 - References

8 - Appendices