AI DIRECTOR ANALYSIS OF MUTLI-AGENT GUIDING

Adam Shrimpton, s014088m, Supervisor: Shaun Reeves

DTIB
BSc Games Design & Programming

Table of Contents

Table of Figures

Glossary

AI - Artificial intelligence
A* - A Star
ANN – Artificial Neural Network(s)
BT – Behaviour Tree
DDA – Dynamic Difficulty Adjustment
DL – Deep Learning
DNN – Deep Neural Network
DRG – Deep Rock Galactic
EA – Electronic Arts Inc.
EQS – Environment Query System
FSM – Finite State Machine
FuSM – Fuzzy State Machines
JPS – Jump point search
NavMesh – Navigation Mesh
ML – Machine Learning
SWAT – Special Weapons and Tactics
TTK – Time to Kill

Key Words

Artificial Intelligence

Decision Making

Behaviour

Game Balance

Difficulty Scaling

Environment Query System

Abstract

AI systems within games often lack in depth and cohesion which leads to predictable behaviours and no meaningful difficulty scaling. This project explores several key components of Artificial Intelligence using technologies for group behaviour. It examines algorithms used to optimise locomotion using pathfinding and steering behaviours and evaluates decision making models for effective AI behaviour. Additionally, the study considers the role of token systems to manage weapon firing and how this system could be used to dynamically scale difficulty of an AI. Whilst promising results were observed in some areas, further research is necessary to reach conclusive outcomes and refine the solution.

Introduction

What is the reasoning behind this project?

With an interest in artificial intelligence (AI), and a fan of First-person shooters, a focus to research and investigate methods that incorporate AI in a director system to better the understanding in AI and practice techniques to improve AI programming skillset.

Why is an AI Director system the focus of this study?

This specific interest for an AI Director to manage artificial intelligence was inspired by the game Alien Isolation, which uses a dual AI controller to provide a thrilling experience. This project is to try to create and expand a director system into a multi agent system allowing for dynamic game balance.

Aim

What does this study hope to achieve?

The aim of this project is to create an AI Director system that controls a group of AI in a SWAT team through a combat environment.
The AI group must detect and overcome hazards within the level as they appear, hazards will present themselves as traps and enemies. The AI will respond to these hazards with the appropriate actions as managed from the "Director" AI

Objectives

- How can a director manage agents to act cohesively in movement and interactions?
- How can an AI be created to respond to Hazards in real-time?
- How can difficulty be scaled for an AI?
- How can group AI be balanced for a player?

Literature Review

How can a director manage agents to act cohesively in movement and interactions?

How can AI travel as a group?

Steering Behaviours

"Steering behaviours are an amalgam of different behaviours that are used to organically manage the movement of an AI agent.", usually on a 2D Plane due to behaviours being easier to implement (jb-dev, 2018).
A steering behaviour system works with child behaviours calculating a vector representing how that

behaviour would like the agent to move. The result of all vectors combined produce a final desired velocity that the agent will steer towards (Fray, 2013).

Common steering behaviours include:

Seek & Flee
A force acting to steer the agent towards a location, the behaviour adjusts the agent so that the agent's velocity is radially aligned towards the target (Reynolds, 1999). Calculating a seek vector is simple, subtract the current position from the seek target, this results in a desired vector towards the target.
Flee is the opposite behaviour of seeking calculating a force to move the agent away, this is calculation is simple as subtracting the flee target from the current agent's position (Coenen, 2016).

Pursuit & Evade
Pursuit and evade are very alike to their base function of seek and flee except for the fact these behaviours use the targets velocity to predict where the target should be (Coenen, 2016). This is calculated by getting the desired velocity like flee and seek. Then the distance and targets current velocity is multiplied together and added to the targets location resulting in the agents desired vector to flee or seek.

Arrival
The arrival behaviour is the idea of an agent that will gradually slow down until it reaches its destination. When the behaviour is active the distance between the target and the current position of the agent and will change its maximum move speed accordingly (jb-dev, 2018).

Wander
Wander is often used in games when the agent does not have a task to complete and awaits an action. Simple options include picking a random position and seek towards that position, but this results in unrealistic behaviour. A better implementation is having a circle projected in a set distance away matching the velocity of the agent. This circle can then have a random radial direction from the central point on the circle which acts as a displacement from that location to seek to. This is updated each frame with slight adjustments to the random radial direction by subtracting or adding small increments (Coenen, 2016).

Collision Avoidance

Unaligned collision avoidance behaviour attempts to stop agents moving in arbitrary directions from colliding. To implement this an agent needs to consider other agents velocities and determines when the two will make their nearest approach. If a potential collision will occur then the two agents will steer to turn away from the potential collision (Reynolds, 1999).

Separation

Separation behaviour gives the agent the ability to maintain a distance from other agents within its vicinity, this can be used to prevent agents from crowding together. To calculate separation of agents a radial search is done to find the other agents that it would separate from, the flee method is invoked for each position of those found agents, added together and normalised to find the final steering direction (Reynolds, 1999).

Cohesion

Cohesion behaviour gives an agent the ability to form a group with other agents, this is a similar calculation to the separation behaviour but inversed so agents are all seeking towards a point determined by the averaged position (Reynolds, 1999).

Whilst steering behaviour locomotion is quick to calculate, "steering behaviours will not navigate through a complex environment." (Roberts, 2023) This error could be seen in the racing genre when using the flock behaviours to race the AI cars as seen in the F1 game (Codemasters, 2010). Often small movements from the player could cause immersion breaking results which led to missed overtaking opportunities, poor overtake blocking and sometimes even collisions (Fray, 2013).

Pathfinding

Pathfinding is the process of finding the shortest route between two end points (Xiao Cui, 2011). Many algorithms deal with this problem such as A*, Dijkstra and JPS. In a game it is important for a pathfinding algorithm to remain somewhat realistic, that is why the AI must calculate the path quickly and generate a path that is reasonable to the player. (Roberts, 2023, p. 91)

Generating a path quickly is imperative for games because no other game system will be able to run whilst the pathfinding algorithm is running which will cause lag spikes and frame drops, this effect when calculating paths can be mitigated by calculating the process over several frames to allow other systems to run their processes. The drawback of calculating over several frames is a reduction in perceived intelligence as the AI will have minimal action whilst waiting for the calculation to finish. To further minimise this behaviour of standing still, shorter paths can be created which follow the rough path whilst the AI is travelling. This way the AI will have the next path ready by the time it reaches its current pathfinding destination. (Roberts, 2023, pp. 91-94)

Spatial Representation – Pathfinding Subheading

Spatial representation is the process of graphing a space into mathematical equations that can be later used by pathfinding algorithms to find the quickest route to the desired location. Representing the space mathematically is possible in multiple ways, but the core approaches focus on Grid Based, Waypoint and NavMesh.

Grid Based involves breaking down the navigable area into cells to explore, when calculating a path each cell has a value and once the target location is found the shortest path is calculated by working through nodes with the smallest value. This method of representation works for small simple areas, but on a larger scale calculation increases due to the volume of cells that would need a path calculating.

Waypoint search requires nodes in the world that can selected, the AI is then able to jump navigate between these nodes similar to the grid-based approach.
Waypoint nodes require hand placement but could be generated through the use of a tool. Although this approach is good for large complex environments, it lacks the ability to navigate to anywhere between those waypoints which is not reasonable for believable game AI.

NavMesh spatial representation requires existing points similar to the waypoint method, more nodes are required initially due to the mesh needing outer nodes to link to form polygons.
As the polygons generate, unnecessary nodes are removed, and a node is created in the centre of the polygon which acts just like the waypoint method. By allowing for traversal between the nodes, this works well for user curated levels due to the adaptability of the algorithm.

Among pathfinding algorithms, A* is the most used (Roberts, 2023, p. 91). Popular game Engines such as Unreal Engine and Unity both use A* for their AI systems with a NavMesh to reduce pathfinding calculation time whilst maintaining realistic pathing.

How can AI find information about the world?

AI can obtain information about the world using few methods that mimic human levels of perception. Frequently used methods include perception and Environmental Query Systems (EQS) or custom trace systems.

Perception

As games progress to include increasingly advanced AI, perception awareness models have become common for AI systems. Perception systems are used to mimic human primary senses and prompt decisions from the data obtained by the artificial senses. This data can aid in creating a believable AI that only knows information from the perspective of the agent.
Visual perception models in games commonly use a cone to model what an NPC can see in front of them but lacks in modelling other aspects such as peripheral and distanced vision (Walsh, 2015, p. 314).

Distanced vision from a cone model is inaccurate because as the cone distance increases so does the search radius which suggests that the ai will perceive objects more laterally the further away something is. This problem can be reduced by using a box view (a cuboid shaped collider) to detect primary vision instead of a cone, this is due to the box not having an increased size as it gets further away.

In the game 'Tom Clancy's Splinter Cell Blacklist' the solution was to use a coffin shaped collider to --
--allow for peripheral and primary vision, this collider would expand outwards to a point like a cone to create the peripheral blindness and then contract as the collider gets further away from the NPC which reduces the lateral visibility (Walsh, 2015, pp. 314-316).

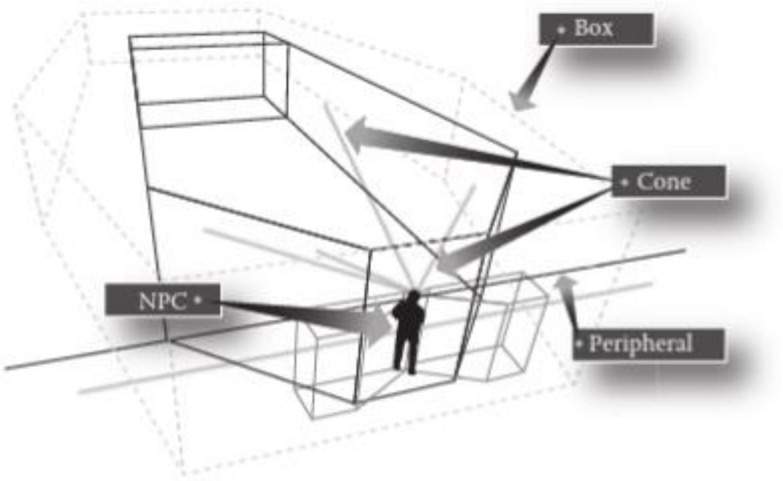


Figure 1 Coffin Collider

Auditory perception, another key sense often used in games, is used to create an additional layer of behavioural intelligence to an AI.

Creating an audio event for an AI to hear is easy, but the calculation of hearing distance is not as simple. Checking the Euclidean distance from the noise source to the agent like it would be for sight would not be effective, because an AI could hear through walls using that method (Walsh, 2015, p. 320).

A solution is to calculate an area path that the sound would travel to the agent, this is done by getting the closest route to the doors leading to the player and calculating the length of the vectors as seen in Figure 2 – Area path & Euclidean Distance Representation

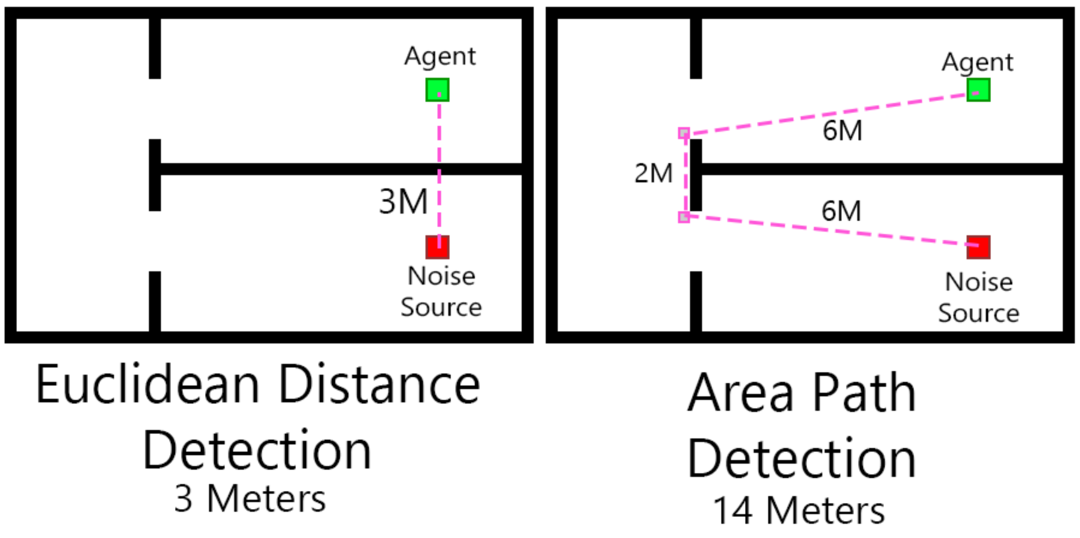


Figure 2 – Area path & Euclidean Distance Representation

Unreal Engine has a specialised AI perception system which uses artificial senses to obtain information about the environment, these senses search for actors that can return a positive value based on a sense configuration. Sense configurations allow for different detection methods such as sight, hearing, damage and touch. Stimuli data returns an actor which information such as location, owner or team can be returned. As an AI receives stimuli information, it filters data and stores any relevant information for decision making. (Epic Games, 2024)

Environment Queries
Environment queries can used to get information about the world through a variety of traces and collision tests. These tests are used for making simple decisions such as choosing between health or ammo pickups, enemy threat analysis and distance from player. A Query has three main components:

- Context Object: a game entity that is querying the environment. This is crucial as the entity will be able to ask if locations are visible from its current location or if they were in range of an object.
- Query Template Id, which question is it that we are asking the entity to run.

- Items: An array of items returned in the test. This list can be filtered by items failing further query tests

(Zielinski, 2013).

Unreal Engine has their own system to handle this called EQS (Environment Query System) which allows for these tests to be ran in tandem with their behaviour tree systems (Epic Games, 2024).
In the Unreal Engine EQS workflow, a generator node is required to state the sample locations from which the system will collect data. This generator is supplied with an object's position and will act as the reference point for the tests conducted. The best fed option(s) are returned as the result of the query (Haapanen, 2021, p. 11).

The Flexibility of EQS allows for tweaking the logic of a query and their attributes easily but requires a complex approach for creating custom generators which will slow down development iterations due to the increased workload.
Whilst the iteration process is fast for non-custom generated queries the system is limited by the amount of unnecessary data produced by empty generated points which each need to be weighed and tested against the queries. (Haapanen, 2021, pp. 40-41)

How can an AI be created to respond to Hazards in real-time?

What types of real-time decision-making methods are there?
Realtime decision making is critical for games because players actions will affect how the AI should respond. There are a few common decision-making models that are widely used within gaming that allow a range of simple to complex actions varying in complexity to implement.

State Machines (Finite State Machines & Fuzzy State Machines)
Finite State machines (FSM) are solutions to managing how an AI can act, typically used in simple or early models of AI (Roberts, 2023). Only one state can be active at a time, these states are comprised of actions that need to be followed, for example an explore state would contain an action to move randomly and seek opponents (Yannakakis, 2018, p. 33).
A limited number of behaviours are programmed into the AI which are selected that allow transitions between each state. Dependant on use, states are capable of transitioning between a single or multiple others dependant on the required exit condition to transition. Pac Man is a good example of this with two exit conditions from the evade state as seen in Figure 3 – Pacman State Diagram. If the power pellet is not active or the ghost has been eaten, the ghost will transition from the state out into a chase mode or return home.

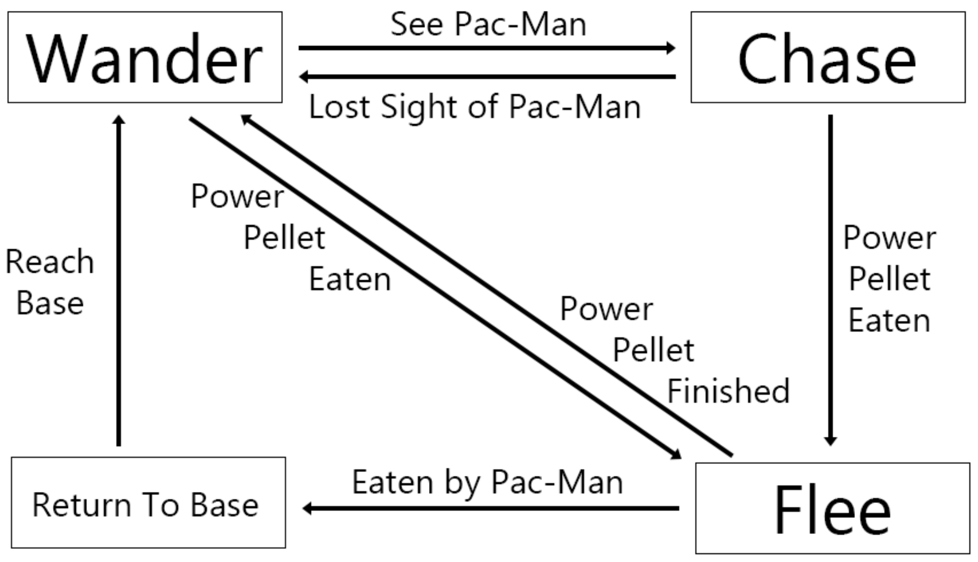


Figure 3 – Pacman State Diagram

FSMs are limited in complexity due to the use of the same conditions which allow for the state swapping, this can potentially break the emersion for a player if they spot the pattern. Another drawback is that the AI will only be in a single state at any time. (Roberts, 2023) Another issue is state thrashing, The AI could swap between two states with opposing actions repeatedly which gives the appearance of becoming stuck.
Fuzzy State Machines (FuSM) help solve the issue of single states that FSMs have by implementing available states that can be partially triggered, this means that an agent could have 40% towards state 1, and 60% towards state 2, these are fuzzy sets. Fuzzy sets allow for membership to be calculated towards a state to have a partial output. Instead of having set value ranges we can assess the membership based on a graph, this will allow for membership scaling to target different behaviours together with different activation points in the same query which will reduce the number of conditional statements that a FSM has. (Roberts, 2023, pp. 155-159)
This mixing of states allows for layer of complexity and unpredictable behaviour compared to the state machines but will also have a higher calculation cost due to the requirement of calculating every state contribution anytime the states are evaluated.

Behaviour Trees
A Behaviour tree is a knowledge system similar to an FSM. BT's due to their transitions between behaviours, "the strength of BT's compared to FSMs is their modularity". (Yannakakis, 2018, pp. 34-35)
BT uses a tree structure with a root node and branching child nodes which represent behaviours. Executing a behaviour tree starts with the root and sequentially executing parent-child pairs. Dependant on the type of composite node a child will return a value of failure or success.
Sequence Nodes will execute all the children until one fails, if one child fails the sequence is aborted and returns a fail.
A **Selector** will execute a node on a condition, if there is no condition the selector will try to find a child that succeeds then return the success result.

Decorators will enhance the complexity of a single child behaviour, this can be improved through a select number of times to execute or a set time for the task to be completed. This will return the condition of the task.
In addition to failure and success states a tick state will continue execution until a result state is returned, the branch execution will not proceed until this has finished. This is useful for nodes that require a pathfinding to be completed first.
In comparison to FSMs, BTs are more flexible to design and easier to test but suffer from low dynamicity due to being a static knowledge representation (Yannakakis, 2018, p. 37).

Utility-Based AI
Utility Based AI, a popular decision-making method, removing the modularity limitations of BTs and FSMs by implementing an ad-hoc style of behaviour authoring (Yannakakis, 2018, p. 37). This approach to AI is less popular in academia than in the game industry because Utility AI is primarily a design-based technique (Świechowski, 2024, p. 1).

The design of a Utility based System follows a similar method as a FuSM with its membership functions and fuzzy sets.
The data measured in this system can be anything from an observable object (e.g., health, ammo) to subjective data such as emotions, mood and threat (Yannakakis, 2018, p. 37). The fuzzy sets allow for linear or non-linear formulas to be evaluated every *n* frames which make a decision based on the aggregated utility value of each function. Whilst BTs and FSMs only evaluate one decision at a time, Utility Based AI architectures will examine all options and assign a utility. Once all utilities have been assigned the highest utility value will be selected (Yannakakis, 2018, p. 37).
It is critical that for utility based AI should not be assigned fixed weights as it would take away the responsiveness and dynamicity of the system at runtime (Dill, 2015, pp. 23-24).

The advantage of Utility Based AI is that it better handles the different options where more than one action could be taken. Additionally, behaviour can be manually tailored by changing the heuristic functions of each utility which allows for designers to maintain more control over the behaviour that the AI will have (Dill, 2015).

A drawback for Utility AI is having no built-in mechanisms for handling uncertainty, this could be seen in games such as poker where the data in the hand is known but everything else is unknown (Świechowski, 2024, p. 1).

Neural Networks, Machine Learning & Deep Learning

Recently AI Researchers focus on breakthroughs in generative AI, a type of AI that creates new media such as text, audio and controversially images/video. Machine learning (ML) is a large part of this debate with lots of different techniques, but one of the most important learning algorithms is a Neural network (Cole Stryker, Eda Kavlakoglu, 2024).

<u>Neural Networks</u>
(Artificial) Neural Networks (ANNs) are a biological inspired approach for intelligence and machine learning which use neurons, (also referred to as nodes) to process information. (Yannakakis, 2018, p. 59). Every ANN involves layers of nodes, an input layer, one or more hidden layers, and an output layer (IBM, 2024).

Each layer accepts inputs from the prior layer, modifying it with built in weights before passing it through an activation function. The output from that function is then passed forward to the next layer, and the process repeats until the output layer is reached (Roberts, 2023, p. 264).

Machine Learning

Machine learning (ML) is a subset of AI that gives computers the ability to learn without being explicitly programmed. ML is defined as the capability of a machine to imitate intelligent human behaviour through and perform tasks in a way similar to how humans solve problems (Brown, 2021).

Within ML three main learning methods exist. Supervised, Unsupervised and Reinforcement Learning (IBM, 2024).
Supervised Learning is defined by the use of labelled datasets to train the algorithm to classify data and predict outcomes. As data is given to the model it adjusts the weights of the neurons until it provides an accurate result. This method is used for classification of spam emails in the real world and is used at scale (IBM, 2024).
Unsupervised Learning is providing the model with unlabelled data to find patterns or trends that a human may not be looking for, such as sales correlation between demographics (Brown, 2021). Benefits of this approach is that the model does not require monitoring to correctly identify information (IBM, 2024).
Reinforcement Learning is an approach similar to how humans learn using positive and negative reinforcement for actions. In reinforcement learning samples of positive results are usually not available which makes the process similar to trial and error as to find the correct result (Yannakakis, 2018).

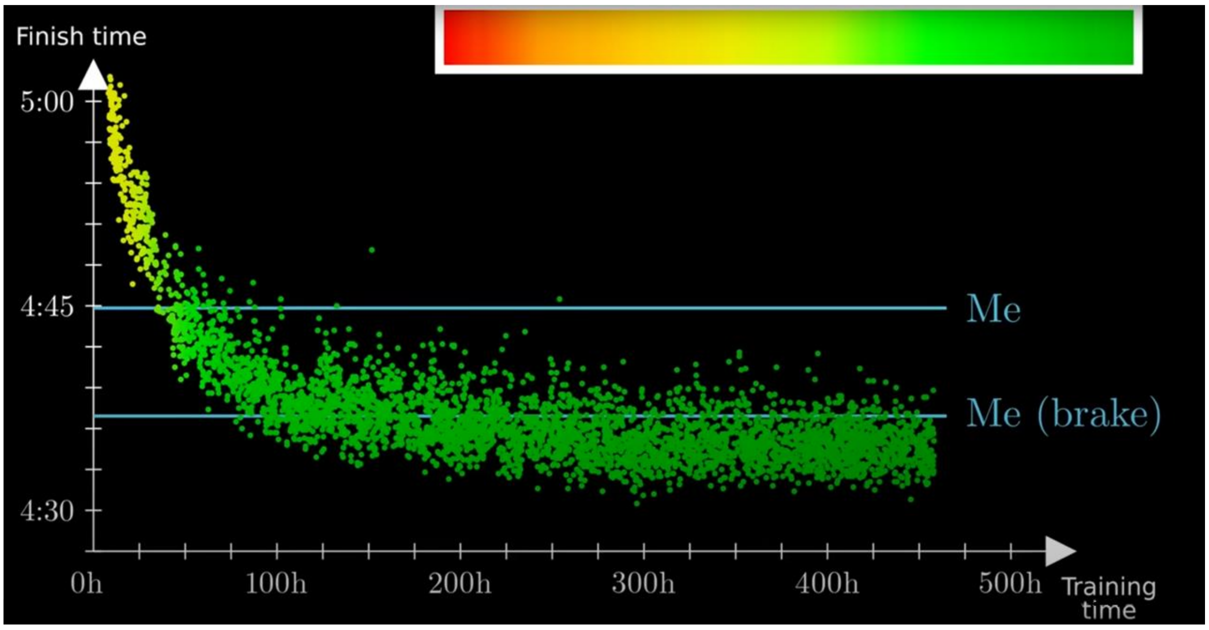
Deep Learning

"Deep Learning is a further subset within Machine Learning and is often thought of as scalable machine learning because it automates a lot of the feature extraction process and eliminates some of the human intervention involved. " (Aggarwal, 2024)
Deep Neural Networks (DNN) are similar to ANN with each layer building on the previous to optimise and predict by passing data forward, this method is called forward propagation (Jim Holdsworth, 2024). It is considered that a Neural Network is actually a DNN when it is comprised of three or more layers and outputs (IBM, 2024). Typically, within DNN, hundreds or thousands of layers are used to train the models (Jim Holdsworth, 2024).
With DL a large amount of computing power is required due to the volume of calculations that the Neural Network needs to complete. Graphical Processing Units (GPUs) are ideal for this due to the large volume of calculations and multiple cores with surplus memory available (Jim Holdsworth, 2024).
In comparison to ML where supervised learning models require structured data, DL models are capable of unsupervised learning by extracting characteristics, features and relationships that allow for accurate outputs using raw data (Jim Holdsworth, 2024).

Uses of Machine Learning and Deep Learning in Games
ANNs are also applicable to games by teaching an AI to play a game.
YouTuber Yosh, has a series wherein he attempts to use ML with the reinforcement model to train an AI to play Trackmania (Ubisoft, 2020). In his project he sets a time for the AI to compete with on a snaking track with no side barriers set by an experienced player. The AI learned using a reinforcement learning strategy and actions assigned a point value on success. After hundreds of attempts and at least 75 hours of learning, the AI beat the drivers time as seen in Figure 4 - Yosh

Trackmania Learning Time Graph. As training time increased, with 400+ hours of learning the AI were capable of beating the goal time by 7 seconds, this makes beating the AI time on the same track as a human near impossible (Yosh, 2023).

Figure 4 - Yosh Trackmania Learning Time Graph



The problem with the AI using ML is that even after 500 hours of training the AI was incapable of consistently producing results that the driving route on other maps. “The AI was less precise; it makes more mistakes and sometimes gets completely confused” (Yosh, 2023, p. 11:15).

Games using ANNs/MLs is not ideal because it slows developers adding new AI features in post launch updates, due to the AI will have to be partially or fully retrained to learn the new behaviour. In addition to the time it takes to train, “results can be unpredictable, which goes against the idea of games being an authored experience” (Roberts, 2023, p. 279).

The use of ANNs is to create repetitive tests trying different values for weights for each neuron to get the AI to ‘learn’ how to behave. This method of AI Decision making requires a lot of training to get an accurate result due to the iterative design in the learning, each test is given a rating on result desirability and the most successful result is used as a baseline for the next test. Once the AI is fully trained it should be able to work in the environment it was trained in accurately (Roberts, 2023, pp. 273-274). The drawback of ANNs using reinforcement learning is that procedural systems will need a longer model training time to be accurately trained due to the environment changing often. This means that an AI will have to learn the patterns of the environment generation as well as the best actions to take from within the environment which will inflate training time.

How can difficulty be scaled for an AI?

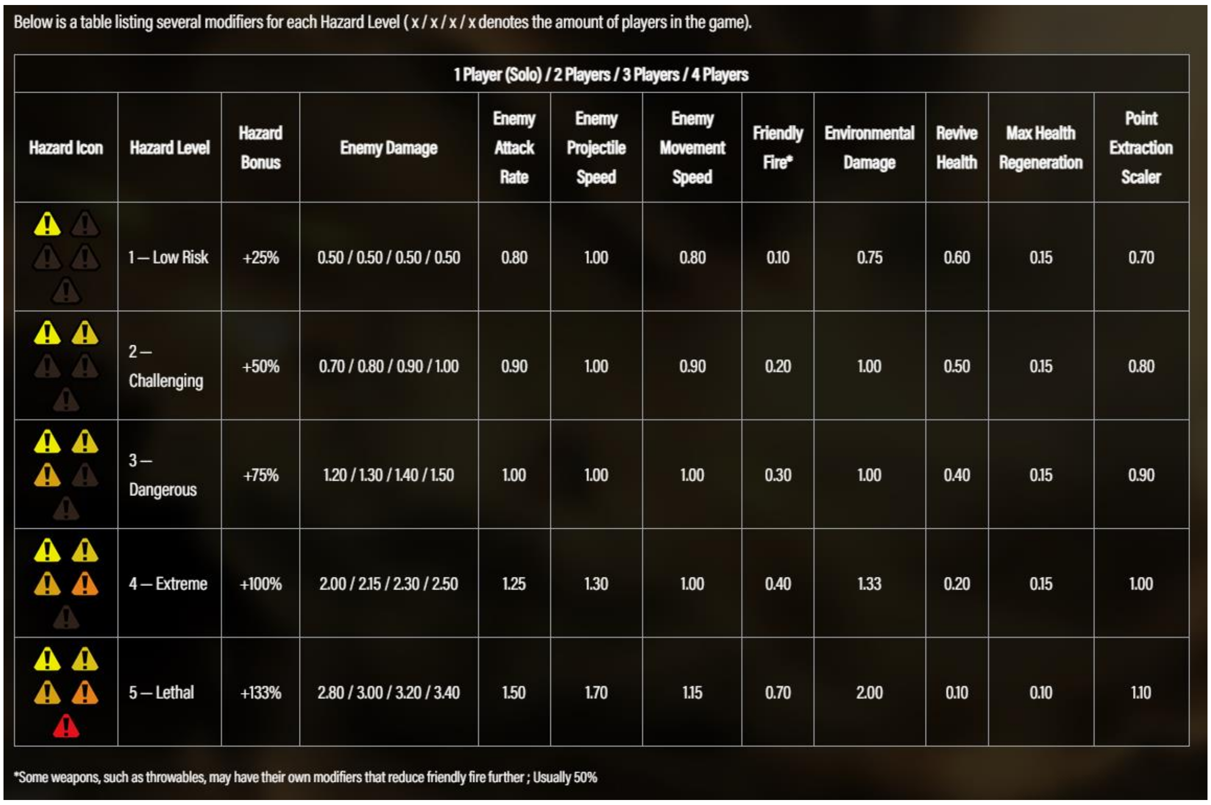
A game experience is considered satisfying or entertaining when it is difficult to beat, which may be true for a large portion of experienced gamers but when it comes to the casual market, The game is most entertaining when it is challenging yet beatable (Chin Hiong Tan, 2011). By having a game AI scale difficulty to match the performance of a player a game could be made for players of all skill levels.
To make games more difficult a developer can increase difficulty by making enemies more

aggressive or giving them more powerful weapons, additionally they can spawn more enemies or reduce player pickups. (Barriales, 2017)

Value Scaling

When scaling AI Difficulty, tweaking values is a useful way to quickly affect change as they would be readily available and will not require adding behaviour which will take longer to program. Many games tweak values between difficulties but a game that handles value tweaking well is Deep Rock Galactic (DRG) (Ghost Ship Games, 2018).
In DRG there are 5 hazard levels which act as difficulties. Hazard 1 is the easiest, and 5 the hardest difficulty. Each difficulty scales differently based on how many players are in the lobby. When it comes to scaling there are both player and enemy values to consider.
To scale DRG difficulty simple metrics such as attack speed, movement speed and damage are tweaked on the AI end which causes them to become far more difficult to overcome (Ghost Ship Games, 2024). As seen in Figure 5 - Deep Rock galactic Difficulties Chart*,* enemies become much harder to kill from their increased stats like damage, attack speed and movement speed. This small tweak creates a large change in the game feel because the survivability of the player decreases when paired with the rank and the number of enemies increasing.

Figure 5 - Deep Rock galactic Difficulties Chart



Below is a table listing several modifiers for each Hazard Level (x / x / x / x denotes the amount of players in the game).

			1 Player (Solo) / 2 Players / 3 Players / 4 Players								
Hazard Icon	Hazard Level	Hazard Bonus	Enemy Damage	Enemy Attack Rate	Enemy Projectile Speed	Enemy Movement Speed	Friendly Fire*	Environmental Damage	Revive Health	Max Health Regeneration	Point Extraction Scaler
⚠	1 – Low Risk	+25%	0.50 / 0.50 / 0.50 / 0.50	0.80	1.00	0.80	0.10	0.75	0.60	0.15	0.70
⚠⚠	2 – Challenging	+50%	0.70 / 0.80 / 0.90 / 1.00	0.90	1.00	0.90	0.20	1.00	0.50	0.15	0.80
⚠⚠⚠	3 – Dangerous	+75%	1.20 / 1.30 / 1.40 / 1.50	1.00	1.00	1.00	0.30	1.00	0.40	0.15	0.90
⚠⚠⚠⚠	4 – Extreme	+100%	2.00 / 2.15 / 2.30 / 2.50	1.25	1.30	1.00	0.40	1.33	0.20	0.15	1.00
⚠⚠⚠⚠⚠	5 – Lethal	+133%	2.80 / 3.00 / 3.20 / 3.40	1.50	1.70	1.15	0.70	2.00	0.10	0.10	1.10

*Some weapons, such as throwables, may have their own modifiers that reduce friendly fire further ; Usually 50%

Behaviour Frequency – Combat Accuracy.

Within shooter games AI will try to find a firing location to the target with a variety of other behaviours included such as taking cover or reloading in order to kill the player. With this basic behaviour set, a problem arises of multiple AI shooting and hitting the target at the same time, which instantly kills the player (Barriales, 2017).
A Tokenised System will address the one-hit kill by taking turns to shoot (and hit) the player by

granting an AI a token. This token determines whether the AI can deal damage. Within this system the AI can freely enter any shooting behaviours but will deliberately miss the shot and hit any location that is not the player if they do not have a token (Barriales, 2017).
Deciding how to distribute the Token is another measure that designers should be able to tweak, a token will have many factors to decide which agent gets it. This may include anything from time since last player hit, distance from target, or weapon dependant on how expansive the system is (Barriales, 2017).

Dynamic Difficulty

Difficulty is a critical factor in games and is challenging to set correctly. (Su Xue, 2016, p. 1) Dynamic Difficulty Adjustment (DDA) helps alleviate the issues of manually setting difficulty values and adding behaviour to scale difficulty. The use of DDA allows for maximising player engagement by measuring metrics such as wins, number of rounds played, duration played and session days and used tuning the game to make it easier or more difficult based on success (Su Xue, 2016, pp. 1-5).
Electronic Art's Inc (EA) Conducted a study using their match-three genre of games, Candy Crush Saga and Bejeweled, which tracked player engagement using the DDA method and found that they could optimise play using this method to maximise the player engagement (Su Xue, 2016).

How can group AI be balanced for a player

What metrics can be used for balancing games?

Balancing a game has no standardised metrics that a designer should use, but this does not mean metrics cannot be created that will help a designer balance the game based on the product they have created.
Slay the Spire is a 2D rogue-like turn-based card fighting game where the player uses cards and attack the enemies based on what cards are pulled from their deck.
Previously the developers have stated that their balancing approach, was to gather any relevant information at the end of a run and analyse the data. This ranged from enemies, cards, relics, events, player choices and time to complete (Giovannetti, 2019, p. 9:25). With these metrics it was not about aiming to make every card equally good but rather shooting for a more sensible win rate target.

Overwatch 2 uses metric tracking to enable the designers to tweak the live service experience. Statistics for each character are collected such as character selection, win rate, accuracy, eliminations and damage taken / given to name a few, this can be seen in game from the career history tab. These statistics are used to tweak each balance patch by interpreting the data.

How can AI make the game fair for the player?

AI Perception Balancing
Stealth games are known to have perception and awareness models for NPCs making it important that they are balanced correctly as to not break the immersion of the game world. It is believed that for a successful AI Perception system it must include the following.

<u>Fairness</u>
In a stealth game, getting detected will be a pivotal moment, it could be the difference between a player winning or failing the mission. Having perception models feel fair, like they are not cheating is a difficult task since the models themselves are subjective to what is fair. (Walsh, 2015)

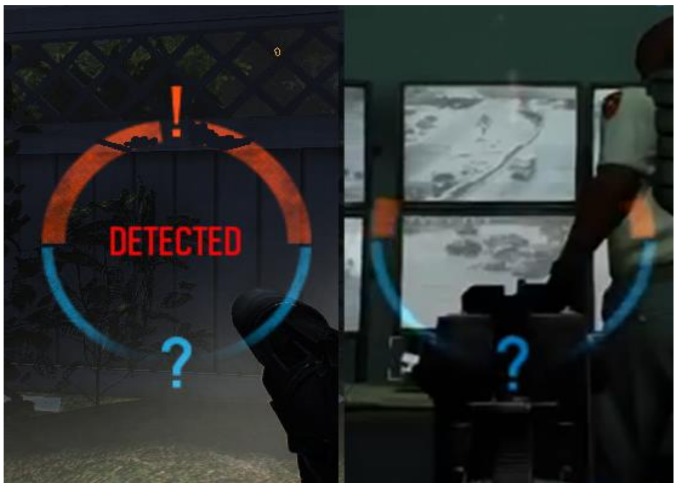
<u>Consistency</u>
The player must be able to perceive the AI's behaviour so that they can strategise and plan, this

means creating a somewhat AI predictable. This contrasts to all humans being unpredictable and varying reactions between individuals (Walsh, 2015).

Feedback

Whilst the player needs to be able to predict the AI's behaviour, there needs to be affordance to help the player understand what is happening, this can be seen within Payday 2 having a radial awareness meter that ticks up until the player is detected by the AI. Once detected, the radial wheel adds the words detected onto the screen as well as having an exclamation mark at the top as seen in Figure 6 - Payday 2 Detection Meter, this directly tells the player that they are spotted and need to act.

Figure 6 - Payday 2 Detection Meter



Perceived Intelligence

To have an AI that does not achieve a level of intelligence that is believable, is not satisfying to beat. This requires having an AI not necessarily be dumb but does not also require it to be smart. Balancing intelligence is about having plausible behaviour to convince a player that the character the AI is controlling would act in that way (Walsh, 2015). This means that if an AI hears a gunshot or has the player hit them, the expected reaction is to enter an alert / combat state, if the AI was to continue with the current state, the AI would not be convincing and looks inept.

Limiting AI Knowledge

Having an AI know everything feels like cheating and could lead a player to get frustrated. Many developers will attempt to limit the knowledge of an AI to prevent this frustration. A great example of this would be Alien Isolation.

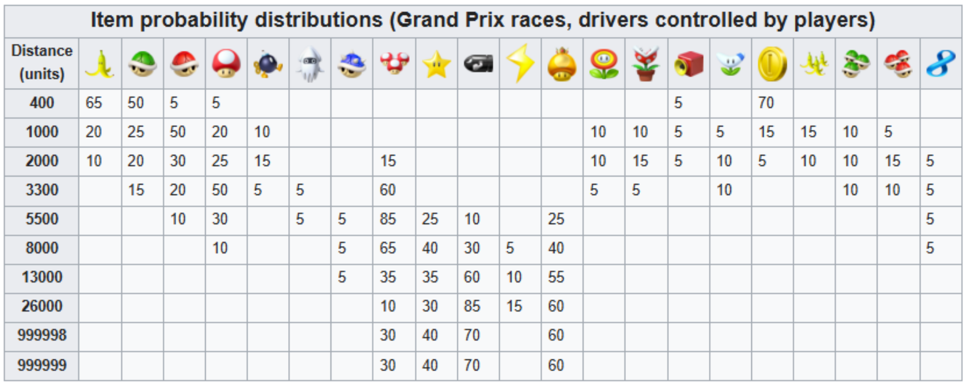
Alien Isolation's AI is held in high regard and still talked about many years after its release for the usage of a twin AI controller known as a director system. This system would feature an overarching AI that knows all the information about the world, for example, it would know the position of points of interests, hiding spots and most importantly the player. With this information it could direct the AI towards areas that would allow the steering behaviour agent (The alien) to find and hunt the player. By obfuscating this data, the alien must hunt for the player which creates a more intense experience as the alien is not mimicking a search state by pretending to know the player's location (Thompson, 2016).

Removing Behaviour
In ‘The Last of Us’ during development, the buddy AI ‘Ellie’ had trouble in combat as she would become “a killing machine”. The developers did not want to change the damage output of the weapons that Ellie had, so altering firing values and hit behaviour was the route they chose to take. The behaviour was tweaked so that when the player is not looking at the enemy Ellie was shooting at, the buddy did not need to do damage but still appeared to be in combat. When the targeted agent was visible again, the buddy would be required to do damage. This helped keep players immersed within the game rarely realising that the buddy was not doing damage (Dyckhoff, 2015).

Mario Kart – Item Probability

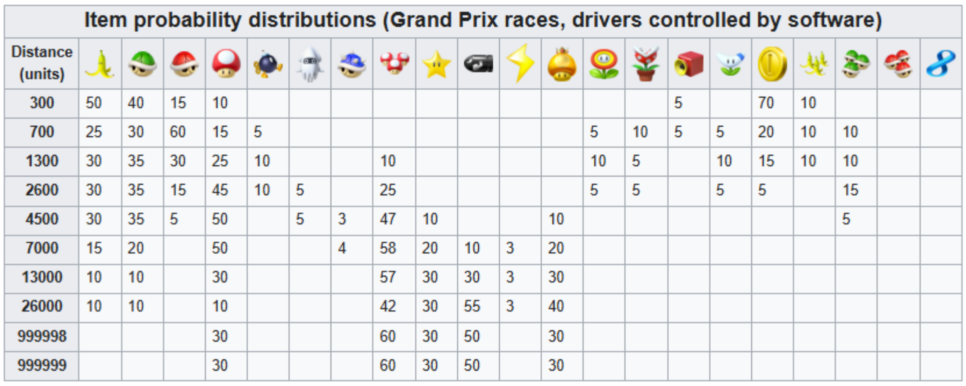
Mario Kart, although not being an exclusive example of AI balance, explores game balance through in-game items probability. Mario Kart 8 Deluxe has 23 Items which the drivers have access to which have different abilities (Nintendo, 2024).
As the player gets further behind the first-place driver, more powerful items are likely to be selected in the loot table. Items such as the Star or Bullet bill are the most effective items due to the ability to speed up the player and forcefully control the driver’s movement to increase their podium position. As seen in Figure 7 Community researched probability distributions for Mario Kart 8 Deluxe (Anon., 2024), Some items such as the red shell trio are disabled for an AI driver, this is because these items are powerful. Given the situation of a driver in 2nd could potentially receive a red shell trio, this gap between 1st and 2nd will be easily shortened by the first-place driver being stunned by shell after shell with minimal defence from their own items.

Figure 7 Community researched probability distributions for Mario Kart 8 Deluxe (Anon., 2024)



Item probability distributions (Grand Prix races, drivers controlled by players)

Distance (units)																					
400	65	50	5	5											5		70				
1000	20	25	50	20	10								10	10	5	5	15	15	10	5	
2000	10	20	30	25	15			15					10	15	5	10	5	10	10	15	5
3300		15	20	50	5	5		60					5	5		10			10	10	5
5500			10	30		5	5	85	25	10		25									5
8000				10			5	65	40	30	5	40									5
13000							5	35	35	60	10	55									
26000								10	30	85	15	60									
999998								30	40	70		60									
999999								30	40	70		60									



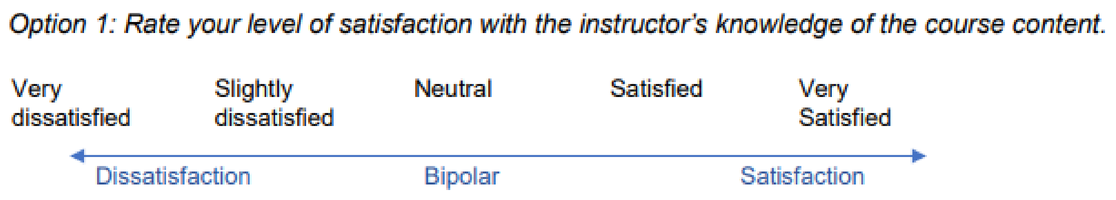
Item probability distributions (Grand Prix races, drivers controlled by software)

Distance (units)																					
300	50	40	15	10											5		70	10			
700	25	30	60	15	5								5	10	5	5	20	10	10		
1300	30	35	30	25	10			10					10	5		10	15	10	10		
2600	30	35	15	45	10	5		25					5	5		5	5		15		
4500	30	35	5	50		5	3	47	10			10							5		
7000	15	20		50			4	58	20	10	3	20									
13000	10	10		30				57	30	30	3	30									
26000	10	10		10				42	30	55	3	40									
999998				30				60	30	50		30									
999999				30				60	30	50		30									

Research Methodology

Quantitative research asks for an individual to fill form(s) with questions asking for opinion-based answers with a numerical output after a test has been completed. This scale be plottable data that can be used for visual graphics to help analyse data and reinforce statements.
Quantitative research allows for lots of participants and allows for data to be represented in real-time due to the data having set parameters. This allows for bar graphs/ line graphs and other graphics to help convey better than a table. Research is anonymous so users are likely to answer more honestly in comparison to qualitative options like focus groups that are more interactive and possibly has a name or ID attached to their result for further feedback.
Another common example of quantitative testing is a Likert scale; this survey type asks specific questions that rank response levels to assess levels of satisfaction. This places a focus on variable answers that are not directly Yes or No questions but rather a gradient of options (University, 2021). As seen in Figure 8 - Likert Scale, the options are not phrased "Are you satisfied" but rather a rating level, this allows for a wider range of results to more accurately measure the customer's satisfaction.

Figure 8 - Likert Scale



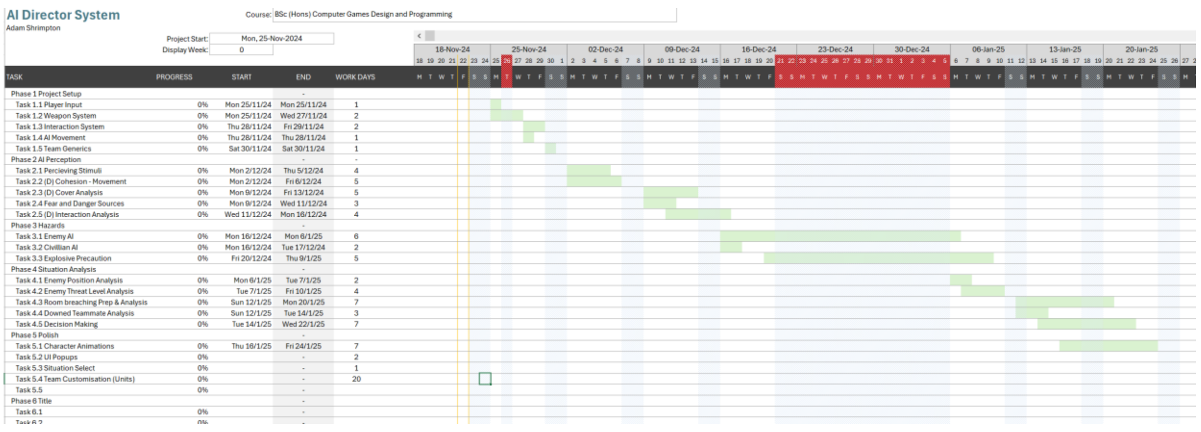
Qualitative research is requesting an individual to answer a question with non-numerical data, this is an opinionated evaluation to a given question which can be used to help further improve development of a product or service from detailed user feedback. This data is harder to plot to on a graph because categorising data which is alphabetical requires analysing unlike numerical data which can be added quickly.
Qualitative examples include open questionaries that ask for detailed opinions on a topic, this can be a question like "what did you like about your stay with us?" (Survey Monkey, n.d.), this allows for a wider expression of opinion than a quantitative survey but is not as easily plottable as results must be interpreted first.
Focus groups are also another option to qualitative research, this is gathering small groups to gather and have a free-flowing discussion on the topic, this can give a researcher insight into areas that the questionnaires may not cover and allows for further questions based on previous answers given (Survey Monkey, n.d.).

Product testing is a research methodology that allows for collection of both quantitative and qualitative data about the consumption, preferences or reaction to a product. By using this approach quality assurance testing can be conducted and results can be compared against user opinions to improve the product.

The research methodology chosen to test the artefact is a mixed approach of qualitative and quantitative feedback through the product testing route. The aim of using product testing in this project is to use data to identify and improve the AI to become more realistic. Questions will be asked about any issues that have come up during testing and how the user would like the product to be improved. The numerical data provided by the system will be provided to help analyse the AI performance on a statistical level and identify tweaks that can be created to alter the product.

Gantt Chart

Figure 9 - Gantt Chart



AI Director System
Adam Shrimpton

Course: BSc (Hons) Computer Games Design and Programming

Project Start: Mon, 25-Nov-2024
Display Week: 0

TASK	PROGRESS	START	END	WORK DAYS
Phase 1 Project Setup			-	
Task 1.1 Player Input	0%	Mon 25/11/24	Mon 25/11/24	1
Task 1.2 Weapon System	0%	Mon 25/11/24	Wed 27/11/24	2
Task 1.3 Interaction System	0%	Thu 28/11/24	Fri 29/11/24	2
Task 1.4 AI Movement	0%	Thu 28/11/24	Thu 28/11/24	1
Task 1.5 Team Generics	0%	Sat 30/11/24	Sat 30/11/24	1
Phase 2 AI Perception			-	-
Task 2.1 Percieving Stimuli	0%	Mon 2/12/24	Thu 5/12/24	4
Task 2.2 (D) Cohesion - Movement	0%	Mon 2/12/24	Fri 6/12/24	5
Task 2.3 (D) Cover Analysis	0%	Mon 9/12/24	Fri 13/12/24	5
Task 2.4 Fear and Danger Sources	0%	Mon 9/12/24	Wed 11/12/24	3
Task 2.5 (D) Interaction Analysis	0%	Wed 11/12/24	Mon 16/12/24	4
Phase 3 Hazards			-	
Task 3.1 Enemy AI	0%	Mon 16/12/24	Mon 6/1/25	6
Task 3.2 Civilian AI	0%	Mon 16/12/24	Tue 17/12/24	2
Task 3.3 Explosive Precaution	0%	Fri 20/12/24	Thu 9/1/25	5
Phase 4 Situation Analysis			-	
Task 4.1 Enemy Position Analysis	0%	Mon 6/1/25	Tue 7/1/25	2
Task 4.2 Enemy Threat Level Analysis	0%	Tue 7/1/25	Fri 10/1/25	4
Task 4.3 Room breaching Prep & Analysis	0%	Sun 12/1/25	Mon 20/1/25	7
Task 4.4 Downed Teammate Analysis	0%	Sun 12/1/25	Tue 14/1/25	3
Task 4.5 Decision Making	0%	Tue 14/1/25	Wed 22/1/25	7
Phase 5 Polish			-	
Task 5.1 Character Animations	0%	Thu 16/1/25	Fri 24/1/25	7
Task 5.2 UI Popups	0%		-	2
Task 5.3 Situation Select	0%		-	1
Task 5.4 Team Customisation (Units)	0%		-	20
Task 5.5	0%		-	
Phase 6 Title			-	
Task 6.1	0%		-	
Task 6.2	0%		-	

Scope

The scope of the project, provided the student can keep to the schedule, should be completed to a satisfactory level and will be achievable. The setup of the project is a large chunk of time due to base project structure for AI setup is required and other accessory features need to be created such as team states, perception and weapon setup.

The core scope of the project is limited to.

- AI movement in a cohesive manner focusing on formation-like movement with minimal errors, this should be developed early on with iterations.
- Token firing systems focusing on difficulty scaling allowing for a director to tweak token assignments balancing damage outputs based on the number of enemies in sight.
- System for agents to reach a safe location in cover based on position of enemies.
- Suitable testing environment for each test

Additional supportive features are planned such as visualisations for damage output values, automatic test recording and animations to show AI state. This is achievable and should be considered halfway through development to allow for the results to be used within testing.

Results and Findings

The Project criterion to determine if the created artefact is a success, is as follows:

- AI is capable of scaling damage based on a difficulty value (Easy, Medium, Hard)
- The AI can move through an environment using a formation like positioning
- The AI can correctly select cover in an environment
- The Director AI can move through an environment doing the above-mentioned criteria fighting enemies without getting lost faster than the single agent solution.

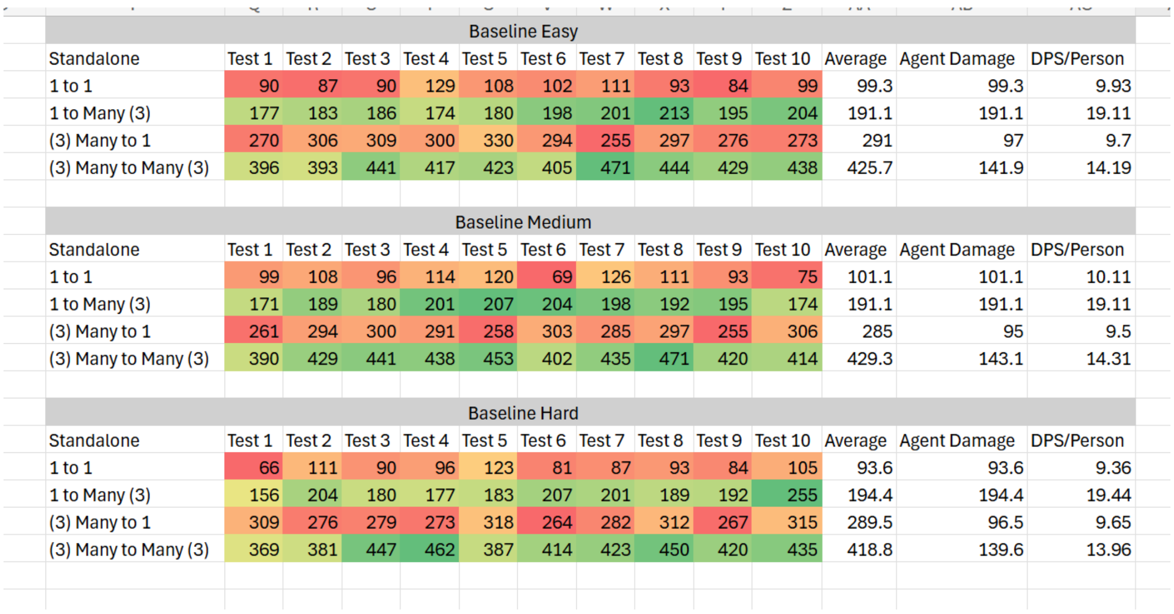
Firing Tests

Test Method
The method used for testing firing damage output will be a listener actor recording how much damage each unit receives as units fire at them, this value is then totalled and outputted as a float value for total damage received. Each test lasts 10 seconds and continues from the previous test. The 10 tests will last 1 minute 40 seconds. Every test will be conducted at the same time to allow for quicker data collection and recording.

The difficulty for each test is set before to ensure correct values are outputted for each difficulty group. The only failure condition for these tests is an anomalous result (which will be removed from data set) or AI failing to fire at the target. In the event of a failure a note will be recorded and the artefact tested again.

Baseline / Standalone Test Results

Figure 10 - Standalone Firing Tests



Baseline Easy													
Standalone	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average	Agent Damage	DPS/Person
1 to 1	90	87	90	129	108	102	111	93	84	99	99.3	99.3	9.93
1 to Many (3)	177	183	186	174	180	198	201	213	195	204	191.1	191.1	19.11
(3) Many to 1	270	306	309	300	330	294	255	297	276	273	291	97	9.7
(3) Many to Many (3)	396	393	441	417	423	405	471	444	429	438	425.7	141.9	14.19

Baseline Medium													
Standalone	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average	Agent Damage	DPS/Person
1 to 1	99	108	96	114	120	69	126	111	93	75	101.1	101.1	10.11
1 to Many (3)	171	189	180	201	207	204	198	192	195	174	191.1	191.1	19.11
(3) Many to 1	261	294	300	291	258	303	285	297	255	306	285	95	9.5
(3) Many to Many (3)	390	429	441	438	453	402	435	471	420	414	429.3	143.1	14.31

Baseline Hard													
Standalone	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average	Agent Damage	DPS/Person
1 to 1	66	111	90	96	123	81	87	93	84	105	93.6	93.6	9.36
1 to Many (3)	156	204	180	177	183	207	201	189	192	255	194.4	194.4	19.44
(3) Many to 1	309	276	279	273	318	264	282	312	267	315	289.5	96.5	9.65
(3) Many to Many (3)	369	381	447	462	387	414	423	450	420	435	418.8	139.6	13.96

The standalone enemies are programmed without a token system in comparison to the director system which guides how many hits an AI should have. The standalone AI defaults to a random Boolean value for whether the AI should hit the target or not. This means the expected hit results will be 50% of the total damage potential.

In the data collected, the more units in the world the greater the damage output is. The “1 to many” tests has unintended results. Expected results should display that the AI miss enemies at the same rate as the 1 to 1 test because the AI has the same Boolean hit potential, but the results show a significantly higher damage output, almost twice as much. The reason for this outcome not aligning with the 1-1 tests damage output is that the AI shoots to the left or right of the target, but it does not account for If another target is in that offset location. This offset results in the AI hitting the agent next to the target accidentally, this is an unintended effect of the system which requires patching, this could be patched through recalculating the output vector or just nullifying the damage if not the intended target.

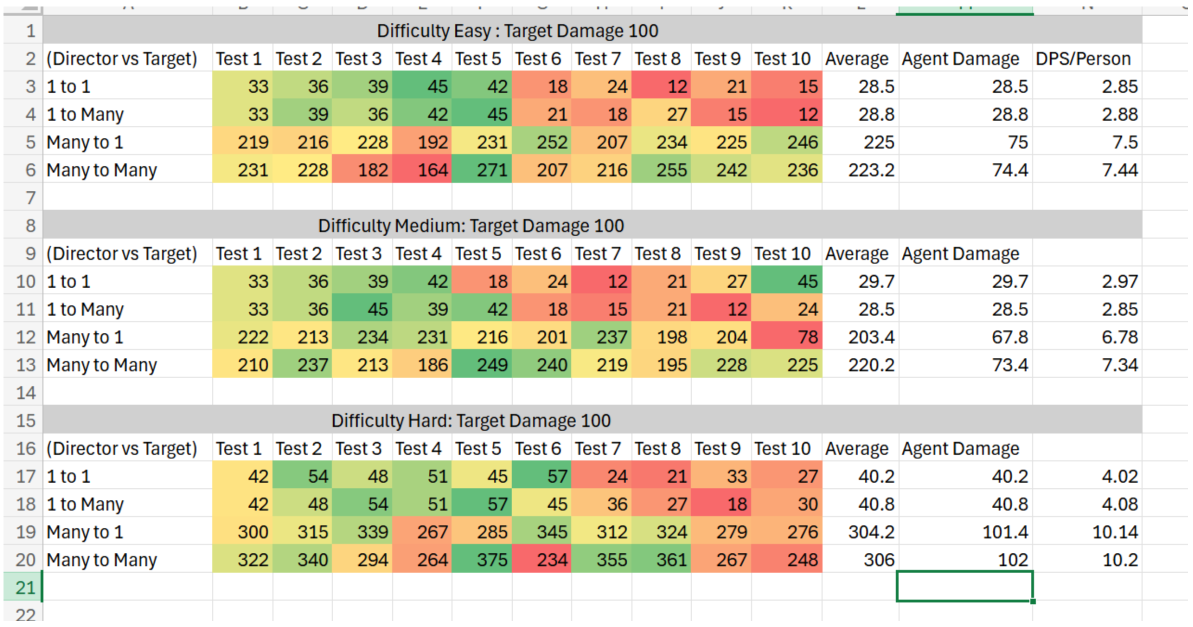
The system keeps a consistent level of damage between all the “X to 1” tests. With an overall damage average of **9.7Hp/s** against a single target, the time to kill is around 10 seconds per AI.

Based on personal experience from multiplayer shooters, this is a very high time to kill (TTK) regardless of game and this value should be adjusted to the game feel.

The “many to many” tests yielded an unexpectedly lower DPS than the “1 to many”, upon further investigation it was shown that the group would opt to choose the target that was furthest left due to the spawn order. This would create the effect of shooting to the left and hitting nothing whereas in the 1 to many the AI would be shooting the centre target which would then hit the targets to either side. Provided that AI were hitting the same targets its predicted that the results would be similar.

Director Token System Tests

Figure 11 - Director Token Firing Tests



Difficulty Easy : Target Damage 100													
(Director vs Target)	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average	Agent Damage	DPS/Person
1 to 1	33	36	39	45	42	18	24	12	21	15	28.5	28.5	2.85
1 to Many	33	39	36	42	45	21	18	27	15	12	28.8	28.8	2.88
Many to 1	219	216	228	192	231	252	207	234	225	246	225	75	7.5
Many to Many	231	228	182	164	271	207	216	255	242	236	223.2	74.4	7.44

Difficulty Medium: Target Damage 100													
(Director vs Target)	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average	Agent Damage	
1 to 1	33	36	39	42	18	24	12	21	27	45	29.7	29.7	2.97
1 to Many	33	36	45	39	42	18	15	21	12	24	28.5	28.5	2.85
Many to 1	222	213	234	231	216	201	237	198	204	78	203.4	67.8	6.78
Many to Many	210	237	213	186	249	240	219	195	228	225	220.2	73.4	7.34

Difficulty Hard: Target Damage 100													
(Director vs Target)	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average	Agent Damage	
1 to 1	42	54	48	51	45	57	24	21	33	27	40.2	40.2	4.02
1 to Many	42	48	54	51	57	45	36	27	18	30	40.8	40.8	4.08
Many to 1	300	315	339	267	285	345	312	324	279	276	304.2	101.4	10.14
Many to Many	322	340	294	264	375	234	355	361	267	248	306	102	10.2

Director tests yield different results to the baseline tests; A Significant difference in DPS per person in comparison to the average DPS per person in the standalone can be observed with the director tests averaging is 50% less for the single target tests on easy. The trend displayed in the results of the token system, is that the more people, the higher total damage the group provides. This is an ideal result because a player should expect to receive more damage with a higher enemy count, whereas the baseline tests show, the more AI targets there are the greater the damage output. This result, whilst the potential to be logistically correct, does not provide an accurate result for testing damage output values with targets that do not disappear.

The director tests show that each curve used for calculating when a token is assigned should be adjusted to have a lower time to kill (TTK), all difficulties appear to have a very low DPS for a single agent, it would take a single unit 35 seconds on easy to eliminate a target, this means that the token curve needs to be improved to have a lower token threshold, this in turn would increase the damage for a single unit.

The Token firing system can be seen to be more effective at changing the damage output when there are more units to assign tokens to. On Medium (Excluding anomalous test result 10) the average builds upon the damage from the easy difficulty and again for the Hard difficulty.

Movement Tests

Movement tests are conducted by timing an AI moving through 3 different sections of an environment. A test will consist of 6 AI agents being controlled by either a director system or running standalone behaviour trees. This is to test the effectiveness of group cohesion and how they can navigate through obstacles.
Section 1 of the Movement tests is an open area which with an opening into the next section 20 Meters from point A, this is created to test formations in an open environment with no interference, this also helps to see if an AI can manage basic cohesion without breaking formation, results here are expected to maintain cohesion with no issues.

Section 2 of the movement tests is an enclosed area 22 meters long with a width of 2 Meters. This is to simulate a corridor situation to test how a formation would mould to match the shape of the room. Houses and other establishments that a SWAT team would have to enter will have elements of tight area combat which makes having a correct formation important otherwise squad members are potentially in danger. Results here are expected to potentially break by losing formation but shape together quickly.

Section 3 of the Movement tests is an area that has obstacles designed to break a squad formation by having obstacles in the way. The idea behind this section is that an environment like a house will have tables, chairs and other items that will impede the movement of a team and could impede the groups or an individual's path.

There is only one failure condition for a movement test, if none of the AI agents in the team can continue, then a failure is recorded.
If a lone AI stops moving and gets separated from the group or stops moving, the counter for formation broken will be increased. The lower number of times the formation has been broken, it would be fair to assume the better performing the AI cohesion is.
Another metric being recorded in movement tests is time. The AI navigates a full lap from point A to B and then back to point A, upon reaching Point A, the time will be recorded. The test will run 3 Laps in succession to get an average time which should help determine performance of each system.

Figure 12 - Movement Test Route

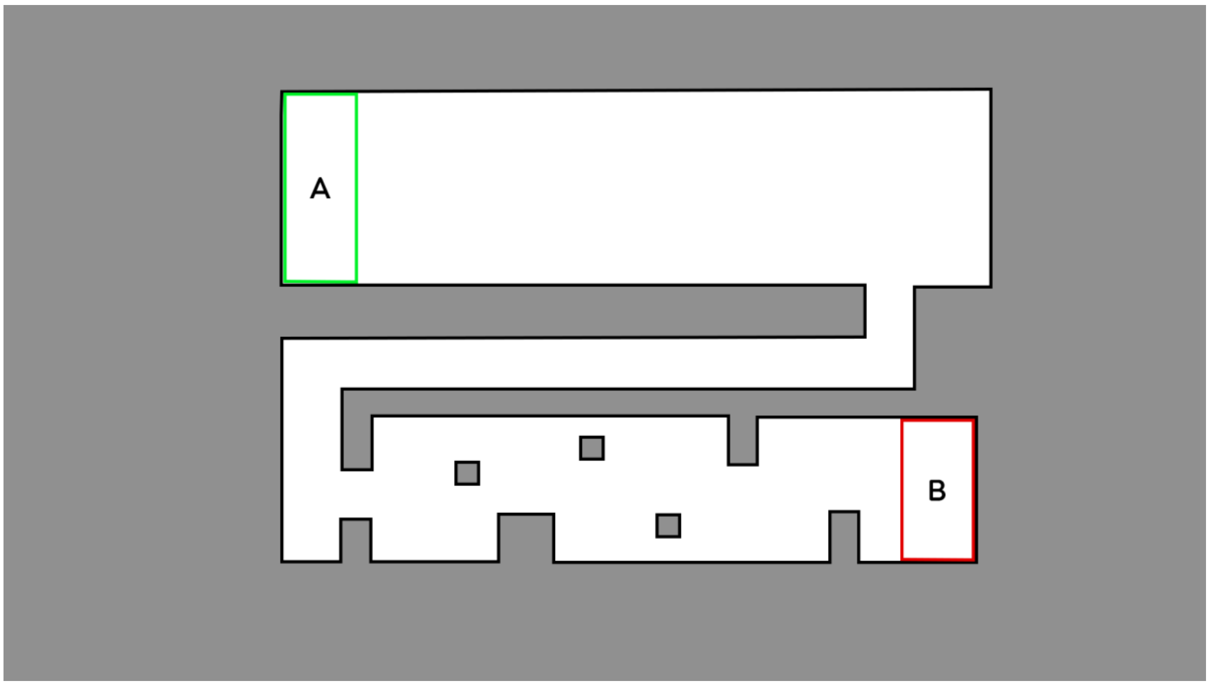
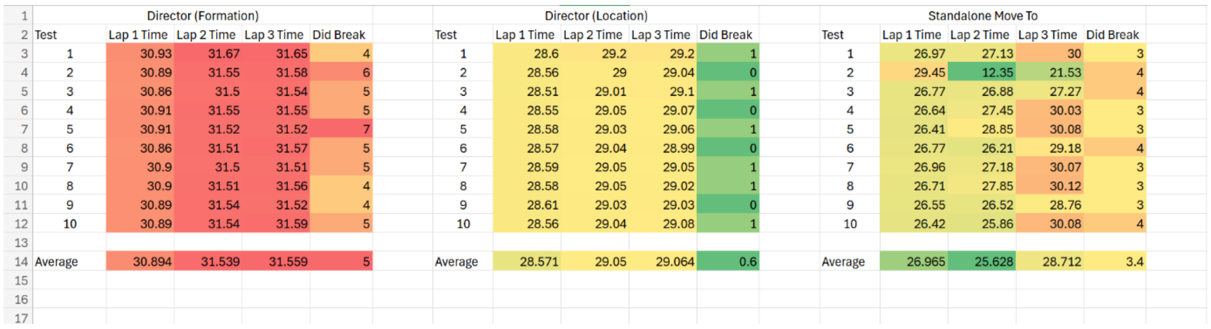


Figure 13 - Movement Test Results



	Director (Formation)					Director (Location)					Standalone Move To			
Test	Lap 1 Time	Lap 2 Time	Lap 3 Time	Did Break	Test	Lap 1 Time	Lap 2 Time	Lap 3 Time	Did Break	Test	Lap 1 Time	Lap 2 Time	Lap 3 Time	Did Break
1	30.93	31.67	31.65	4	1	28.6	29.2	29.2	1	1	26.97	27.13	30	3
2	30.89	31.55	31.58	6	2	28.56	29	29.04	0	2	29.45	12.35	21.53	4
3	30.86	31.5	31.54	5	3	28.51	29.01	29.1	1	3	26.77	26.88	27.27	4
4	30.91	31.55	31.55	5	4	28.55	29.05	29.07	0	4	26.64	27.45	30.03	3
5	30.91	31.52	31.52	7	5	28.58	29.03	29.06	1	5	26.41	28.85	30.08	3
6	30.86	31.51	31.57	5	6	28.57	29.04	28.99	0	6	26.77	26.21	29.18	4
7	30.9	31.5	31.51	5	7	28.59	29.05	29.05	1	7	26.96	27.18	30.07	3
8	30.9	31.51	31.56	4	8	28.58	29.05	29.02	1	8	26.71	27.85	30.12	3
9	30.89	31.54	31.52	4	9	28.61	29.03	29.03	0	9	26.55	26.52	28.76	3
10	30.89	31.54	31.59	5	10	28.56	29.04	29.08	1	10	26.42	25.86	30.08	4
Average	30.894	31.539	31.559	5	Average	28.571	29.05	29.064	0.6	Average	26.965	25.628	28.712	3.4

Baseline tests

Unreal Engines default "Move to Location" node for AI movement is a quick solution to AI group movement. This method yields results that are within an acceptable margin consisting of a moderate error rate, averaging 3.4 errors every 27 seconds. In the baseline tests, a single AI would be stuck in the middle of section 2 during the second lap, this would cause the stray agent to run from point B back to point A faster than the rest of the group. This error drastically reduced the lap time of the standalone tests and are considered somewhat unreliable. With the default Reciprocal Velocity Obstacles avoidance (RVO), the AI can suitably move through the environment with minimal collision in section 3. The AI do not have any structured formation which is clearly visible throughout the tests which is undesirable for a working AI group.

Director Formation

The current implementation of the director formation system uses a mixed approach of Offset vectors and EQS systems to generate a breadcrumb like path that the AI must follow. If the offset is inside a collider of a static object (walls, obstacles), an EQS test is ran to find the nearest desired pathfinding location.

The system results show that the time to complete the laps is slower than the other two tests by 2-3 seconds per lap. The reduced speed would have been an expected result of the formation system needing to run different queries. The unintended result of the system is that the error rate is much higher than both the Director location and the standalone AI movement systems. Out of the three methods this is the least desirable result.

Director Location
Using an agent controlled by the Director AI, this system assigns other units the invisible controlled pawn as the pathfinding location. This method manages the final location for each agent and refreshes the path every behaviour tree refresh which allows for many corrective calculations. The error rate of this system is the best of the three with an average error rate of 0.6 times every 3 laps which is an improved result from standalone tests as errors are 5.6x less likely to occur.
The lap time despite not being the fastest is still close to the standalone test with a nominal difference of about 1-3 seconds. The lap time is quick because there are not the positional requirements like in the formation system which is an alternative effect of the system but still looks cohesive due to the AI still navigating close to one another.

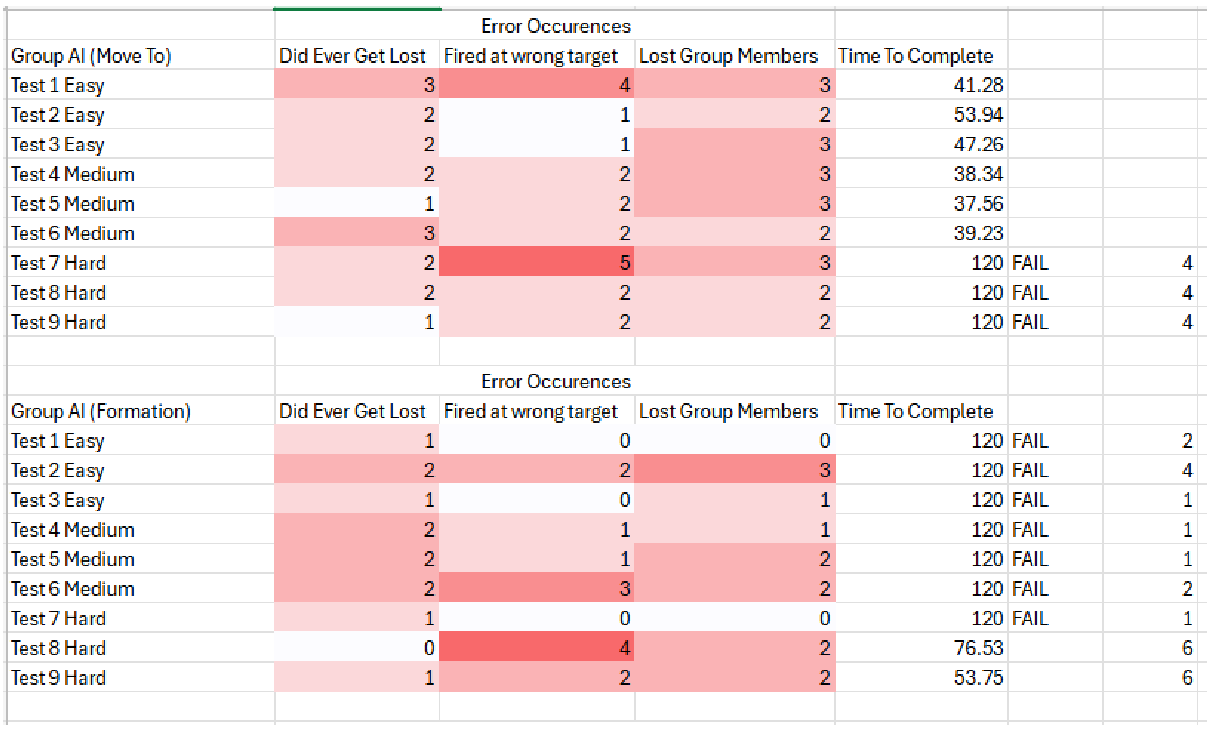
Cover Tests

The Cover tests consist of an enemy appearing in random locations near the AI group and the agents solving a location to stand at using an EQS query. The time for all agents to reach the desired location will be recorded as well as the effectiveness of the cover. This is determined by the proportion of limbs sticking out, if the limb would be hit easily then it would add to the failure counter.
Note* This test was removed due to the time constraints of the project, the cover system query is generated but not included in the final product.

Environment Tests

The Environment tests consist of 5 group agents navigating through a simulated environment, this environment is not based on real-life structures but has elements to help test the AI in different ways.
Initially the fork in the road is to test which AI the group should move to eliminate. The intended result is the closest to the agent, this is so that the environment is cleared from front to back.
Next is a sightline test, an AI is placed in the distance with another AI closer to the group behind a wall, this requires the group to choose between navigating towards the closest target or shooting at the currently visible target. The Intended result is for the sightline test is to choose the closest target that is visible.
And finally, there is a section with multiple AI targets to test the cohesion of the group firing skills, if the AI do not cohesively shoot at the same target, then the test gets an error added to the wrong target counter. Each failure is on a per agent basis which means if 4 AI are shooting at the wrong target +4 is added to the error total.

Figure 14 - Environment Test Results



		Error Occurences				
Group AI (Move To)	Did Ever Get Lost	Fired at wrong target	Lost Group Members	Time To Complete		
Test 1 Easy	3	4	3	41.28		
Test 2 Easy	2	1	2	53.94		
Test 3 Easy	2	1	3	47.26		
Test 4 Medium	2	2	3	38.34		
Test 5 Medium	1	2	3	37.56		
Test 6 Medium	3	2	2	39.23		
Test 7 Hard	2	5	3	120	FAIL	4
Test 8 Hard	2	2	2	120	FAIL	4
Test 9 Hard	1	2	2	120	FAIL	4
		Error Occurences				
Group AI (Formation)	Did Ever Get Lost	Fired at wrong target	Lost Group Members	Time To Complete		
Test 1 Easy	1	0	0	120	FAIL	2
Test 2 Easy	2	2	3	120	FAIL	4
Test 3 Easy	1	0	1	120	FAIL	1
Test 4 Medium	2	1	1	120	FAIL	1
Test 5 Medium	2	1	2	120	FAIL	1
Test 6 Medium	2	3	2	120	FAIL	2
Test 7 Hard	1	0	0	120	FAIL	1
Test 8 Hard	0	4	2	76.53		6
Test 9 Hard	1	2	2	53.75		6

Director Location

In results for the director location, the simulation was tested 3 times on each difficulty. In the current implementation the trend (from non-failure results) that occurs is that with difficulty, the time to complete decreases, this is because the damage output is higher allowing for the AI to clear rooms faster. In the projects current implementation, the combined features fail on the Hard difficulty. This data set consistently errored which rendered the results as a failure. During observation of the Director system and through debugging the behaviour tree, it was discovered the AI would get stuck choosing between states despite having information to transition into a state this left the director unable to move to the next location. This means that the director location system is a partial failure because it cannot fully adapt to the dynamic difficulty and requires further patching to get more consistent working results.

Director Formation & Standalone Tests

The director formation has consistent failure with the group of agents during testing stopping at targets with no clear reasoning, this unwanted behaviour and comes from a direct logic failure rather than a methodology failure and thus renders the test results defunct. Additional iterations require more defined decision to produce effective results.

The Standalone tests are erroneous for a different reason to the Director Formation results. AI cannot distinguish between team and enemy when searching for a new target despite explicitly filtering the characters by team, this problem results in any ai groups larger than 2 from seeking to a single target. This fault in the system prevents the AI tests from being conducted the same as the director making testing an impossibility and requires a core foundation for enemy detection.

Discussion and Analysis

The investigation into AI Director systems and supporting AI techniques demonstrates success in a small range of techniques that an AI Director system can be used to manage agents cohesively and effectively. The system overall was unable to perform and meet the objectives of this project, with the objective of hazard perception being removed from scope due to time constraints and others not fully meeting success criterion. Although the project was a failure the literature researched allows for an in depth understanding of techniques and how to improve upon the system created with potential for use as a framework in other projects.
Some AI programming techniques proved effective whilst others may not have been the correct choice for the system where another could have been used to get more consistent results.

AI Perception systems prove effective in targeting enemy players successfully providing a result that an entity can use to fire at. This system was effective for perceiving stimuli based on team data but with the current implementation has some flaws on the decision making from the information provided, notably by assigning targets based on incoming information. The current implementation could be improved by checking the currently active stimuli when receiving a new stimulus, this allows for the target chosen to be assigned for the agent receiving based on information around. Alternatively, a Utility Based AI decision approach may be used to determine the desired target. Both solutions implemented correctly would create a reactive AI that can respond to new threats quicker.

Using Token Systems to manage the damage output of a group AI has proven to be an effective method. The token system meets the success criterion, adapting to changing difficulty based on a factor, in this instance it was the difficulty of the game. (Barriales, 2017) in his token system design does not detail on how the system they implemented is calculated for providing a token, but with an introspective into how games should balance this implementation calculates currently enemies perceived against those perceiving and uses the difficulty value against a curve to calculate how many shots need to be fired before a hit can be granted. Though this evaluation works a separate evaluation of time elapsed, or damage dealt could be used to balance the system which may work better for a dynamic scaling system.

Behaviour trees are a core component in the project and were the primary driving force behind the controlled agents, this was chosen so that both non directed and directed AI could use the same behaviour tree and have a comparison to show the effectiveness of the director. Behaviour trees proved to be effective as a method for decision making in AI, but the inefficient use of behaviour trees result in the artefact cause AI to break and no longer move. A state-machine like structure is implemented within the behaviour tree which has hindered the scalability that the behaviour tree system is designed for as described by (Yannakakis, 2018).
Within this behaviour tree system the results show that the AI breaks frequently because the state gets stuck, this is an issue with state machines rather than the behaviour tree. State machines require specific conditions to transition which the project failed to meet, this could have been improved by increasing code quality and patching out bugs that prevent transitioning.

Environment Query systems, although an ineffective use in the formation system, have proved some potential in use for other systems untested such as the intended cover system. The creation of the cover system was completed but not implemented due to time constraints. As shown in Figure 14 - Environment Test Resultsand Figure 15 - Custom EQS Generator, the generated locations from the AI could be used to mark locations for the AI to stand. Once additional context is added to the scene such as agent location and other enemies this could change results and a utility-based system for cover may be a better fit.

Figure 15 - Custom EQS Generator

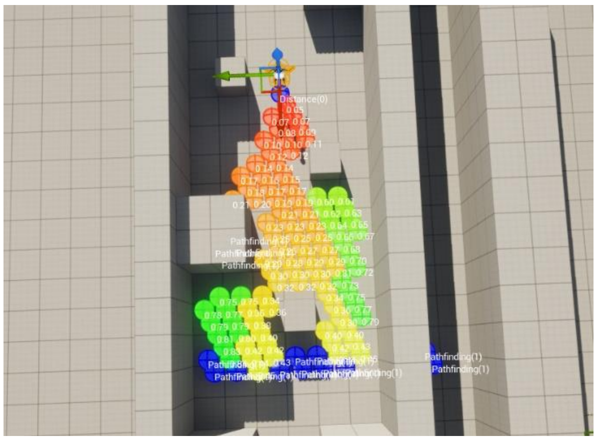
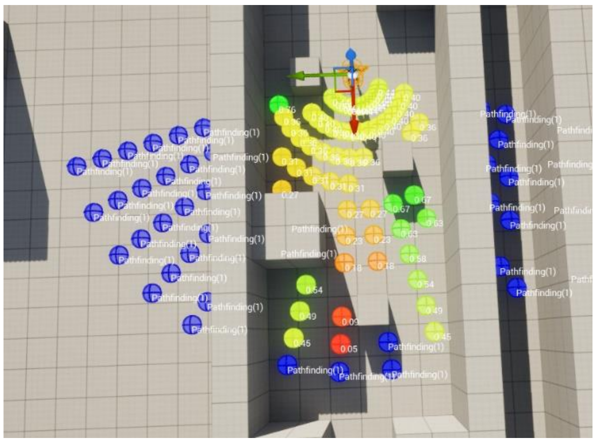


Figure 16 - Cone EQS Generator



Critical Evaluation

In this project the student has over-scoped and managed their time insufficiently, this led to delays and falling behind schedule which affected the success of the project. Due to personal reasons the student also was unable to work on the artefact as much as they wished resulting in a scale back of the artefact to an unsatisfactory standard. Had the student created contingency plans, the artefact and testing could have been more in depth focussing on a core component of AI cohesion such as the movement or the token system rather than a broad range of AI concepts.
With a single core focus, time and resources would ~~have~~ not have been spread thin resulting in an ~~broken~~ artefact with ~~and~~ insufficient results. Despite the artefact not producing the desired results, effort to implement a structure that can be further developed in the future has been created using the type object pattern and readable code which allows for new systems to be easily implemented. The testing provided has no ~~depth~~ little depth beyond surface results as very few tests have been conducted per feature, ~~and~~ key test variables such as movement speed or weapon damage were not altered to create a wider range of results. In addition to the limited results provided, the results were not iterated upon which could have potentially yielded different results. The weapon firing tests could have been created as a dynamically adjusted feature by implementing a scale that the user could have dragged using a UI element to adjust the damage. The knowledge gained through these limited results does allow for further development opportunities for ways to balance AI and

token systems. Token systems notably will be useful for other systems in different game mechanics such as AI spawning or movement opportunities for an RTS game.

Given the chance to improve the project, other implementations such as steering behaviours mixed with a pathfinding approach could be tested as an extra result. This may help with the formation issues that the Director has currently whilst still pathfinding towards a point, alternatively a better formation system without an EQS may ~~have~~ be~~en~~ best due to the errors present in the current implementation, further development is required.

The Artefact has limited user interaction which has not proved anything for a player versus AI interaction like stated in objective four “How can group AI be balanced for a player?~~”~~” but does include situations against AI to mimic what competing against the system would look like. Testing against AI doesn’t give a good benchmark for how players would react due to the critical thinking skills that humans possess unlike AI which follow pre-programmed rules, this is why player testing should be a focus if development of the artefact continues.

The research methodology chosen would have been an excellent way to improve the artefact allowing for feedback to iterate and gathering plottable data that would have been used to optimise the systems given enough time and better management strategies. In the results a pure quantitative approach was used and tested based on the programmer’s view which does not allow for a refined product, tests should have been conducted using other people. Additionally, the notes from the programmer should have been recorded so that the iterative process is displayed, and development results can be displayed.

Conclusion

The project failed to work cohesively when implemented and did not meet the success criteria for the solution provided due to poor time management and issues within the artefact. However, with gained insight on various AI technologies, it is believed that with more time and a shifted focus into optimising decision-making code structure, most problems could be resolved to provide more accurate results. ~~-~~Individual components of the project show that the use of a director system could be used to manage some aspects of AI cohesion for weapon systems and navigation of many agents but were unable to complete the task as intended due to errors in the artefact~~individually but fail to work when implemented together~~.

~~Although the resulting failure of the project, t~~Through development and research of the project a wide variety of AI technologies have been discovered and new interests found such as Machine Learning within video games. Overall, a greater understanding of AI technologies and the difficulties faced when creating in depth systems gives a new perspective into AI programming, with ~~and~~ a desire to continue developing, ~~and~~ refining skills ~~to~~ and to create a robust AI with a potential to further a ~~my~~ career in this~~at~~ area. This project has added more than just AI knowledge to my skillset but also has shown me areas for improvement in life when it comes to time and resource management and to look at problems through more than one solution.

If this project was repeated a focus on EQS systems to gather information about the environment would be a key element of research and development because of the potential it can provide for AI decision making. This would allow the intended objective of real-time hazard perception to be compared to a stimuli perception system and test results against the EQS which could make for interesting optimisations.
Additionally, in recreation of the project, ~~I would like to~~ using~~e~~ the ‘Lyra’ template from unreal engine to start as a solid point for basic AI handling would be wise, because the setup of the project took longer than necessary drawing time away from crucial development of AI systems and this template provides that base.

Bibliography

Aggarwal, L., 2024. *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks | IBM.* s.l.:IBM.

Anon., 2024. *Mario Kart 8 item probability distributions.* [Online]
Available at: https://www.mariowiki.com/Mario_Kart_8_item_probability_distributions
[Accessed 11 December 2024].

Barriales, S. O., 2017. Using Your Combat AI Accuracy to Balance Difficulty. *Game AI Pro 3.*

Brown, S., 2021. *Machine Learning, Explained | MIT Sloan.* [Online]
Available at: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained
[Accessed 12 November 2024].

Chin Hiong Tan, K. C. T. A. T., 2011. *Dynamic Game Difficulty Scaling Using Adaptive Behaviour-Based AI.* s.l.:IEEE.

Codemasters, 2010. *F1.* s.l.:Codemasters.

Coenen, S., 2016. *Steering Behaviours in C# and C++.* s.l.:s.n.

Cole Stryker, Eda Kavlakoglu, 2024. *What is Artificial Intelligence (AI)? | IBM.* [Online]
Available at: https://www.ibm.com/topics/artificial-intelligence
[Accessed 12 November 2024].

Dill, K., 2015. Dual Utility Reasoning. *Game AI Pro 2 : Collected Wisdom of Game AI Professionals,* pp. 23-26.

Dyckhoff, M., 2015. Human Enemy AI in The Last of Us. *Game AI Pro 2: Collected Wisdom of Game AI Professionals,* pp. 431-442.

Epic Games, 2024. *AI Perception in Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community.* [Online]
Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/ai-perception-in-unreal-engine
[Accessed 11 November 2024].

Epic Games, 2024. *Environment Query System.* [Online]
Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/environment-query-system-overview-in-unreal-engine

Fray, A., 2013. Context Steering - Behavior-Driven Steering at the Macro Scale. *Game AI Pro 2 : Collected Wisdom of Game AI Professionals,* pp. 183-193.

Ghost Ship Games, 2018. *Deep Rock Galactic.* s.l.:Coffee Stain Studios.

Ghost Ship Games, 2024. *Difficulty Scaling - Official Deep Rock Galactic Wiki.* [Online]
Available at: https://deeprockgalactic.wiki.gg/wiki/Difficulty_Scaling

Giovannetti, A., 2019. *'Slay the Spire': Metrics Driven Design and Balance.* s.l.:GDC Vault.

Haapanen, L., 2021. *The Environment Query System and the Spatial Query System used for AI-agency in games, a comparison,* s.l.: University of Jyväskylä.

IBM, 2024. *AI vs Machine Learning vs Deep Learning vs Neural Networks | IBM.* [Online]
Available at: https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-

networks
[Accessed 12 November 2024].

IBM, 2024. *What is a Neural Network? | IBM.* [Online]
Available at: https://www.ibm.com/topics/neural-networks
[Accessed 12 November 2024].

IBM, 2024. *What is Machine Learning (ML) | IBM.* [Online]
Available at: https://www.ibm.com/topics/machine-learning
[Accessed 12 November 2024].

jb-dev, 2018. *Steering behaviours: seeking and arriving.* [Online]
Available at: https://www.gamedev.net/blogs/entry/2264855-steering-behaviors-seeking-and-arriving/

Jim Holdsworth, M. S., 2024. *What is Deep Learning.* [Online]
Available at: https://www.ibm.com/topics/deep-learning
[Accessed 12 November 2024].

Nintendo, 2024. *Mario Kart 8 Deluxe for Nintendo Switch - Official Site.* [Online]
Available at: https://mariokart8.nintendo.com/
[Accessed 11 December 2024].

Reynolds, C. W., 1999. *Steering Behaviors For Autonomous Characters,* s.l.: s.n.

Roberts, P., 2023. *Artificial intelligence in games.* Florida: Boca Raton.

Su Xue, M. W. K. A. A. Z., 2016. Game Design Document Format For Video Games With Passive Dynamic Difficulty Adjustment. *Register Jurnal Ilmiah Teknologi Sistem Informasi.*

Survey Monkey, n.d. *Qualitative Survey Types.* [Online]
Available at: https://uk.surveymonkey.com/mp/conducting-qualitative-research/
[Accessed 20 February 2025].

Świechowski, M., 2024. *Fuzzy Utility AI for Handling Uncertainty in Video.* s.l.:IEEE.

Thompson, T., 2016. *AI and Games - YouTube.* [Online]
Available at: https://www.youtube.com/@AIandGames

Ubisoft, 2020. *Trackmania.* s.l.:Ubisoft.

University, J. M., 2021. *Guidelines for Likert Survey Questions,* s.l.: James Maddison University.

Walsh, M., 2015. Modeling Perception and Awareness in Tom Clancy's Splinter Cell Blacklist. *Game AI Pro 2 : Collected Wisdom of Game AI Professionals,* pp. 313-326.

Xiao Cui, H. S., 2011. A*-based Pathfinding in Modern Computer Games. *IJCSNS International Journal of Computer Science and Network Security,* p. 2.

Yannakakis, G. N., 2018. *Artificial Intelligence and Games.* s.l.:s.n.

Yosh, 2023. *Training an unbeatable AI in trackmania.* s.l.:s.n.

Zielinski, M., 2013. Asking the Environment Smart Questions. *Game AI Pro,* pp. 423-426.

Appendices

No appendices to date.

Appendix 1: Gantt Chart

Appendix 2: Title