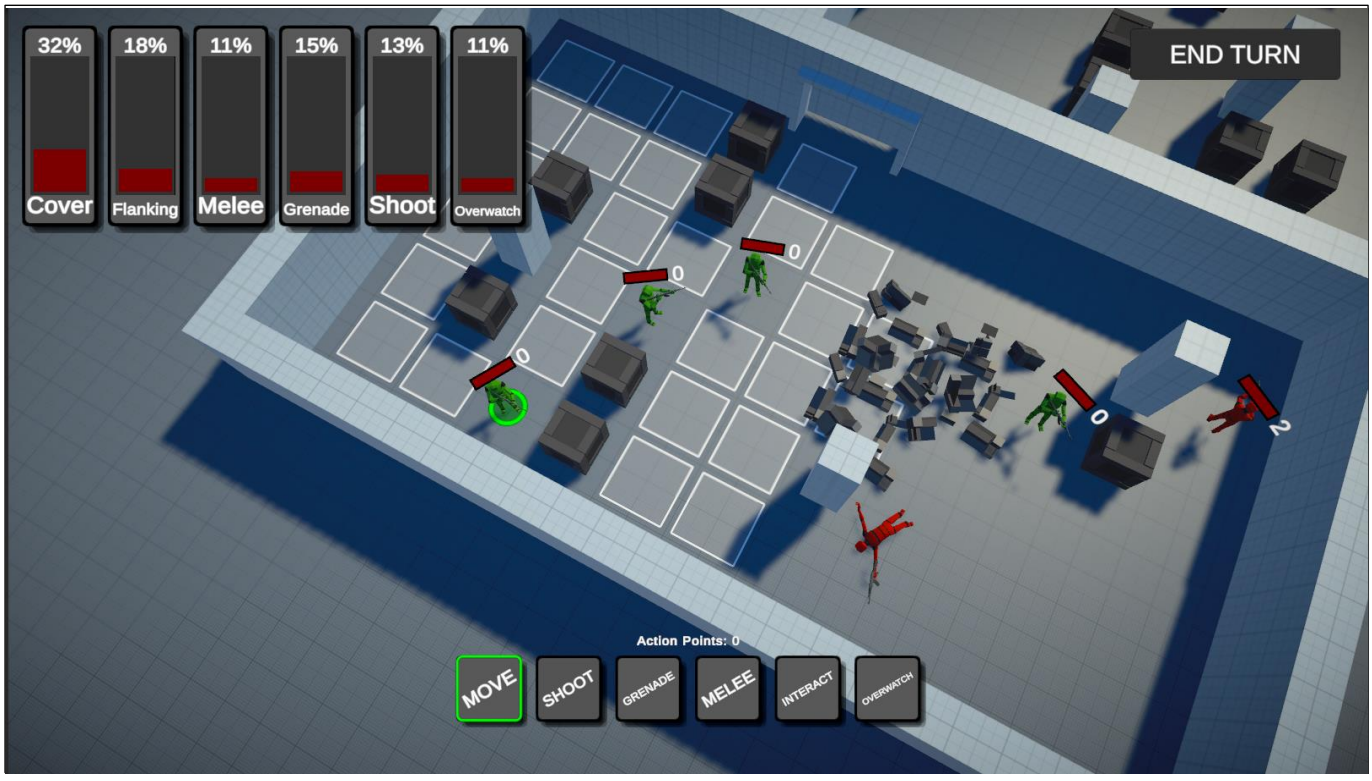
Comprehensive Creative Technologies Project: An AI Combat System that Analyses Player Combat Patterns to Predict and Counter Players Actions

Samuel Crockett
Supervisor: Mark Palmer

Department of Computer Science and Creative Technology
University of the West of England
Coldharbour Lane
Bristol BS16 1QY



Abstract

This report discusses how the use of a system similar to a Goal Orientated Action Planning (GOAP) system has been implemented in a real time strategy game to create an AI system that records the actions performed by the player and calculates an overall probability of the likeliness that the player will perform an action. These probabilities are then used to determine which actions the enemy AI will perform to best counter the players strategies. The report details how this system was created, what optimizations could be made to improve the system and overall determines if this system would be viable for use in enemy AI system in games.

Keywords: Goal Orientated Action Planning, GOAP, Artificial Intelligence, Real Time Strategy, Behaviour Recognition

Brief biography

Having been a fan of real time stratergy for a long time I have always been interested in coming up with different stratergies and tactics in order to overcome different challenges. However I found that with some games the AI only had set patterns of behaviour that they would perform. And this set behaviour became very easy to recognize and counter after a short period of time, leading to a less enjoyable experience. From this repetative behaviour I came up with the idea of an AI system that would react to actions that the player takes and produce different stratergies to try and defeat the player. This would hopefully result in a more interactive, engaging experience.

How to access the project

The project files can be found at:

The project video can be found at:

---

1. Introduction

Many games implement a combat system. For some games combat may only be used in a limited way, but for others AI combat systems can be the main aspect and the driving force for the success and popularity of the game. Especially given the increased expectations from players for better and more realistic AI in games. There are two key aspects to making an AI system that can keep players motivated to play and these are effectiveness and diversity (Szita, Ponsen and Spronck, 2009), AI must be effective so that it can provide a challenge and be believable. It also needs to be diverse so that it can create several different situations that prevent the gameplay from becoming repetitive.

The main issue is that factors often conflict with each other as the need to be effective tends to result in a small number of definite actions, limiting the diversity that the system can produce. The aim of this project would be to create an AI system that analyses the combat habits and patterns of the player and produce a variety of different effective solutions so that the player consistently faces new challenges and does experience repetitive combat.

An example of this was implemented in a First-Person shooter horror game called F.E.A.R (2005) (see Fig 1) where AI opponents analysed the players position and formed tactics to counter and defeat the player. Some examples of this within the game include the enemies communicating with each other to lay suppressive fire so others can move in, only leaving cover when a threat occurs to them and blind firing if they have no better position. (Orkin 2006).

The aim of this project is to create a system that can accurately analyse the players combat habits and behaviours. Then using that analysis, determine the best use of actions to counter the player. The aim is to provide a variety of different

Figure 1 – combat scenario in F.E.A.R (Orkin 2006)

outcomes and strategies based on the players actions to ensure the system does not feel repetitive while also providing a good level of balance.

Project Objectives:

- Create a system that can analyse player combat habits and patterns and learn to counter them
- Produce a system that can find efficient tactics based on a range of different actions
- Produce a balanced system that does not feel too powerful or too weak.

Key Deliverables:

- A working prototype scene that demonstrates a working version of the system.

2. Literature review

The first stage of this research was to find similar systems that have already been researched and created. After some initial research a paper about a game called F.E.A.R was found (Orkin 2006). The paper examines the use of a goal orientated action planning (GOAP) system to produce different enemy tactics based on player position

and surrounding environment. From this Additional research into GOAP systems and Adaptive AI was carried out.

2.1 GOAP System

Orkin, 2003) Applying Goal-Oriented Action Planning to Games, A paper that describes the basic theory behind the GOAP system (see Fig 2)The paper also explains how this system could be implemented within games and how it would work within certain types of games, (see Fig 3) which shows the basic planning that happens in the GOAP system in the context of a shooting game. From this more research was conducted to determine the strengths and weaknesses of this type of system

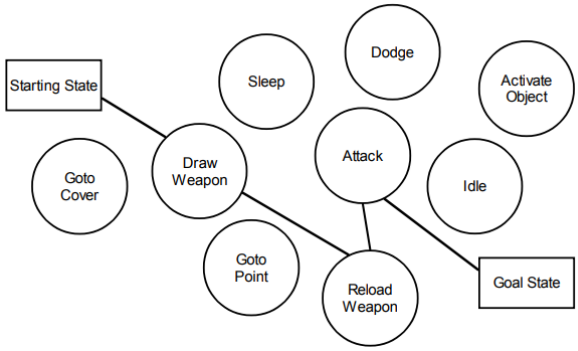


Figure 2. example of GOAP (Orkin, 2003)

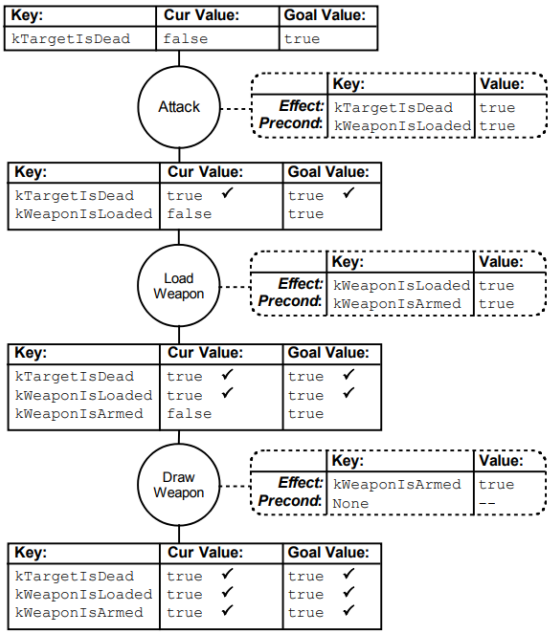


Figure 3 – example of the planner system used in a GOAP System (Orkin, 2003)

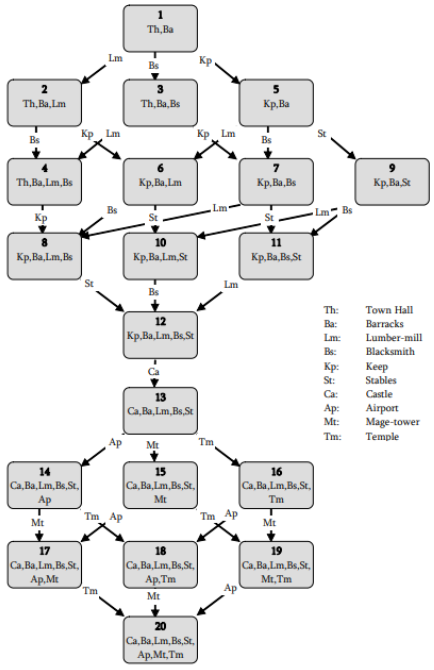
(Wenniger, 2008) GOAP, A paper that further examines the GOAP systems strengths and weaknesses. While also examining the integration of a GOAP system with different forms of AI techniques such as A* and Finite State Machines. The main strength of the GOAP system is that it uses actions that have preconditions and effects that change the world state allowing it to produce multiple plans for the same goal and respond to dynamic changes within the world. The main weakness of the system is that if it is not designed and implemented correctly then a similar effect could be produced using other systems such as behaviour trees which would result in better performance but not produce the overall goal for project. This paper was helpful in further understanding the GOAP system and learning about different potential AI systems that could be implemented while also highlighting the weaknesses of the system. This showed that more research was needed into the GOAP system to fully understand it and research into other systems to see if there are better options to achieve the goal of this project.

(Long, 2007) This paper goes into more Depth about each stage of the GOAP system by explaining what happens in each stage of the GOAP process and how they connect with each other to form a system. The paper also shows different implementations of the GOAP system, such as a standalone system and one implemented with a Finite State Machine.

2.2 Adaptive AI

While the research into GOAP systems was promising it did highlight potential weaknesses within the system. To avoid these potential issues and any bias in the research outcomes other adaptive systems for AI were investigated.

(Ponsen, 2004) A paper that looks at dynamic scripting for its AI. The system works with rules and states with each state having a set of possible corresponding rules. The system then starts randomly assigning rules to the first state until a change occurs and when it does a new state is created (see Fig 4). While this system could produce a wide range of different results and actions to achieve a goal, the randomness of which the rules are applied to a current state and the way the game states are constantly updated could result in issues with performance.



Rule type	Count
Build Rules (e.g. Build new barracks)	12
Economy Rules (e.g. Train more workers to harvest resources)	4
Military Rules (e.g. Attack the enemy)	25
Research Rules (e.g. Upgrade your weapons)	9

Figure 4 – Example of Dynamic scripting where each box represents a state, and each arrow is a transition forming a new state after a rule has been applied (Ponsen, 2004)

(Szita, 2009) A paper that investigates dynamic scripting further by introducing macro actions into the system. The paper examines the difference in performance between a standard dynamic scripting system and one with macro action introduced to determine the most effective way of using dynamic scripting.

The research from these papers has provided another potential solution to the problem that does not include implementing a GOAP system. However, based on the nature of this game and what this project is aiming to achieve, the GOAP system will be used for this project

2.3 Behaviour Analysis

(Kabanza, Bellefeuille and Bisson, 2010) This paper investigates the algorithmic challenges behind behaviour recognition and developing a system that addresses those challenges. The paper produces a solution to these issues with the use of a Hostile intent capability and opportunity recognizer (HICOR) system and explains how it is used for behavioural recognition.

(Doirado and Martinho, 2010) A paper that examines player behaviour and predictability. The paper examines the theory of intention and applies it to a model of intention (see Fig 5). They determine that intent can be split into three sections, movement, target and intent. Then they devise a way in which each component of an action can be detected within a game.

Figure 5 – Model of Intention (Doirado and Martinho, 2010)

3. Research questions

There were two main questions regarding the further research that needed to be conducted. These questions were:

- Can a system be created that accurately reads the players behaviour patterns?
- Can this system be integrated into an AI system to determine what actions would be best to counter potential player behaviour?
- Can an AI system that accurately predicts players actions be balanced to produce a challenging yet fun and fair experience within a game?

4. Research methods

With the development of new technology and an increased demand for more advanced and realistic AI within games, extensive research has been carried out into this topic. Due to the amount of research that has already been carried out, Secondary Research will be the main method used within this report. Primary research will not be used because the data that they could produce would not be the information that is needed to carry out this project and the information will most likely not be as accurate or as well developed as previous research. The main forms of secondary research consisted mainly of academic articles and journals that have covered different designs and implementations for the different aspects of the proposed system.

While a large amount of research will be examined, Qualitive analysis will be the focus of this document. With the amount of data that is available and the amount of variation between different findings, qualitive analysis will be able to determine the best parts of all the different designs and approaches to each of the different systems that will be implemented in this project. Whereas with quantitative research, the results from the findings may be to varied or outdated resulting in difficulty analysing and implementing later in the project.

5. Ethical and professional principles

There were no ethical concerns identified with this project. There were no external human participants involved at any stage of the project. Both the research and development carried out within the project was performed professionally to ensure that all secondary sources of information have been fully referenced.

Also, the aim of this report and project is to demonstrate a way of designing an AI system for real time strategy games. Given this fact there are no ethical concerns in the way of potentially effecting communities including other developers in a negative way.

6. Research findings

Based on all the research that has been found, key papers have been identified for each of the main aspects that needed to be researched.

6.1 AI Combat / Counter System

(Long, 2007) The paper discusses the GOAP design and how it functions. The system has 6 key elements to it, the world state, agent action and goal sets, actions, action containers, goals, and a goal manager.

The world state is a collection of world state symbols. World state symbols are values that are set and are used for testing against other world states. (see Fig 6) The world state is a collection of 3 Booleans which are used to check which actions can be completed and what effect the actions have, for example in the reload action the isWeaponLoaded bool is set to true when completed.

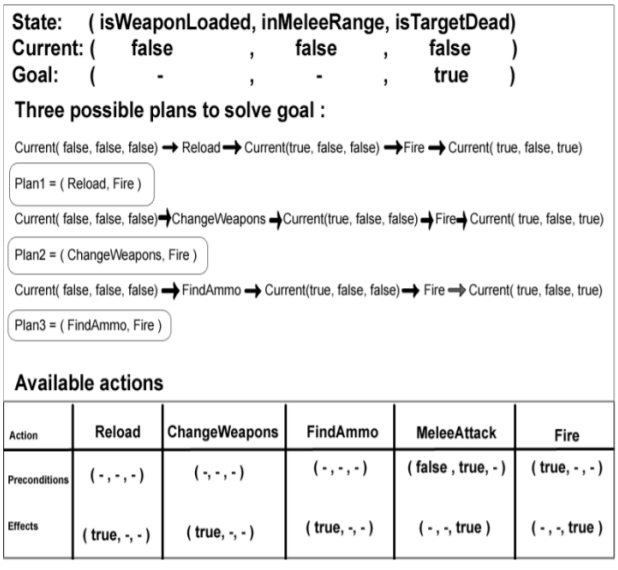


Figure 6 – World State for an agent in the level

The Agent action and goal sets are the list of actions that the individual agent can use, and the goals sets are the different goals that the agent can be set.

The Actions are the pre-set manoeuvres that the agent can make that are used to achieve the agents goals. For an action to trigger all the preconditions that are set in the world state must be met. If more than one action has all of its preconditions set then the action with the higher priority, based on the current goal, will be executed. For instance, for the attack action (see Fig 7) to trigger the isWeaponLoaded bool must be true and this triggers the isTargetDead bool in the world state to be true.

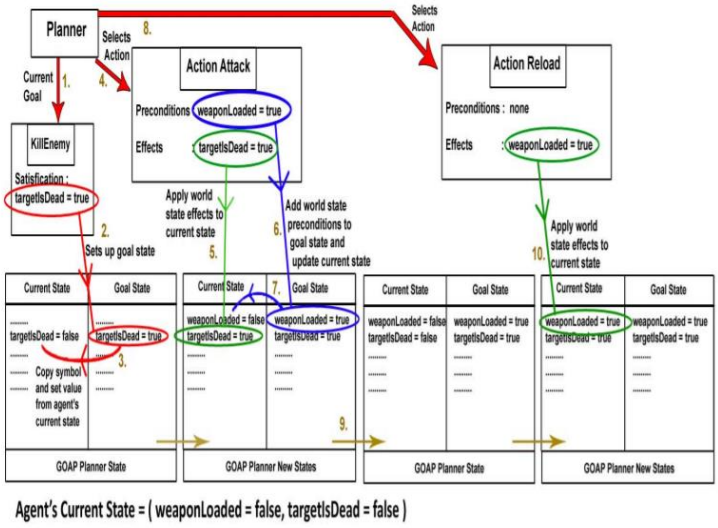


Figure 7 – Applying actions to the agent to achieve its current goal

The Action container is where all the different actions that all of the agents can take are created. When an agent is created it requests all the actions that it can make from the Action container.

The Goals are used by the agent to determine what actions to take. Each goal has a float value which determines its priority, so when a goal is completed, or a certain precondition is being met each goal will change its point value between 0 and 1 and the goal with the highest value will be the one that is set. For instance, if the agent has an enemy in its crosshairs, then the shoot enemy goal value would be 0.9 but if the enemy moved out of its crosshair the shoot enemy goal value would be set to 0.

Finally, the goal manager is where all the possible goals are stored. When an agent is created it will list all its goals in the goal manager and then the goal manager will constantly check if the agents' goals are achievable or should be change based on their current point value.

(Szita, Ponsen and Spronck, 2009) This paper examines the use of macro actions within GOAP systems. The paper also examines the difference between a regular GOAP system and a GOAP system that includes pre-set macros. In the paper a list of different macro actions were created which consisted of multiple different actions to take in a certain order. The macros were created to counter common tactics that the enemy system could use. The aim of this test was to reduce the amount of decision making that the system had to make in the hopes of creating a faster and more efficient system. The results from these tests showed that the GOAP systems with pre-set macros had results that varied more which resulted in the system being more likely to achieve its goal.

6.2 Behaviour Analysis

Kabanza, Bellefeuille and Bisson, 2010) The paper examines the use of a Hostile intent capability and opportunity recognizer (HICOR) system for behavioural recognition. The HICOR system uses three main processes for recognizing the players intent, capability, and opportunity.

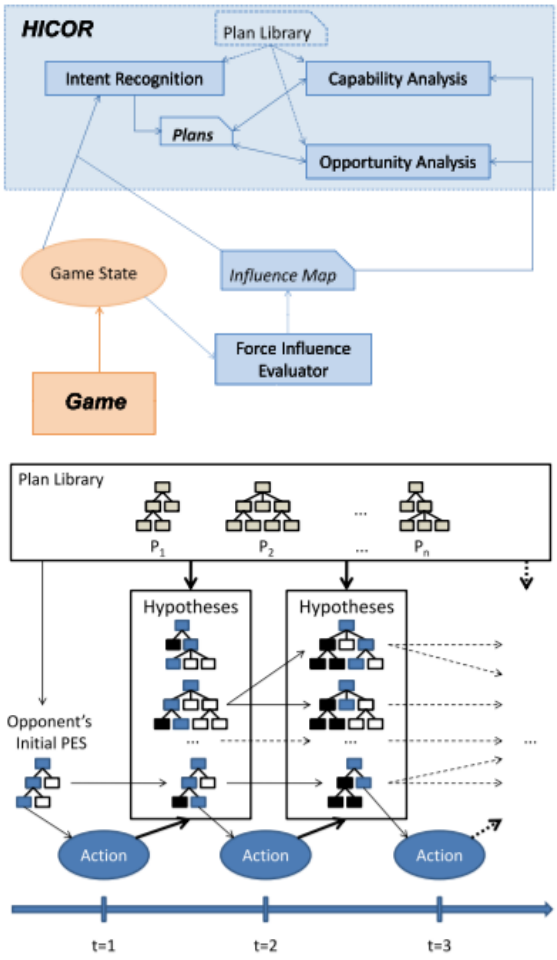


Figure 8 – HICOR architecture (Kabanza, Bellefeuille and Bisson, 2010)

The intent recognition stage takes in an influence map, plan library and game state updates (see Fig 8).

The game state, which is constantly updating, for instance current position of the enemy etc, are fed into the influence map and recorded.

Then using this information, the intent recognition stage outputs a set of plans that the enemy could potentially make including the actions that the enemy will most likely take to achieve their goal.

The capability analysis section then examines these plans and the individual components of these plans to determine if they are achievable. For instance, if the plan included the enemy entering cover, the capability analysis would check to see if the enemy was near any cover and what they would have to do to reach that cover.

The final stage, the opportunity analysis, then takes these refined plans, examines their potential success. This type of system could work very well for this project as it would be able to analyse the players actions and their potential actions which is the first half of being able to counter the player.

6.3 Summary

Based on the research into different combat systems and the evaluation of their different strengths and weaknesses and how they can be implemented, a similar system to a GOAP system has been chosen due to its ability to create multiple different plans of action based on constantly updating game states which should combine nicely with the behaviour system.

Also based on the findings of the different behaviour analysis systems a similar system to the HICOR system will be implemented as it can produce accurate predictions of the players intent and based on the way it is set up examining actions it should be compatible with the GOAP system.

7. Practice

There were many different stages involved in the development of this project. The first stage was to create the basics for a real time strategy game. For this project the style of game would be similar to Xcom (2012), so the first steps were to create a turn based game with a basic grid system for movement. After this the next stage was to create the GOAP system and implement both the player and enemy units along with the basic GOAP AI. Next was to implement they action probability system to be able to calculate different probabilities of the player taking certain actions. The final step was to then merge the GOAP and

probability systems together to get the probabilities changing based off the GOAP actions the player preforms.

7.1 GOAP System

After creating the basics for a real time strategy game, including a basic level, and implementing a grid system with basic A* pathfinding, the first main step was to create the GOAP system. Given the way the system functions, by providing an agent with different actions that they can perform to achieve a specified goal, the first step was to create the base class for all the actions.

For the GOAP system to work each action must have a pre-requisite to check if the action can be performed, an action priority value that is used to determine which actions will be prioritised, a take action function that handles what the action does when used and a way of updating the current world state.

With these four key factors identified the base action class was created with abstract functions that could be overridden by each individual action to fulfil the base requirement for each action.

Given that this project implemented a grid-based movement system it was decided that this would be used for determining the pre-requisite for each action. For this an abstract function was implemented in the base action class that takes in the unit's current position and returns a list of grid positions. Then for each action a range is determined and each grid position within than determined range from the unit position is checked for certain conditions (See Appendix D.1). For instance, within the move action if a grid position is within range but is not walkable because it is occupied already then, that position is removed, or similarly with the shoot action, if an enemy unit is not occupying a space within range, then that position is removed. After each position within range is checked all the valid positions are returned and if the valid positions list returned is not empty then it means that an action can be performed from where the current unit is, and the pre-requisite has been met.

The next stage was to set up the action priority values. For this another abstract function was created that returned a custom AI action class. This contained the position of the action on the grid and the action priority value. Then within each of the different actions the function is overridden, and a new action is returned with custom action priority values (see Snippet 1), for instance with the shoot action the priority value that was returned was a base amount plus the amount of health that each player unit had within range so the player unit with the lowest health was prioritised more. This was setup this way so that later in the development different priority values could be returned for each action based on the action probability system's results.

```
<p>
public override AIAction
GetEnemyAIAction(GridPosition
gridPosition)
{
        return new AIAction()
        {
        gridPosition = gridPosition,
        actionPriority = 10 +
        Mathf.RoundToInt((1 -
        targetUnit.GetHealthNormalized()) *
        100f)
         };
}
</p> Snippet 1: calculating action priority
```

After this the next stage was to implement the take action function. This would handle all the action's main setup logic, for instance within the move action when the take action function is called it calls upon the pathfinding system to set the players path. After this there needed to be a way to tell when an action had been started and ended to update the current world state. For this two different functions were set up in the base action class. Within each of these the logic for starting and stopping an action was handled and events were assigned to these so that other systems could run functions when actions had been started and finished. This was later used by the probability system to updated the probabilities every time a player unit had performed an action.

One the base action class was setup, and the basic actions were created, the final stage was to create the action planner. This is responsible for examining each of the actions the unit has, checking if their pre-requisites have been achieved and then comparing the action priority values to determine the action that the unit will perform.

Given how the action priority had been set up, each action had a sub AI action that contained the grid position and action priority attached to each grid position within the list of valid grid positions created for the pre-requisite check. For instance, with the shoot action for every valid enemy within range the priority would be adjusted based on the enemy's health, so the enemy with the lowest health would have the highest priority value. Given this the first step to setting up the planner was to sort through each actions sub AI actions to return the one with the highest priority value. For this a function was created in the base action class that created a list of every AI action using the pre-requisite function. These actions were then sorted by order of highest priority value and the action with the highest value was returned (see Appendix D.2).

Once there was a way to get the highest priority value for each unit action, these then needed to be compared to each different action. For this an action planner was created. Within this action planner a function was created to find the best action to perform by comparing every actions highest priority value. This worked by creating a new empty action for the best action and then looping through every action that the unit has. Then it firstly checks if the unit has action points to spend on the action and if it does it continues. Then a check is done to see if the best action is null and if it is it sets itself to the current iteration of the loop. Then if the best action is not null it will check the current best action against the loop iteration action. If the current loop iterations action priority value is higher than the currently assigned best action, then the best action is set to the current loops action. After the loop has finished running the resulting best action will be the action with the highest priority value that has had its pre-requisite met. This function then performs the best action. This is the bases for the GOAP system's operation (see Appendix 5.3).

7.2 Action Probability System

Once the GOAP system and all the actions had been implemented the next stage was to implement an action probability system. This system would be responsible for calculating a probability increase for each of the players corresponding actions based on the frequency of the used action and the cost of the used action. So, for instance one of the example probabilities created later within the project was a cover probability, so the probability of the player moving a unit into cover. The probability would increase every time the player entered cover however the amount that the probability would increase would vary based on the distance that the player had to move to get into cover and the cover type. So, if the player only moved one square to enter full cover, then the probability would only increase a small amount, however if the player moved a further distance and spent more action points to enter half cover then the priority would increase by a larger amount as it shows that the player is willing to spend more effort to achieve the overall goal of entering cover.

With this design in mind the first stage was to create a base probability class was created. This class would be the base for every action probability statistic and would need to contain the overall probability of a sub action being performed. From this multiple different sub action probabilities were created from the base actions that the player could take. For instance, an example of a base action of the player is the move action, however the move action has many different uses such as flanking an enemy or entering cover. So, the probabilities that are being analysed are sub actions of the players main actions.

With all the probabilities created the next stage was to create a manager to handle calculating the percentages of each probability. The manager contains a list for every action probability and based on every probability in the list calculates a percentage value by adding all the probability values together to get a maximum value and then dividing each actions current probability by the max value. Multiplying this figure by one hundred results in an overall percentage for each action probability. For instance, if there were 5 sub actions and every sub action had a probability score of 2 then the overall percentage for each sub action would be 20%, then if one sub actions probability increased by 1 then that sub actions overall percentage would rise to 27% rounded and every other percentage would decrease to 18% rounded (see Fig 9).

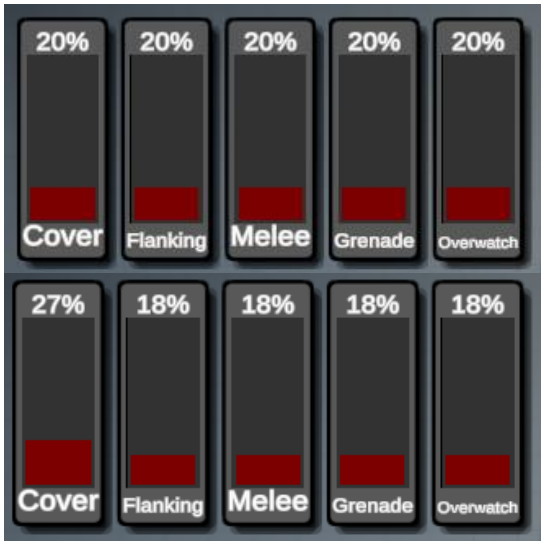


Figure 9 – Example of probability increase calculation result

With these sub action probabilities set up the next stage was to link this system to the GOAP system to adjust the probabilities based on the actions the player takes.

7.3 Combining Both Systems

The first stage in combining both of the core systems was to link the probability changes to the players GOAP actions. To do this events were assigned to each base action to trigger when their action had been completed and a function to calculate probability increase was linked to each base action through their sub action. For instance, an event was implemented within the move action to be invoked when the unit had stopped moving, and within both the flanking and taking cover probabilities an empty function was linked to said event.

After the events and actions had been linked the next step was to occupy the functions to calculate how the increase in probability would be calculated for each action. Given how the different actions vary each probability increase would need to be customised to provide an efficient and accurate change to each probability.

Once the action probabilities had all been setup the final stage was to implement different return values for each actions priority value and adjust the pre-requisites for different actions based on the counter actions highest probability. For instance, if the probability of the player entering cover was the highest then the grenade action's priority for the enemy would increase. Also, when checking the spaces for the grid positions during the pre-requisite check the key grid positions should either be covering that the player is using or cover that the player units could reach in their next turn.

During this process some issues did arise. For instance, with the previous example of the probability of the player entering cover being the highest, an issue arose where the enemy units would only throw grenades despite the fact that after having thrown one the players cover had been destroyed and it would have been more effective to have shot the player as they would have had a high hit chance and done more damage. Because of this a different approach was implemented that limited the units' actions so an action could only be performed once per turn. This limitation did see an improvement to the system as it meant that there was more variation within the enemy's play style while still countering the players actions.

After this issue was fixed and after some tweaks to each actions probability increase rate due to testing the system was complete.

8. Discussion of outcomes

The overall combination of both a GOAP system and a player behaviour recognition system for use in a real time strategy games AI was a success. The resulting system was able to accurately produce probabilities of actions based on the players behaviour and produce counter measures to said player. After multiple different play tests with different play styles purposely being tested, the results showed that the correct probabilities were both increasing and decrease corresponding to the testing playstyle. For instance, during different aggressive play styles where the player would favour using grenades and shooting, the enemy units would favour entering cover and spreading out so grenades could not hit multiple targets. Then when the rushing playstyle was tested, where the player would rush up to the enemy and melee attack them, the enemy units would favour moving out of range of the players so they couldn't melee attack them, as well as entering overwatch to catch the player units when they are running towards them.

Another result that was made clear from testing was that it was very difficult to purposely weight the probabilities. One of the concerns that was brought up through development was the possibility that the players could purposely adjust the probabilities by performing actions that they would not take multiple times to try and get the AI to act in certain ways. This did become an issue while combining the two systems, where the user was able to alter probabilities by performing actions when they were out of combat to increase the overall probability. The main instance of this was the player entering cover over and over. However, once this issue was recognized a solution was implemented so that the players action probabilities would only increase when in combat with the enemy. This proved to be a suitable solution as after this was implemented even when the player would purposely try and alter the probabilities, their actions would leave them vulnerable to the enemy's attacks. This meant that only players who were playing as the game intended were really able to alter the probabilities.

Despite there being successes to this project and it being a good start to a new type of AI for real time strategy games, there are still some aspects that would need to be improved upon if this were to be used in an actual game. The main concern with this project at this stage is the lack of variety in the testing.

While the project had been tested throughout the stages of development, the only person to actual test the project was the developer. This means that the results of this project could have some potential bias as the tester designed the systems and might not play as people who have not used the system before would. With this in mind if this type of system was going to be used as part of an actual game, a wider variety of player testing would need to be undertaken to prove the effectiveness of the system. However, as the aim of this project was to see if this type of system was possible and testing was undertaken to account for different standard playstyles, the results from this should be enough to draw a valid conclusion from.

The final improvements that would need to be undertaken would include extensive balancing. At the moment the system can fairly accurately predict the players intentions, and while this is the main aim of the project one concern does arise from the accuracy of this system. If the system is too accurate and can constantly predict and counter the players every action than this can

produce an unbalanced and overpowered enemy. To prevent something like this the probability calculations would need to be balanced to provide a less accurate representation of the players actions, or the action system could be re-worked so that every action is not dictated by the probability results, instead the base GOAP system could handle the AI's behaviour for more time and only in certain situations would the probabilities calculated be used to counter the enemy. This way the players would still have a good level of difficulty while still providing a good challenge.

This project was undertaken due to the repetitive nature that can be found in games AI. While pre-determine AI behaviour is reliable as it allows the designer to know the exact behaviour of the enemies and adjust accordingly, it can become very predictable and easy to counter. While this may be favourable for certain games where the player may prefer weaker enemies that they can defeat to feel stronger, or within games that are known for being more difficult where the player has to learn the behaviour of different enemies to defeat them, this type of behaviour can be tedious and unfun when implemented into other styles of games. This is where the main question for this project arose, can an AI system that accurately predicts players actions be balanced to produce a challenging yet fun and fair experience within a game?

Overall, yes, the system is a good starting point for exploring this type of AI system within games. It has a good base for developers to implement new actions and probabilities to test, as well as being able to easily adjust how the actions effect the probabilities. However, it is just a starting point. For this system to be a viable option within published games, the system would need to be further tested with more users and would need to go through several stages of balancing to ensure that the system provides a good balance of challenge while not being overpowered.

9. Conclusion and recommendations

The combination of both a GOAP system and a player behaviour recognition system had some promising results, both in terms of combining the two and in the overall systems use in a prototype real time strategy game. The results from this project show that this type of system could be feasible for use within games.

However, it is recommended that first steps taken regarding this project should focus on balancing of the system, as well as more user testing and expanding upon the different actions the player can take and expanding the number of different action probabilities. The results from the developer testing show that even with a small number of actions and probabilities being checked the system was able to produce a good variety of gameplay that did not feel to overpowered. So, with more time spent on introducing more actions and probabilities, along with more balancing as development progresses and user testing this system has the potential to be used within a published game.

This main aim of this project is to be a first step into further progression of player behaviour recognition within games. And with further development there is potential for this type of system to be used within different types of games as a new form of AI.

10. References

Szita, I., Ponsen, M. and Spronck, P., 2009. Effective and Diverse Adaptive Game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1), pp.16-27.

F.E.A.R. 2005. Xbox 360, PlayStation 3, Windows [Game]. Warner Bros

Orkin, J., 2006. Three States and a Plan: The A>I> of F.E.A.R. Available at: <https://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf>[Accessed 9 October 2021]

Orkin, J., 2003. *Applying Goal-Oriented Action Planning to Games.*[online] Available at: <http://alumni.media.mit.edu/~jorkin/GOAP_draft_AIWisdom2_2003.pdf>
[Accessed 28 October 2021]

Wenniger, G., 2008, January, GOAP.[online] Available at: <https://www.computing.dcu.ie/~gmdbwenniger/Finished_Projects/GOAP-Report.pdf>
[Accessed 23 November 2021]

Long, E., 2007. Enhanced NPC behaviour using goal oriented action planning. *Master's Thesis, School of Computing and Advanced Technologies, University of Abertay Dundee, Dundee, UK*.

Szita, I., Ponsen, M. and Spronck, P., 2009. Effective and diverse adaptive game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, *1*(1), pp.16-27.

Ponsen, M. and Spronck, P., 2004. *Improving adaptive game AI with evolutionary learning* (Doctoral dissertation, Masters Thesis, Delft University of Technology).

Kabanza, F., Bellefeuille, P., Bisson, F., Benaskeur, A.R. and Irandoust, H., 2010, July. Opponent behaviour recognition for real-time strategy games. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Doirado, E. and Martinho, C., 2010, May. I mean it! Detecting user intentions to create believable behaviour for virtual agents in games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1* (pp. 83-90).

11. Bibliography

Appendix A: Project Log

19007136	A 1 on 1 AI combat system that analyses player combat patterns and learns to predict and counter the player		
Date	Tasks set for the week	Brief summary of outcomes achieved, research or practical aspect completed	Questions arising and/or tasks to be taken forward
8/11/2021 – 14/11/2021	• Setup unity project and GitHub repo • Create base for the real time strategy game • Continue research	Project and GitHub setup were completed which will help with version control for the project. Basic game functions have been setup including the grid system and turn based game system which will make it easier later on when trying to implement the combat features. Research into the GOAP system was useful as it helped to provide a better understanding for the basics of how it works.	Research showed some weaknesses within the GOAP system so will research other types of AI systems to see if there other more favourable systems that could be used. Also, more research into behaviour systems will need to be done.
15/11/2021 – 21/11/2021	• Research into different AI systems for combat • Further research GOAP systems • Start research behaviour analysis systems	Further research into how the GOAP system works and how it could be implemented was carried out as well as research into other types of AI systems that could be used. The results from this research showed that the GOAP system will be the better choice for this project. More initial research into Behaviour systems was carried out showing some good results with some better understanding of basic systems.	More research into behaviour analysis systems will be carried out to determine what type of system should be used for this project. Once this research has been carried out the research document will be started.
22/11/2021 – 28/11/2021	• Further research behaviour analysis systems • Start writing up the research document	More research into behaviour systems was carried out and based on this, methods were identified which could be used in this project that would work well with the GOAP system. Based on the current amount of research the next stage was to start writing the research document and identifying what was needed to be talked about and how to structure it.	Build upon the basic plan for the research document and start writing it out fully. Start on a basic GOAP system based on the research that has been found.

29/11/2021 – 5/12/2021	• Continue writing the research document • Start on a basic GOAP system	The research document has been added to and is close to a first draft. The creation of the base classes for the GOAP system was successful. At the moment there are not a wide range of different actions but the basic foundation for the system has been implemented which will make it easier to add upon later in the project.	Finish the research document for the deadline.
6/12/2021 – 12/12/2021	• Finish research document	The research document was completed with all of the research done in the previous weeks being fully evaluated and showing what how the results from the research have affected the project and what the next stages of the project will entail.	Continue working on the GOAP system by adding more actions and goals.
10/01/2022 - 16/01/2022	• Continue creating GOAP system	The GOAP system has been worked on more, a basic range of actions has been implemented as well, the action planner has been created but not finished. Once this is finished the base GOAP system will be complete.	Continue working on the action planner for the GOAP system which will handle calculating which action the enemy unit will take.
17/01/2022 - 23/01/2022	• Continue creating GOAP system	The action planner has been setup and is working correctly so different actions are taken based on different world states. This means that the GOAP system is ready to go.	Test current finished GOAP system to ensure there are no bugs or errors with the system. Create the presentation and video for the upcoming deadline.
24/01/2022 - 30/01/2022	• Test project, bug fixing window • Create presentation and videos	The GOAP system had multiple tests run to check for any bugs or errors. Apart from some very minor issues which were quickly fixed, the system seems ready and functional. The Presentation and demo video have also been created ready for the upcoming deadline.	Now that the GOAP system has been setup the next step is to begin the pattern recognition system.
07/02/2022 - 13/02/2022	• Start pattern recognition system	The pattern recognition system has been started, the base classes for the system have been implemented and the UI has been created to track the different probabilities.	Continue working on the probability system, this includes creating all of the different probability scripts using the base script created this week.
14/02/2022 - 20/02/2022	• Continue pattern recognition system	The majority of the probability scripts have been created and a base manager has been created to assign every probability to a UI element. The base calculation for calculating the percentages for each probability has been implemented and tested with default figures.	Finish off creating all of the different probabilities as well as testing basic probability increase equations ready to merge with the GOAP system.

07/03/2022 - 13/03/2022	• Finish pattern recognition system	All of the different probability's scripts have been created and the increasing and decreasing of different probabilities has been setup so when one percentage increase the others decrease by the correct amount.	Some final testing of the system will take place to ensure that the system is ready for merging with the GOAP system.
21/03/2022 - 27/03/2022	• Test project, bug fixing window	The action probability system was tested multiple times to ensure that there were no bugs or errors before merging into the GOAP system. No major issues arose, and every minor issue has been sorted.	Begin merging the GOAP system into the action probability system to get the probability increases to be based off the GOAP systems actions.
04/04/2022 - 10/04/2022	• Begin combining both the GOAP system and the Pattern Recognition System	Began to merge both of the systems together with good results. Each probability increase has been linked to an action so when an action is performed it will increase the corresponding probability by an amount based on the effort of performing the action. For instance, if the player moves a unit into cover, then the cover probability will increase based on how far the unit had to move into cover.	Now that the probabilities are linked to the different actions and changing overall percentages correctly, the next step is to use the percentages within the GOAP system to affect the actions that are being chosen.
11/04/2022 - 17/04/2022	• Continue to work on combined system and begin testing	Finished the merging of both systems so now the probabilities that are being calculated are determining the actions that are being taken by the enemy AI. Testing of the system has started to ensure that there are no bugs before finishing off the project. Only some minor bugs have been found so far and all of those have been fixed.	Continue checking for any issues within the combined system and then finish the final parts of the project/system.
25/04/2022 - 01/05/2022	• Continue testing system • Finish final project	Bug testing continued and nothing major was discovered. After this the final touches were performed, including expanding on the base level to produce a test level and checking over all code.	Now that the main project has been completed the next stage will be to start writing the final report and to do one final test of the finished system to ensure it's ready for submission.
09/05/2022 - 15/05/2022	• Started writing final report • Final Test of project	The report has been started, the main introduction and research sections have almost been completed. The final test of the project did not raise any concerns or identify any bugs, so the project is ready for submission.	Continue writing the final report, this includes finishing the research section and starting the practice sections.
16/05/2022 - 22/05/2022	• Continued writing final report	The research section has been completed as well as the practice section. Everything necessary has been talked about and every image or snippet has been added.	Finish the last sections of the report and then record the final video.

23/05/2022 - 29/05/2022	• Report finishing • Recording of video	The report has been finished and everything has been included. The video has also been recorded and is ready to be uploaded to YouTube.	

Appendix B: Project Timeline

October	Research Draft Proposal	5 days 2 Days
November	Final proposal to be submitted by (04/11/21) Create unity project and set up GitHub Repo Create unity assets and create base code (Weapons, movement etc) Continue research and plan how to setup a GOAP system and pattern recognition system Start Research Report	1 Day 1 Day 4 Days 3 Days 4 Days 2 Days
December	Continue Research Report Write up research Hand-in of Research Documentation (16/12/21) Start creating GOAP system	5 Days 6 Days 2 Days 3 Days
January	Continue creating GOAP system Test project, bug fixing window Create presentation and videos Assessed Presentations (Video Demos & Q&A) (24/01/22)	14 Days 3 Days 2 Days 1 Day
February	Continue Implementation Test project, bug fixing window Start pattern recognition system	5 Days 3 Days 9 Days
March	Continue Pattern Recognition system Test project, bug fixing window	15 Days 4 Days
April	Begin combining both the GOAP system and the Pattern Recognition System Continue to work on combined system ad begin testing	3 Days 16 Days
May	Finish final system Test project, bug fixing window Start writing final report	6 Days 6 Days 10 Days
June	Finish final report Create video for hand in Package final artefact	5 Days 1 Day 2 Days
July	Final Hand In (18/07/2022)	1 Day

Appendix C: Assets used in the Project

Polygon Prototype Assets (Assets/PolygonPrototype) - https://syntystore.com/products/polygon-prototype-pack

Walking, Shooting and Knife animations (Assets/Animations/Unit) - https://www.mixamo.com

Further Appendixes D

D.1

```
<p>
public override List<GridPosition> GetPositionsList(int offset = 0)
{
    List<GridPosition> validGridPositionList = new List<GridPosition>();
    GridPosition unitGridPosition = unit.GetGridPosition();
```

```
    for (int x = -maxMeleeDistance; x <= maxMeleeDistance; x++)
    {
        for (int z = -maxMeleeDistance; z <= maxMeleeDistance; z++)
        {
            GridPosition offsetGridPosition = new GridPosition(x, z);
            GridPosition testGridPosition = unitGridPosition + offsetGridPosition;

            if (!LevelGrid.Instance.IsValidActionPosition(testGridPosition)) continue;
            if (!LevelGrid.Instance.HasUnitAtPosition(testGridPosition)) continue;

            Unit targetUnit = LevelGrid.Instance.GetUnitAtPosition(testGridPosition);
            if(targetUnit.IsEnemy() == unit.IsEnemy()) continue;

            validGridPositionList.Add(testGridPosition);
        }
    }

    return validGridPositionList;
}
</p>
```

D.2

```
<p>
public AIAction GetHighestPriorityEnemyAIAction()
{
    List<AIAction> enemyAIActionList = new List<AIAction>();

    List<GridPosition> validActionPositions = GetPositionsList();

    foreach (var gridPosition in validActionPositions)
    {
        AIAction aiAction = GetEnemyAIAction(gridPosition);
        enemyAIActionList.Add(aiAction);
    }

    if (enemyAIActionList.Count > 0)
    {
        enemyAIActionList.Sort((AIAction a, AIAction b) => b.actionPriority -
a.actionPriority);
        return enemyAIActionList[0];
    }
    return null;
}
</p>
```

D.3

```
<p>
private bool CanTakeAIAction(Unit enemyUnit, Action onEnemyAIActionComplete)
{
    AIAction bestAIAction = null;
    BaseAction bestBaseAction = null;
    foreach (var baseAction in enemyUnit.GetBaseActionArray())
    {
        if (!enemyUnit.HasActionPoinsToTakeAction(baseAction)) continue;

        if (bestAIAction == null)
        {
            bestAIAction = baseAction.GetHighestPriorityEnemyAIAction();
            bestBaseAction = baseAction;
        }
```

```
        else
        {
            AIAction testAIAction = baseAction.GetHighestPriorityEnemyAIAction();
            if (testAIAction != null && testAIAction.actionPriority >
bestAIAction.actionPriority)
            {
                bestAIAction = testAIAction;
                bestBaseAction = baseAction;
            }
        }

        baseAction.GetHighestPriorityEnemyAIAction();
    }

    if (bestAIAction != null && enemyUnit.TestActionPointsToTakeAction(bestBaseAction))
    {
        bestBaseAction.TakeAction(bestAIAction.gridPosition, onEnemyAIActionComplete);
        return true;
    }
    return false;
}
</p>
```