

Computer Games Development CW208

Research Report

Year IV

[Stephen Ennis]

[C00181305]

[24/04/2017]

[Oisin Cawley]

[Declaration form to be attached]

**Contents**

[Acknowledgements](#_gjdgxs)

[Project Abstract](#_zdsonw8pifus)

[Project Introduction and Research Question](#_1wc7vzgw3xv3)

[Literature Review](#_nxlxgycy5yk8)

[Study](#_jhfmi9ucgtkr)

[Project Description](#_d0xc8jskzpoc)

[Project Milestones](#_gfut6fq37ndq)

[Results and Discussion](#_iqi83f22fnsl)

[Project Review and Conclusions](#_6d7c2qv0d0oq)

[References](#_17dp8vu)

# **Acknowledgements**

# **Project Abstract**

The purpose of this research is to implement tactically aware Artificial Intelligence. The way the AI behaves can really influence the enjoyment a player gets from a game. If the AI is perceived to be smart enough to move in real-life squad formations then the game is likely to be of a higher quality. Many AI in existing games following simple scripted behaviour such as “follow and attack”. I aim to implement an AI such that they will respond to changing tactical situations. This means they will behave with mainly two objectives. First they must ensure the elimination of enemies and second they should aim to keep casualties taken to a minimum. I will be using a simplified STRIPS like planning architecture called Goal Oriented Action Planning (Goap) to accomplish this. This allows a set of actions to be specified and then a plan can be formulated by creating a tree graph of all possible action combinations to reach the desired goal state when given a specific world state followed by comparing the costs of all the solution leaves in the graph to find the best. Traversing in reverse through the actions that came before the cheapest leave will give the best path of actions applicable to completing the goal.

# **Project Introduction and Research Question**

The goal is to make the AI behave in response to its current situation and plan accordingly. There are many examples of tactical behaving AI out there. Many games implement some form of Finite State Machine in order to bring their AI to live. Some games are for educational, realistic simulations such as those used by the American army. An example of this is “America’s Army” whose purpose is to show the realistic side of battle and squad tactics. The purpose of these games are for military training.

Generally the defining feature of a squad in a military situation is their ability to work as a cohesive entity. Rather than each soldier being given direct instructions on how to behave, the squad leader is tasked with planning the entire squads next move. The units within the squad will move as commanded but in a way in which they support each other to limit casualties. The effectiveness of this type of AI varies depending on a number of factors for example; should cover be sacrificed in order to help a teammate or should a longer path be taken to ensure a safer route.

An AI that can be seen to behave more human like in battle scenarios causes the player to be more interested with the gameplay. I believe in pursuing the area of research I will produce an AI capable of of responding to its surroundings appropriately. This AI will appear to behave a lot more sensible when compared to AI that does not plan out it’s actions.

*How does a more tactically aware Artificial Intelligence implemented using a planning architecture (GOAP) perform in comparison to the random selection of decisions.*

First I will discuss existing research into the area of militaristic AI for use in games and then from here outline the aims and procedure of this project. The project will test which type of AI performs better and which behaves more realistically.

# **Literature Review**

There has been a great amount of time and effort put into the advancement of AI over the last decade. This is evidenced by an increase in the amount of high quality games which take advantage of smarter AI to provide captivating gameplay. Some games such as XCOM 2 ( released February 5 2016) show examples of turn based, squad AI. Other releases have shown off advances in real time strategy Starcraft 2 ( released July 27 2010). The aim of all of these varying implementations is to make the player believe that they could be facing real humans rather than the computer. There are a number of different ways to implement smart AI capable of behaving like a squad.

The vital key to achieving squad behaviour, lies in the movement. A squad moves together as if it was one entity. The individual units plan their paths of approach. The traditional approach to movement in games is to use a pathfinding algorithm with A\* being one of the most commonly used. In [2] it is brought to attention that not only is movement important but so are two additional concepts. That is the concept of concealment and cover. We must move in such a way that we are reduce our visibility to the enemy while staying behind protective obstacles that prevent enemy fire from hitting us. We have to rethink our approach to movement. No longer is the shortest path the most optimal path. A path from one location to another that may have been chosen at a previous point in time may no longer be the path that's chosen again as our algorithm now relies on the costs of this movement. A change in enemy location may result in the particular route being exposed and a longer,more defensive, path is required instead. In [2] an approach is presented whereby a precalculation step is used to generate a roadmap and corridors of movement. On this roadmap we use tactical information values to show how tactically valid points on the corridors are. These values will change during gameplay. The movement is the squad is the same, the leader takes point and rest follow while trying to maintain firing positions behind them. This method is best used in outdoor areas with corridor like spaces (streets and buildings). In [3], there is still an overall path calculated with A\*. This path is used to guide the individual units in their calculation of a path. Each unit does it’s own path finding using a multitude of factors of costs, similar to those in [3]. Costs include distance, time, cover and distance from the leader. Search times are staggered so they will not move at the same time. An estimated time of arrival manager is used to prevent collisions from taking place. In this approach we get a squad movement that is more suited to indoor locations. A unit can clear a room or use different areas of cover along pathways. If they get too separated they will come back together to ensure they can help each other. In [5] more information that determines a path is introduced. A location may be a particularly suited to a certain unit type such as a sniper spot. This factors into the best path calculation.

An important factor in this movement lies in the detection of threat. A threatening area is one in which we have spotted enemy units. In [5] this takes the form of an influence map. This is constantly been updated throughout the running of the application. The damage that can be expected to be taken at each grid location is calculated. This can then be used to find the ideal cover locations that minimises the damage. [5] also introduces the concept of suppressive fire. This is fire a unit the squad can lay down on an opposing units location to prevent that unit from firing or moving. This is all added into the cost calculation when moving to determine the best paths to take and whether to lay down suppressive fire on certain units to reduce that damage. In [3] the movement algorithm takes into account the available cover. It uses waypoints to plot out moves. Each waypoint has tactically useful information available. This allows the pathfinding to also prefer high cover areas that provide suppressing fire opportunities. It uses a Threat Probability Model in order to find areas which are more likely to contain threats. This is done using a Isla occupancy map (cells containing probabilities and diffusing to neighbors) but modified to use waypoints instead of a regular grid. [3] is about a 3d application so a lot of its cover functions are in relation to the 3d models and visibility of pixels which isn’t relevant for this project. In [2] for every point on the corridor a threat level is similarly calculated. In all these examples some type of map is vital for the finding of cover spots and danger zones. [2] using an interesting approach where once the initial roadmap is provided it can calculate all of the corridors from the freespace. These corridors are paths which can be used. It then generates all the points of danger as well. [1] uses the ANTS algorithm for its threat assessment. Each unit is predicted to move in a direction. This direction is decided not entirely randomly as the military significance of surrounding grid spaces is taken into account. Once a favourable location is reached (provides good cover, sniper spot) then a “pheromone trail” is left along the path it took. This is used by other units to help them determine their paths. The result is the likelyness of grid spaces to be targeted by enemy units.

In [1] a map is used to store the probabilities of the threat for the grid locations. A roadmap is created first in [2] followed by corridors of the freespace. Default tactical information is generated at points on the corridors. This data structure is created initially. As tactical information changes throughout the application's run time then so too can the information stored in this data structure. Searching is optimised as first you search for which corridors should be traversed before finding the optimal path through points within these corridors. This avoids the need to search all corridors. [3] uses a map to store information. It’s a Isla occupancy map (cells containing probabilities and diffusing to neighbors), This has to be updated in realtime in order to reflect the correct threat level. Threat is increased when there are more visible enemy units and lingers for a while after they are gone. [5] again uses a map to store the threat at each grid location. A grid based system is common to all these papers with an influence map and waypoints also appearing regularly.

Jeff Orkin implemented Goal Oriented Action Planning in order to create the AI for F.E.A.R, a first person shooter 3d game. He refers to it as “a simplified STRIPS-like planning architecture specifically designed for real-time control of autonomous character behavior in games”. It was used to great effect in F.E.A.R. They only needed a Finite State Machine (FSM) with three states instead of something a lot more complex [6]. This approach to the development of AI means less work on the part of the developer as they do not have to create a response to every situation. There is a group of usable actions available to the AI and a goal to reach. GOAP will take the usable actions and determine the most cost effective combination in order to reach the goal state.

These papers all provide an interesting insight into what other relevant work is out that in the creation of realistic tactical squad behaviour. The Monte Carlo technique in use in [4] will not be the best approach to this project as it was used in an implementation of a RTS game which had other factors to consider such as resource collection and economic and military power compared to just a single squad. The [1] ANTS algorithm will not be used as the waypoint techniques present in the other papers will be sufficient for the development of this project. Despite [3] being a 3D application it still gives incredible insight into the concepts required to perform squad based movement. GOAP is the technique I am investigating. These other algorithms and technique I have discussed have provided a value insight to the different options available.

# **Study**

The project will be a battle arena where two teams of units are pitted against each other. There are walls surrounding the arena and also within it. These walls block the line of sight of units and provide protection from attacks. Team A and Team B will consist of 4 units each. Units have ammo and can fire at each other. A set of actions are available for use by the units. These actions will change the state of the world. Actions may require preconditions to be met before they can be accomplished. In order to determine the best set of actions to take to reach a certain state, each action also has a cost. The lowest cost path is chosen to complete the goal. A tree is built up with all the leave nodes containing solutions. Each node contains the action to get to that state and the action applied previously. This can be traversed in reverse to get back to the first action. The AI will automatically face and fire at nearby enemies. They will prioritise the closest enemy. The GOAP AI will use these actions in such a way to ensure it always has good cover from the enemy and enough ammo. The random AI will random select target nodes to move to and occasionally go to cover. Being in cover will provide a small amount of healing which is another action the GOAP AI takes into account. The determination of what goal for the GOAP AI to take will be decided with the use of a Finite State Machine (FSM). The FSM will only contain 4 states. One state is for the normal random AI and not the advanced GOAP AI. The 3 states are Idle in which a goal is selected, MoveTo and PerformAction. Actions may require the unit to be in range and the state will keep switching to MoveTo until you are. A\* is used extensively for pathfinding between a list of nodes. Each node has information defining its location and connecting nodes. A\* will traverse along the lowest cost path between connecting nodes to teach an end destination. Cover nodes are also implemented in this graph so they can be reached.

Aim:

* Investigate if the use of a planning architecture such as GOAP for an AI will cause it to perform better than an AI using the random selection of targets
* Investigate if the use of such a technique will appear to a spectator to be more realistic
* 2 different types of AI.

1. Randomly choose target locations
2. GOAP: Have a set of actions that can be undertaken to reach a goal. This goal is chosen based on what is currently the most required state to be in. Using this, target locations will be chosen.

Measures:

In order to compare which is performing better. I will record the following data over the same time frame.

* Total kills for each team
* Total deaths for each team
* Total shots fired for each team
* Total nodes traversed for each team.

This data will be measure in 3 situations:

* random AI vs the random AI
* random AI vs the GOAP AI
* GOAP AI vs the GOAP AI

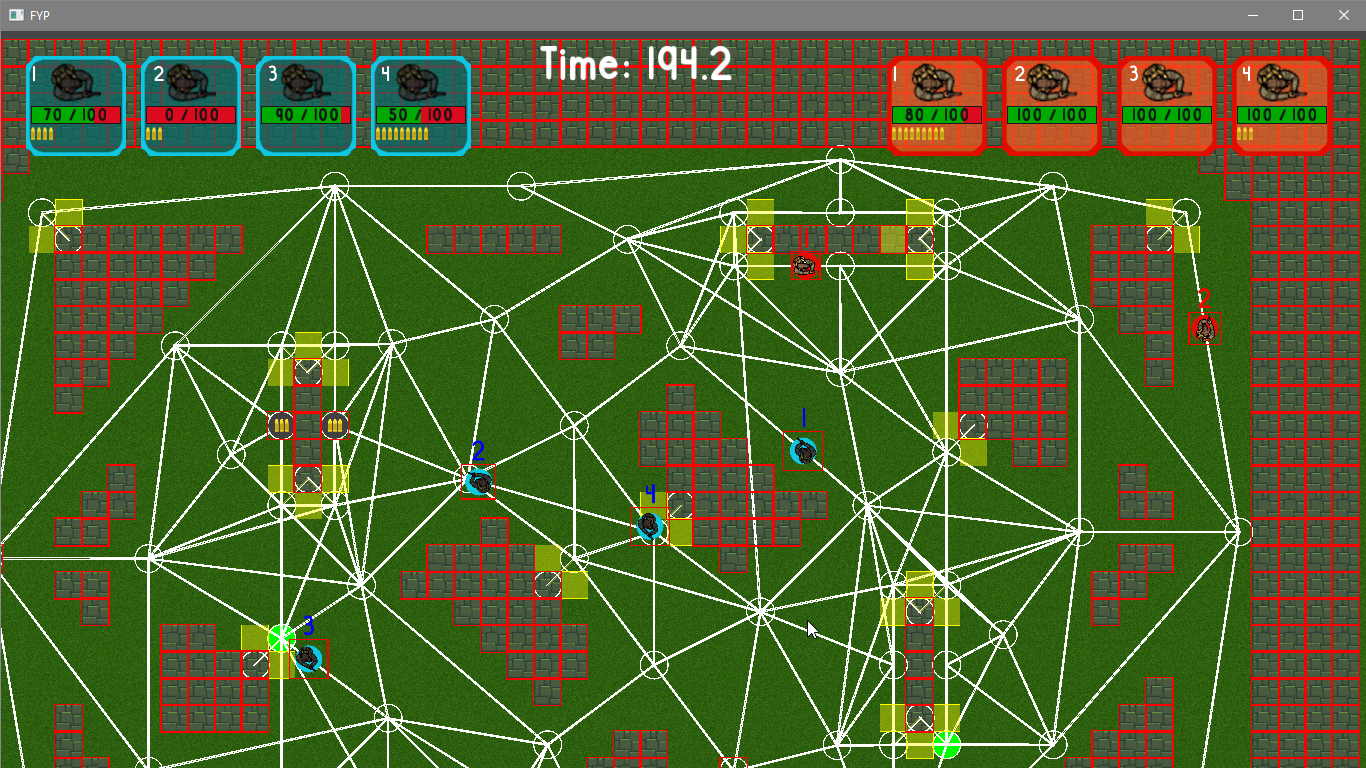
The tests will be performed multiple times and then I will get an average.

# **Project Description**

The project will be a top down 2d simulation. This simulation is of a two teams of units (A and B) fighting against each other. They can fire bullets and collisions with each other has no response. Bullets cause damage to units hit. Walls split up the play arena. The AI take cover behind these walls in order to avoid being hit or spotted by enemies. Ammo regularly spawns on the map so the units can reload. The two groups of units will keep fighting until a time limit is reached in which the statistics of the game will be shown. The GOAP AI will make use of cover and ammo pickups better than the regular AI which just choose its target randomly. The health,ammo,cover status and ID of the units of both teams are displayed at the top of the screen. The ID is also displayed above the corresponding unit. This provides the current status of all the units. The units heal over time and twice as much while in cover.



Here is another image but of the debug information enabled in order to display the nodes and their connections.



The red squares are collision rectangles. The white circles are the nodes in the pathfinding graph. The white lines are connections between nodes. The nodes surrounded by yellow rectangles are the cover nodes and the cover slots available.

**Description of Conformance to Specification and Design**

For the most part the final product matches my specification and design. There were changes I made in order to simplify it and focus only on the GOAP AI vs the regular AI. I had planned to polish it a lot more and make it more like a game. The product submitted is still capable of seeing the performance of GOAP AI. I was going to use a threat influence map to make the detection of enemies more dynamic. Instead I ended up using just raycasting to detect enemies in sight. This worked out better as it made it easier to detect when you were in cover (no enemies in sight compared to the probabilities if an influence map was used).

**Description of Learning**

On a technical level I have learned a lot. I have an excellent comprehension of Goal Oriented Action Planning. I also have a more complete understanding of A\* as I implemented it in a way that I had not before. (Using arcs to form connections between nodes rather than a tile map). I learned a lot about loading from JSON for data and map loading(created using Tiled) as I used this to load in all of the information needed. I found this helpful to have one central location to change variable and see the results reflected in the project.

On a more personal level I have explored more approaches to AI programming to mimic human thinking. Before this project most of my AI were extremely simple. The use of planning techniques makes AI programming even easier and modifiable. Using planning, AI characters are less repetitive and can perform in unexpected human like ways.

# **Project Milestones**

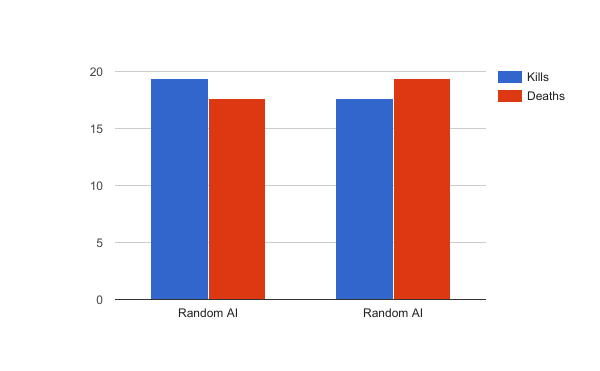
# **Results and Discussion**

I did 3 different comparisons:

* Random AI vs Random AI
* Random AI vs GOAP AI
* GOAP AI vs GOAP AI

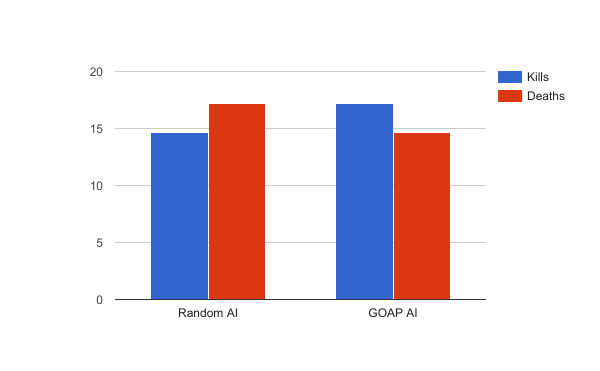
Random AI randomly selects nodes to pathfind to and cover locations to move to.

GOAP AI uses a plan to help it decide.

For each one of these comparisons I got the average number of Kills, Deaths, Shots and over 100 times. Each test was a sped up simulation of 5 minutes of play.

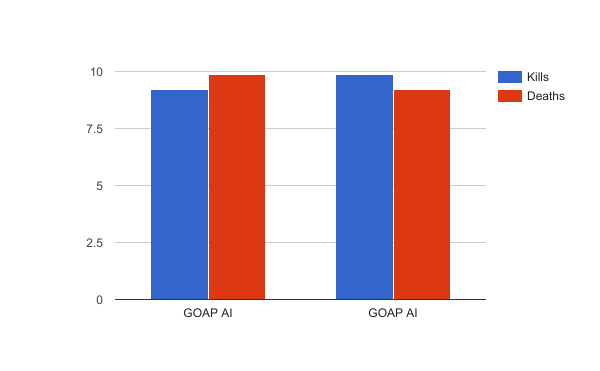
*Shots - Random AI 305 | Random AI 274*

Even when both AI are the same, there are still differences in results. This is due to its random nature. Shots fired is different if the AI did not randomly pick a path that let it pick up ammo therefore meaning it didn't have as many opportunities to shoot.



*Shots - Random AI 267 | GOAP AI 227*

Now this is the most interesting test. Here we can see the GOAP AI outperforms the random AI. This is a result that was expected. GOAP allows it to prioritise getting ammo if it is low so it's always ready to shoot. As it stays in cover longer, it also heals more in order to outlast the damage it takes. It runs away from enemies if they get too close so they helps it dodge bullets.



*Shots - GOAP AI 172 | GOAP AI 181*

Finally GOAP vs GOAP is a lower kill/death match up. This is due to GOAP AI being a lot safer than the random AI. It will run to cover that will protect it from enemy fire if they get within range.

Overall these results are as I expected and helps to prove that the use of advanced AI techniques can improve the performance of AI.

GOAP also does result in more realistic behaviour. The AI move from cover to cover and in general appear to stay near each other. This is likely due to them trying to move away from enemies and the areas without enemies are likely to have allies.

# 

# **Project Review and Conclusions**

I believe the project was successful in proving that GOAP AI can perform much better than an inferior AI. In individual tests the degree to which the GOAP AI beats the random AI can vary. This is due to the random nature putting the random AI into precarious situations which the other AI will try to avoid. It can also happen that the random AI will win which is to be expected as again it might get “lucky” with with its random locations. Both comparisons that compared the same AI with other turned out as expected with little real differences.

If i was to start this project again I would choose a different situation in order to test out the performance of GOAP. I would have liked to have many more options for the planner to check so there are a lot of different ways to reach the same goal. Using Tiled to create the tile map was a great help as it really simplified the process. RapidJson is a great library to parse the JSON from the map editor so in combination it worked well. There were a lot of features I had to drop as it would have taken too long to do in SFML. I enjoyed SFML but a different approach I would have taken would have to use Unity. Unity would have been a good choice as it takes away a lot of the build up and would have let me focus on the algorithms more.

I think it would be interesting to has many different goals and actions shared among different types of AI. I am curious to see the modularity of GOAP put to use.

# **References**

[1] Svenson, P. and Sidenbladh, H. (2003). *Determining possible avenues of approach using ANTS* [Online]. (URL <http://www.nada.kth.se/~hedvig/publications/fusion_03b.pdf>). (Accessed 24 October 2016).

[2] Kamphuis, A. (unknown). *Tactical Path Finding in Urban Environments*

[Online]. (URL <http://www.academia.edu/4825358/Tactical_Path_Finding_in_Urban_Environments>). (Accessed 24 October 2016).

[3] Darken, C.J., McCue, D. and Guerrero M. (unknown). *Realistic Fireteam Movement in Urban Environments*

[Online]. (URL <https://www.aaai.org/ocs/index.php/AIIDE/AIIDE10/paper/viewFile/2143/2537>). (Accessed 24 October 2016).

[4] Balla, R. and Fern, A. (2009). *UCT for Tactical Assault Planning in Real-Time Strategy Games* [Online]. (URL <https://www.semanticscholar.org/paper/UCT-for-Tactical-Assault-Planning-in-Real-Time-Balla-Fern/17e4be4e289403fa6c73a9374bb227fc5b350cb3>). (Accessed 24 October 2016).

[5] Shi, Y. and Crawfis, R. (2014). *Group Tactics Utilizing Suppression and Shelter* [Online]. (URL <http://0-ieeexplore.ieee.org.www.library.itcarlow.ie/stamp/stamp.jsp?arnumber=6934139>). (Accessed 2 November 2016).

[6] Orkin, J. *Three States and a Plan: The A.I. of F.E.A.R.* . (URL <http://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf>). (Accessed 6 December 2016).