|  |
| --- |
| Southampton Solent University |
| DAC619 AE1 |
| Artificial Intelligence for Games |

|  |
| --- |
| Q13375466\_RAY\_STEPHEN\_DAC619\_AE1\_Report |

Contents

[Design 3](#_Toc534248235)

[Identification of Required Behaviours 3](#_Toc534248236)

[Chosen Algorithm 3](#_Toc534248237)

[Other Algorithms Considered 3](#_Toc534248238)

[Testing 4](#_Toc534248239)

[Test Plan 4](#_Toc534248240)

[Testing Results 4](#_Toc534248241)

[Implementation 4](#_Toc534248242)

[Conclusions 4](#_Toc534248243)

[Appendices 4](#_Toc534248244)

[Appendix A – Source Code 4](#_Toc534248245)

[Appendix B – State Diagram 4](#_Toc534248246)

# Design

## Identification of Required Behaviours

These are the behaviours that each of the AI agents will need to perform to successfully play the game of capture the flag.

|  |  |
| --- | --- |
| * Go to Enemy Base * Grab Enemy Flag * Go to Friendly Base * Heal * Use Powerup | * Chase Enemy * Grab Friendly Flag * Go for Powerup * Defend Base * Attack Enemy |

## Chosen Algorithm

The algorithm that I decided to implement was a state machine. The primary reason behind this choice was the presence of distinct states in the behaviours required, which would heavily enable itself to that of a state machine. A state diagram of the state machine can be seen in Appendix B. Some of the advantages of using a state machine for the AI logic is that it is quite easy to trace a path of the AIs behaviour as it will be playing, making the logic that it will be applying easy to understand and to debug for any potential issues whilst it is going through the game.

A large disadvantage of the state machine is that even with such a small amount of states, connections between them and transitions are becoming complex, with many states all leading to many places.

The algorithm will work by first placing the AI in the state of going to the enemy base, as at the start of the game it is known that both flags are in their respective bases, and both teams will need to take the flag back to their base in order to start scoring points. Along the way to this goal, the AI can choose to do other actions based on what they can see, so they can be opportunistic and chose to go for Powerups or Med kits in order to have them for later use if needed, and if they encounter an enemy along their way, they will try to attack them in combat. AI agents will then try to claim flags and bring it back to their base, with an ideal situation being both having the enemy flag, and their own flag in their base to defend. This will continue forever, with flags going back and forth between the AI teams.

## Other Algorithms Considered

One algorithm that was considered for the game was goal orientated behaviour. Goal orientated behaviour would be a very good selection for this project, as it would be easy to identify needs / goals such as need to heal, claim flags, and to defend. This would be a very robust solution to use, as the variety of actions and goals allows for a wide range of behaviour, and the AIs would seem to have very different behaviours between them using some fuzzy logic and a few actions that meet multiple goals and needs. The main reason why this was not used, was the difficulty of designing a successful set of goals and actions, as to make the behaviour feel more realistic, it would need to be very vast in its capabilities, which in this project it would not be the easiest to do.

The other option that was considered was the use of a decision tree to determine the behaviour of the AI agent. However, this was not chosen as the final algorithm as it would be required for the AI to be able to change their behaviour quite fast based on conditions, and it was felt that it would not provide enough freedom for the AI to move between the actions or behaviours that it can be doing. However, there would be benefits to using a decision tree, for example, the decision tree would be easy to debug, as you can trace the behaviour along the tree, it would also be a very efficient solution if it is designed correctly, as careful planning of the tree would minimise the number of decisions that the will need to be made at any point, with the least common decisions such as using powerups right at the bottom of the tree, and more used ones, such as going to the enemy base and attacking enemies closer to the top of the tree.

# Testing

## Test Plan

## Testing Results

# Implementation

# Conclusions

# Appendices

## Appendix A – Source Code

Appendix B – State Diagram