**ARTIFICIAL INTELLIGENCE BEHAVIOUR PLAN**

**What am I going to make?**

I’m going to make a 2D defence game similar to the old arcade game “Missile Defence”. The aim of the game is simple: to defend yourself against aerial attacks from the enemies by shooting them before they can get close enough to attack you.

**Decision Making Technique?**

The enemies’ decision-making technique is going to be a Finite State Machine, by which it will only have a few basic behaviours, such as seek, avoid, and wonder. The enemy is going to be actively seeking the player, whilst avoiding contact with other enemies and the player’s projectiles. It is also going to have the wander behaviour implemented into it’s movement so as to give it a sense of foreign and sporadic movement.

**Pathfinding?**

I am going to be implementing Dijkstra’s path, and if time permits me, upgrade it into A\* pathfinding for optimisation and an extra challenge.

**Pros and Cons of this Choice?**

The pros to using a Finite State Machine is that it is fairly simpler to write than some of the other decision-making techniques, and it is suitable as my AI will only have a small number of states.

The cons to using a finite state machine is that, should I implement more states or take my game further than I am currently anticipating, then the finite state machine may be too simple to handle all the different states, and I may have to rewrite a different Decision-making technique to adjust for the new influx of states. It is a good choice for what I currently have in mind, but runs the possibility of making things difficult should I add on to or change my project in the future.

The pros to implementing Dijkstra’s path (and hopefully A\*) is that it is efficient at finding a short route from point A to point B. However, Dijkstra’s path may prove difficult to implement, and with multiple enemies on screen at the same time it could become quite expensive as the algorithm uses a “flood” approach in it’s calculations.

**Alternative Techniques I could have chosen?**

I could have chosen a Jump-Point Search; however I feel it is unnecessary as the path from the enemy to the player will be relatively empty, and therefore too inexpensive to worry about this type of optimisation. There was also the possibility of implementing a Decision Tree, though again, I feel that there will not be enough behaviour states to justify having such an optimised decision-making technique, and so the Finite State Machine will be perfectly suitable.