# Interactive Agents

# S190988

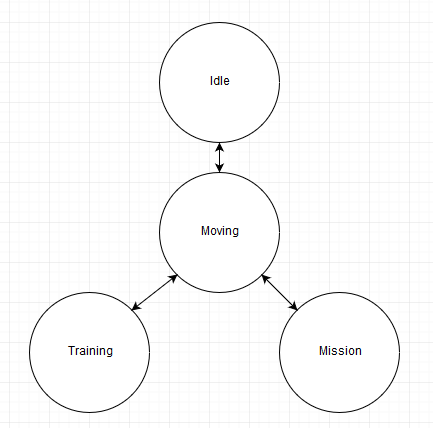
Instructions:  
- Launch in 800x600 windowed

- You can click an Agent, The Spy, A Mission (door) or one of the Items in the Training room for more info.  
  
- 2 more missions can be unlocked

- if you want to test the pathfinding, you can enable gizmos in editor and It will show the last path generated. (you can also add any tile into the Wall layer or a collider with the wall LayerMask and they will path around it)

The aim of “Interactive Agents: The Game”, was to implement a gamified version of the brief.  
The game has 4 different types of Agent:

Agent:  
Primary actors in the scene who use A\* Pathfinding to move between points of action. They have a FSM which governs their primary behaviours. It looks like this:



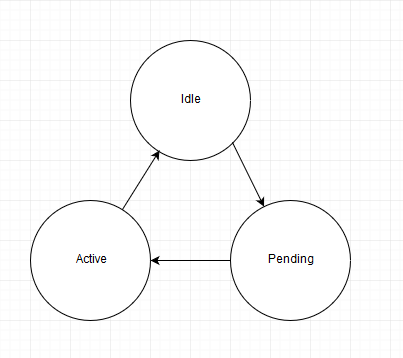
Rather than adding more states for the finer points of movement and behaviour, I looked to extend the complexity of the transitions and the flexibility of states.

Move is esseitially a transition state, and it drops out of it into another state once the movement is complete.

In Three States and a Plan (Orkin, 2006) Orkin discusses the idea that States don’t need to be numerous to be functional, and he jokes that his “3 states” are in-fact “2 states” but could really be simplified down to one. I took inspiration from this. My Agents have a substate which operates inside them, it’s referred to as Movetarget but can be used with more flexibility than the name implies. (there are two switch statements which check this state, one in the primary movement state and one in the primary idle state). Movetarget could be considered a plan with only one step.

Agents will look for an available mission, attempt to complete it then move back to a resting point.  
if they failed their previous mission, they will look to train their stats rather than try a second time.

Missions / Training:

  
both of these Agent types have very simple behaviours which simply allow them to be used by agents and remain exclusive. They don’t have a substate as they don’t need one.

Spy:  
The Spy doesn’t use a State Machine and instead has steering behaviours.  
it operates a flee behaviour for walls and Agents as well as seek behaviour to try and hack the mainframe.

Other Notes:  
I didn’t finish the gamification elements as they wouldn’t had an impact on grade proportional to time required (if any) and as such I kept them as a personal project and only included the proof of concept ideas.  
Missions already have a completion tracker, reward system and training has a value for upgrading (as well as a sprite) however these elements are all unimplemented.   
  
I’d have liked to build a FSM class and used it for all three of the FSM enabled elements (4 if you include chairs) but felt it wasn’t needed as the Agents and Missions have such different behaviours and I wasn’t sure I’d be storing much in it other than an Enum and some empty functions. It would have been a nice starting point, but by the time it was considered was already redundant.  
  
The Pathfinding system was based on a YouTube Video (McCluskey, 2018) however I found critical flaws which were corrected and improved upon. (Operated on incorrect axis, didn’t check fCost, had a broken loop) I found his use of Gizmos particularly interesting and learned a lot from it.

My steering behaviours are based on the flocking principles (Reynolds, 1986) however I didn’t need to go into the detail that Boids does and didn’t need to make my Spy consider the location of other Spy’s as there is only one.

# Bibliography

McCluskey, D. (2018) *Unity - A Star Pathfinding Tutorial.* [Online]   
Available at: https://www.youtube.com/watch?v=AKKpPmxx07w&t=512s

Orkin, J. (2006) *Three States and a Plan: The A.I. of F.E.A.R.* [Online]   
Available at: http://alumni.media.mit.edu/~jorkin/gdc2006\_orkin\_jeff\_fear.pdf

Reynolds, C. (1986) Boids.