# Artificial Intelligence for Game Development – Written Report (CW2)

# Video Link: -

# <https://youtu.be/PYrJFTyf9nE>

GitHub Repository Link: - <https://github.coventry.ac.uk/neulk/Artificial_Intelligence_5026CEM.git>

# Game’s Design: -

My game’s design is based on Hide and Seek, the Artificial Intelligence (AI) is the searcher for the player, the AI will have states which will mean that I will be implementing a Finite State Machine from the ground up. I am planning to make my own A\* Navigation system for the enemy but with me being conscious of time this is just an idea. If I am unable to make this system, I will be using Unity’s NavMesh Navigation System to allow the game to have some working functionality. I will code the movements and responses from the Enemy itself.

My Finite State Machine (FSM) will consist of 4 to 5 states that the enemy should seamlessly be able to transition between, this will require a lot of thought to ensure bugs and errors do not occur. I have made two plans for the FSM, one rough and one more advanced. Both plans are below to show my thought process and the intended outcome of the FSM. I am choosing to use an FSM as my technique because I will need to have the enemy be able to exhibit multiple behaviors. FSMs and Markov Machines are the industry standard method of creating characters that need similar traits. To clearly display the state that the enemy is currently in I will be using Gizmos as an extra feature for clarity. FSMs are the banes of decision making in games and as my game isn’t a direct copy of Hide and Seek, I am choosing to add a state that will make the game harder for the player but easier for the enemy which will be discussed in the state section.

Diagram

Description automatically generated

Figure 1 is my basic plan for my FSM, so I have a rough idea of what I will need

Diagram, whiteboard

Description automatically generated

Figure 2 is a more descriptive flow of what I plan to make within the FSM.

A\* Pathway navigation might be my other chosen AI technique, and this is because I plan to have the enemy moving around the map for all other states than Idle. To get around the map efficiently without walking into walls and quickly, the enemy will constantly need to be aware of its surroundings and be constantly searching for the quickest route. I think that A\* requires a lot of effort to create and with me choosing a to have an FSM as my main AI technique I doubt that I will be able to create a navigation system in time just due to it being an option to base a whole coursework piece around. I believe that this might become a strain on me when it comes to making it which is why I have chosen to use the Unity NavMesh as a backup. I would use A\* Algorithm as it takes in the pros of Dijkstra's Algorithm which is to find the shortest path and have it be reliable and the pros of Greedy Algorithm which focuses on performance over reliability.

I briefly thought about using Fuzzy logic as my main code base for the AI technique that I would use for this game, and it would work but I found that a lot of the variables that I would need would be based on true/false checks with further background coding to set them to true or false. For example, the Enemy being able to see the player or not is a massive part of the structure for the games and AI overall and will be affecting the states and their changes massively; if the Enemy can only sort-of see the player, then it will not logically have anything to chase, I suppose the enemy could check for the player, in the location that the enemy thinks it's in but I am really trying to wean out as many states as I can get for the enemy to show off the complex thinking I will have put into ensuring that there are no issues. Fuzzy logic can also leave room for unpredictable behaviour traits and as I am trying to present a working AI piece this wouldn’t be very compelling, and results would change from time to time. If I stick with a normal FSM, I can certainly guarantee that everything should be working fine and if not, at least I can blame it on myself and there will be a genuine reason as to why it's not working as planned.

Fuzzy logic would be better suited to the second part of the game idea which would consist of the enemy being the hider and the player being the seeker. This is because it would be easier to have the enemy detect if the player can be seen, I would give the player dampened visions as well just to make it harder for the player, depending on how much of the player is seen the enemy could then either make the choice to change hiding place or move.

Having an FSM could potentially be a bad idea as well due to it leaving many gaps for errors, if an FSM is coded the way I intend to code it (using Booleans) all possible scenarios have to be thought about and covered; if not this could lead to the game crashing, or the incorrect State being used for the enemy.

Unity comes with a third-person template which I am going to be using, this template comes with a premade map and a player with controls already working. I will be using this template to ensure that I can put as much time into the main focus of the coursework as possible; the idea is that I could switch up the avatars and general aesthetics later as a personal project and that the Unity provided assets are only temporary placeholders. This package appears to use Cinemachine as well.

# Development: -

My main AI method will be an FSM and it is going to run off the basis of two Boolean variables that will have background coding behind them; these two variables will be to determine whether the ‘player can be seen’ or not and if the ‘enemy knows the location of the player‘ or not. These Booleans determine which state will be running at a given time and I am aiming to use if statements to check and cycle through each state.

Text, letter

Description automatically generated

The State Manager script will be an abstract script so that it can be easily derived/ inherited from, and it will contain a function that will also be abstract as I will have each state running that function and they all want to do a similar job. This means that I will spend less time trying to find a workaround for each State types’ scripts and have them all work in a similar manner that is coherent and understandable.

Text, letter

Description automatically generated

I will need a script that will manage the changing between the states, the current state will need to be defaulted to the Idle in the beginning so that the State Manager can actually change between the states, this script will essentially take in the active script for the ‘RunCurrentState()’ and will change the current state based upon which script has the active function.  
Text, letter

Description automatically generated

Idle will be the default state, it has the two Boolean inputs, and depending on whether these are true or false the script will either return itself for the state, or the script will return the appropriate state that has been predeclared within each if statement. Each state will follow this same process.

Text, letter

Description automatically generated

The search state is the state that is going to change the usual play style of the game, and this is because I am going to have it so that the enemy will know the position of the player every now and then which means the Player is going to have to stay on their toes and be a bit wearier of their positioning. This idea will be created with a Coroutine that sets the other important Boolean to true every now and then for a designated amount of time and I mainly want the enemy to go back to the wandering state after this. The enemy should also walk Faster in this state to give it more of a fighting chance at winning.

Text, letter

Description automatically generated

The wandering state is the main state that every state should default to if not in use, this is because whilst in the wandering state the enemy would realistically be constantly on the lookout for the player, and I want the enemy to be doing this for the majority of the game as it’s the main goal of the game- to find the player.

Text, letter

Description automatically generated

I expect to write more for the scripts than what has been planned in the pseudocode as this is just for the basic structures for each of the states and changing between them.

I want to make a Field of View (FOV) cone for the enemy and whether the player is within a certain viewing angle or not will set a Boolean to true or false. This Boolean is very important as it will be a deciding factor for the state changes which is a major part of how the game runs. For testing the FSM, I will be using trigger colliders at first and then I expect to expand on these and potentially turn one of the colliders into the FOV cone to make the seeking more realistic; I will keep one of the colliders as a catching collider. I will be using a guide to aid me in coding the FOV cone (2022).

I am using Unity’s NavMesh system which has not been coded or developed by me and is only a tool to help me with the creation of the game, I may add scripts to the system to help it function better and I will be using specific code to allow me to access the Navmesh using functions that have not been written by me. The NavMesh AI will be used for letting the Enemy wander around the map without walking into walls and for the searching and chasing of the player in an efficient manner. The NavMesh uses A\* Pathfinding so it’s not too far away from what I’ve planned and if I do find the time to add my own version of it I will.

# Evaluation: -

I originally intended to use an Ienumerator system where the states would have been Ienumerators instead of Monobehaviour scripts that have a state type. The main problem that I have with the methodology that I chose is that it consists of a large amount of logic checks in the form of if statements; although this was the easiest method for me to be able to visualise the way that everything would work together, it’s just quite easy to lose track of what is happening within some scripts. Luckily, I had made an attempt to keep certain variables within the scripts that they relate to. I tried this Ienumerator method before I wrote my pseudocode, and I went in without a concrete plan or idea which led to the failure of the construction of the code. If I had been able to implement the original idea almost everything regarding the states would be on one script which would mean that it would become rather hard to find specific areas of code so having the scripts separated helps me stay organised.

I have spent a lot of time stressing that the FSM wouldn’t be enough to get a decent grade and originally did a lot of research into making an A\* Pathing system as I am going to have the enemy seeker look for the player using the A\* pathing navigation. This was going to be my only AI method but then I realised that it didn’t fit the full functionality of the game and that I would need more than that to make the idea that I had. I also intended to use Unreal Engine which I am not as familiar with, so I was trying to place a lot of work upon myself when the FSM was the best-suited method for the coursework and my idea together. One way that this project could have been extended is by creating more states to transition between as this leaves the enemy with more behavioural room for a more natural response.

I have had really poor time management skills when it came to this coursework but despite this I am also proud with how flawlessly my FSM runs, this is probably because I put more effort into making sure that there shouldn’t be any loopholes within the respective state scripts and that if you cross-compare my pseudocode with my scripts there is clear evidence that I have expanded upon what I originally intended to write. It took a lot of time to make sure that there shouldn’t be any room for error when the game is running. I should have made more solid plans from the very beginning as this would have helped to ensure that a smoothly run coursework piece would have been provided.

In conclusion, my finished project isn’t what I originally intended for it to be, but I feel that it shows that I am competent in creating an almost flawless FSM system and I do intend to go away with the knowledge I have picked up along the development process and expand upon it after handing in. I enjoyed making this piece and now that I have made it once it should be far easier to incorporate other FSMs into other games in the future.

# References: -

# (2022). [Video]. Retrieved 10 March 2022, from [Field of view visualisation (E01)](https://youtu.be/rQG9aUWarwE?t=490)

