

IGB383 Assignment 1

By Christian Milicevic, Student Number: N10194444



April 2, 2020

# Statement of Completeness

**A:** 4 Different types of NPC’s that respond to a given set of Triggers and Place Markers = Completed

* Has 4 different types of agents (RegularEnemy, CowardEnemy, Sentry and Roamer
* 2 Sets of triggers (2x AgroTrigger and 1x Hide Trigger)

**B:** Implementation of Finite State Machine (FSM) for the reactive agents = Sort Of

* Different types of colour materials are displayed on the agents, when the player triggers a *OnTriggerEvent* object
* FSM state machines work but are not done in the format of a 2D Array, instead they have been coded via multiple scripts (*Enemy*, *RoamerEnemy*, *CowardEnemy* and *SentryEnemy*)
* I choose to get them working even if it wasn’t via DFA format, so I could get the assignment functions and completed
* I have also got the 2D array’s with values of each state for the 4 enemies is still displayed in the code. This is located in the *Enemy* script.

**C:** Pathfinding using Greedy and A\* Search algorithms = Completed **(but has an issue)**

* Both algorithms are implemented and written out via C#, however Greedy does have some stackoverflow exemptions due to the recursive function (which I can’t seem to figure out due to time constraints)
* Besides that they work but the build can be a little laggy due to the *stackoverflow* exemptions.

# Path Finding

# Greedy Algorithm

# Greedy works by choosing the best possible answer at the time to get to the goal and constantly keeps moving through the steps of code until it gets to goal. Hence why the algorithm is called greedy and is seen as a “brute force” algorithm.

# A\* Search Algorithm

A\* is a pathfinding searching algorithm and calculates the shortest distance between the initial state (the source) and the goal (the final state)

The algorithm is made of three components which are:

1: **G** cost

Which is the cost of moving from one node to the next, it sums up all the nodes that you have vested since leaving the source

2: **H** cost

The H cost (also known as the heuristic value) is the estimated cost of moving from the 2nd last node to the final node (the final state).

3: **F** Sum

The **F** sum is the result of G and H

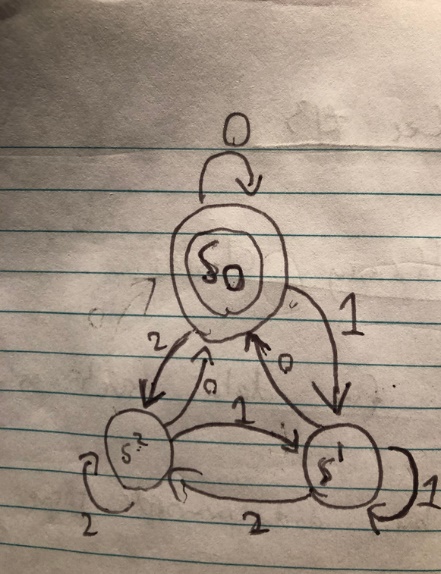
**G + H = F**

# Finite State Machines

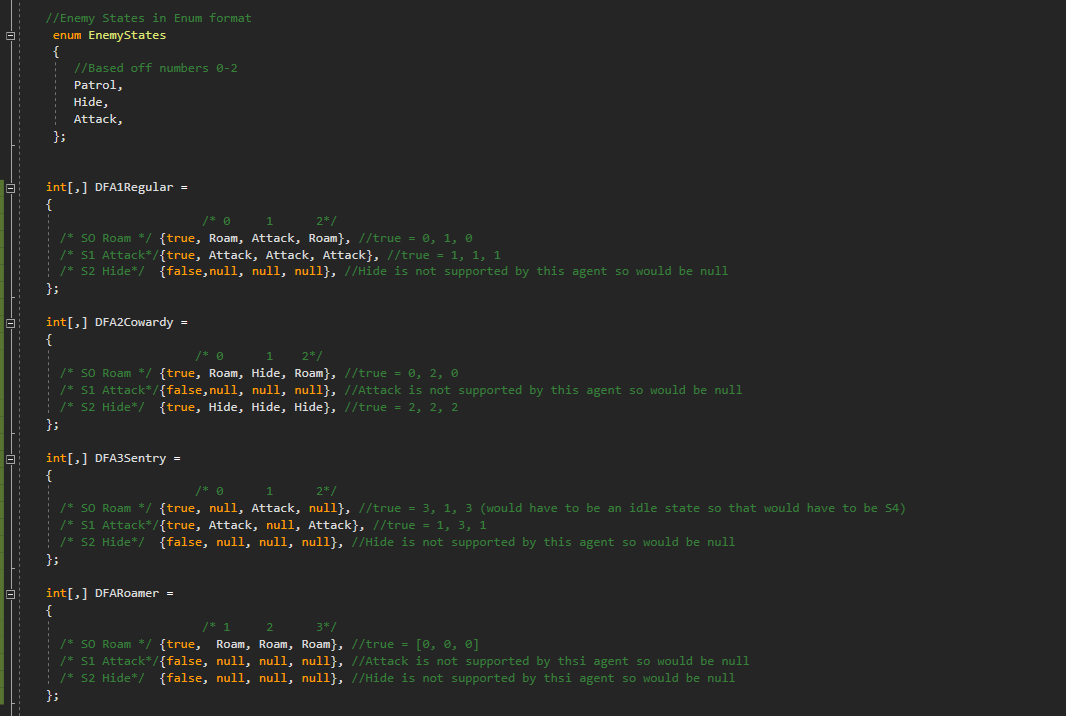
As you can see in the following image this was the diagram that I drew from the workshop to show that I knew the understanding of finite state machines and since we had three states being:

* S0 Roam
* S1 Hide
* S2 Attack

I drew this diagram to replicate that:



Below you can also see the format that I drew for the 2D arrays in code for C#. Even though I never implemented them I still put/left them in to show that I sort of understand how to put them in digital with C#, just not completely.

C# DFA Diagram Logic:

Trigger diagrams for each NPC:



The red circled are the *RegularEnemies* they have the following transitions for their states:

* Roam on Exit
* Attack on Entry
* Hide is null
* They are triggered via the red boxes

The blue circled are the *CowardEnemies* they have the following transitions for their states:

* Roam on exit trigger
* Hide on entry trigger
* Attack is null
* They are triggered via the blue boxes

The black circled are the *SentryEnemies* and they have the following transitions for their states:

* Attack on exit trigger
* Idle on entry trigger
* Hide is null
* Attack is null
* They are triggered via an enum timer

The enemies that are not circled are the *RoamerEnemies* they have no triggers and only use the *Roam* state the rest are null.

# Conclusion

In conclusion since I am taking CAB301 (the prerequisite) as the same time as this unit (IGB383) I found the learning curve a little hard but not too bad. The process of algorithm is interesting to see how which one is more efficient/effective.

Which we saw in the A\* and greedy algorithm, one being a brute force style algorithm which continues to step through until reaching the goal. And finally, A\* which executes with a better pathfinding method (finds the shortest route) and returns once it has found the goal, instead of stepping through the whole algorithm like Greedy.

Also,

* algorithms can be a pain to bugfix…
* DFA’s are simple by logic but are weird to code (it’s more trying to get enum values in the 2D arrays were the hardest part)

Following extra stuff, I put in the game that I made was the:

* Lum particle system and PNG’s
* Player trial
* Rain particle system and PNG’s
* The rest was from the asset store from Unity

Thanks