Task 3 – Design and Implement a State Machine for an NPC

**Point 1**

The environment being simulated here is very basic. There will be a player-controlled agent which moves about the node map by clicking of the left or right mouse button. The left mouse button uses the Dijkstra’s pathfinding algorithm to find a node and the right mouse button uses the A-Star pathfinding algorithm. I set this feature up to display both algorithms working for the first and second task of this assessment. I will include multiple NPC’s that will wander randomly around the map and either follow or flee from the player.

**Point 3**

Wander behaviour – The wander behaviour enables the NPCs to roam around the map moving from one starter node to a random end node at a set speed and colour which I have dictated in the behaviour functionality. This behaviour is constantly checking if the agent has completed its current path. This is a function that checks if the agent’s current path is empty and returns a true or false value. If this function returns true the agent calls its “GoToRandom” function, this creates and stores a random node from the node map in a Node\* variable. It then tells the agent to go to this node.

Follow behaviour -

Task 4 – Design and Implement a Decision Tree for an NPC

Task 5 – Write an A.I. Game Strategy Report

Behaviour trees are a popular AI technique used in many games to model the behaviour of NPC characters. Halo 2 was one of the first mainstream game to use behaviour trees and they started to become more popular after a detailed description of how they were used in Halo 2 was released. Ultimately, it is a tree of predefined node types aimed to represent how “something” behaves. Each node returns either Success, Failure or Running. One of their main advantages is that they are easy to understand and can be created using a visual editor.

* **Composites,** which are branch nodes that contain child behaviours. Composite Behaviours are nodes that contain one or more children and dictate how they are run and when to stop. Two common types of composites are a selector and sequence. A **Selector** is a node that returns success if one of its child nodes returns success without executing its remaining child behaviours. If a child returns failure, then it executes the next child behaviour. If all child behaviours return failure, then the selector returns failure. Acts as an ‘OR’. A **Sequence** is a node that returns success if all its child nodes return success. If a child returned failure, then it would return failure and not execute the remaining child behaviours. All child behaviours must be a success for the sequence to return success. Acts as an ‘AND’.
* **Actions,** which are Leaf nodes in the tree. An Action behaviour is a behaviour that “does” something (e.g. move forward or perform this animation), generally always return success.
* **Conditions,** which are also Leaf nodes in the tree. A Condition Behaviour is a behaviour that “asks” something (e.g. is health empty or can see enemy), returns success or failure.

https://opsive.com/support/documentation/behavior-designer/what-is-a-behavior-tree/

https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work

<https://blog.zhaytam.com/2020/01/07/behavior-trees-introduction/>

https://en.wikipedia.org/wiki/Behavior\_tree\_(artificial\_intelligence,\_robotics\_and\_control)