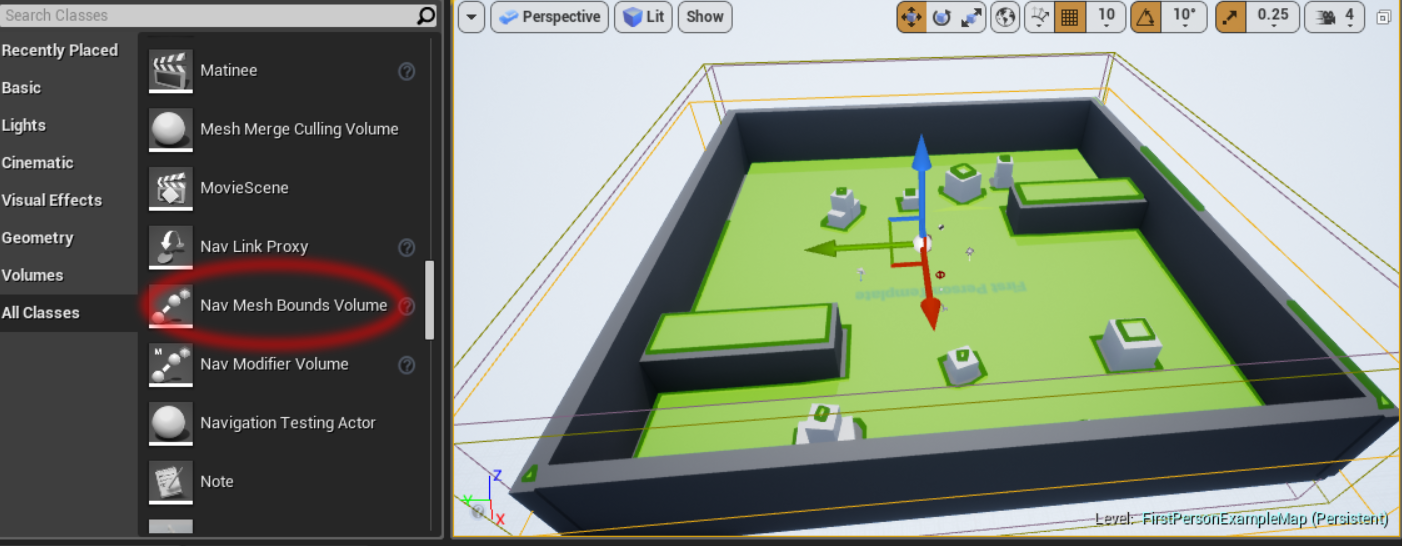
Week 8 – AI Basics

**Section 1 – Adding Navmesh**

*Navmesh is a very powerful pathfinding tool used by UE4, adding one will allow agents to path from any position on the navmesh to a connectable position.*

1. In editor use the left hand panel to find and add a Nav Mesh Bounds Volume by dragging it into the scene.
2. Scale it up to encompass the entire level
3. Press P and the navigatable areas of the level should highlight green as below

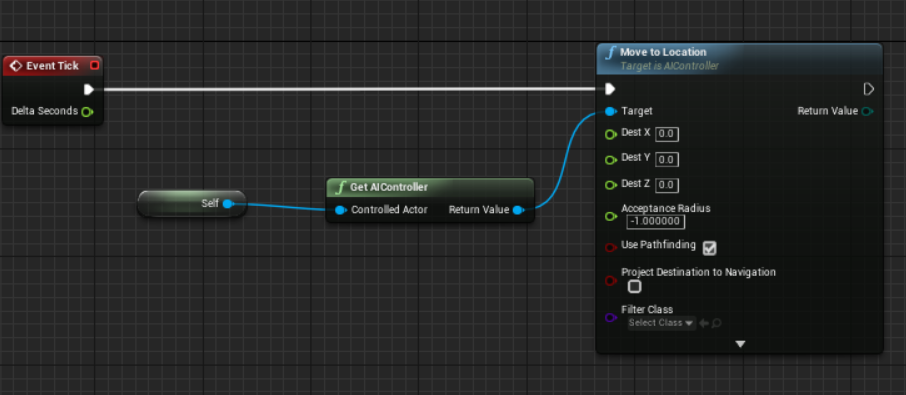


**Section 2 – Testing the navmesh**

*By Default anything that derives from Character has navigation set up already.*

Open up your NPC blueprint

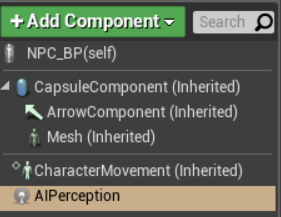
1. Create a Get AIController node
2. Get a self node and plug it into Controlled Actor pin
3. Drag out of Get AIController and select MoveToLocation plug it into Event Tick.
4. Right click on the yellow Dest Pin and select Split Struct Pin (to give 0,0,0)



Run the scene and the NPC should move towards the centre of the map.

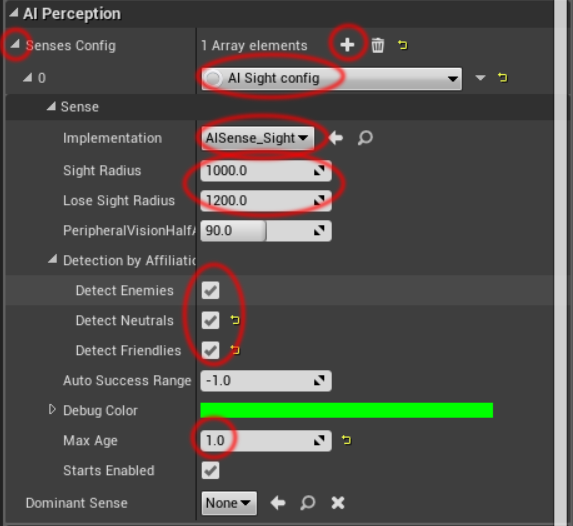
**Section 3 – AI Perception**

Add an AIPerception component to your NPC



In the Details panel for your new AIPerception component:

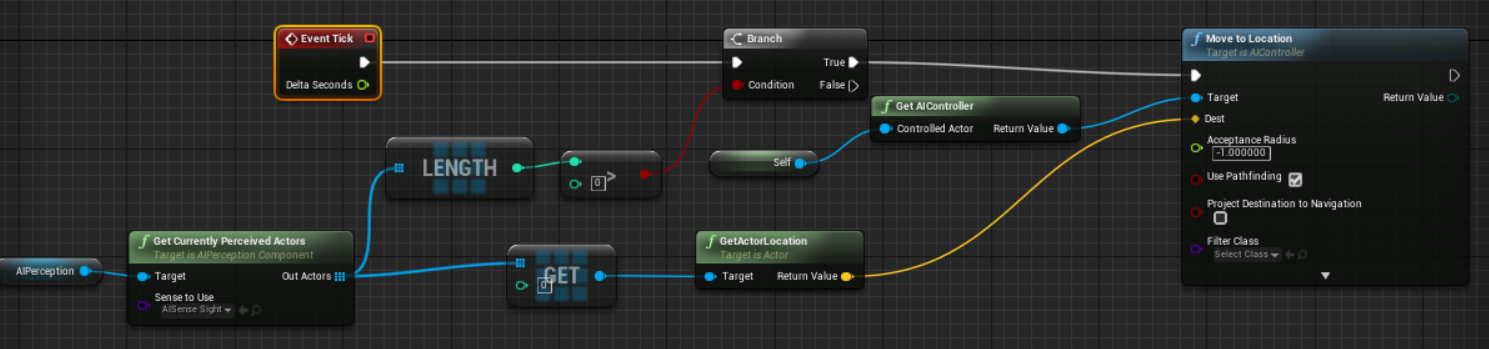
1. Add a new Senses Config
2. Select AI Sight config
3. Lower the Sight Radius and Lose Sight Radius
4. Allow detection by Affiliation to be Enemies, Neutrals and Friendly’s
5. Set the Max Age to 1.0



Section 4 – Basic AI Perception Example

*In this example we are going to have basic stealth AI, the NPC will move the location that they lost sight of the player. Feel free to tweak the AI Perception settings and NPC max move speed.*

1. *Drag your AIPerception component into the graph*
2. *Drag out a Get Currently Percieved Actors node*
3. *Set the “Sense to Use” to “AI Sense Sight”*
4. *Drag out of the array of actors pin and create a Get (copy of) node with index of 0*
5. *Get the actors location*
6. *Plug it into the Move To Location node we created earlier (may need to Recombine the Struct pin of dest)*
7. *Check that the length of Currently Perceived actors is greater that 0 (else it will move to centre when null is passed through)*



*Today we have created a very crude implementation of AI, it would be worth exploring using a state based implementation (see challenge 3). Next week we will be looking at behaviour trees, we will be using what we learnt today about navigation and perception and applying it to a much more robust AI system.*

Challenges

1. Set up an array of waypoints and get the enemy to patrol until it sees the player
2. Create a simple prototype of a stealth game. You could use pickups as objectives, you lose if the enemy spots you.
3. Refactor your code to use a state based implementation of patrolling and chasing.
   1. Basic – Use enums for the state, using a switch statement to run different behaviour
   2. Advanced – Create base State class and derive other states from it. Run the current state, it returns what new state to set (or itself if no change).