**Behaviour Tree**

Create a new character BP named NPC and give it a static mesh and an animation BP. Create an AI controller BP. The AI controller is like the brain of the AI character. Call it NPC\_AI. Make a Behaviour Tree named NPC\_BT and a blackboard NPC\_BB. Open the NPC BP and click on the self on the left-hand side and in the controller section in the right-hand section select the NPC\_AI option.

Open the NPC\_AI node and extend the begin play and extend it and add a run behaviour tree node and set it to NPC\_BT. Open the NPC\_BT and set the blackboard as NPC\_BB.

A behaviour tree is a decision-making tree. The root can have only one output but selectors and sequences can have several. Selectors select based on a condition and only execute a set of functions at a time. Sequences execute multiple tasks sequentially.

The events in a behaviour tree are executed based on priority. Priority decreases from top to bottom and left to right.

Add a navigation mesh and press P to see the green regions where the NPC can navigate.

Let’s try making a random navigation NPC –

It does 2 tasks – find a location to navigate to & navigate to that location.

Go to the behaviour tree, extend the root and add a sequence. Extend the sequence and in the left (as that has higher priority and finding the location comes first) you’ll be adding the task to find the location. However, finding a location isn’t an inbuilt task so click the create new task button. In the new task’s event graph add 2 nodes – Event Receive Execute Ai node which is the node that activates when the event/task is under execution and add the finish execute node without which we can never get out of the task.

Add a new node called get random point in navigable radius. Extend the controlled pawn pin of event execute at node and add a get actor location node and plug the return value to the origin pin. With this we get the random location but we need a method to communicate between tasks.

This is where Blackboard comes in. Open the Blackboard and add a new key called TargetLocation of vector type. Then save it. Go back to the behaviour tree and make a new variable (any name) say vector and set the type as blackboard key selector. Add in a reference to the variable, extend it and add a set blackboard value as vector. Hook in the Random Location pin into the value and hook its input exec pin to the receive execute node and the output exec pin with the finish execute node and check the success pin so as to tell that the whole thing got executed successfully. Rename the task as find random location. Make the vector variable public so we can use it outside the class.

Go back to the behaviour tree graph extend the sequence and extend the sequence and add the Find random location node (because we made the vector variable public, we can see it in the node). Extend the sequence node again and add the move to node and the move to must be set to target location. Extend it again and add the wait node and set the wait value to 3s.

**AI Perception**

Between the root and the sequence node of the BT add a selector node. Rename the sequence to Go to Random Location. Add another sequence node to the right of the Go to Random Location sequence node and rename it to Chase Player. Add a new task and in its BP add an event execute ai node and finish execute node. Add a get player character node and extend it and add a get actor location node. Extend it and add a get random position in navigable radius. Set the radius to about 100 units. We need to store it to a blackboard key. Make a new variable called Vector of Blackboard Key type. Add a get Vector node and extend it and add a set blackboard value as vector node. Extend the key and hook it to the random location pin. Hook the output exec pin of the event execute ai pin to the input exec pin of the set blackboard value as vector. Extend the output exec pin of the set blackboard value as vector node to the input exec pin of the finish execute node and check the success box.

Rename the task to find player location and go to the BT and extend the chase player node and add a find player location node. Extend it again and add a Move To node. For both these nodes the target would be TargetLocation.

To trigger this chase player sequence the npc should somehow trigger it. To do that we go to the BB and make a new Boolean key called CanSeePlayer. Now we need to give both the sequences a decorator so that they can trigger the respective actions. A decorator is some condition that needs to be satisfied so as to execute the node. Add a decorator of blackboard type to the Go to Random Location node. Change its name to Can’t See Player, set key query to Is Not Set and blackboard key to CanSeePlayer. Add another decorator of blackboard type to Chase Player and rename it to Can See Player. Set the key query to Is Set and the blackboard key to CanSeePlayer.

So now control goes from selector to go to random location checks the decorator condition and if it fails goes back to the selector and then goes to chase player node and checks the condition and so on.

To set the Boolean variable we will use the perception system. Go to the NPC\_AI (ai controller) and add a new component called AIPerception. Click the + sign beside the Senses Config in the right hand side and set the first entry to AI Sight Config. Tweak the settings if you want. For test purposes we set the detection by affiliation to detect friendlies, neutrals and enemies.

We are not going to use perception based on object instead we are going to use the stimuli method so the same technique can be later on used for applying to objects so that auditory stimuli etc can be used to distract guards etc.

Go to the third person character and add an AIPerceptionStimuliSource. Select it and in the right-hand box add a register source for the senses. Set the entry to AISense\_Sight.

Go back to NPC\_AI and click on the AIPerception on the left-window and in the event graph right click and select the add event for AIPerception > Add on Target Perception updated. Extend the stimulus and add a break stimulus node. Extend the actor pin and add a cast to third person character node. Add a get blackboard node and extend it and add a set value as bool node and hook its input exec pin to the cast to third person character node’s output exec pin. Hook the successfully sensed pin of the break stimulus node to the bool value pin of the set value as bool node. Extend the key name pin and add a make literal name node and set the value as CanSeePlayer (Exactly how it’s in the blackboard).

When we run this the enemy follows us but the moment he looses sight of use he runs to random points again which is dumb behaviour so this needs tweaking.

**Chasing the player**

Right now, if the NPC is randomly roaming and sees the player it still continues to run and then runs to the place where it last saw the player. To instantly make this better go to the BT and click on the can’t see player decorator. On the right-hand window, we see that the notify observer is set to On Result Change and observer aborts option is set to none. This means that even if results change it doesn’t abort and the rest of the tasks in the sequence still execute. Change the observer aborts to self which means that it aborts itself and goes back to the parent i.e. selector in this case. The selector then chooses which to execute based on the decorators.

Another option is to set it to lower priority in that cases it aborts the actions whose priority is lower than the current task. By default, the tasks on the left are of higher priority than the tasks on the right. So, if we set it to lower priority the chase player aborts as it’s of lower priority.

If you set it to both then both the tasks i.e. go to random location as well as chase player aborts and control goes back to the selector which then decides what to do. This is a safe way of doing it as this leads to re-evaluation of the decorator conditions.

Now if we run into the NPC’s field of vision, he abandons his current actions to chase us but still we see that he does it in straight lines which is unnatural. This is because of spasmodic updating of the player location in the Move To node we need to try to make it look more natural.