Flying AI Tutorial

Create a BP of type character and name it flying AI BP. Set the AI controller to My\_AIC (that’s the AI controller we created in a previous tutorial). Near the event begin play node add a get AI controller node and extend the Controlled Actor pin and add a self node. Extend the return value pin and add a cast to My\_AIC node and hook its input exec pin to the output exec pin of the event begin play node. extend the as My\_AIC pin and add a setup AIC (function) node and hook its input exec pin to the output exec pin of the Cast to My\_AIC node. Extend the BT pin and promote it to a variable called BT and extend the Pawn Sensing pin and add a get player pawn node.

Create a behavior tree called FlyingAI BT. Let’s use the previous blackboard which contains: MoveToLocation (vector), CanSeeTarget (boolean), RecentlySeenTarget? (boolean), Target (actor), Origin (vector), SelfActor (actor), Cooldown (float), TeleportLoc (vector). Open FlyingAI BP and set the BT variable to FlyingAI BT. Click on the character movement component and set the gravity to 0 and also click on the capsule component and set the gravity to 0.

Open the FlyingAI BT and extend the root and add a selector. Extend it and on the right add a wait node and then extend the selector and to the left of wait add a sequence and name it to random roaming sequence. We need to find a random roam location.

Create a new BTT task and rename it to FindValidFlyingLocationBTT and open it. Add an event execute AI node and extend the controlled pawn pin and promote it to a variable called Controlled Pawn Ref. To find the location create a new function called GetNewProvisionalRoamLocation and check the pure option to make it a pure function so that it has no input exec pin. Open the function definition.

We want the flying pawn to move only in the XY plane i.e have no elevation so we cancel the Z axis movement to do that we could either lock movement in the Z axis or just make vectors with Z axis value = 0. Add a get random unit vector node and split the struct pin, add a make vector node and hook the X & Y value of the get random unit vector node to the corresponding X and Y values of the make vector node. As we stripped of the Z it’s no longer a unit vector so we need to normalize it. Extend the return value of the make vector node and add a normalize node and extend its return value and add a vector \* float node. To make the pawn cover random distances along with random directions the float must be a random number too. Extend the float value pin of the vector \* float node and add a random float in range node. Extend the min pin and promote it to a variable called MinDistance and extend the max pin and promote it to a variable called MaxDistance and set them to say 500 & 1000 and make them public so that each character BP that inherits the BT can set different values to this. Extend the return value of the vector \* float node and add a vector + vector node. Add a get Controlled pawn ref node and extend it and add a get actor location node and hook its return value to the other pin of the vector + vector node. Hook the output pin to the return node and name the output as NewProvisionalRoamLoc . This gives us a random vector location in the same plane the pawn is.

Go back to the event graph. Although we have a roam location we don’t know if it’s valid i.e. the path to the location might contain obstacles or might be out of bounds etc. To check for obstacles here we do a box trace (To ensure that the whole pawn can pass through). Add a GetNewProvisionalRoamLocation node and extend the return value and promote it to a new variable called ProvisionalRoamLocation and extend the output exec pin of the set Controlled Pawn Ref node and hook it to the input exec pin of the ProvisionalRoamLocation node. Extend the output exec pin of the Set ProvisioanlRoamLocation node and add a box trace by channel node. Extend the Controlled Pawn Ref value pin and add a get actor location node and hook its return value to the start pin of the box trace by channel node. Hook the value of the set ProvisioanlRoamLocation node to the end pin. Either manually set the box half size (avoids the additional step of finding it) or add a get controlled pawn ref node and extend it and add a get actor bounds node and make sure to check the Only colliding components option otherwise all other components like bounding boxes etc. will also get factored in. extend the box extent pin of the get actor bounds node and hook it to the half size pin. Make sure to check the Ignore self option on the box trace by channel node or else the collision might hit on the object itself. If you want the pawn to move even if other actors might be in the way then make an array of those actors and hook them to the Actors to ignore pin. Set the draw debug type to for duration and extend the output exec pin of the box trace by channel node and add a finish execute node and check the success pin.

Now on simulation a box trace will be done in random directions of the XY plane of the pawn.

Delete the finish execute node. extend the return value of the box trace by channel node and add a branch node. If there’s a hit then we need to try again but you should only try it a particular number of times or else there’s a chance for an infinite loop to occur. So create a new variable called counter of int type. Add a get counter node and extend it and add an increment integer node and hook its input exec value to the true pin. Extend the output exec pin of the increment int node and add a branch node extend the condition pin and add an int <= int node and extend the A pin and hook it to the get counter node and set the B value to say 25. Extend the true pin and hook it to the input exec pin of the set ProvisionalRoamLocation node so that a new loc is calculated each time. Extend the false pin and add a finish execute ai node but uncheck the success pin as we couldn’t get a location despite the iterations. Now to set the blackboard key MoveToLocation with the value you could create a variable MoveToLocationKey of type BlackboardKey and make it public and near the false pin of the branch node connected to the box trace and add a get MoveToLocationKey and extend it and add a set blackboard value as vector node and extend its value pin and add a get ProvisionalRoamLocation node and extend its output exec pin and add a finish execute ai node and check the success pin.

Now in the Flying AI BT extend the Random Roaming Sequence node and to the extreme left add a Find Valid Flying Location BTT and on the right hand window the MoveToLocationKey value should be set to MoveToLocation.