**Game AI**

Almost all enemies within the game have some form of reaction to the player. Some turn themselves to aim at the player, others run towards them, and some actively try to block the players progression. All of these have one thing in common- a form of AI. To implement this, we have opted for the use of collider checks and Unreal Engine’s built in navmesh system for pathing the enemy to the player. When an enemy has the need to target the player, a sphere (or frustum) collider will be used as the sight area of the enemy. This will be a different shape and different size depending on which enemy is involved. Once the player enters these regions, the enemy will attack depending on its pre-determined behaviour.



Figure - Regional Enemy Eyesight

**Lobuzz AI:**

**Yellow**

The yellow Lobuzz is completely stationary and as such does not require any pathfinding or attack AI. Its only behaviour will be to float in place.

**Blue**

The blue Lobuzz will follow a predefined patrol route. To do this the Lobuzz will be given a spline to follow to define its patrol route. It will then follow this constantly.

**Green and Red**

Upon seeing the player, the Lobuzz will move towards the player. this can be done easily by using a sphere collider around the Lobuzz and upon a player entering that sphere the Lobuzz should path find to the player using the nav mesh. The red Lobuzz will stop upon reaching the player and explode after a delay instead of colliding with the player.

**Domino AI:**

The Domino will move sideways to block the player within a region, so it will have to move so that it is in front of the player on a horizontal plane. The Domino does not move forwards and backwards so will not use the nav mesh for path finding. It will use basic vector movement.

**Shew Caravan AI:**

Before being attacked, the shrews will follow a predefined path created by a spline. When one of the shrews is killed, all the shrews behind it will panic. This will be done using a Boolean variable, with each shrew calling a set function in the shrew behind. In order to make them move randomly, they will use a nav mesh to path towards a random position which will then change after a certain amount of time, bouncing off walls. This will make them appear to run around randomly and panic.

**Dizzy Charger AI:**

This enemy will charge directly at the player without consideration for obstacles. As such it will not use a nav mesh, instead following a vector directly to the player. If it collides with an obstacle it will die.

For the blue and silver enemies, they will generate a single vector to the player upon them entering a radius around the enemy, using a sphere collider. The red and gold variations will generate a new vector every set number of seconds to correct their path and aim at the player.

**Catapult AI:**

The catapult will not move but will receive the location of the player and fire a rock at that location. This will be done using a sphere collider so that while the player is within range the catapult will fire at them.

**Boomerang AI:**

The boomerang fires at the player following a wide arc. It will aim the furthest point of the arc from itself at the player and use its distance from the player as the diameter of the arc. While flying, the boomerang will generate a force left or right in local space to move itself towards the player.

After moving a distance equal to diameter, it will land. The blue variant will apply an additional local force following a sine wave in order to move back and forth while flying.

**Path finding in Unreal Engines**

When implementing the AI described above, we will be giving each enemy a blueprint. With this, we will also attach a nav-mesh to the region the enemies will be allowed to operate in. An example of which is shown below:

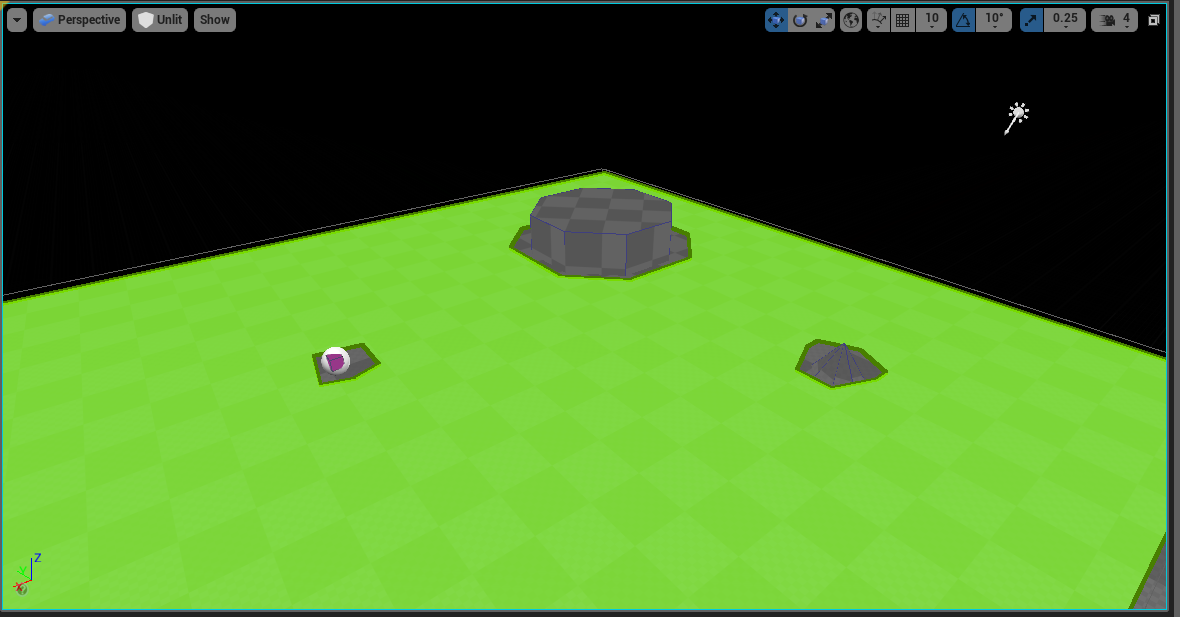


Figure - Unreal Nav-Mesh. Green: Area enemy can path find around.

Getting the enemy to use a navmesh requires writing a C++ class or a blueprint so that the enemy moves in a way that utilises the mesh. This is simple due to Unreal Engine’s pre-made AI systems, however, customizations will be made to ensure enemies behave as specified

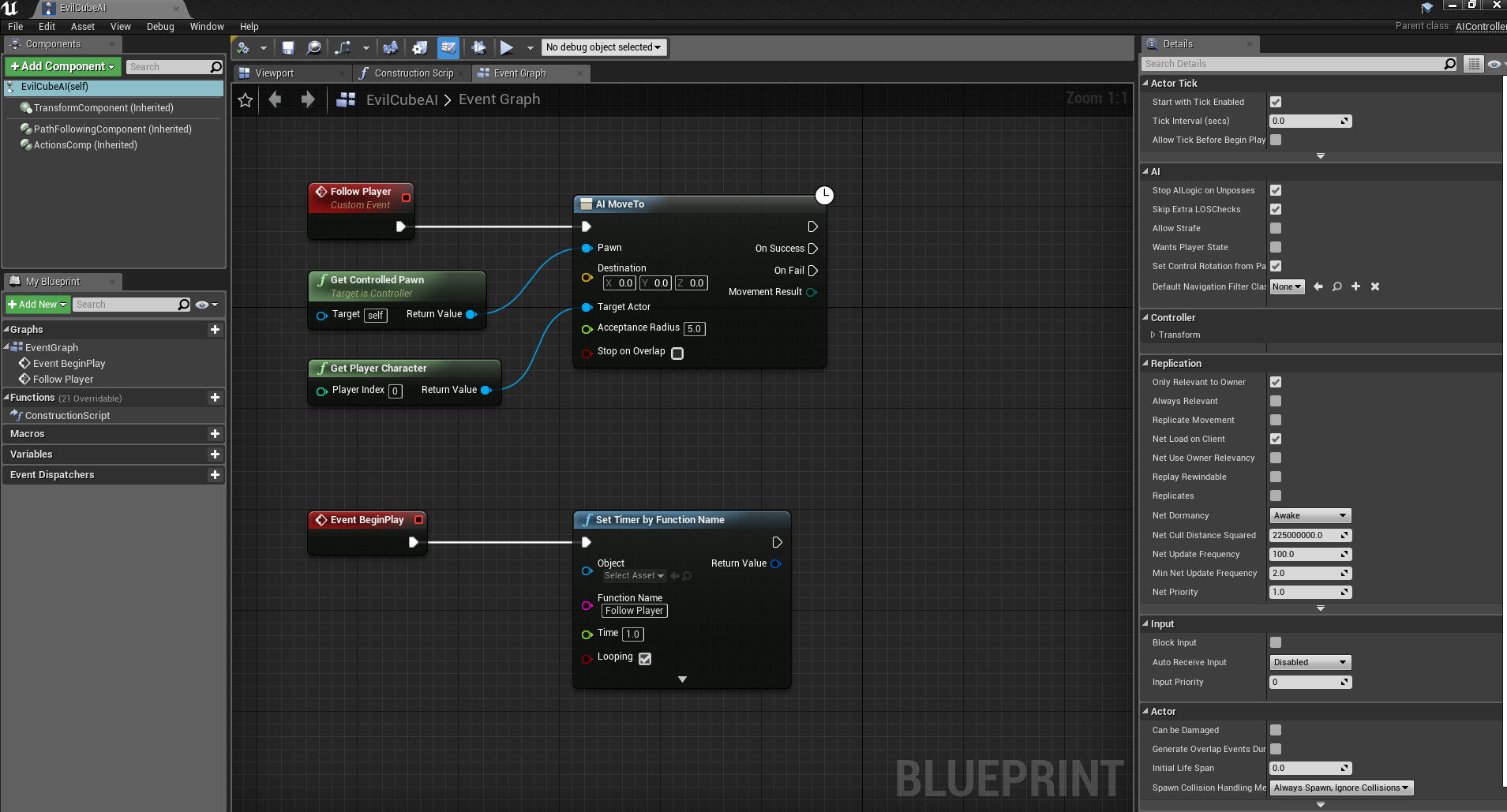


Figure - Blueprint displaying functional AI module.

**Implementation of Pathfinding**

Shown below basic pathfinding AI that will continuously chase the player while they are in view..

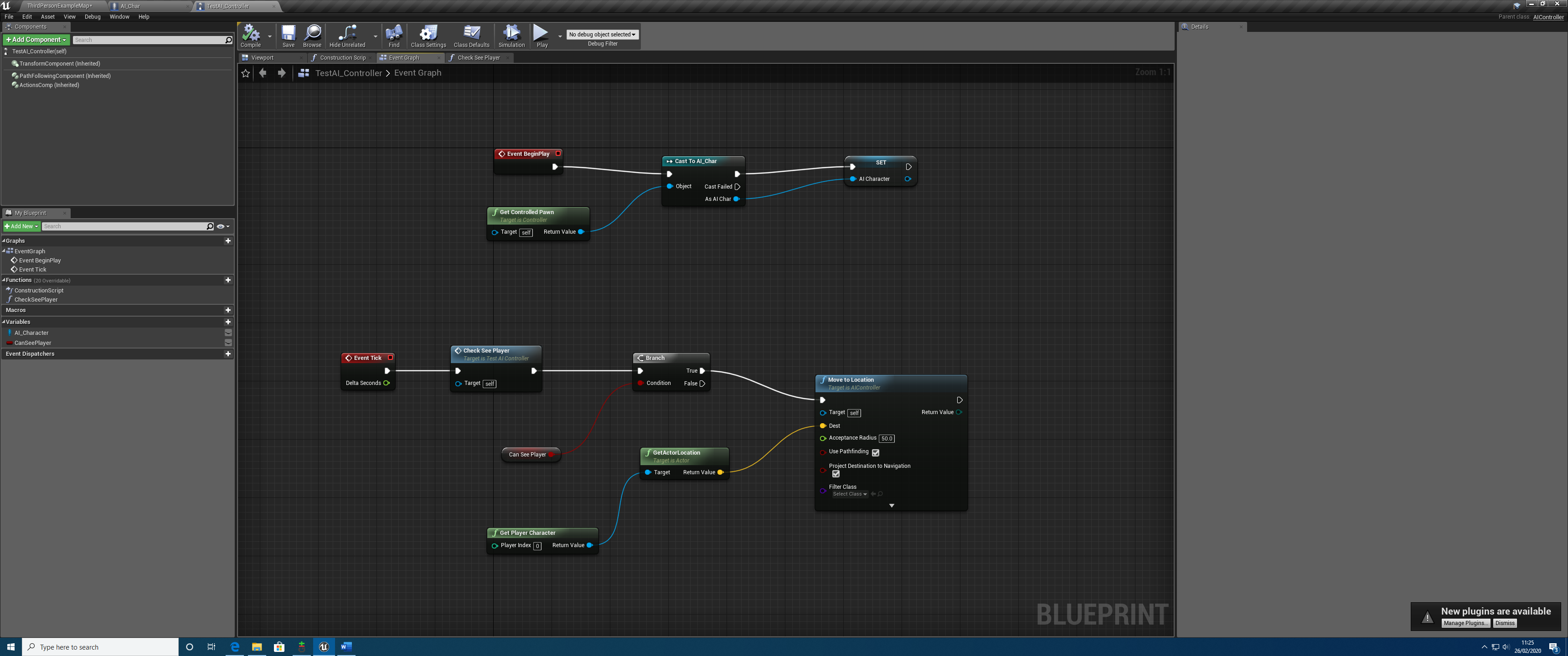


Figure - Blueprint displaying functional AI module.

The image below is the function that checks if the player can be seen by the character

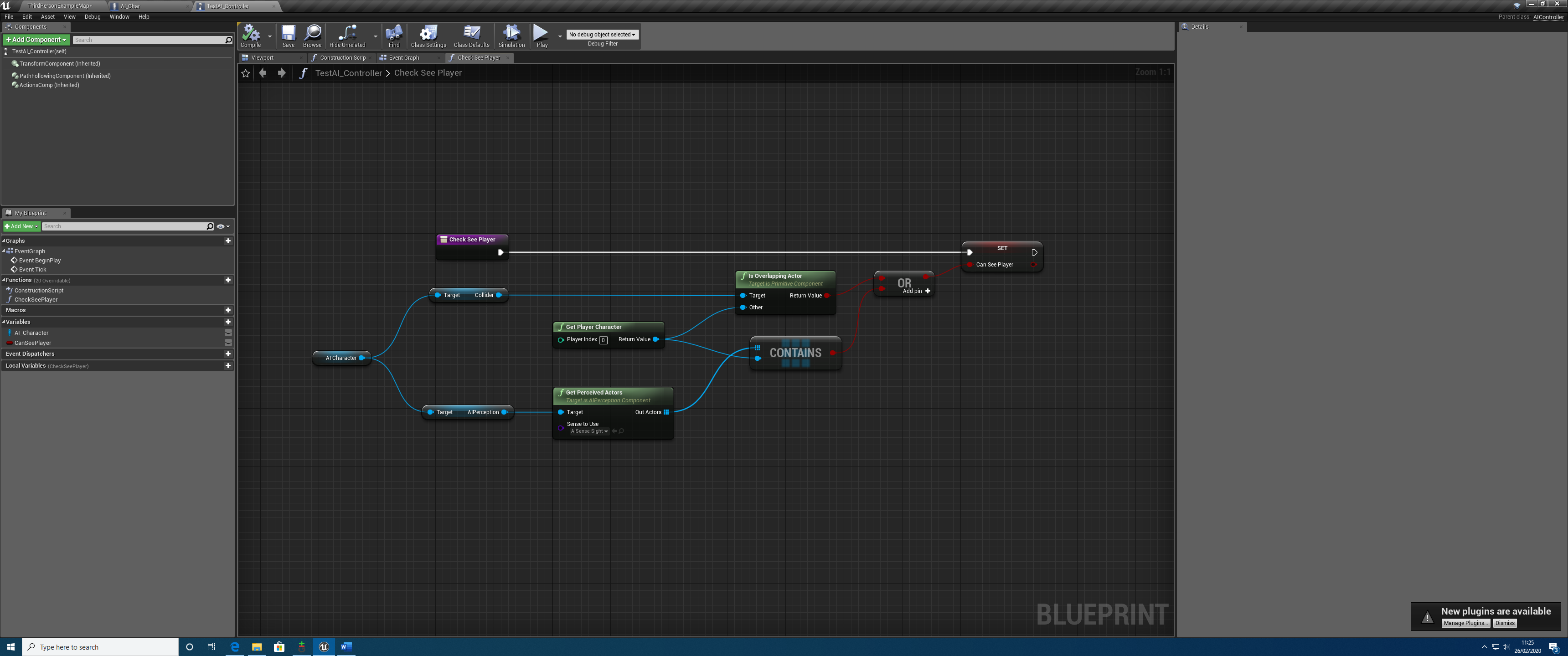


Figure - Blueprint displaying functional AI module.

This image makes the character move to the place the player is first spotted, then check again for the player location and move there. This creates an AI that will only update the path target once it finishes its last move, and could work for something that is unable to stop its own charge.

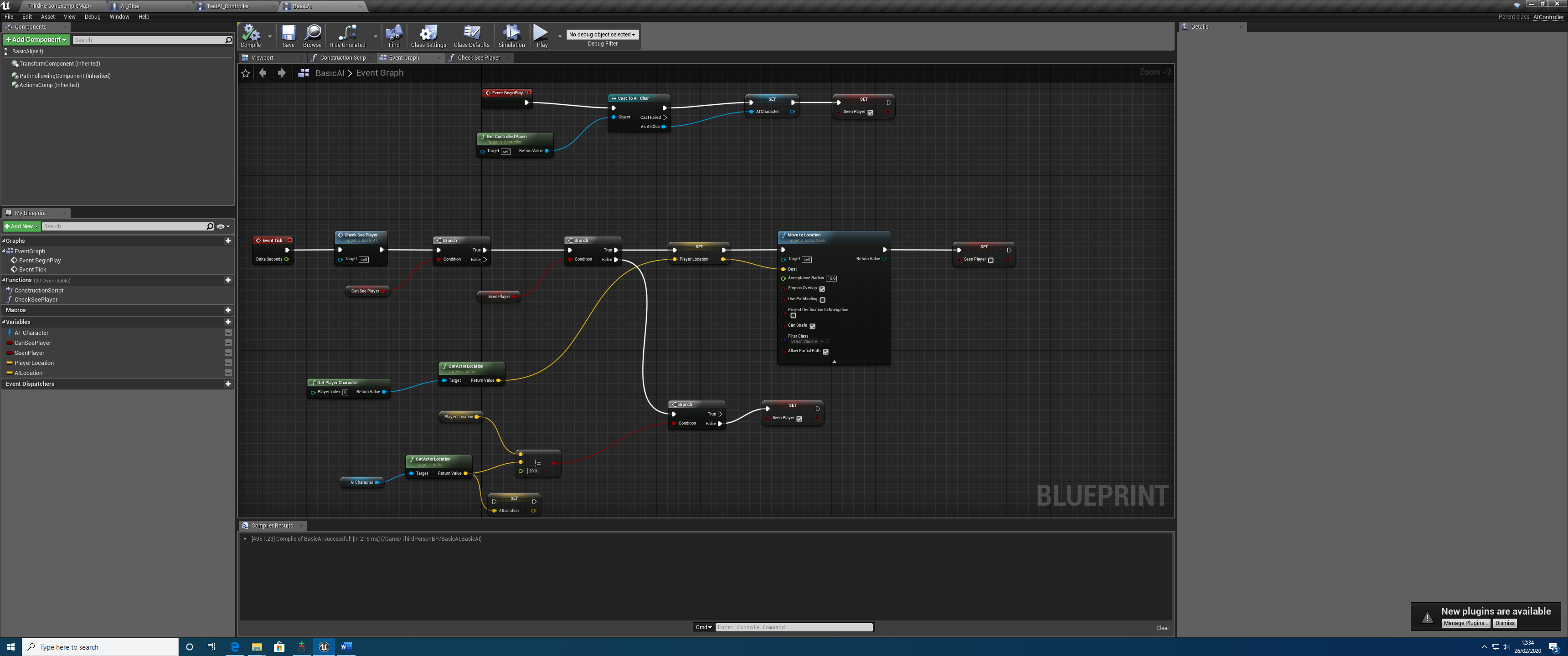


Figure - Blueprint displaying functional AI module.