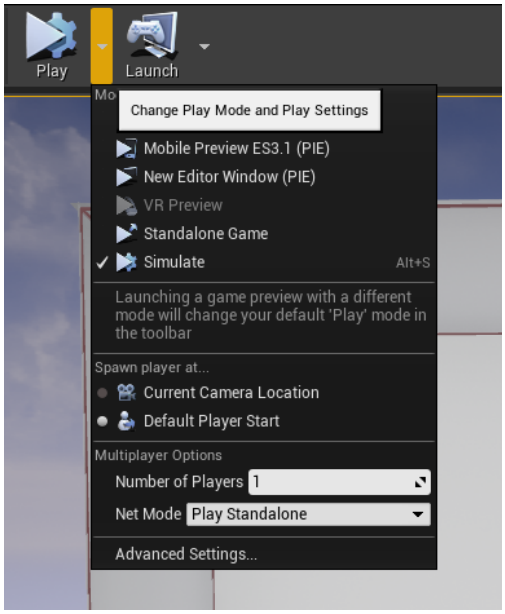
Agent Flocking Simulation Tutorial

The flocking engine itself can be quite confusing to use since a lot of the controls for the simulations are hidden in different places. I have created this tutorial sheet to guide you on how to use the class-based and Niagara simulation. There are 2 maps that can be found in the content browser on the bottom left within a folder called maps.

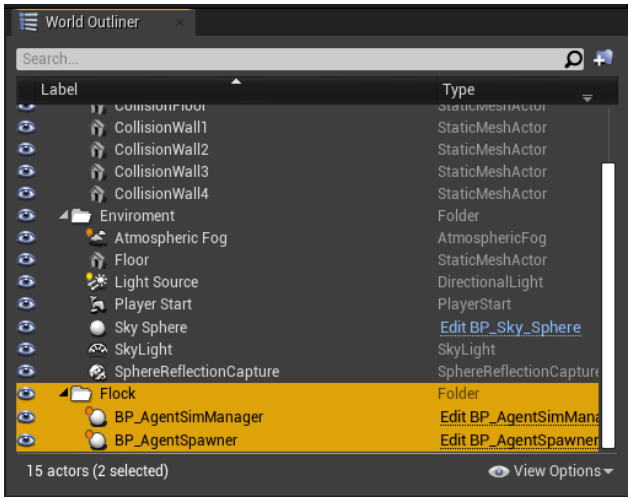
I should also add that all the code is fully commented, so explanations on how different parts of the project work can be found within the CPP files. The HLSL code for the compute shader simulation can be found in the “SimulateAgents” module script. This is also fully commented and explained.

Class-Based Simulation

To start, I would suggest changing the play mode to simulate as this will make it much easier to access the world outliner on the right side as most of the settings that will be changed for the simulation are within this menu.

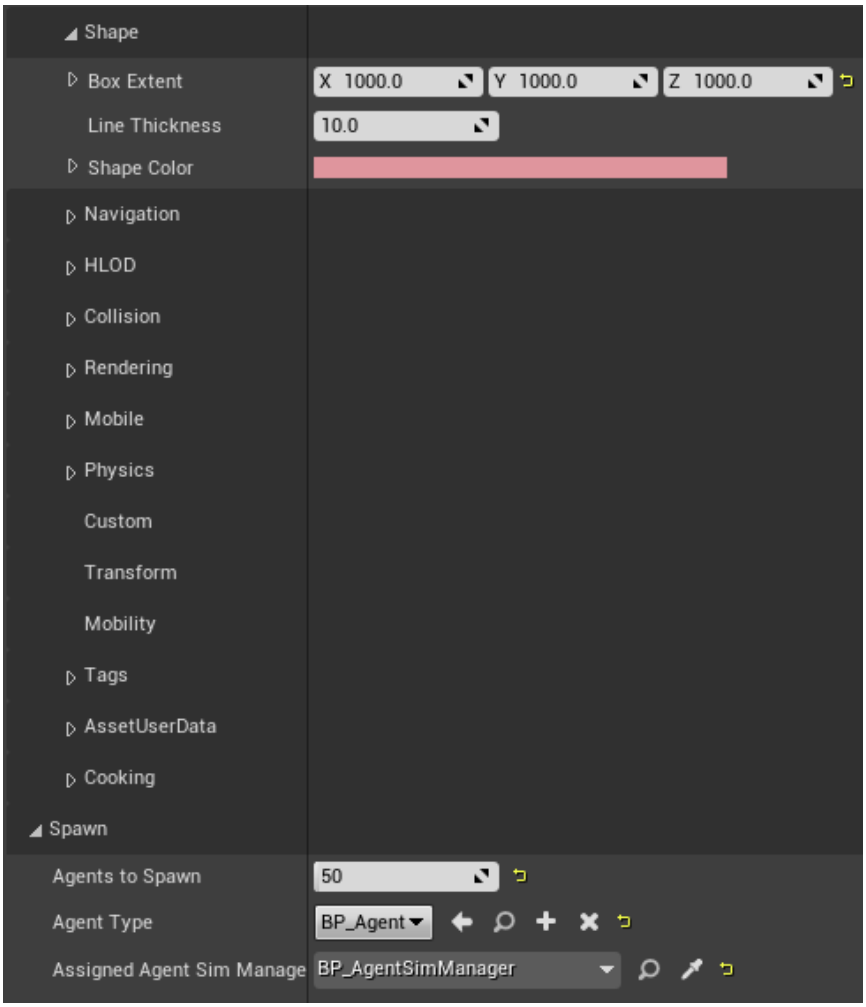


In the world outliner on the right, a folder called “Flock” holds 2 actors which need to be accessed to change the simulation’s different variables.



BP_AgentSpawner

Inside “BP_AgentSpawner”, you can first change the size of the box the Agents will occupy. The box extent vector can be increased and decreased on the XYZ. Below, the number of agents spawned can also be changed. It is currently set to 50 which should work on most computers. The max can be set up to 1600 but this can cause crashes depending on the performance of the computer running the simulation. I suggest slowly increasing in increments of 50 for stability. The agent type and assigned agent sim manager have already been set and shouldn’t be changed as this can cause issues.



For the box collision teleport effect to work with the agents, you will need to remove the collision walls found in the Collision Zones folder within the World Outliner. I suggest just deleting these after using the simulation a few times and then undoing the deletes using ctrl+z.

BP_AgentSimManager

Inside “BP_AgentSimManager”, different flock settings can be changed, altering the way the agents will behave. The movement settings will allow you to change the max and min speeds of the agents; the steering settings will allow the flock’s behaviour weighting to be changed and the avoidance strength; the perception sensor’s FOV can be changed per the behaviour rule; the number of avoidance sensors and the radius can also be altered.

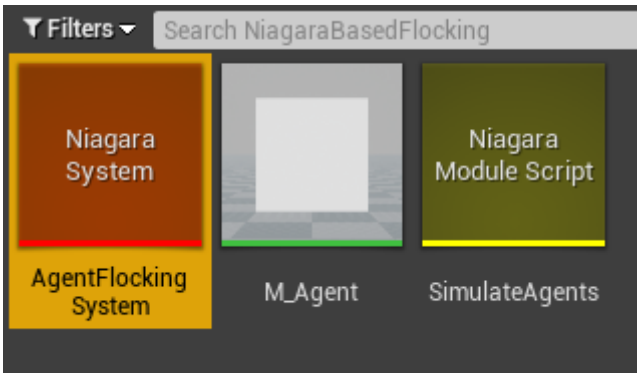
The agents also by default use debug lines to show the collision avoidance system working. This does slow down the performance, so if you want a faster simulation, these lines can be commented out in the Agent class. Comment out lines 288, 289, 293, 338, 339, 344 and 347.

```
284         // Debug lines showing collision checks
285         // Indicates if this hit was a result of blocking collision.
286         if (Hit.bBlockingHit)
287         {
288             DrawDebugLine(GetWorld(), this->GetActorLocation(), this->GetActorLocation() + NewSensorDir * AgentSimManager->GetSensorRadius(), FColor::Red, false, -1.0f, 0, 1.5f);
289             DrawDebugSphere(GetWorld(), Hit.ImpactPoint, 8.0f, 12, FColor::Red, false, -1.0f, 0, 1.0f);
290         }
291         else
292         {
293             DrawDebugLine(GetWorld(), this->GetActorLocation(), this->GetActorLocation() + NewSensorDir * AgentSimManager->GetSensorRadius(), FColor::Green, false, -1.0f, 0, 1.5f);
294         }
```

```
335         // Debug lines visualising the new sensor direction and what would be the impact
336         if (Hit.bBlockingHit)
337         {
338             DrawDebugLine(GetWorld(), this->GetActorLocation(), this->GetActorLocation() + NewSensorDir * AgentSimManager->GetSensorRadius(), FColor::Red, false, -1.0f, 0, 1.5f);
339             DrawDebugSphere(GetWorld(), Hit.ImpactPoint, 8.0f, 12, FColor::Red, false, -1.0f, 0, 1.0f);
340         }
341         // Returning the steering forces and visualising the obstacle avoidance using debug lines
342         else
343         {
344             DrawDebugLine(GetWorld(), this->GetActorLocation(), this->GetActorLocation() + NewSensorDir * AgentSimManager->GetSensorRadius(), FColor::Green, false, -1.0f, 0, 1.5f);
345             SteeringForce = NewSensorDir.GetSafeNormal() - this->AgentVelocity.GetSafeNormal();
346             SteeringForce *= AgentSimManager->GetAvoidanceStrength();
347             DrawDebugDirectionalArrow(GetWorld(), this->GetActorLocation(), this->GetActorLocation() + SteeringForce, 5.0f, FColor::Yellow, false, -1.0f, 0, 3.0f);
348             return SteeringForce;
```

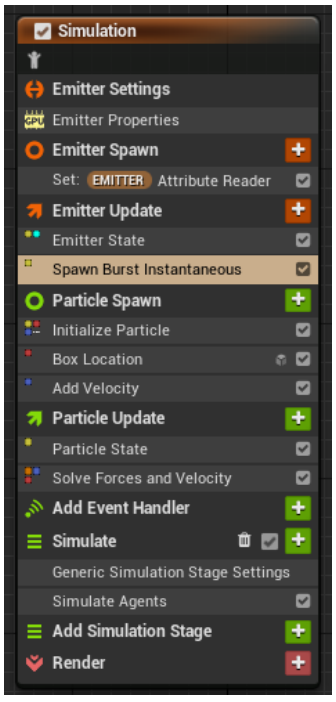
Niagara Flocking

The Niagara simulation should start when entering the map, if not, just press play and it should start working. The way to alter the settings for this simulation is different compared to the class-based sim and instead is found within a Niagara System.



AgentFlockingSystem

Within the “Simulation” emitter, under the emitter update module heading, a module called “Spawn Burst Instantaneous” can be opened, which will allow you to change the spawn count of the agents. By default, this is set to 100, but depending on the computer running the simulation, this can go up to 100,000. Again, I suggest turning this up incrementally so there are no crashes.



The agent's behavioural settings can also be changed within the simulation module. Under the Simulate header, a module called "Simulate Agents" can be opened which will allow you to change the strength and distance of each behavioural rule. The "Space Scale" setting is a vector that will allow you to change the size of the area the agents will occupy, and "Wall Avoid Intensity" will alter the strength the agents will try and avoid that boundary.