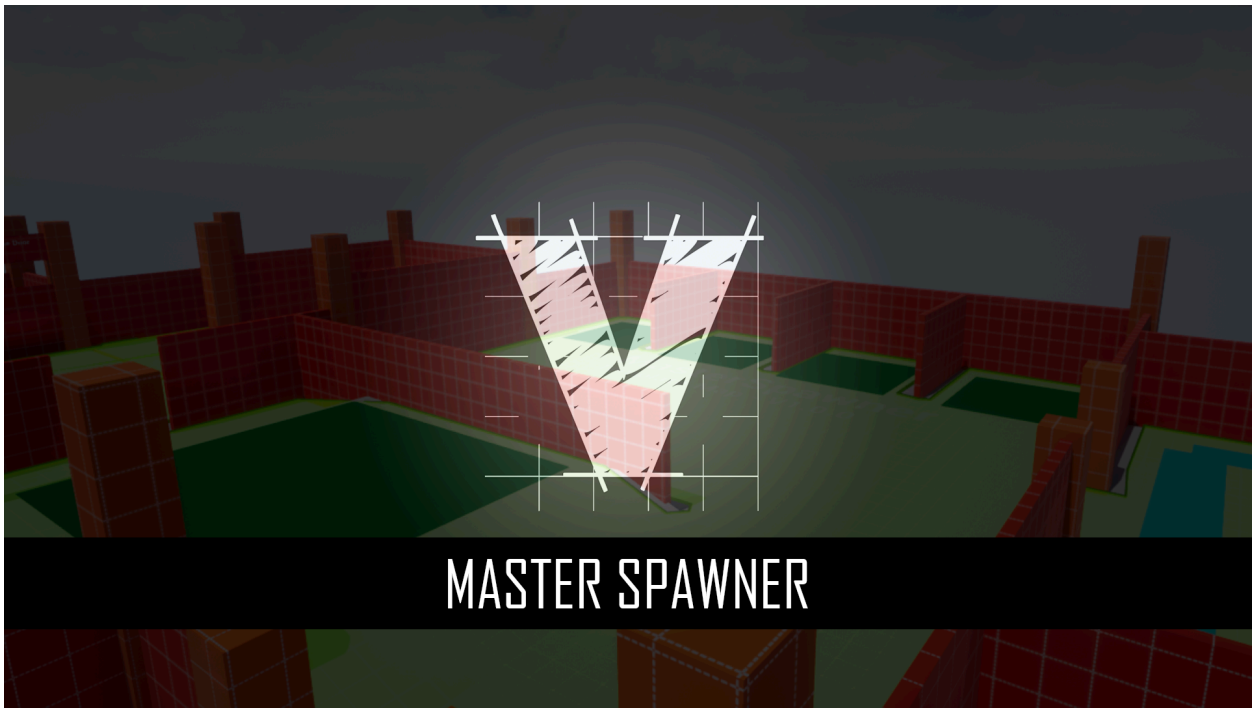
MASTER SPAWNER

Unreal Engine 4 Asset

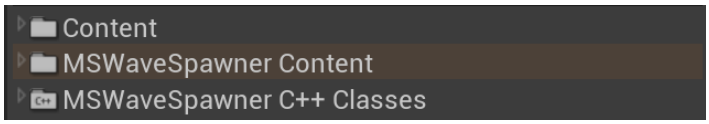
Plugin Setup

MasterSpawner is a C++ based wave spawning system. The main premise of this asset is that it's easy to use and implement. We tried to reduce all the work from the user's side to give a smooth and easy experience. However, this doesn't mean that it's not customizable. For advanced users, all the necessary functions are exposed to a blueprint and can be used in any shape and form.

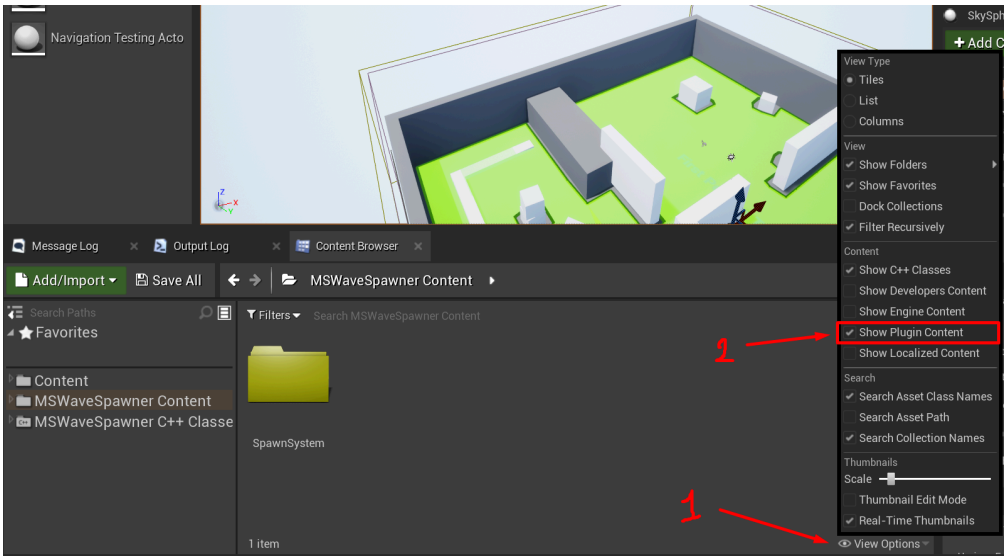
MasterSpawner consists of 4 primary elements: Master spawner, Spawnlocation, Spawn Activator, and the spawn areas.



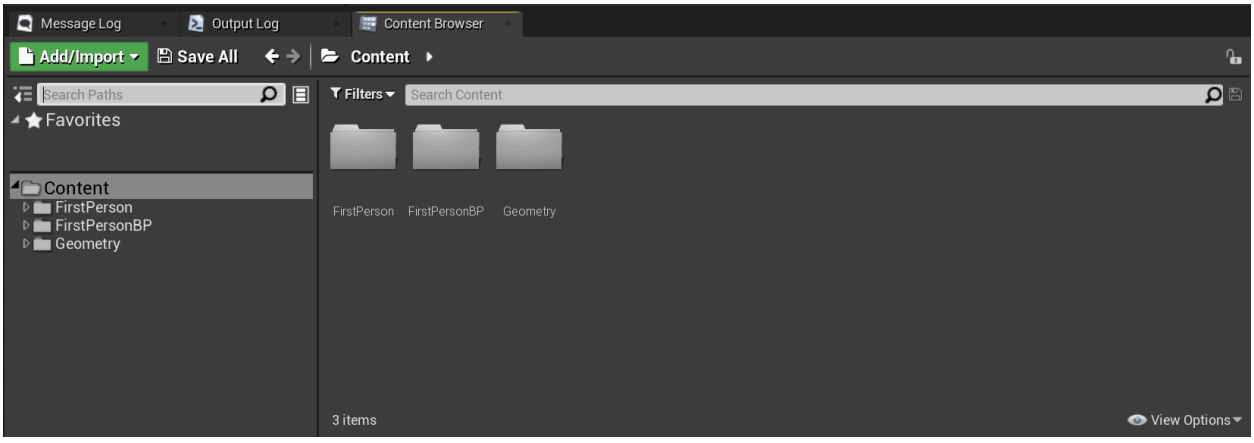
Step 0.1: First checks



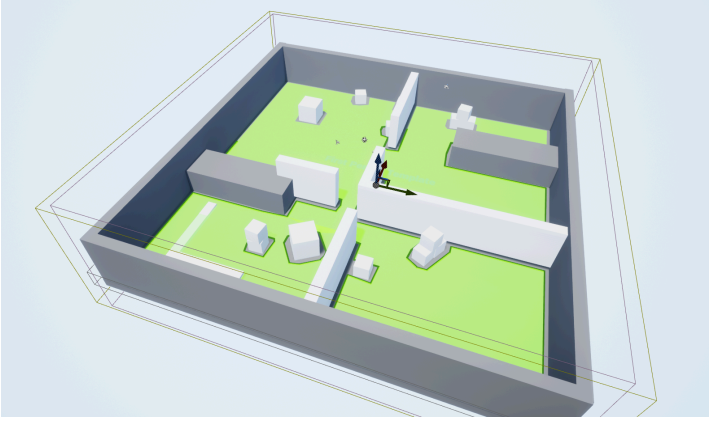
If you can **[MSWaveSpawner Content]** folder in your content browser do the following. If you can see the file, you can move on to the next step. If it didn't work you can try the other option below.



Try this one:



Step 0.2: Nav mesh setup



For the most basic and initial step, you'll need a Nav Mesh Bounds Volume that covers the parts of your map that you want your AI to move and spawn on.

If you don't know what a navmesh is, here is a quick video that explains it: https://www.youtube.com/watch?v=hE7aMBeT53o

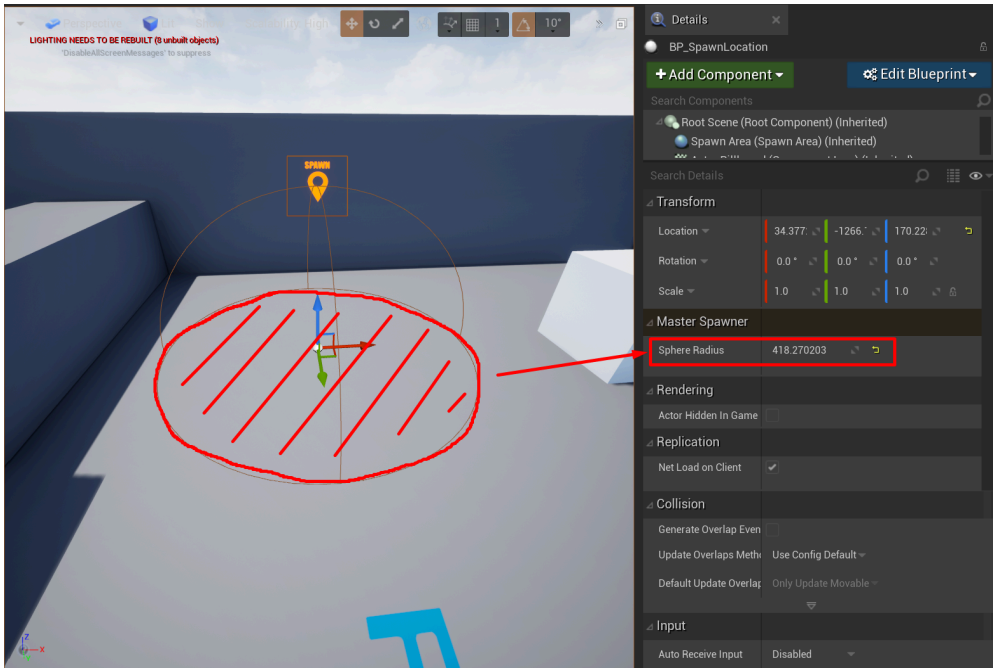
Step 1: Setting up the spawnLocations

You can find the **BP_SpawnLocation** under MSWaveSpawner Content > SpawnSystem > Blueprints

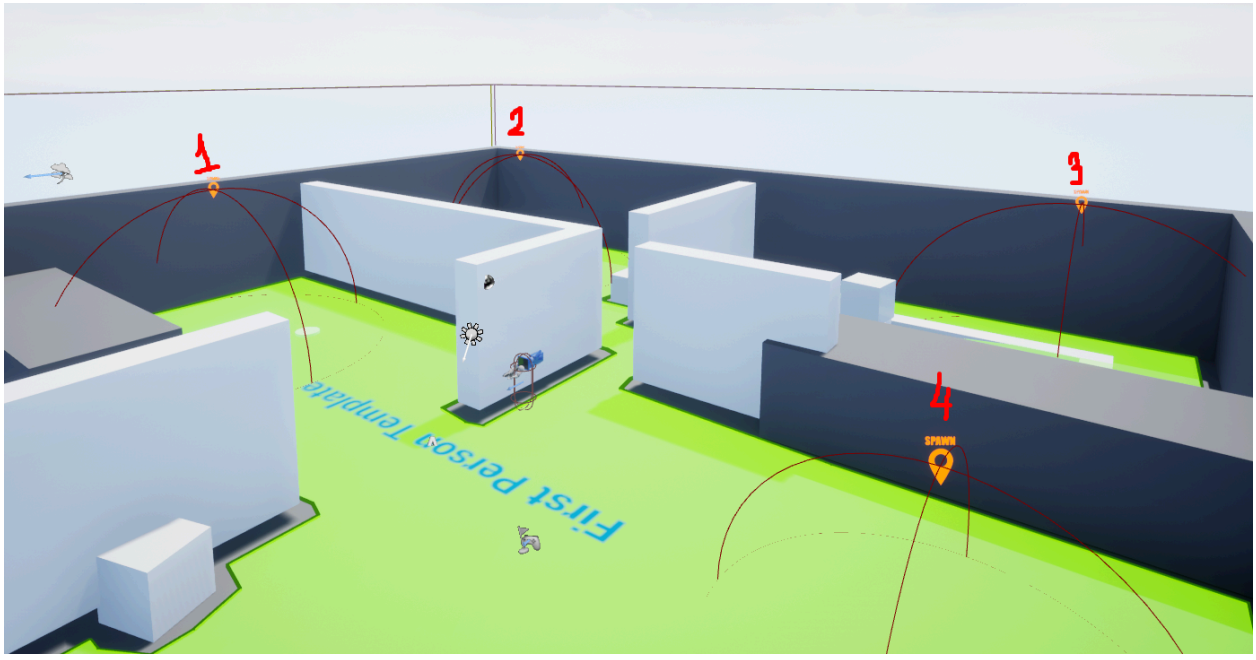
BP_SpawnLocation's are the components that our plugin will be using to get a random location to spawn an AI from. But just having spawnLocations is now enough. You'll also need to have spawnAreas which I'll explain soon.

There is a little warning here! BP_SpawnLocation is not a single location, the name of the BP might give you that idea but it's not. What it does is that limits the area for the plugin to select a random location from. So if the plugin wants to get a random location from this BP_SpawnLocation, that location will be within the bounds of the sphere as you can see down below.

You can change the size of the sphere with the variable called "Sphere Radius" in the details panel. Now go ahead and place as many BP_SpawnLocations as you want on your level.



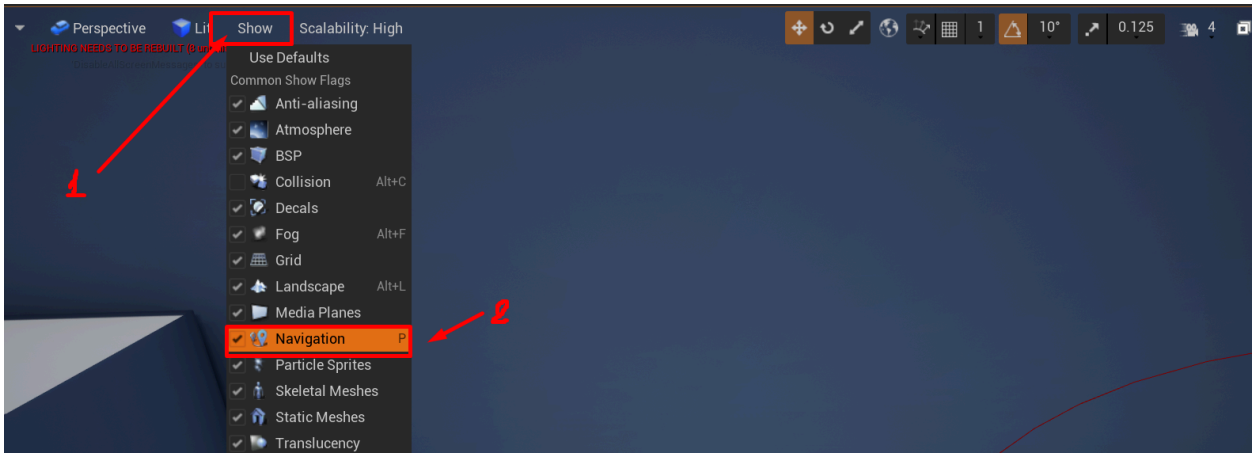
I've placed four BP_SpawnLocations on my level:



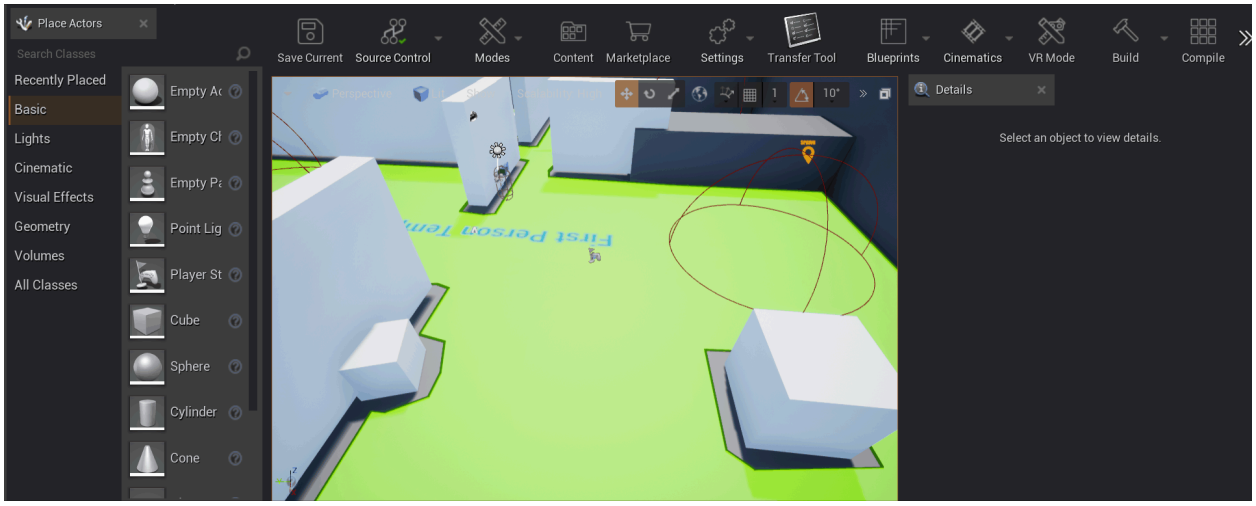
Now that you placed all of your BP_SpawnLocations we can move on to the next step. Which is defining the safe spawn areas. Without the safe spawn areas, the plugin will not work. The reason why we have safe spawn areas is

that BP_SpawnLocations are not that precise, they are just huge spheres. However, we want to spawn our AI's in specific locations, not everywhere.

IF YOU CAN'T SEE ANY GREEN AREAS: Press "P" on your keyboard or do the following:



To set up the safe spawn areas you can do the following:

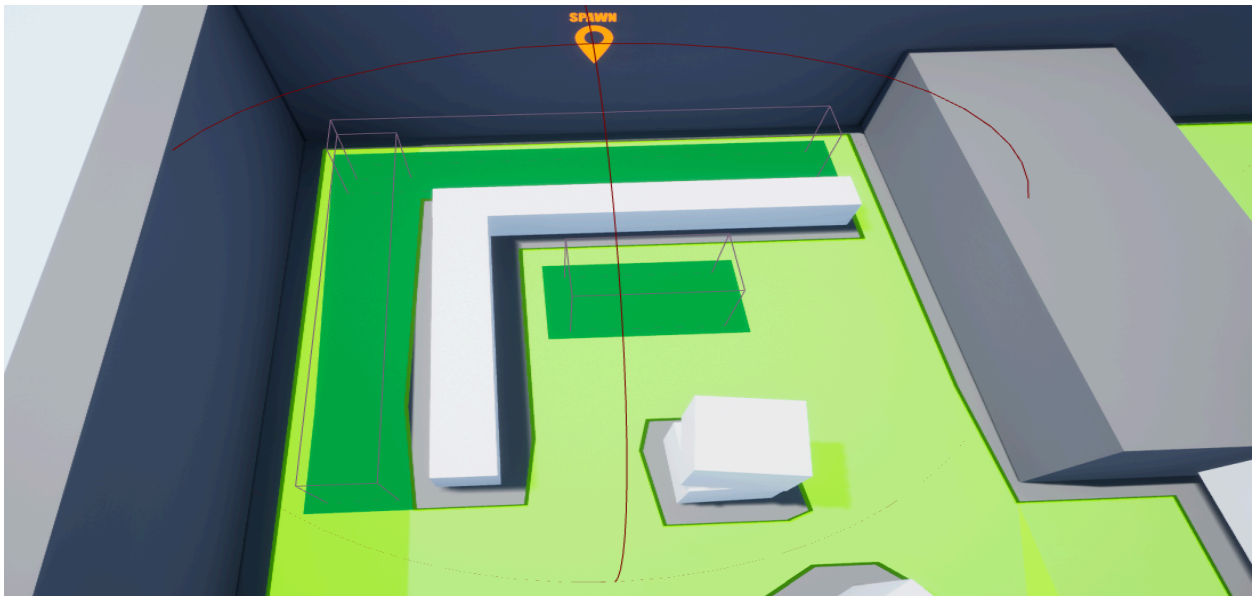


[CAUTION I'M NOT SELECTING ALL OF THOSE ELEMENTS! THERE IS A VISUAL BUG WITH MY ENGINE]

Put these dark green areas on the location that you want your AI's to spawn at. Don't forget that they'll have to be within the bounds of a BP_SpawnLocation to work.

Example Spawn Location: If this location is selected, the random location will be selected from one of the dark green areas that

are inside of the BP_SpawnLocation sphere.



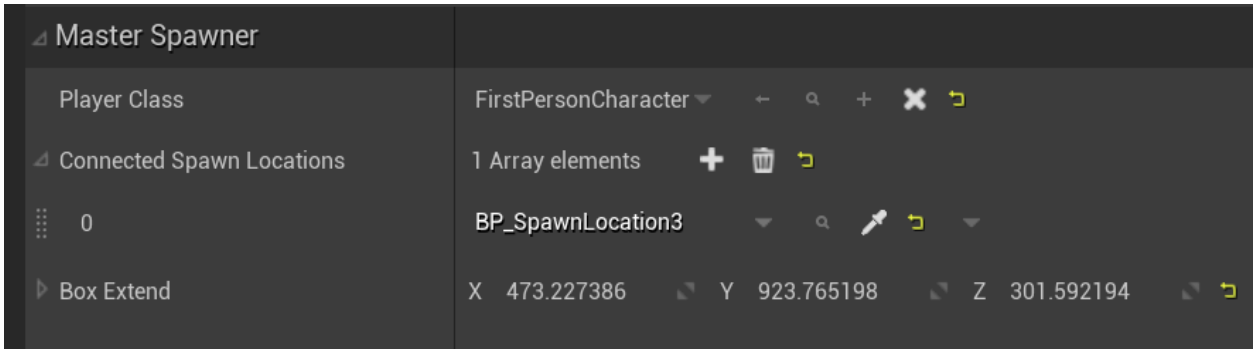
Before moving onto the next step do this to your whole level!

Step 2: Setting up the spawnActivators

Now that we have our spawnLocations, the next step is to decide when they should be used. If you don't want to use all of the spawnLocations at the same time and all the time, this is the step for you. For this we will be using the BP_SpawnActivator under MSWaveSpawner Content > SpawnSystem > Blueprints

BP_SpawnActivator is basically a trigger. When the player enters the activator the corresponding BP_SpawnLocation's will be activated and from then on, the AI's will be spawning from the spawn locations.

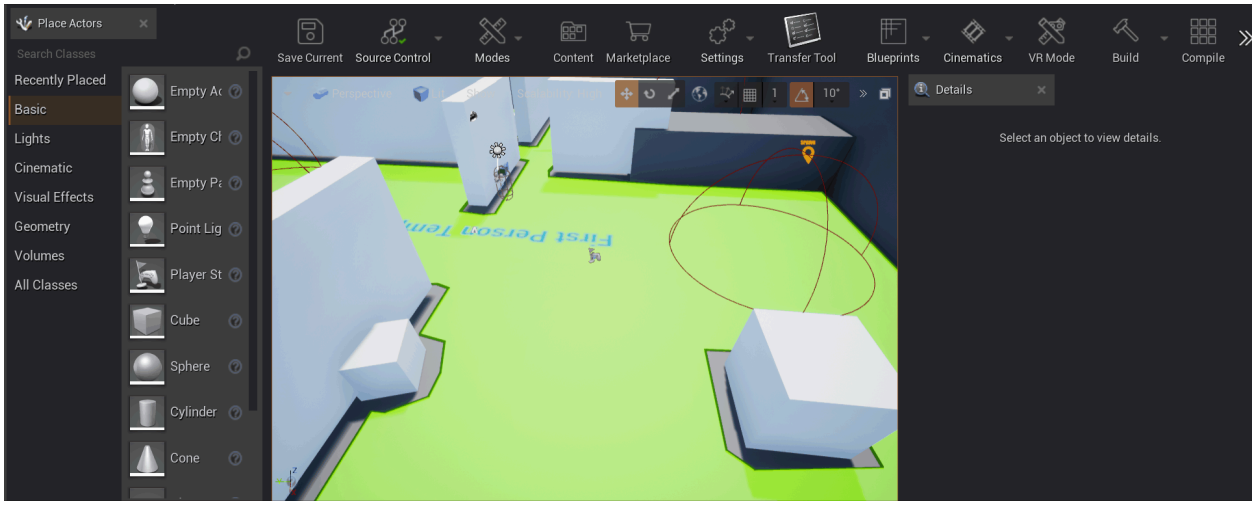
You have to do 3 things to make it work.



Player Class: The class to track for overlap. Set this to your player character.

Connected Spawn Locations: Spawn locations to activate after overlapping with this activator.

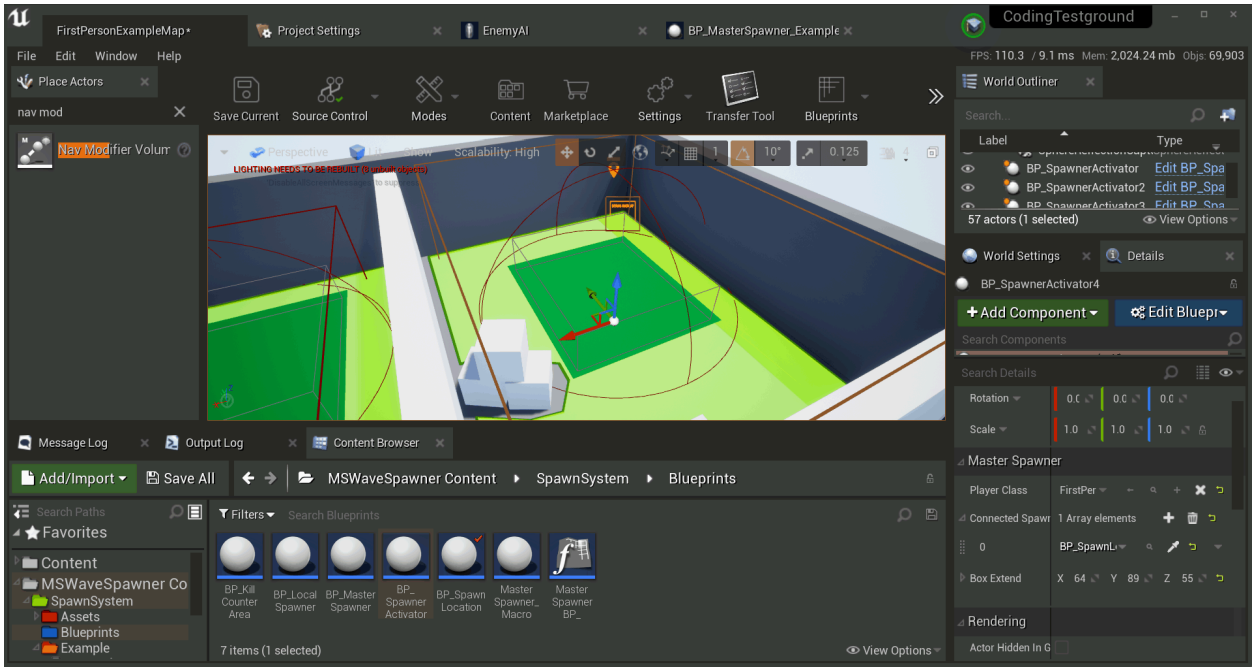
Box Extend: Size of the activator.



Step 3: Creating the TraceChannelToIgnore

The MasterSpawner has a couple of safety checks to spawn the AI's at the correct locations (To avoid spawning within the walls, or within each other. And that is the Trace check. Whenever the plugin wants to spawn something it places a sphere at that position (the size can be changed). If the sphere detects an object that blocks the trace channel that we are going to create, it'll look for another location instead of spawning at that location. Because that means we are either trying to spawn our AI inside of an unwanted object or we are trying to spawn it close to an unwanted object.

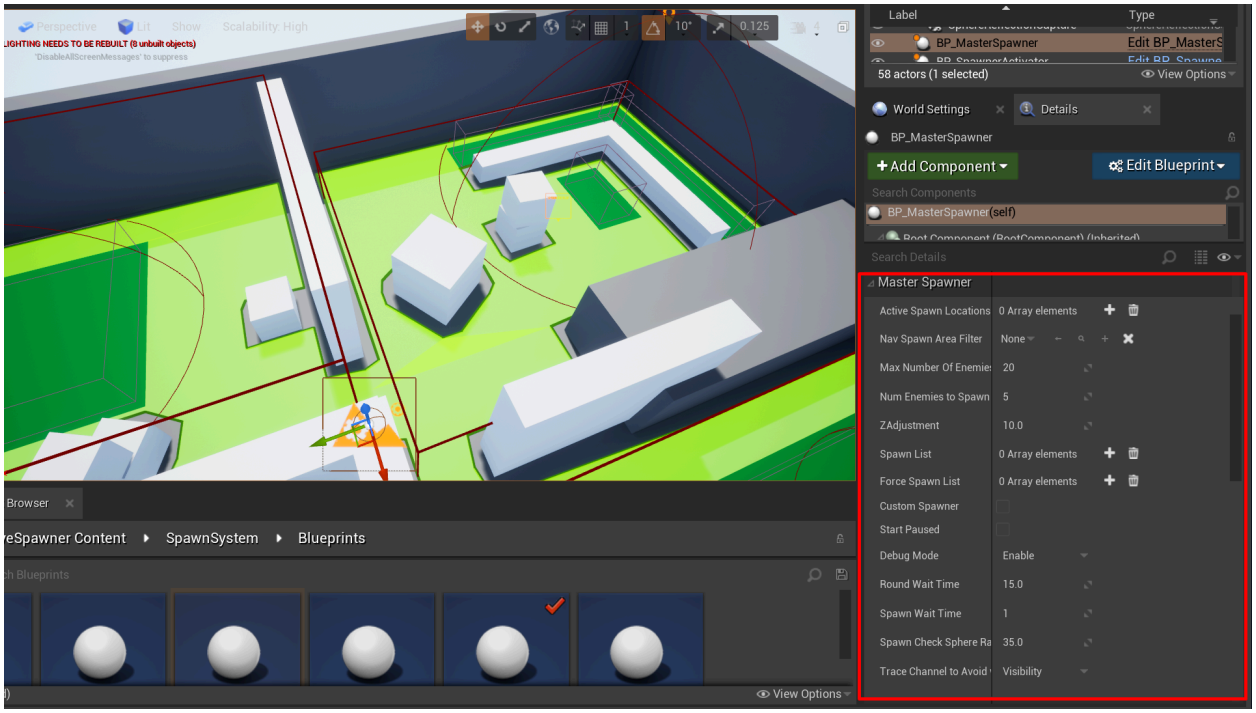
To create the trace channel do the following.



Step 4: Master Spawner

BP_MasterSpawner is the core of our plugin. It handles almost everything. So it's crucial to know how to set it up properly. Let's grab the blueprint from the asset folder and place it on the scene. (You must have exactly one BP_MasterSpawner on your level).

So let's go through the properties of the BP_MasterSpawner:



Active Spawn Locations:

Add the BP_SpawnLocation's that you want to be activated at the beginning of the game to here. Don't leave it empty. But what is it? As you can guess from the previous sections, this is the part where we store the activeSpawnLocations (The spawner will choose a random BP_SpawnLocation from this list whenever it wants to spawn an AI). Going into BP_SpawnActivators will change this list with the ones in the BP_SpawnActivator.

Nav Spawn Area Filter:

This is the filter that says "Yeah we want to use dark green areas to spawn our AI's". For this, you'll want to choose the [Spawn_Area_Filter_Default].

Max Number Of Enemies:

Max number of enemies to have on-screen. The bigger the number, the more enemies will be alive at the same time. Reducing this number might improve performance.

Num Enemies To Spawn:

Number of enemies to spawn on around. Increase this to spawn more enemies.

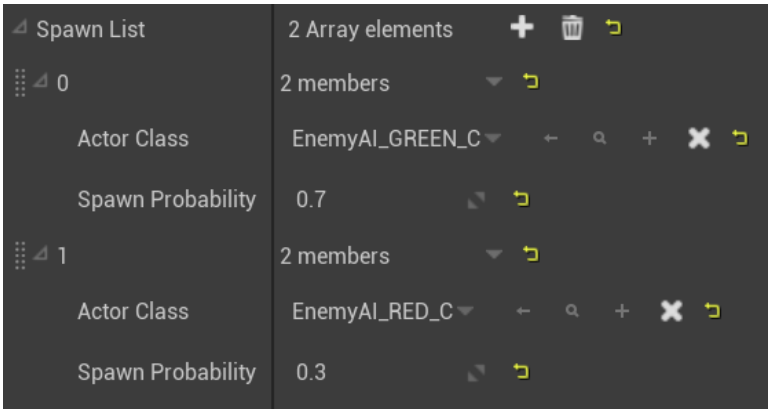
Z Adjustment:

If any of the AI's are spawning inside of the ground increase this. If not, leave it at 10.

Spawn List:

This is the list of the AI's that will be spawned in every round (based on their probabilities). You can always add or remove items from this list programmatically. Use this list for your constantly spawned AI's.

So here is an example:



Actor Class: is the AI to be spawned

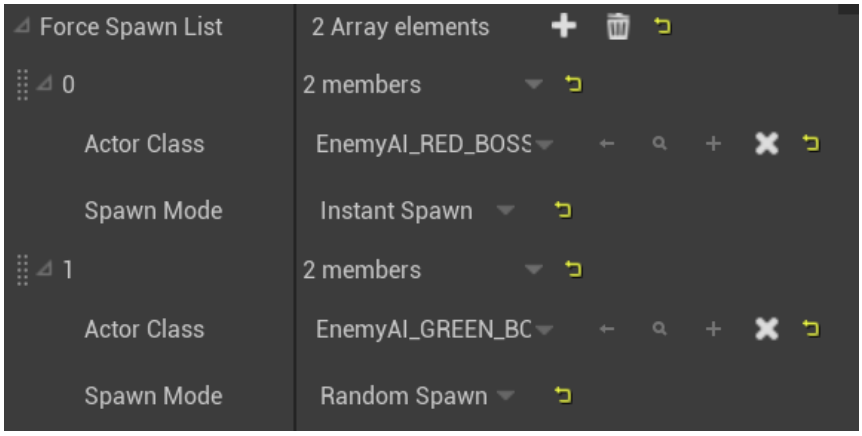
Spawn Probability: is the probability for this actor to be spawned (0.0 means 0%, 1.0 means 100%)

The Sum of all probabilities doesn't need to be 1.0. While spawning they will be scaled.

Force Spawn List:

This is the list of the AI's that will be spawned at the specific round. Use this list for special spawns such as bosses or unique enemies. If an AI gets spawned from this list the corresponding element will be removed from this list. So this list will be empty at the end of every round.

And here is an example:



Actor Class: is the AI to be spawned

Spawn Mode: This defines when to spawn this actor

Instant spawn means they will be spawned at the beginning of the round.

Random spawn means they will be spawned randomly during the round.

Custom Spawner:

Turn this on if you want to override the spawner logic and make your own.

Debug Mode:

For default keep this off. Turn this on to see debug messages from the plugin. It will give you more information about your problems if you encounter any in the Output Log tab.

Round Wait Time:

How long to wait before starting a new round. (Round ends after all of the AI's are killed)

Spawn Wait Time:

The number of seconds to wait before spawnmöing an AI. Default is 1

Spawn Check Radius:

This is the sphere that I mentioned in step 3.

> The MasterSpawner has a couple of safety checks to spawn the AI's at the correct locations (To avoid spawning within the walls, or within each other. And that is the Trace check. Whenever the plugin wants to spawn something it places a sphere at that position (the size can be changed). If the sphere detects an object that blocks the trace channel that we are going to create, it'll look for another location instead of spawning at that location. Because that means we are either trying to spawn our AI inside of an unwanted object or we are trying to spawn it close to an unwanted object.

Trace Channel to Avoid When Spawning:

Set this to the traceChannel that you've created at step 3. This basically tells the Spawner to avoid objects that block this collision. If you leave this at visibility there won't be any AI's being spawned.

Player Blueprint: Variables, Functions, Explanations

(G) Get : You can read the value
(S) Set: You can set the value

MasterSpawner is a C++ based wave spawning system. The main premise of this asset is that it's easy to use and implement. We tried to reduce all the work from the user's side to give a smooth and easy experience. However, this doesn't mean that it's not customizable. For advanced users, all the necessary functions are exposed to a blueprint and can be used in any shape and form.

BP_MasterSpawner: This is the main blueprint that will be handling the spawning and control of everything. Some key components of the MasterSpawner.

Variables

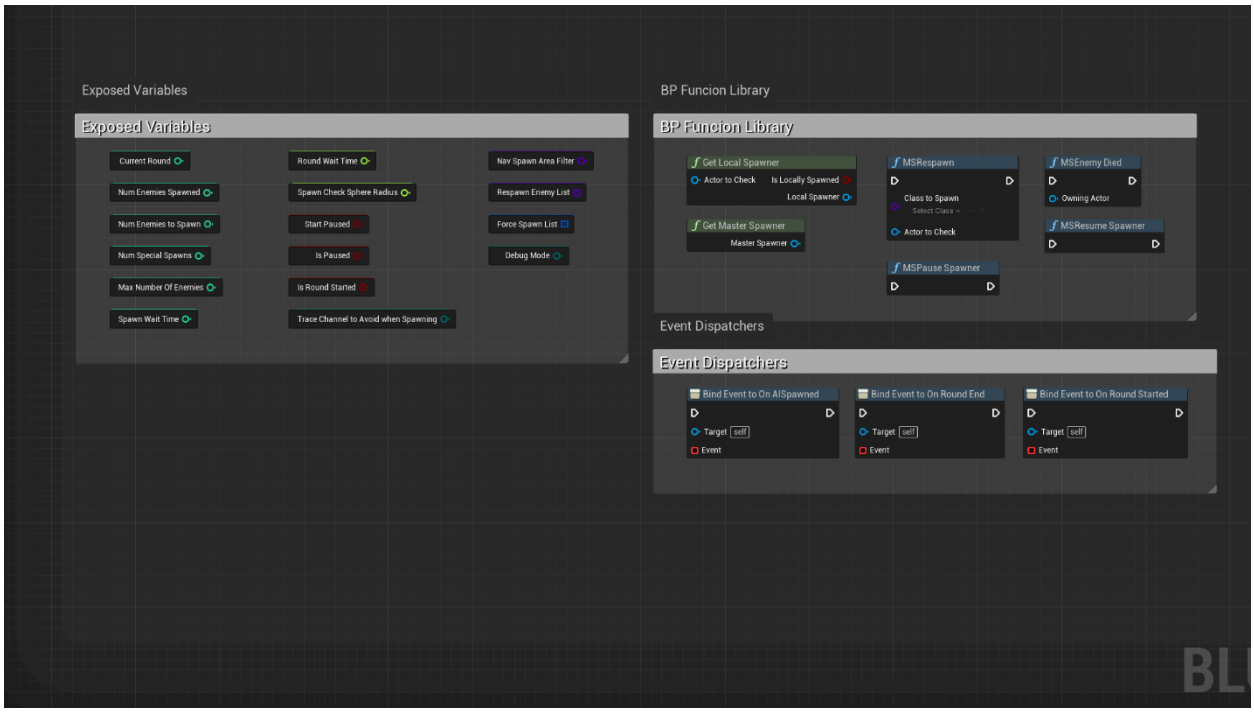
1. Current Round (G): Returns the current round.
2. NumEnemiesToSpawn (G)(S): Number of enemies to spawn on that round. Increase this to spawn more enemies.
3. NumEnemiesSpawned (G): Number of enemies that are already spawned.
4. NumSpecialSpawns (G): Number of *forceSpawned* enemies.
5. MaxNumberOfEnemies (G)(S): Max number of enemies to have on-screen. The bigger the number, the more enemies will be alive at the same time. Reducing this number might improve performance.
6. SpawnWaitTime (G)(S): How long to wait before spawning an AI.
7. RoundWaitTime (G)(S): How long to wait before starting a round. You can use this to give some more time to the player to prepare.
8. SpawnCheckSphereRadius (G)(S): Size of the sphere that will be used to check the location for overlaps. If the AI's spawn inside of each other or objects. Increase this.
9. StartPaused (G)(S): Start the spawner paused.
10. IsPaused (G): Returns if the spawner is paused or not.
11. IsRoundStarted (G): Returns false if the round is finished and waiting for the next one. True if the round is still ongoing.

12. TraceChannelToAvoidWhenSpawning (G)(S): This is the trace channel that you want to use to avoid obstacles in the spawn areas such as walls.
13. NavSpawnAreaFilter (G)(S): *NavAreaFilter* to spawn the AI's in.
14. RespawnEnemyList (G): AI's that will be respawned.
15. DebugMode (G)(S): Returns true if the debug mode is active.
16. ForceSpawnList (G)(S): This is a struct array of AI classes and their spawn methods. After getting spawned, the spawned AI class gets removed from the list.

    For the spawn methods:

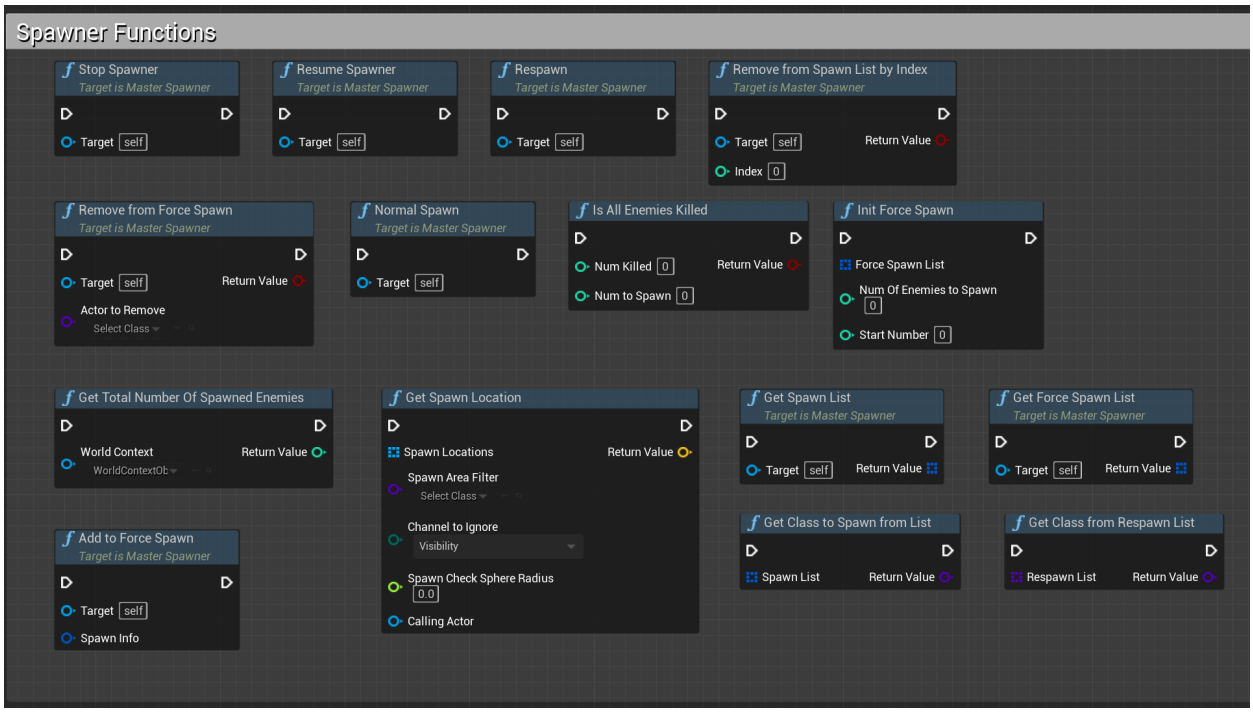
    Instant spawn means they will be spawned at the beginning of the round.

    Random spawn means they will be spawned randomly during the round.
17. ActiveSpawnLocations (G)(S): These are the spawn locations that the spawner will be choosing from while spawning.
18. SpawnList (G)(S): This is a struct array of AI classes and their probability to spawn. The classes in this array will be spawned in every round based on their probabilities.
19. NumKilled (G): Number of enemies that died



Functions

1. StopSpawner: Pauses the spawner
2. ResumeSpawner: Resumes the spawner
3. RemoveFromSpawnListByIndex: Removes an item from the *spawnList* by index.
4. RemoveFromForceSpawn: Removes the element with the given class from the *forceSpawnList*
5. IsAllEnemiesKilled: Returns true if all the enemies are dead
6. InitForceSpawn: Call this to initialize the *forceSpawnList*. Assigns the turn order for the random spawns.
7. GetTotalNumberOfSpawnedEnemies: Returns the number of AI's that are spawned and alive.
8. GetSpawnLocation: Returns a suitable location to spawn the AI from
9. GetSpawnList: Returns the spawnList.
10. AddToForceSpawn: Add the given struct to the *forceSpawnList*.
11. GetClassToSpawnFromList: Randomly returns a class from the *spawnFromList* based on their probabilities.
12. GetClassFromRespawnList: Gets an AI from the respawn list.



Event Dispatchers

1. Event Enemy Spawned: Gets called when an AI gets spawned. Use this to set values for the spawned actors.
2. Event Round Started: Gets called when the round starts.
3. Event Round Ended: Gets called when the round ends.

BP_SpawnLocation: If this spawn location is selected the AI will be spawned in a random suitable location (on a spawnNavAreaFilter) within the given radius of the SpawnLocation.

BP_SpawnActivator: This includes an array of spawn locations. Overlapping with the activator will replace the ActiveSpawnLocations at the MasterSpawner with the ones that the SpawnActivator has. With this, you can change the spawnLocations based on the player's location.

1. ConnectedSpawnLocations (G)(S): *SpawnLocations* to switch to.
2. PlayerClass (G)(S): *PlayerClass* to check for overlaps.

SpawnNavAreaFilter: This nav filter will be used to determine the spawn-able areas.

BP_LocalSpawner: These are like MasterSpawners but their purpose is to host small challenges such as kill and survive challenges. Includes similar variables and components as the master spawner.

BP_KillCountArea: If an AI dies within the bounds of this area the kill count will increase, and a function will be notified. You can use the kill count to create custom challenges.

Variables

1. IsActive (G)(S): Whether the *killCounter* should count or not.
2. KillTrackList (G)(S): An array of classes to track. Classes in this array will be counted if they die within the bounds of the *killCountArea*. They will also be individually counted.
3. TotalKillCount (G): Total number of AI's that died within the bounds of the *killCountArea*.

Functions

1. GetKillCountForTrackedClass: Returns the specific kill count for a class. Returns -1 if the class doesn't exist.
2. ResetKillCount: Resets the total and the specific kill counts.
3. ResetSpecificKillCount: Resets the kill count for a specific class
4. AddKillToSpecificKillCount: Adds kill to a specific class kill.
5. IsInKillCountList: Returns true if the given class is in the *killTrackList*.
6. AddKill: If the AI class is in the *killTrackList* array it will increment the *totalKillCount* and the specific *killCount* for that class.

Event Dispatchers

1. Event Kill Added: Gets called when a registered AI dies within the bounds of the *killCountArea*.
2. Event Counter Reset: Gets called when the counter gets reset.