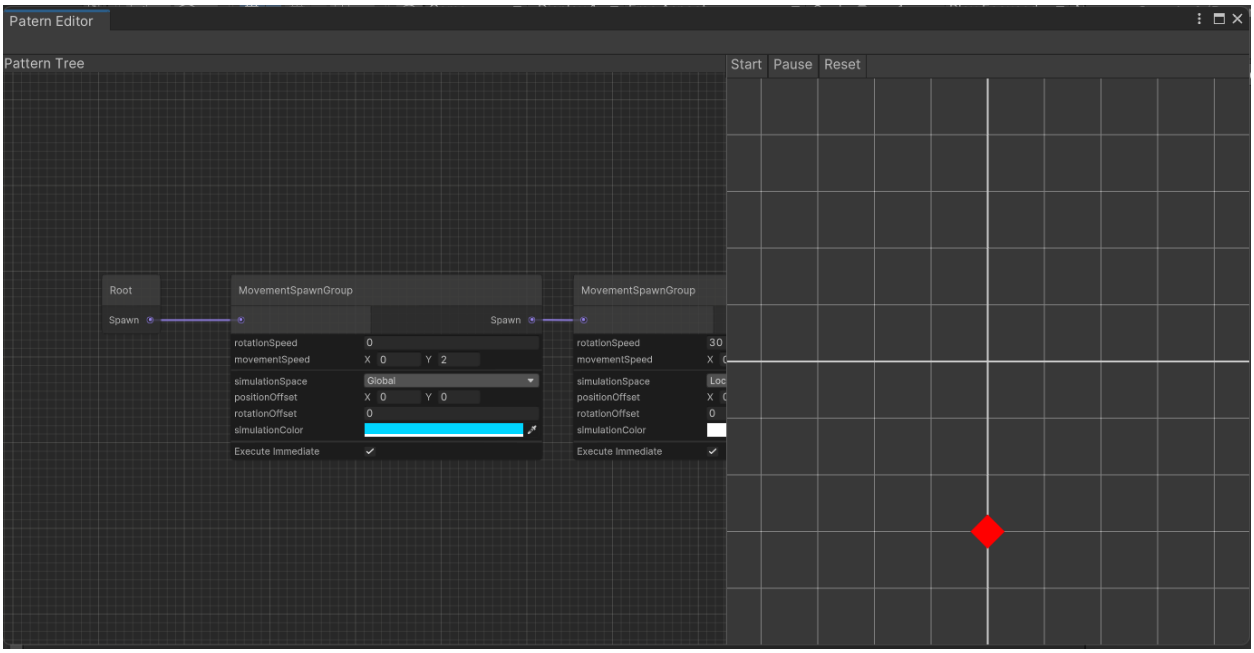
Bullet Pattern Editor Documentation.

Getting Started:

To create your first bullet pattern, RightClick => BB => Pattern.

Then click “Open Editor Window” in the inspector. (This is the only way to edit the pattern)

The Editor:



The Left is the Pattern Tree and right is the Simulation View.

Pattern Tree:

I need to clarify what a “SpawnGroup” is because I am going to use that word a lot.

A SpawnGroup is a object that spawns more SpawnGroup. How and what the SpawnGroup creates is determined by the type of SpawnGroup it is. Each SpawnGroup have it’s own position and rotation.

The tree starts out with only a “Root” node. This is where the pattern starts.

Rightclick and choose which type of SpawnGroup to create.

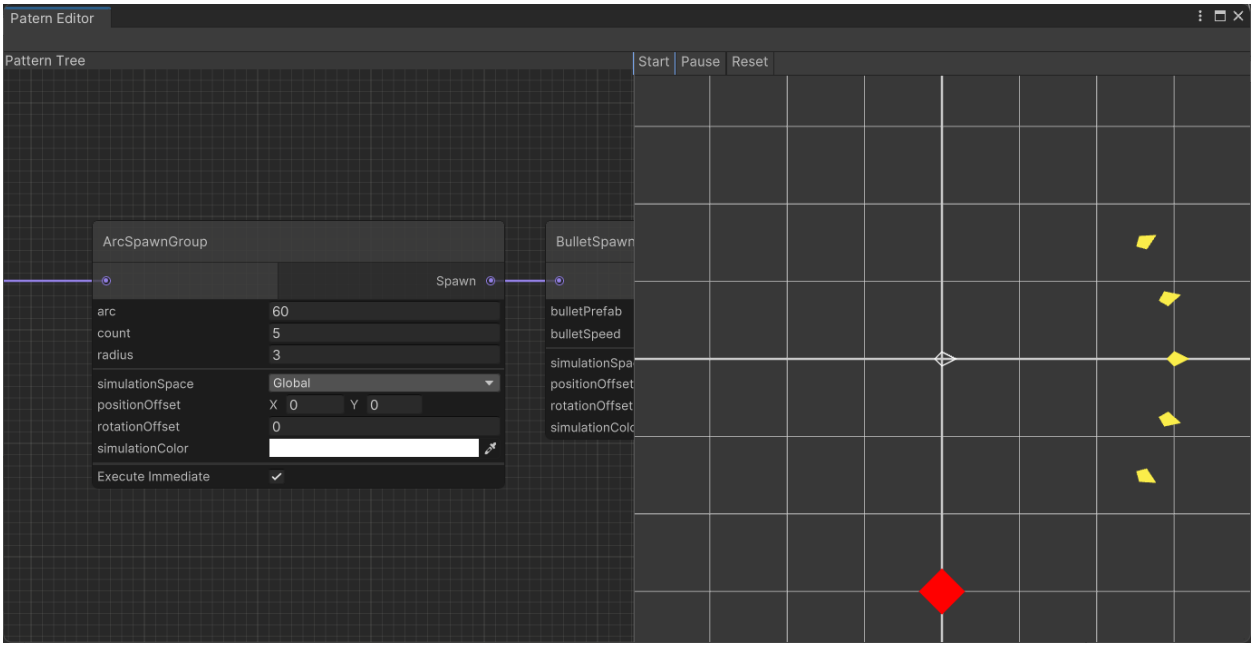
You build a tree of SpawnsGroups from left to right, each connection indicates that the left one spawns the right one.

Every SpawnGroup have a few variables:

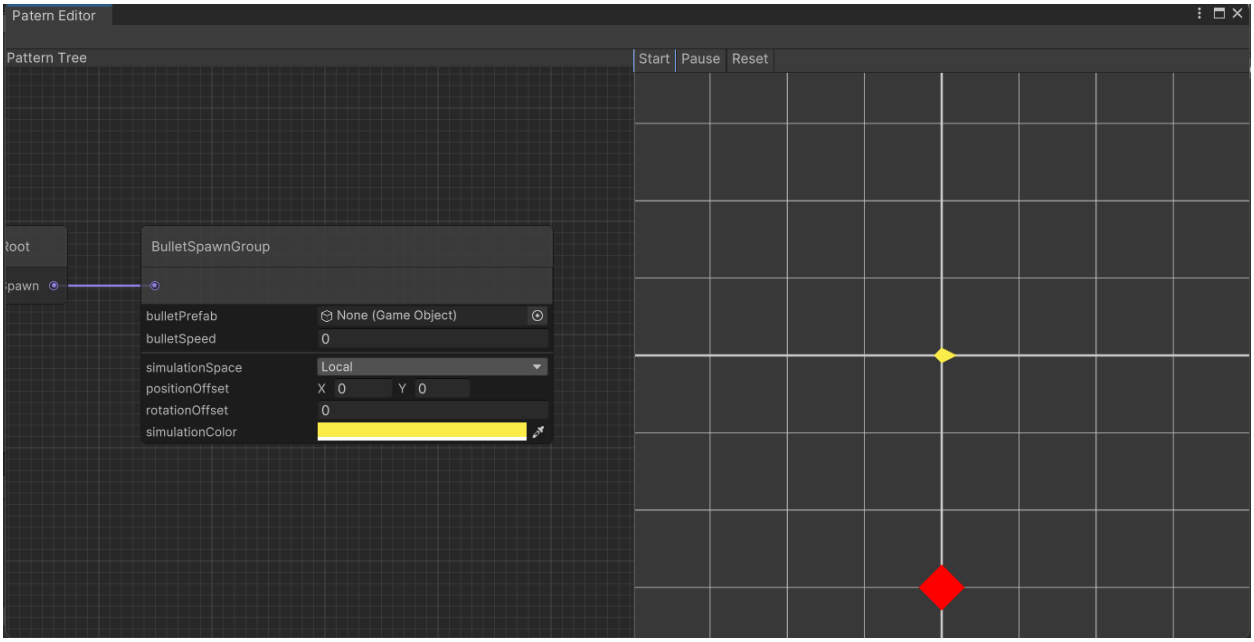
- SimulationSpace: if this is Local, the SpawnGroup will follow and inherit the position and rotation of it’s parent even after spawning.
- PositionOffset: The position when this group is spawned will be offset from it’s parent.
- RotationOffset: The same thing but with rotation.
- SimulationColor: The color that will be shown inside the simulation view. Have no effect when used in the game scene.
- ExecuteImmediate: If toggle, the SpawnGroup will spawns the next spawn group once immediately upon creation. This is functionally similar to setting delay to 0 and repetition to 1.
- Delay: The amount of time to wait after creation before spawning the next SpawnGroup.
- Repetition: The amount of time to spawn the next SpawnGroup.
- Interval: The amount of time to wait between repetitions.

Currently, there are 7 different types of prebuild SpawnGroups:

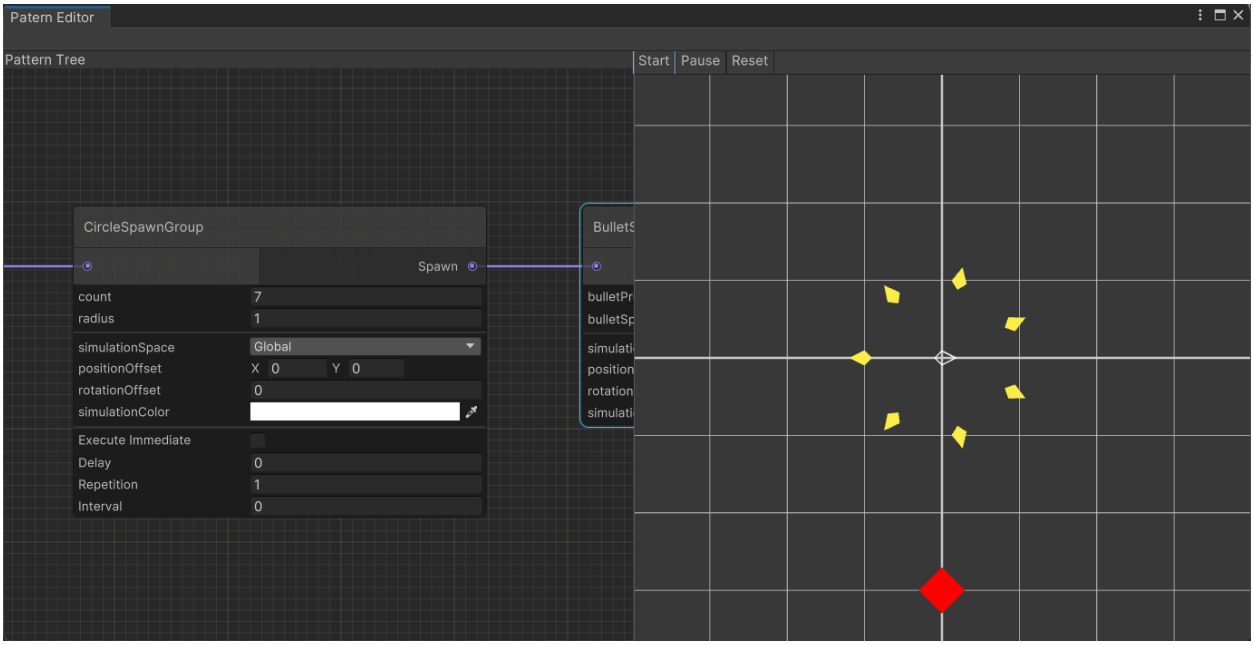
1. ArcSpawnGroup: Spawns SpawnGroups in an arc



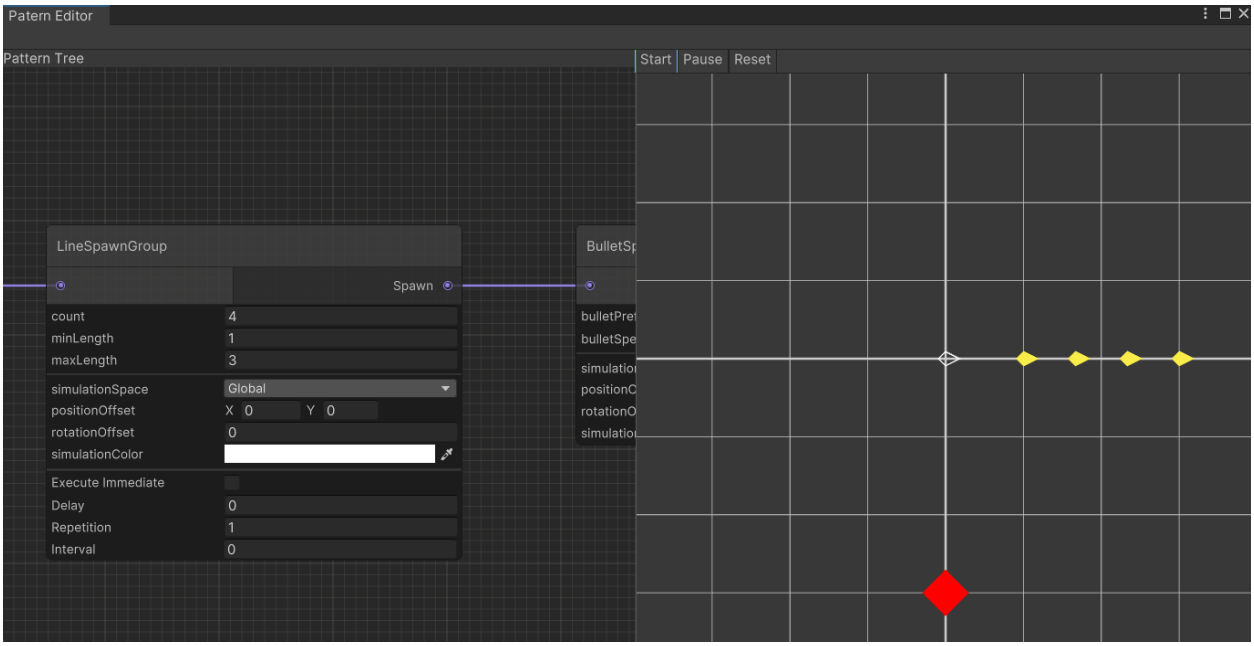
2. BulletSpawnGroup: This is the only one that doesn't spawn new spawngroup but spawns a bullet object. This is always immediate, local and can only spawns 1 bullet.



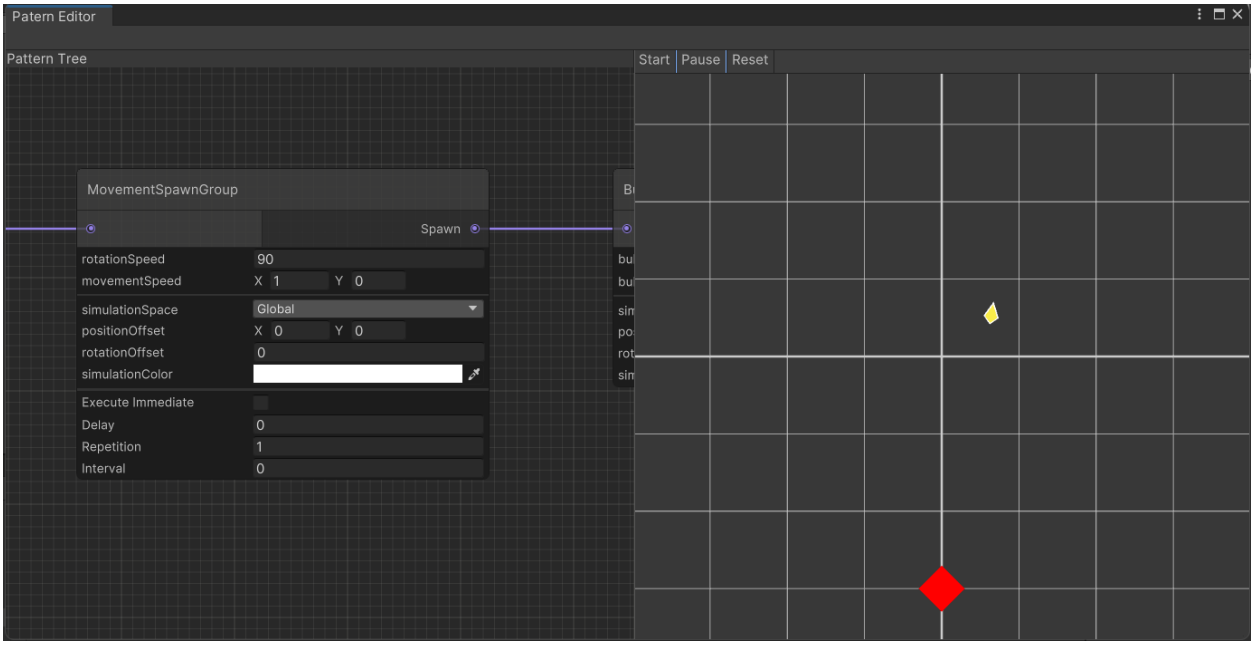
3. CircleSpawnGroup: Spawns SpawnGroups in a 360 degree with equal spacing.



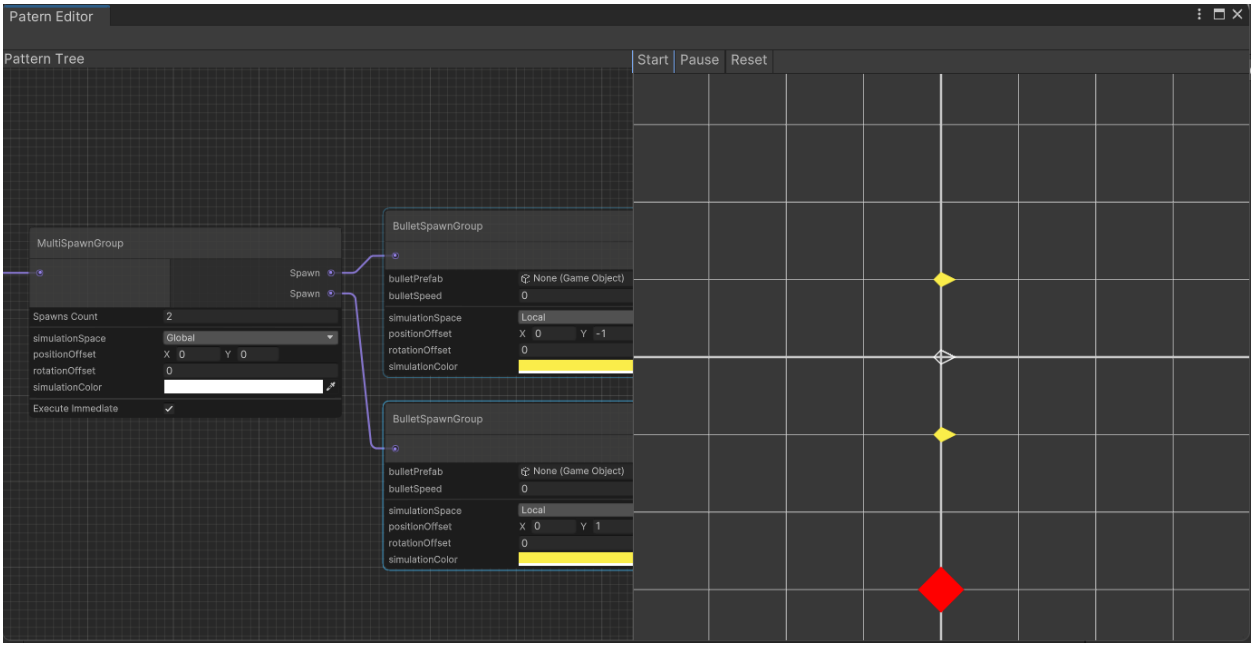
4. LineSpawnGroup: Spawns SpawnGroups in a line



5. MovementSpawnGroup: This is one of the few SpawnGroup that can move on it’s own. With it’s own rotation speed and move speed.

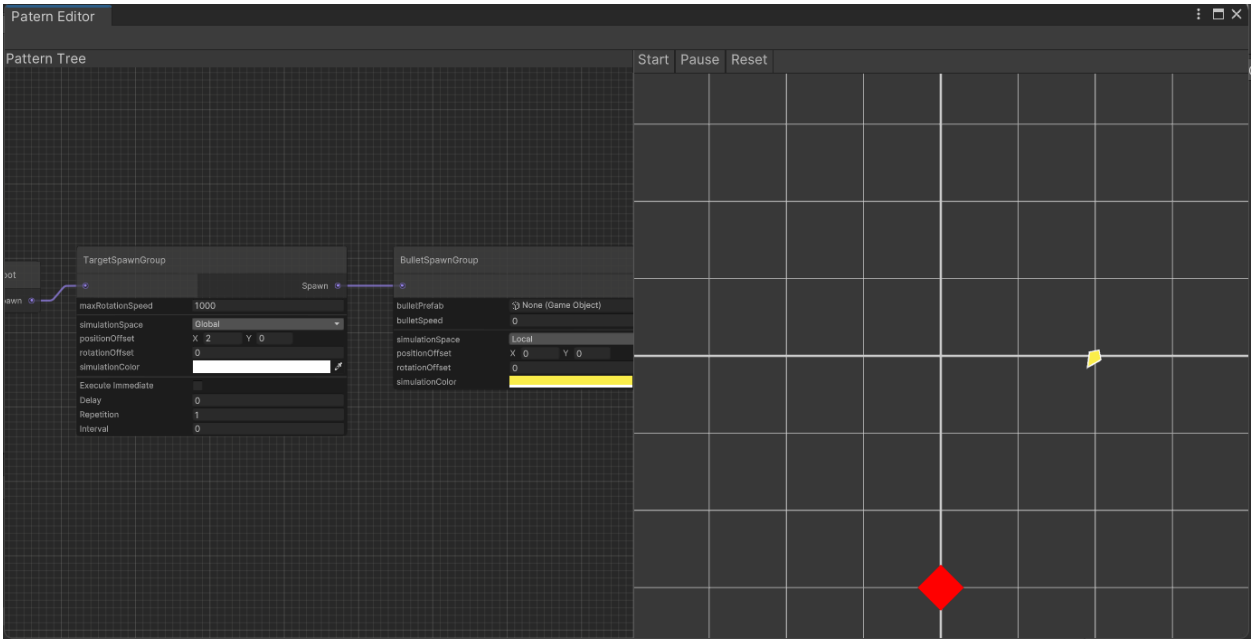


6. MultiSpawnGroup: Spawns multiple types of SpawnGroups at the same time.



7. TargetSpawnGroup: Rotates toward the player. If maxRotationSpeed is over 1000, the rotation is instant. The player is the first object with the “Player” tag. If you want other

player, override this SpawnGroup with your own class.



Simulation View:

The right part of the editor is the Simulation View. This is used to view the current bullet pattern without having to run the game itself. Allowing fast prototyping and debugging.

The grid square is 1 unit.

Each SpawnGroup in the Tree will appear in the simulation with the color set in the simulationColor. To make it disappear, set the color's alpha to 0. The shape is a hollow arrow, while bullets are solid color.

SpawnGroups disappear when they have spawned all their next SpawnGroups and have no child with their simulation set to local.

BulletSpawnGroup specifically will have it's bullet move in a strait line for 5 second. If the bullet in game have more functionality, it will not be reflected inside the simulation.

The red diamond represent the player at the position (0, -3). So TargetSpawnGroup will try to rotate towards that.

How to use the Pattern in game:

Examine the BB.PatternRunner class.

First you Instantiate the Pattern (If you don't do this, you might run into trouble using multiple copies of the same pattern).

Call Pattern.Reset() if you want. This resets the state of the pattern. It does not clear the bullets generated by the patterns.

Then call Update(float deltaTime) from Update with Time.deltaTime or FixedUpdate with Time.fixedDeltaTime.

Seting position and rotation of the Pattern will affect every Bullet that was generated by the pattern. If you want more complex behavior without complex bullets, This is how you can manipulate entire groups of bullet at once.

Note: If the pattern gets destroyed, paused or reset, the bullet that was generated by the pattern will not disappear. Instead, the bullets will continue to act without being influenced by the pattern.

Extending the SpawnGroups:

For custom actions and behaviors, you can extend the "BB.SpawnGroup" class.

You can override these methods:

- `protected void UpdatePosition(float deltaTime)`
    - This is ran every time the pattern is updated.
    - You can do whatever you want in this method. But it is meant to be used for moving the SpawnGroup

- `protected bool ShouldBeDestroyed()`
    - Checks if the SpawnGroup should be destroyed yet.
    - Override if you need the SpawnGroup to survive longer or match the lifetime of something else.

- `protected virtual void Spawn()`
    - Called when the SpapwnGroup supposed to spawn the child SpawnGroup.
    - Use SpawnSP() and RemoveSP() to create or remove SpawnGroup.

- `public void DrawSimulation(MeshGenerationContext mgc, Vector3 offset, float zoom)`
    - Draws the SpawnGroup on the Simulation View.
    - You shouldn't touch this if you are not sure about what you are doing.

Any Class that inherits BB.SpawnGroup will shows up in the editor. Though not every field will be serialized in the nodes. For the field of the class to be serialized, It must be

1. Public.
2. one of these types:
    a. Int
    b. Float
    c. Bool
    d. Vector2
    e. BB.SimulationSpace
    f. GameObject
    g. Color

Customize Bullet:

The BB.Bullet class from the Sample extends the BB.IBullet interface.

The Bullet prefab that is put in the editor must have a component that inherit the interface.

The interface have 4 method that needs to be implemented:

- `public void SetPosition(Vector2 position)`
    - Sets the position of the bullet.

- `public void SetRotation(float rotation)`
    - Sets the rotation of the bullet.

- `public void SetSpeed(float speed)`
    - Sets the speed of the bullet.

- `public void UpdateLocalPosition(Vector2 deltaPosition, float deltaAngle)`

    - If you want the bullet to follow and inherit the position and rotation of it's SpawnGroup.
    - Copy the BB.Bullet class from the Sample.

- `public IBullet SpawnBullet(SpawnGroup parentSpawnGroup)`
    - Creates a new bullet based on this one, and returns that.

There are more functions and behaviours that you want the bullet to have, but this is the bare minimum.

The BB.Bullet class in the samples demonstrate how to use the Interface. It also includes a object pooling manager.

HELP:

If you have any questions, need help, want to request features or report bugs. Feel free to contact me through my email at fetss1511@gmail.com