Procedurally propagating paths Documentation

Online Documentation

Please refer to the online documentation if possible.

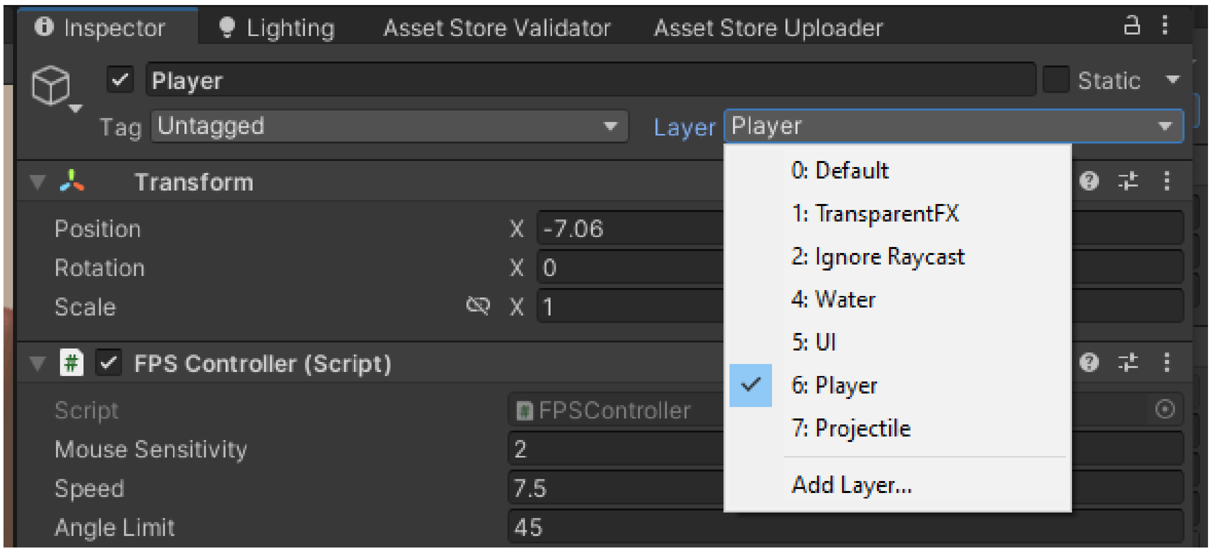
Get Started

⚠ **Important** ⚠

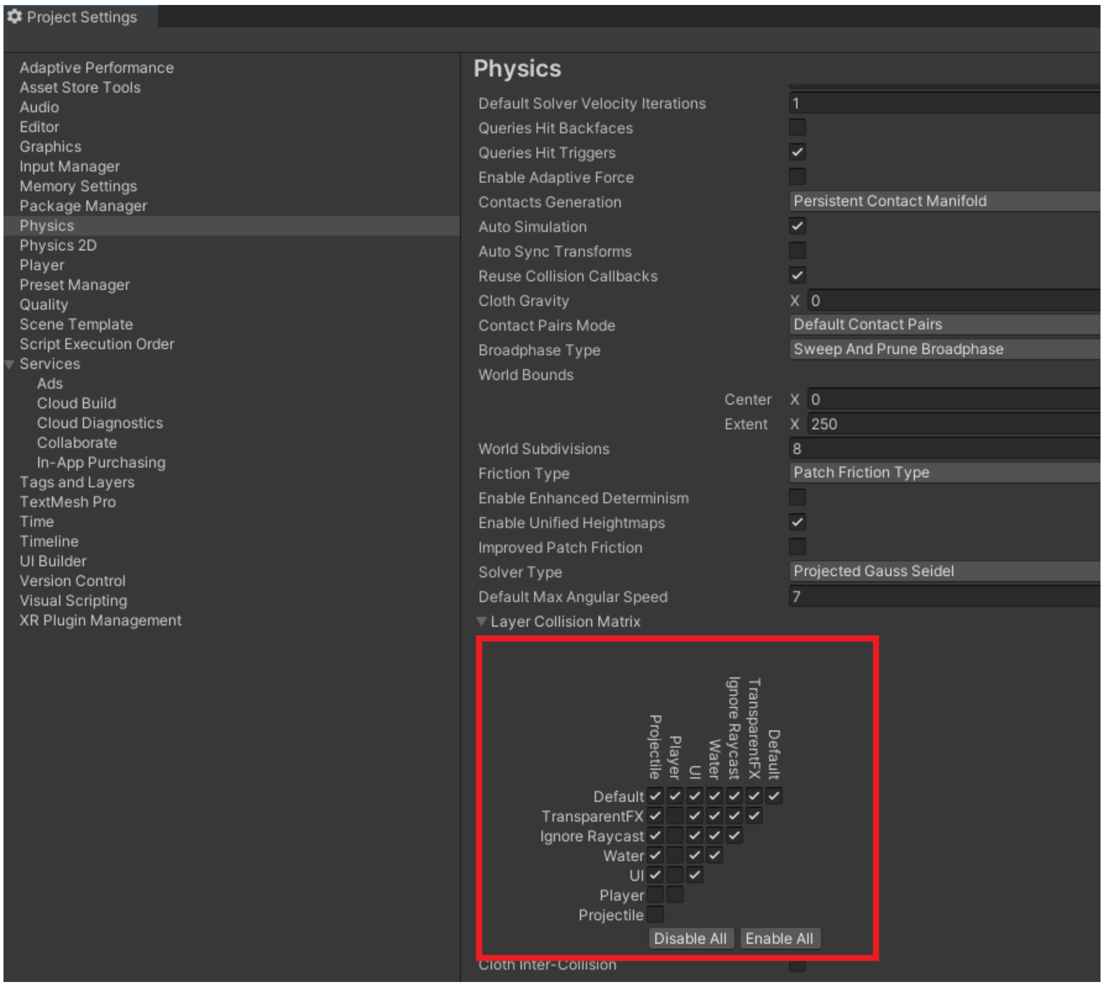
Set up the physics layers

If you don't set up the layers then the effects won't work.

- In the inspector, create two physical layers. One for the player and one for the projectile. You can just click any GO in your scene and add them from there:



- Then in the physics settings, set the new layers as follow:



Physics

Default Solver Velocity Iterations	1
Queries Hit Backfaces	☐
Queries Hit Triggers	✓
Enable Adaptive Force	☐
Contacts Generation	Persistent Contact Manifold
Auto Simulation	✓
Auto Sync Transforms	☐
Reuse Collision Callbacks	✓
Cloth Gravity	X 0
Contact Pairs Mode	Default Contact Pairs
Broadphase Type	Sweep And Prune Broadphase
World Bounds	
Center	X 0
Extent	X 250
World Subdivisions	8
Friction Type	Patch Friction Type
Enable Enhanced Determinism	☐
Enable Unified Heightmaps	✓
Improved Patch Friction	☐
Solver Type	Projected Gauss Seidel
Default Max Angular Speed	7

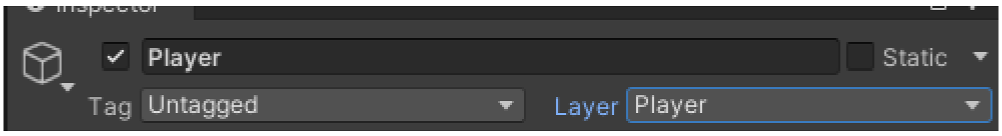
Layer Collision Matrix

	Projectile	Player	UI	Water	Ignore Raycast	TransparentFX	Default
Default	✓	✓	✓	✓	✓	✓	✓
TransparentFX	✓	☐	✓	✓	✓	✓	
Ignore Raycast	✓	☐	✓	✓	✓		
Water	✓	☐	✓	✓			
UI	✓	☐	✓				
Player	☐	☐					
Projectile	☐						

Disable All Enable All

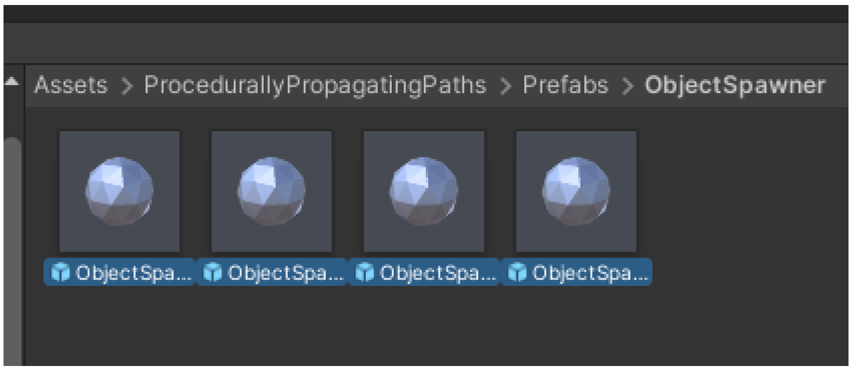
Cloth Inter-Collision

- Make sure you assign the Player layer to the player GO in the scene and its children.



✓ Player ☐ Static
Tag Untagged Layer Player

- Finally, assign the projectile layer to the object spawner prefabs in this folder, and also to the prefab located in the scene.



Assets > ProcedurallyPropagatingPaths > Prefabs > ObjectSpawner

Open the scene

Launch the scene CollisionBasedEffects.unity located in:
ProcedurallyPropagatingEffects/Scenes/CollisionBasedEffects.unity

Controls

- Move around with WASD
- Look around with the mouse
- Fire a projectile that will spawn an effect with Left-Click
- Change effect by pressing 1, 2 or 3 (not on the numpad)

You can also customize the effects by modifying the prefabs located in:
ProcedurallyPropagatingEffects/Prefabs/ObjectSpawner

You can learn how to modify them in the **Reference** section.

Alternative

You can also open the scene
ProcedurallyPropagatingEffects/Scenes/ClickBasedEffects.unity and enter play mode.
There you can move around the camera like you would in the scene window, and left click to spawn some ants.
You can also change the effect to whatever you want and test it in there.

Reference

Object Spawner

Properties

Property	Type	Description
Layer Mask	LayerMask	The layer on which the raycast will be performed to find the anchor points for the paths
Number of Paths	uint	The number of paths that will be created from the initial point
Divergence	float [0;1]	Closer to 0 means that the angle between the path will be initially evenly distributed, closer to 1 makes it more random
Angle Range	float [0;360]	Defines the portion of the initial circle that will be used to create the paths.
Iteration Angle Divergence	float [0;360]	Defines the maximum angle between two steps of a path
Are Paths Sorted	boolean	Defines if the paths should be used in the geometrical order or if they should be randomly shuffled
Spawnables	Spawnables	The objects that can be spawned and used along the paths. More details here Spawnables
Random Spawn Delay	float > 0 (seconds)	The objects will spawn between a delay of 0 and this value

Note

The length of the paths is not defined here, but in the subclasses. This is because the length of the paths is implicit for the static object spawner, but explicit for the dynamic one.

Examples

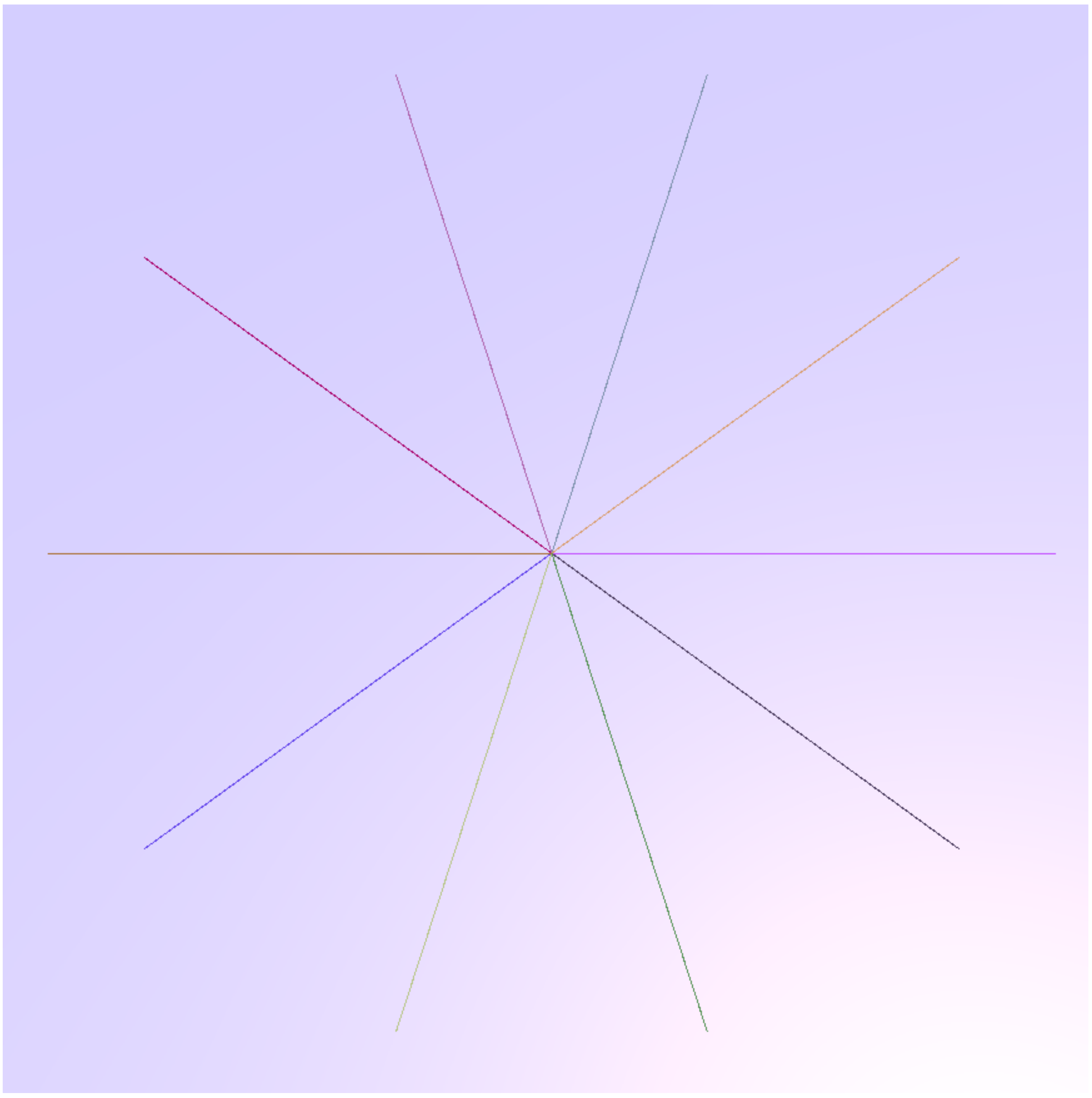
Let's take an initial set of properties and construct paths on a flat surface to make the visualization easy. Then, let's tweak them one by one to see how the paths are affected by these properties. For the sake of this example, we consider paths of length 3. This means that each path is constituted of 3 anchor points, or 2 segments.
Considering these properties:

- Layer Mask : Default
- Number of Paths : 10
- Divergence : 0
- Angle Range: 360
- Iteration Angle Divergence : 0
- Are Paths Sorted : True

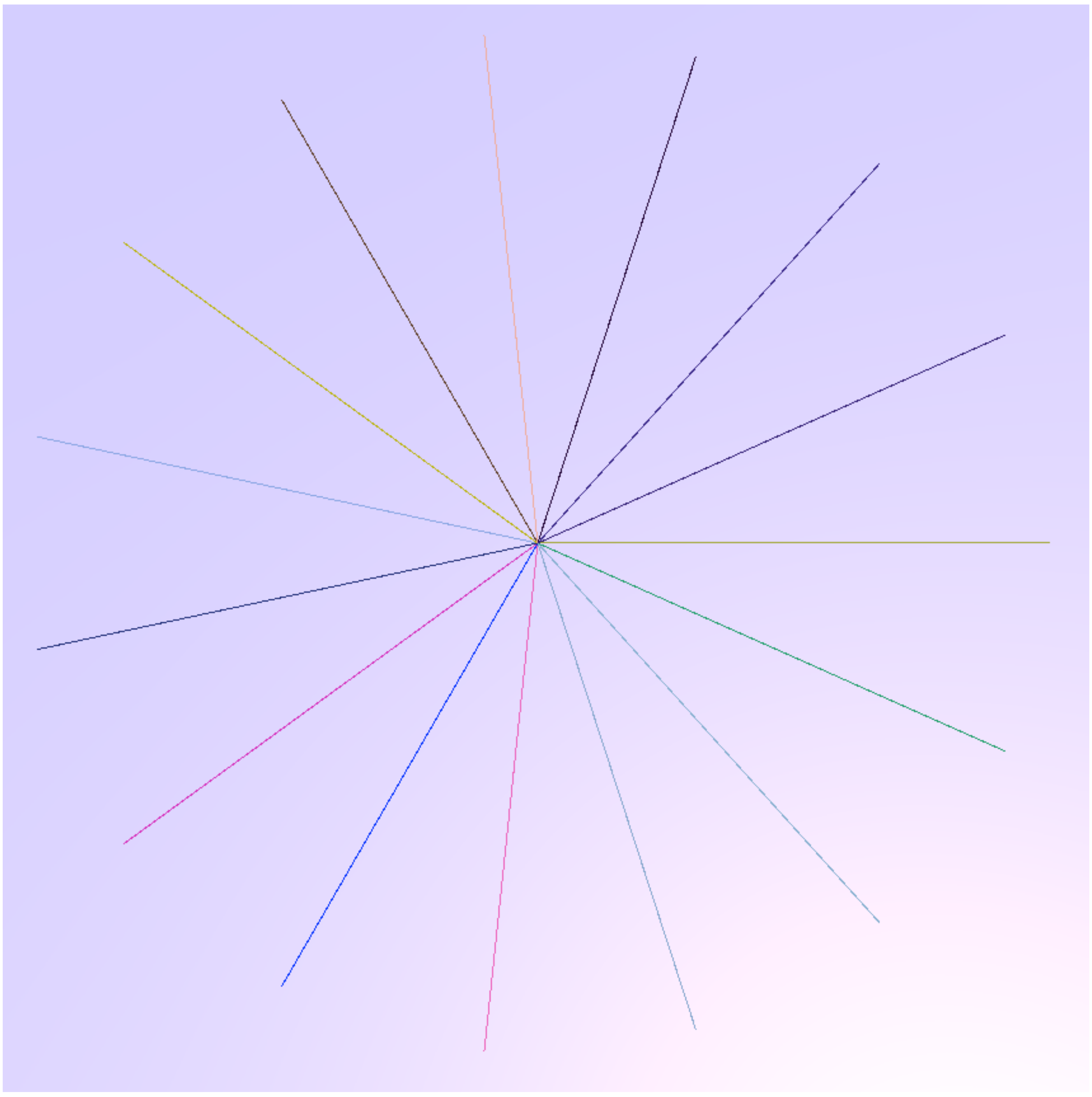
And these that will only be used in the end of the page:

- Spawnables : Ant.Prefab
- Random Spawn Delay : 3

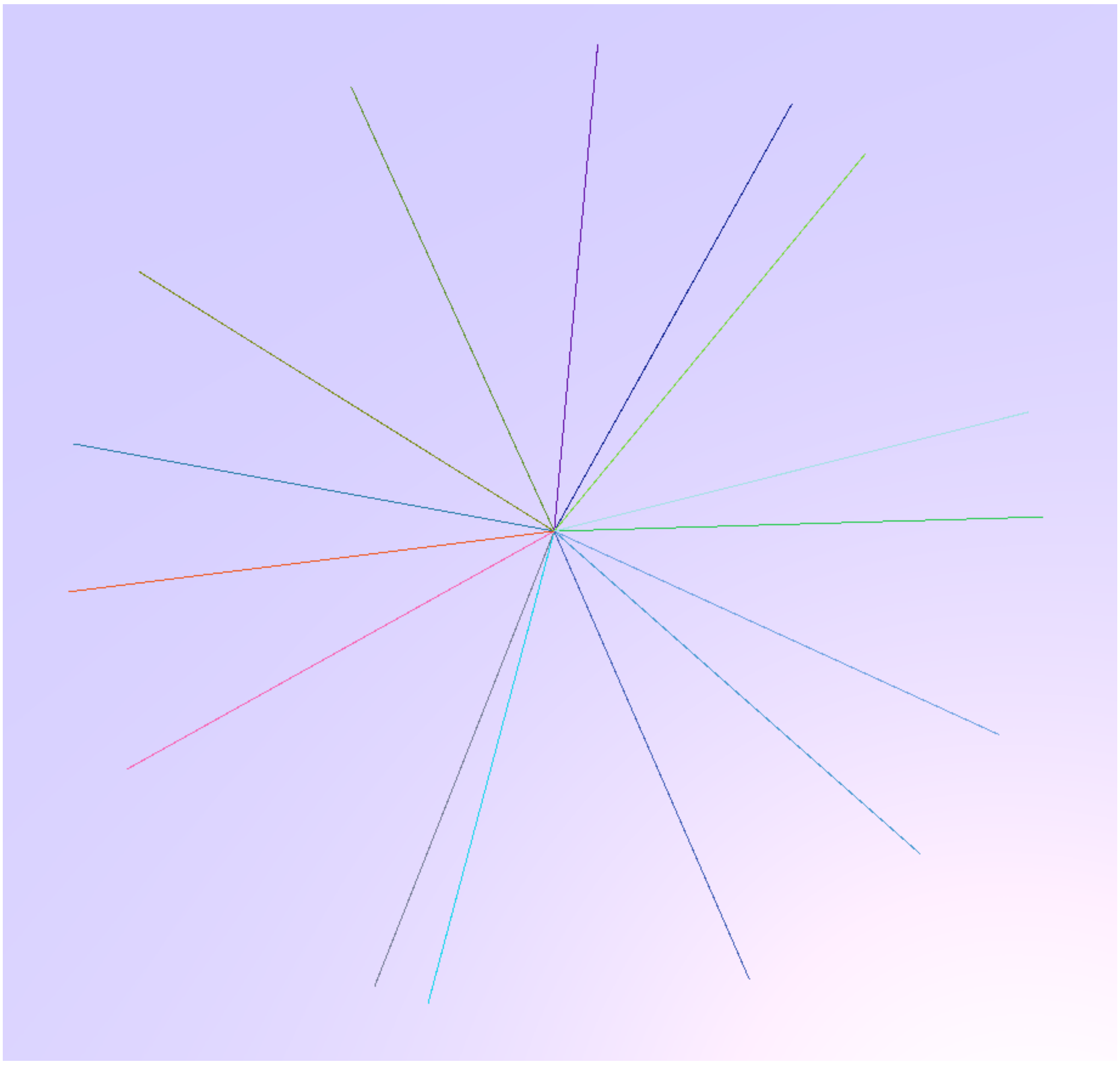
We can obtain such paths which go in a straight line, they are equally distributed around the full circle of the initial point:



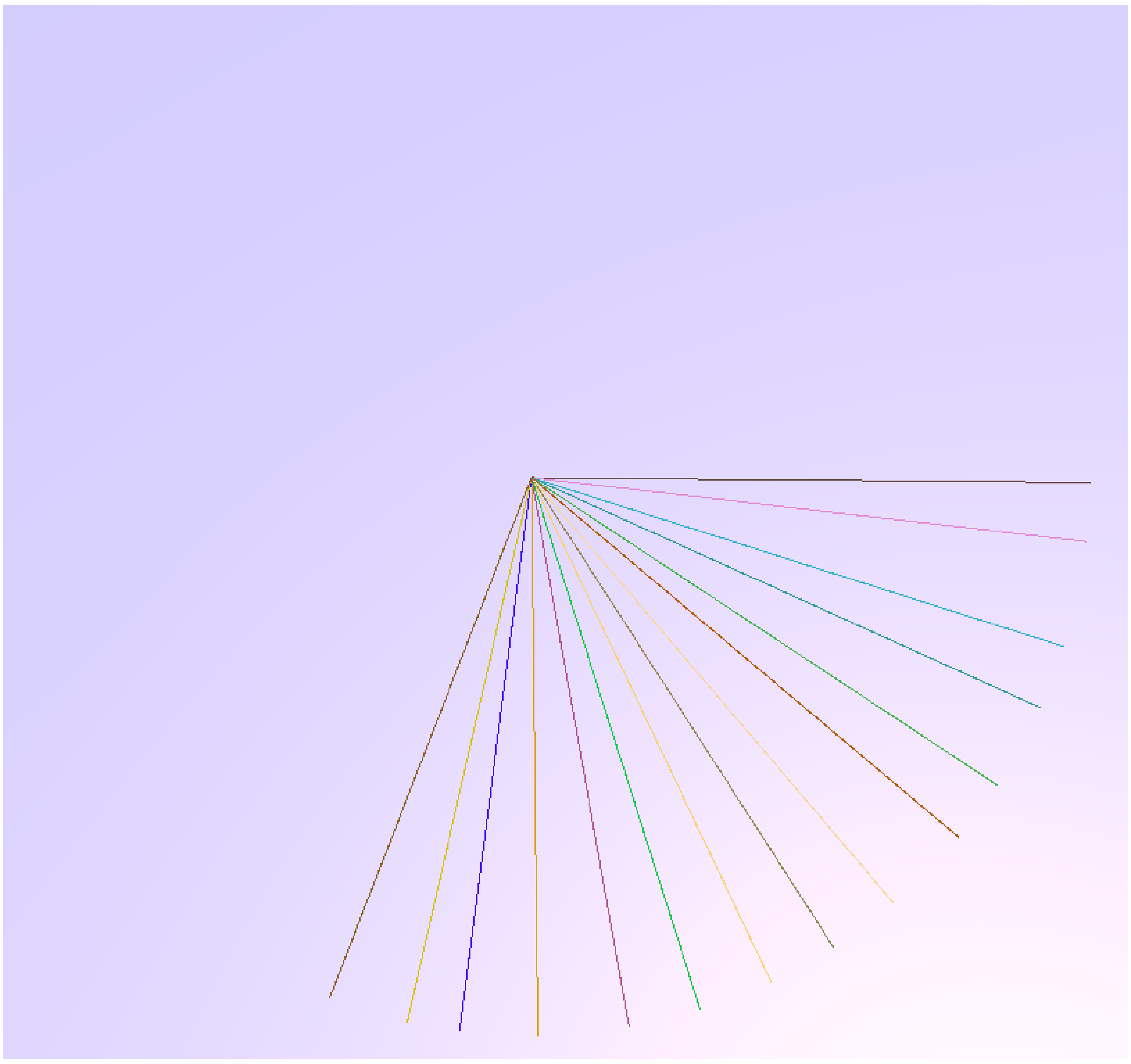
Now let's bump up the number of paths to 15, as expected we just have 15 paths instead of 10:



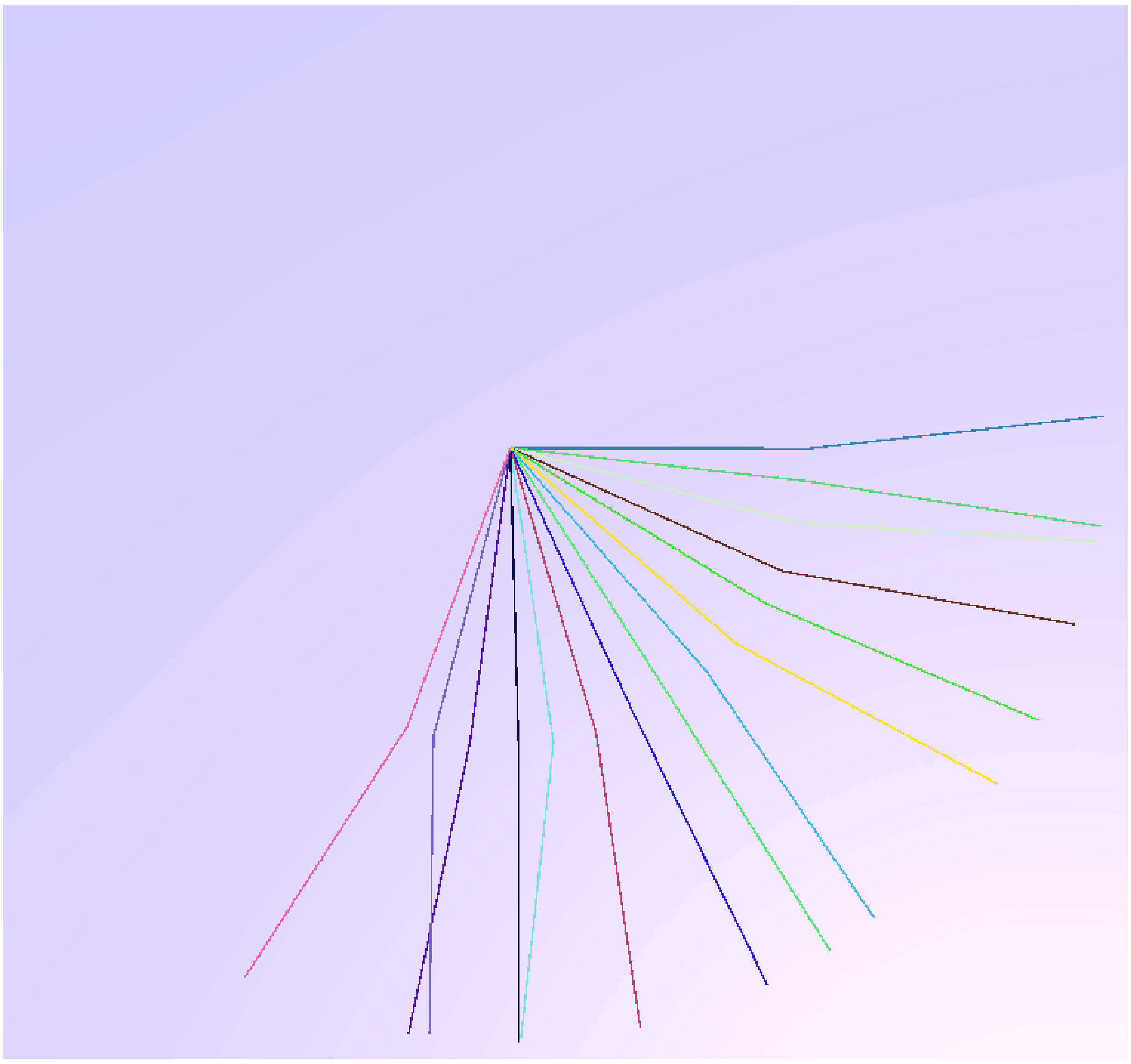
Now let's dive into the more mysterious parameters. While keeping all previous parameters, we can increase the divergence to 1. As you can see, the paths are no longer equally distributed. The closer we get to 0, the closer we are from the previous picture, and the closer to 1 to more random it gets.



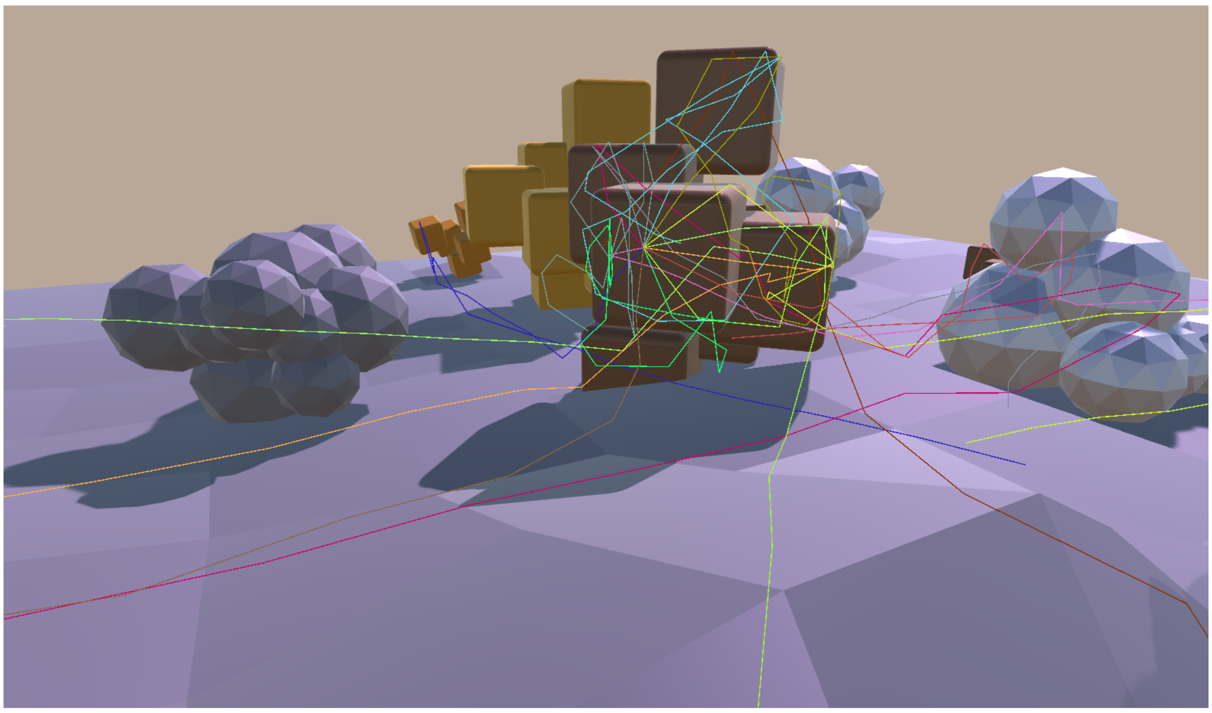
Let's still keep other parameters as they were, but change the angle range to 120 degrees. We still have 15 straight paths that are built but they are now closer to each other as only a third of the circle around the initial point is used. We can still see that the divergence is 1 because the angle between them is not the same. This is how it looks like:



Now, a very key parameter that is going to go away from these straight lines and have random little paths: Iteration Angle Divergence. For each iteration on the path, we allow it to diverge from the straight line of an angle from 0 to the value of Iteration Angle Divergence. If we set it to 15 while keeping the same parameters from before, this is what we get:



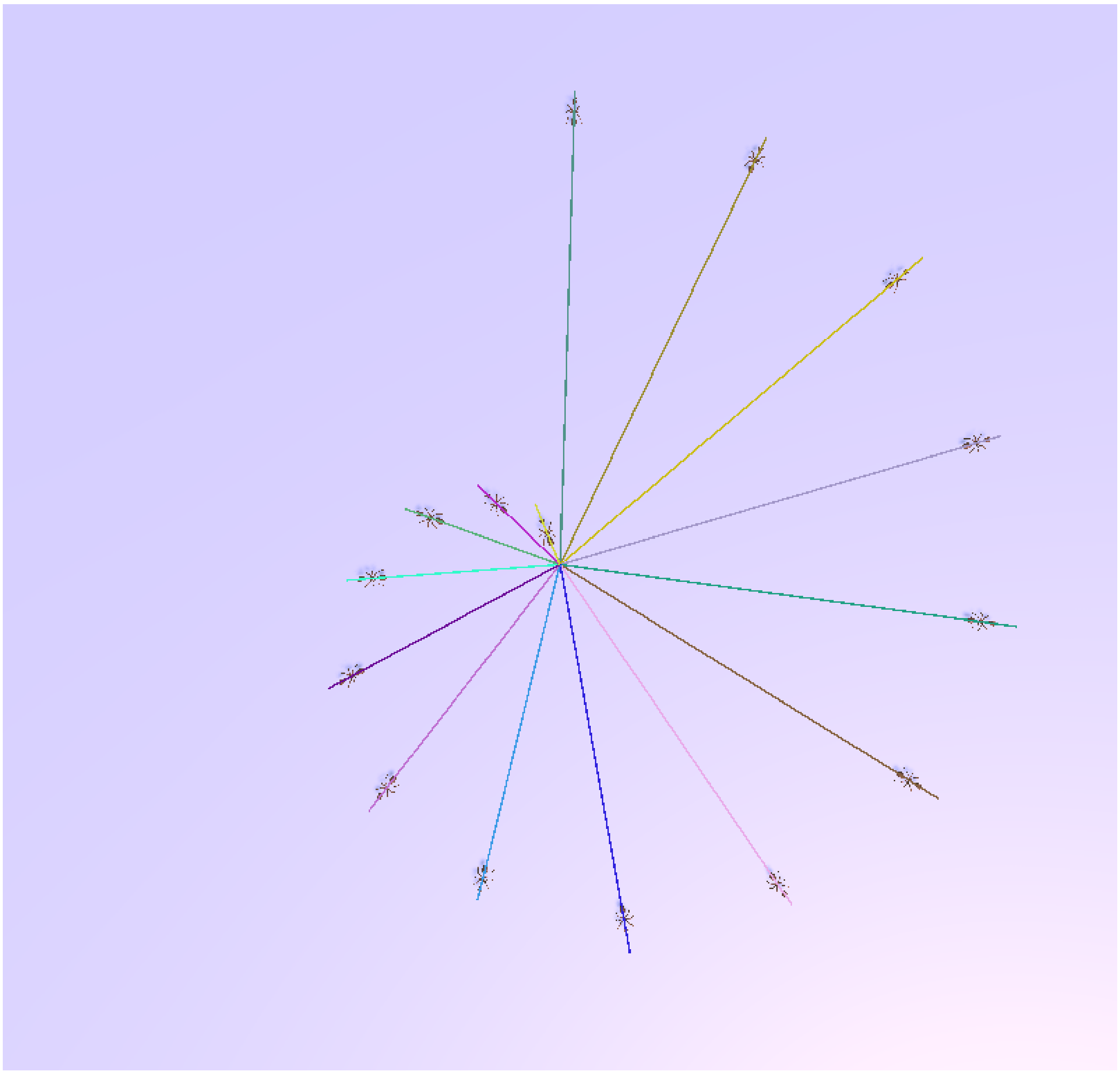
Finally, some procedural looking stuff! Each segment is constituted of 3 anchor points, and they all share the initial point as the first anchor point. We can then play around with the parameters and look at it on a 3D environment, and obtain results such as this:



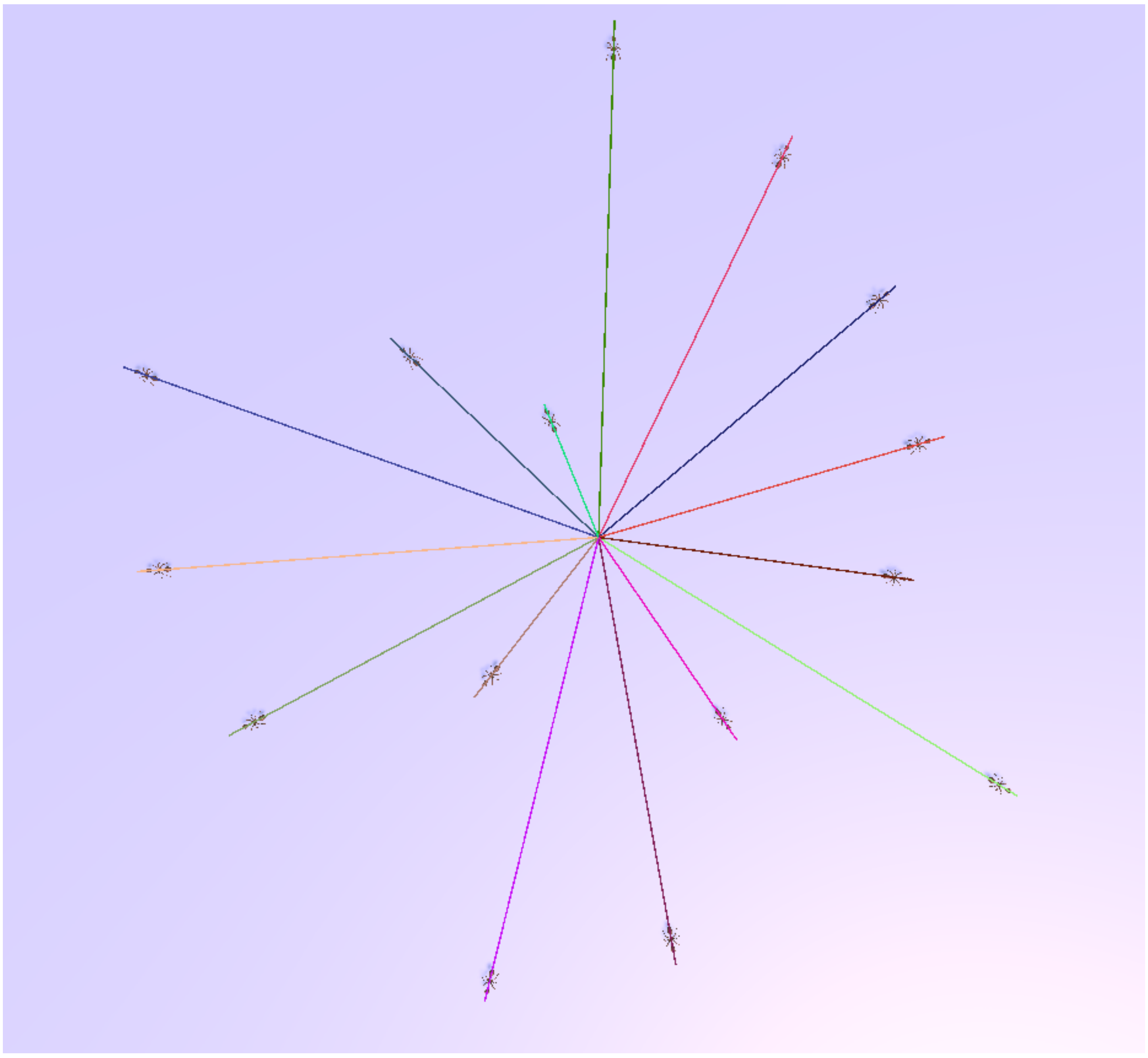
It is pretty much the same as what we've seen before, but with longer paths (20 anchor points) and wrapped around 3D colliders.

Finally, the last parameter Are Paths Sorted will determine, in the process of the effects you create, if the paths should be used in the natural geometrical order, or randomly. I am aware that "natural geometrical order" is an expression I made up, so here is an example with ants to understand better.
In this example the ants are spawned with the Random Spawn Delay set to 3 seconds, and then enjoy their lives along the paths. When the paths are sorted, we give the first ant to spawn the first path, the second ant the second path etc. In the other scenario, we just give them a random path.

With Are Paths Sorted = true:



With Are Paths Sorted = false:



Dynamic Object Spawner

Principle

The goal of this effect is to use the paths to create some dynamic effects. It instantiates objects at the initial point, these objects can be anything but in the example I created I chose ants and fog particles. Then each of these object is going to spawn after a random delay, and then travel along one of the path at a given speed, for a given lifetime.
The paths are created on the go so they will adapt to moving colliders, as long as the relative speed between the objects and the colliders is not too big, then weird behaviors could occur. You can diminish that by having a higher Objects Speed value or a smaller Preferred Step Distance

Properties

Property	Type	Description
Preferred Step Distance	float >= 0.01	The distance between each anchor point of the path, can differ from it if it is impossible to find an anchor point in that range
Objects Lifetime	float >= 0	The time in seconds before the object gets destroyed after they spawned
Objects Speed	float >= 0	The speed (arbitrary unit) at which the objects travel along the paths
Destroy Duration	float >= 0	The time taken for the object to scale down to 0 before getting destroyed

Examples

Please check to the online documentation for examples as they are videos

Static Object Spawner

Principle

This object spawner is using the paths to spawn static objects. These objects will be using the anchor points of all the paths, including the initial anchor point. In this effect, the preferred step distance varies between a minimal and a maximal value, to allow for a more organic feeling.
A lot of properties allow to custom that effect further, and it could be expended even more.

Properties

Property	Type	Description
Objects Count To Spawn	uint	The amount of objects that will be spread among all the paths
Min Preferred Step Distance	float >= 0.01	The distance between each step will be, if possible, between the min and max step distance.
Max Preferred Step Distance	float >= Min Preferred Step Distance	The distance between each step will be, if possible, between the min and max step distance.
Should Randomize Rotation	bool	If true, will add a random rotation to the spawned object along the up axis
Parent Spawned Object	bool	If true, the spawned objects will be a child of the object where the anchor point was found
Normal Shift	float	Use this to give an offset along the normal to the spawned objects. Note that there is already a constant offset that is setup in the AnchorPoint.cs script
Normal Axis For Spawned Objects	Transform	Can be use to make object spawn on a specific sides of objects
Scalar Product Tolerance	float [-1;1]	The closer to 1, the more the anchor points' normal has to be close to the Normal Axis For Spawned Objects for the object to be spawned

Examples

Please check to the online documentation for examples as most of them are videos