Pool Spawner Pro

Lustrum Games

v.1.1

Contents

Introduction

Why Use This Asset In Particular?

This asset is powered by the object pooling system. It is much more powerful that Unity’s built-in Instantiate() and Destroy() functions which use a lot of memory. The Instantiate() function creates new instances of objects every time it is called. However, object pooling re-uses the objects it has created before the game even started giving the users a smooth and lag-free experience. The best part about this asset is that it is platform-independent. It can run the smoothly in mobile, pc and console devices.

Purpose of This Asset

This asset serves the purpose of fulfilling the needs of beginning developers and sparing time for more advanced developers who are looking to integrate the frequently used functionality for their games. It is designed to be set up easily and quickly for your projects in a drag-and-drop manner.

Demonstrations

This assets contains a demo scene for each type of spawning. The possibilities in the demos are just examples and are not only limited to the ones shown in the demos. This asset can be used for many different things but the demos only show the most popular possible use cases.

Main Contents of This Asset

This asset features three different scripts that handle different types of object spawning. The first one is DynamicSpawn.cs and it handles spawning infinite worlds like in Temple Run video game. The second one is FixedSpawn.cs and it spawns objects from a specified spot and it can be used for many things such as firing bullets or objects that need to be spawned from a fixed or specified location. The third script is AreaSpawn.cs and it handles spawning objects in a specified area. It can be used for spawning soldiers like in Lords Mobile video game and spawning AI bots in groups.

License

You may use this asset for any of your projects whether they are for personal use or commercial use. You may also modify the code within this asset to extend its capabilities. However, you are not permitted to redistribute any code that resides in this asset.

Refunds

You are qualified for a refund if the asset is not yet downloaded and/or imported since the time of your purchase.

Dynamic Spawn

DynamicSpawn.cs provides you the ability to spawn objects after the other. You can create as many different pools you want and each of them are independent of one another. The best part about this script is that once the player reaches the distance you specify in the inspector, the world objects and the player are reset back to the origin of the world which prevents floating-point issues which exist in all devices.

Some video games that use a similar system:

- Subway Surfers
- Temple Run
- Jetpack Joyride
- Sonic Dash
- Alto’s Adventure

Options

You can set multiple input parameters from the inspector to control generation. Also, this asset contains key options that you should know:

1. Spawn Point Object (Required)

    You can specify this object within the DynamicSpawn.cs inspector under the ‘Points’ dropdown. The Spawn Point object is responsible for determining when to spawn the new chunks of the world. It travels in front of the player and is used to check whether it’s in front of the last spawned object.

2. Deactivation Point Object (Required)

You can specify this object within the DynamicSpawn.cs inspector under the ‘Points’ dropdown. The Deactivation Point object is responsible for determining when to disable the objects that are behind the player. It travels behind the player to disable objects that are behind the player by a specified amount.

3. Parent Object (Optional)

This object is responsible for grouping the spawned objects under itself to make it collapsible to make space in the hierarchy window.

4. Reset Transition Object (Optional)

As you can set a max travel distance, when the player reaches that distance, the world is reset/ brought back to the center/origin of the world. During that process, the users might see the world moving back and it might look like a glitch for them. Therefore, this object eliminates that issue by traveling with the player from the maximum distance to the origin. As an example, this object can be a tunnel where nothing is visible when inside it.

Uses

1. Infinite Platforms

   Using this system as a platform spawner or generator might be the best use case scenario. It provides all of the necessities for accomplishing this type of infinite platform spawner.

2. Loops

   Using an external script that controls the different parameter in DynamicSpawn.cs, you can create circular loops that go on forever.

3. Zig-Zag Game

   This system is also a good option for creating a zig-zag game as it constantly generates new objects and disables the ones in the back.

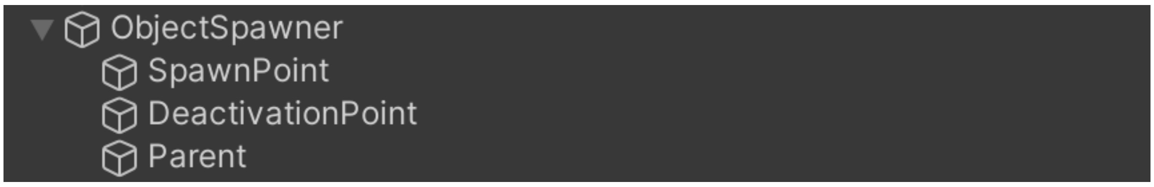
4. Your Imagination

   Although this system is made for generating infinite platforms, we left some space for your imagination. You can control the parameters at runtime to extend the capabilities of this system.

Setup

You can add this in three simple steps:

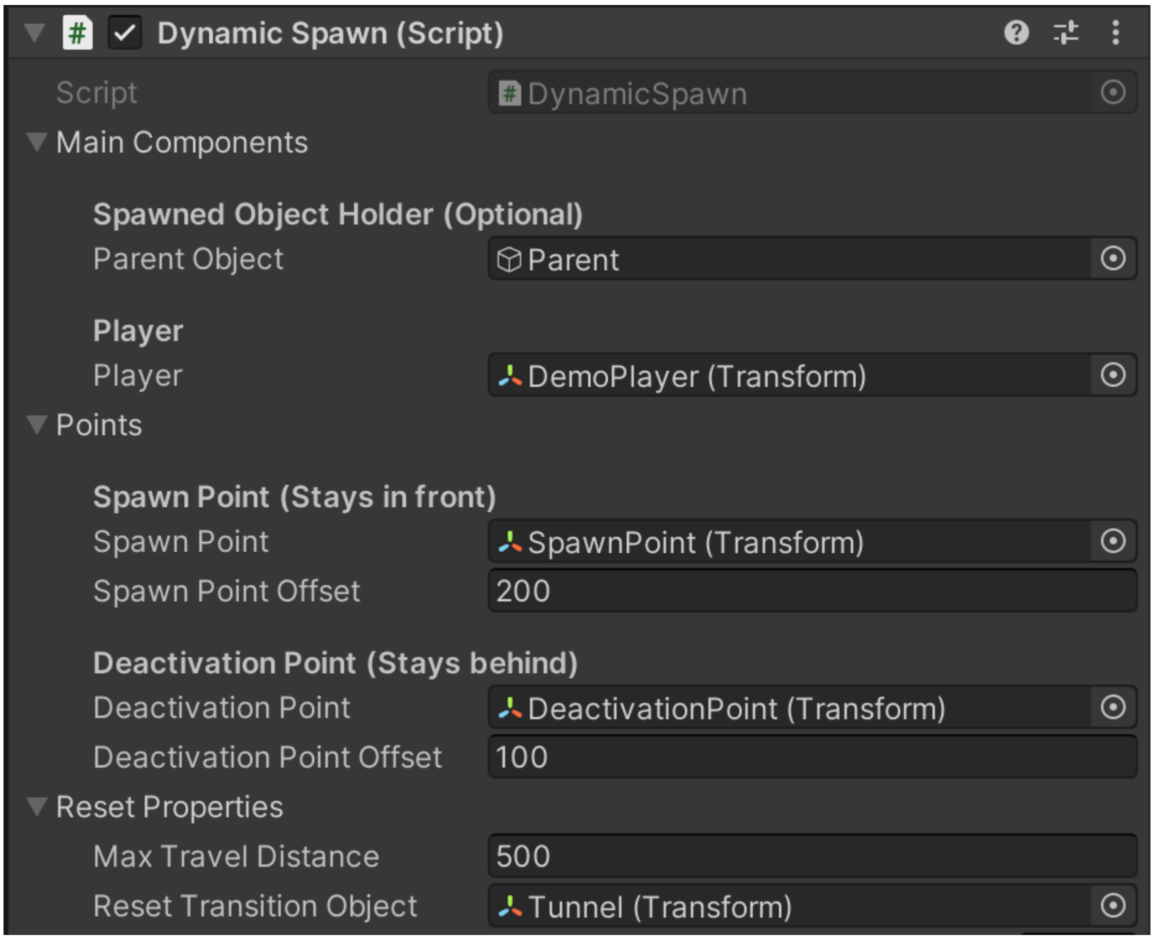
1. Configure the objects in the hierarchy and add the DynamicSpawn.cs script on the parent object.



```
ObjectSpawner
    SpawnPoint
    DeactivationPoint
    Parent
```

Note: You don't have to nest the GameObjects under ObjectSpawner GameObject.

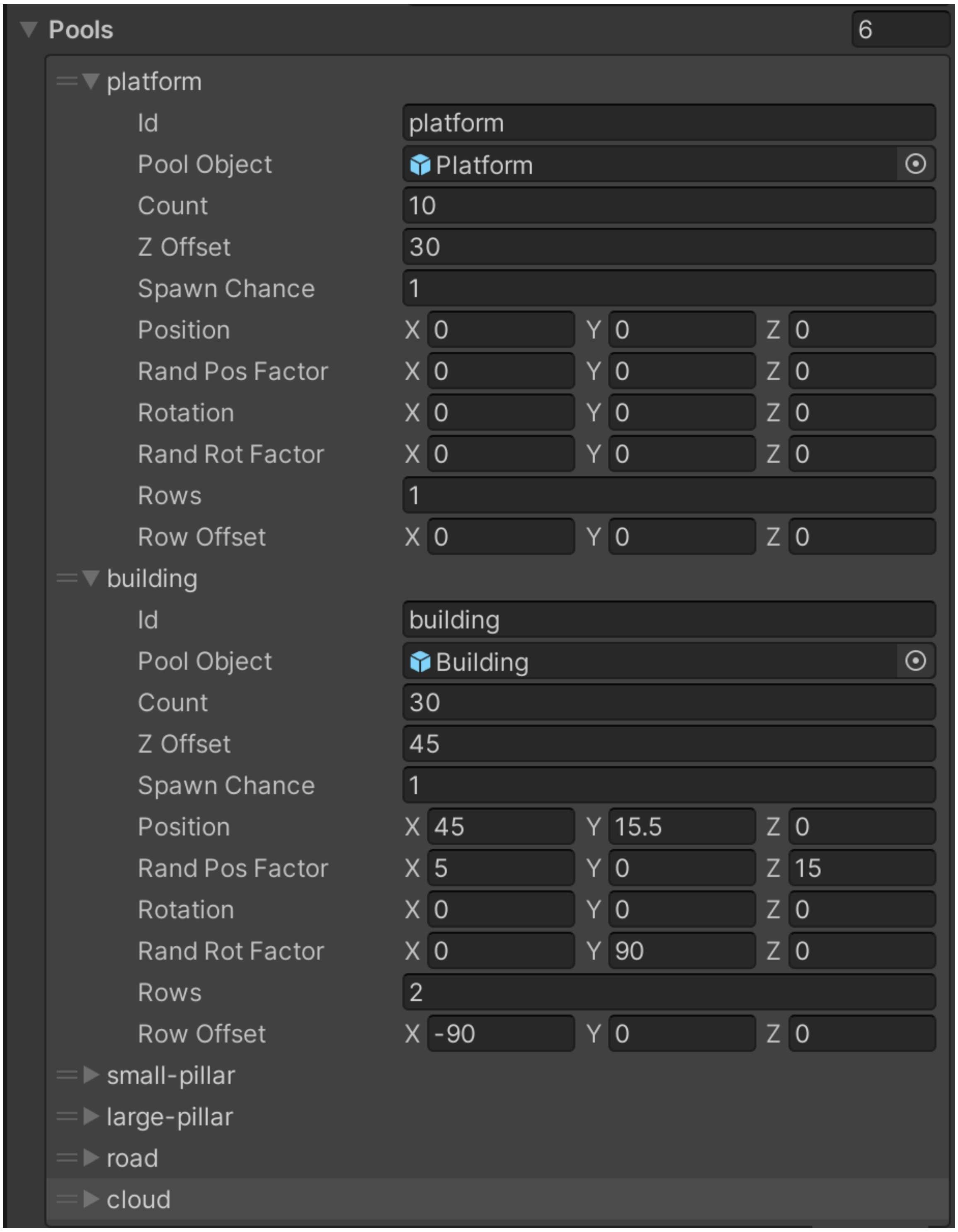
2. Assign the objects and values in the general fields.



Dynamic Spawn (Script)

Script	DynamicSpawn
Main Components	
Spawned Object Holder (Optional)	
Parent Object	Parent
Player	
Player	DemoPlayer (Transform)
Points	
Spawn Point (Stays in front)	
Spawn Point	SpawnPoint (Transform)
Spawn Point Offset	200
Deactivation Point (Stays behind)	
Deactivation Point	DeactivationPoint (Transform)
Deactivation Point Offset	100
Reset Properties	
Max Travel Distance	500
Reset Transition Object	Tunnel (Transform)

3. Create pools of objects and configure them based on your needs.



Pools	6		
platform			
Id	platform		
Pool Object	Platform		
Count	10		
Z Offset	30		
Spawn Chance	1		
Position	X 0	Y 0	Z 0
Rand Pos Factor	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Rand Rot Factor	X 0	Y 0	Z 0
Rows	1		
Row Offset	X 0	Y 0	Z 0
building			
Id	building		
Pool Object	Building		
Count	30		
Z Offset	45		
Spawn Chance	1		
Position	X 45	Y 15.5	Z 0
Rand Pos Factor	X 5	Y 0	Z 15
Rotation	X 0	Y 0	Z 0
Rand Rot Factor	X 0	Y 90	Z 0
Rows	2		
Row Offset	X -90	Y 0	Z 0
small-pillar			
large-pillar			
road			
cloud			

Note: Each ID should be unique. Also, if there are not enough objects spawning or they are being disabled too early, increase the value of “count.”

Fixed Spawn

FixedSpawn.cs provides you the ability to freely spawn or generate objects anywhere you want. It also handles nesting objects under a parent so that they travel with that parent object and their positions are local to that parent object. You can spawn objects by a custom key press or tell it to automatically keep spawning forever.

Options

Similar to DynamicSpawn.cs, you can similarly configure the pool. However, FixedSpawn.cs contains other options that are not present in other scripts.

1. <u>Parent Object (Optional)</u>

    This object is responsible for grouping the spawned objects under itself to make it collapsible to make space in the hierarchy window.

2. <u>Disable By Distance (Optional)</u>

    Disable an object once it has passed a certain distance in the forward direction.

Uses

1. Bullets

   You can assign a gun GameObject as the parent of the bullets in pools. After that you can add force to the bullets once they are spawned.

2. Fountain

   As shown in the demo, you can create fountains. They can be made of lava, water, or something else. This option can enhance the graphics of your game.

3. Rain Effects

   Although this effect can be achieved with particles, it is also possible to achieve with this system. You just need to give the objects some altitude and gravity.

4. Simulations

   Some simulations require a lot of props and characters to be generated in the scene. For example, a zombie apocalypse simulations require hundreds and thousands of zombies which can be achieved with this system.

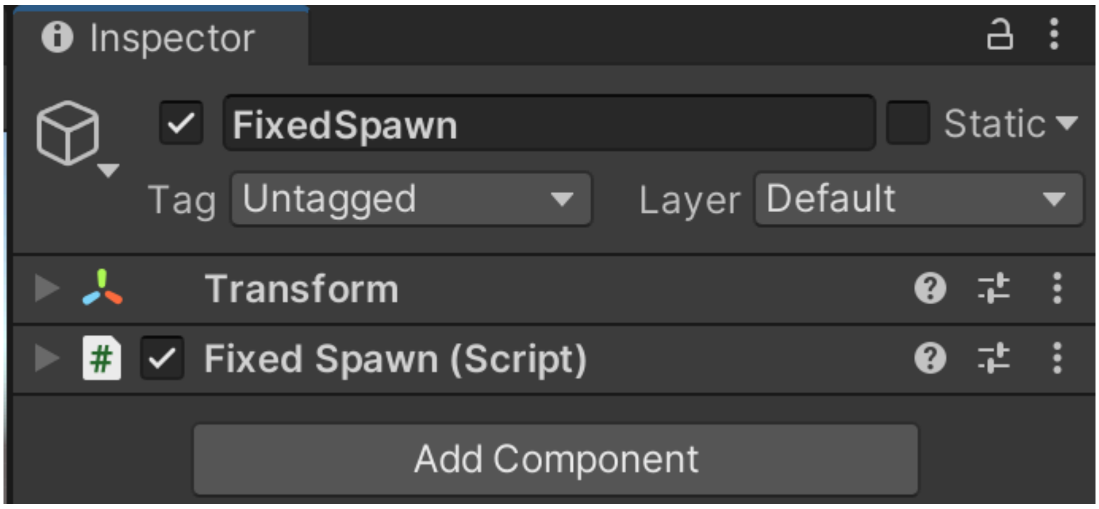
5. Your Imagination

   This system is even more generic than the other two systems in this asset package. The options above are there just to give you an idea about some of the possibilities. The possibilities with this system are endless and you just got to come up with them.

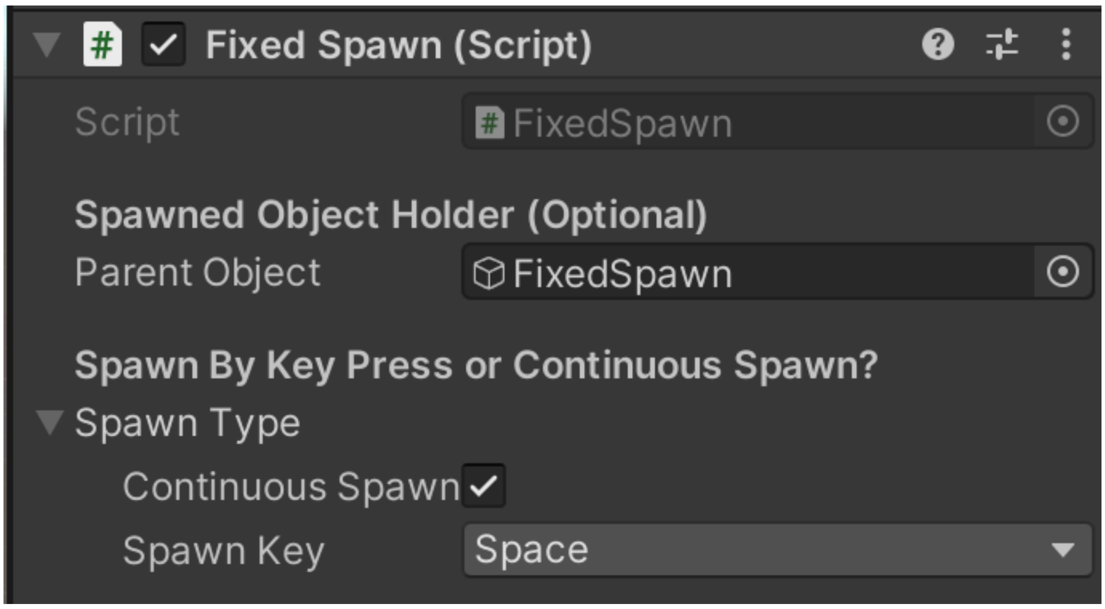
Setup

Unlike Dynamic Spawn, you don't need any child objects. You can set up this system using three simple steps:

1. Create a GameObject and attach FixedSpawn.cs script to it.



2. Configure the general fields.


Fixed Spawn (Script)

Script: FixedSpawn

Spawned Object Holder (Optional)

Parent Object: FixedSpawn

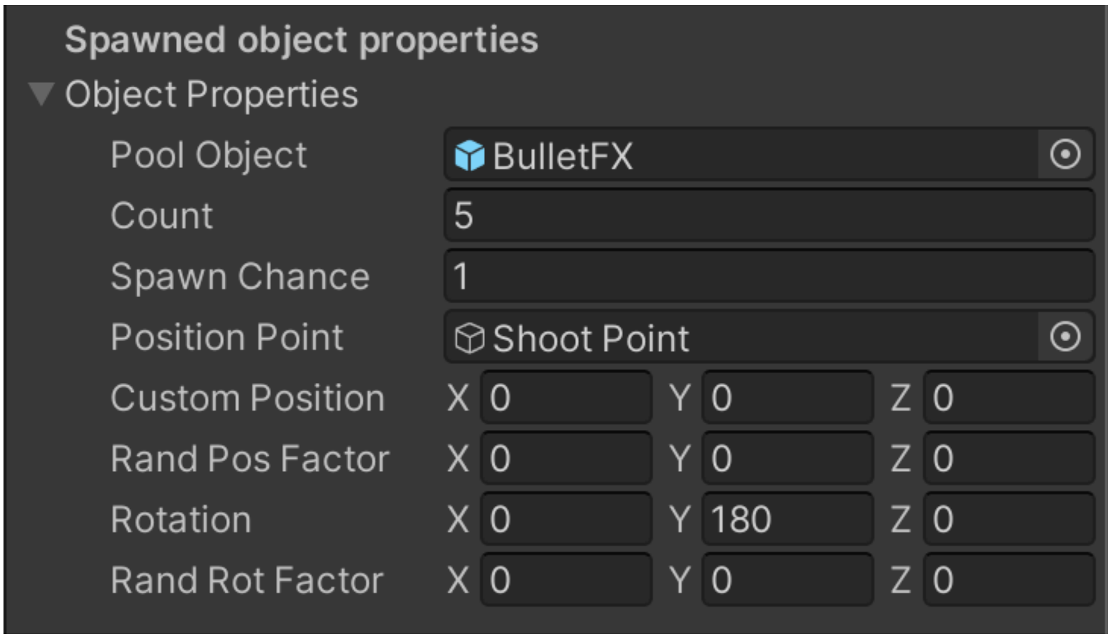
Spawn By Key Press or Continuous Spawn?

Spawn Type

Continuous Spawn ✓

Spawn Key: Space

3. Configure the pool/pool object properties.



Spawned object properties

▼ Object Properties

Pool Object	BulletFX		
Count	5		
Spawn Chance	1		
Position Point	Shoot Point		
Custom Position	X 0	Y 0	Z 0
Rand Pos Factor	X 0	Y 0	Z 0
Rotation	X 0	Y 180	Z 0
Rand Rot Factor	X 0	Y 0	Z 0

Area Spawn

AreaSpawn.cs provides you the ability to organize objects in even spaces in a given area. It also handles nesting objects under a parent so that they travel with that parent objects and their positions are local to that parent object. You can also choose to change or modify the area size, object offset and other properties by checking the isModify boolean parameter through the script's inspector window.

Options

This system has slightly different pool properties. It has a similar options and a different option.

1. Parent Object (Optional)

   This object is responsible for grouping the spawned objects under itself to make it collapsible to make space in the hierarchy window.

2. isModify Boolean

   With this variable set to true, you can directly modify the area properties at runtime while the game is running. You can modify the area size, object offset, object scale and object rotation.

Uses

1. Army

    This system is perfect for generating army characters and positioning them neatly and evenly. Then, it can be used for a war game as an example.

2. City Buildings.

    Typically, city buildings are evenly apart from one another. This system can save you some time by taking care of positioning the buildings in precisely and evenly.

3. Fence

    Since you can control the area size, you can make it narrow so that there is only one row for the fences. If the fences are overlapping, you can change their offset through this script.

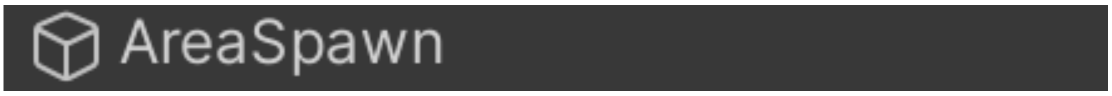
4. Your Imagination

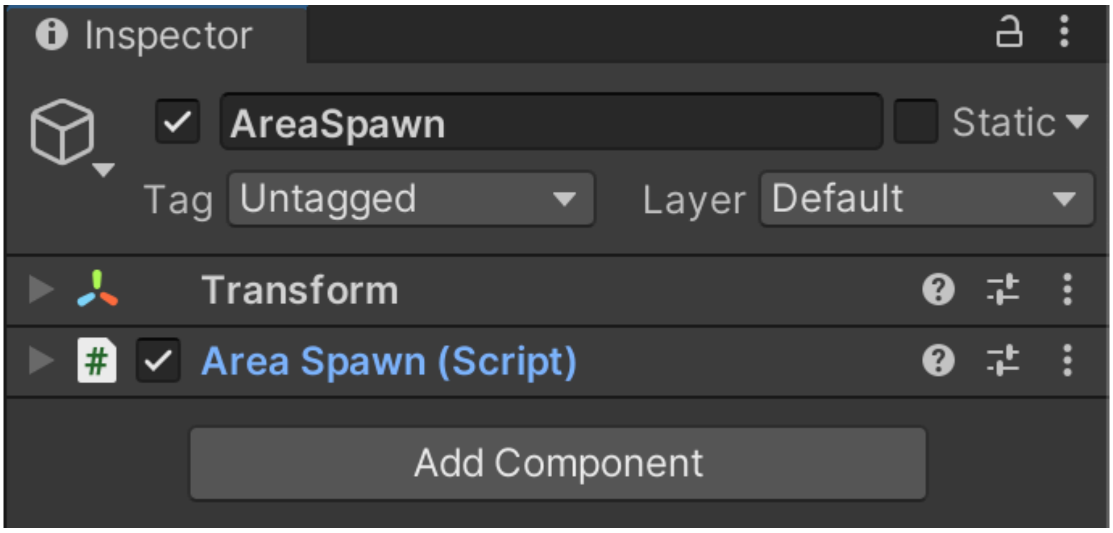
    This system is not only made for only one purpose. You can use this for many different things that require organizing and positioning in a precise manner.

Setup

Just like Fixed Spawn, you don’t need any child objects. You can set up this system using three simple steps:

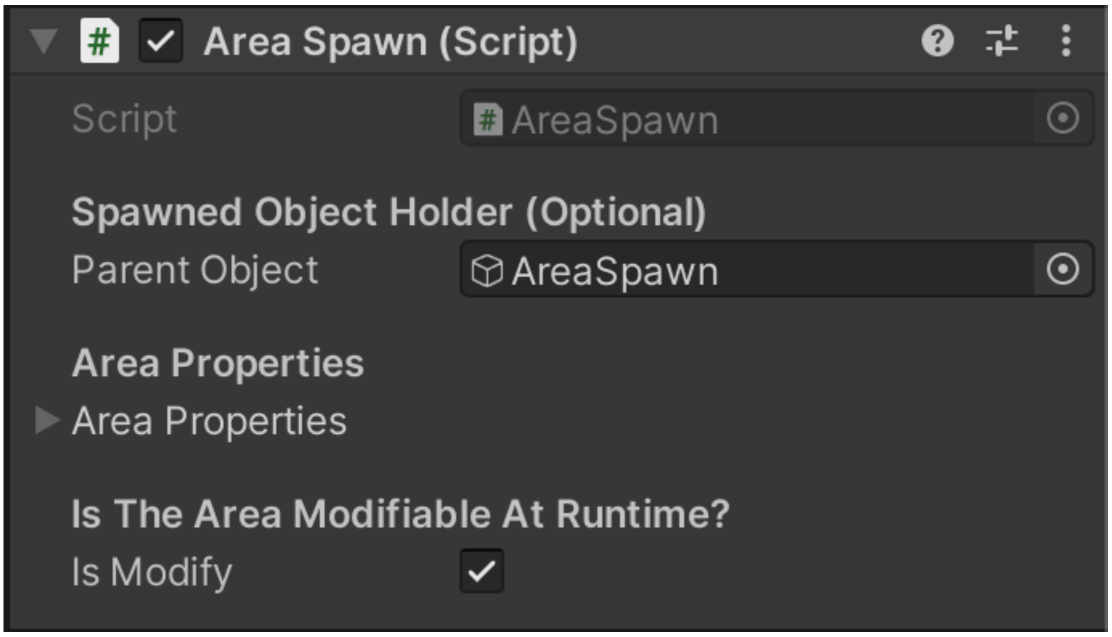
1. Create a GameObject and attach AreaSpawn.cs script to it.





Tag: Untagged
Layer: Default

2. Configure the general fields.



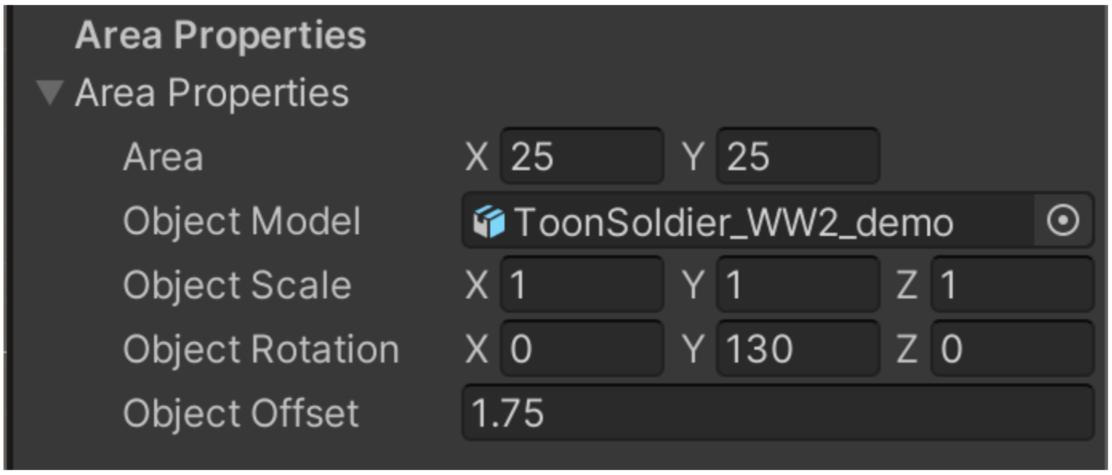
Script: AreaSpawn

Spawned Object Holder (Optional)
Parent Object: AreaSpawn

Area Properties
Area Properties

Is The Area Modifiable At Runtime?
Is Modify: ✓

3. Configure the pool/pool object properties.



Area Properties

▼ Area Properties

Area	X 25	Y 25	
Object Model	ToonSoldier_WW2_demo		
Object Scale	X 1	Y 1	Z 1
Object Rotation	X 0	Y 130	Z 0
Object Offset	1.75		

Support

First of all, we want to thank you so much for your support in purchasing this asset. Your purchase helps us post new updates to this asset and create new tools. Thank you!

If you encounter any issues or want to suggest or request a new feature, send us an email at LustrumGames@gmail.com or post an issue on our GitHub page.