User Guide

Ultimate Spawner

Spawning made easy

Trivial Interactive

Version 1.3.4

Features

- Quickly setup enemy spawn points and waves
- Quickly locate free spawn points at any time
- Quickly setup a finite number of predefined waves
- Quickly setup an infinite number of increasingly difficult waves
- Quickly setup boss waves.
- Specify spawn areas for larger maps to ensure enemies are never far from the player
- Spawn enemy types based on a chance value
- Support for networking using uNet with examples.
- Supports object pooling using our UltimatePooling asset.
- Highly customizable
- Full examples of all features from basic to advanced
- Suitable for 3D and 2D games
- Fully commented C# source included
- Many other useful features for spawning and enemy waves

Version Changes

1.3.4

The wave controller now has the ability to used mixed mode wave advancement. This means you are able to manually configure a number of starting waves allowing for greater control and then allow the continuous waves to take over on completion providing an infinite wave spawner with greater control over the starting waves.

1.3.0

Support for boss waves has been added to the wave controller. Boss waves are essentially special waves that are designed to increase the difficulty and typically make use of different enemy types with much greater health values. See the 'Wave Controller' section for more information.

1.2.0

Pooling support has been directly incorporated into Ultimate Spawner using our separate asset 'Ultimate Pooling'. Pooling is enabled by adding a scripting define symbol to the project and requires that you have purchased 'Ultimate Pooling'.

The method 'SpawnableManager.informSpawnableDestroyed' has been modified to accept an additional optional parameter which can allow the manager to handle the destruction of the object. This change has been added to better incorporate pooling support and should not break upgrading projects.

1.1.0

As of version 1.1.0, Ultimate Spawner now combines all networking code into the same scripts and uses pre-processor directives to determine whether or not to include the networking code. This removes a lot of duplicated code caused by providing networked script alternatives and simplifies the API due to the fact that one script can provide both networked and non-networked behaviour.

If you are upgrading from version 1.0.0 and use the networking code in your game then unfortunately the updated version will not be compatible and you may receive missing script warnings. It is recommended that you upgrade to this newer version to receive future bug fixes and updates but version 1.0.x can still be used if upgrading is not an option.

Version 1.1.0 also slightly modifies the API by replacing the method used to inform spawn managers that a spawnable object is about to be destroyed. 'EnemyManager.informEnemyDeath' now becomes 'SpawnableManager.informSpawnableDestroyed' to provide a general case.

1.0.2

Minor bug fixes and moved all demo scripts out of the global namespace to avoid conflicts (UltimateSpawner.Demo).

Getting Started

This section will quickly take you through the steps of creating a basic spawn system using Ultimate Spawner. Note that only basic functionality is demonstrated I this section and that more advanced features are covered at a later stage.

Step 1 – The spawn manager

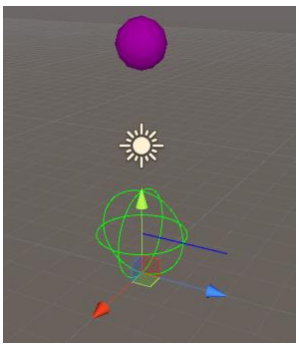
The first step to setting up our basic spawning system is to create the spawn manager. The spawn manager is responsible for handling all spawn requests and selects spawn points based on its configuration and availability. For now we don't need to know much about the spawn manager but if you want to know more take a look at the Spawn Manager section.

For ease of use there is a spawn manager prefab supplied as part of the package. Go to the prefabs folder of Ultimate Spawner and then drag the 'SpawnManager' prefab into the scene. You should now see a purple node which represents the manager:

Step 2 – Spawn points

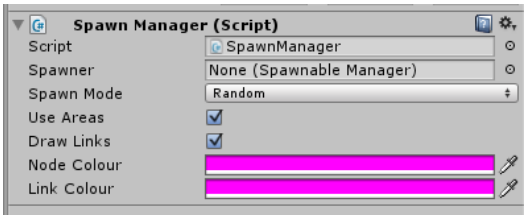
Once we have our spawn manager in the scene then we are ready to add some spawn points. Spawn points are markers that you place in you game environment to represent the locations that enemies may spawn at. For more information on spawn points take a look at the Spawn Point section.

As with the spawn manager, Ultimate Spawner provides a spawn point prefab for ease of use. Simply drag the 'Spawner' prefab into the scene and you should now see a spawn point displayed in the scene represented by a bounding sphere and direction indicator.

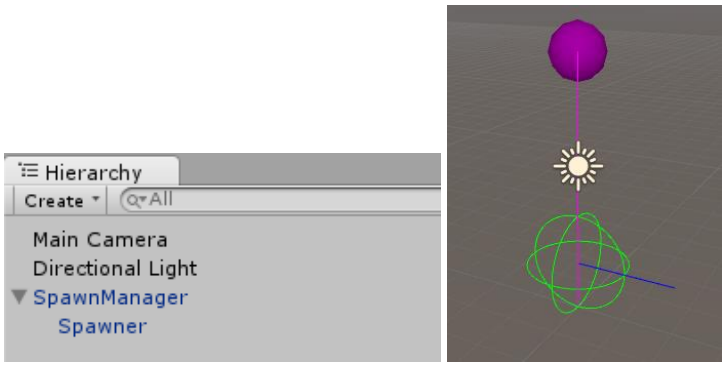


Step 3 – Link spawn points

Now you should have both a spawn manager and a spawn point in the scene. The next thing we need to do is link the spawn point which informs the spawn manager that it is responsible for this specific spawn point. In order to achieve this you will first need to uncheck the 'useAreas' property on the spawn manager component. Spawn areas are used to achieve more complex spawning systems and are covered at a later stage.



Once you have disabled the areas settings then you will now be able to link the spawn point with the spawn manager. This is achieved by simply making sure that the spawn point is a child object of the spawn manager. You should see a purple link line from the spawn manager to the spawn point if you have successfully linked the spawn point.



Part 1 complete

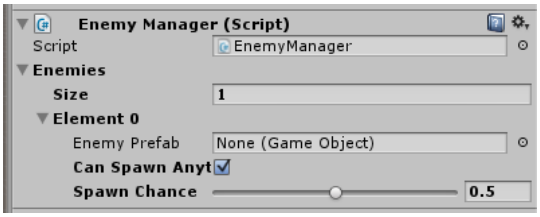
At this point we have now successfully setup a spawning system that has all the required components. You can access the spawn manager component though scripts and spawn objects manually at any time. Continue to the next part to implement a basic wave system with automated spawning of enemies.

Step 4 – The enemy manager

Before we can begin adding waves to our game we first need to think about how the enemies will spawn. Ultimate Spawner provides an enemy manager component which is used to store a collection of enemies which may be spawned based on a chance value. For more information about the enemy manager take a look at the Enemy Manager section.

In order to create the enemy manager we need to go back to the prefabs folder and drag the 'EnemyManager' prefab into the scene. You will note that the component is empty by default.

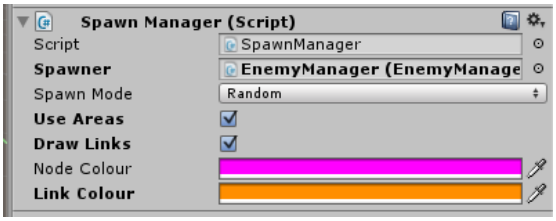
We will now want to add an enemy to this enemy manager so that an object will spawn when we implement our wave system. In the inspector, change the enemy's array size to 1 and you should now see the following display:



The only value we will modify for this new enemy is the enemy prefab field. We need to assign a prefab object to this field which represents the enemy. For testing purposes this could be a simple cube object.

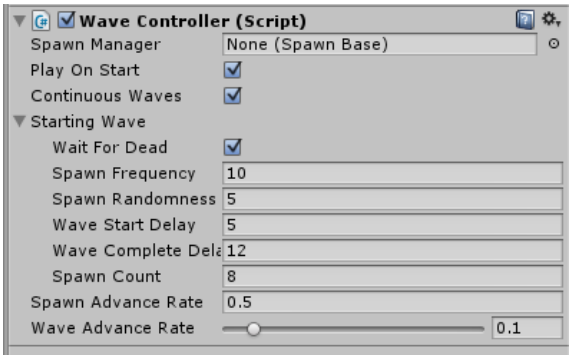
Step 5 – Link the enemy manager

Once we have setup our enemy manager and are happy with its configuration we can link it to the spawn manager. This is a quick and easy step and provides the spawn manager with a collection of enemies that it can create at its spawn locations. In order to link the enemy manager we will first need to select our spawn manager in the scene. Under the spawn manager component you should see a field called 'Spawner'. This is where we want to drop our enemy manager component.



Step 6 – The wave controller

The final step in our short example is to create a wave controller which will be responsible for generating an infinite number of increasingly difficult Ultimate Spawner. In order to create the wave controller we will drag the 'WaveController' prefab from the prefabs folder into the scene. You should see the following inspector display:



The wave manager is configured by default to generate an endless number of Ultimate Spawner. The only thing we need to do is to assign the spawn manager and the event manager that this wave system will make use of.

Simply drag the appropriate manager object from the hierarchy into the correct slots on the wave manager component and you should end up with the following setup:

Part 2 complete

We have now successfully setup a basic spawning and wave system using Ultimate Spawner. If you enter play mode you will note that after a few seconds an enemy will be spawned at the spawn point however that will be all that happens. This is because the enemy is now occupying the spawn point and no more enemies are able to spawn.

Improvements to add:

You will now have a very basic but functional spawning system in place but there are a few things you can do to improve upon it:

- Add a few more spawn points so that more than one enemy is able to spawn
- Add movement to the enemies so that they move away from the spawn point once they have spawned. This will allow the spawn point to become available and potentially spawn another enemy.
- Implement a way of killing the enemies so that the waves can advance
- Play around with the settings and see what happens ☺

Enable Networking

Ultimate Spawner has built in support for networking using Unity's uNet which is disabled by default for compatibility reasons. If you require networking support then you will need to add a scripting define symbol to your project.

Add 'ULTIMATE_SPAWNER_NETWORKED' to the scripting define symbols.

For information on adding define symbols take a look at the following 'Scripting define symbols' section.

Once the symbol has been defined you will need to make use of the prefabs located in the 'Ultimate Spawner/Prefabs/uNet' folder.

Enable Pooling

Ultimate Spawner also has support for pooling which relies on an additional asset called 'Ultimate Pooling'. In order to make use of the pooling features you will first need to purchase this asset and ensure that it has been added to your project. You can find Ultimate Pooling on the asset store here: https://www.assetstore.unity3d.com/en/#!/content/64950

Once you have added Ultimate Pooling to your project you can add the following scripting define symbol to your project.

Add 'ULTIMATE_SPAWNER_POOLED' to the scripting define symbols.

For information on adding define symbols take a look at the following 'Scripting define symbols' section.

Scripting define symbols

Scripting define symbols are used to allow conditional compiling which allows code to be enabled or disabled via the project settings. Ultimate Spawner uses define symbols for all features that are disabled by default due to compatibility issues. Scripting defines symbols are trivial to add and can be done by following these steps:

1. Go to 'Edit -> Project Settings -> Player'
2. Locate the section labelled 'Other Settings' for your target platform.
3. Locate the field labelled 'Scripting Define Symbols'
4. Add the required scripting define symbol to the field, For example: ULTIMATE_SPAWNER_NETWORKED. Take care of correct spelling and case.
5. Close the settings and allow Unity to recompile all scripts.
6. The scripting define symbol should now be activated.

Note: Multiple scripting define symbols may be entered into the filed by separating values with a semicolon ';' and no white space.

Concepts

Ultimate Spawner can be used as a general purpose spawner for any object such as collectable items as well as an enemy spawner with added functionality. The spawning code is completely independent of any enemy related spawning code such as wave spawning and means that the system is far more flexible in what it can achieve.

Ultimate Spawner makes use of spawn points to mark the location where an object can spawn. These spawn points represent a position and rotation in 3D space (Although 2D games can use this system) and are only responsible for deciding whether this point is free (No object is overlapping the spawn point). Spawn points are the lowest contact point in Ultimate Spawner and the API allows you to directly request a specific spawn point to spawn an object.

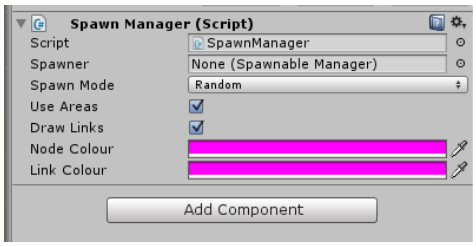
As well as spawn points, Ultimate Spawner uses a couple of other higher level concepts to add advanced spawning behaviour. The first of these is the spawn manager which is a container for a number of spawn points. The manager is responsible for handling spawn requests and depending on its configuration will select an appropriate spawn point to delegate the request to. Another higher level concept is a spawn area which is used to ensure that enemies are spawned near to the player to keep the level of challenge consistent. All of these concepts are explained in detail in a later section.

Spawn Manager

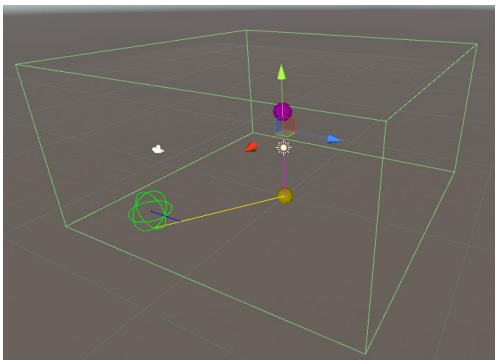
The spawn manager is the most important component of Ultimate Spawner as it is responsible for managing all other spawning component to provide a functional system. The main responsibility for the span manager is to delegate spawn requests to available spawn points based on the spawn mode selected.

Spawn Manager Component

Ultimate Spawner includes a Spawn Manager component which is responsible for managing a collection of spawn areas or spawn points depending on configuration. In order to create the component you will need to create an empty game object and then attach the 'SpawnManager' script to the object. Alternatively you can make use of the included 'SpawnManager' prefab by simply dragging it into the scene. Once you have created the spawn manager you should see the following inspector display:



A spawn manager requires one or more spawn areas in order to provide any spawning functionality. In order to add a spawn area to the manager you will simply need to make sure that any spawn area you want to associate with the manager is a child of the manager's game object. If the area is setup correctly then you should see a link between the area and the spawn manager as shown below:



In the above image the spawn manager is represented by the purple node.

Spawn Manager Properties

- **Spawner:** A reference to a 'SpawnableManager' component that should handle spawning enemies, typically an 'EnemyManager'.
- **Spawn Mode:** Which mode to use to select a child spawn point. SpawnMode.Random will randomly select the spawn point from all active spawn points. SpawnMode.Sequential will select the next spawn point in the linear sequence.
- **Use Areas:** When enabled the spawn manager will make use of all child spawn areas. When disabled, the manager will make use of all child spawn points regardless of the child areas.
- **Draw Links:** When enabled, all relationship links between spawn nodes will be drawn in the scene view

Spawn Manager Behaviour

- A spawn manager can only spawn if one or more areas have an available spawn point
- A spawn manager can only spawn if one or more areas have been activated by the trigger object
- Spawn manager will not make use of any child areas if the useAreas field is true. Instead it will access all spawn point for the child areas.

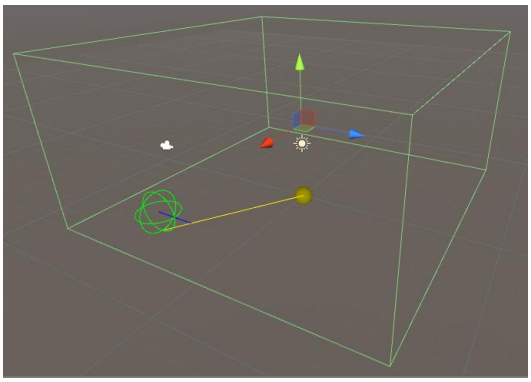
Spawn Area

A spawn area represents a playable area in the game environment and is responsible for managing one or more spawn points. An area is responsible for activating assigned spawn points when a specific game object lies within its bounds. The main purpose of these spawn areas is to ensure that enemies never spawn too far from the player which becomes important in games that utilize large or complex environments. Take a look at the demo scenes for an example of spawn areas in use.

Spawn Area Component

Ultimate Spawner includes a spawn area component which can be used to represent a playable area in the game environment and acts as a trigger volume for its associated child spawn points. In order to create the component you can simply add the 'SpawnArea' script to an empty game object. You will then need to make sure that a suitable trigger collider is attached. Alternatively you can use the build in prefab for a spawn area which utilizes a box collider for its area trigger. Simply drag the 'SpawnArea' prefab into the scene. Once you have created a spawn area you should see the following inspector display:

A spawn area requires on or more spawn points in order to provide any spawning functionality. In order to add a spawn point to the spawn area you will simply need to make sure that any spawn point you want to associate with the area is a child of the area's game object. If the spawn is setup correctly then you should see a link between the area and the spawn point as shown below:



Note: Spawn areas are semi-standalone components and can be used independent of the Spawn Manager if required. A spawn area does however require at least one spawn point.

Spawn Area Properties

- **Spawn Settings:** What settings should the spawner use to generate spawnable items
- **Spawner: (**Visible depending on configuration) A reference to a 'SpawnableManager' component that should handle spawning enemies, typically an 'EnemyManager'.
- **Spawn Mode:** Which mode to use to select a child spawn point. SpawnMode.Random will randomly select the spawn point from all active spawn points. SpawnMode.Sequential will select the next spawn point in the linear sequence.
- **Always Active:** When enabled, the area will be able to spawn regardless of whether its trigger volume is active or not. This is useful for a fall back area that encapsulates the entire game environment and will continue spawning even is the trigger object goes out of bounds.
- **Area Tag:** The tag to look for when performing volume checks. The default tag is player since spawn areas will typically be activated and deactivated by the player object.
- **Collision Layer:** The layer that the spawn point will check for collision on.

Spawn Area Behaviour

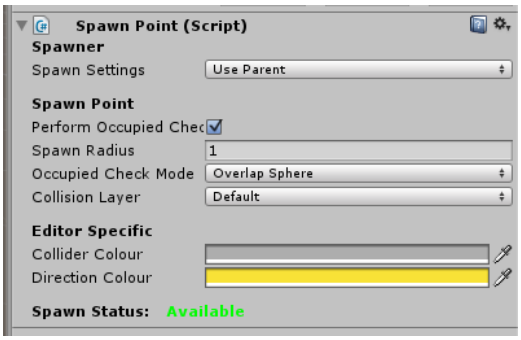
- A spawn area can only spawn an object if one or more of its spawn points are unoccupied.
- A spawn area will refuse to spawn an object if the trigger object is not inside its trigger volume (Depending on configuration)
- A spawn area can be activated by multiple objects (multiplayer situations)

Spawn Point

A spawn point represents a position and rotation in 3D space and is used to mark the position where an object may be placed during the game. There can be any number of spawn points in the environments and they can be used either via a manager or individually.

Spawn Point Component

Ultimate Spawner includes a spawn point component which can be used to represent a spawn point in a 3D environment. In order to create the component you can simply add the 'SpawnPoint' script to an empty game object. If you prefer you can also use the built in prefab for a spawn point. Simply drag the 'Spawner' prefab into the scene. One you have created a spawn point you should see the following inspector display:



Note: Spawn Points are standalone components and can be used independent of Spawn Areas or Spawn Managers if required.

Spawn Point Properties

- **Spawn Settings:** What settings should the spawner use to generate spawnable items
- **Spawner: (**Visible depending on configuration) A reference to a 'SpawnableManager' component that should handle spawning enemies, typically an 'EnemyManager'.
- **Perform Occupied Check:** When enabled, the spawn point will check if it is occupied by another object prior to spawning.
- **Spawn Radius:** The radius of the spawn point, used in the occupied check to overlap a sphere for collision detection.
- **Occupied Check Mode:** When set to 'OverlapSphere' the spawn point will using the physics class to perform a sphere overlap collision check with the specified radius. When set to 'TriggerEnterExit' the spawn point will make use of the attached collider to detect when object enter or exit the spawn point. If there is not a trigger collider attached to the spawn point and the mode is set to 'TriggerEnterExit' then an error message will display in the inspector.
- **Collision Layer:** The layer that the spawn point will check for collision on.

Note: A spawn point contains status information at the bottom of its component display to indicate whether it is able to spawn at any given point or if it is currently occupied.

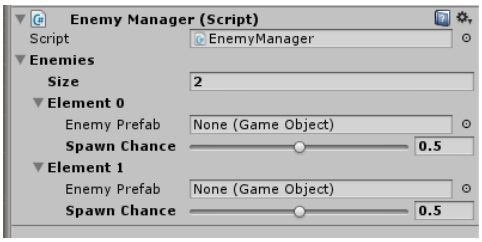
Spawning Behaviour

- A spawn point can only spawn an object if it is currently unoccupied. (Depending on individual configuration)
- A spawn point will trigger an ‘onSpawned’ event when it has successfully spawned an object
- A spawn point will fail to spawn if its collision sphere is occupied by a collider object that shares the layer.

Enemy Manager

The enemy manager is where you will specify which enemies can be spawned into the game and how likely they are to spawn. You can have as many enemy managers in the scene as you want each with different enemy collections and you are then able to assign them to specific spawn managers if required. Alternatively you can also assign these enemy managers to spawn areas or spawn points to specifically choose which points may spawn a specific type of enemy. This is achieved through the settings for the individual spawn node where you are able to assign a reference to an enemy manager.

The enemy manager is essentially an array of enemy prefab references, each with their own spawn chance value. Once you have an enemy manager in the scene you should see the following inspector display:



Enemy Manager Properties

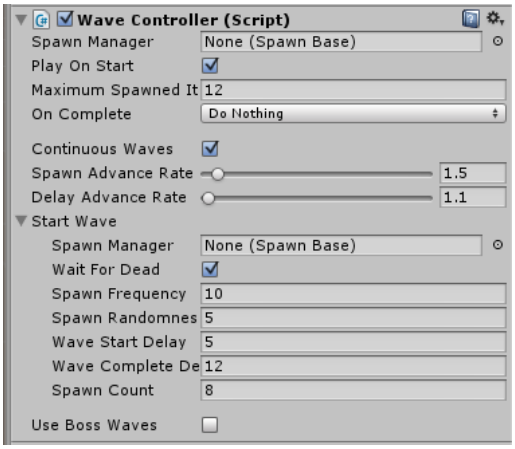
- **Enemies:** The enemies field is an array of type 'EnemyInfo' which is a serializable class that allows each enemy to be assigned a spawn chance value.
- **(EnemyInfo) Enemy Prefab:** (Per element): This slot is present for each enemy in the array and is where the prefab for the enemy should be assigned to.
- **(EnemyInfo) Spawn Chance:** (Per element): This value determines the likelihood of the enemy being spawned at any time. It is represented by a normalized float value between 1 – 0 where higher values improve the chance of spawning. If the spawn chance is set to 0 then the enemy will not spawn unless it is the only enemy present in the array. It's worth noting that the spawn chance is dependent upon the spawn chance of all other enemies in the array. With this in mind, you are able to improve the spawn chance of a specific enemy by reducing the spawn chances for every other enemy.

The 'EnemyManager' class inherits from the 'SpawnableManager' class and can be inherited to provide modified behaviour.

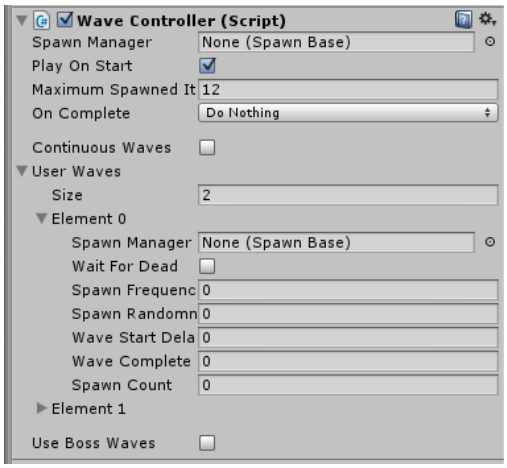
Wave Controller

The wave controller component allows you to setup a basic enemy wave system where enemies will be spawned based on the predefined settings. The wave controller is able to operate in two different modes which can provide automated or predetermined wave behaviour. The first mode allows the user to specify each wave individually allowing great control over the progression of difficulty. The secondary mode takes a predefined starting wave and is then able to produce and endless amount of automatically generated, progressively difficult waves allowing for infinite gameplay.

The following inspector image shows the first mode using automatically generated waves of increasing difficulty:



The following inspector image shows the second mode using manually assigned waves:



Wave Controller Properties

- **Spawn Manager:** A reference to an existing 'SpawnBase' component (typically a Spawn Manager) that this controller should use for spawning.
- **Play on Start:** When enabled, the wave will start as soon as the scene is loaded. Alternatively you can activate the waves from a script.
- **Maximum Spawned Items:** The maximum number of items that can exist at any time. Useful to cap the maximum amount of enemies that can be on screen at any time, especially where performance is a factor.

- **On Complete:** What action should be taken when the controller has finished its pre-assigned wave sequence. DoNothing: This option will simply trigger the 'onExitLastWave' event and end spawning. RepeatFromStart: This option will reset the wave controller and begin spawning again from wave 1. RepeatLast: This option will partially reset and then start the last wave in the array from the beginning. Unless this setting is changed at runtime, the last wave will repeat endlessly.
- **Continuous Waves:** Specifies which wave mode to use, manually assigned waves or automatically generated waves. As of version 1.3.4 this settings has been replaced with a drop down menu called 'Wave Mode' which allows for an extra option called mixed mode waves.
- **Spawn Advance Rate:** (Visible depending on settings): This value determines how much the enemy spawning for the auto generated wave should advance. By default this is set to 1.5 which means that 0.5 (or half) of the enemies from the first wave will be added onto the next wave. For example, if the first wave contained 8 enemies, the next wave will contain 12.
- **Delay Advance Rate:** (Visible depending on settings): This value determine how much the wave timings will change for the next wave. By default this is set to 1.1 meaning that each wave will start and end 1.10^{th} of a second quicker than the previous wave. This value will also modify the spawn frequency to increase the spawn rate at later waves.
- **Start Wave:** (Visible depending on settings): The starting wave information used to generate the subsequent waves for a continuous waves setup.
- **User Waves:** (Visible depending on settings): The waves field is an array of type 'Wave' which is a serializable class that allows configuration of a specific wave.
- **(Wave) Spawn Manager:** The spawner to use for this wave. Default is 'None' in which case the wave controllers spawner is used. (Revert to parent settings)
- **(Wave) Wait for Dead:** When enabled the next wave will not begin until all existing enemies for the current wave have been destroyed.
- **(Wave) Spawn Frequency:** The amount of time (in seconds) to wait before another enemy is able to spawn
- **(Wave) Spawn Randomness:** The amount of time (in seconds) that can be used to randomly delay enemy spawning. The greater the value, the more random the enemy spawning appears
- **(Wave) Wave Start Delay:** The amount of time (in seconds) to wait before the wave can start spawning enemies
- **(Wave) Wave Complete Delay:** The amount of time (in seconds) to wait before the end of the wave is triggered. Allows a cool off period where no spawning will take place
- **(Wave) Spawn Count:** The number of enemies to spawn during this wave

The 'WaveController class inherits from the 'ControllerBase' class and can be inherited to provide modified behaviour.

Wave Modes

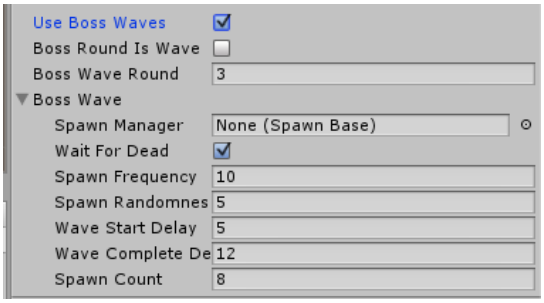
The wave controller is able to spawn enemies using one of three modes:

- **Preset Mode:** This mode allows all waves to be manually specified for ultimate customization.

- **Continuous Mode:** This mode allows the controller to automatically generate the next wave sequence using a value that specifies the advance rate for the wave. This mode has the added benefit of being able to infinitely spawn waves.
- **Mixed Mode:** Mixed mode was introduced in version 1.3.4 and combines the previous modes to gain the advantages of both. At the start of the game, the controller will use the preset waves allowing the developer to precisely configure the progression of the waves. When all preset waves have been completed then the controller will take over the generation of the waves to create an unlimited number of rounds.

Boss Waves

As of version 1.3.0 you are now able to easily add boss waves to your game. The wave controller includes a new group of settings that allow you to setup boss waves that occur when a specified wave is reached. The following image show the wave controller inspector including the settings for boss waves.



In order to activate boss waves you need to ensure that 'Use Boss Waves' is checked which should reveal a new list of settings that can be used to modify the boss wave behaviour.

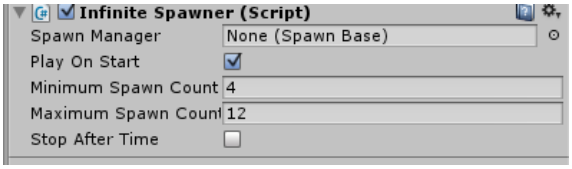
- **Boss Round Is Wave:** When enabled, the boss wave is treated like any other wave meaning that the wave counter is increased and wave start and end events are raised. If it is disabled then the boss round will simply appear at the end of a previous wave and the wave counter will not be modified.
- **Boss Wave Round:** The wave when the boss round will activate. For example, if this value is set to '3' then the boss wave will activate at the end of round '3'. Boss rounds are also repeated meaning that the boss wave will continue to activate after waves '6'' 9' and so on.
- **Boss Wave:** These settings are much the same as standard waves but only apply to the boss wave. There is also the ability to reference a different spawn manager if other spawn locations or enemies should be spawned during boss rounds.

For more information about the setup of boss rounds take a look at the 'BossExample' scene located in the demo folder.

Infinite Controller

The infinite controller is a component that allows you to setup constant spawning of enemies either for a specified period of time or indefinitely. This behaviour is commonly seen in games where the player is required to kill an endless stream of enemies until some event occurs, at which point they may progress.

The following inspector display shows the settings for the infinite controller:



Infinite Controller Properties

- **Spawn Manager:** A reference to an existing 'SpawnBase' component (typically a Spawn Manager) that this controller should use for spawning.
- **Play on Start:** When enabled, the controller will begin spawning enemies as soon as the scene is loaded.
- **Minimum Spawn Count:** The minimum number of enemies to exist at any time. Only achievable where sufficient spawn points are available providing that no other spawning rules are broken.
- **Maximum Spawn Count:** The maximum number of enemies that can exist at any time.
- **Stop after time:** When enabled, an additional field will be displayed where you can specify how long spawning should continue for.
- **Stop After:** The amount of time (in seconds) to spawn enemies for. When this time has elapsed, spawning will stop automatically.

The 'InfiniteController' class inherits from the 'ControllerBase' class and can be inherited to provide modified behaviour.

Report a Bug

At Trivial Interactive we test our assets thoroughly to ensure that they are fit for purpose and ready for use in games but it is often inevitable that a bug may sneak into a release version and only expose its self under a strict set of conditions.

If you feel you have exposed a bug within the asset and want to get it fixed then please let us know and we will do our best to resolve it. We would ask that you describe the scenario in which the bug occurs along with instructions on how to reproduce the bug so that we have the best possible chance of resolving the issue and releasing a patch update.

http://trivialinteractive.co.uk/bug-report/

Request a feature

If you feel that Ultimate Spawner should contain a feature that is not currently incorporated then you can request to have it added into the next release. If there is enough demand for a specific feature then we will do our best to add it into a future version. Please note, there are some features that are not possible to implement due some limitations of Unity but a workaround may be possible.

http://trivialinteractive.co.uk/feature-request/

Contact Us

Feel free to contact us if you are having trouble with the asset and need assistance. Contact can either be made by the contact options on the asset store or buy the link below.

Please attempt to describe the problem as best you can so we can fully understand the issue you are facing and help you come to a resolution. Help us to help you :-)

http://trivialinteractive.co.uk/contact-us/