User Guide

Ultimate Spawner 2.0 Waves Add-On

Seamlessly Adds wave spawning tools and capabilities

Trivial Interactive

Version 2.0.x

IMPORTANT: THIS ASSET IS AN ADD ON FOR ULTIMATE SPAWNER 2.0 AND WILL NOT WORK AS A STANDALONE ASSET. PLEASE MAKE SURE THAT YOU PURCHASE ULTIMATE SPAWNER 2.0 FROM THE UNITY ASSET STORE PRIOR TO PURCHAING THIS ASSET.

Features

- Seamlessly integrates with the base asset.
- Quickly and easily create or edit wave graphs to get the requried spawning behaviour.
- Highly versatile node editor for creating and editing wave spawner logic graphs.
- Includes many built in wave nodes allowing for an unlimited amount of spawning variety.
- Ability to create custom wave nodes which can then be added to the wave graph.
- Suitable for 2D and 3D games.
- Fully commented C# source code

Contents

Project Management

Installing Add Ons

To install add on packages you must first own Ultimate Spawner 2.0 which can be purchased via the asset store. Open the Unity project where you want to install the add on and install the main Ultimate Spawner 2.0 asset if it is not already added to the project. Take a look at the separate documentation for installation help. Once done you can now import add on packages such as Ultimate Spawner 2.0 Waves Add On as you would normally. The waves add on will import into a separate root folder called 'UltimateSpawner2.0-WavesAddOn' and you will be ready to use the asset.

Quick Start

The following section will take your though the basics in order to get Ultimate Spawner up and running as quick as possible. Please note that many of the features will not be discussed in this section but will be covered later in this document.

1. Create a spawnable items asset. With the project window focused, right click and select 'Create -> Ultimate Spawner -> Spawnable Items' in a suitable folder and give the asset a name. A spawnable items asset is required to provide a collection of items that can be spawned.

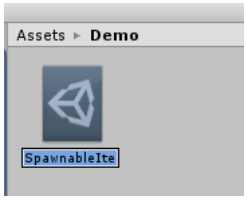


Figure 1

2. Ensure that the spawnable items asset is selected and focus the inspector window. Click the 'Add' button to add a new spawnable item entry. Once you have added a new entry you will need to drag a prefab into the object field named 'Prefab' and we will leave the other options at their default values. This prefab should be something you want to spawn into your scene like an enemy or you could use one of the example prefabs included with Ultimate Spawner for testing purposes.

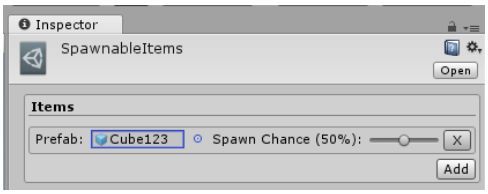


Figure 2

Note that you can also click the small icon to the right of the field to show the asset browser window and an unlimited number of spawnable items may be added to the asset.

3. We will need to turn our attention to the scene to create a spawner which will provide a specific location in 3D space where these items/enemies can be spawned. To do this go to the menu 'GameObject -> Ultimate Spawner -> Spawn Point' to create a simple spawn point game object in the scene. You can move and rotate this spawn point as required.

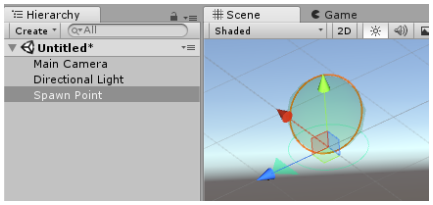


Figure 3

4. Ensure that the newly created spawn point is selected and focus the inspector window. Locate the field on the ‘SpawnPoint’ script component named ‘Spawn Items’ and drag the previously created spawnable items asset into this slot.

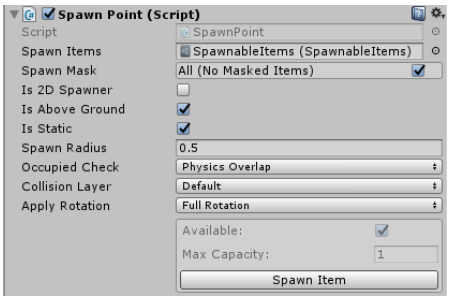


Figure 4

Wave Graphs

Ultimate Spawner 2.0 Waves Add On uses a dedicated wave graph editor window where you can build wave spawner sequences by connecting nodes as shown in figure 5.

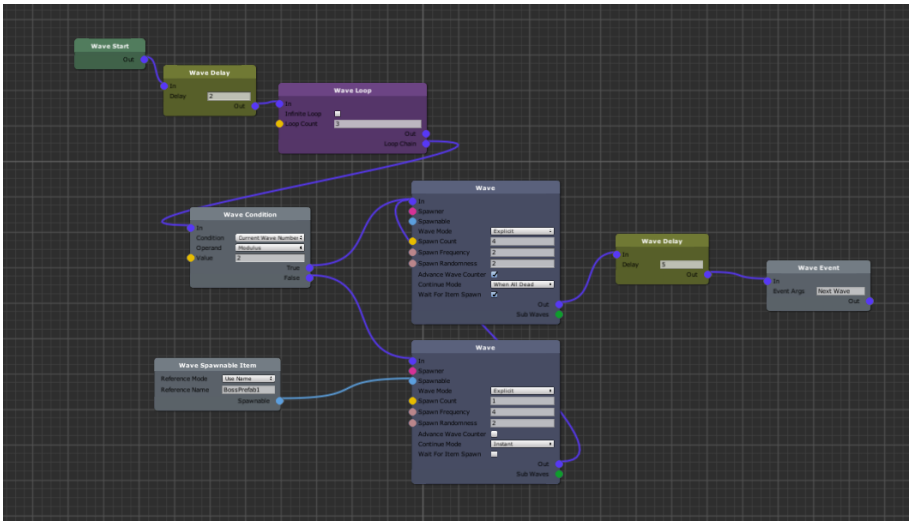


Figure 5

In order to create a wave sequence using the wave graph editor window you will first need to create a Wave Graph asset where all of the data will be stored. You can create a wave graph asset from the project window by right clicking and selecting 'Create -> Ultimate Spawner -> Wave Graph'. A new asset will be created and you will be prompted to give it a name.

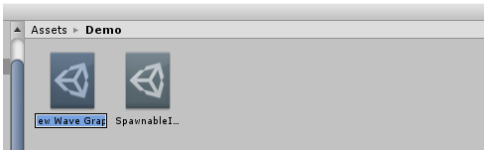


Figure 6

Once you have a wave graph asset you can edit it in the wave graph editor window by simply double clicking on the asset or right clicking and selecting 'Open'. You will see a single node which is added by default called the 'Wave Start' node. As you would expect, this node is the entry point and the wave sequence will flow though all connected nodes when the game is run. If nothing is connected to this star node then the wave sequence will not do anything.

Using the Wave Graph Editor

This following section will cover all the mouse and keyboard controls needed to use the wave graph editor although if you know your way around the Unity editor it should be fairly intuitive to use.

Grid Navigation

- **Zooming**: Use the mouse scroll wheel to zoom the viewport in or out allowing you to see more or less of the wave graph.
- **Panning**: Hold down the middle mouse button while dragging the mouse to pan the viewport around the wave graph.

Working with Nodes

- **Add a Node:** To add a new node to the wave graph simply right click in an empty area and use the context menu entry ‘Create’ to select which type of node to add. A new node will be added to the wave graph at the approximate position of the mouse cursor.
- **Remove a Node:** Right click on an existing nodes header area and select ‘Remove’ from the context menu. The selected node will be deleted from the wave graph.
- **Duplicate a Node:** Right click on an existing nodes header area and select ‘Duplicate’ from the context menu. The specified node will be duplicated and all values will be copied.
- **Bring To Front:** It is possible for multiple nodes to overlap each other so it may be useful to draw some nodes on top of others. Right click on an existing nodes header area and select ‘Move To Top’ from the context menu. The selected node will now be drawn on top of other nodes.
- **Move a Node:** Nodes can be moved around the wave graph by dragging them. Simply click and hold the left mouse button in a nodes header area and then drag using the mouse to reposition the node. Release the left mouse button to drop the node at the current position. Note that nodes will snap to the grid for better organisation.

Working with Node Connections

Nodes will typically have a number of ports attached to them which are denoted by a colours circle and can either be an input positioned on the left of the node, or an output positioned on the right of the node. Connections can be made to these ports which can be used to link different nodes together to create wave sequences which flow from ‘Out’ ports to ‘In’ ports. Connections have the following rules:

1. Connections can only be made between ports of the same colour. For example: Purple ‘Out’ ports can only connect to purple ‘In’ ports.
2. Connections can only be made from output ports on the right of a node to input ports on the left of a node.
3. An out port can only have a single connection.
4. An input port can have multiple connections.
5. A node with no input connection will have no effect when the game is run.

- **Create Connection:** Move the cursor over the output port you want to connect. Click and hold the left mouse button and drag the mouse over to the input port you want to connect to. You will see that the connection wire will follow the mouse cursor as you go. With the mouse cursor hovering over an input port release the left mouse button and you will see that a new connection is made. If the wire disappears upon connection then you may have made an invalid connection. Consult the connection rules above.
- **Delete Connection:** Connections can only be removed from the input side of a node. Move the mouse cursor over the input port of the connected wire that you want to remove. Click and hold the left mouse button to pickup the wire. Move the mouse cursor to an empty space in the grid and release the left mouse button. You should see that the connection is deleted.

Node Types

Ultimate Spawner 2.0 Waves Add on includes a number of pre-set nodes that can be used to create a variety of wave based spawning sequences. Each node will do a specific task such as waiting for a specified amount of time before continuing and by connecting many varying nodes you can create complex wave spawning systems. The following section will cover all pre-set nodes and what their purpose is along with any modifiable properties which can be used to change how the node behaves.

Wave Start Node

This is probably the most important node because it is the entry point for the wave graph and all subsequent nodes must be connected or indirectly connected to the start node in order to be executed. The wave node is also treated specially in that there can only ever be 1 start node and it will be automatically created the first time you create a wave graph asset.

Figure 7

Connection Ports

- **Out (Output Port):** The out port is the only port on the start node and it will execute the connected node when the wave graph is evaluated during the game. If no other node is connected to this port then the wave graph will do nothing.

Wave Condition Node

The wave condition node is a useful node that will evaluate a preset condition and evaluate either the 'True' or 'False' branch depending upon the conditional check. This can be useful when used within loops to modify flow control as certain conditions are met.

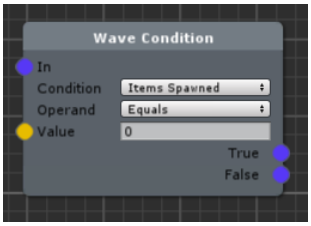


Figure 8

Connection Ports

- **In (Input Port):** The standard input port that should be connected to a previous node.
- **Value (Optional Input Port):** The value port can have an optional input attached to it in the form of a 'Wave Randomizer' node or if there is no connection then the preset value will be used instead.
- **True (Output Port):** The true port can connect to another nodes In port and will be evaluated when the configured condition results in a true value.
- **False (Output Port):** The false port can connect to another nodes In port and will be evaluated when the configured condition results in a false value.

Node Properties

- **Condition:** The conditional value to check against. Possible values are:
    - **Items Spawned Total:** The total number of items spawned by the wave controller
    - **Items Spawned:** The number of items spawned by the wave controller during the current wave
    - **Items Alive:** The total number of alive items that are remaining. Alive items are simply items that have not been destroyed or removed from the scene.
    - **Items Destroyed:** The number of items destroyed during the current wave that were initially spawned by the wave controller.
    - **Items Destroyed Total:** The total number of items destroyed which were initially spawned by the wave controller.
    - **Current Wave Number:** The wave number f the current wave.
- **Operand:** Determines the type of operation that is used during the conditional check. Possible values are:
    - **Equals:** Check whether the two values are equal.
    - **Greater Than:** Check whether the condition value is greater than the specified value
    - **Greater Than Or Equal:** Check whether the condition value is greater than or equal to the specified value.
    - **Less Than:** Check whether the condition value is less that the specified value.
    - **Less Than Or Equal:** Check whether the condition value is less than or equal to the specified value.
    - **Modulus:** Check whether the modulus of the two values equal to zero.
    - **Inverse Modulus:** Check whether the modulus of the two values is not equal to zero.

- **Value (Optional Input):** The value property is use as the second input to the conditional check. You can set this value manually by entering a value or you can connect a ‘Wave Randomizer’ node to randomly generate a value.

Wave Delay Node

The wave delay node is a useful node whose purpose is to wait for a fixed amount of time before continuing to the next node. The time specified is in seconds.

Figure 9

Connection Ports

- **In (Input Port):** The in port must be connected to an out port of another node in order for the delay node to be evaluated.
- **Out (Output Port):** The out port should connect to another nodes In port which will cause that node to be evaluated after the specified amount of time has passed. If the out node is not connected then the delay node is essentially obsolete as the sequence will not continue.

Node Properties

- **Delay:** The amount of time in seconds that the node should wait before continuing to the next connected node. Decimal numbers can be specified to denote fractions of a second.

Wave Event Node

A wave event node is a useful way to trigger gameplay events at certain points in the wave spawning sequence. The event node will cause the `'OnWaveCustomEvent'` event of the associated wave controller to be invoked when the node is evaluated. This event is implemented as a Unity event which accepts a string argument.

Figure 10

Connection Ports

- **In (Input Port):** The default input port used to connect to a previous node in the sequence.
- **Out (Output Port):** The default output port used to connect to the next node in the sequence.

Node Properties

- **Event Args:** The string argument that will be passed when the 'OnWaveCustomEvent' event is invoked. This argument can be useful for determining which wave event node triggered the call if there are multiple event nodes in use.

Wave Loop Node

A wave loop node is used to loop through the same wave graph sequence chain multiple times making it quick and easy to build long or infinite spawning sequences without needing to manually create every node. When combined with multiplier waves, it is possible for the difficulty of the waves to automatically increment in difficulty based upon the previous wave. See the 'Wave Node' section for more information.

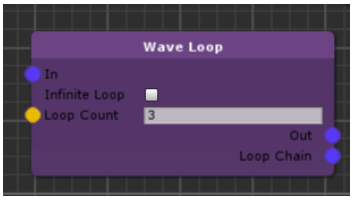


Figure 11

Connection Ports

- **In (Input Port):** The default input port used to connect to a previous node in the sequence.
- **Loop Count (Optional Input Port):** An optional integer input port which can be connected to a Wave Randomizer node in order to randomly generate the loop count value. When this port is not connected the manually specified value is used instead.
- **Out (Output Port):** The default output port used to connect to the next node in the sequence. The connected node will only be evaluated once the 'Loop Chain' node has been fully evaluated the amount of times specified by the 'Loop Count' value.
- **Loop Chain (Output Port):** A special output node used to create the looping branch of the spawning sequence. This loop chain branch will be evaluated fully every iteration of the loop before continuing evaluation of the 'Out' branch.

Node Properties

- **Infinite Loop:** When true the loop chain branch will be evaluated infinitely meaning that the 'Out' branch will never be reached.
- **Loop Count (Optional Input):** The amount of times the loop node should evaluate the 'Loop Chain' branch. You can optionally connect a 'Wave Randomizer' node to the input port of this value to randomly generate the loop count value.

Wave Node

The wave node is the most important node type as it allows you to setup a specific wave structure that can be explicit or auto advanced based on the previous wave settings. Each wave node will usually represent a single wave in the wave spawner system although this is not always the case depending upon the configuration of the nodes settings. Each wave can specify the number of spawnable items which should be created, how other they should spawn and much more. You can also associate any number of 'Sub Waves' with a wave node which can be used to spawn different items within the same wave which allows for more complex spawning structures.

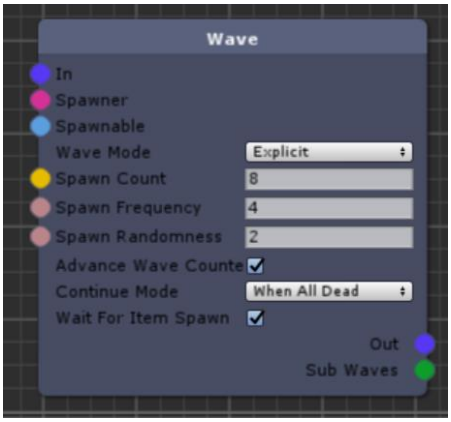


Figure 12

Connection Ports

- **In (Input Port):** The default input port used to connect to a previous node in the sequence.
- **Spawner (Optional Input Port):** An optional connection to a 'Spawner Reference' node which can be used to specify which spawner in the associated spawn system hierarchy will be used to handle the spawning requests for the duration of the wave.
- **Spawnable (Optional Input Port):** An optional connection to a 'Spawnable Reference' node which can be used to specify which spawnable item will be spawned during the duration of the wave. Note that this value cannot override the associated spawner systems spawning masks assigned on a per spawner basis so it may not be possible to spawn the specified item depending upon the spawner used to handle the spawn request.
- **Spawn Count (Option Input Port):** An optional integer input port which can be connected to a Wave Randomizer node in order to randomly generate the spawn count value. The value represents the number of items that will be spawned during the wave When this port is not connected the manually specified value is used instead.
- **Spawn Frequency (Optional Input Port):** An optional decimal input port which can be connected to a Wave Randomized node in order to randomly generate the spawn frequency value. This value denotes an amount of time in seconds where '2.5' is equal to two and a half seconds. The value is then used to calculate the minimum amount of time which must pass before a new item can be spawned by the controller. When this port is not connected the manually specified value is used instead.
- **Spawn Randomness (Optional Input Port):** An optional decimal input port which can be connected to a Wave Randomizer node in order to randomly generate the spawn randomness value. This value denotes an amount of time in seconds where '2.5' is equal to

two and a half seconds and is used to randomly generate a time value between 0 and this value. The resulting value will then be combined with the final 'Spawn Frequency' value which will be the minimum amount of time required to pass before a new item can be spawned. When this port is not connected the manually specified value will be used instead.

- **Out (Output Port):** The default output port used to connect to the next node in the sequence. The connected node will only be evaluated once the 'Loop Chain' node has been fully evaluated the amount of times specified by the 'Loop Count' value.
- **Sub Waves (Output Port):** Allows any number of 'Sub-Wave Nodes' to be connected and will be run in parallel with the main node.

Node Properties

- **Advance Wave Counter:** This value determines whether the wave counter used to track the current wave number will be advanced when the wave is complete. Disabling this option allows you to create a wave which does not increment the wave number.
- **Continue Mode:** This value determines when the connected 'Out' node is evaluated. Available options are:
    - **Never:** The wave will never continue and any connected 'Out' node will never be evaluated. This option can be useful for testing purposes to disable an evaluation branch.
    - **Instant:** The wave will continue and start evaluation the connected 'Out' node as soon as all item spawn requests have been sent to the associated spawner system.
    - **When All Dead:** The wave will continue and start evaluation of the connected 'Out' node when all items spawned during the wave have been destroyed. Each item must be removed from the scene in order to be registered as 'dead' but you can also call 'SpawnableItems.InformSpawnableDestroyed' and pass the 'Transform' component of an object to mark it as 'dead'. See the scripting reference for more details.
    - **When All Spawned:** The wave will continue and start evaluation of the connected 'Out' node as soon as all items have been spawned into the scene. This option differs to the 'Instant' option because it will wait for any pending spawn requests to complete before continuing. Pending spawn requests can occur when a spawn request is sent to a spawner that is currently unable to spawn due to another physical item occupying the spawners collision volume. In this case the spawn request will be queued until the associated spawned becomes available for spawning.

Sub-Wave Node

A wave sub node is a special node that can only be connected to a Wave node. It allows you to add additional spawning behaviours to an existing node which could be used to spawn a different type of item, or use a different target spawner etc. Almost all ports and properties for this node are identical to the 'Wave Node' so duplicates will not be covered in this section.

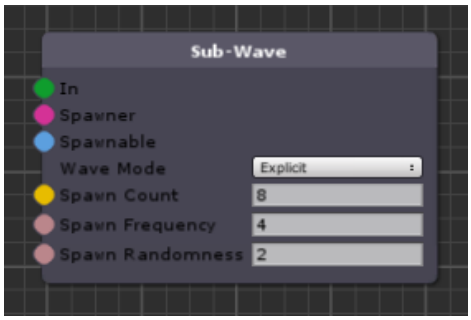


Figure 13

Connection Ports

- **In (Input Port):** The input port for the node which must be connected in order for the node to be evaluated. This port may only be connected to a 'Wave' node.

Wave Randomizer Node

A wave randomizer node can be used to generate a rando decimal or integer value between the specified values. This value can then be fed into another connected node either as a 32-bit integer or a float variable depending upon the connected port. This node can be invaluable for randomizing many wave spawning parameters such as spawn count, spawn frequency, loop counts etc.

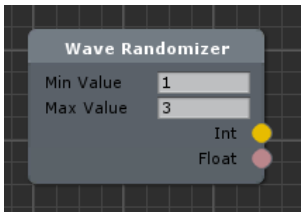


Figure 14

Connection Ports

- **Int (Ouptut Port):** Connect this port to the matching input port of another node in order to feed a random integer value between the specified values into the node property.
- **Float (Output Port):** Connect this port to the matching input port of another node in order to feed a random float decimal value between the specified values into the node property.

Node Properties

- **Min Value:** The lowest value that the randomly generated output value can be. Decimal values are accepted.
- **Max Value:** The highest value that the randomly generated output value can be. Decimal values are accepted.

Spawnable Reference Node

A spawnable reference node can be used to reference a specific spawnable item which can be used by the connected node. The spawnable item can be referenced in a number of ways and must be present in the wave controllers spawnable items collection in order to work correctly.

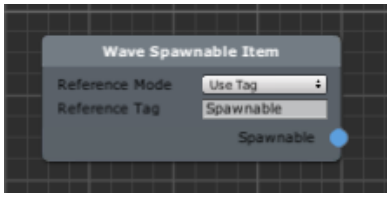


Figure 15

Connection ports

- **Spawnable (Output Port):** Connect this port to another node with a suitable input connection in order to feed the specified spawnable item into the target node.

Node Properties

- **Reference Mode:** The mode used to try and find the spawnable item from the controllers spawnable items collection. Possible options are:
    - **Use Tag (Default):** Try to find the spawnable item with the tag specified via the 'Reference Tag' field.
    - **Use Name:** Try to find the spawnable item with the name specified via the 'Reference Name' field.
    - **Use ID:** Try to find the spawnable item with the ID value specified via the 'Reference ID' value. The spawnable item ID is equivalent to the items array index position into the spawnable items collection.
- **Reference Tag (Visible when reference mode is set to 'Use Tag'):** Accepts a string value which represents the tag that should be searched for in the spawnable items collection.
- **Reference Name (Visible when reference mode is set to 'Use Name'):** Accepts a string value which represents the name of the spawnable item which should be referenced.
- **Reference ID (visible when reference mode is set to 'Use ID'):** Accepts an integer value representing the spawnable item ID to search for.

Spawner Reference Node

A spawnable reference node is very similar to a spawnable reference node but instead is intended to reference. As a result the properties and connections are identical so will not be covered.

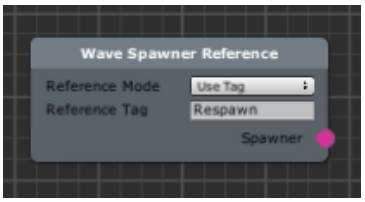


Figure 16

Connection ports

- **Spawnable (Output Port):** Identical to Spawnable reference node.

Node Properties

- **Reference Mode:** Identical to spawnable reference node.
- **Reference Tag (Visible when reference mode is set to 'Use Tag'):** Identical to spawnable reference node.
- **Reference Name (Visible when reference mode is set to 'Use Name'):** Identical to spawnable reference node.
- **Reference ID (visible when reference mode is set to 'Use ID'):** Identical to spawnable reference node.

Wave Wait Condition

A wave wait condition node can be used to holt the evaluation flow until the specified condition is met. This is quite similar to the Wave condition node in terms of its properties however there is only a single output port instead of branching ports since the node will wait for the condition to become true.

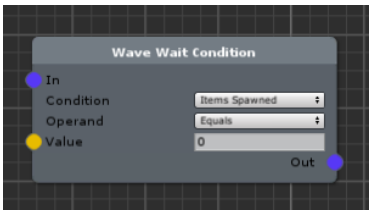


Figure 17

Connection Ports

- **In (Input Port):** The standard input port that should be connected to a previous node.
- **Value (Optional Input Port):** Identical to wave condition node.
- **Out (Output Port):** The out port can connect to other nodes which will be evaluated once the setup node condition has been met. Until then the node will suspend evaluation.

Node Properties

- **Condition:** Identical to wave condition node.
- **Operand:** Identical to wave condition node.
- **Value (Optional Input):** Identical to wave condition node.

Spawn Controllers

Ultimate Spawner 2.0 Waves Add On adds a new spawn controller component to the asset in order to integrate the wave spawning system seamlessly. This wave component can be attached to a game object in the scene and is responsible for evaluating the reference wave node graph at runtime.

Wave Controller

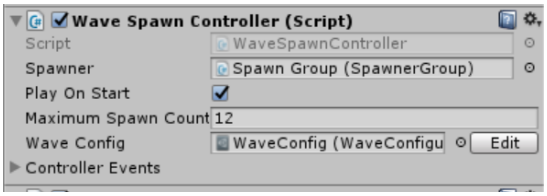
The wave controller component is attached to a scene game object and references a pre-created 'WaveConfig' asset. The controller is then responsible for running the wave configuration at runtime based on the inspector values shown below:

Create Menu

A new wave spawn controller can be created in the scene via the menu **'GameObject -> Ultimate Spawner -> Wave Controller'**. A new wave controller object will be created in the active scene which consists of a game object and an 'UltimateSpawner.WaveSpawnController script component. A wave controller has no scene representation.

Inspector

The wave controller component has a number of inspector value which can be modified to change the behaviour of the controller. The component must have a valid 'WaveConfig' asset assigned otherwise the controller will have no effect when the game is run.



- **Spawner:** The root spawner reference that all spawn requests will be issued to. Any type of spawner can be assigned to this slot in order to handle the spawn requests including group based spawners.
- **Play On Start:** When enabled, the controller will automatically begin its spawning sequence when the game starts. The spawning sequence must be running in order for the physics events to be registered and handled by the controller.
- **Maximum Spawn Count:** The maximum number of concurrent items that can exists in the scene at any time. If the number of alive spawned items equals this value then subsequent trigger events will be queued until on or more items are removed from the scene. You can set this value to '-1' in order to remove the upper limit checks.
- **Wave Config:** A reference to a 'WaveConfig' asset which was previously created. Take a look at the previous 'Wave Graphs' section for more information about wave configuration assets.

Troubleshooting

SYMPTOMS	POSSIBLE SOLUTIONS
NO ITEMS SPAWNING	Ensure that the controller is active. You can enable the 'Play On Start' option to force the controller to run on start-up.
	If 'Maximum Spawn Count' is not set to '-1', ensure that the controller has not already reached the spawn limit.
	Check that the wave configuration asset is valid. Open the asset in the node editor and make sure that any wave nodes with an optional 'Spawner' input assigned is being referenced correctly (Matching name, tags or spawner ID's). Make sure that any wave nodes with an optional 'Spawnable' assigned is also being referenced correctly (Same criteria applies). Check the Unity console for any errors or warnings generated by the wave controller which may offer more information about any issues.

If you are encounter an issue which is not listed here and cannot find a solution to the problem then please reach out to us for support. See the end of this document for contact details.

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 Waves Add On 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 or feature scope but a solution 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/