Tower Defence Wave Manager

User Guide

Overview

Tower Defence Wave Manager is a set of tools designed to provide developers with a quick and easy way to create waves of enemies for a 2D or 3D tower defence game. The aim of this pack is not to create the enemies or combat but provide a generic set of tools that can manage wave creation, weighted enemy spawning, randomised spawn locations and wave playback.

The Wave Player

The wave player stores a list of all waves alongside the logic to play, reset and test for player wins. The wave manager prefab can be found in the content browser and should be dragged into the scene where the waves will take place.

Wave List
A list of all user defined waves that will be played back.

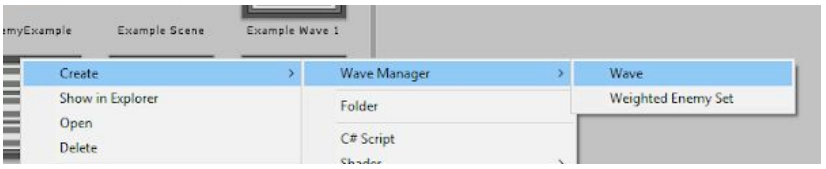
Spawn Location
Defines the location where enemies can spawn. This can either be defined as another game object's location or a vector 3.
Random Spawn Position will randomise the spawn radius in each axis. Setting these values to 0 will force enemies to spawn at the exact defined location.
The spawn location can be visualised in the scene by checking 'Debug Spawn Position'.

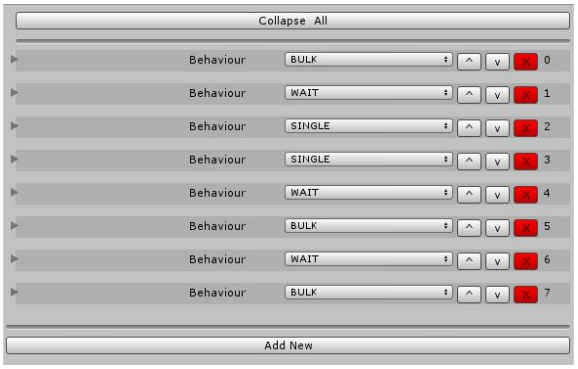
Events
Defines a list of events which will be called at certain points throughout playback. These events can be bound to other actions in the game e.g. reward player at the end of a wave. Examples of their use can be found in the example scenes.

Currently Alive
Displays a list of all alive enemies and a total count when the wave is playing.

Wave Asset

A wave defines a list of elements present in a single wave. An element can be one of three types WAIT, SINGLE or BULK. New waves can be created by right clicking in the asset browser as seen below.





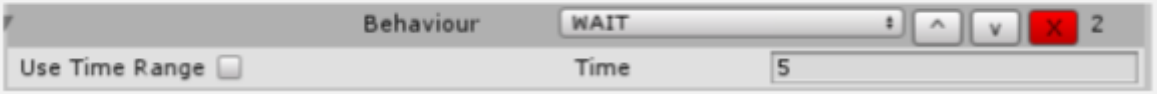
New events can be added with the add new button.

Each element contains the information related to its behaviour. This can be accessed by clicking the arrow to the left of the element.

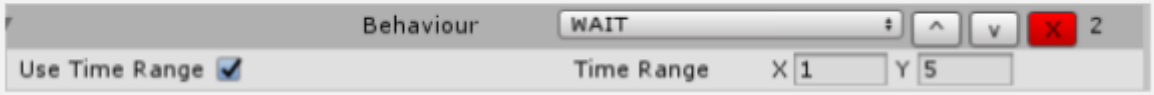
Element types

WAIT

Used to delay for a set amount of time.

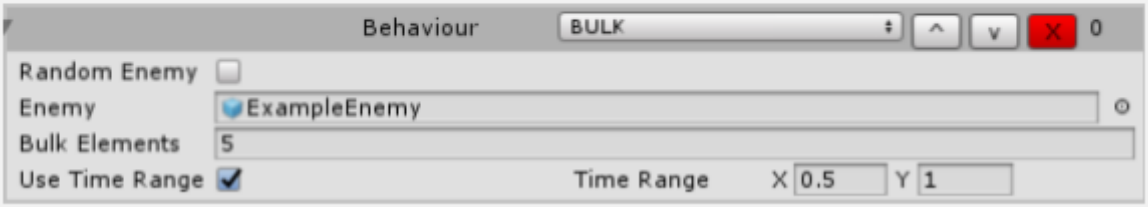


By checking the 'Use Time Range' box, the wave player will delay for a random amount of time between the two set values.



BULK

Used to spawn a set number of enemies in sequence. This element uses the same time fields as the wait type. The time represents the delay between each enemy spawn.



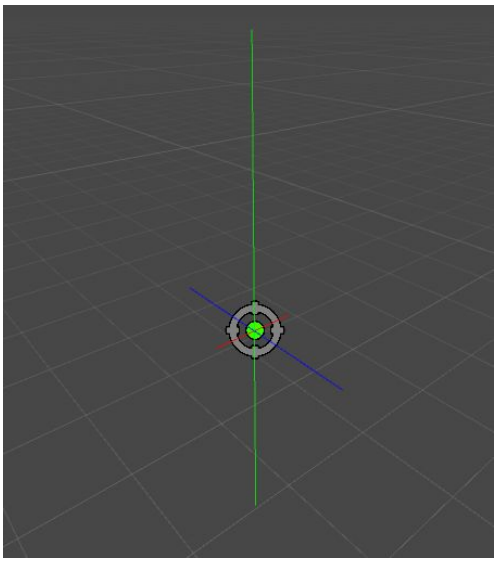
SINGLE

Used to spawn a single enemy.

Randomisation

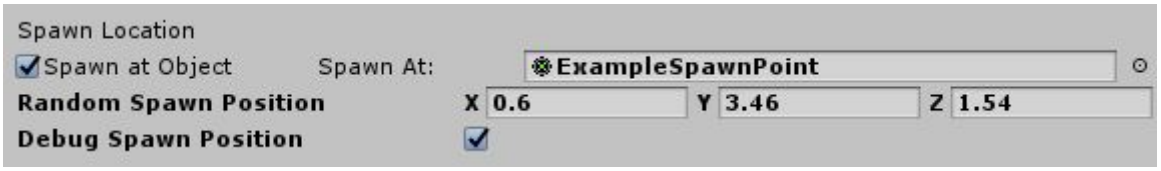
Spawn Point

Within the Wave Player the spawn location can easily be adjusted allowing for enemies to spawn in a random range from the spawn point.



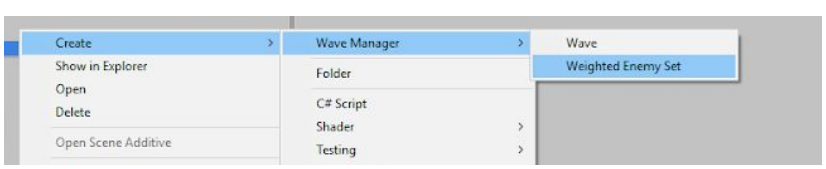
Each field of 'Random Spawn Position' represents the max spawn radius in each axis. Both the 2D and 3D example scenes represent this mechanic.

This distance can be visualised in the scene by selecting the Wave Player and checking 'Debug Spawn Position'

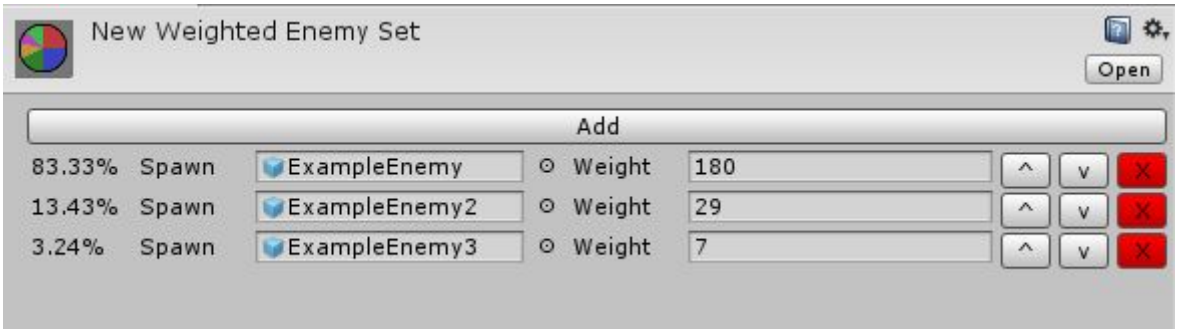


Weighted Enemy Set

A Weighted Enemy Set is an asset which allows the Wave Player to randomly select an enemy based on the weight tied to it. These assets can be created in the asset browser as seen below.



A list of enemies can be defined with a weight tied to them. e.g. a weight of 1 and 3 = 25% - 75%.
The weighted enemy sets can be used in any wave SINGLE or BULK element by checking the field ' Random Enemy' field and providing a reference to the enemy set.



User Callable Functions

The Wave Player includes multiple functions which can be called explicitly by the developer.

The following functions can be called from any other script using WavePlayer.instance. An example using the StartWave() function can be found below.

```
WavePlayer.instance.StartWave();
```

Wave Control Functions

`public void StartWave(){}`
Starts the current wave if no wave is playing.

`public void EndWave(){}`
Ends the current wave, destroys all remaining enemies and moves to the next wave.

`public void ResetCurrentWave(){}`
Ends the current wave, destroys all remaining enemies but does not move to the next wave.

Getter Functions

`public int GetCurrentWaves(){}`
Returns the current wave that the Wave Player has selected.

`public int GetWavesLeft(){}`
Returns the number of waves left in a set. Includes the current wave.

`public int GetRemainingEnemies(){}`
Returns the number of spawned enemies left in the current wave.

`public int CurrentlyInWave (){}`
Returns true if a wave is playing.

`public int lastCompletedWave (){}`
Returns the last waves that has been completed. Returns -1 if no waves have been complete.

`public int GetAllEnemiesSpawned (){}`
Returns true if all the enemies in the current wave have been spawned.