Table of Contents

Killer Waves Quick Start Guide	1
Section One: Setup with Syncro Spawners	2
Section Two: Prefab Pools	8
Section Three: Triggered Spawners	10
Section Four: Child Spawners	12
Section Five: Despawners	13
Section Six: World Variables	14
Section Seven: Killables	16
Section Eight: Listener Scripts	21
Section Nine: Miscellaneous Other Scripts	23
Section Ten: Integration with Other Plugins (Playmaker too!)	23
Conclusion	24

Killer Waves Quick Start Guide

Congratulations on your purchase of Dark Tonic's Killer Waves! You now have a very flexible yet easy to use wave editing system and multi-genre game kit at your fingertips. We also include many event-driven spawning and despawning scripts to save you countless hours of coding. This short guide will show you the quickest way to get moving with Killer Waves. This plugin is designed to do as much as possible without even one line of code, and yet if you wish to go beyond its extensive capabilities and hook up custom code to the events, you can also easily do that.

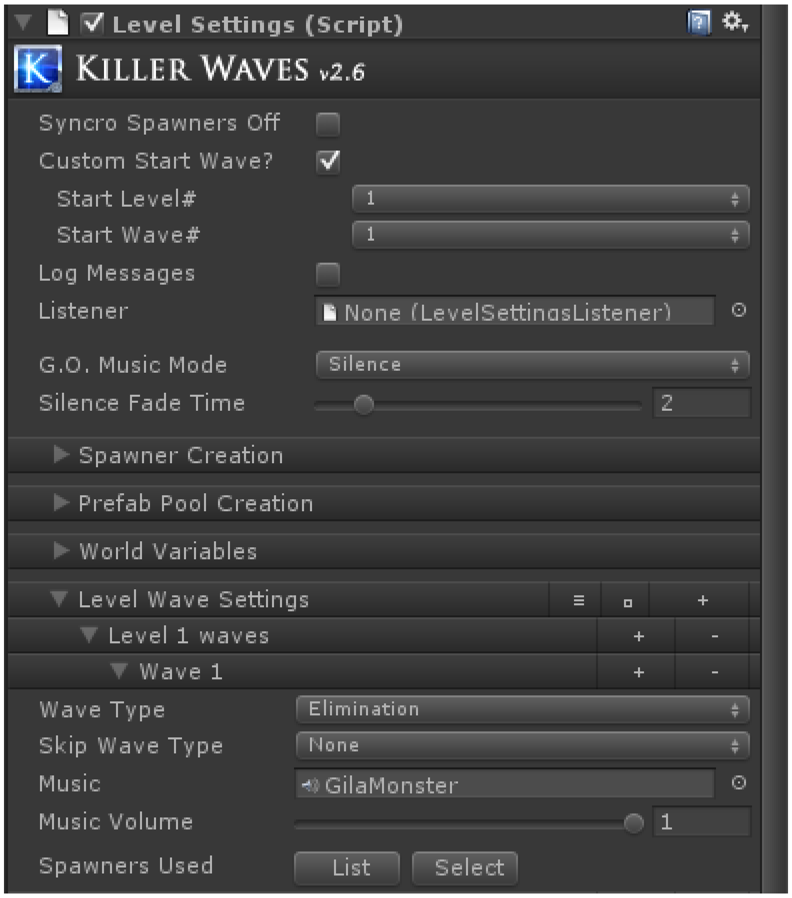
There are multiple demo videos at: http://youtu.be/4QbQdAoAmzQ that go through a quick demo of the new features in each version. Make sure to watch all the videos to get an idea of the more advanced features; however everything is outlined in this document.

Note: All features of the full version are included with Killer Waves Free! However, Triggered Spawners and Killables will stop working 45 seconds after each Scene begins. Also, you may only use 1 Prefab Pool. Other than that, everything is the same!

Section One: Setup with Syncro Spawners

1) Before specifying what each wave contains (enemies, power-ups, etc), you will need to set up at least one "Wave Setting". In Killer Waves, you create one or more Levels, and each Level contains one or more Waves. Each Wave can have zero or more Spawners synched to it to spawn things. So first, drag the LevelWaveSettings prefab into your Scene or Hierarchy. Oh, I must mention that there can only be one LevelWaveSettings prefab in any scene. There are some things that will definitely not behave correctly if you use more than one.

2) Now take a look at the LevelSettings script in the Inspector. It looks something like this:



Level Settings (Script)

KILLER WAVES v2.6

Syncro Spawners Off	☐
Custom Start Wave?	✓
Start Level#	1
Start Wave#	1
Log Messages	☐
Listener	None (LevelSettingsListener)
G.O. Music Mode	Silence
Silence Fade Time	2

Wave Type	Elimination
Skip Wave Type	None
Music	GilaMonster
Music Volume	1
Spawners Used	List, Select

There are a couple checkboxes.

a. You can disable all Syncro Spawners at any time by checking the “Syncro Spawners Off” checkbox (not during game mode though). There are other Spawner types, outlined later in this document.

b. Custom Start Wave - if you turn this on, you can choose the Level and Wave the game will start on. This is good for testing higher waves without having to play the lower ones.

c. The "Use Music Settings" checkbox will enable you to specify music for each wave. Turn this off if you want to control music from outside Killer Waves and all music settings will be invisible to you.

d. The “Log Messages” checkbox, if checked, will log to the Console events such as Level Up, Wave Up, and Spawner Wave initiating, for debug purposes when you have a lot going on.

e. Game Over Music section. This is the same as each Wave’s Music settings (covered below), but is for the music you want played when all waves are completed or game is over (see Other Settings section for setting **LevelSettings.IsGameOver = true if player dies**).

f. New Spawner Section

    i. You set the name and color, then click “Create Spawner”. You may wish to use red for enemy Spawners and green for power-ups or something similar. We may add other colors / textures later.

g. New Prefab Pool Section

    i. We will describe Prefab Pools later. This is a new feature for V 1.3.

h. World Variables Section

    i. We will describe World Variables later. This is a new feature for V 2.0

i. Below that, the Level Wave Settings are listed. You’ll start with zero. To create your first Level, click the plus icon on the right. This will also create the first Wave for that Level.

3) Each Wave in LevelSettings has the following controls:

a. Wave Name - for your reference only. You may reference this in Listener classes and display it when waves are completed.

b. Wave Type. There are two wave types: Timed and Elimination. Timed waves last a certain number of seconds (see Duration settings, next) whereas Elimination waves don’t care about time. An elimination wave is over as soon as all spawned items are either destroyed or despawned (inactive game objects).

   i. **Note** - You can end the current wave by calling **LevelSettings.EndWave()** from code. By making a timed wave of say 10 seconds with no Spawners assigned to that wave and calling **EndWave** when a user says they are ready, you can create common Tower Defense in-between wave setup time.

c. Skip Wave Type - this lets you dynamically skip the wave based on some criteria. The choices are:

   i. None - never skip

   ii. Always - always skip. Useful for testing the high waves without having to play the low levels.

   iii. If World Variable Value Above - this will let you add one or more World Variables (explained later) with target values so you can say that if the player's score is >= 3000, then skip this wave for example.

   iv. If World Variable Value Below - this will let you add one or more World Variables (explained later) with target values so you can say that if the player has <= 2 lives, then skip this wave for example.

d. Duration (number of seconds). This is how long the Wave lasts. This is not visible for Elimination waves.

e. Music Settings

   i. You can specify music (or sound effect clip) to play during each wave (more on this later). You can set the volume of the music here as well. If it is not the first Wave of Level 1, there is also a dropdown for Music Mode which can keep playing the previous wave's music, or stop the music (silence).

   ii. If you choose "silence" as the mode, there is a slider to control how long it takes the previous music to fade out once you reach the silence wave.

f. Wave Completion Bonus - you can enable this section if you want to award Score or any other World Variable modifiers to the player for completing the wave alive. Simply select the name of the World Variable from the dropdown list and enter the value to add (Negative or Positive whole numbers).

g. There are buttons for Listing the Spawners (in the Console) that spawn for the specified Wave, and also a button to select (in the Hierarchy) the same Spawners. Also the number of Spawners is shown there for reference.

h. You can click the plus icon in the Wave to insert a new Wave after that one. Clicking the minus deletes the wave.

i. You can click the plus and minus icons in the Level Settings for the same type of thing for Levels.

j. There are Expand All and Collapse All icons in the top-level Level Wave Settings row.

Cool, now you have a single Wave. Let's create a Spawner to spawn some things for that Wave!

4) Go ahead and click "Create Spawner" in the Inspector to add a green Spawner (or any other color). You can make Spawners invisible in game mode by putting them in a layer that you then turn off in your Camera's Culling Mask.

   a. Expand LevelWaveSettings in the Hierarchy and click on the green Spawner there, then look at the Inspector.

   b. The Spawners also start with zero waves set up. Click the plus to add one.

   c. The up and down arrows that appear when you have more than one wave in a Spawner will move the wave up or down in the list.

   d. Notice that it's set to spawn for Level #1, Wave #1 by default. These are indicating which settings to use from your Level / Wave settings in LevelSettings.

   e. Just for fun, crank up the Min to spawn to 20 and the Max to spawn to 30, then drag any prefab into the "Prefab to spawn" box and press play. Notice how it spawns between 20 and 30 of your prefab within a few seconds. Wasn't that easy? Now I'll explain the other settings.

5) Each Spawner can spawn things during any number of waves. Each Spawner wave has its own settings, as follows:

   a. Level # and Wave#. This specifies which Wave / Level you want the Spawner to spawn things during. The dropdowns for these show only Levels & Waves that exist in LevelWaveSettings. Whenever you change the Level of a Spawner Wave, the Wave# resets to Wave 1.

   b. Min to spawn / Max to spawn. A random number between these two numbers will be chosen for how many of your prefab to spawn.

   c. Time to spawn all. This is the length of time it will take to spawn all X of your prefab. So if you specify 20 for Min to spawn and Max to spawn, and 2 for Time to spawn all, it will spawn 10 per second for 2 seconds. This will not be able to be set higher than the Wave duration in Level Settings.

d. Delay wave (sec). This is used if you want to start spawning your wave later than the precise Wave change in LevelSettings. Just set this to how long you wish to delay in seconds.

e. Prefab Type – Specific or Prefab Pool. If you choose “specific” (the default), you will use the next field (Prefab to spawn). If you select Prefab Pool, you will instead use the “Prefab Pool” dropdown that appears in its place.

    i. Prefab to spawn. As you have seen, this is the prefab you will be spawning copies of. This only is shown when you have selected Specific as the Prefab Type.

    ii. Prefab Pool. Select which prefab pool you want to use as the source of this wave. Prefab Pools are explained later in this document. This field is shown only when you have select Prefab Pool as the Prefab Type.

f. Spawn Limit Controls – checking this box will allow you to limit items spawned by various criteria (only one currently).

    i. Min. Distance – before spawning each item, the Spawner will make sure all items spawned so far are at least this distance from the Spawner. If not, it will delay spawning another item until they are. Note than on timed waves this could result in less items being spawned than your wave size.

g. Repeat Wave. Turning this on will force your wave to repeat automatically when the full wave of prefabs have been destroyed or despawned.

    i. Repeat Mode - This is not visible for Timed waves, otherwise it is here and there are 4 choices:

        1. Number of Repetitions - you choose the number of repetitions with the next field.

        2. Endless - this loops the wave forever.

        3. Until World Variable Above - this will repeat the wave until 1 or more World Variables equal or exceed values you specify below with the "Variable Limit" dropdown and controls.

        4. Until World Variable Below - this will repeat the wave until 1 or more World Variables equal or are less than values you specify below with the "Variable Limit" dropdown and controls.

    ii. Timed Repeat Mode - This is not visible for Elimination waves, otherwise it is here and there are 2 choices:

1. Elimination Style (the default)- the wave will behave like a time-boxed repeated Elimination wave. i.e. wave will repeat for X seconds and repeat every time all items are despawned. Repeat Pause time (below) will begin after all items are despawned.

2. Strict Time Style - the wave will immediately start the Repeat Pause time after the last item has spawned.

iii. Repetitions - only visible for Elimination waves & when Repeat Mode is "Number of Repetitions". If it's a Timed wave, it will repeat until the time has elapsed.

iv. Add Variable Limit - use this to add Limits for the "Until World Variable Above / Above" Repeat Modes. For example, add "Health" and the value 1000 with Until World Variable Below to make the wave repeat until you pass 1000 score.

v. Repeat Pause Min & Max. A random time interval between these two numbers (in seconds) will be paused for before repeating your wave again.

vi. Spawn Increase - this allows you to make the repeated wave have more items spawned than the original. Or vice versa.

vii. Spawn Limit - this allows you to set a ceiling that the Spawn Increase plus current wave size will never exceed. Think of it as "max wave size"

viii. Time Increase - this allows you to make the repeated wave "time to spawn all" longer than the original wave. Or vice versa.

ix. Time Limit - this allows you to set a ceiling for the "max wave spawn time" so that the Time Increase being added every repeat will never make it longer than this value.

h. Randomization. Turning this on will allow you control over randomizing the position and rotation of your spawned prefabs:

i. Random Rotation X / Y / Z. You can specify you want random rotation on the X, Y, and Z axes, or any combination thereof. Any axes you do not check will use the original prefab's rotation for.

1. For each axis you enable random rotation for, a pair of sliders will appear underneath called "Rand. X Rot. Min" and "Rand. X Rot. Max" for the X axis, etc. These will limit the minimum and maximum angle for your spawned items. By default the entire 0-360 degree range is used. You can narrow this to enable smaller ranges of random angles as seen in a popular fruit slicing mobile game.

ii. Random Distance X / Y / Z. You can specify that you want the prefab's spawn location to be a within a certain distance of the Spawner instead of exactly where the Spawner is. If you set Random Distance X to 10 for instance, it will add a random number between -10 and 10 to the spawn position. The same goes for Y and Z.

i. Incremental Settings. Turning this on will allow you to have each successive item in a wave spawn with a changing starting position or rotation. This allows you to spawn a row of enemies (a la Space Invaders), or a wave of enemies rotated to cover 180 degrees of fire.

   i. Distance X / Y / Z. You can specify that each successive spawned item has an extra X * item number to its spawn location. In other words, if you put 10 for X:

      1. Item 1 spawns from the Spawner location.
      2. Item 2 spawns from the Spawner location + 10x.
      3. Item 2 spawns from the Spawner location + 20x, etc.

   ii. Rotation X/ Y/ Z. You can specify that each successive spawned item gets rotated an extra X degrees from the prefab rotation. So if you put 40 for X:

      1. Item 1 spawns with the prefab's rotation.
      2. Item 2 spawns with the prefab's rotation + 40 degrees X (Euler).
      3. Item 3 spawns with the prefab's rotation + 80 degrees X (Euler), etc.

j. Post-spawn Nudge Settings. Turning this on will allow you to alter the spawned prefab's position immediately after it has been spawned and rotation has been applied, whether that rotation came from Incremental or Random Settings. This can make it possible to spawn a ring of enemies where you set the radius of the ring here, and other cool patterns.

   i. Nudge Forward – This controls the distance to nudge the prefab forward after spawning.

   ii. Nudge Right - This controls the distance to nudge the prefab to the right after spawning.

   iii. Nudge Down – This controls the distance to nudge the prefab down after spawning.

6) WaveMusicChanger script

   a. In order to get the Wave Music settings to work, you will need to use another script we have provided. Don't worry, no coding is needed! Simply go to the prefab that contains your music file Audio Source and attach the script by selecting Dark Tonic -> Killer Waves -> Wave Music Changer. That's all there is to it! Note that you will not need an audio clip in that Audio Source component. It will be replaced at wave change time by the script.

7) SpawnUtility.cs

   a. If you are using the Pool Manager plugin, look inside this file and use the commented out code instead of the uncommented code, making sure to specify your spawn pool name instead of "YourPoolName" for both Spawn and Despawn . It will now automatically work with your spawn pool!

8) Other settings

   a. Waves and Levels that are deleted from LevelWaveSettings automatically have their Wave Settings deleted from all Spawners. Also, whenever you insert a new Wave or Level in LevelWaveSettings, the Wave Settings of all Spawners will be adjusted to stay with the correct Level and Wave. If this is undesired functionality for some people, let me know and I'll make a checkbox to turn it off.

   b. Level Settings will continue to advance through your Levels and Waves until there are no more. It has no awareness of if your game has in fact ended by other means. If you want the Spawners to stop at another time, you need to notify LevelSettings that the game is over by setting the following static property:

      LevelSettings. IsGameOver = true

      You can also pause and unpause all Spawners and wave advancement in Level Settings.

      LevelSettings. PauseWave();

      LevelSettings.UnpauseWave();

      This is useful when you might want to show a cutscene before the next wave starts.

   c. Active Mode - this controls whether when the Spawner is active. There are four settings:

      i. Always (default).

      ii. Never

      iii. If World Variable In Range - this lets you define a range for a World Variable value that must be satisfied for the Spawner to be active. For example, the Spawner will be active if your Experience Points is between 1000 and 2000.

iv. If World Variable Outside Range - opposite of "In Range". This will let you not have to worry about upper limits. For example you could say a Spawner is active if outside the range of 0-1000 XP. So it will start to show up after you reach 1000 no matter how many experience points you get.

d. Game Over Behavior – this settings allows you to choose whether the Spawner continues to function when LevelSettings.IsGameOver = true or not. The default is for the Spawner to stop spawning (disable).

e. Wave Pause Behavior - this settings allows you to choose whether the Spawner continues to function when LevelSettings is paused (called by LevelSettings.PauseWave). The default is for the Spawner to stop spawning (disable).

Section Two: Prefab Pools

Using Prefab Pools allows your Spawners to use a weighted list of prefabs instead of a single prefab. Meaning, not only can you add multiple different prefabs into the Prefab Pool, but you can give each prefab a weight, so that you can specify that you want prefab A to appear 10 times as often as prefab B. You can power any number of Spawners with a single Prefab Pool. Spawners that use Prefab Pools for a wave will choose a random prefab from the weighted pool.

i. You create a Prefab Pool from LevelWaveSettings, the same place you create Spawners. There is a Prefab Pool Creation section where you type the Prefab Pool name and a button to create it. Note that Prefab Pool names must be unique. To set up the items in a Prefab Pool, look under the LevelWaveSettings prefab in the Hierarchy and find the child named PrefabPools. Under that you will find all the Prefab Pools.

ii. When you select a Prefab Pool in the hierarchy, it has the following settings in the Inspector:

1. Spawn Sequence – you can choose from either randomized (the default) or original pool order. The latter will spawn from the pool exactly as you have set the items in the pool. Example: 5 of item A, then 3 of item B, the repeat.

2. Exhaust before repeat – this checkbox is on by default and means that the pool items must all be used before it can refill the pool. In other words, if your pool has prefab A with a weight of 2 and prefab B with a weight of 1, and your Spawner spawns 2 of prefab A in a row, then the next item spawned MUST be prefab B, since that is the only item left in the pool (and then the pool will refill and start over). If Exhaust before repeat is unchecked,

each random spawned item is just “another flip of the coin” and the pool is never depleted. This field only appears and is used in random Spawn Sequence.

3. Spawners using – these buttons for “List” and “Select” function exactly the same as the similar ones under LevelWaveSettings. They allow you to see which Spawners use the current Prefab Pool. This is useful so you know what you’ll be affecting before you make changes or delete the pool.

4. The plus icon adds a Prefab Pool item into the pool.

iii. Prefab Pool items have the following settings

1. Each item has a plus and minus icon for deleting the current item or adding new items. There are up and down arrows to move items up and down within the pool (this has no effect on the randomness of the spawning, it’s only for your own organizational visual purposes).

2. Active Mode - this controls whether when the item is included in the Prefab Pool. There are four settings:

    a. Always (default). It's in the pool.

    b. Never (it's never in the pool)

    c. If World Variable In Range - this lets you define a range for a World Variable value that must be satisfied for the item to be in the pool. For example, the item will be in the pool if your Experience Points is between 1000 and 2000.

    d. If World Variable Outside Range - opposite of "In Range". This will let you not have to worry about upper limits. For example you could say an item is in the pool if outside the range of 0-1000 XP. So it will start to show up after you reach 1000 no matter how many experience points you get.

    Note: Options C & D are rechecked for current World Variable values every time the Prefab Pool refills when it has been emptied.

3. Prefab – drag a prefab into this field to control what will be spawned for this item.

4. Weight – the relative weight of this item compared to other items in this pool. The sum of all weights in this pool is the pool size (for depletion and randomness purposes).

Section Three: Triggered Spawners

1) In addition to the Syncro Spawners, there are Triggered Spawners. This type of Spawner does not care about the global waves set up in Level Settings. Waves in this type of Spawner are triggered by various events such as OnBecameVisible, OnCollision, etc. You do not create these Spawners from Level Settings. It is a script you can add to any prefab. You do so from the Component menu under Dark Tonic -> Killer Waves -> Spawners -> Triggered Spawner. By default, it is in "minimal mode" where you simply choose an event from the "event to activate" dropdown and it will then be used as long as the checkbox for that section is checked. To see a list of all events all at once, uncheck minimal mode.

2) Other top-level settings (not per event)

    a. Child Spawner settings. You can set up the Triggered Spawner to have Child Spawners. More on this below. I will just mention the settings here for completeness and we'll go into detail later.

        i. Trigger Source - see Child Spawners section.

        ii. Propagate Triggers - see Child Spawners section.

        iii. Active Mode - this controls whether when the Spawner is active. These are the same settings you saw in the Prefab Pool items earlier. There are four settings:

            1. Always (default).

            2. Never

            3. If World Variable In Range - this lets you define a range for a World Variable value that must be satisfied for the Spawner to be active. For example, the Spawner will be active if your Experience Points is between 1000 and 2000.

            4. If World Variable Outside Range - opposite of "In Range". This will let you not have to worry about upper limits. For example you could say a Spawner is active if outside the range of 0-1000 XP. So it will start to show up after you reach 1000 no matter how many experience points you get.

b. Game Over Behavior – this settings allows you to choose whether the Spawner continues to function when LevelSettings.IsGameOver = true or not. The default is for the Spawner to stop spawning.

3) The full list of events is as follows.

   a. General Events (usable with no plugin requirement)

      i. Enabled Event
      ii. Disabled Event
      iii. Visible Event
      iv. Invisible Event
      v. MouseOver Event
      vi. MouseClick Event
      vii. Collision Event
      viii. Trigger Enter Event
      ix. Trigger Exit Event
      x. Collision2D Event (Unity 4.3+ users only for this and the next 2)
      xi. Trigger Enter 2D Event
      xii. Trigger Exit 2D Event

   b. Custom Events (these are meant to be only called by code you write, if you want to start waves at times other that the events listed)

      i. CodeTriggeredEvent1
      ii. CodeTriggeredEvent2

   c. Pool Manager Events (fired by the Pool Manager plugin only, if you have it)

      i. OnSpawned Event
      ii. OnDespawned Event

   d. NGUI Events (fired by NGUI plugin only, if you have it)

      i. OnClick

Note: OnBecameVisible (and OnBecameVisible) will only work inside a prefab that has a Renderer component inside it. In cases where batching will reassign or not use the Renderer (NGUI / 2D Toolkit, etc), you may opt to use the OnEnable / OnDisable events instead (or the Pool Manager events). They don't provide exactly the same functionality but it will work for most purposes.

4) To spawn a wave from an event, enable the event type using either the minimal mode event dropdown or checking the box for the event (i.e. "Visible Event") if not in minimal mode. Then you will see the section expand and all the settings will be visible. The settings for each section are mostly the same, but there are a few settings that don't show up for certain event types because it would not make sense for them to be there, and would cause a lot of confusion to users tweaking it. The settings for the waves in Triggered Spawner are mostly the same as you have seen in Syncro Spawner waves, but there are a few differences.

   a. Prefab Type, Prefab To Spawn, Number to Spawn, Time to spawn all, Delay Wave (sec) – these settings are all the same as in Syncro Spawners, except the number to spawn is no longer a random number in a range, just a single number.

   b. Layer Filter / Tag Filter – these options show up only under Collision and Trigger Events. By default, all layers and tags will spawn the wave you specify in those sections. These refer to the layer and tag of the other object you're colliding or trigger-colliding with.

      i. If you check the Layer Filter checkbox, you can specify which layer(s) will trigger the wave. All other layers will not.

      ii. If you check the Tag Filter checkbox, you can specify which tag(s) will trigger the wave. All other tags will not.

      iii. Tag and Layer filters work together, so if you select a tag that used by objects in the selected layer filter, the wave will never trigger. Most likely you will only use a layer OR tag filter on a single prefab, but be aware of this.

   c. Repeat Wave

      i. Repeat Mode - the same as in Syncro Spawners, but it's always visible in Triggered Spawners.

      ii. Add Variable Limits - these are visible in the "Until World Variable Above / Below" Repeat Modes. See explanation in the Syncro Spawners section.

      iii. Wave Repetitions: Number of the times to repeat the wave.

      iv. Pause before repeat: time span of seconds between repeating the wave. Delay Wave (sec) will be added to this each time if it is non-zero. So if you have Delay

Wave of .5 and Pause before repeat of .5, there will be 1 second of pause time between waves.

  v. Spawn Increase - the amount of additional items to spawn with each wave repetition. Can be positive or negative.

  vi. Spawn Limit - this allows you to set a ceiling that the Spawn Increase plus current wave size will never exceed. Think of it as "max wave size"

  vii. Time Increase - the amount to change "Time To Spawn All" by with each wave repetition. Can be positive or negative.

  viii. Time Limit - this allows you to set a ceiling for the "max wave spawn time" so that the Time Increase being added every repeat will never make it longer than this value.

d. Randomization, Incremental Settings, Post-spawn Nudge Settings – these settings are all identical to Syncro Spawner wave settings.

e. Other settings

  i. Retrigger Limit – this allows you to prevent the prefab from spawning multiple of the same wave (by that event type) by specifying the minimum reactivation time in seconds or frames since the previous activation. In other words, if you don't want a mouse click to spawn a wave more than twice a second, you can set Retrigger Limit Mode to "Time Based" and Min Seconds Between to 0.5. This settings defaults to "none", meaning no limitation is made.

  ii. Despawn this – this is a checkbox that shows up for certain events that is primarily to keep you from having to create a triggered despawner also when you want to do prefab replacement. So if you want your prefab to turn into an explosion when it has a collision, you simply check "despawn this" and specify the explosion(s) to spawn, and you're done.

Section Four: Child Spawners

With Triggered Spawners, there is the concept of Child and Parent Spawners. If you wish to do things like the following, Child Spawners are worth a look:

1) Would like things to spawn from more than one location during a single triggered event (on visible etc). Maybe you want explosions to spawn from three different unrelated locations when a large enemy gets hit.

2) Want to spawn complex patterns of things not otherwise possible with the Triggered Spawner controls.

Note: Child Spawners are not required to have a Renderer Component to spawn OnVisible event waves. We only need the Parent Spawner to have it. Now let's take a look at how to use Child Spawners.

1) Child Spawners and Parent Spawners both use the same script – TriggeredSpawner.cs. If you already have a Triggered Spawner set up, simply add a child prefab under it.
2) Add a Triggered Spawner script to the child prefab.
3) In the Inspector, change Trigger Source for the Child Spawner to "Receive From Parent".
4) Now all that's left is to configure the event(s) on the Child Spawner that you want to spawn waves of prefabs. Note that Parent Spawners will trigger the same event type, when they are activated, on the Child Spawners. In other words, if the Trigger event is activated on the Parent Spawner, it will then activate only the Trigger event on the Child Spawners. So set up your events accordingly.

Child Spawner Settings (present in all Triggered Spawners though).

1) Trigger Source
    a. None – This basically disables the Triggered Spawner. This can be used for testing purposes or to disable Child Spawners without deleting them.
    b. Self – the default setting. Events will be triggered as normal like a Parent Spawner should.
    c. Receive from Parent – The setting for Child Spawners. All events on the Child Spawner itself will not cause any waves to be spawned, but when the Parent Spawner has those events occur, the Child Spawner will be notified to follow suit and also spawn its waves.
2) Propagate Triggers – this checkbox (only visible when your Spawner has Child Prefabs under it) controls whether or not to trigger the Child Spawner. If you uncheck this box, the Child Spawners will not spawn anything.

Note: You can nest Child Spawners any number of levels deep. That's where the real power of these simple settings comes into play.

Section Five: Despawners

There are three different kinds of despawners included with Killer Waves. They each are for different scenarios.

1) Triggered Despawner – This is an extension of the Triggered Spawner, but for despawning purposes. This can be used to make prefabs automatically despawn when they go off camera, hit objects from certain layers and more. It is attachable to a prefab from the Component menu under Dark Tonic -> Killer Waves -> Despawners -> Triggered Despawner.
    a. It has the following events you can trigger despawns from:
        i. General Events (usable with no plugin requirement).
            1. Invisible Event
            2. MouseOver Event
            3. MouseClick Event
            4. Collision Event
            5. Trigger Enter Event
            6. Trigger Exit Event
            7. Collision2D Event (Unity 4.3+ users only for this and the next 2)
            8. Trigger Enter 2D Event
            9. Trigger Exit 2D Event
        ii. NGUI Events
            1. OnClick Event
    b. For Collision and Trigger Events, there are the same optional Layer and Tag Filters as we used in Triggered Spawners. They work the same way.
    c. For the other events, no other settings are needed.
2) Particle Despawner – This is a script that automatically despawns a prefab with a Particle System (Shuriken) as soon as all the particles have disappeared. It has no settings in the Inspector. It is attachable to a prefab with a Particle System from the Component menu under Dark Tonic -> Killer Waves -> Despawners -> Particle Despawner.
3) Timed Despawner – This is a script that will despawn a prefab after a preset amount of time that you set. It is attachable to a prefab with a Particle System from the Component menu under Dark Tonic -> Killer Waves -> Despawners -> Timed Despawner. It has a couple settings.
    a. Start Timer on Awake – If checked (the default), the timer will start as soon as the prefab is spawned or awake. If not, you will need to call the "StartTimer" method through code when you wish to start the timer.

b. Despawn Timer (sec) – This is how long the countdown timer is before despawning.

Section Six: World Variables

Killer Waves has the highly flexible concept of World Variables. These can represent in-game integers (whole numbers) such as Score, Health, Game Currency or anything else. Killer Waves ships with these and a couple others, although the list is completely customizable by you to include as many as you like. These values are managed by Killer Waves using a cached version of PlayerPrefs that has very fast performance on mobile devices. Let's take a brief tour of their uses!

1) Under the LevelSettings prefab, there is a child prefab called World Variables. Click on it. Now in the Inspector pane is a summary of the World Variables you have to start with. Each variables has the following settings:

    a. Name – this is important to establish early on. If you rename it later after other scripts are using it, you may need to change the value in the other scripts. Don't worry, this does not require any coding, just copy / paste in the Inspectors. I recommend first setting up all the World Variables you think you will need, then using them elsewhere (described later).

    b. Persistance Mode – there are 2 modes.

        i. Reset to Starting Value – this will reset the variable to the value in the next field (Starting Value) when the scene starts. A variable like Score may or may not want to be set this way. The Unity engine can't tell whether you are starting a new game, or just progressing to level 2 for instance. They both are just a scene change. I will show you how to control this in either case shortly.

        ii. Keep from previous – this will allow the variable to be carried over from the previous scene or game. You may want Experience Points or Game Currency to be set like this.

    c. Starting Value – this is the value the variable will be set to initially on the current scene, if you have "Keep from previous" selected as the Persistence Mode. Otherwise the starting value will not be used unless the variable doesn't exist yet when you play the scene. This would only happen the first time any scene with the variable was played (first game on a device).

    d. Allow negative - check this if you want to allow the variable to go below zero, otherwise any modifications to the variable that result in a negative number will instead set the variable to zero.

e. Triggers game over – Check this if a certain value (or range of values) of the variable can end the game. For example, zero health or zero lives.

f. G.O. min value / G.O. max value – G.O. stands for Game Over. This pair of values defines a range of the variable that will trigger game over. When game over occurs, Level Settings.IsGameOver will be set to true, shutting off most of Killer Waves (although you can prevent this with the Game Over Behavior setting on some scripts).

Note: You can see the current value of all World Variable by selecting the top-level World Variables prefab during play in the editor and looking in the Inspector.

2) So, after you're satisfied with your set of World Variable, let's see how to reset some of them when the game starts.

   a. The easiest way is if you have a separate scene, like a main menu, that the user has to go to every time they want to start a new game. If you have one of these, do the following:

      i. Put a World Variable Resetter script into any prefab in that main menu scene. It's under Components -> Dark Tonic -> Killer Waves -> World Variables -> World Variable Resetter.

      ii. Add (click plus icon) a Score Reset for each World Variable you want to reset when this scene is reached, and set its Reset Value. So I might add "Score" and put its Reset Value at zero.

      iii. Make sure this menu scene does not contain a LevelSettings prefab at all or it will not work consistently (or at all).

   b. If this method doesn't always work for you, you can use a line of code to retrieve of modify a World Variable value.

      i. To retrieve, use this static method:

         InGameWorldVariable.GetCurrentWorldVariableValue("*VariableName*")

      ii. To set the value, use this static method:

InGameWorldVariable.SetCurrentWorldVariableValue(“*VariableName*”, newValue)

c. You can add a “listener” to each of your World Variables so that when they get updated, the listener gets notified (so it could update the score display for example). I will be happy to provide an NGUI example file if you email me, but I cannot include it in the package or it would not compile for non-NGUI users unfortunately.

d. Inside the aforementioned class (InGameWorldVariable.cs), if you wish to “go Cloud” with the World Variables, simply change all PlayerPrefs method calls to whatever class uses Cloud and you’re good to go!

3) So far, unless you wrote custom code, nothing will ever change the value of your World Variables during a game. That will not do! That brings us to the one-stop-shopping script, Killable. A Killable script can easily allow multiple-hit targets, auto-despawning, explosions, prefab replacement, World Variable modification (at despawn time) and more!

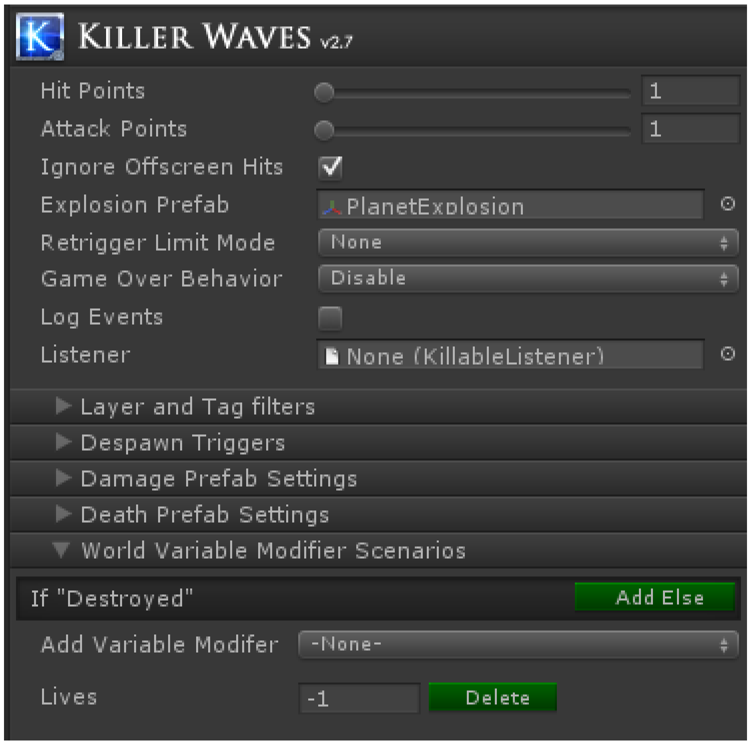
Section Seven: Killables

Killable is a powerful and compact script. It eliminates the use of Triggered Despawner in many cases, and consolidates a lot of functionality into a concise unit. You can set Killables to modify any number of World Variables when they are destroyed. Killables do damage to each other through Attack Points and Hit Points. Go ahead and add one to a scene object and let’s take a look at the settings.

Note: to use Killable, Unity collisions or triggers need to take place for Hit Point exchanges to happen. That means (and this is how Unity works) that for a collision or trigger to take place, both objects need to have a collider Component, and both need to have a Rigidbody Component. And *at least one* of the object's Rigidbody need to be "gravity". For performance reasons, enable gravity on which ever Rigidbody there are less of in any given moment. For me, that may mean that all enemies are non-gravity and both my character and his weapons are gravity.

Note: both 3d and 2d colliders and triggers work for Killable combat.

To add Killable to an object, under the Component menu, choose Dark Tonic -> Killer Waves -> Utility -> Killable. The Inspector should look something like this:



Hit Points 1
Attack Points 1
Ignore Offscreen Hits ✓
Explosion Prefab PlanetExplosion
Retrigger Limit Mode None
Game Over Behavior Disable
Log Events
Listener None (KillableListener)

Lives -1

Note: For Visible and Invisible events to work normally, there must be a Renderer Component on the object you put Killable on. Hopefully the Rigidbody, Collider and Renderer are all on the same component. If you have the Renderer in a child component instead, you will need to add a Killable Child Visibility to the child object that contains the renderer. Do this from the Component -> Dark Tonic -> Killer Waves -> Utility -> Killable Child Visibility menu item. Now, if this child with the Renderer Component is a first-level child of the Killable (it's parent is the Killable), you are done. If it's a second-level child or higher, you will need to drag the Killable object into the "Killable To Alert" field in the Inspector. Then you are good to go!

Let’s go over the settings!

1) Main section

   a. Hit Points / Attack Points. This allows single and multiple hit targets automatically. Hit points is the amount of damage it takes to destroy this Killable. Attack Points is how much damage this Killable does to other Killables when it collides through a collider or trigger. Note that other non-Killable objects will not do any “damage” to Killable if colliding. You may end up in many game genres use Killable for almost every object!

      i. For example, if I’m doing a tower defense game, the first enemy might have five Hit Points and one Attack Point. My first weapon might only have one Attack Point, so it will take 5 direct hits of my weapon to destroy the enemy.

        ii. For items you might want to collect but not damage your player, you can choose zero attack points.

   b. Ignore Offscreen Hits – this is checked by default. This means collisions between Killables that are offscreen (invisible to camera) will not do damage to each other.

   c. Explosion Prefab – If you specify a prefab here, it will be spawned when the Killable is destroyed (by Hit Points, not other means). It does not actually have to contain a Particle Effect at all. But we have another "death prefab" for prefab replacement, with more options, below.

   d. Retrigger Limit Mode – the settings are the same as for Triggered Spawners. On a Killable this will prevent Hit Point reductions from happening again for the specified time or frame limit, if you specify one.

   e. Game Over Behavior – this settings allows you to choose whether the Killable continues to register collisions when LevelSettings.IsGameOver = true or not. The default is for the Killable to stop registering Collisions.

   f. Log Events - turn this on if you want detailed logs of why Hit Points are not being changes via Killable collisions.

   g. Listener – you can add a listener script to this Killable. More on listeners later.

2) Despawn Triggers

   a. Invisible Event – despawn if this is checked.

   b. Not Visible Too Long – this is for edge cases like if something spawns offscreen traveling even further offscreen. In such a case if would never become visible to then become invisible. So you can use this setting to tell the Killable to despawn if it's not visible after it has been spawned for X seconds.

   c. Not Visible Max Time – used in conjunction with Not Visible Too Long. Otherwise it's invisible.

   Note: a-c will not work if you don't have a renderer Component on this object, unless you use the KillableChildVisibility script.

   d. OnClick Event (NGUI event) – despawn when this happens if this is checked.

e. HP Despawn Mode – 3 options

   i. Zero Hit Points – the default. If Killable's HP reaches zero, despawn it.

   ii. Lost Any Hit Points – you would sometimes choose this for your weapons maybe. Even if they lose a single hit point, they despawn. Otherwise if they were triggers, they would continue to pass through many enemies.

   iii. None - the Killable will never die automatically from HP loss. This may be used when integrating with other plugins. In this case, when you decide the Killable should be killed, called the DespawnKillable method of the Killable script.

3) Damage Prefab Settings – a "Damage Prefab" will spawn things whenever your Killable takes damage. You could spawn sparks, smoke, actually enemies, power-ups or whatever you like. Settings are as follows:

   a. Spawn Frequency – this will choose when to spawn damage prefabs. There are four modes:

      i. None – the default.

      ii. Per Hit – this will spawn X (defaults to one) damage prefabs each time the Killable gets hit by another Killable, regardless of how many hit points were lost.

      iii. Per Hit Point Lost – this will spawn X (defaults to one) damage prefab for each hit point lost when the object gets hit by another Killable.

      iv. Per Group Hit Points Lost – this will spawn X (defaults to one) damage prefab each time a certain amount of hit points are lost. You specify that amount.

   b. Group H.P. Amount – this field only shows up for "Per Group Hit Points Lost" and is the amount of hit points the object must lose to spawn one damage prefab.

   c. Spawn Quantity – this controls the "X" in the 3 active modes of Spawn Frequency above. So if you want 4 damage prefabs to spawn instead of 1 (the default), change this settings.

   d. Damage Prefab Type – choose Prefab Pool or Specific. Then you use the appropriate one of the following two settings (only the correct one will be visible).

      i. Death Prefab Pool – the prefab pool to use

      ii. Death Prefab – drag the prefab itself in here.

   e. Random Rotation (x/y/z) – selecting these will assign a random rotation to the damage prefab after it has spawned.

4) Death Prefab Settings – a “Death Prefab” is for prefab replacement. Say you want a car prefab to be replaced with a smashed car prefab with you click it. The smashed car prefab we call a Death Prefab. Here are the settings:

   a. Death Prefab Type – either from Prefab Pool or a specific prefab. Then you use the appropriate one of the following two settings (only the correct one will be visible).

      i. Death Prefab Pool – the prefab pool to use

      ii. Death Prefab – drag the prefab itself in here.

   b. Spawn % Chance – If less than 100, the Death Prefab will only be spawned X% of the time (random). If you specify 50 for this value, 50% of the time you will get a Death Prefab. This is great for spawning power-ups from certain power-ups sometimes.

   c. Spawn Quantity – this is how many of the Death Prefab will spawn if any are spawning. Note that this the Spawn % Chance must be passed before the Spawn Quantity is evaluated. So if you have Spawn % Chance of 20% and Spawn Quantity of 5, then 20 % of the time you will spawn 5 Death Prefabs, and the rest of the time you will spawn zero.

   d. Inherit Velocity – If this is checked and both the Killable and the Death Prefab are gravity rigidbodies, the current velocity of the Killable will be applied to the Death Prefab. This is cool if for instance you wanted a speeding car that hit something turn into a smashed car that still will fly around off the walls due to its momentum. Otherwise it would just stop there, which looks unrealistic (although cool also).

   e. Rotation Mode

      i. Inherit Existing Rotation – this will take the current rotation of the Killable and apply it to the Death Prefab when it is spawned.

      ii. Use Death Prefab Rotation – this will use the Rotation of the actual Death Prefab assigned to the Killable.

      iii. Custom Rotation – you can specify a custom X/Y/Z angle

   f. Custom Rotation Euler – this is for the Custom Rotation setting above. Otherwise, it is not visible.

5) World Variable Modifier Scenarios

   a. A collection of values to add to any number of World Variables is called a Scenario. By default each Killable has just one Scenario called "Destroyed". You can also add any number of additional Scenarios with the "Add Else" button.

b. Within each Scenario, you specify which World Variables to modify (and by how much) when the Killable is destroyed by Hit Point destruction. If you're working in a Scene that has Level Settings in it, the World Variables will show up in the dropdown. Note that you will not be able to assign new modifiers here if your scene does not have a LevelSettings prefab.

   i. Add variable modifier – this works like Minimal Mode in Triggered Spawners. Choosing a World Variable from the list will add a row below.

   ii. After adding a row, there will be text box for the delta you want to apply to the World Variable. For instance if you attached this to your player object, you could use the Lives variable and put the delta at -1. So you will lose a life when your player gets killed.

c. To get any extra Scenarios' to trigger besides the default "Destroyed" one, you have a few choices:

   i. Kill off the Killable by using the Killer Waves Destroy Killable (Playmaker Custom Action). It has a field you can enter the Scenario name in.

   ii. Call the DestroyKillable method on the Killable script yourself, passing in a Scenario name.

   i.e. myKillable.DestroyKillable("scenarioName");

   iii. Subclass the Killable script and override the DestroyKillable method. Add logic to change the Scenario name parameter before calling **base.DestroyKillable**.

   Note: I recommend not choosing this method on Unity 3, because the custom Inspector for Killable will not work on subclasses unfortunately.

   iv. Use a Killable Listener and add logic in the DeterminingScenario method to change the Scenario name before returning it.

Note: If you change or delete the World Variable name in Level Settings after having added modifiers to a Killable, the modifiers will no longer work. Unfortunately, I couldn't figure out a consistent way to get these to auto-delete since your Killables aren't necessarily all in the current scene. However, Killer Waves will alert you when you try to modify a World Variable not in the current scene, during game play, in the Colsone.

6) Layer / Tag filters. These work exactly the same as Triggered Spawners. If you specify a filter, Hit Point collisions will not occur unless the colliding object layer / tag is in the filter list.

Additional note: Killables can only kill other Killables automatically.

Killable has two public methods you can call from code (or Playmaker Custom Action)

1) **`AddHitPoints(int pointsToAdd)`** - this will add X hit points (and update current hit points accordingly). You can use a negative value here as well. This will not register any damage though so if you want to be able to kill a Killable from current hit points changing, use the next method instead.

2) TakeDamage(int damagePoints) - this will subtract the damagePoints from currentHitPoints. This can kill the Killable based on your despawn criteria.

Section Eight: Listener Scripts

To extend Killer Waves to the limit of your imagination, we have, new in V2.0, Listener scripts so that you can add custom behavior through code that will coincide with most Killer Waves events. Each Listener script is an empty skeleton code template that you should not modify. Instead you should create subclasses from them to use for your purposes.

We have included Listeners for:

1) TimedDespawner.cs
2) TriggeredDespawner.cs
3) LevelSettings.cs
4) WaveMusicChanger.cs
5) WavePrefabPool.cs
6) TriggeredSpawner.cs
7) WaveSyncroSpawner.cs
8) Killable.cs
9) WorldVariable.cs

They can be found in the Component menu under Dark Tonic -> Killer Waves -> Listeners. However as I said you should not use these but subclass them instead. The list of events they give you access to are (by Listener class):

1) TimeDespawnerListener
    a. Despawning
2) TriggeredDespawnerListener
    a. Despawning
3) LevelSettingsListener

    a. WaveItemsRemainingChanged (to update a display or any logic)
    b. WaveTimeRemainingChanged (to update a display of seconds)
    c. Win
    d. GameOver
    e. WaveStarted
        i. **Tip**: use LevelSettings.CurrentLevelWave if you want to get the wave #.
    f. WaveEnded
    g. WaveEndedEarly
    h. WaveSkipped
    i. WaveCompleteBonusesStart (in case you want to modify the bonuses before they're awarded).

4) WaveMusicChangerListener
    a. MusicChanging
5) WavePrefabPooListener
    a. PrefabGrabbedFromPool
    b. PoolRefilling
6) TriggeredSpawnerListener
    a. EventPropagating
    b. PropagatedEventReceived
    c. ItemFailedToSpawn
    d. ItemSpawned
    e. WaveFinishedSpawning
    f. WaveStart
    g. WaveRepeat
    h. SpawnerDespawning
7) WaveSyncroSpawnerListener

   a. ItemFailedToSpawn
   b. ItemSpawned
   c. WaveFinishedSpawning
   d. WaveStart
   e. WaveRepeat

8) KillableListener
   a. Despawning
   b. TakingDamage
   c. DamagePrefabSpawned
   d. DamagePrefabFailedToSpawn
   e. DeathPrefabSpawned
   f. DeathPrefabFailedToSpawn
   g. ModifyingWorldVariables
   h. DestroyingKillable
   i. DeterminingScenario - used to change the Scenario used (which set of World Variables are going to be modified)

9) WorldVariableListener
   a. UpdateValue

Note – in each case, you are passed context objects so you can tell exactly what is happening. If you need more events to listen to, let me know!

Using Listener Scripts is easy. Let's take a look at Killable Listener. To set it up:

1) Select a Killable object in your Scene.
2) In the Inspector, add a Killable Listener script from the Component menu (let's just use the stock one for now).
3) Under the Inspector for the Killable script, notice the "Listener" field which is currently unassigned. Drag the Killable Listener script from below into there. Note that you have to grab the line that says "Killable Listener (Script)" and it will drag.

That’s it! Your Listener is now hooked up and will get called from Killable when the events occur. Note that the Listener does not need to be in the same object as the Killable.

Note: There is now an example subclass of the KillableListener in Example Scene 1, attached to the Main Camera prefab. It listens to the Killable on the Player prefab.

Section Nine: Miscellaneous Other Scripts

There are some simple scripts included with Killer Waves.

1) Player Spawner - found under Dark Tonic -> Killer Waves -> Spawner -> Player Spawner. This script can be used to automatically spawn your player, delay respawning by X seconds, and spawn an optional particle prefab as well. This is used in both example scenes.

Section Ten: Integration with Other Plugins (Playmaker too!)

If you are using other popular plugins, we have included some extra packages within out packages to speed up integration.

1) NGUI_KillerWaves - this includes a replacement for the WorldVariableListener that uses a UILabel NGUI element to display the World Variable value and name (optional).

2) Playmaker - there are some custom actions included in the Playmaker_CustomActions package. These show up under the Script Control category.

   a) KillerWavesSyncroSpawnerSpawnOne

   b) KillerWavesDespawn

   c) KillerWavesKillableAttackOrHitPointsChange

   d) KillerWavesKillableDespawn

   e) KillerWavesKillableDestroy (with Scenario parameter).

   f) KillerWavesKillableTakeDamage

   g) KillerWavesWorldVariableChangeValue

   h) KillerWavesWaveEnd

   i) KillerWavesWavePause

   j) KillerWavesWaveUnpause

3) Pool Manager - see the note about SpawnUtility.cs on page 8.

If you would like to see other Custom Actions, let us know!

Conclusion

That’s it for now! We hope your enjoy this plugin as much as we have. Now get making awesome games with this! Dark Tonic will also help you promote your games made with Killer Waves. Email us for details!

Thank you,

All at Dark Tonic

Make sure to check out our other plugins such as Master Audio at http://u3d.as/content/dark-tonic-inc-/master-audio/3PY. Support is available by emailing info@darktonic.com. You can also post on the Unity Forum Killer Waves thread here: http://forum.unity3d.com/threads/167910-Brand-new-Killer-Waves-(wave-spawning-plugin)-demo-video-up!

P.S. The songs used in the example scene are from Brian Hunsaker’s CD “Across the Galaxy”, available here for purchase: http://www.brianhunsaker.net/Store.aspx.