Table of Contents

Core GameKit: Quick Start Guide

Congratulations on your purchase of Dark Tonic's Core GameKit! You now have a very flexible yet easy to use wave editing system and multi-genre game kit at your fingertips. Core features include a full-featured object pooling system and a full combat system! 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 Core GameKit. This plugin is designed to do as much as possible without you writing 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 here 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.

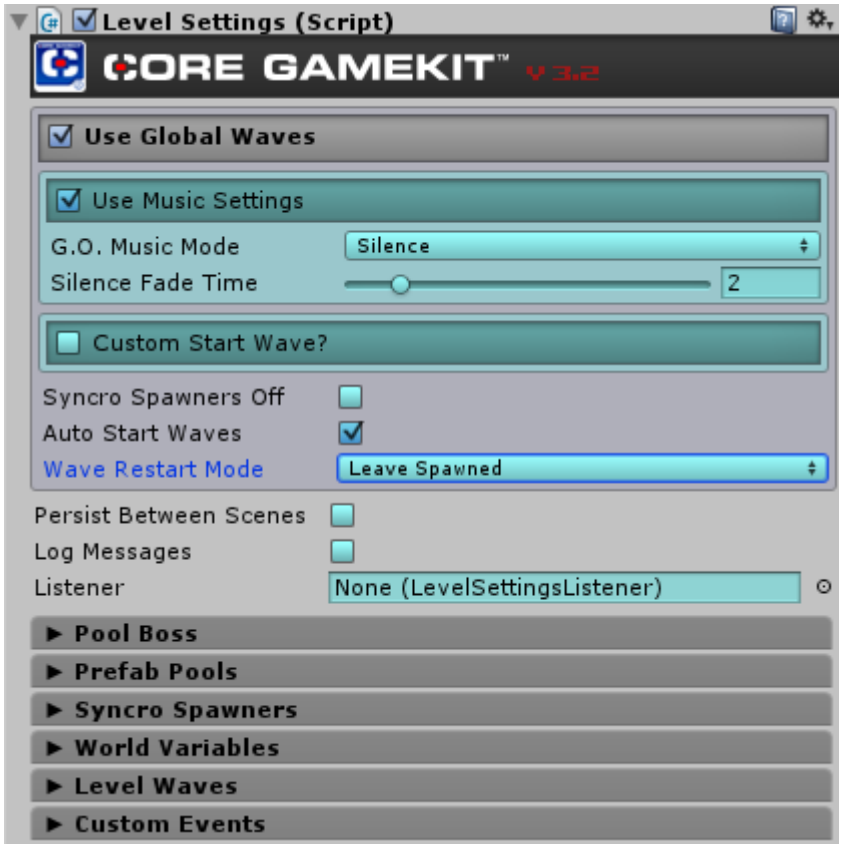
The programming API is located here:

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 Core GameKit, you create one or more Levels, and each Level contains one or more Global waves. Each global wave specifies things like how many seconds it lasts, and can have zero or more Syncro Spawners that use it to spawn things. So first, select the LevelWaveSettings prefab in Project View - it can be found in DarkTonic/CoreGameKit/Prefabs. **Do not drag it** into the Scene or you risk losing your data the next time you update CoreGameKit. Then click the "Create LevelWaveSettings prefab" button.

   Note: 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:



There are a few controls up top.

a. Use Global Waves - turn this on if you use Syncro Spawners so that you can set up Global Waves. When your Global Waves complete, it will trigger "game over", which will turn off all Triggered Spawners and Killables unless you change their Game Over Behavior from the default of "Disable". Game over can also be triggered by World Variable values if you configure them that way.

Note: This checkbox is off by default.

b. 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.

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 Core GameKit and all music settings will be invisible to you.

    i. Game Over Music section (only visible and working when Use Music Settings is checked). 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**).

d. Auto Start Waves - this is turned on by default. If you uncheck it, the first wave starts paused. You will need to call LevelSettings.UnpauseWave to start the first wave when you want (or use the Playmaker Custom Action).
e. Wave Restart Mode - this has 3 choices and controls what happens to already spawned objects when you restart a wave of syncro spawners.
    i. Leave Spawned - nothing happens.
    ii. Destroy Spawned - any spawns that are Killables will have their Destroy methods called, so Death Prefabs will spawn if any. If they aren't Killables, they will despawn.
    iii. Despawn Spawned - any spawns will despawn.
f. 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, or custom continues if you use World Variables for the values.
g. Persist Between Scenes - checking this box will mean the Level Settings game object is not destroyed when you load a new Scene.
h. 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.
i. Listener - here you can drag in a LevelSettingsListener to hook into key events and do customized coding. More on Listeners later.
j. Pool Boss Section
    i. Core GameKit includes Pool Boss! A powerful object pre-loading and reuse system (similar to Pool Manager) to avoid Instantiate and Destroy calls during game play.
k. Prefab Pools Section
    i. We will describe Prefab Pools later. This shows you a list of any Prefab Pools you have created and a settings icon to navigate to each one.
l. Syncro Spawners Section
    i. This section shows you each Syncro Spawner and a settings icon to navigate to each one. You also can create more Spawners here.
        1. 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.
m. World Variables Section
    i. We will describe World Variables later. This section shows a summary of each World Variables, its type and a settings icon to navigate to each one.
n. 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.
o. Custom Events Section
    i. There's a section about Custom Events later (check the table of contents up top). This section lets you define the events. More on this later!

3) Each Level has the following settings:

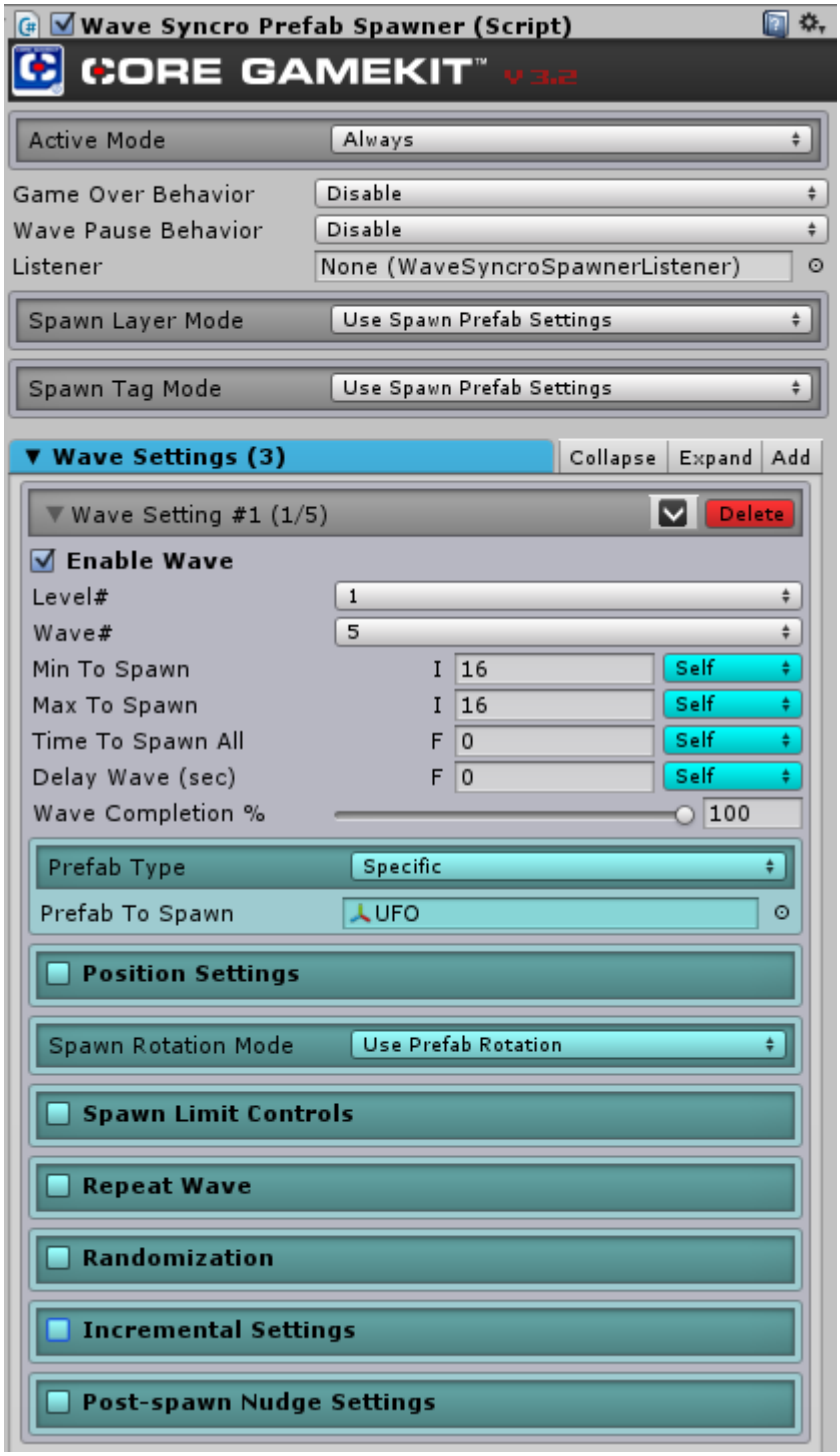
a. Wave Sequence - 2 choices.
    i. Specified Order (default). The waves will be in the order you specify in the Inspector, top to bottom.
    ii. Random Order. If you select this, the waves in the Level will be ordered randomly when the Level is started. You will still play all the waves.

4) 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. There are also various other scripting methods you can call to do things. Please refer to the API link on page 2 of this document.
    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!

5) 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 just before you press play, go to the Pool Boss prefab (explained later) under LevelSettings and click Auto-Add Missing Items. Now 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.
6) Each Spawner can spawn things during any number of waves. Each Spawner wave has its own settings. Here's a screen shot.



Note: You can click on the Core GameKit logo banner that appears at the top of all Core GameKit script Inspectors to be taken back to the LevelWaveSettings prefab. It's a shortcut.

The controls are explained here. Note the blue dropdown with "Self" selected. This will be explained in the World Variables section. You can ignore it for now.

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. Wave Completion %. This field only appears if the wave you are configuring is an Elimination Wave. The default is 100%, but here you can specify that the spawner's wave is complete if X% of the items spawned are destroyed or despawned.
f. 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.
g. Position Settings
    i. X / Y / Z position mode: Use Spawner Position or Custom Position. Each axis lets you use a number, a variable or the Spawner's position for spawning.
    ii. Wave Offset - this is a Vector3 that will be added to the Spawner's position before any Incremental or Post-Spawn nudge settings. This will allow you to spawn from a different location than the spawner. This respects the spawner's rotation so X, Y and Z are actually Right, Up and Forward. This Vector3 gets added to the position created from the field above (X/Y/Z position mode settings).
h. Spawn Rotation Mode -3 choices.
    i. Use prefab rotation - this is the default. The prefab will be spawned with whatever rotation the prefab has itself.
    ii. Use spawner rotation - this will rotate the spawned prefabs to match the spawner's current rotation immediately after spawn.
    iii. Custom rotation - this allows you to specify a custom rotation of X, Y, and Z euler degrees. Spawned prefabs will be rotated to match this immediately after spawn.

i. 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.

j. 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.
x. Wave Repeat Bonus. If you check this box, you can assign a set of modifications to any number of World Variables every time the wave is repeated. This is identical to the "Killable destroyed" section for World Variables. Read more about that later when we tackle Killables.

k. 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.
l. 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.
    iii. Keep Center - check this if you would like the incremental rotation settings be equally split between left and right. It enables effects like Contra's infamous spread shot. The Triggered Spawner example scene has an enemy using this.
m. 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.

7) 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 -> Core GameKit -> 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.
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.

      LevelSettings.RestartWave();

      This can be used to start a wave over. Repeat count goes to zero and wave unpauses.

   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).
f. Spawn Layer Mode / Spawn Tag Mode - these settings allow you to change the layer and/or tag of each object spawned from this spawner. You can "inherit" the layer / tag of the spawner itself, keep the prefab's original, or use a custom layer / tag.

Section Two: Prefab Pools

Using Prefab Pools allows your Spawners to use a weighted list of prefabs instead of a single "hard-coded" 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. A Prefab Pool however has nothing to do with the other type of pooling - the type that gets around Instantiate and Destroy calls by enabling and disabling objects (like Pool Manager).

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. Scene Objects Using – these buttons for “List” and “Select” function exactly the same as the similar ones under LevelWaveSettings. They allow you to see which Spawners & Killables 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).

Note: During runtime, you can click on the Prefab Pool and in the Inspector, it will show the number of each item remaining in the pool at all times.

Section Three: Pool Boss

Pool Boss is a prefab pre-loader / recycler similar to Pool Manager. If you're familiar with Pool Manager, you will have no trouble using Pool Boss. Here's a screen shot.

If you are unfamiliar with the reasons for using a pooling solution, here are the basics. Instantiate and Destroy calls (what normally happens if you don't use pooling) can cause stuttering when complex objects are created (and destroyed) during game play, especially on mobile devices. A pooling solution will create everything up front when the Scene starts, and everything created at start will be in a despawned state initially. You simply specify which items you want in that pool, and how many of each. They are disabled GameObjects when despawned and are enabled when you spawn them. Performance is massively affected and more predictable by a system such as this. The settings are as follows:

1. Top-level controls
    a. The arrows, plus icons and delete icon are self-explanatory and you've seen them all over Core GameKit on other features.
    b. Auto-Add Missing Items - checking this box will actually allow you to "half-pool", meaning that you can check the box and hit play without adding any Pool Items. What will happen is that nothing will be created up front, but Pool Items will be created (Instantiated - bad for performance) as you need them, and can be reused. Also, any prefab that is in the Hierarchy (already in the Scene and doesn't need to spawn) will get

despawned in to the pool when it goes away so you can re-use it later in that Scene. If you do not check this box, any prefabs you try to spawn that don't have a matching Pool Item will instead log an error and nothing will spawn.

   c. Log Messages - checking this box will tell you when each prefab spawns, despawns, or if Pool Boss needs to Instantiate an extra copy because you are already using all copies of the prefab.
   d. Alpha Sort - clicking this button will sort the Pool Items A-Z.
   e. 2 more buttons are available at runtime:
      i. Kill All - this will destroy all active Game Objects that are set up in Pool Boss and are using a Killable component. Those that are not using a Killable component will be unaffected.
      ii. Despawn All - this will despawn all active Game Objects that are set up in Pool Boss.

2. Per Item Settings
   a. Prefab - drag a prefab here to specify which prefab you want to pool.
   b. Preload Qty - here you specify the amount of copies of the prefab to create when the Scene starts. They will all begin despawned (inactive in the Hierarchy) and will appear as a child of the PoolBoss prefab that is under LevelWaveSettings.
   c. Allow Instantiate More - this defaults to off. If you check this box, and there are no remaining copies of the prefab despawned, another copy will Instantiate (not good for performance). Leave this checkbox on if you aren't sure how many you need. Later you can adjust the Preload Qty and turn this off.
   d. Item Limit - this is only visible and used when Allow Instantiate More is checked. This is used to put a limit on the amount of items that can be Instantiated after the Scene begins. This number will default to your Preload Qty.
   e. Log Messages - checking this box lets you log messages for only the Pool Items you check instead of all Pool Items. Good for troubleshooting.
3. Notice the yellow text "2/9 spawned" in the screen shot above. That appears at runtime only to let you know how many are spawned and what the total number of all copies of the Pool Item is (spawned + despawned). This is also great when you have turned on Auto-Add Missing Items. You can play the busiest section of your game, and just before pressing stop, pause the game and click on PoolBoss to take note of the totals so you can create or modify your Pool Items with those numbers in mind. Good time-saver! You can click on this text to select all the spawned items in the Hierarchy.
4. Also during runtime, more buttons will appear on each item row. These are the same as the same-named buttons in the top-level section, but apply only to the single prefab configured in that pool item.
   a. Kill All
   b. Despawn All
5. Code usage - there are events broadcast whenever a prefab is spawned or despawned. Any script on any prefab (or sub-prefab) can perform actions in these event handlers. Just to keep

things easier for people migrating from or to Pool Manager, I have named the events the same as they did.

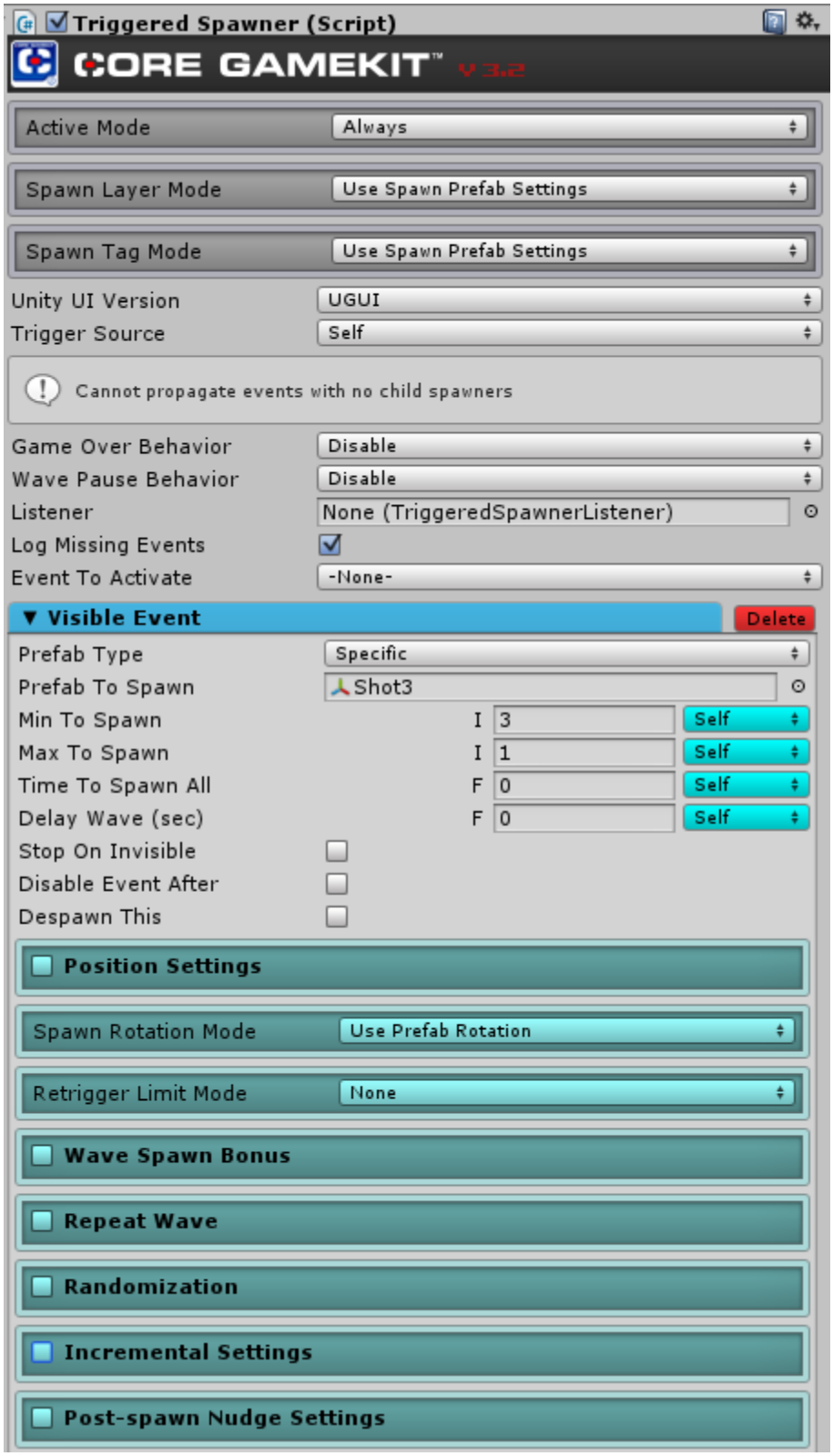
a. OnSpawned - this code fires immediately after the prefab has spawned. Note that when not using pooling solutions, you put a lot of initialization code in the Awake or Start event. However, you will want to move some or all of that code into the OnSpawned event so that it fires each time the prefab is respawned. This is because the Awake and Start events will not fire again when the prefab is respawned. Take a look in the KW_Enemy.cs script in the Example Scenes for a useful pattern. I made a method called AwakeOrSpawned and call it from both events (Awake and OnSpawned). That way it will work even if the prefab doesn't spawn the first time (already in the Scene).
b. OnDespawned - this code fires immediately before the prefab despawns. This should be cleanup code only, and it should not take longer than one frame to execute (starting a CoRoutine is not advised). When despawn occurs, the prefab will be re-parented as a child of the PoolBoss prefab under LevelWaveSettings automatically.

   Note: If you are using a pooling solution, never call Instantiate or Destroy! If you destroy a cloned prefab in the pool by code, it will no longer be available the next time Pool Boss needs it. Pool Boss will log an error if this happens so you can find your code that did it.

   Note: If you want to spawn Particles or audio items, make sure to attach the Timed Despawner script to those prefabs and set the Life Seconds to the amount of time the audio should play or the amount of time the particle takes to disappear. Then these prefabs will return to the Pool. However, if you need a way to pool audio, we do recommend that you check out our flagship product Master Audio.

Section Four: 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 -> Core GameKit -> Spawners -> Triggered Spawner. 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.
2) Here's a picture of a Triggered Spawner Inspector



Other top-level settings (not per event)

a. Log Missing Events - if checked, this will log an error to the Console if it is using Custom Events that don't exist in your Level Settings game object.
b. Unity UI Version - if you are on Unity 4.6+, you can select whether you're using Legacy UI events of uGUI. This will filter out the other choices from the Event To Activate dropdown.
c. 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.

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.

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. Mouse Over (Legacy) Event (old Unity GUI, before V4.6)
        vi. MouseClick (Legacy) Event (old Unity GUI, before V4.6)
        vii. Slider Changed (uGUI) - all the uGUI events are only visible and usable if you put the Triggered Spawner component on a Unity 4.6 component (has RectTransform component).
        viii. Button Click (uGUI)
        ix. PointerEnter (uGUI)
        x. PointerExit (uGUI)
        xi. PointerDown (uGUI)

xii. PointerUp (uGUI)
xiii. Drag (uGUI)
xiv. Scroll (uGUI)
xv. UpdateSelected (uGUI)
xvi. Select (uGUI)
xvii. Deselect (uGUI)
xviii. Move (uGUI)
xix. InitializePotentialDrag (uGUI)
xx. BeginDrag (uGUI)
xxi. EndDrag (uGUI)
xxii. Submit (uGUI)
xxiii. Cancel (uGUI)
xxiv. Collision Enter Event
xxv. Trigger Enter Event
xxvi. Trigger Exit Event
xxvii. 2D Collision Enter Event (Unity 4.3+ users only for this and the next 2)
xxviii. 2D Trigger Enter Event
xxix. 2D Trigger Exit Event

b. Code-Triggered 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 Boss Events
    i. OnSpawned Event
    ii. OnDespawned Event
d. NGUI Events (fired by NGUI plugin only, if you have it)
    i. OnClick
e. Custom Events (covered in detail in the section about Custom Events - see table of contents). Each Triggered Spawner can be set up to receive 1 or more Custom Events with separate waves defined for each.

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 Boss 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 the Event To Activate dropdown. 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, Min/Max to Spawn, Time to spawn all, Delay Wave (sec), Wave Offset, Position Settings – these settings are all the same as in Syncro Spawners.
b. Stop On Invisible (or Stop On various Trigger events) - This lets you end an wave when the opposite event happens. So for example you can end a "visible" wave when the Game Object becomes invisible.
c. 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.
d. Wave Spawn Bonus - instead of Repeat Wave Bonus (as Syncro Spawners have), Triggered Spawners have a Wave Spawn Bonus, which works a little differently. This is a set of World Variable modifiers as well, but it triggers even on the first spawn of a triggered wave.
e. 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.

ix. Use Wave Spawn Bonus - this is only visible if you have enabled Wave Spawn Bonus for this wave. This checkbox controls whether all repeats of the wave also modify the same variables.

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

g. Other settings

i. Spawner Rotation Mode - This field only shows up for Custom Event waves. It has 2 choices.

1. Keep Rotation - the default. Don't change the rotation of the spawner when the custom event happens.
2. Look At Custom Event Origin. Using this option will make the spawner turn to look at the event origin. Good to make enemies chase the player when he performs certain actions!

ii. Spawn Rotation Mode -4 choices.

1. Use prefab rotation - this is the default. The prefab will be spawned with whatever rotation the prefab has itself.
2. Use spawner rotation - this will rotate the spawned prefabs to match the spawner's current rotation immediately after spawn.
3. Custom rotation - this allows you to specify a custom rotation of X, Y, and Z euler degrees. Spawned prefabs will be rotated to match this immediately after spawn.
4. Look At Custom Event Origin (only valid for Custom Event waves). This will spawn all wave items looking at the event origin point.
5. If you choose this rotation mode, you have the option to ignore specific axes X, Y or Z and use the spawner's X,Y or Z instead for the LookAt.

iii. 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.

iv. Disable Event After - check this if you want the event to only be able to happen once in this Spawner. For instance, only the first time a trigger is entered, it will spawn. The event will be disabled after that wave spawns.

v. 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. Note that the despawn will happen immediately, so if you've chosen a Time To Spawn greater than 0, it may not finish spawning items.

Section Five: 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 Six: Despawners

There are three different kinds of despawners included with Core GameKit. They each are for different scenarios.

1) Triggered Despawner – (Generally not needed if you use Killable on your Game objects). 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 -> Core GameKit -> 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 -> Core GameKit -> Despawners -> Particle Despawner.

    Note: if you are especially concerned with performance (mobile games are), you can put a Timed Despawner component (described next) on a particle system. It is much lighter on performance. You just need to figure out how long the particle should be visible if you go that route.

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 -> Core GameKit -> 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 Seven: World Variables

Core GameKit has the super flexible concept of World Variables. These can represent in-game integers (whole numbers) and floating point numbers (commonly called decimals) such as Score, Health, Game Currency or anything else. Core GameKit 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 Core GameKit using a cached version of PlayerPrefs that has very fast performance on mobile devices.

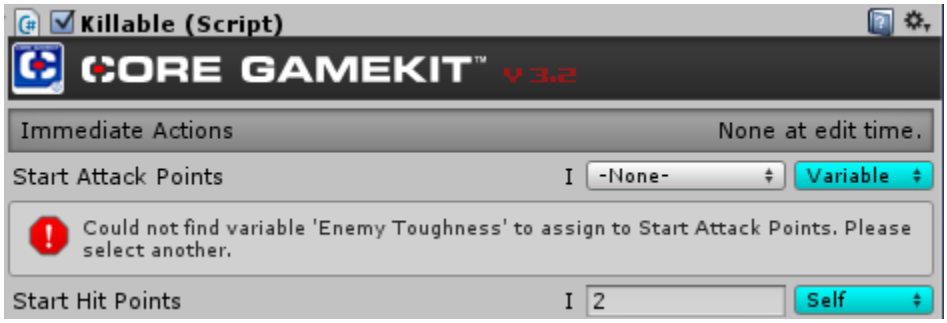
World Variable are not just for displayed stats like score and currency. You can use these to control and manipulate any integer or float field in any of the Core GameKit Inspectors! Notice below the blue dropdown with the word "Self" in the below image. This of this as a "value source" dropdown. That means it will get the value (for Hit Points let's say) from what you enter into the number field to the left. The green "I" or "F" denotes whether the Variable is an int or a float.

If you choose "Variable" from the blue dropdown, the number entry field will change to a World Variable selection dropdown, showing only the World Variables of the same type (int or float). See that I have now selected the "Enemy Toughness" Variable for the Hit Points in the next image.

This means that whenever the Hit Points of this prefab is evaluated, it will use the current World Variable's value! Powerful stuff, considering that every wave beat, Killable destroyed, wave spawned and tons of other events can modify the World Variable values.

Note: Incorrect / deleted / renamed variables. Core GameKit will tell you about these in two ways

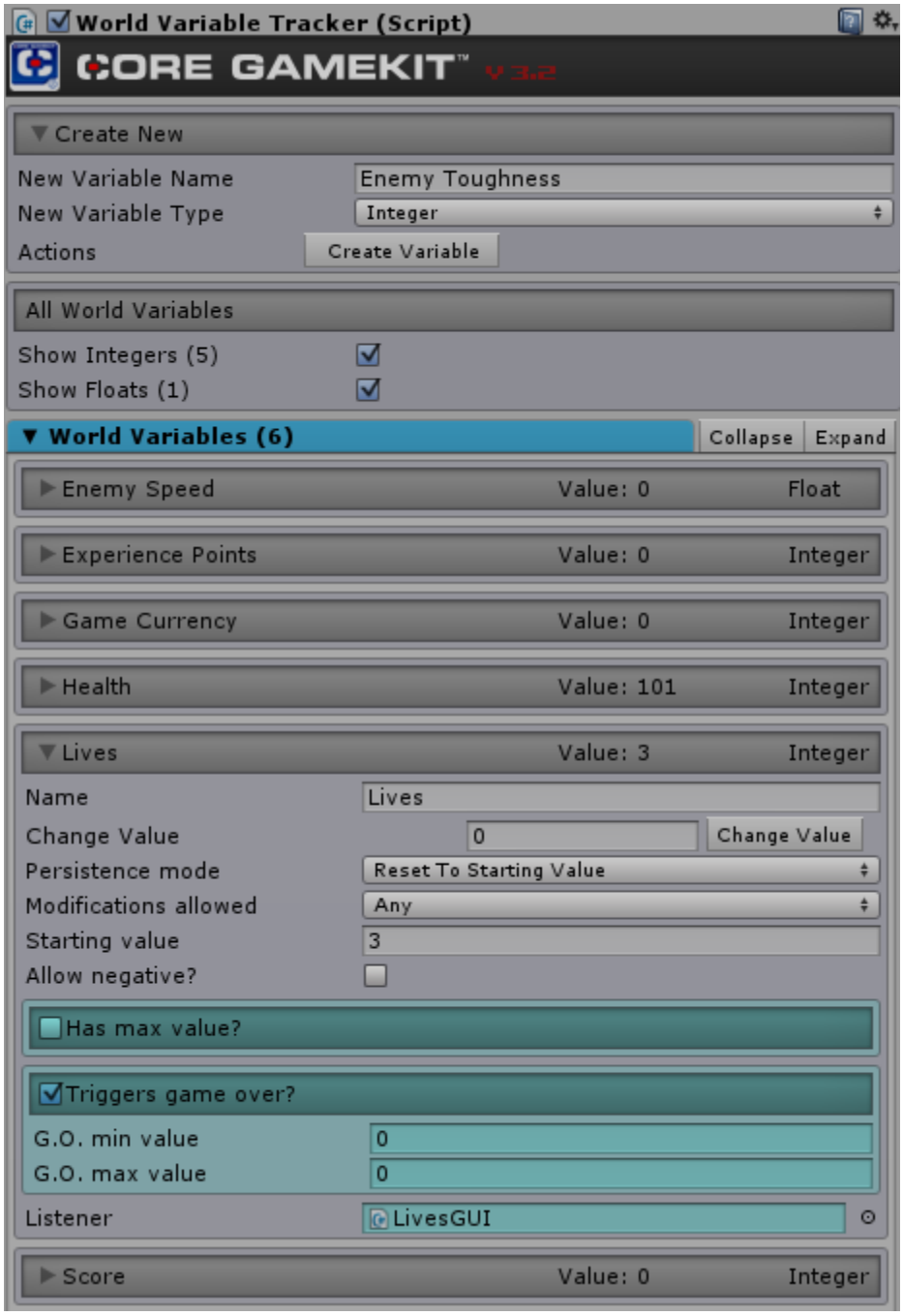
1) It will be prominently display in the Inspectors like so



2) All scripts that use World Variables check all their configuration upon the Awake event, and will log errors to the Console window to tell you very specifically what is wrong.

Now let's see how to set them up!

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. The type of each Variable is shown in yellow near the right. Also, during play, the value of each Variable is displayed there as well and there are controls to change the Variable value at any time.



2) Each variable 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. Variable Type (integer or float) - this is only settable when you're just about to create a World Variable, so make sure you choose the correct one. If you only need whole numbers, choose integer. If you need decimal points, choose float.

c. Persistence 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. If it happens that you have Keep from previous selected, but there is no previous value, the Start Value will be used (next field).
d. Modifications Allowed - controls how the variable can be modified.
    i. Any (no restrictions)
    ii. Only Increase - the variable can only be increased. Useful for something like a high score.
    iii. Only Decrease - the variable can only be decreased.
e. 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).
f. 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.
g. Has Max Value - this defaults to off. If you check this box, you can choose a maximum value for the variable. This is useful for things like making sure Health never goes over 100.
h. Max Value - this only shows up and is used if Has Max Value is checked.
i. 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.
j. 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 Core GameKit (although you can prevent this with the Game Over Behavior setting on some scripts).

Note: At runtime, you can see the current value of **all** World Variable at once by selecting the top-level World Variables prefab during play in the editor and looking in the Inspector.

3) 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. Add a LevelWaveSettings prefab.
ii. Go to the World Variables node under it.
iii. Add entries just for the World Variables you would like to reset (score etc). Make sure their Type matches the Type in the other scene (integer etc).
iv. Change their Persistence Mode to Reset To Starting Value and change the Starting Value field to the value you want.
v. Delete the entries for all other World Variables you don't need to reset.

b. If this method doesn't fit your scenario, you can use a line of code to retrieve of modify a World Variable value.
   i. To retrieve, use this static method:

      var variable = WorldVariableTracker.GetWorldVariable("yourVarName");

      Then, you will retrieve the value with one of these properties, depending on the type of the Variable. Only Int or Float is used within each Variable.

      1) variable.CurrentIntValue
      2) variable.CurrentFloatValue

   ii. To set the value, use the same code above, but assign a value to the property. i.e.

      var variable = WorldVariableTracker.GetWorldVariable("yourVarName");
      variable.CurrentIntValue = 45; // or CurrentFloatValue for a float!

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 have provided an optional NGUI package that converts the WorldVariableListener into an NGUI one.

4) 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 Eight: Killables

Killable is an extremely powerful and compact script, used for pickups (loot / health), enemies, player objects, weapons and more! 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 normally, 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.

****Important for 2D users:** to get Killable to both damage each other when one is getting destroyed, you will need to open the Physics 2D screen from Edit -> Project Settings -> Physics2D and turn off the checkbox labeled "Change Stops Callbacks". This is important because Unity 2D collisions are treated differently by default.

Note: If you have child objects under a Killable that have their own triggers or colliders, this can cause unwanted "combat hits" to occur from colliding with the other child triggers. To prevent this, give the child collider Game Objects a Kinematic Rigidbody.

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

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. However, 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 -> Core GameKit -> Combat -> 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!

Note: If you have the collider on a child object, add the Killable Child Collision component to the child. Its setup & usage is the same as Killable Child Visibility (above). Add it from the Component -> Dark Tonic -> Core GameKit -> Combat -> Killable Child Collision menu item.

Let’s go over the settings!

1) Main section

a. Immediate Actions. These buttons are only visible when you're playing (not in edit mode).
    i. Kill - this will instantly kill the Killable.
    ii. Despawn - this will instantly despawn the Killable (death prefab settings and Death World Variable Modifiers will not be used.
    iii. Take 1 Damage - this will inflict 1 point of damage on the Killable.
b. 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.
    iii. Start Attack Points - the attack points that the Killable will begin with.
    iv. Start Hit Points - the attack points that the Killable will begin with.
    v. Max Hit Points - the maximum amount of hit points possible.
    vi. Sync H.P. Variable - this checkbox is only visible if you're using a World Variable for Start Hit Points. Defaults to off. If you check the box, the World Variable will be updated with the Current Hit Points of that Killable every time they change. This is good for a player Killable to update a Health variable for instance. It is not a good idea to use this on a prefab that has more than one instance in the Scene, because the World Variable value will not be predictable and may jump around.
c. 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.
d. 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.
e. 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.
f. Game Over Behavior – this settings allows you to choose whether the Killable continues to register collisions or use Respawn settings when LevelSettings.IsGameOver = true or not. The default is for the Killable to stop registering Collisions and not respawn.
g. Log Events - turn this on if you want detailed logs of why Hit Points are not being changes via Killable collisions.
h. Listener – you can add a listener script to this Killable. More on listeners later.

2) Invincibility Settings

a. Invincible - check this to make the Killable only deal damage but not receive it while it's checked. Good for temporary invincibility and other effects.
b. Inv. While Children Alive - defaults to true. This means that you cannot damage this Killable until you destroy all sub-Killables it has under it (read next section on Composite Killables for more info on sub-Killables).
    i. Disable Colliders Also - only visible and works when the previous checkbox is checked. Defaults to false. Turning this on will disable the colliders for this Killable while it has any sub-Killables under it. This is useful if your sub-Killables wouldn't be able to be hit otherwise because of their positioning in relation to their parent (this Killable). When the last sub-Killable is destroyed, the collider will be re-enabled automatically. A good option to have.
c. Invincible On Spawn - check this if you want the Killable to be invincible for a configurable amount of time, each time it spawns. If this is checked, the next field is visible.
    i. Invincible Time (sec) - the amount of time the Killable will be invincible when it spawns, only if Invincible On Spawn is checked.

3) 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.
    a. Ignore Killables I Spawn - checked by default. Checking this means that regardless of layer or tag, any Killables you collide with that were spawned from this Killable will not do any damage to this Killable. This can allow cool things in multiplayer games such as spawning projectile weapons (even if they're in your layer filter) that don't hurt you. However, that same projectile shot by another player will damage you.
4) 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. Damage Prefab Pool – the prefab pool to use
ii. Damage Prefab – drag the prefab itself in here.

e. Spawn Offset (Vector3) the distance from the Killable's Transform to spawn the damage prefab.
f. Random Rotation (x/y/z) – selecting these will assign a random rotation to the damage prefab after it has spawned.

5) Damage World Variable Modifiers
   a. Here you can specify any number of World Variable modifications to make any time the Killable takes a valid hit.
6) Despawn & Death 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. The Not Visible Too Long controls are now in KillableChildVisibility's Inspector. Use those instead of b-c above if you need to use them and the child has the renderer.

   d. MouseDown Event - destroys if you click on the Killable. Note that this will not work if your Killable is in the Ignore Raycast layer.
   e. OnClick Event (NGUI event) – destroys when this happens if this is checked.
   f. If Spawner Destroyed? This allows you to make the Killable die or despawn (difference explained below) if the object that spawned it is destroyed (or despawned). This works if the Killable was spawned by a Killable damage prefab or death prefab, a Triggered Spawner, or a Syncro Spawner. Anything in the KW plugin.
      i. Do Nothing - this is self-explanatory and is the default.
      ii. Despawn - these other 2 modes will keep track of when the object that spawned this Killable is despawned or destroyed. If that happens, this mode will despawn the Killable.
      iii. Die - same as Despawn mode, but will spawn the Death Prefab is any was specified.
   g. Use Death Timer - if you check this box, you can have a Killable destroy or despawn itself after X seconds. The following controls will appear when it's checked.
      i. Death Timer (sec) - the number of seconds before the Killable will destroy or despawn itself.
      ii. Time Up Action - choose Do Nothing, Destroy or Die. The action you select will happen when the Death Timer has finished.

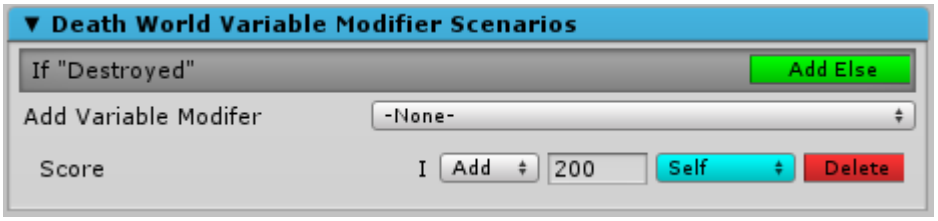
h. HP Death Mode – 4 options
    i. Zero Hit Points – the default. If Killable's HP reaches zero, despawn it.
    ii. Lost Any Hit Points – you would sometimes choose this for your weapons maybe. Even if they lose a single hit point, they despawn. Otherwise if they were triggers, they would continue to pass through many enemies.
    iii. Collision or Trigger - same as Lost Any Hit Points, but it will fire even when the Killable in invincible. Useful for making sure your projectile doesn't go through an object if they're both invincible for instance.
    iv. 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.

7) 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 Delay (sec) - here you can specify a time delay before the Killable dies after lethal damage. That way you can play a death animation for example (using WaitingToDestroyKillable method in the KillableListener).
    b. 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.
    c. Keep Same Parent - defaults to on. If you uncheck this box, the death prefab will be spawned not as a child of its parent Transform (if it has one), but under the Pool Boss prefab instead.
    d. 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.
    e. 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.
    f. Spawn Offset - this lets you specify a distance from the dying Killable's position to spawn the death prefab(s).
    g. 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).
    h. 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

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

8) Death 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.
      iii. You can also opt to not type in a value and choose "Variable" from the blue dropdown. Then you can chose to Add, Set, Multiply or Subtract the value of a World Variable you select from the Variable on the left. For instance, the following would multiply your score times the number of lives you have when this Killable is destroyed!



   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 Core GameKit 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, Core GameKit will alert you when you try to modify a World Variable not in the current scene, during game play, in the Console.

9) Respawn Settings
    a. Death Respawn Type - 3 choices.
        i. None (the default) - Killable will despawn when dead. To change the respawn location, set the property RespawnLocation on the Killable at any time.
        ii. Infinite - Killable will respawn at its original spawn point the next time it dies, an infinite number of times.
        iii. Set Number - Same as Infinite option, but you specify the number of times it will respawn. After that number, it will despawn the next time it dies.
    b. Respawn Delay (sec) - If you have chosen a Death Respawn Type other than "None", this field appears and allows you to delay the respawning by X seconds.

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 Nine: Composite Killables

Composite Killables are prefabs that have a top-level Game Object with a Killable script on it, and one or more sub-prefabs that also have Killable scripts on them. There are settings to allow the top-level Game Object to be invincible (and optionally disable its collider) while any child Killables are still alive. You can also have the sub-prefabs also have sub-prefabs with Killables on them - any number of hierarchy levels are supported. Constructing a prefab like this can make huge boss enemies where you have to destroy all the turrets or other moving parts before you can take down the boss itself. There is an example scene called "5.CompositeKillables" that shows off this feature.

Note: Composite Killables cannot be respawned (Respawn settings in Killable script) or returned to Pool Boss for re-use if one or more of its child Killables have been despawned or destroyed. Instead, the Composite Killable Game Object will automatically be set to inactive when it is despawned or destroyed

if that is the case. The reason is that Unity does not provide a way to revert a prefab instance during runtime currently . This isn't normally a problem as prefabs like this aren't re-used very often, just be aware of it. So, make sure to not turn on any despawn checkboxes on the child Killables (like Invisible) as it can make things not despawn even though you didn't destroy anything.

Tip: if you want to make multiple levels of destruction of sub-Killables, make sure the "Keep Same Parent" box is checked on the Death Prefab settings of that sub-Killable (it is the default though).

Section Ten: Custom Events

You can define any number of Custom Events in the Level Settings prefab's last section. Core GameKit automatically notifies all receivers when the event is fired so they can perform an action if they are configured to respond to that exact event. Custom Events can be used for the following things (with more uses coming soon):

1. Trigger a wave in all Triggered Spawners in the Scene that "receive" that Custom Event. You set up receiving by adding a wave of event type "Custom Event" in the Triggered Spawner, then selecting the Custom Event name from a dropdown. Then fill out your wave specifics as normal.

The interface - all Custom Event Receivers (including TriggeredSpawner) must implement the ICgkEventReceiver interface. That way Core GameKit will keep track of the receiver and be able to automatically notify it when events are fired. You can look at the bottom of the Triggered Spawner script in a region called ICgkEventReceiver events if you wish to write your own custom implementation and want to see an example.

In the Level Settings Inspector, you can create Custom Events by doing the following:

1. Type the name of your new Event and click the Create New Event button. An event will be created and shown in a row underneath.
2. Choose an option from the Send To Receivers dropdown for the new event. This controls what happens each time the Custom Event is fired. The options are:
    a. Always - always send the event to all receivers. This is the default.
    b. Never - never send the event to any receivers (this disables the event without deleting).
    c. When Distance Less Than - only send the event to all receivers that are closer than the Distance Threshold. Cool for smart enemies that attack when you get close.
    d. When Distance More Than - only send the event to all receivers that are further away than the Distance Threshold.
3. Distance Threshold - the distance to use for When Distance Less / More Than.

During runtime in this Inspector, the number of receivers for each event will be shown on the event row. Buttons also appear then to select all receivers in the Hierarchy and to fire the custom event from the position of the LevelSettings prefab.

Typical event setup:

1. Create the event as described above.
2. Set up the Custom Event Receiver. Either use the Triggered Spawner script (enable Custom Event and select the event name) or create your own class implementing the ICgkEventReceiver class.
3. Be able to fire the event. The following code will do it:

LevelSettings.FireCustomEvent(string customEventName, Vector3 eventOrigin);

The event original is normally the position of the Transform of the object that is firing the event. It will be used for distance calculations and other useful things as well.

Section Eleven: Listener Scripts

To extend Core GameKit to the limit of your imagination, we have Listener scripts so that you can add custom behavior through code that will coincide with most Core GameKit 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 - can display / update the value on a Text component in Unity 4.6.

They can be found in the Component menu under Dark Tonic -> Core GameKit -> 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. EliminationWaveCompleted (will tell you when an elimination wave is completed, whether it is about to repeat or not).
    b. GameOver
    c. Lose (Game Over but you didn't win)
    d. WaveItemsRemainingChanged (to update a display or any logic)

    e. WaveTimeRemainingChanged (to update a display of seconds)
    f. WaveStarted
        i. **Tip**: use LevelSettings.CurrentLevelWave if you want to get the wave #.
    g. WaveEnded
    h. WaveEndedEarly
    i. WaveSkipped
    j. WaveCompleteBonusesStart (in case you want to modify the bonuses before they're awarded).
    k. WaveRestarted
    l. Win
4) WaveMusicChangerListener
    a. MusicChanging
5) WavePrefabPooListener
    a. PrefabGrabbedFromPool
    b. PoolRefilling
6) TriggeredSpawnerListener
    a. CustomEventReceived
    b. EventPropagating
    c. PropagatedEventReceived
    d. WaveEndedEarly
    e. PropagatedWaveEndedEarly
    f. ItemFailedToSpawn
    g. ItemSpawned
    h. WaveFinishedSpawning
    i. WaveStart
    j. WaveRepeat
    k. SpawnerDespawning
7) WaveSyncroSpawnerListener
    a. ItemFailedToSpawn
    b. ItemSpawned
    c. WaveFinishedSpawning
    d. WaveStart
    e. WaveRepeat
8) KillableListener
    a. SpawnerDestroyed
    b. Despawning
    c. TakingDamage
    d. DamagePrefabSpawned
    e. DamagePrefabFailedToSpawn
    f. DeathDelayStarted
    g. DeathPrefabSpawned
    h. DeathPrefabFailedToSpawn

i. ModifyingDamageWorldVariables
j. ModifyingDeathWorldVariables
k. WaitingToDestroyKillable (used to play a death animation or other side effect)
l. DestroyingKillable
m. DeterminingScenario - used to change the Scenario used (which set of World Variables are going to be modified)
n. Spawner Destroyed

9) WorldVariableListener
    a. UpdateValue

Note: WorldVariableListener has some fields for display logic.
1) Decimal Places (number of decimal places).
2) Use Comma formatting.

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).

That's it! The Listener will look for a compatible object to listen to in the same GameObject automatically. i.e. adding a Killable Listener to a GameObject with Killable will automatically hook them together.

If you want the listener to live in a different GameObject, the steps are a little different.

1. In the Inspector, add a Killable Listener script from the Component menu (let's just use the stock one for now) to the GameObject you want to listen.
2. Select the Killable you want to listen to.
3. Under the Inspector for the Killable script, notice the "Listener" field which is currently unassigned. Drag the Game Object with the Killable Listener script from below into there.

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 Twelve: Miscellaneous Other Scripts

There are some simple scripts included with Core GameKit.

1. Player Spawner - found under Dark Tonic -> Core GameKit -> 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.
2. ClickToKillOrDamage - found under Dark Tonic -> Core GameKit -> Combat -> Click To Kill Or Damage. This script you can put onto an object that has a single instance in the Scene (Main camera is a good candidate). What it does is, whenever the left mouse button is clicked down or a finger is tapped down (on mobile), it will destroy or apply X damage (whichever one you specify) to the first Killable directly under your finger or mouse, determined by a RayCast. The settings are:
    a. Kill Objects - checkbox. Defaults to checked. When this is checked, the Killable will be destroyed if you tap or click on it.
    b. Damage Points To Inflict - whole number. If Kill Objects is unchecked, the Killable you tap or click will have this number of points damage taken immediately. If that takes its current Hit Points to zero, it will be destroyed.

    This script is used in Example Scene 4. Take a look!

    Note: There's also a 2d version of this script called ClickToKillOrDamage2D. That one should be used instead to tap 2d sprites.

Section Thirteen: 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_CoreGameKit - 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 several custom actions included in the Playmaker_CustomActions package. These show up under the Script Control category in Playmaker.
    - Core Game Kit Syncro Spawner Spawn One
    - Core Game Kit Goto Wave
    - Core Game Kit Killable Attack Or Hit Points Add
    - Core Game Kit Killable Attack Or Hit Points Mod
    - Core Game Kit Killable Despawn
    - Core Game Kit Killable Destroy (with Scenario parameter).
    - Core Game Kit Killable Get Current Hit Points
    - Core Game Kit Killable Take Damage
    - Core Game Kit Killable Temporary Invincibility
    - Core Game Kit Pool Boss Despawn
    - Core Game Kit Pool Boss Despawn All Prefabs
    - Core Game Kit Pool Boss Despawn Prefabs Of Type
    - Core Game Kit Pool Boss Item Despawned Count

- Core Game Kit Pool Boss Item Is In Pool
- Core Game Kit Pool Boss Item Spawned Count
- Core Game Kit Pool Boss Item Total Count
- Core Game Kit Pool Boss Kill All Prefabs
- Core Game Kit Pool Boss Kill Prefabs Of Type
- Core Game Kit Pool Boss Prefab Count
- Core Game Kit Pool Boss Spawn
- Core Game Kit Triggered Spawner End Wave
- Core Game Kit Triggered Spawner Has Wave Type
- Core Game Kit Variable Float Add
- Core Game Kit Variable Float Get Value
- Core Game Kit Variable Float Multiply
- Core Game Kit Variable Float Set
- Core Game Kit Variable Int Add
- Core Game Kit Variable Int Get Value
- Core Game Kit Variable Int Multiply
- Core Game Kit Variable Int Set
- Core Game Kit Wave End
- Core Game Kit Wave Pause
- Core Game Kit Wave Unpause
- Core Game Kit Wave Restart

There are also Playmaker Listeners for each of the 9 Core GameKit Listeners. The Playmaker Listeners let you choose an FSM and FSM event on the same game object to fire off when the Listener event occurs. You can select the FSM and event from dropdowns! Couldn't be easier!

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

Section Fourteen: Other Plugins that have Core GameKit Integration

The list is small currently, but we will see it grow over time.

1. Dialogue System - this powerful quest / dialogue system has incorporated Core GameKit support (as well as tons of other plugins) in their latest version.
2. Behavior Designer - this powerful AI / behavior plugin is extremely easy to use and is in many cases a faster performing replacement for Playmaker. A demo version of Behavior Designer with Core GameKit integration is included in our plugin download package. If you already have this plugin, we have also supplied a package that installs all the tasks for Core GameKit instead.
3. AI for Mechanim - a very versatile tool that uses Mechanim-looking graphs to accomplish AI with no code writing necessary! Has integration with most of the top plugins!

Section Fifteen: Using JavaScript or UnityScript

You will need to move the Assets/DarkTonic/CoreGameKit/Scripts folder into the Assets/Plugins folder so that it will compile in the right order and be accessible from other scripting languages. It's wise to also rename the "Scripts" folder to "CoreGameKit" so you know what they are after moving.

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 Core GameKit. Email us for details!

Thank you,

All at Dark Tonic

Make sure to check out our other plugins such as the top-selling 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 Core GameKit thread here: http://forum.unity3d.com/threads/167910-Brand-new-Killer-Waves-(wave-spawning-plugin)-demo-video-up! Or use the official Dark Tonic forums here: http://darktonic.freeforums.net