Asset Management: Spawning & Despawning

Overview

Every world in Horizon is saved as a world snapshot. The snapshot contains the list of shapes, scripts, gizmos, and how they are all organized. When you edit a world in build mode a new snapshot is generated. If you ever make a mistake, you can use the My Creations Page to revert a world back to a previous snapshot).

An **asset** is a collection of objects and scripts that you can save outside of all of your world snapshots. For example, you can make a hat, save it in your Asset Library, and then delete the hat from the world you made it in. In the future you can then add that hat to any world by pulling it out of your Asset Library in build mode. A copy of that asset is then saved in that world's current snapshot. Note that if you edit the asset in the library, the copy you just created is not updated.

Additionally, scripts can load and unload assets into a running instance without the asset needing to be stored in the world's snapshot. You can have a world that has 1000 outfits (which would likely be over capacity) by choosing which of them to dynamically load while the world is running in an instance. To load objects into a running world instance you **spawn an asset**. This loads in a copy of that asset as a collection of new objects and scripts. To unload that copy, you **delete the spawned object(s)** ("despawning" that copy of the asset).

This feature makes it possible to spawn in items such as scenery, wearables, interactive objects, and much more, but only when you actually need them. For example, a world could unload portions of a map to free up capacity to then load in others. As another example, you don't need to make 10 copies of a hat; instead, you can make the asset once, and put it in your asset library. Then, without even having any copies in the world snapshot, you can spawn in as many as you need when the world is running (as long as the spawned objects stay under 100% on all capacity meters).

Spawning *is similar to placing objects/gizmos/assets from your build menu in build mode except that it does not modify the save-state of the world.*

Deleting spawned object *is similar to deleting objects/gizmos/assets in build mode.*

Spawning Considerations

Keep track of all spawned objects. If you want to delete spawned objects then you need to keep track of them. At minimum, when you spawn in an object you should store it in a script variable or list so that you can reference it in the "delete spawned object" code block. Capacity limits are still there. Even though the spawning occurs in play mode, the capacity limits are in effect. Each asset has a record of how much capacity it needs on each meter (e.g. objects, complexity, vfx, etc). When the asset spawns in, it takes up the capacity on each of those meters. If spawning the asset would take any of the meters over 100% then the spawn will fail. Currently there is no way to detect a failure. There is also currently no way to see how much capacity an asset needs (other than to pull out the asset in build mode and look at the relative change in the meters).

Asset spawning follows all the same capacity rules as build mode. If you publish a world with 90% object capacity and then have that world spawn assets in publish mode, the 90% object capacity will keep increasing until it hits 100% object capacity after which no new assets will spawn. At that point you would need to despawn previously spawned objects to free the object capacity again.

Make assets self-contained. Assets must be entirely contained to work properly (whether imported in build mode or spawning in scripts). This means that all referenced objects and script gizmos attached to objects must be contained within the asset. Assets currently cannot include persistent gizmos such as leaderboards and achievements since they would not be self-contained. When an asset is spawned, the scripts are renamed if needed to prevent name clashes (just like when you pull out the asset from the library).

Be mindful of asset size. Larger assets will take longer to spawn and once they do spawn, the lighting calculation will take longer to settle.

Current Limitations (as of June 2022)

- **Only spawn single-group assets.** There are currently issues with spawning assets that are not a single group: the event callback doesn't reference all spawned objects, there is no way to delete all the spawned objects, and the spawned objects will not be correctly positioned or rotated. For now you should only spawn assets where all objects are grouped together into a single group.
- **No failure detection.** If an asset fails to spawn there is no way to detect such failure.
- **Not all gizmos can be included.** Since assets must be self contained you cannot use any gizmos that "reference" specific world data. This means that you cannot use the leaderboard gizmo in an asset and likewise cannot use leaderboard code blocks in a script used inside an asset. The same applies to the gizmo and code blocks for achievements and also to the code blocks for persistent variables.
- **Existing instances won't get updates to an asset.** If you modify an asset, new instances of your world will get the updated asset. Any already-running world instances will continue to spawn the version that existed when the instance started. There is currently no way to force an already-running instance to refresh / update assets.
- **Wait a frame before sending messages.** Currently you can't send events to objects in the same frame they were spawned; those events get ignored (which is a bug). When you get a reference to a spawned object you need to send any messages to it with a delay or on a later frame. The bug only occurs right when the object spawns, so after that one frame you can send events to the object as normal.
- **Assets cannot be scaled when spawned.** - The spawn asset code block allows you to specify the position and rotation but not the scale. If you want to scale a spawned asset it will need to be dynamic. The same applies to changing colors or other properties.

How To Spawn Assets

1. **Create an "asset" variable in a script.** To spawn an asset you need a script with an asset variable (which you will set to contain an asset in step 5). You can call this variable whatever you like, e.g. "hatAsset", "hat", "asset", or whatever else you like.
2. **Apply the script to an object.** The script from step 1 needs to be on an object (just like any other script); that object will run the code blocks that spawn the asset.
3. **Open the scripted object's property Panel.** Open the property panel of the object from step 2; here you will see the Asset Variable field as "empty" in the variables section of the property panel.
4. **Find the asset in Asset Library.** From your build menu navigate to your asset library, then navigate to the asset you want to spawn. Select the view info ("i") icon on that asset.

5. **Connect the reference pill.** On the property panel you opened in step 4, scroll down to see the asset reference pill (blue in color with the asset name){picture}. Select and drag this reference pill to the Asset Variable field "empty" on the objects property panel (Example Object "Obj") where the script was applied in step 3. The field will have the variable name that you chose in step 1. ***Note:*** *Unlike object references, there is no 'wire' drawn back to the asset in the asset library, just the name of the asset is shown in the variable value field.*

6. **Spawn the Asset.** The script from step 1 can now use the "spawn asset" code block. You use the variable that you created in step 1 to specify what to spawn. ***Note:*** *you can have one script spawn many different types of assets; you just need to wire them all to different variables in the script.*

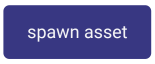
Code blocks

spawn asset

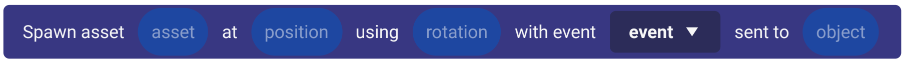
Actions ➤ Object ➤ spawn asset

Spawn a new instance of an asset into a running world. The spawn may fail if there isn't enough capacity available.

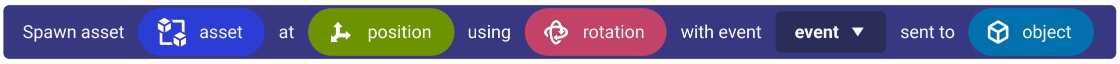
Appearance in Library


spawn asset

Appearance in Composition Pane


Spawn asset asset at position using rotation with event event ▼ sent to object

Parameters


Spawn asset asset at position using rotation with event event ▼ sent to object

asset	The asset to spawn.
position	The location the object should be at when it spawns. If the spawned object is static then it cannot be moved again from this location.
rotation	The orientation the object should have when it spawns. If the spawned object is static then it cannot be rotation again from this orientation.
event ▼	The callback event. When the object finishes spawning, this event will be sent to the object variable. The event is sent with the parameter spawned object.
object	The receiver object. When the object finishes spawning, the event ▼ will be sent to this object along with an object parameter containing the newly spawned object.

delete spawned object

Actions ➤ Object ➤ delete spawned object

Delete an object that was previously spawned, removing the objects and freeing up their capacity.

Appearance in Library

delete spawned object

Appearance in Composition Pane

delete spawned object object

Parameters

delete spawned object object

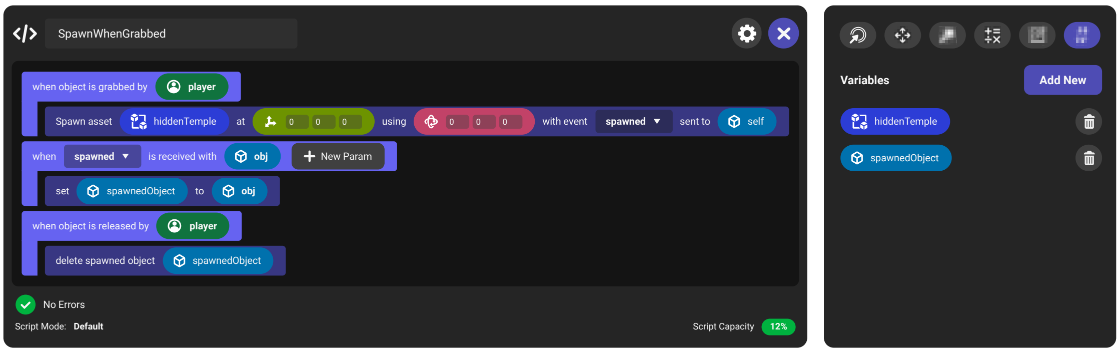
object	The object to delete. This object must have been spawned using the spawn asset code block.

Examples

Spawn an object when grabbed

What it does: This example spawns a hidden temple whenever a player grabs an object. The temple despawns if the player lets go of the object.

How it works: When a person grabs the object the script is one, we spawn the hiddenTemple asset and specify that the event spawned ▾ be sent to self when the spawn completes. When we receive the spawned ▾ event we save the spawned object in a variable, so that we can despawn it when the object is released.



SpawnWhenGrabbed

```
when object is grabbed by player
    Spawn asset hiddenTemple at 0 0 0 using 0 0 0 with event spawned ▾ sent to self
when spawned ▾ is received with obj  + New Param
    set spawnedObject to obj
when object is released by player
    delete spawned object spawnedObject
```

No Errors

Script Mode: **Default**

Script Capacity 12%

Variables

Add New

hiddenTemple

spawnedObject

Updated Assets Only Appear in New Instances

One benefit of spawning is that you can modify an asset and have it update in all the worlds that reference it. However, once a world instance starts, asset versions are frozen for that instance. Only instances of worlds created after updating the asset will see the updates. Instances that were running when the modification was made will continue to spawn in the version of the asset that existed when those instances started. Updating an asset does not require worlds that use it to be republished. Any new instances will get the updated asset.

Note that this also impacts build mode (as of June 2022). Updating the asset in build mode does not update the spawning asset in that instance of build mode. Either that instance of build mode needs to be closed by leaving build mode instance or you can create a new asset hence getting around the asset being frozen for that instance of build mode.

Example Uses of Spawning

- **Spawning environment gizmos:** In published mode spawning in environment blocks with different properties will take effect in your world as soon as they are spawned. If your world has no environment gizmos in it, you can spawn them in and out, changing the sky from “day” to “night” or turning the fog off or on. This makes it possible to create **dynamic environments**.
- **Spawning static assets:** If objects are marked as static (their motion type is “none”) when put into an asset then they will still be static when they spawn. If you spawn in a static asset that contains emissive objects, then once it spawns in, the objects will cast light into the world. This allows you to load and unload lights without using dynamic lighting. Note though that spawning takes time to happen whereas dynamic lights are instantaneous. You can also spawn in and out of sections of the world as people move around it; this allows you to simulate a much larger world (but only if the visitors stay close enough together). This makes it possible to create **procedural environments and “loading” levels**.
- **Spawning animated/dynamic assets:** When dynamic assets are spawned, the resulting objects will receive the “when world is started event” and start up as usual (e.g. fall due to gravity, auto-play an animation if “play on start” is enabled, etc. You can spawn and despawn dynamic objects as a way to create achievements (e.g. despawn a hat and spawn in a cooler one), to create variety (e.g. choose which car you want to drive or create variance in NPCs), and more. This makes it possible to create **wearable costumes, rewards, “weapon skins”, unlockable items, and so much more**.

FAQs

Q) Can I send events to spawned assets? A) Yes, but currently you will want to use a delay for any initial events right after the spawn. After that you can use any events (delayed or not) as usual.

Q) Can I receive events from spawned assets? A) Yes, once assets spawn in they act like any other object and can send and receive events. When the asset spawns, you can send events to the spawned object to connect it to other objects in the world. Likewise, the spawned object can interact with triggers, collisions, be grabbed, etc.

Q) Can I spawn assets from a non-owner or non-editor's asset library in my world? A) Yes, once the reference to the asset is made in the asset variables you can spawn in assets from anyone even if they are no longer an editor on your world.

Q) Can I spawn doors as an asset? A) Yes, but the doors will only work if the owner/editor of that world is an editor or tester in your world.

Q) Do spawned objects impact the “dynamics” gauge of the capacity meter? A) Yes, but only if the assets themselves are dynamic. Spawning and despawning assets is not like “setting visibility” on an object in a script. If an asset is marked as static then spawning it in *will not* impact the dynamics meter; if the object has interaction or physics enabled (or is a text or FX gizmo) then it will.