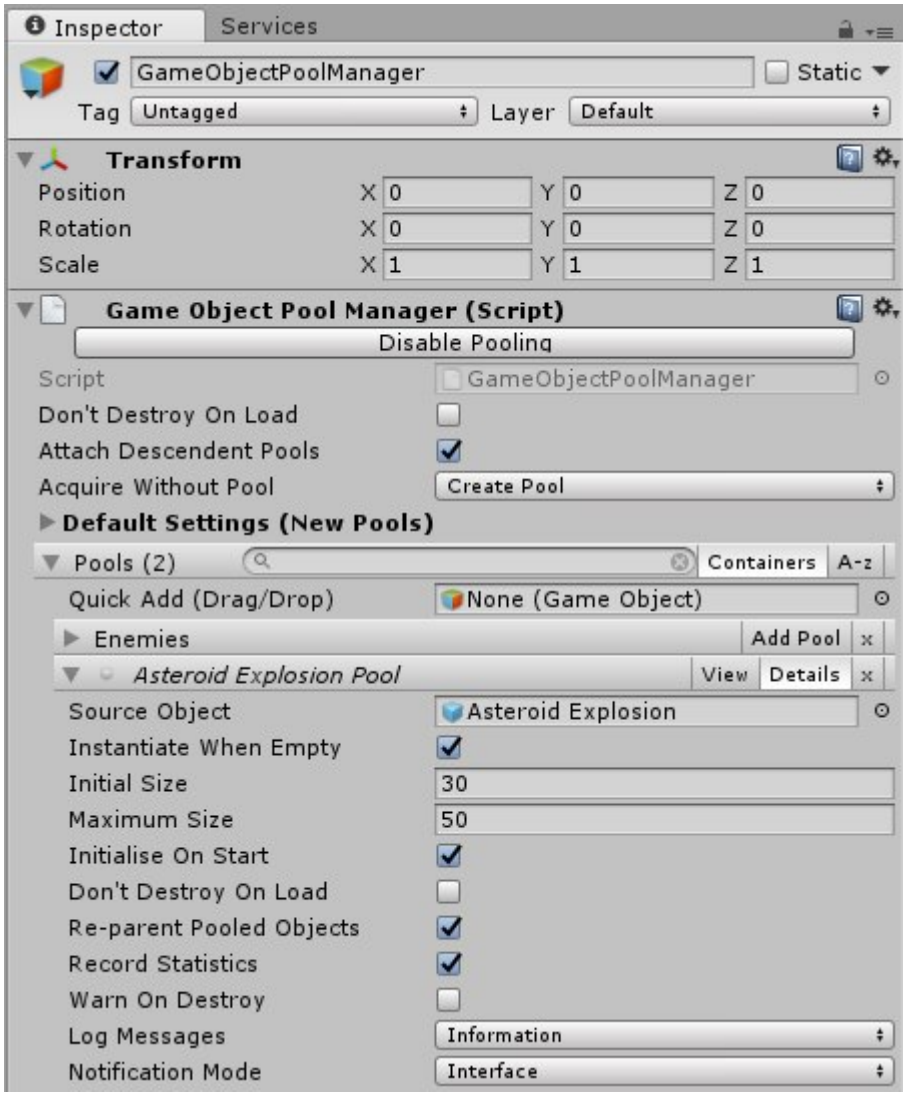
Contents

- Custom Inspectors
- GameObjectPoolManager Inspector
- GameObjectPool Inspector
- Settings foldout
- UnityEvents foldout

Custom Inspectors

To help you make the most use of Pure Pool and provide the easiest controls over your pools, several custom inspectors have been included. They customise the look and behaviour of the Inspector window when viewing the Pure Pool components.

GameObjectPoolManager Inspector



In order of appearance:

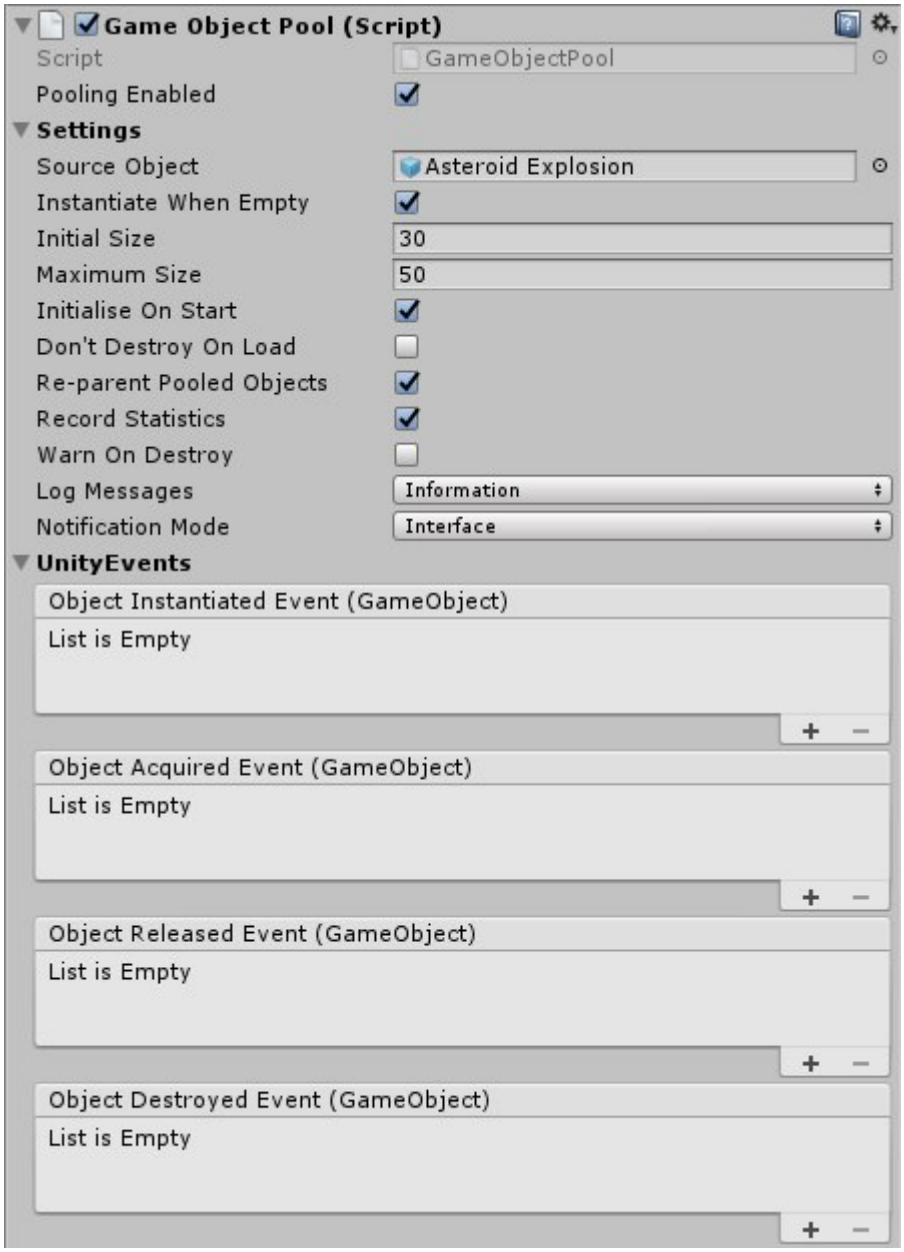
1. **Disable Pooling button** - This toggle button will disable pooling in the manager. The attached pools will all appear to be empty, and any actions taken will be based on the pools being empty. This is an easy way to check performance improvements, and locate bugs with your recycling/resetting implementations.
2. **Don't Destroy On Load checkbox** - This checkbox sets the manager's `DontDestroyOnLoad` property. When set to true, the GameObject and its parent hierarchy will be marked to remain in the scene when a new scene is loaded. When set to false, the GameObject will be destroyed when a new scene is loaded. You will often find it useful to keep the manager across all scene changes, to maximise recycling of your objects.
3. **Attach Descendent Pools checkbox** - This checkbox sets the manager's `AttachDescendentPools` property. When set to true, all `GameObjectPool` components on GameObects beneath the manager will automatically be attached to the manager. The automatic attaching only occurs during the manager's `Awake` method. Only initialised pools can be attached; any uninitialised pools will have event handlers set up to attach them when they initialise.

4. **Acquire Without Pool setting** - This setting lets you choose how to handle `Acquire` being called with a source object for which no pool currently exists. The available options are quite self-explanatory:
   - **Error** - An exception is thrown and logged in Unity. No object is returned.
   - **Instantiate** - A new object is instantiated using `Object.Instantiate`. When released, the object is destroyed.
   - **CreatePool** - A new pool is created using the manager's default pool settings (`GameObjectPoolManager.DefaultPoolSettings`), to contain instances of the object. An instance is returned from the newly-created pool.
5. **Default Settings foldout** - This foldout expands to show the default settings for new pools, from the `GameObjectPoolManager.DefaultPoolSettings` property. These settings are used when a new pool is created in the editor using the manager's inspector, as well as when a new pool is created in Play mode through code. You can also specify your own settings in code rather than using the defaults, of course.
6. **Pools Toolbar and Foldout** - The main pools toolbar also functions as a foldout, so you can hide the list of pools.
   1. The number in brackets next to the word "Pools" is the number of `GameObjectPool` components currently attached to the manager. When the list of pools is filtered using the search box, the number in brackets becomes two numbers, for example "Pools (7/20)". The first number is the number of pools that matched the search filter, while the second is the total number of attached pools.
   2. The search box allows the use of simple regular expressions, so a search for ***_Death** will find all pools whose name end with **_Death**. The search filter looks at the pool name as well as the names of all parent GameObjects, so you can search by container too.
   3. The **Containers button** toggles whether containers are displayed in the list of pools. Containers are simply the GameObjects that are parented beneath the manager. When you create a pool in the manager, a new GameObject is created beneath the manager with the `GameObjectPool` component attached, which you can see in the Hierarchy window. To help you organise your pools, you can create new GameObjects beneath the manager, and parent your pools to them. For example, you could have the following hierarchy of GameObjects beneath your manager: **GameObjectPoolManager > Enemies > Slime Pool**. In this way, you can have all your enemy prefabs visible under one heading. Feel free to nest the containers as much as you like!
   4. The **A - z button** toggles whether the pools are sorted alphabetically, or whether they appear in the order that they're in on the Hierarchy.
7. **Quick Add (Drag/Drop) field** - This field allows you to quickly create new pools. Simply drag-and-drop the source object (the object you wish to pool) to the field area on the right. You can also click the small circle to the right of the field, which brings up a list of objects in your project, and find the object you wish to add in the list. You can also drag-and-drop prefabs to container toolbars and pool toolbars. If you drag a prefab to a container toolbar, a new pool will be created using the prefab within that container. If you drag a prefab to a pool toolbar, the existing source object will be replaced with the dragged prefab.
8. **Enemies container** - Containers are displayed as toolbars in the inspector, and are purely

for organisation. Clicking the **Add Pool button** brings up a window to choose which source item to pool, and when an item has been chosen, a new pool is created beneath the container. The **X button** deletes the container and any pools or other containers beneath it. Context-clicking on the container (usually the right mouse button) brings up a context menu, with options to create a new container, expand all, or collapse all. Expanding and collapsing applies to all containers and pools beneath the container you clicked on, including the clicked container.

9. **Asteroid Explosion Pool toolbar** - Pools are displayed as toolbars in the inspector too. An icon on the left side of the toolbar indicates the current state of the pool: grey for uninitialised, yellow for initialised but not attached to the manager, blue for initialised and attached but not enabled, and green for initialised, attached, and enabled. The name of the pool in the toolbar is taken from the name of the pool's GameObject. To rename the pool, just rename the pool's GameObject in the Hierarchy window. The **View button** selects the pool's GameObject in the Hierarchy, so that the pool itself appears in the inspector window. The **Details button** toggles whether advanced pool settings are displayed, or whether they're hidden to save space. The **X button** deletes both the pool and its GameObject. Context-clicking on the pool toolbar (usually the right mouse button) brings up a context menu, with options to show the definition, show statistics, and show details. The definition of the pool is the settings that the pool had at the time it was initialised. This is useful to compare the initial settings with any changes that you've made in Play mode. The statistics show general operational data about the pool.

GameObjectPool Inspector



Settings foldout

1. **Pooling Enabled checkbox** - This checkbox will enable or disable pooling. When disabled, the pool will appear to be empty, and any actions taken will be based on the pool being empty. This is an easy way to check performance improvements, and locate bugs with your recycling/resetting implementations.
2. **Source Object field** - This field lets you choose the object to be pooled (the source object) by setting the pool's `SourceObject` property. You can drag a GameObject or prefab to the field, or click on the circle to the right to choose from a list.
3. **Instantiate When Empty checkbox** - This checkbox sets the pool's `InstantiateWhenEmpty` property. If an attempt is made to acquire an object from the pool while it's empty (or while pooling is disabled), this setting determines whether a new object will be instantiated using `Object.Instantiate`, or whether the attempt will fail.

4. **Initial Size field** - This field sets the pool's `InitialSize` property. It's only visible while the pool is uninitialised. The initial size is the number of objects to populate the pool with, when the pool is initialised. If InitialSize is set to 50, then 50 objects will be created in the pool when the pool is initialised.
5. **Maximum Size field** - This field sets the pool's `MaximumSize` property. The maximum size limits the number of objects that can exist in the pool. This can help reduce overall memory usage, as every pooled item is using memory. If an object is released to the pool while it has the maximum number of objects, the released object will be destroyed using `Object.Destroy`.
6. **Initialise On Start checkbox** - This checkbox sets the pool's `InitialiseOnStart` property. It's only visible while the pool is uninitialised. It determines whether the pool will be initialised in the pool's Start method. If this setting is unchecked, you will have to manually initialise the pool by calling the `Initialise` method.
7. **Don't Destroy On Load checkbox** - This checkbox sets the pool's `DontDestroyOnLoad` property. When set to true, the GameObject and its parent hierarchy will be marked to remain in the scene when a new scene is loaded. When set to false, the GameObject will be destroyed when a new scene is loaded. You will often find it useful to keep the pool across all scene changes, to maximise recycling of your objects.
8. **Re-parent Pooled Objects checkbox** - This checkbox sets the pool's `ReparentPooledObjects` property. It determines whether the objects in the pool are parented beneath the pool in the Hierarchy. While reparenting helps reduce clutter in the hierarchy and can make it easier to test your scenes, it does come with a performance cost. Ideally it should be disabled in your final release, but be aware that the objects will remain parented where they were being used, and you may end up accidentally destroying them - you should take care to ensure this does not cause any issues.
9. **Record Statistics checkbox** - This checkbox sets the pool's `RecordStatistics` property. It determines whether general operational statistics about the pool should be recorded. The statistics can be accessed through the pool's `Statistics` property.
10. **Warn On Destroy checkbox** - This checkbox sets the pool's `WarnOnDestroy` property. It determines whether a warning will be logged when a pooled object is being destroyed. In general, you should avoid destroying pooled objects, whether they are still inside the pool or not. Unfortunately, scene changes may also cause pooled objects to be destroyed. In this case, the warning message will be shown incorrectly, and can safely be ignored.
11. **Log Messages setting** - This setting sets the pool's `LogMessages` property, and lets you choose the level of log messaging that the pool will output. The available options are quite self-explanatory:
    - **Off** - Logging is disabled. No messages will be shown.
    - **Information** - Informational messages, warning messages and error messages are displayed.
    - **Warning** - Warning messages and error messages are displayed. Informational messages are not shown.
    - **Error** - Only error messages are displayed. Informational and warning messages are not shown.

12. **Notification Mode setting** - This settings sets the pool's `NotificationMode` property, and determines the way in which notifications are sent to the pooled objects. The notifications consist of a message when the object is acquired from the pool, and a message when the object is returned to the pool, and allow the implementation of a resetting mechanism, to restore the object's original state.

UnityEvents foldout

1. **Object Instantiated Event** - This event occurs when a new object is instantiated by the pool. This may occur when the size is increased, or when the pool runs out of objects if `InstantiateWhenEmpty` is true.
2. **Object Acquired Event** - This event occurs when an object is acquired from the pool. The object may have been sitting in the pool, or may have been instantiated if the pool was empty and `InstantiateWhenEmpty` is true.
3. **Object Released Event** - This event occurs when an object is released back into the pool.
4. **Object Destroyed Event** - This event occurs when an object is destroyed by the pool, using `Object.Destroy`. This may occur when the size is decreased, or when an object is released to the pool while it contains the maximum number of objects.