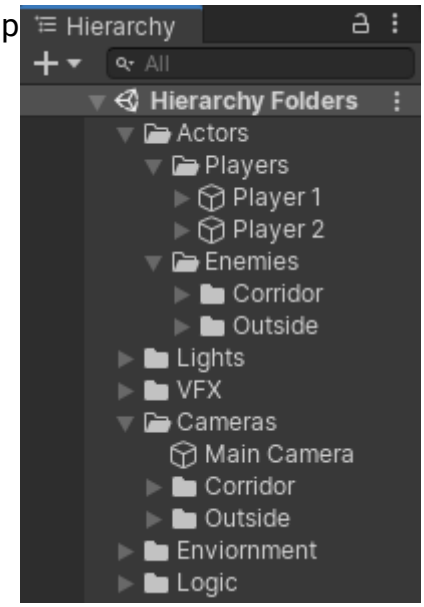
Documentation

What Is a Hierarchy Folder?



A **hierarchy folder** is an **editor-only GameObject** meant to help in **organizing** the **GameObjects** in your scenes.

Technically speaking it can be seen as a GameObject that contains the HierarchyFolder component.

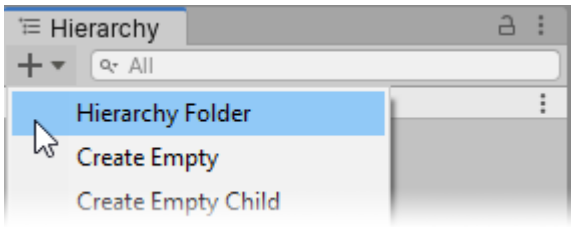
In practice though, it is more intuitive to think of it as a whole new GameObject type, because it acts in many ways very differently from your typical GameObjects:

1. A hierarchy folder **only exists in the editor**. When you make a build, all hierarchy folder GameObjects are removed from every scene included in the build. All GameObjects that were nested under the hierarchy folders are pushed up the parent chain until they are the child of a GameObject that isn't a hierarchy folder, or they exist in the root of the scene hierarchy.
2. You **cannot add components** to a hierarchy folder. The moment you do, the GameObject seizes to be a hierarchy folder.
3. For all intents and purposes a hierarchy folder has **no transform component**. If you look at a hierarchy folder in the inspector, you won't see a transform component. If you try to move, rotate or scale a hierarchy folder, it just immediately reverts back to its previous state (without affecting the world space state of any of its children).

Creating New Hierarchy Folders

There are various ways you can use to create new hierarchy folders.

1. Hierarchy View Create Menu

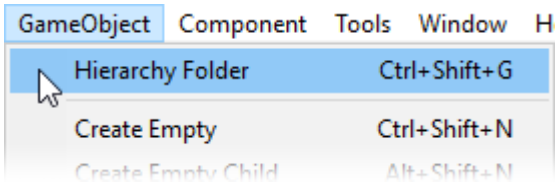


To create a new hierarchy folder into the loaded scene, click the plus button found on the Hierarchy view toolbar and select the item **Hierarchy Folder**. This will add a new hierarchy folder into the active scene.

If you have one target selected in the hierarchy, the new hierarchy folder will be created above it in the hierarchy.

If you have more than one target selected in the hierarchy, all of the selected objects will be grouped under the hierarchy folder that is created.

2. GameObject Menu



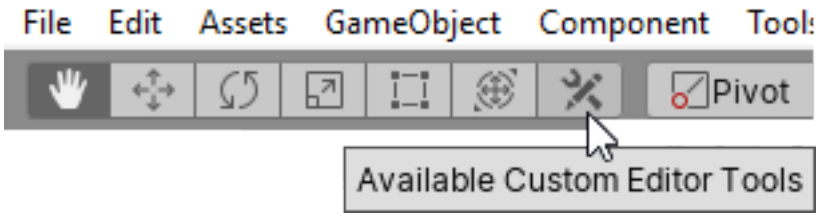
You can also create a new hierarchy folder using the main menu item **GameObject > Hierarchy Folder**, or the shortcut key **Ctrl + Shift + G**.

3. Editor Tools Toolbar

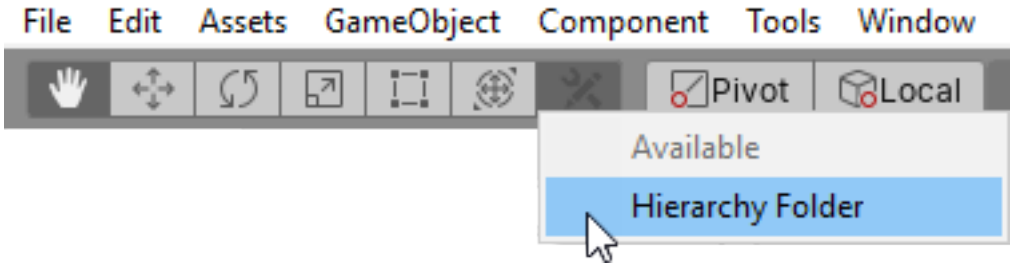
A third way you can create new hierarchy folders is by clicking the button found on the **Editor tools toolbar**.

In order for the button to appear in your toolbar you have to activate it first. Here are the steps needed to do this:

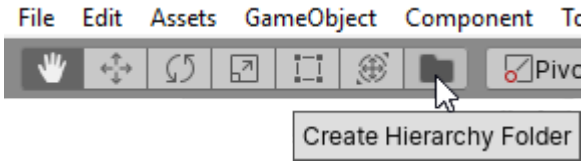
1. Click the **Custom Editor Tools button** found on the Editor Tools toolbar.



2. Select the **Hierarchy Folder** item to set it activate.



After this the **Create Hierarchy Folder** button appear in the toolbar. New hierarchy folders can be created by clicking this button.



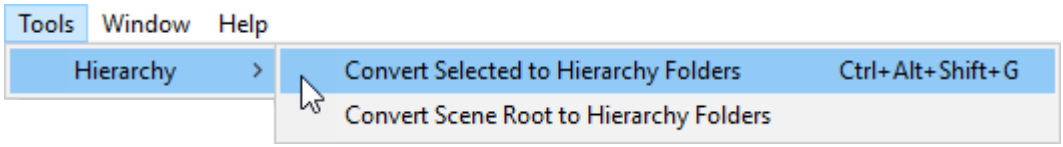
Converting GameObjects Into Hierarchy Folders

In addition to creating new hierarchy folders from scratch, it is also possible to take existing GameObjects and convert them into hierarchy folders.

This can be useful if you have a project that contains scene hierarchies where you have grouped your objects under empty GameObjects, which you would now like to convert into hierarchy folders.

There are various methods you can use to convert empty GameObjects into hierarchy folders.

1. Tools Menu



You can convert empty GameObjects into a hierarchy folder using the menu items found in the main menu under **Tools > Hierarchy**.

1.1. Convert Selected

To convert one or more empty GameObjects into hierarchy folders, select them in the hierarchy view and then select the menu item **Tools > Hierarchy > Convert Selected to Hierarchy Folders**.

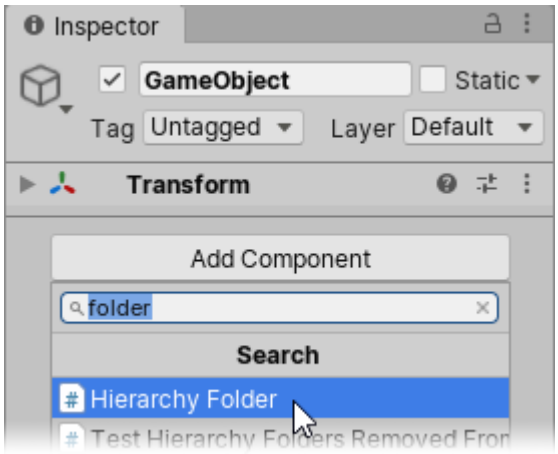
You can also do this by using the keyboard shortcut **Ctrl + Alt + Shift + G**.

1.2. Convert Entire Scene

You can also automatically convert all root GameObjects in the active scene into hierarchy folders by selecting the menu item **Tools > Hierarchy > Convert Scene Root to Hierarchy Folders**.

This will turn all empty GameObjects in the root of the active scene into hierarchy folders, as well as group all GameObjects in the root that contain components under newly created hierarchy folders.

2. HierarchyFolder Component



You can also turn any empty GameObject into a hierarchy folder by **adding the HierarchyFolder component** to it.

When you do this, the **Transform** component of the GameObject will be reset to its **default state**. This will be done in such a way that the world space positions of child GameObjects will not be affected in any way.

This method can be especially useful if you want to **programmatically** convert a GameObject into a hierarchy folder from a script.

Hierarchy View Enhancements

1. Folder Icons

By default all hierarchy folders will have a unique icon in the hierarchy view to clearly distinguish them from normal GameObjects.

You can customize these icons or turn them off completely in the preferences view.

2. Select All Children

When you **double-click** a **normal GameObject** in the hierarchy view it causes the Scene view camera to focus on that GameObject.

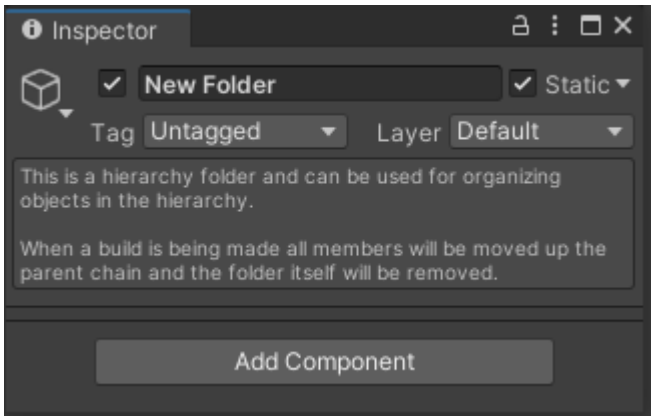
Hierarchy folders however are **positionless** entities and as such the same behaviour would not make sense when they are double-clicked.

Instead when you double-click hierachy folder the following things happen:

1. The **hierarchy folder is unfolded** in the hierarchy view.
2. All the **child GameObjects** and hierarchy folders inside the double-clicked hierarchy folder are **folded** in the hierarchy view.
3. All the **child GameObjects** and hierarchy folders inside the double-clicked hierarchy folder are **selected** in the hierarchy view.

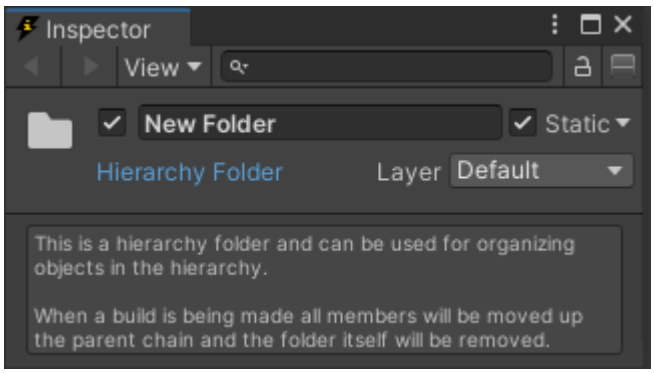
Inspector Integration

The inspector for hierarchy folder is clearly distinguished from that of normal GameObjects to guide the user on how hierarchy folders function and which values on them can be modified.



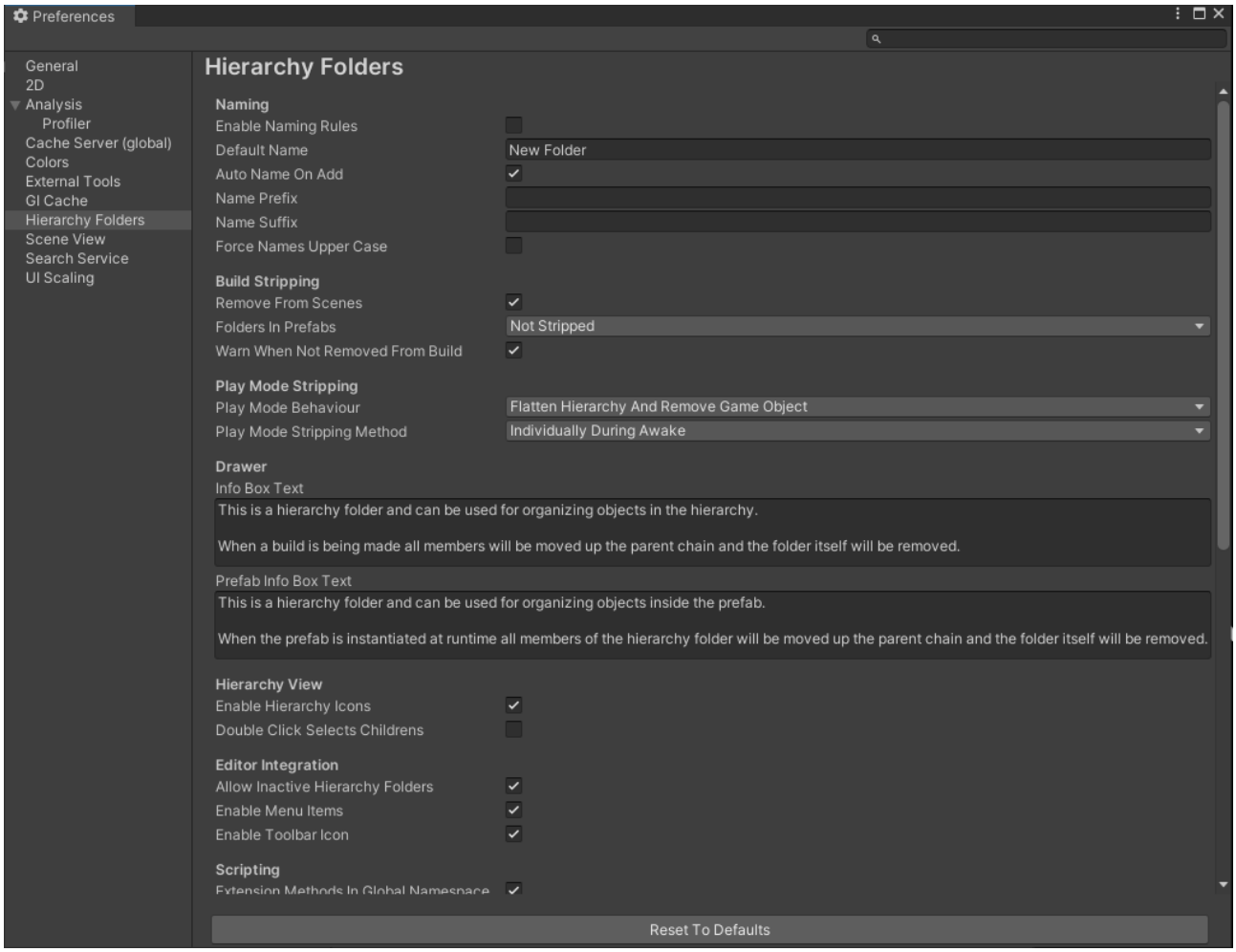
1. **Info text box** – A convenient info text box will be shown in the inspector clearly explaining what the hierarchy folder. This text can be customized in the preferences.
2. **No transform component** – The transform component has been hidden to prevent users from making any modifications to them.

If you are using Power Inspector then the inspector will also contain some additional improvements on top of the ones already listed.



3. **Hierarchy Folder Icon** – A folder icon will be shown instead of the GameObject icon in the inspector.
4. **Hierarchy Folder label** – The label ”Hierarchy Folder” will be shown on the inspector header.
5. **No tag field** – The tag field has been removed from the inspector to discourage users from trying to find hierarchy folders by their tag – since they don’t exist in builds this could lead to bugs.
6. **No Add Component button** – The add component button has been removed from the inspector completely, since hierarchy folders cannot contain components.

Customizing Hierarchy Folders



You can customize many aspects of hierarchy folders to fit the specific needs of your project from the preferences view.

To open the hierarchy folders preferences view select the main menu item

Edit > Preferences... and activate the **Hierarchy Folders** view from the side bar in the window that opens.

IMPORTANT NOTE: Always press the **Apply** button when you want to apply the changes you have made in the preferences. Otherwise the changes will not be saved.

The following preferences are available for customization in the view...

Naming

- **Enable Naming Rules** – Should all naming rules such as prefix, suffix and upper casing be in effect?
- **Default Name** – The initial name given to new hierarchy folders when created through the create menu, main menu or editor tools toolbar.
- **Auto Name on Add** – If true, then the name of GameObjects with a default names (”GameObject”, ”GameObject (1)” etc.) will be changed to match the value of the Default Name preference item, when a HierarchyFolder component is added to them.
- **Name Prefix** – A prefix with which the name of all hierarchy folder GameObjects should begin.
    - **WARNING**: if you change the value of this item, the old prefixes will currently not be automatically removed before the new ones are added. So if you change it to ”===” from ”- - -” , then all your existing hierarchy folders will get the prefix ”===- - -”.
- **Name Suffix** – A suffix with which the name of all hierarchy folder GameObjects should end.
    - **WARNING**: if you change the value of this item, the old suffixes will currently not be automatically removed before the new ones are added. So if you change it to ”===” from ”- - -” , then all your existing hierarchy folders will get the suffix ”- - - ===”.
- **Force Names Upper Case** – If true, then the names of all hierarchy folder GameObjects will use to upper case lettering.
    - **WARNING**: if you activate this preference item, its changes cannot be automatically reverted. This means that if you want to change back, you will have to manually rename all affected hierarchy folders to not use all upper case letters.

Build Stripping

- **Remove From Scenes** - If true then all hierarchy folders will be removed from all scenes during the build process. All child GameObjects will be moved upwards the parent chain until they are the child of a normal GameObject or exist in the root of the hierarchy.
    - **NOTE:** If this false, then the hierarchy folder GameObjects will survive the build process, and none of their child GameObjects will be unparented. However in builds the hierarchy folders will lose all of their special properties, and essentially becomes just normal GameObjects. This means that their transform states will no longer be locked, and they won't react to new components being added on the GameObject.
- **Folders In Prefabs** – This item determines whether or not hierarchy folders inside prefabs and prefab instances are supported and if so how they are stripped from builds.
    - **Not Allowed**: Hierarchy Folders are not allowed to exist inside prefabs or prefab instances at all.
    - **Stripped At Runtime**: Hierarchy Folders inside prefabs will be stripped at runtime during instantiation. This takes place before any other code on the prefabs executes.
        - **NOTE**: If your prefabs contain inactive hierarchy folders, they will not be stripped at runtime during instantiation, but only when they become active.
    - **Stripped At Build Time**: Hierarchy Folders inside prefabs will be stripped at build time.
        - WARNING**:** Note that selecting this method can make the build process take substantiatilly longer because **all prefabs** in the project need to checked for hierarchy folders and all prefabs containing hierarchy folders need to be backed up, stripped and finally restored to their original state once the build has finished.
        - **NOTE:** hierarchy folders at the root of a prefab can not be stripped at build time, so runtime stripping will still be utilized for them even when this setting is selected.
    - **Not Stripped**: Hierarchy Folders can be placed inside prefabs yet they will not be stripped at any point.
        - WARNING**:** this will result in sub-optimal performance and is not recommended for anything other than benchmarking reasons.

- **Warn When Not Removed From Build –** Determines whether or not a popup should appear to warn you when you attempt to make a build with the preference item Remove From Build disabled.

Play Mode Stripping

- **Play Mode Behaviour -** This item determines how hierarchy folders will operate when play mode is active in the Unity editor.
    - **None** – Hierarchy folders are not removed when entering play mode, nor are any of their child GameObjects unparented. This means that the behaviour is identical between play mode and edit mode. Selecting this can negatively affect performance in play mode.
    - **Flatten Hierarchy -** All the child GameObjects of hierarchy folders are moved up the parent chain. The hierarchy folders themselves are not destroyed, but will remain in the root of the scene as visual dividers between your GameObjects. Any GameObjects that are grouped under hierachy folders in play mode will automatically get detached and moved below them it in the hierarchy. This negates most of the negative performance impact that hierarchy folders normally would have in play mode, but the reparenting itself and the checking if reparenting needs to be done can have some overhead.
    - **Disable Component** – All the hierarchy folder components are disabled when entering play mode. All the child GameObjects of hierarchy folders will remain grouped under these disabled hierarchy folders. GameObjects that are grouped under the hierarchy folders in play mode also will not automatically get detached.
    - **Destroy Component** – All the child GameObjects of hierarchy folders are moved up the parent chain, and the hierarchy folders themselves are converted into normal GameObjects. The GameObjects that used to be hierarchy folders however are not destroyed, but will remain in the root of the scene as visual dividers between your GameObjects.
    - **Flatten Hierarchy And Disable Component** – All the child GameObjects of hierarchy folders are moved up the parent chain, and the hierarchy folders themselves are disabled. The hierarchy folders are not destroyed, but will remain in the root of the scene as visual dividers between your GameObjects. Any GameObjects that are grouped under these disabled hierarchy folders in play mode will not automatically get detached.
    - **Flatten Hierarchy And Destroy Component** – All the child GameObjects of hierarchy folders are moved up the parent chain, and the hierarchy folders themselves are converted into normal GameObjects. The GameObjects that

used to be hierarchy folders however are not destroyed, but will remain in the root of the scene as visual dividers between your GameObjects.

- **Whole GameObject** – All the child GameObjects of hierarchy folders are moved up the parent chain, and the hierarchy folder GameObjects are completely removed from the scene. This means that the scene hierarchy will be identical to the way it is in the final build.

Drawer

- **Info Box Text** – Text that will be shown in the inspector when a hierarchy folder is selected in the scene hierarchy.
- **Prefab Info Box Text** - Text that will be shown in the inspector when a hierarchy folder is selected inside a prefab.

Hierarchy View

- **Enable Hierarchy Icons** – If true, then all hierarchy folders will get a nice folder icon in the hierarchy view, distinguishing them from other GameObjects.
- **Double Click Selects Children** – If true then double-clicking a hierarchy folder in the hierarchy view will unfold it, select all of its child GameObjects and fold all of its child GameObjects. If false then default double-click behaviour – focusing the Scene View camera on the GameObject – will be used.

Editor Integration

- **Allow Inactive Hierarchy Folders**
  - If false then hierarchy folders can not be set inactive. The active tick box in the inspector for hierarchy folders will be changed to toggle the active state of child GameObjects instead. If true then hierarchy folders
  - If true then hierarchy folders can be set inactive. Note that unless you have disabled build stripping, this will have no practical effect in builds, as inactive hierarchy folders will still be stripped from builds.
- **Enable Menu Items** – If you do not want the item **GameObject > Hierarchy Folder** and the items under **Tools > Hierarchy** to exist in your main menu, you can disable them by setting this item to false.
- **Enable Toolbar Icon** – If you do not want the **Create Hierarchy Folder** button to appear as on option for your Editor Tools toolbar, you can disable it by setting this item to false.

Scripting

- **Extension Methods In Global Namespace** – Determines whether or not hierarchy folders related extension methods such as GameObject.IsHierarchyFolder and Transform.GetRoot should always be shown, or only when you add ”using Sisus.HierarchyFolders;” at the top of your class.
  - **NOTE**: you will also need an Assembly Definition File with a reference to Sisus.HierarchyFolders for the extension methods to become available in your classes.

Icons

You can customize the icons used by hierarchy folders in the hierarchy view for the four different skins used by Unity. If no icon is specified, then a default icon will be used instead.

- **Modern Light** – Icons used in unity versions 2019.3 and higher when using the personal editor theme.
- **Modern Dark** – Icons used in unity versions 2019.3 and higher when using the professional editor theme.
- **Classic Light** – Icons used in unity versions older than 2019.3 when using the personal editor theme.
- **Classic Dark** – Icons used in unity versions older than 2019.3 when using the professional editor theme.

In addition, if support for hierarchy folders inside prefabs has been turned on, then an additional selection of icons can be used to customize the icons seen for these in the hierarchy view in different contexts.

- **Prefab *** - Icons used hierarchy folder prefab instances in the scene hierarchy.
- **Prefab Addition *** – Icons used for hierarchy folder prefab instances when they have been added under other prefabs instances in the scene hierarchy.
- **GameObject Addition *** - Icons used for hierarchy folders which are not prefab instances, when they have been added under prefabs instances in the scene hierarchy.
- **Prefab Variant *** - Icons used for hierarchy folder prefab variant instances.
- **Prefab Variant Addition *** - Icons used for hierarchy folder prefab variant instances when they have been added under other other prefab instances in the scene hierarchy.