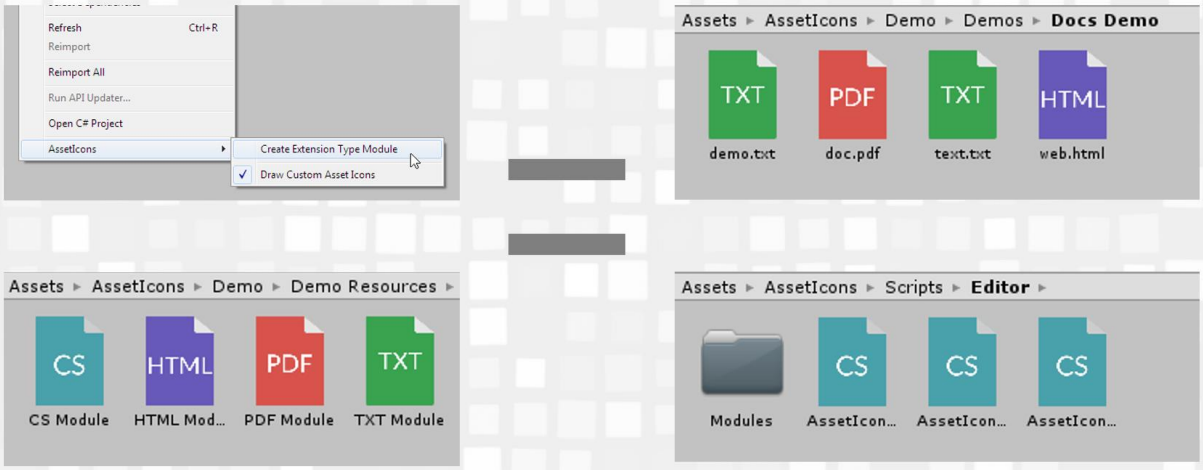
AssetIcon

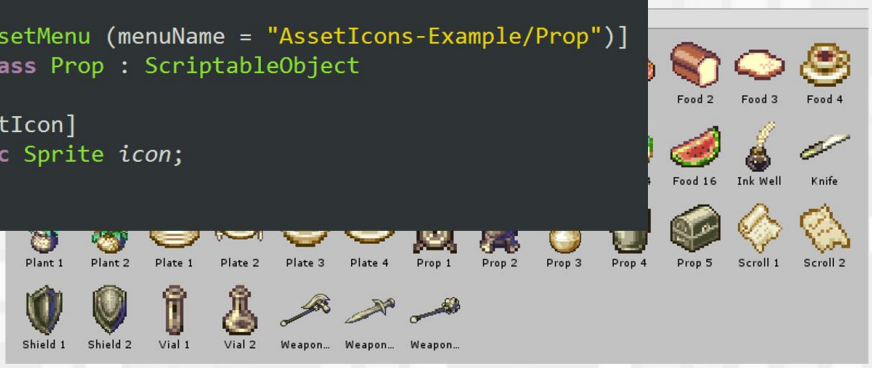
Dynamic ScriptableObject Icons with a Single Attribute

Custom File Icons



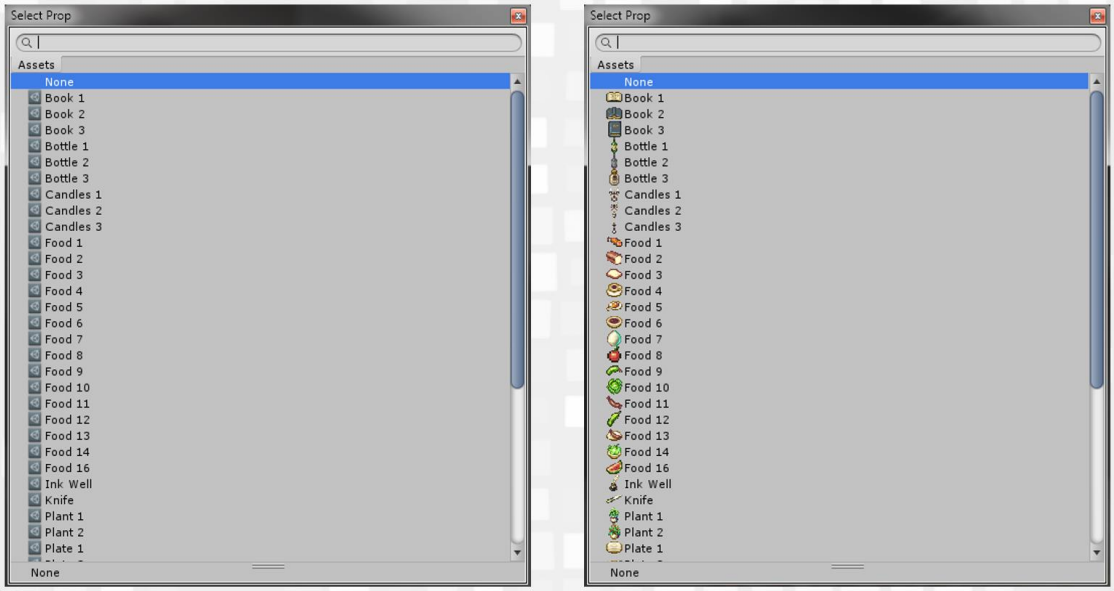
With a Single Attribute

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  namespace AssetIcons.Example
6  {
7      [CreateAssetMenu (menuName = "AssetIcons-Example/Prop")]
8      public class Prop : ScriptableObject
9      {
10         [AssetIcon]
11         public Sprite icon;
12     }
13 }
```

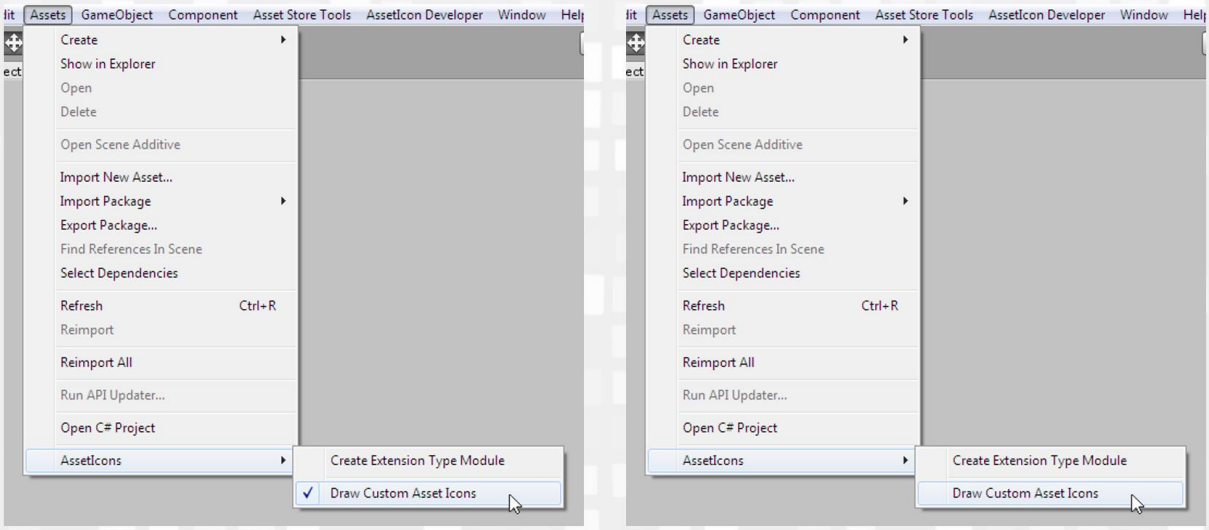


(2017.1 ONWARDS)

WITH OBJECT SELECTOR SUPPORT



ENABLE AND DISABLE WITH EASE



AssetIcon Usage Guide

How to use the AssetIcon Attribute

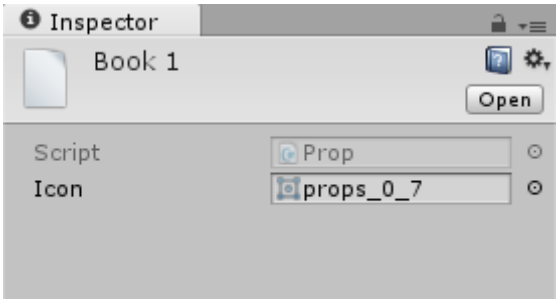
It's simple, just insert the below line of code before and Sprite or Texture2D variable, method or property.

```
[AssetIcon]
```

And that's it. You're done.

```
[CreateAssetMenu (menuName = "AssetIcons-Example/Prop")]
public class Prop : ScriptableObject
{
    [AssetIcon]
    public Sprite icon;
}
```

Now simply navigate to an asset of that ScriptableObject type and assigned an icon in the field. If you are using a property or method ensure that it returns an icon properly. You should now see that your assets have icons.

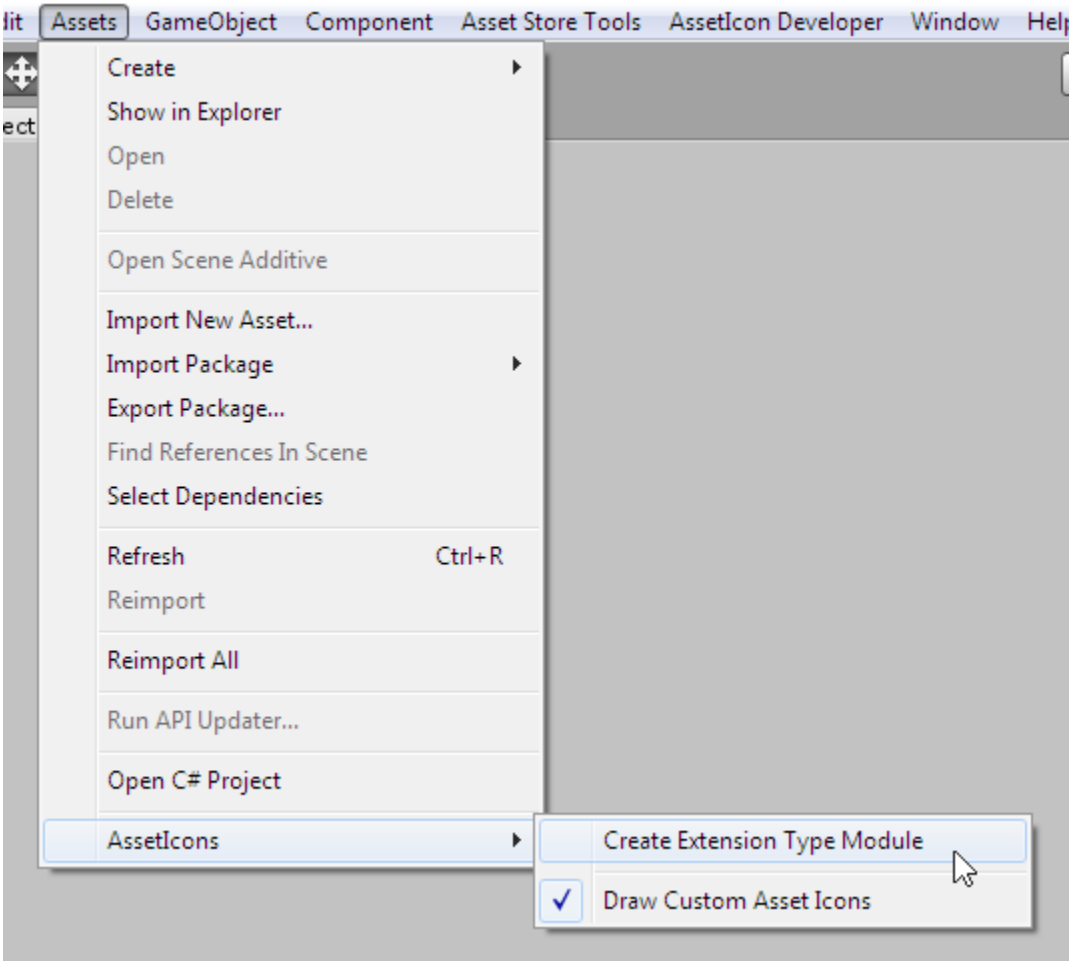


Anthony Marmont

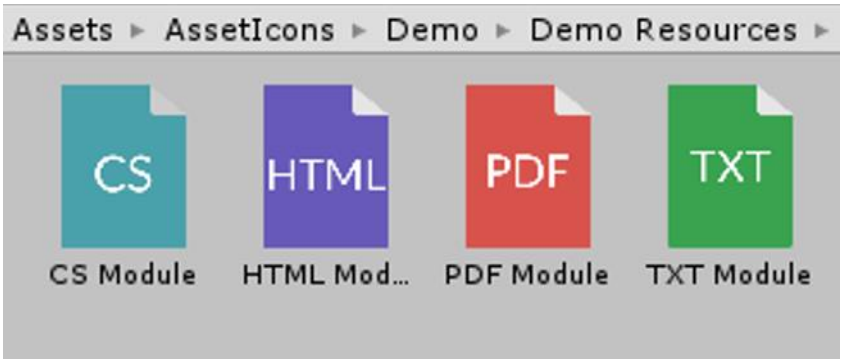
How to draw Custom File Type Icons

To draw custom icons on assets dependant on their file type (.txt, .html, e.t.c), navigate to "AssetIcons/Create Extension Type Module", to create a new Extension Type Module.

These can be located anywhere in your project, however they have to be in an "Editor" and "Resources" directory for it to be findable by the asset and not to be included in the build. By default they are created in the AssetIcons directory.



Below are the demo modules – their icons resemble how the assets with the extensions draw, however it might be a bit confusing that the module and target types have the same icon. I plan to change this in the future.

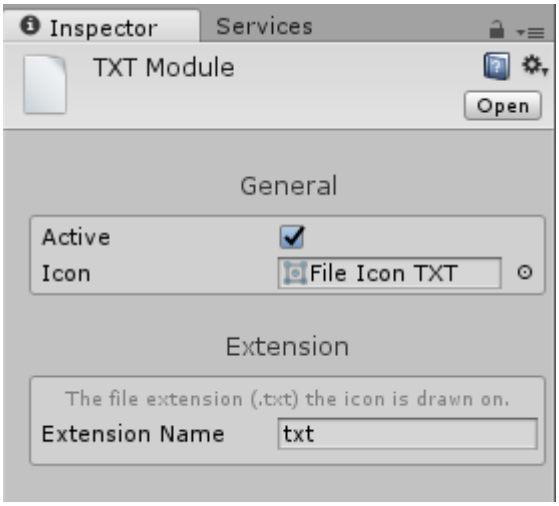


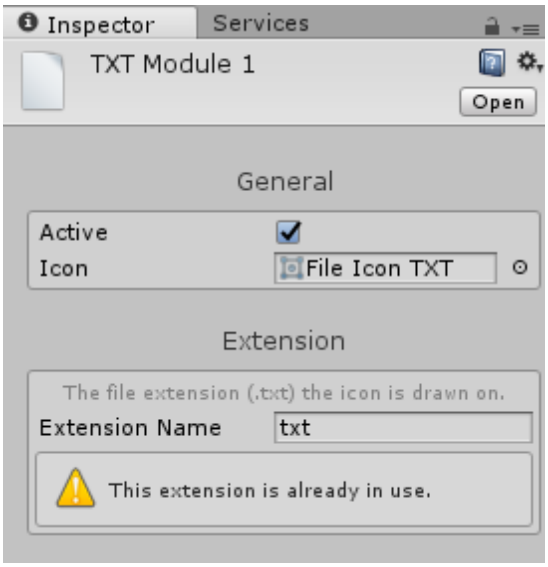
Using my custom editor, setting up the module is simple. Simply ensure that the Active checkbox is checked, an icon is assigned and that the extension name is set to a valid name. The custom inspector will warn you if that extension name is already in use, and it will also warn you if it's an invalid extension name.

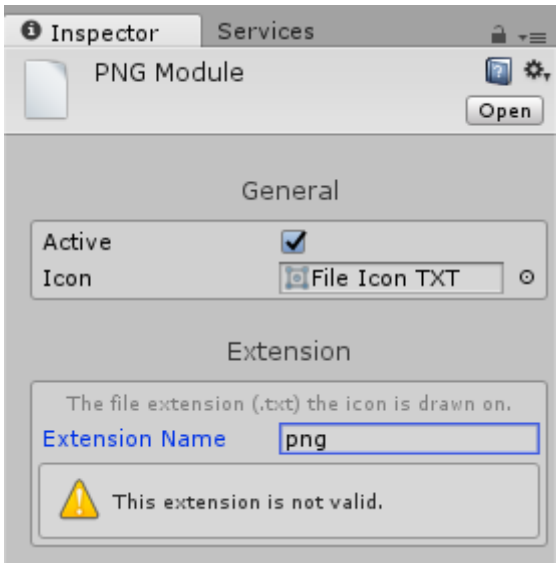
Invalid types are as follows: "png", "psd", "tiff", "jpg", "tga", "gif", "bmp", "iff", "pict" and "asset".

The images are invalid because my asset doesn't properly mask out the background of the image, and "asset" is invalid because I want to minimize confusion for creating dynamic icons – however you can edit ExtensionTypeModule.cs and remove the invalid types from line 20 and it will draw fine on all of them. Using an "asset" type override will become a fallback icon for the ScriptableObjects if it doesn't utilize the [AssetIcon] attribute.

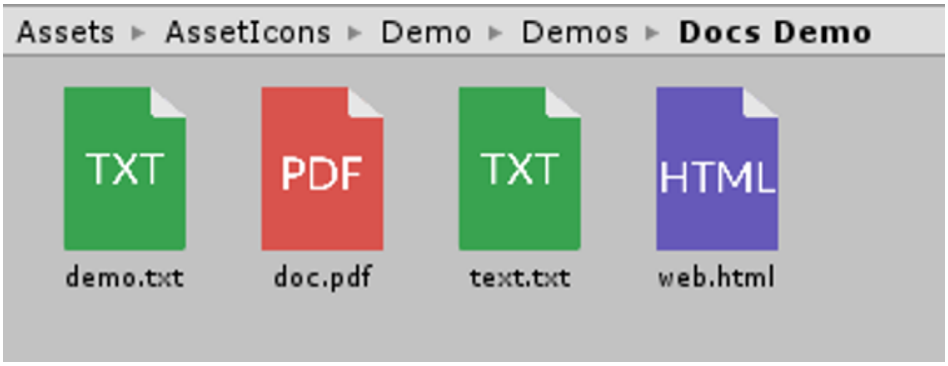
You can include or exclude the dot/period, but either way the editor omits it when you click off of the field.



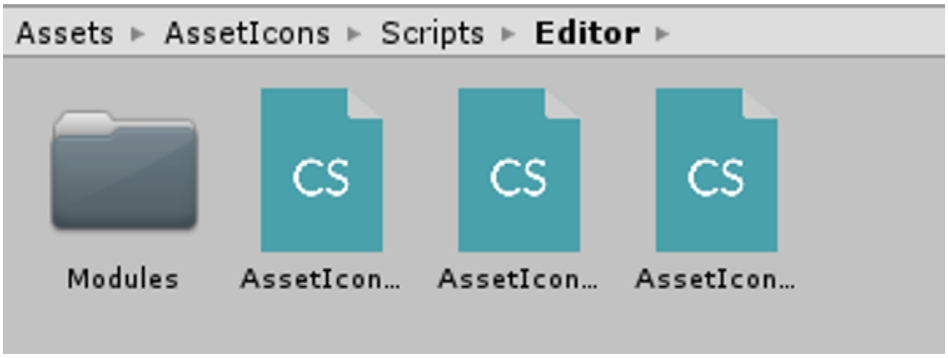




Below is the finished product of setting up the modules. As you can see, the assets have a unique icon depending on their extension.

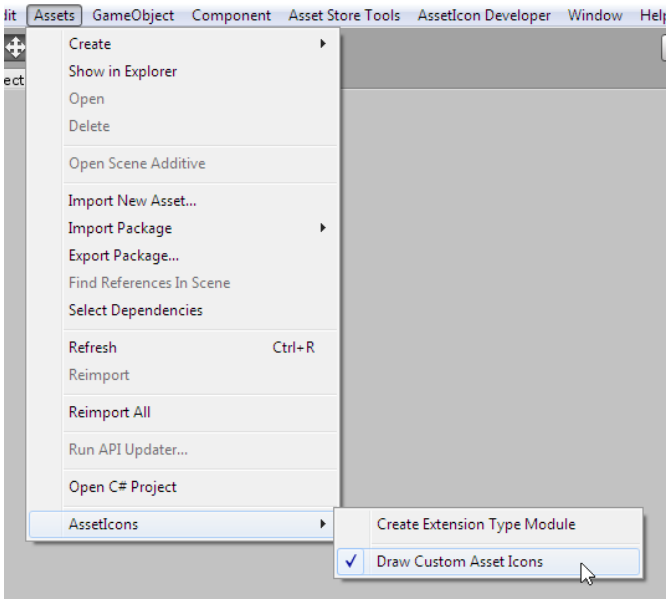


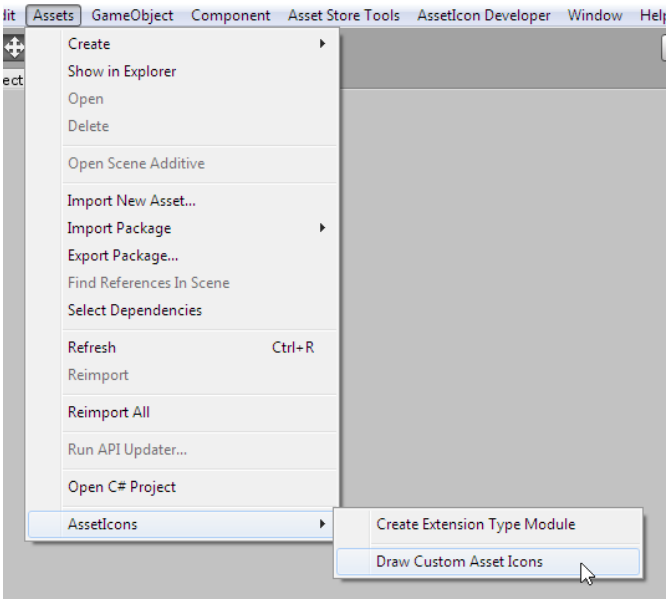
This also works on ".cs" and ".js" file types, so you can override what they look like as well. Below is the result of this in action – however I disabled the "CS Module" by default because not everyone wants their script icons being tampered with.



WANT TO DISABLE ASSETICON?

It's as simple as navigating to the assets dropdown in Unity, navigating under "AssetIcons", and clicking on the "Draw Asset Icons" toggle. You should see that the custom asset icons are no longer being drawn, and returning to this menu will show a toggled menu button.





AssetIcon Documentation

Dynamic ScriptableObject asset icons with a single attribute.

Context

AssetIconAttribute

class in **global** / Inherits from: **UnityEngine.PropertyAttribute**

Description

Marks a member of a ScriptableObject to be used as the source for the asset in the project windows icon.

Usage

```
[AssetIcon]
```

Can be used inside objects which inherit from:

- ScriptableObject

Can be placed on members with the following modifiers:

- public
- private
- internal
- protected
- static
- virtual
- override
- readonly

Can be used on members of the following types:

- Field
- Property
- Method

Can be used on members which return the following types:

- Texture2D
- Sprite

AssetIconAttributeDrawer

class in **AssetIcons.Editors** / Inherits from: **UnityEditor.PropertyDrawer**

Description

Draws the AssetIcon property, which repaints the project window when the icon is changed. Allows for instant feedback in the project window when changing the icon on a ScriptableObject.

Usage

Draws on all fields marked:

```
[AssetIcon]
```

Public Functions

`GetPropertyHeight`	Used by the Unity Edtior to know the height of the property to draw.
`OnGUI`	Called by the Unity Edtior, where in the property is drawn.

AssetIconDrawer

class in **AssetIcons.Editors**

Description

Used to draw custom icons in the project window. Contains multiple methods for doing so, using multiple types of images.

Public Functions

`DrawCustomIcon`	Draws an image in the given Rect using the supplied graphic.

DrawCustomIcon

method in **AssetIconDrawer**

---

public static void **DrawCustomIcon** (string **guid**, Rect **rect**, Sprite **sprite**)

Parameters

`guid`	The guid of the asset to be drawn.
`rect`	The area in which the sprite should be drawn.
`sprite`	The graphics which should be drawn.

Description

Draws an icon inside the provided rect using the provided sprite. If the provided guid is selected and selection tint is enabled, the sprite will be tinted.

---

public static void **DrawCustomIcon** (string **guid**, Rect **rect**, Texture **texture**)

Parameters

`guid`	The guid of the asset to be drawn.
`rect`	The area in which the texture should be drawn.
`texture`	The graphics which should be drawn.

Description

Draws an icon inside the provided rect using the provided texture. If the provided guid is selected and selection tint is enabled, the texture will be tinted.

---

public static void **DrawCustomIcon** (string **guid**, Rect **rect**, Texture **texture**, Rect **textureRect**)

Parameters

`guid`	The guid of the asset to be drawn.
`rect`	The area in which the texture should be drawn.
`texture`	The graphics which should be drawn.
`textureRect`	The area of the texture to be drawn.

Description

Draws an icon inside the provided rect using the provided texture. If the provided guid is selected and selection tint is enabled, the texture will be tinted.

AssetIconModule

class in **AssetIcons.Editors**

Description

The base module, used to evaluate targets and draw simple static icons.

Public Functions

`Draw`	Draws a simple static icon. Called when matched by EvaluateTargets by AssetIconTools.
`GetIcon`	Returns the icon the module draws.
`EvaluateTarget`	Details are passed about the target and it is evaluated to decide whether Draw should be called by AssetIconTools.

ExtensionTypeModule

class in **AssetIcons.Editors**

Description

Used to draw custom icons on assets depending on their extension name.

Public Functions

`EvaluateTarget`	Returns true if the target details has a matching extension name to the one of this module.
`SetDirtyExtension`	Marks the extensions type as dirty, forcing the validity of the extension to be rechecked.
`IsValid`	Checks if the extension of the module is valid.
`IsInUseExtension`	Checks AssetIconTools.ExtensionDrawers for other ExtensionTypeModules and compares extension names.

Static Functions

`IsValidExtension`	Checks if the passed extension is contained in InvalidExtensions.

ModuleManifest

class in **AssetIcons.Editors**

Description

Keeps a record of all modules currently loaded by AssetIcons.

Static Variables

`ExtensionDrawers`	A list of all modules currently tracked by AssetIcons. Loads from Resources, but tracks adding globally.

Static Functions

`ReloadModules`	Reloads all modules from the Resources directories.