



## What is this package?

This is BitCake's set of useful tools that will let you quickly bootstrap a new Unity 5 project. We use it in our projects and hope it will be of help for yours!

This project is also Open Source, you can find it in the url:

<https://bitbucket.org/Bitcake-Studio/bitstrap>

## How do I get set up?

This is the easy part! Since this is a Unity Project, just download it and then continue from there. Alternatively, you can just add all these scripts to an already existing project.

## What is inside?

The scripts inside are divided in 6 categories, we're gonna list everything below, with a quick explanation of each feature.

# Helpers

Helpers are scripts that facilitate some common tasks you might find during Unity development.

### EditorHelper

Bunch of EditorGUI[Layout] helper methods to ease your Unity custom editor development.

### FileSystemHelper

Complementary methods to the System.IO classes.

## ProjectBrowserHelper

Bunch of helper methods to work with the project window.

## BitHelper

Helper class for working with the bits inside an int.

## VectorHelper

Bunch of Vector math methods that you won't find neither in Vector3 or Vector2.

## EmailHelper

Bunch of helper methods to work with email strings.

## ReflectionHelper

Complementary methods to the System.Reflection classes.

## StringHelper

Bunch of helper methods to work with the string class.

## ScriptDefinesHelper

Helper to work with scripting define symbols.

## ScriptDefinesHelper

Helper to work with scripting define symbols.

## StaticReflectionHelper

Bunch of static reflection helper methods.

# Inspector

Some useful attributes and scripts for debugging or quick testing features of your game

## ButtonAttribute

Put this attribute above one of your MonoBehaviour method and it will draw a button in the inspector automatically.

## LayerSelectorAttribute

Put this attribute above an int field and it will draw like a layer picker in the inspector.

## BrowserConsoleLogHook

Debug helper class that will redirect Unity's Debug.Log messages to your browser's console via the javascript equivalent: "console.log()".

## MirrorTool

Tool that lets you mirror several Transforms in the hierarchy given a reflect normal. Useful when level designing.

## RuntimeConsole

Enables a console that shows Unity's Debug.Log messages when user holds "Shift+C" down. It's meant for debug purposes.

## TagSelectorAttribute

Put this attribute above an int field and it will draw like a tag picker in the inspector.

## ComponentContextMenu

Provides cool features to the component context menu such as: "Fold All", "Move to Top", "Move to Bottom" and "Sort Components".

## Isolate

Isolates part of the game objects hierarchy by activating the selected ones and deactivating the others. Behaves similarly to the isolate tool present in Autodesk Maya. Just select a game object and press Ctrl+E to isolate it and its children and parents.

## EditorCoroutine

Makes it possible to run coroutines while inside an editor script using the EditorApplication.update callback.

## ScriptCreator

Create C# Script and C# Editor Script through the "Assets > Create" menu.

# Util

Scripts that make day-to-day life programming easier, usually used during gameplay programming itself.

## Callback

System.Action helper class.

It allows you to bypass the C# delegate idiom "if( callback != null ) callback();".

Use it like "Callback.Trigger( delegate );"

Example:

```
System.Action<int> myCallback;  
  
// ...  
  
Callback.Trigger( callback, 17 );
```

## Create

Bunch of prefab utility methods.

Allows easy instantiation of prefabs and scripts.

It is also better than regular Instantiate as it copies the prefab transform.

Examples:

```
MyAwesomeScript newInstance = Create.Behaviour<MyAwesomeScript>( parentTransform );
```

```
public MyAwesomeScript prefab;  
  
// ...  
  
MyAwesomeScript newInstance = Create.Behaviour( prefab, parentTransform );
```

```
public GameObject prefab;

// ...

GameObject newInstance = Create.Prefab( prefab, parentTransform );
```

## Modifiable

Wraps a type that can be modified given an aggregate function.

Example :

```
Modifiable<float> speed = new Modifiable<float>(2.0f, (a, b) => a * b);

speed.AddModifier("sprint", 1.5f);
Debug.Log(speed.OriginalValue); // Prints 2.0f
Debug.Log(speed.ModifiedValue); // Prints 3.0f
speed.RemoveModifier("sprint");
Debug.Log(speed.ModifiedValue); // Prints 2.0f
```

## “NestedPrefab”

Simple nested prefabs achieved by instantiating a list of prefabs at runtime and parenting them inside of "Attach To" transform.

## SafeAction

Safe version of System.Action that envelopes each call in a try/catch to prevent execution flow interruption just because of one bad callback.

## Singleton<T>

Simple singleton class that implements the singleton code design pattern.  
Use it by inheriting from this class, using T as the class itself.

Example :

```
public class GameController : Singleton<GameController>
{
    // Access the singleton instance with GameController.Instance
}
```

## Timer

Timer utility class. Allows you to receive a callback after a certain amount of time has elapsed.

## EditorPrefProperty (String, Bool, Float, Int)

Makes it easy to work with EditorPrefs treating them as properties.

## PlayerPrefsProperties

Makes it easy to work with PlayerPrefs treating them as properties.

## EditorPrefsProperties

Makes it easy to work with EditorPrefs treating them as properties.

# Math

Some useful mathematical functions.

## NumberBounds<T>, IntBounds, FloatBounds

Represents a number bounds. Contains a minimum and maximum value.

Also it has a nice inspector with auto validation.

## SecureInt

Use this class if you want to protect a sensitive value in your game from Memory Modifying attacks, such as memory lock.

This does not fully protect your game against malicious attacker, though, it just makes it more difficult.

Performance impact is quite small, however, use it only on sensitive informations like Health or currency, for instance.

A downside of this class is that you can't change it's values while in play mode through the inspector, as it's encrypted during play.

## FastRandom

Fast and simple pseudo random generator class that allows you to reset its seed. Used mainly inside SecureInt, but you can also use it whenever you need a garbage-free random generator where you can set a specific seed.

# Graphic

Scripts relating to Unity graphics stuff.

## ParticleController

A Particle System wrapper that does not generate garbage.

Sometimes in Unity when you have a particle system with sub-particle systems, playing or stopping it will end up generating garbage in memory, this component eliminates this problem, just replace ParticleSystem with ParticleController.

## TweenShader

A simple tween class for shader param interpolation.

It uses the MaterialPropertyBlock class to modify the renderer's material properties.

It is better to use it than directly setting a material property because it does not create a new Material instance and, thus, does not generate a new draw call.

# Animation

Scripts that makes animators happy! :D

## AnimationParameter ( Bool, Float, Int, Trigger )

Helps with optimization, caching the index of the parameter name for you automatically, also makes animation parameters type-safe.

Comes with a nice editor that allows you to choose the parameters from a dropdown menu in the inspector if the current object has an Animator component.

## AnimatorControllerEditor

An editor script for the AnimatorController that allows you to add/remove child clips inside the AnimatorController. This is useful when you want to duplicate an animator, as all references in the new animator will be to its own child clips

# Extensions

Scripts that add functionality to the existing Unity classes.

## ArrayExtensions

Utility extension methods to the Array class.

Also, it contains some System.Linq like methods that does not generate garbage.

## ColorExtensions

Utility extension methods to the Color class.

## ComponentExtensions

Utility extension methods to the Component class.

## DictionaryExtension

Utility extension methods to the generic Dictionary class.

These methods are intended to be System.Linq substitutes as they do not generate garbage.

## GameObjectExtensions

Utility extension methods to the GameObject class, for now it only contains a version of GetComponentInParent <T>() that includes inactive objects in the search

## ListExtensions

Utility extension methods to the generic List class.

These methods are intended to be System.Linq substitutes as they do not generate garbage.

## StringExtensions

Utility extension methods to the string class, things like SeparateCamelCase(), Distance(),