Singleton Liberation

Let's free your hands.

- Need MonoBehaviour becomes singleton?   ~~~**Yep!**
- Want a more considerate implementation?   ~~~**If possible.**
- Have to make it from scratch?   ~~~**Better not.**
- Costs your only chance to inherit in C#?   ~~~**Definitely not!**

Overview:

- Full source code and examples included, and for MonoBehaviour in C# only.
- Auto find existing instance, and warning if duplicates found.
- Auto create if none, and ask user if to create when edit mode, without making a ghost silently.
- Not required to derive from any specific class, since C# inheritance is so precious.

The further technical documentation is available here.
And the tutorial is right below.

Coding:

- All you need to do is to declare a property.

```
using WanzyeeStudio;
public class YourSingleton : YourMonoBehaviour{
    public static YourSingleton instance{
        get{ return Singleton<YourSingleton>.instance; }
    }
}
```

- If you wanna do more for safe, check it in Awake().

```
private void Awake(){
    if(this != instance) Destroy(this);
    else DontDestroyOnLoad(gameObject);
}
```

- Or if you don't need custom inheritance, it becomes much easier.

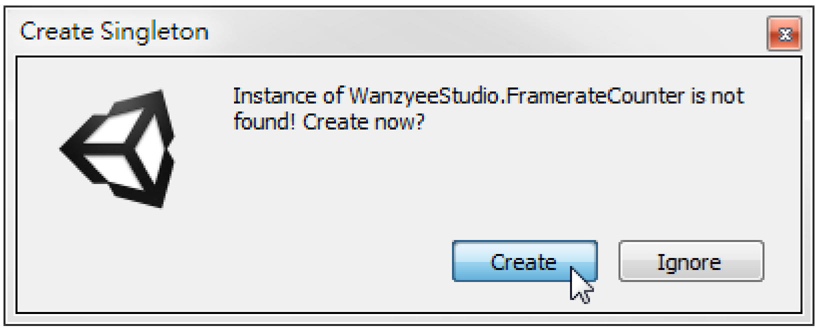
```
using WanzyeeStudio;
public class YourSingleton : BaseSingleton<YourSingleton>{
    //...
}
```

- Nothing more to do, just access the singleton and keep creating.

```
public class AnotherClass{
    private void DoSomething(){
        Debug.Log(YourSingleton.instance);
    }
}
```

Notifications:

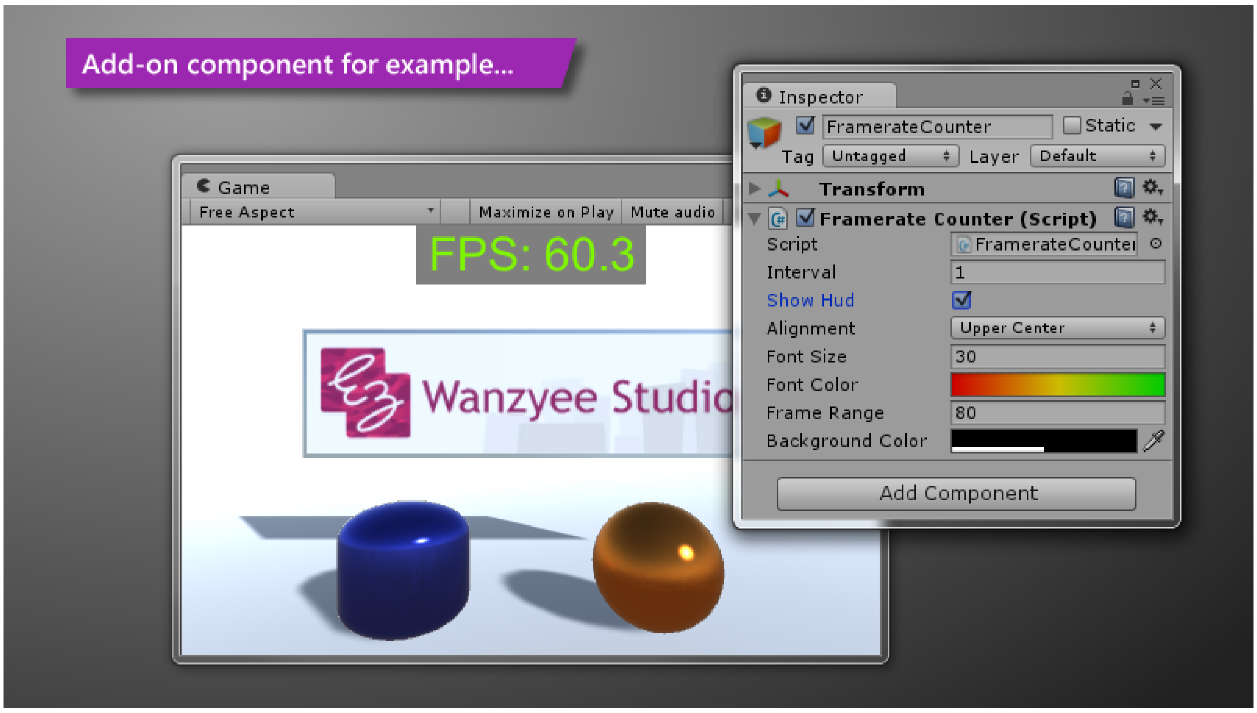
- For preventing mistakes, this'll notify when things below happen.
- Throw exception when coding incorrectly, see the documentation for detail.
- Throw exception or log error when multiple instances found existing.
- Pop up a dialog to make sure user allow the script to create a new instance in edit mode.


Create Singleton

Instance of WanzyeeStudio.FramerateCounter is not found! Create now?

Examples:

- The class **BaseSingleton** is also the example for **Singleton**.
- And a simple **FramerateCounter** is the example for **BaseSingleton**.



FramerateCounter
Static
Tag Untagged
Layer Default
Framerate Counter (Script)
Script FramerateCounter
Interval 1
Show Hud ✓
Alignment Upper Center
Font Size 30
Font Color
Frame Range 80
Background Color

Release Notes:

1.0.2
- Update the documentation and the component help links.

1.0.1
- Fix usage of Object.DontDestroyOnLoad() since it only applies to the root object.
- Fix the FramerateCounter's API to allow access while editing.

1.0
- First release.

Publish Notes:

- Requires: Unity 5.1.0f3 or higher.
- Category: Scripting
- Keywords: Singleton Design Pattern Inherit MonoBehaviour Component FPS Framerate

unity Asset Store Free

Singleton< T >

Agent to implement the singleton pattern for `UnityEngine.MonoBehaviour`. More...

Properties

static T	**instance** `[get]`
	Get the singleton instance. More...

Detailed Description

Agent to implement the singleton pattern for `UnityEngine.MonoBehaviour`.

Not required to derive from any specific class, since C# inheritance is so precious. This finds the existed instance and check if multiple, creates one if not found. Set `Object.DontDestroyOnLoad()` if the instance is created by this. The others found, created manually or by scripts, should be maintained by the creator. It might need to check and destroy duplicated when **`Awake()`**.

In the general case, singleton should be implemented in the certain class for better enclosing. And sometimes, we do need to manually create one to edit in the Inspector. Make an agent for `UnityEngine.MonoBehaviour` to make it easier in the common case. This'll throw exception to remind the user to correct, when the code or scene setup wrong. And pop up a dialog if tries to create in edit mode, to avoid making scene dirty silently.

Since we can't protect the constructor without inheritance. For the purpose of better enclosed programming, here's usage limitations:

1. Only allow the class of current singleton to access.
2. Only allow one method to access to keep the code clean.
3. Don't assign to a delegate, otherwise we can't keep the same accessor.
4. Don't access from any constructor or assign to a field, that makes out-of-date.

Example to wrap to access, call this in a method or property:

```
using WanzyeeStudio;
public class SomeComp : MonoBehaviour{
    public static SomeComp instance{
        get{ return Singleton<SomeComp>.instance; }
    }
}
```

Example to maintain in **`Awake()`** to avoid duplicated, check to keep or destroy:

```
private void Awake(){
    if(this != instance) Destroy(this);
    else DontDestroyOnLoad(gameObject);
}
```

Type Constraints

T : *MonoBehaviour*

T : *new()*

Property Documentation

T instance static get

Get the singleton instance.

The instance.

BaseSingleton< T >

Specific Classes » ComponentsReleased Packages » |
Singleton Liberation

Base class to implement the singleton pattern for `UnityEngine.MonoBehaviour`. More...

Inherits MonoBehaviour.

Protected Member Functions

virtual void	**Awake** ()
	Awake, check and handle duplicated instances. More...

Static Protected Attributes

static bool	**autoDestroy**
	Flag if to destroy excess instance automatically when it **`Awake()`**. More...

Properties

static T	**instance** `[get]`
	Get the singleton instance. More...

Detailed Description

Base class to implement the singleton pattern for `UnityEngine.MonoBehaviour`.

This works with **`Singleton`** to minimize coding, if you don't need custom inheritance. Derive from this to make a `UnityEngine.MonoBehaviour` singleton. Only allow the class of current singleton to derive. This applies the singleton's root `Object.DontDestroyOnLoad()` And handle duplicates when **`Awake()`**.

Example to implement singleton, just derive from this:

```
using WanzyeeStudio;
public class SomeComp : BaseSingleton<SomeComp>{}
```

Example to access from another class:

```
public class AnotherClass{
    private void DoSomething(){
        Debug.Log(SomeComp.instance);
    }
}
```

Type Constraints

T : ***BaseSingleton****<T>*

T : *new()*

Member Function Documentation

virtual void Awake ( ) protected virtual

Awake, check and handle duplicated instances.

Apply `Object.DontDestroyOnLoad()` to the root if this's the singleton. Otherwise check if `autoDestroy` to suicide or log error. Call `base.Awake()` if you implement `Awake()` in the subclass.

Reimplemented in **FramerateCounter**.

Member Data Documentation

bool autoDestroy static protected

Flag if to destroy excess instance automatically when it `Awake()`.

Property Documentation

T instance static get

Get the singleton instance.

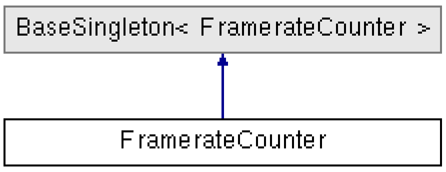
The instance.

FramerateCounter

Specific Classes » ComponentsReleased Packages » |
Singleton Liberation

Calculate FPS at runtime, optional to show on GUI with color warning. More...

Inheritance diagram for FramerateCounter:



Public Attributes

float	**interval** = 1f The interval to calculate FPS. More...
bool	**showHud** Flag if show FPS on GUI. More...
TextAnchor	**alignment** = TextAnchor.UpperRight The alignment of GUI on screen. More...
int	**fontSize** = 20 The size of the font. More...
Gradient	**fontColor** The gradient of the font color, map with frame range for warning. More...
float	**frameRange** = 80f The frame range, used to calculate the font color of current FPS. More...
Color	**backgroundColor** = new Color(0f, 0f, 0f, 0.5f) The color of the GUI label background. More...

Protected Member Functions

override void	**Awake** () Awake, initialize GUI content and style. More...

Static Protected Attributes

static bool	**autoDestroy** Flag if to destroy excess instance automatically when it `Awake()`. More...

Properties

static float	**fps** `[get]` Current frame rate per second, updated every time interval set on instance. More...
static bool	**show** `[get, set]` Flag to show FPS on screen GUI or not. More...

static T **instance** `[get]`
Get the singleton instance. More...

Detailed Description

Calculate FPS at runtime, optional to show on GUI with color warning.

Member Function Documentation

override void Awake ( ) protected virtual

Awake, initialize GUI content and style.

Reimplemented from **BaseSingleton< FramerateCounter >**.

Member Data Documentation

float interval = 1f

The interval to calculate FPS.

bool showHud

Flag if show FPS on GUI.

TextAnchor alignment = TextAnchor.UpperRight

The alignment of GUI on screen.

int fontSize = 20

The size of the font.

Gradient fontColor

The gradient of the font color, map with frame range for warning.

float frameRange = 80f

The frame range, used to calculate the font color of current FPS.

Color backgroundColor = new Color(0f, 0f, 0f, 0.5f)

The color of the GUI label background.

bool autoDestroy static protected inherited

Flag if to destroy excess instance automatically when it `Awake()`.

Property Documentation

float fps static get

Current frame rate per second, updated every time interval set on instance.

The FPS.

bool show static get set

Flag to show FPS on screen GUI or not.

`true` if show; otherwise, `false`.

T instance static get inherited

Get the singleton instance.

The instance.