Hey there!

Huge thanks for buying **Advanced FPS Counter**, tiny counters set for Unity3D!
Current version shows these counters:

- **Frames Per Second Counter** (with optional ms, average and min / max values)
- **Memory Counter** (total, allocated, mono)
- **Device Information Counter** (OS, CPU, GPU, RAM, Screen info)

Full API documentation can be found here:
http://codestage.ru/unity/fpscounter/api/
Please, consider visiting it first if you have any questions on plugin usage.

Plugin features in-depth

Common features

Anchoring System and Labels

Plugin has flexible counters Anchoring System with smartly updatable Labels.

Any Counter can use one of these Anchors: TopLeft, TopRight, BottomLeft, BottomRight, UpperCenter, LowerCenter, covering all four corners, top and the bottom of the screen.

All counters with same anchor are drawn within one Label, e.g. one Label may contain several Counters.
Labels content is refreshed (and reassembled) only if any containing Counter is marked as dirty (has changed value since last update). It allows avoiding unnecessary waste of resources.

Labels are drawn using uGUI Texts placed within the automatically generated Canvas with Screen Space – Overlay mode with customizable Sort Order value.
Thus, in order to see the counters, you don't need to perform any additional actions in most cases.

IMPORTANT:

- If you'll place AFPSCounter somewhere inside the existing Canvas, auto-generated one will inherit options from the parent. This is useful when you wish to use plugin in some special environment, such as a VR project.

Operation Modes

Plugin has three operation modes: **Disabled**, **Normal** and **Background**.

Disabled mode is used to remove all counters and stop all internal processes except the global hotkey listener.
Normal mode should be used in most cases: counters are visible and operate as intended.
Background mode allows reading all enabled counters data keeping them hidden and avoiding any additional resources usage for assembling and showing counters values on screen.
May be useful for performance or hardware statistics collection or proper visual settings suggestion for best gaming experience for example.

Other common features

- Customizable global **Hot Key** to enable / disable plugin.
- **Keep Alive** option allows to keep Advanced FPS Counter's Game Object on new scene load.
  IMPORTANT:
  If AFPSCounter is placed on the nested GameObject, the topmost root object will be kept alive instead.

- **Force FPS** option allows to try your game at specified frame rate, may help to check your game behavior on slow devices; specified frame rate is not guaranteed though (and may not work at all).
- Customizable common look and feel (scale, font, font size, line spacing, counters spacing, anchors padding).
- Shadow & Outline effects with customizable colors and distance.
- All counters may be enabled and disabled at any time.
- AFPSCounter component inspector has detailed tooltips with descriptions and hints.
- Plugin may be used purely from code through the public APIs.

Counters

Frames Per Second Counter

This is a Counter showing **Current FPS**. It also has three optional additions: **Milliseconds**, **Average FPS** and **Minimum / Maximum FPS**.

Milliseconds option shows an approximate time spent on 1 frame processing. Available for all FPS readings.
Average FPS option shows an averaged FPS using accumulates samples.
MinMax FPS option shows minimum and maximum FPS.

Both Average and MinMax FPS values may be reset on new scene load (optionally) or using public API methods.

Average FPS has optional accumulative range, allowing collection of the few FPS samples and getting average from collected samples. Use 0 range to collect all FPS samples since last Average FPS reset (optimized calculation will be used in such case).

MinMax has optional “pre-warm” delay – it may skip selected amount of updates before recording values.

- Has configurable update interval in seconds (I’d suggest something between 0.5 and 1).
- FPS values are colored with customizable colors within three customizable intervals: normal, warning and critical.

Memory Counter

This is a Counter showing memory usage information.
It contains **Total Counter**, **Allocated Counter** and **Mono Counter**. Each may be enabled or disabled at any time.

Total Counter shows total private memory amount, reserved by OS for application, which can’t be used by other applications. E.g. app says to OS: “I wish 10 megs of memory to run well and quickly place new data in memory, please give me 10 megs for private use”. And, if there is 10 megs of free RAM, OS **may** reserve this space for application if it will consider it reasonable and possible. Otherwise, it will reserve only necessary amount of ram required by current application allocations (can happen in low free RAM environment).
Using Total Counter, you’ll be able to know how much RAM were reserved by your application for private use.

Allocated Counter shows amount of private memory currently actually **used** by application. In other words - how much memory all your textures, sounds, etc. use.

Mono Counter shows amount of memory, allocated by managed Mono objects, such as UnityEngine.Object and everything derived from it. As you may know, almost all objects in Unity are just wrappers to their native representations, and these wrappers are managed Mono objects by themselves.

- Has configurable update interval in seconds (I’d suggest something between 1 and 5).
- Has **Precise** option. It allows to output memory values with additional accuracy using float values instead of int ones. Requires some extra resources though, thus try to avoid using it in conjunction with low update interval.
- Has configurable color.

Device Information Counter

This is a Counter showing different information about current device.
Actually it updates only once (on start), so it's not a counter technically.
Anyway, it shows this information (all of these are optional and may be enabled and disabled at any time):

- Platform: outputs Operating System name with version (if possible) and runtime platform type.
  Windows 10 (10.0.0) 64bit [WindowsEditor]
- CPU: outputs CPU model and cores count (including virtual cores from Intel's Hyper Threading).
  Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz [8 cores]
- GPU Model: outputs GPU model name.
  AMD Radeon R9 200 / HD 7900 Series
- GPU API: outputs graphics API name, version and type (if possible).
  Direct3D 11.0 [level 11.0] [Direct3D11]
- GPU Spec: outputs graphics shader model (if possible), fillrate (if possible) and total VRAM (if possible).
  SM: 5.0, FR: 29600 MP/S, VRAM: 3054 MB
- RAM: outputs total RAM on current device in Megs.
  16282 MB
- Screen: outputs resolution with refresh rate, current window size and screen DPI (if possible).
  1920x1080@60Hz [window size: 800x600, DPI: 96]
- Has configurable color.

Setup

There are five possible ways to set up Advanced FPS Counter both from Editor and code:

- Drag Advanced FPS Counter prefab from Prefabs to the Hierarchy (my choice).
- Use hotkey CTRL+ALT+F (doesn't work sometimes, not my fault though).
- Use menu item GameObject > Create Other > Code Stage > Advanced FPS Counter.
- Add AFPSCounter component to the empty GameObject of your choice, as usual.
- Use AFPSCounter.AddToScene() from code and plugin will be automatically added to the scene.

Note: you need to specify package CodeStage.AdvancedFPSCounter to be able to work with plugin from code.
Here is a simple example:

```
// place this line right at the beginning of your .cs file!
using CodeStage.AdvancedFPSCounter;

// ...
private void Start()
{
	// will instantiate AFPSCounter in scene if it not exists
	// with disabled keepAlive option
	AFPSCounter.AddToScene(false);

	// and change spacing between counters
	AFPSCounter.Instance.CountersSpacing = 1;
}
```

Tips

- Please, take a look at the ExampleScene (at the Examples folder) to see how to work with plugin from code and check how you can affect AFPSCounter at runtime.
- You may easily tune counters (colors, intervals, etc.) in Play mode and save adjusted values using these steps:
  - Enter Play mode;
  - Tune AFPSCounter component settings in inspector;
  - Right-click on the AFPSCounter component's header and select "Copy Component";
  - Exit Play mode;
  - Right-click on the AFPSCounter component's header and select "Paste Component Values";
  This technique works for any other components as well.
- To enable and disable whole AFPSCounter from code just use **AFPSCounter.Instance.OperationMode** property (switch it between AFPSCounterOperationMode.Disabled and AFPSCounterOperationMode.Normal).

Troubleshooting

- **I can't see AFPSCounter on the screen.**
  - Make sure you've added it to the current scene (either in Editor or from code).
  - Make sure it's enabled and active.
  - Check console for any error messages or warnings.
  - If you're using new UI (Unity 4.6 +) make sure your Canvas Sorting Order has value lower than AFPSCounter's Sorting Order setting value.
  - If you're working with VR, place AFPSCounter inside an existing Canvas with World Space Render Mode configured to be visible in your VR device (see Examples/Prefabs/World Space Example). Note: KeepAlive option will not work when AFPSCounter is placed on the nested GameObject.
- **Something's wrong with my font I use for the counters.**
  Make sure the font you're using has a Bold style version included. Unity had a bug, which leads to the exclusion of the other styles of the font in case you're using different font files for the different styles.
  You need to touch import settings for your font in order to fix it.
  - Try to select your font and add a whitespace or comma to the Font Names, hit Apply.
  - Check it fixes an issue for you.

Compatibility

Plugin should work fine on any platform generally (including WebGL!).
However, I had no chance to test plugin on all Unity platforms since I just have no appropriate devices and / or licenses.

I've tested Plugin on these platforms:
PC (Win, Mac, WebPlayer, WebGL), **iOS**, **Android, Win Phone, Windows Store**.

In addition, customers reported it as working on these:
Wii U devkit (thx Black Lodge Games).
Xbox One (thx Yaroslav Bakhvalov).

Please, let me know if plugin doesn't work for you on some specific platform and I'll try to help and fix it.

All features should work fine with **any** stripping level and **IL2CPP** runtime.
Plugin works with both **Unity Free** and Pro!

Final words

I hope you'll find Advanced FPS Counter helpful and it will save some of your priceless time!
Please, leave your reviews at the plugin's Asset Store page and feel free to drop me bug reports, feature suggestions and other thoughts on the forum or via support contacts!

AFPSCounter links:

Asset Store | Website | Forum

Support contacts:

E-mail: focus@codestage.ru
Other: blog.codestage.ru/contacts

Best wishes,
Dmitriy Yukhanov
Asset Store publisher
blog.codestage.ru
@dmitriy_focus

P.S. #0 I wish to thank my family for supporting me in my Unity Asset Store efforts and making me happy every day!
*P.S. #1 I wish to say huge thanks to **Daniele Giardini** (DOTween, HOTools, Goscurry and many other happiness generating things creator) for priceless feedback on this plugin!*