Hey there!

First of all, huge thanks for buying this tiny yet flexible **Advanced FPS Counter** Unity3D plugin!
Current version shows these counters:

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

IMPORTANT:
Flash Player is not supported since 1.2.5!

Advanced FPS Counter has a full API documentation here:
http://codestage.ru/unity/fpscounter/api/
Please, consider visiting it first if you have any questions on plugin usage!

Plugin features in-depth

Counters

Frames Per Second Counter

This is a Counter showing **Current FPS**. It also has two optional additions: **Average FPS** and **Minimum / Maximum FPS**. Both Average and Min/Max FPS may be reset both on new scene load (optionally) and 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).

- Has configurable update interval in seconds (I’d suggest something between 0.5 and 1).
- FPS values are colored with customizable colors with three 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 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, all objects in Unity you’re usually work with are just wrappers to their native representations, and these wrappers are managed Mono objects.

- 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 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 allows showing on these information (all of these are optional and may be enabled and disabled at any time):

- CPU: outputs CPU model and maximum threads count.
  Example: Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz (8 threads)
- GPU: outputs GPU model, shader model (if possible) and total VRAM in Megs (if possible)
  Example: AMD Radeon HD 7900 Series (SM: 5.0, VRAM: 3046 MB)
- RAM: outputs total RAM on current device in Megs
  Example: 16282 MB
- Screen: outputs resolution with refresh rate and current screen size
  Example: 1920x1080@60Hz (window size: 912x602)

- Has configurable color.

Common features

Anchoring System and Labels

Plugin has flexible counters Anchoring System with smartly updatable Labels.

Any Counter may be anchored to one of these Anchors: TopLeft, TopRight, BottomLeft, BottomRight, covering all four corners of the screen.

All counters with same anchor are packed within one Label, e.g. one Label may contain several Counters.
Label relies on single GUIText component under the hood. Thus, in order to see the counters, you should have at least one Camera in your scene with these settings:
- AFPSCounter GameObject’s Layer (Default by default =D) included into the Culling Mask
- Has enabled GUILayer component (default for any new Camera)
- Camera should be on top of other cameras (with higher depth)

Consider these requirements for troubleshooting.

One Draw Call (DC) used to render one Label, thus you’ll have only one DC for all counters with same Anchor.
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.

All counters has configurable Anchor.

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 listening.
Normal mode is most common, allows to see counters on screen and runs all internal processes as intended.
Background mode allows to read 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 collecting performance or hardware statistics or suggesting proper visual settings for best gaming experience of your customers for example.

Other common features

- Customizable global **Hot Key** to enable / disable plugin
- **Keep Alive** option allows to keep Advanced FPS Counter on new scene load

- **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
- Customizable common look and feel (font, font size, line spacing, pixel offset)
- All counters may be enabled and disabled at any time
- Most items in the AFPSCounter component inspector has tooltips with descriptions and hints

Setup – Tips – Troubleshooting

Setup

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

- Drag Advanced FPS Counter prefab from CodeStage\AdvancedFPSCounter\Prefabs to the Hierarchy (my choice)
- Use hotkey CTRL+ALT+SHIFT+F
- Use menu item GameObject > Create Other > Code Stage > Advanced FPS Counter
- Use AFPSCounter.Instance.* 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. Example:

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

// ...

void Start()
{
      // will instantiate AFPSCounter if it not exists
      // and disable its keepAlive property
      AFPSCounter.Instance.keepAlive = false;
}
```

Tips

- Please, take a look at the ExampleScene (at the CodeStage\AdvancedFPSCounter\Examples folder) to see how to work with plugin from code and check how you can affect AFPS Counter at runtime
- You may easily use AFPS Counter in each scene in editor – just make sure AFPS Counter instance in your main (launch) scene has “keepAlive” property enabled and all other AFPS Counters in other scenes has **EditorOnly** tag
- 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

If you don’t see the AFPSCounter on the screen:

- Make sure you added it to the current scene (either in Editor or from code)
- Make sure it’s enabled (and avoid using component enable checkbox at all – use either GameObject activation checkbox to completely activate / deactivate plugin or AFPSCounter.Instance.OperationMode property)
- Make sure you have GUILayer on your camera and you have included layer with AFPSCounter in camera’s culling mask

- Check console for any error messages or warnings
- If nothing helps, write to the plugin support (you may find contacts below).

Don't forget to take a look at the CodeStage\AdvancedFPSCounter\Changelog.txt if you wish to know what was changed in last release and to keep your eyes peeled in case something changed in API or file structure (I usually note such changes in the changelog).

Few boring words

Plugin should work on any platform generally. Some features may have platform limitations though; they are mentioned here and / or in API docs in such cases.

I had no chance to test plugin on all platforms Unity capable to build for since I just have no appropriate devices and / or licenses. Plugin mostly tested on these platforms: **PC** (both Win and Mac, both Standalone and WebPlayer), **iOS**, **Android and even WP8 Emulator**.
Please, let me know if plugin doesn't work for you on some specific platform and I'll try to help and fix it.
And I'd glad to know if it works for you on any other platform as well.

All features should work fine with **micro mscorlib** stripping level and **.NET 2.0 Subset** API level!
Plugin doesn't require Unity Pro license.
Feel free to send me any bug reports, suggestions and feature requests via plugin support (you may find contacts below).

Please, don't be indifferent and leave your votes and reviews on the Asset Store plugin page. Every single vote or review is always appreciated!

AFPSCounter links:
Asset Store | Web Site | Forum

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

Best wishes,
Dmitriy Yukhanov
blog.codestage.ru
@dmitriy_focus
my Unity3D plugins

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** (HOTween, HOTools, Goscurry and many other happiness generating things creator) for priceless feedback on this plugin!*