FULLSCREEN EDITOR

Contents

- Overview
- How to Use
- Contact
- FAQ
- API

Thanks for using **Fullscreen Editor**

★ Don't forget to leave a review on the store page if you liked it, this helps me a lot!

Overview

Want to play your game in fullscreen without building it? Want to edit your scene in fullscreen? Now you can!

- **Plug 'n' play**, it works out of the box
- **Better performance** since version 2.1.0, no more framerate drops!
- **Multi-screen support**, use as many monitors as you want
- **Fullscreen on play**, because maximize on play is a waste of space
- **Configurable** to best fit your needs
- **Non-intrusive**
- **Fullscreen for any window**, even the whole editor
- **Keep the state**, don't lose changes made on the fullscreened window
- **Clean code**

An extension that does what its name says, it puts editor windows in fullscreen mode, simple and easy, useful for recording, testing in a real gaming environment and tweaking your scene. Everything is still fully functional in fullscreen.

Supported Platforms

- **Windows:** All features available, out-of-the-box multi-display fullscreen.
- **macOS:** Not tested on multi-display setups.
- **Linux:** Requires `wmctrl` to be installed and a EWMH/NetWM compatible X Window Manager. Has some features limitations.

How to Use

Simply press the shortcut to fullscreen a window or close it, defaults are:

F9 for any focused view;
F10 for game view;
F11 for scene view;
F12 for the main view.

These shortcuts can be changed anytime in the preferences menu.

Supports all Unity versions since 5.6.
Source code included!

Contact

If you have any suggestion, bug report or question you can contact me through my email (samuelschultze@gmail.com) or, if you prefer, this forum thread.

⚠ Send me your invoice number when asking for support, this way I can send you updated files and solve your issue as fast as possible.

Check out my other assets

Follow me on GitHub

Asset Store page

Forum thread

Mail me

Website

Unity Connect

FAQ

How to open the preferences window?

> The preferences are located alongside Unity preferences, go to Edit/Preferences (or Unity/Preferences on macOS), you'll see a "Fullscreen" or "Fullscreen Editor" tab.

The extension seems to be duplicated, why?

> You imported a newer version when you had a previous version installed, delete all the older files and import the plugin again.

How to change the keybindings?

> Just go to the preferences, make your changes and press "Apply shortcuts", unity will recompile and your new keybindings will be working.

What is the "Show toolbar" option?

> It's an option for hiding or showing the Scene View or GameView toolbar while on fullscreen, the toolbar that contains the Maximize on play, Stats, Mute Audio, etc.

My game fail to compile if I use the extension, how to fix it?

> The extension must be inside the "Editor" folder because it uses editor only API.

API

Fullscreening a window by code

You can use `Fullscreen.MakeFullscreen<WindowType>(windowReference)` or `Fullscreen.ToggleFullscreen<WindowType>(windowReference)` to fullscreen any window.
If `windowReference` is null the extension will automaticaly create a new instance of the window based on the `WindowType`, however, this instance will be destroyed as soon as its parent fullscreen exits.

Example code:

```
using FullscreenEditor; // Don't forget this at the top of your script

[MenuItem("Fullscreen Example/Fullscreen Scene View")]
public static void Example() {

  //Find the window instance
  var sceneView = EditorWindow.GetWindow<SceneView>();

  // Make it fullscreen
  var fullscreen = Fullscreen.MakeFullscreen<SceneView>(sceneView);

  // And then, for exiting when needed
  fullscreen.Close();

  // As an alternative for automatically opening/exiting
  Fullscreen.ToggleFullscreen<SceneView>(sceneView);

}
```

Implementing a custom fullscreen rect logic

A custom logic for calculating fullscreen rects can be easily implemented by assigning a custom callback to the `FullscreenRects.CustomRectCallback` property.

Example code:

```
using System.Linq;
using FullscreenEditor;
using UnityEditor;
using UnityEngine;

[InitializeOnLoad]
public static class HookedRectExample {

  static HookedRectExample() {
    // Register the custom rect callback
    FullscreenRects.CustomRectCallback += GetRect;
  }

  private static bool GetRect(RectSourceMode mode, out Rect outputRect) {
    // Get our current mouse position
    var mousePos = FullscreenUtility.MousePosition;
    // Get the rect of the entire screen
    var mainDisplay = FullscreenRects.GetMainDisplayRect();

    // Divide the display size by 2, so we can have 4 rects at the same time
    var splitDisplay = new Rect(mainDisplay.position, mainDisplay.size / 2f);

    // Create a array with the 4 possible rects
    var rects = new Rect[] {
      // Top-left
      new Rect(new Vector2(splitDisplay.xMin, splitDisplay.yMin), splitDisplay.size),
      // Top-right
      new Rect(new Vector2(splitDisplay.xMax, splitDisplay.yMin), splitDisplay.size),
      // Bottom-left
      new Rect(new Vector2(splitDisplay.xMin, splitDisplay.yMax), splitDisplay.size),
      // Bottom-right
      new Rect(new Vector2(splitDisplay.xMax, splitDisplay.yMax), splitDisplay.size)
    };

    // The output will be the rect from the array that is under our mouse pointer
    outputRect = rects.FirstOrDefault(rect => rect.Contains(mousePos));

    // We return true if the extension should use the output rect,
    // otherwise it'll calculate it based on the user preferences
    // In this case, use the output only if any of the 4 contains the mouse position
    return rects.Any(rect => rect.Contains(mousePos));
  }

}
```

⚠ This API may change between Fullscreen Editor versions.

Keep on creating those awesome games!

Made with ❤ by Samuel Schultze