

# uFileBrowser

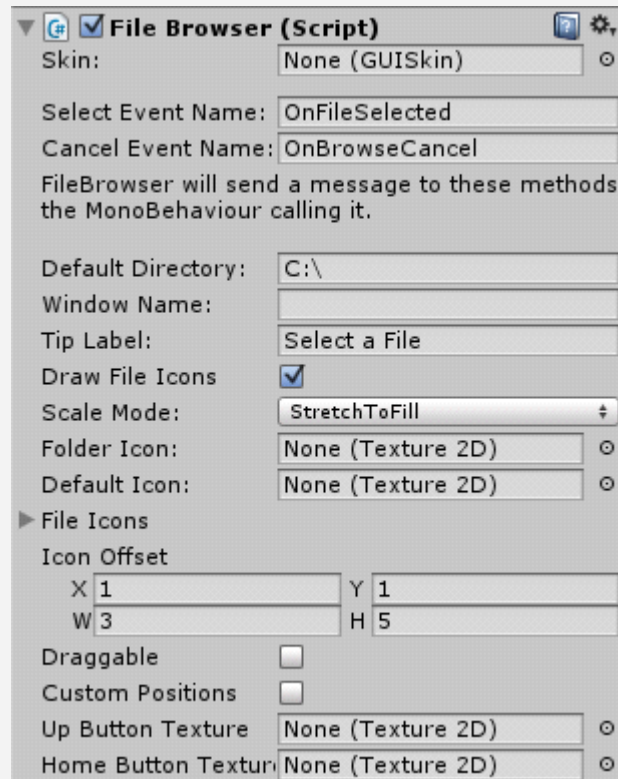
## Documentation

uFileBrowser is a simple File Browser for Standalone platforms, which doesn't require any plugins. It uses standard GUI to display and is easy to implement in your project. Easily customizable Skin, button/text/window positions.

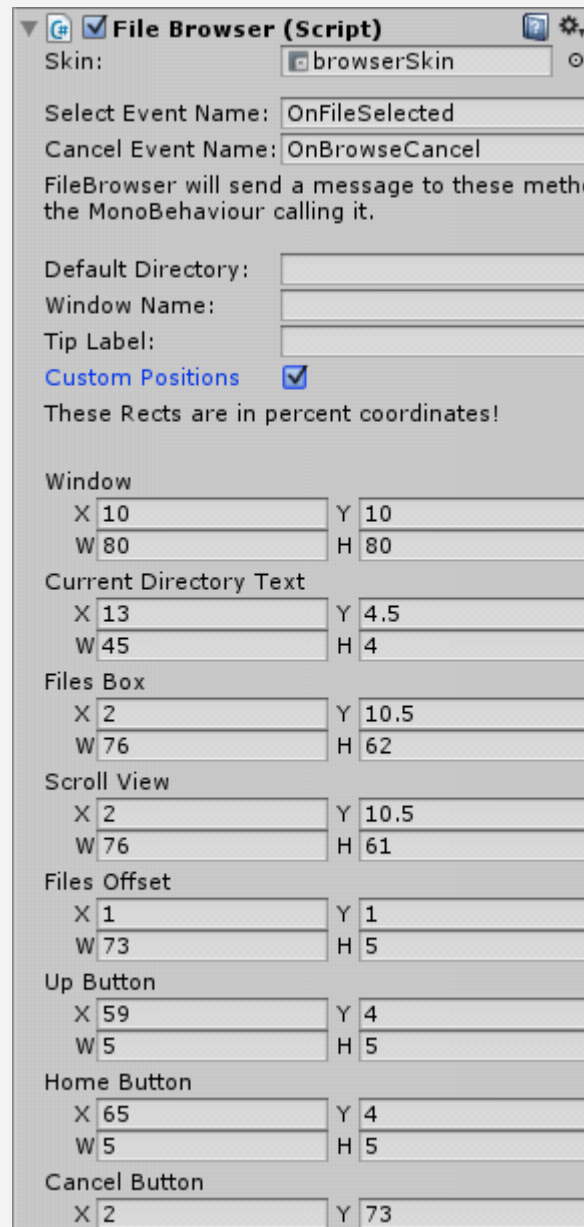
### Importing into your project:

After importing the asset to your project, you should see FileBrowser.cs. To use it, simply add the script to a GameObject.

This is the screen you should see:



You can customize it as you want, if you check "Custom Positions" you will be able to change the position of every displayed element of the browser.



**File Browser (Script)**

Skin:

Select Event Name:

Cancel Event Name:

FileBrowser will send a message to these methods the MonoBehaviour calling it.

Default Directory:

Window Name:

Tip Label:

Custom Positions ☒

These Rects are in percent coordinates!

**Window**

X  Y

W  H

**Current Directory Text**

X  Y

W  H

**Files Box**

X  Y

W  H

**Scroll View**

X  Y

W  H

**Files Offset**

X  Y

W  H

**Up Button**

X  Y

W  H

**Home Button**

X  Y

W  H

**Cancel Button**

X  Y

The Rects are in percent coords, which means the browser is compatible with all screen resolutions. For example if you set the window Width and Height to 50, the window will occupy 50% of the screen no matter what the screen resolution is.

**NOTE:** The GUI skin must have the 3 following Custom Styles:

- redLabel
- normalFile
- selectedFile

## Public Methods:

To Show up the browser in runtime you can make one simple script:

```
public class example : MonoBehaviour
{
    public FileBrowser browser;

    void OnGUI()
    {
        if (GUI.Button(new Rect(10, 1, 100, 30), "Show"))
        {
            if (!browser.isShowing) browser.Show(@"", this, FileSelectMode.File);
        }
    }

    void OnFileSelected(FileInfo info)
    {
    }

    void OnBrowseCancel()
    {
    }
}
```

The Show() method requires the following parameters:

1. string startDirectory: The directory that the browser will show when it first shows up. You can leave it empty and the browser will use the Default Directory you setup in the inspector, if it is also empty, it will display a drives list.
2. searchPattern: You can provide a single string or a string array as search patterns. If you want to search for a single file extension, the call would look like:

Show( startDirectory, "\*.txt", etc...).

If you want to search for multiple file extensions, the call would look like:

Show( startDirectory, new string[n] { "\*.txt", "\*.mp3", ... }, etc...).

Or you can provide an already initialized array like in the Demo2 scene.

3. MonoBehaviour returnMessage: The browser will send a message to this MonoBehaviour when the user selects a file or cancels browsing. You can setup the method names in the inspector.
4. [optional] FileSelectMode mode: With this enum you can select if the user has to select a file or folder, by default it is set to File.
5. [optional] bool CanCancel: If true the user will be able to press the cancel button. If false the Cancel button won't be displayed, by default it is true.

The Hide() method doesn't require any parameters, and simply hides the browser.

## The Select Event:

When the user presses the OK button an message is sent (by using SendMessage) to the MonoBehaviour specified in the show method.

You can specify the method name that has to receive the message in the inspector where "Selected Event Name" is.

The method should look like this:

```
void OnFileSelected(FileInfo info)
{
}
```

Make sure that the names match in code and in inspector. The method has to have the FileInfo parameter, the class is a custom class and it's structure looks like this:

```
public class FileInfo
{
    public string path;
    public string name;
    public bool isFile;

    public FileInfo(string path, string name, bool IsFile)
    {
        this.path = path;
        this.name = name;
        isFile = IsFile;
    }
}
```

1. path: its the full path to the file.
2. name: is the name of the file (file format included)
3. isFile: will be true if the path is a file, false if it is a directory.

## The Cancel Event:

When the user presses the cancel button, a message is sent just like in the Select Event, but doesn't need the FileInfo parameter.

## The File Change Event

When the user selects a file or a folder, this event is called. Note that when in File select mode, the user can't select a folder, so the event won't be called.

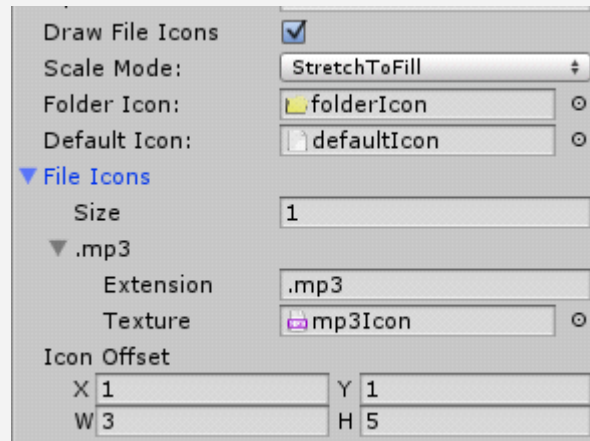
## Customization:

You can customize the browser skin by simply assigning a custom skin in the inspector or modifying the default one. The default skin uses three custom styles for the files list.

You can also easily change the window/buttons/labels position by checking the "Custom Positions" toggle. You can modify the positions at runtime, then click the gear icon in the inspector and click "Copy Component", then exit playmode and click the gear icon again, then click "Paste Component Values", now you will

have the values you changed in runtime.

You can add custom icons for any file extension. If you check the "Draw Icons" toggle, you will be able to set the list of custom icons.



The extensions need to contain the starting '.',  
the Icon offset rect is the X and Y offset from the file box and the width and height of the icon. (always percent coords)

In the inspector, you will also see three Text Fields, these are used for:

1. Default Directory: This is the default directory that will be used if no directory has been specified in the Show() method. If it is empty, the browser will display a list of drives.
2. Window Name: The name of the window that will be displayed. If empty, the browser will assign the default name to it.
3. Tip Label: The tip label is the red label you can see in the screenshots. If empty the browser will assign the default value to it.

The window is now optionally draggable, if you check the Draggable toggle in the inspector, you will be able to setup the 'Draggable Rect', for example if it is (0, 0, 80, 3) the user will be able to drag the window by clicking between 80% of the **Screen** width and 3% of the **Screen** height starting from the top left corner of the window.

## Play Mode

When you enter playmode, a green button appears in the top of the FileBrowser inspector, letting you show/hide the browser.

