FastGUI 1.1 Manual

Online Updated version

Intro

Description:
When you're focused on programming in Unity 3D, the last thing you want to do is stop what you're doing and end up wasting a lot of your time trying to organize and exactly adjust your Game/GUI to how it was already planned and done in Photoshop by the artists. It's like having to do it all over again. With FastGUI you won't have to, because all that time-consuming process will be entirely automatized, and with only 2 simple commands your Photoshop file will be perfectly set up inside Unity 3D, allowing you to focus your time on what really matters: programming your App/Game. After it's set up, you'll have everything you need inside Unity! You can then alter and organize everything the way you want.
***You need NGUI to use this plugin**
http://forum.unity3d.com/threads/114833-NGUI-(Next-Gen-UI)-demo-amp-final-feedback-request

Features:
- Perfect placement of images from Photoshop to Unity.
- Keep the hierarchy of your files organized in Unity just like on Photoshop.
- You can choose where and when to use it in your project.

- Up to 10 times faster than making it by hand.
- Create UIWidgets automatically:
    - Buttons
    - Sliders
    - Progress Bar
    - Dynamic Texts
    - Dynamic Inputs
    - Keep it simple: After the import process is done, everything will become purely Unity3D Objects, so you won't need to keep editing your files in other softwares.

Changelog

1.0 - First Version

1.1 - Completly Remake

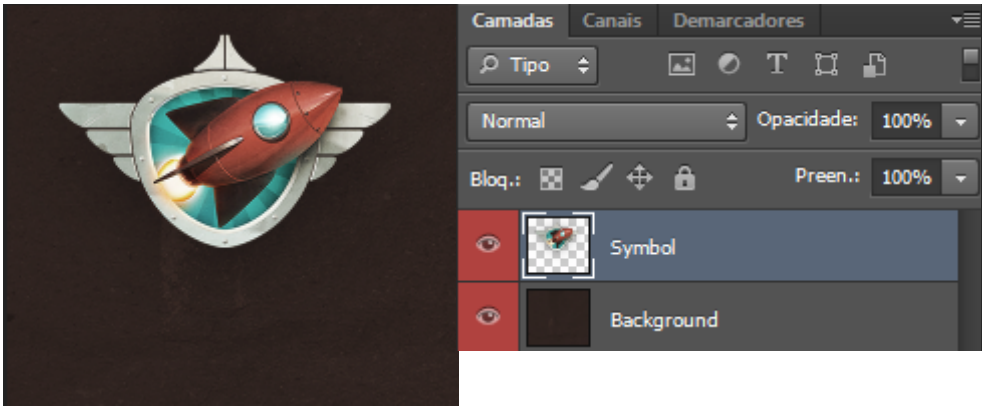
1.2 Update

- New Sprite exporting Methods
- Linkage between Photoshop and Unity3D
- New Widgets Added
    - UISlicedSprite
    - Clipping
    - Sliced Image Button
- Improved Panel Feedback
- Minor Bug Fixes
- Now you can have multiple objects using the same sprites

Under the Hood

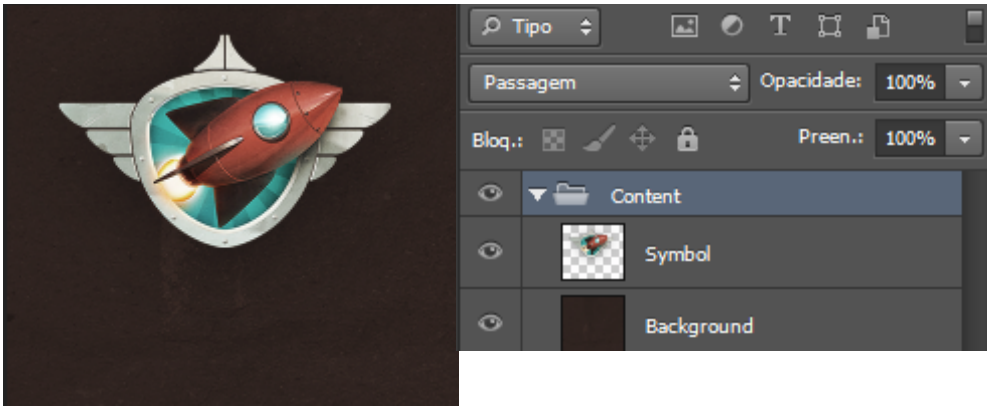
To understand how FastGUI works, you first need to understand how we define the workflow to export our Photoshop files, based on that, think about it this way: To facilitate your files organization, FastGUI works only with folders (Groups) from Photoshop. Each group you create will in the end be exported as a single *PNG file, and everything else that's not in a folder will be ignored.

In this PSD:

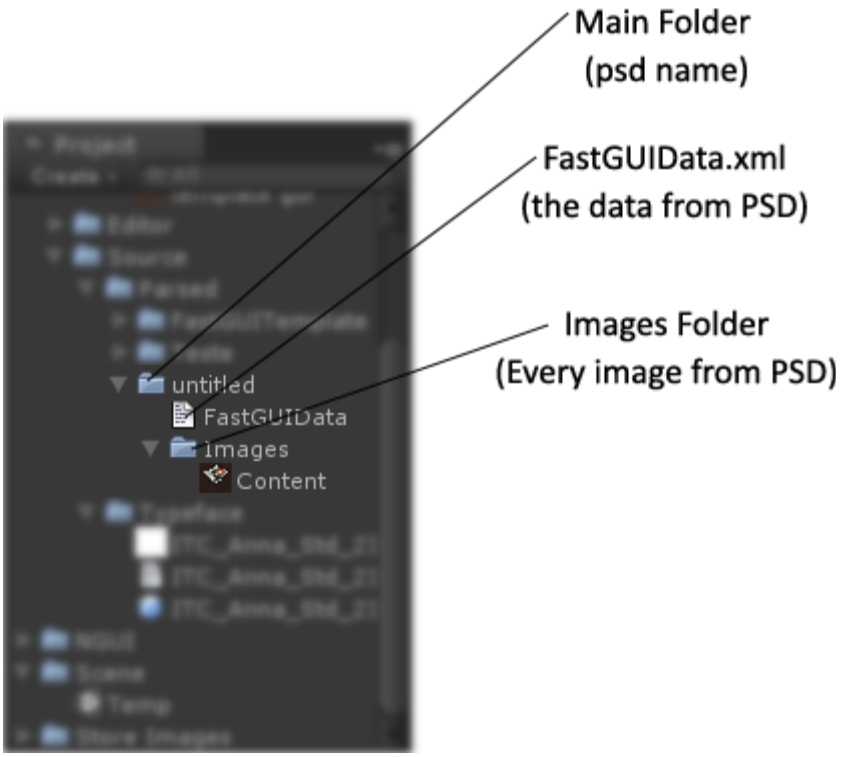


Notice that the layers are not inside any folders, so if FastGUI tries to export a file like this, it won't find any layer groups to work with, discarding everything, meaning that no *PNG files will be exported as a result.

However if we place the items in a layer group, like this:

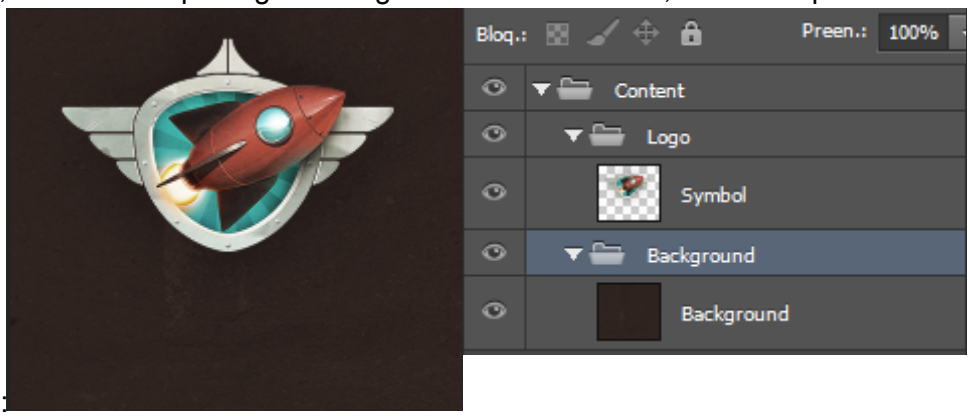


The result on the FastGUI exporter will be the following:

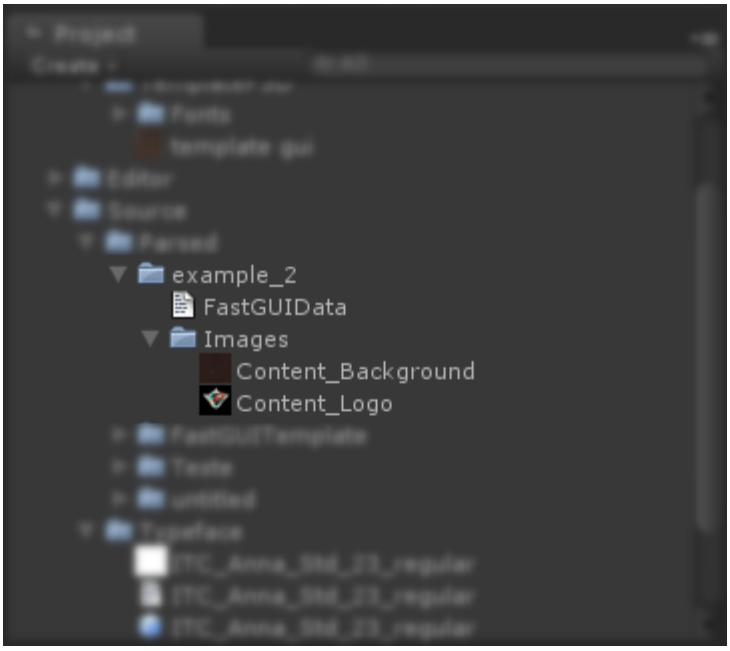


The folder **untitled** contains the exported data from FastGUI, consisting of two parts: **FastGUIData.xml**, which contains all the information related to layers, widgets to create and positions, and a folder named **Images**, where all the all the images that have been exported are stored. In this first case, the **Images** folder only contains one file, because the *PSD file that generated these files has only one **LayerGroup** with two images, which have been merged into a single *PNG file.

If we, instead of exporting the images in the same folder, did it in separated ones:

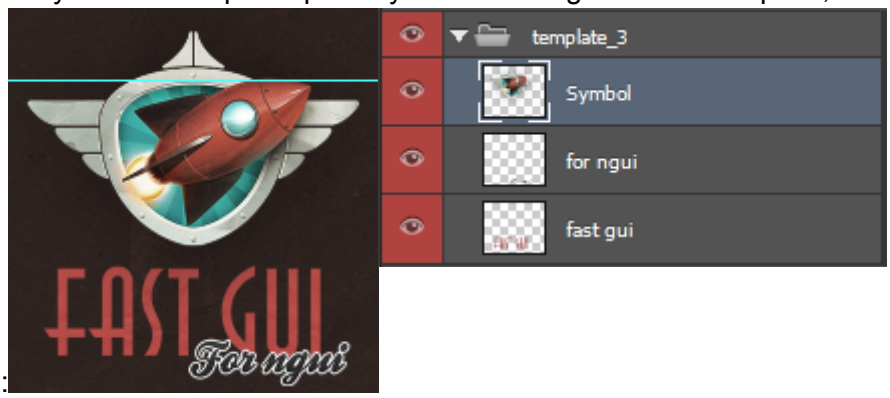


The result on the folders would be this:

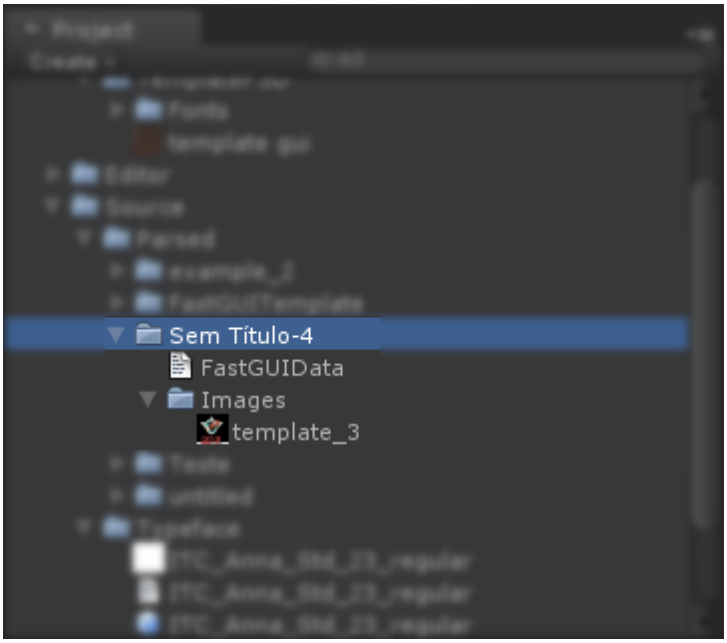


Two objects exported separately, meaning two distinct sprites when mounting on Unity. Another interesting aspect is that to grant that different images are not overwritten by mistake, their names on the file will always be composed by **Group Parent_Group**.

The best way to picture this is thinking that every **Folder** is a **GameObject.**

This way we can keep complex layers in our original Photoshop file, such as this one: 

This one will result in a single image called template_3, containing a single merged PNG of the three layers above:



Another very important part of FastGUI is how the layer groups are named. Every group that is not a simple sprite needs to have a prefix to its name, this is done to differentiate which type of widget needs to be created.

The following are the prefixes used by FastGUI:

- "**ibtn_**" is used for creating UIImageButtons. Other than the prefix, there is also the need for there to be at least one sub-layer group with the name of "**idle**", containing the image of the button while it is idle. Other sub-groups may also be added, containing the names "**hover**" and "**pressed**":

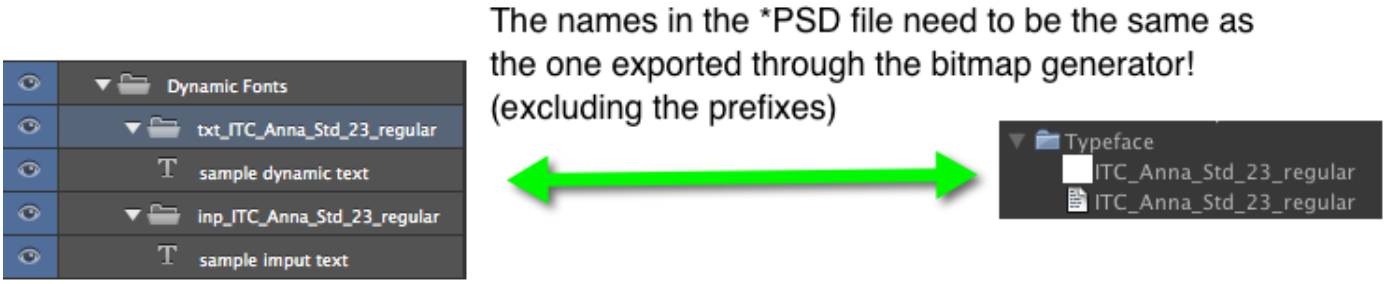
- "**ckb_**" is used for creating UICheckBoxes. Just like image buttons, there needs to be sub-groups for the widget to be created appropriately. The first sub-group should be named "**background**", containing the checkbox's image for when the button is not checked. The second group needs to be named "**checkmark**", and it should contain the "tick" of the checkbox:

- "**sld_**" is used for UISliders. The sub-groups it needs to contain are named "**thumb**", describing that the contained sprite is the slider's draggable border, "**background**" should be the slider's sprite when the drag value is 0 and "**foreground**" should be the sprite for when the drag value is 1:

- "**pgsb_**" is used for creating progress bars. The process of creating a progress bar is very similar to the one of a slider, but there is no need for a "**thumb**" sub-group.

- “**txt_**” is used to create labels, dynamic fonts that aren't editable by the user
- “**inp_**” is used to create text that can be editable by the user, equivalent to UIInput in NGUI.

Following both of these prefixes, should be the name you gave the font after using a bitmap generator, such as Glyph Designer for the Mac:



- “**slc_**” is used to create UISliced Sprites, which are sprites that can be resized using the 9 slice scaling concept. To better understand this concept see this page.
- “**clip_**” is used to create a clipping area. This prefix needs to be used in conjunction with another child layer named as **“clipping-area”**, that serves to determine the size of the clipping area. Any content inside this layer will be clipped based on the clipping-area size.

Your first project with FastGUI

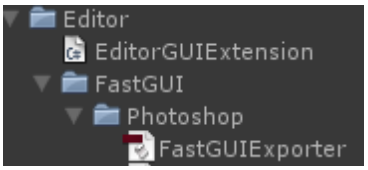
In order to have a first experience with FastGUI, here goes a step by step for you to test it with the Package that goes with the PSD.

1 - The first thing you should do is install the font used in Photoshop, in the folder: Demo/Fonts. We got two fonts:

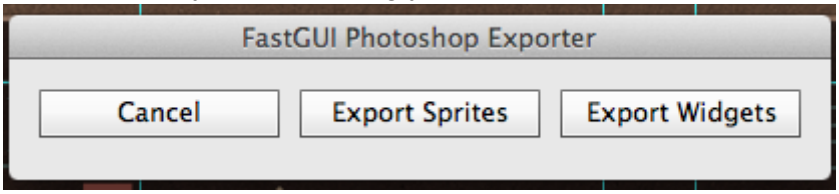
Instal both on your PC/MAC.

2 - Open the file **template gui** on Photoshop: 

3 - Run the file MonsterJuice/FastGUI/Photoshop/**FastGUIExporter:**

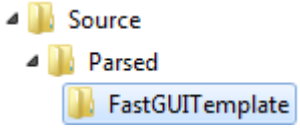


4 - Select the type of exporting you wish the script to perform:



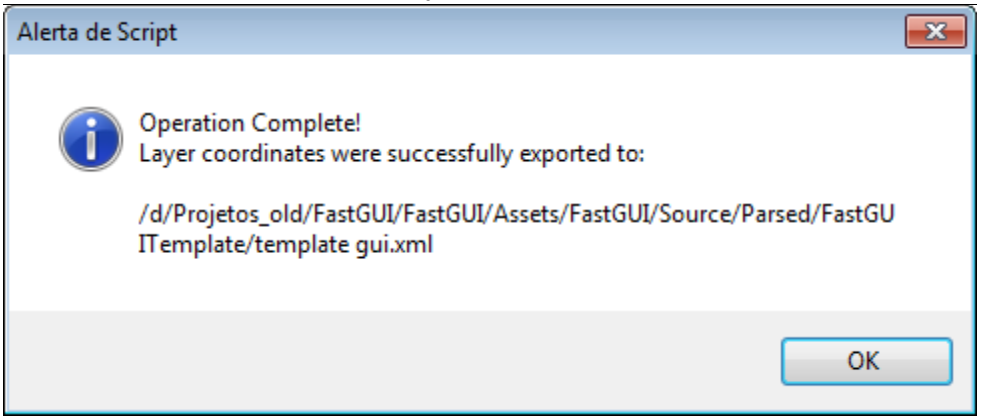
Export Sprites will export all the images of the *PSD file as sprites only, with no functionality whatsoever, while **Export Widgets** takes advantage of all the features available in FastGUI, exporting each **LayerGroup** into different types of widgets, depending on the affixes used on each group.

5 - Select the folder where you want to save the exported files in.



Click OK and wait for the script to read all the Photoshop items, get their positions and save them in files.

Once the operation is complete, you should see a window similar to this:

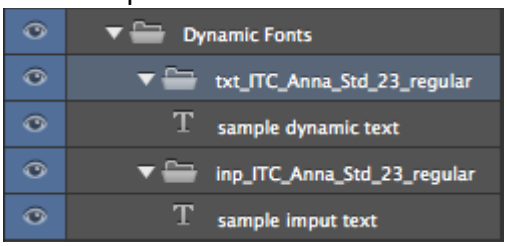


6 - The result after opening Unity will be this:

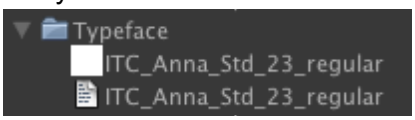
All of the Photoshop **groups** are exported as **images**.

7 - Now we need to add the necessary fonts to the project. This process is the same as on NGUI for creating fonts.
You can do this using whatever Bitmap Font Generator software you like.

Photoshop:



Unity3D:

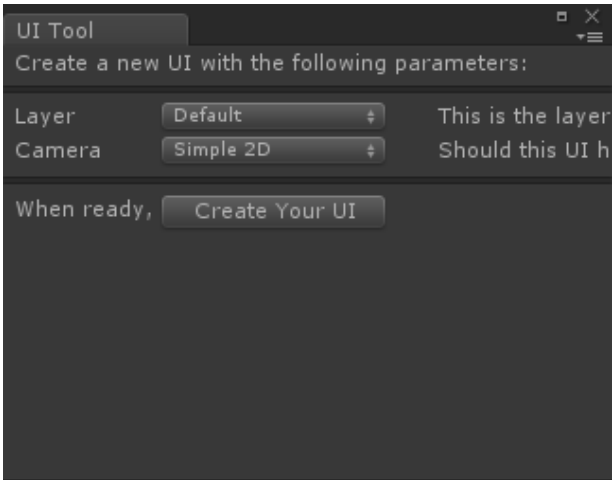


Export the font in .**txt** format into a folder called “Typeface” anywhere in the project, as you can see above the font must have the same name of the layer in .PSD (the layer name doesn’t need to be exactly the real font name).
There are some prefixes required to work with fonts:

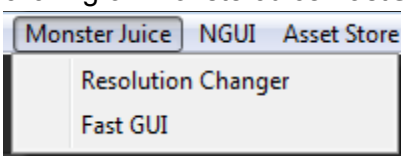
- **txt_** - is used to create labels, dynamic fonts that aren't editable by the user
- **inp_** - is used to create text that can be editable by the user, equivalent to UIInput in NGUI.

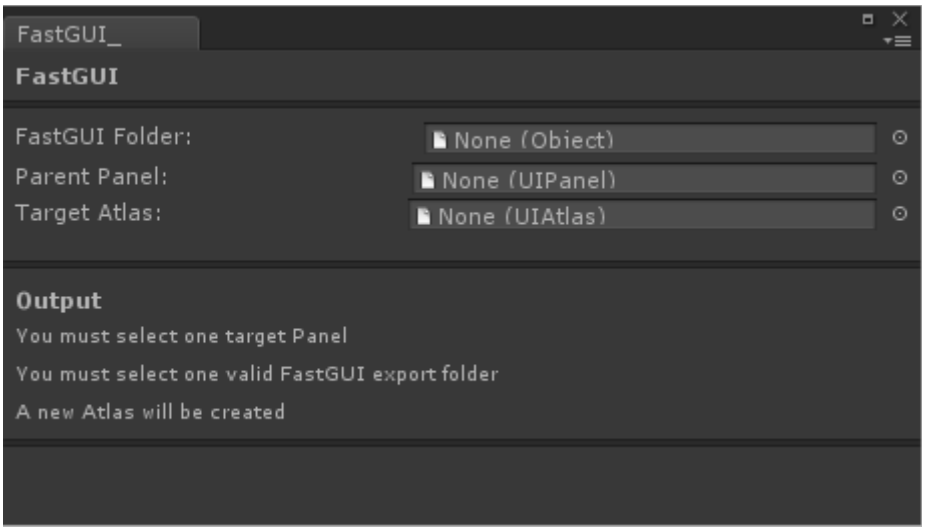
You can also create a text as Sprite, to do this just don't include the prefix of the layer name in Photoshop.

8 - Create a new UI on NGUI:



9 - Now it is important that you check the size of the exported PSD. Open the FastGUI panel by clicking on MonsterJuice/FastGUI on the upper menu:





It's now important that everything in this window is explained. Onto **FastGUI Folder**, you must drag the resulting root folder from the Photoshop exporting. See the example on the gif below:

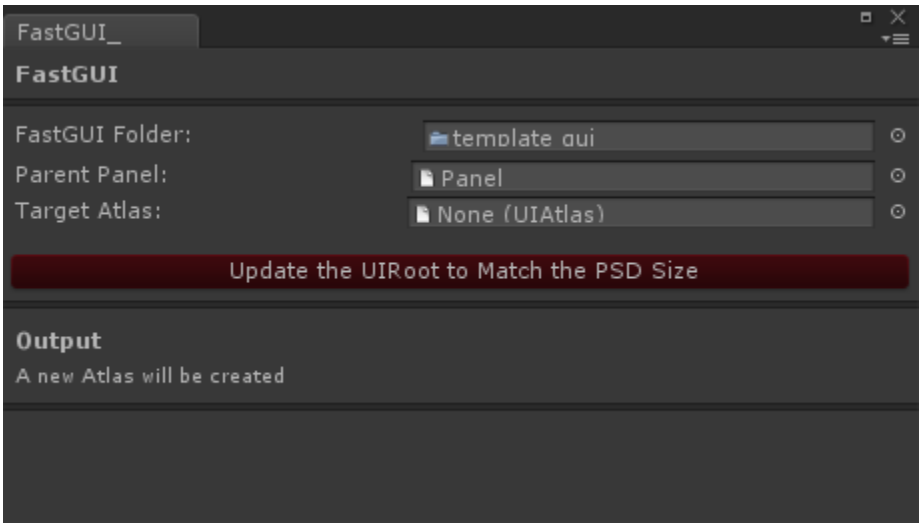
Example Gif

On **Parent Panel**, you will do the same, but instead of the root folder, you should put in an UIPanel.

On the **Output** part of the window you'll have every information about what's ok and what's missing.

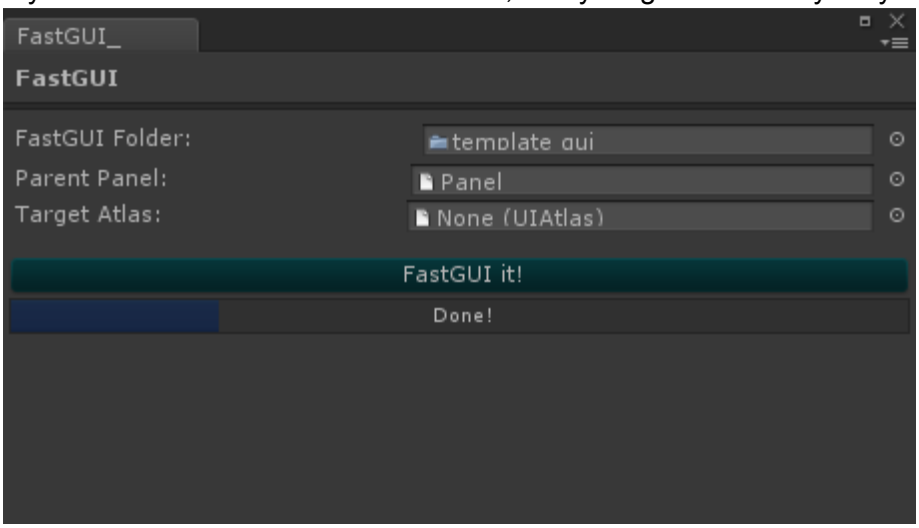
Target Atlas - If you want the images to be added to a specific atlas you can set it manually, if not, FastGUI will create one automatically with the same name of the PSD.

10 - In case your **UIRoot** is different from the size of the image you'll process, a button like the one on the image below will be available, then the only thing you'll have to do is press it to adjust the UIRoot to the settings needed.



It is important to ensure that the parent panel is positioned on 0,0,0 and with 1,1,1 scale!

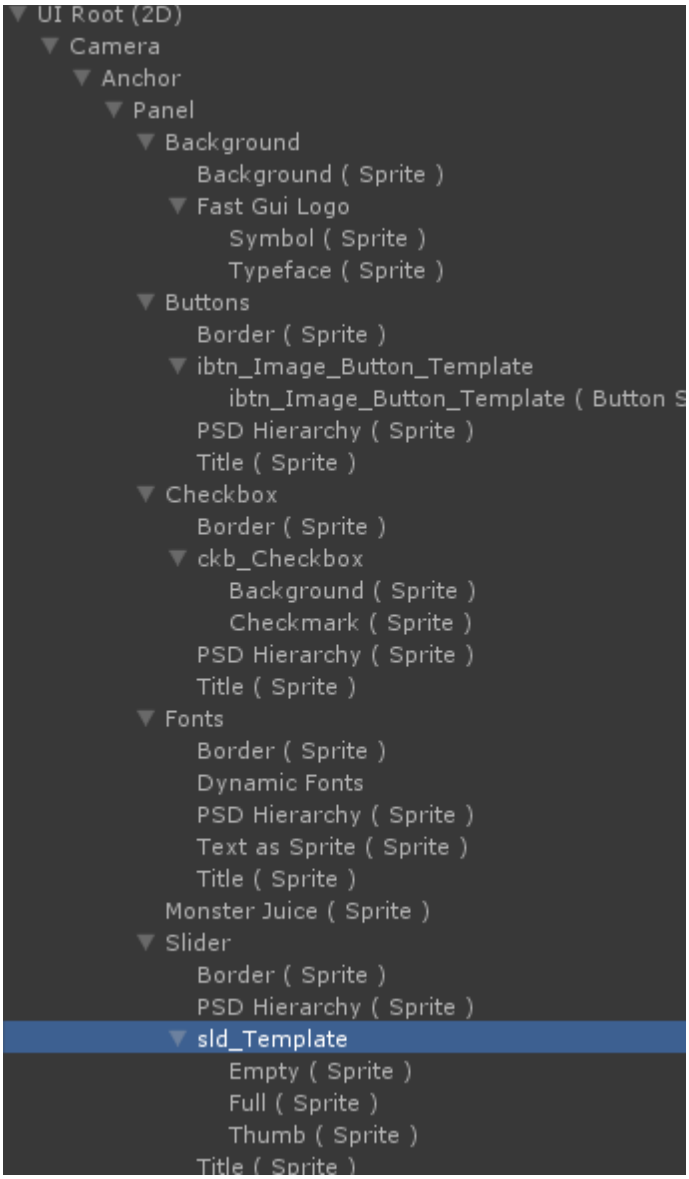
11 -In this case, since we have no Atlas created, if we leave it blank, a new one will be created. If you can see the "**FastGUI it!"** button, everything will be ready for you to process your folder.



12 - Now click on Parsee and wait. This process can be a little slow, mainly for the creation of the Atlas and the adding images to it.

This is the result after the parsing is done:


FAST GUI
For ngui
BUTTONS
IMAGE BUTTON
Button
CHECKBOX
CHECKBOX TEMPLATE
X
SLIDER
FONTS
TEXT AS SPRITE
MONSTERJUICE.COM.BR


UI Root (2D)
Camera
Anchor
Panel
Background
Background (Sprite)
Fast Gui Logo
Symbol (Sprite)
Typeface (Sprite)
Buttons
Border (Sprite)
ibtn_Image_Button_Template
ibtn_Image_Button_Template (Button S
PSD Hierarchy (Sprite)
Title (Sprite)
Checkbox
Border (Sprite)
ckb_Checkbox
Background (Sprite)
Checkmark (Sprite)
PSD Hierarchy (Sprite)
Title (Sprite)
Fonts
Border (Sprite)
Dynamic Fonts
PSD Hierarchy (Sprite)
Text as Sprite (Sprite)
Title (Sprite)
Monster Juice (Sprite)
Slider
Border (Sprite)
PSD Hierarchy (Sprite)
sld_Template
Empty (Sprite)
Full (Sprite)
Thumb (Sprite)
Title (Sprite)

Acceptable Widgets

Atualmente os Widgets aceitados são os seguintes:

F.A.Q

- **Why is the file's path contained in the name of the *PNG file?**

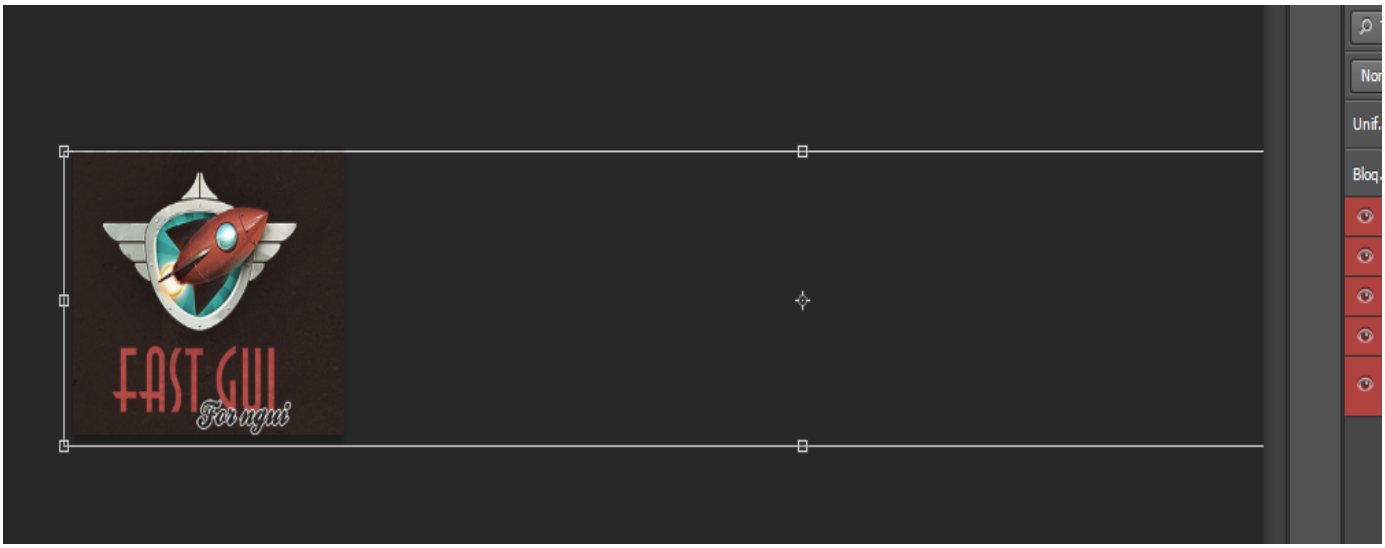
The file's path is a way to ensure that no image is ever overwritten when mounting the file on Unity, if that validation did not exist, Unity would consider two images with the name **"background"** the same thing, and would only place one of them in the atlas.

- **Can I use a pre-existing Atlas?**

Absolutely. Although you must first indicate the Atlas to which you want the images to be included, on the **Target Atlas** field on the **FastGUI Panel**.

- **My image is not positioned correctly in Unity but it is on Photoshop.**

One of the problems FastGUI might encounter when exporting Photoshop images is when Clipping Masks are used, or when the images are bigger than the PSD's file canvas size, as an example:



As you can notice here, the background image is much bigger than the PSD's canvas size which I'm trying to export from, and the anchor point on the image is completely wrong. A simple way to check that is **Selecting the Group** which has that problem and then using **Free Transform** on it, that way you'll check if the anchor point is wrong for any reason.

- **I got an error on the Photoshop exporter**

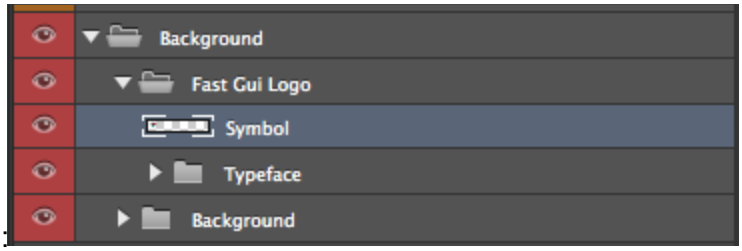
Special characters are one of the biggest problems to how FastGUI works. Any special character in any Groups, Folders or PSD names will result in an error when you try to save the file. Always keep an eye on that.

- **The dynamic fonts are not showing on Unity after the Parsee!**

You have to be really careful when creating the font file, because FastGUI will look for a folder called Typeface on the project's hierarchy. Inside that folder it will try to find two files, one .txt and a .png and they must have the exact same name of the PSD file set on Photoshop.

- **I'm getting an error of nested prefabs! (You should never nest widgets! Parent them to a common game object instead. Forcefully changing the parent.)**

This problem happens when you have objects like this on your Photoshop:



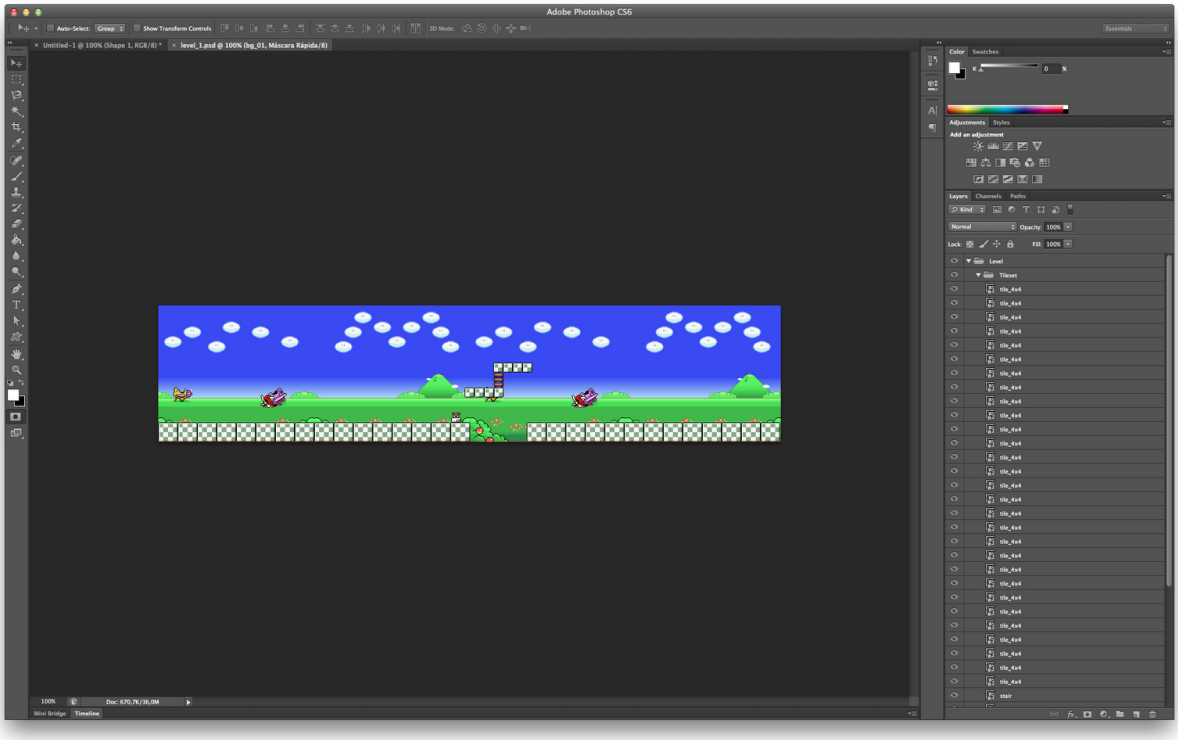
As you can see, inside the **FastGUI Logo** folder, we have a loose **LayerArt**, and also inside the folder there's also another folder! In this case, the **FastGUI Logo** folder is being converted to **Sprite** by **FastGUI** and the **Typeface** folder is also being created as a sprite, and the **Typeface** sprite is organized hierarchically inside the **FastGUI Logo** folder.

- **My image is diferent from what i export in my PSD File**
  Somehting to keep in mind when working with FastGUI are the atlas. If you have two images with the same name, the last one will overwrite the first, so always check if there are no images with the same name.

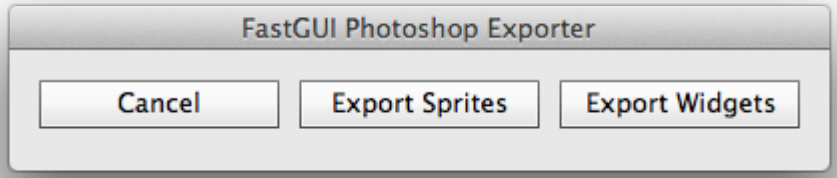
Understand the Sprite exporting method

Since FastGUI 1.2 the sprite exporting method has changed after some requests, we understood that this is much better for a lot of you guys, and to demonstrate it we have created a new example.

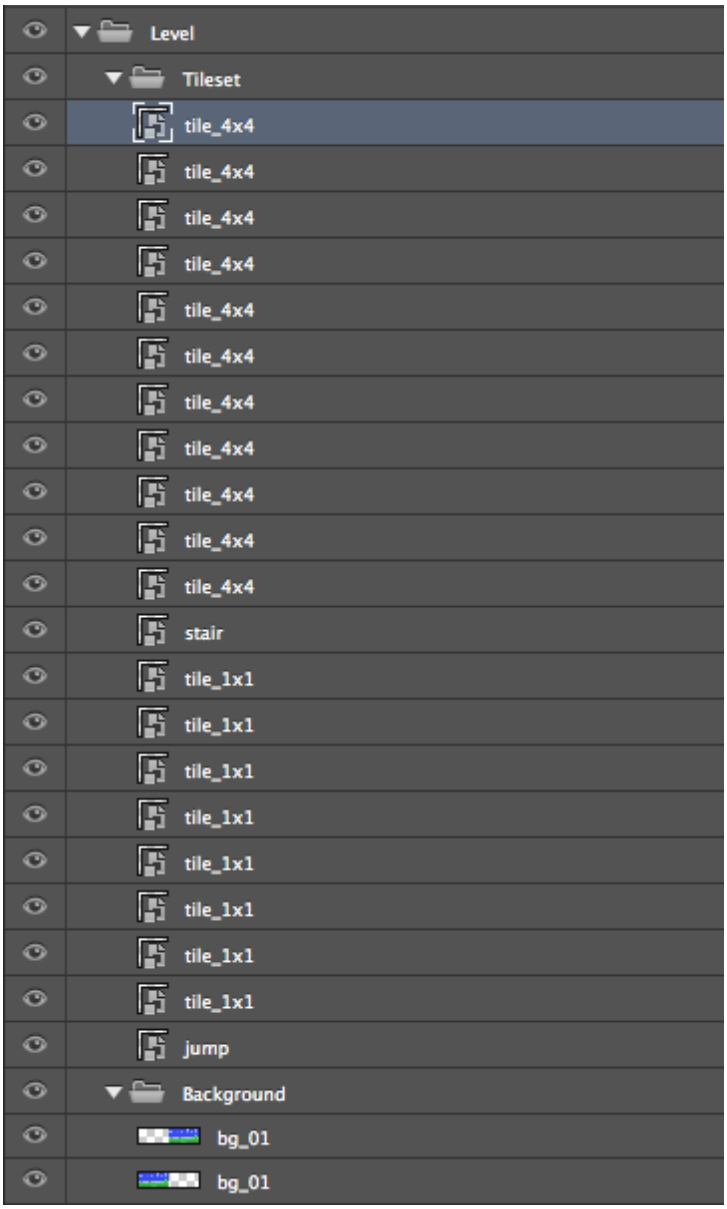
Open the **level_1** example on the folder: **Template Level/level_1.psd**



Now it's important that we explain a few points about the **Export Sprites** method.



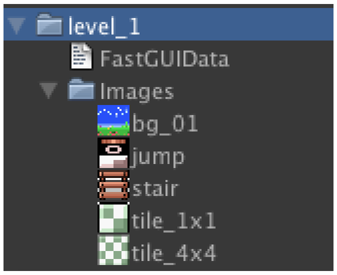
Each Photoshop group/folder will represent a blank GameObject when it's passed on to FastGUI. This is very good to keep things organized, and each **layer** will generate a sprite, like on the following example:



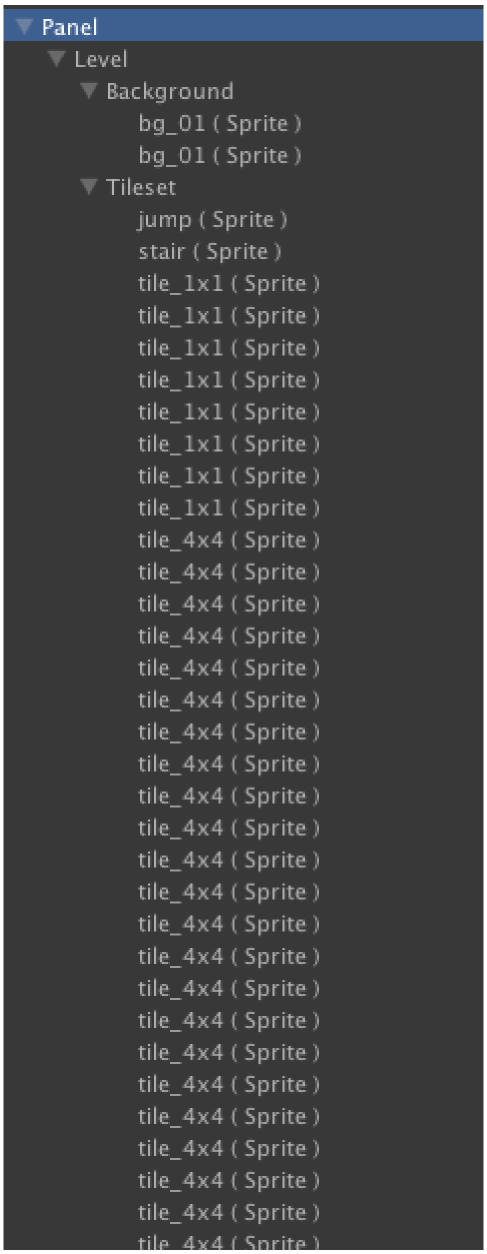
The output will be something like this:

- Panel
    - Game Object (name: Level)
    - Game Object (name:Tileset)
        - -Sprites (name: tile_4x4)
        - -Sprites (name: tile_4x4)
        - ...
    - Game Object (name: Background)
        - -Sprites (name: bg_01)
        - -Sprites (name: bg_01)

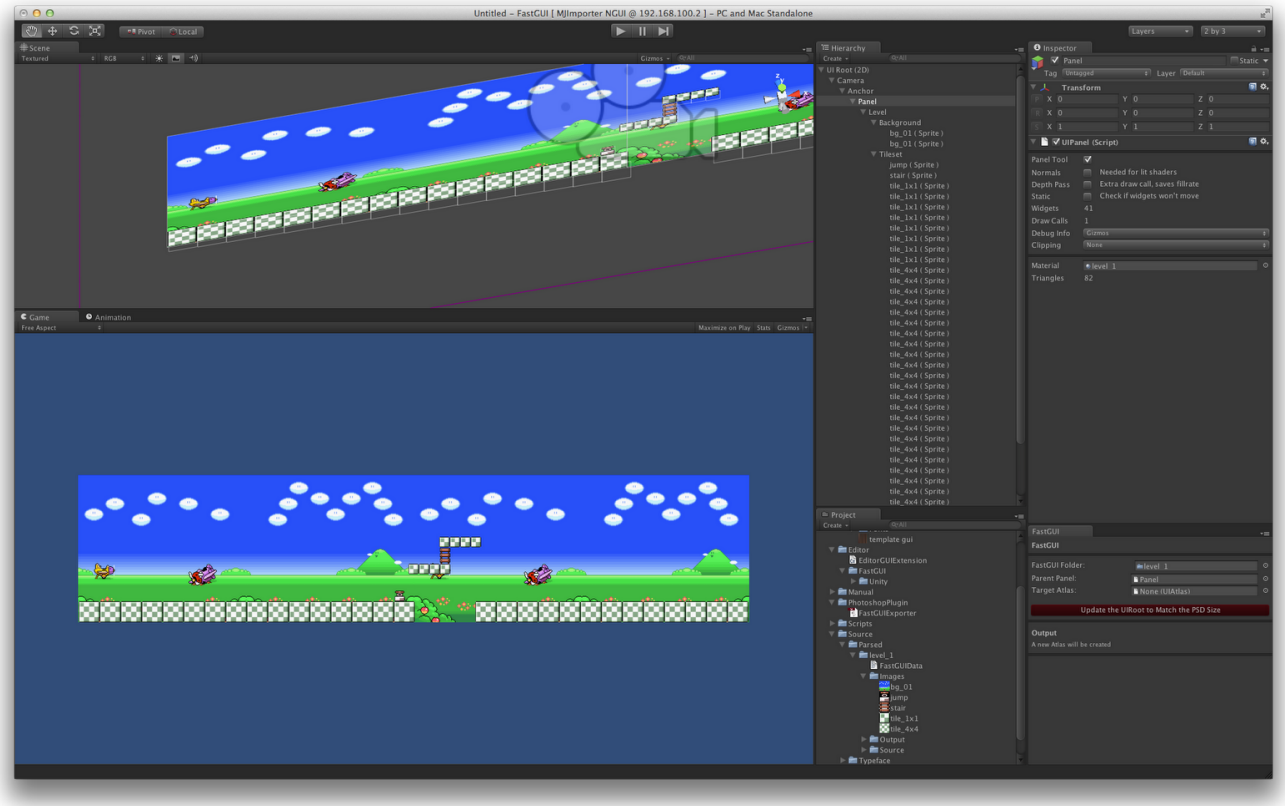
Now let's take a look on the Parsee result after using **Export Sprite**:



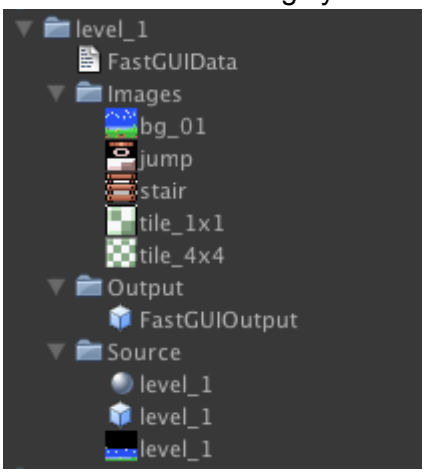
This already represents one of the best features on the new FastGUI version. Now we only export different named things. The entire level is here, so now let's parse this using FastGUI in Unity3D:



And this is the result! All the objects are placed exactly as they were on the PSD, and we got the entire level done in unity in just a few clicks:

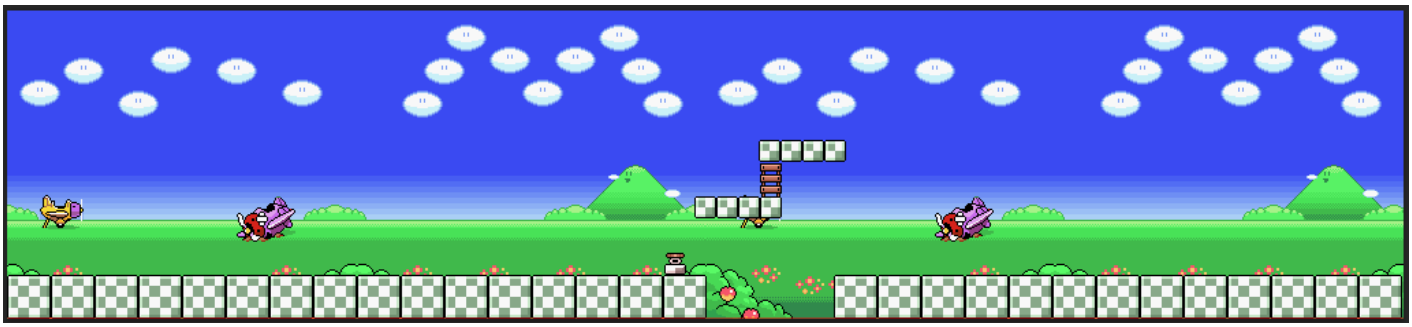


This is one of the things you can notice that were changed from the previous version:



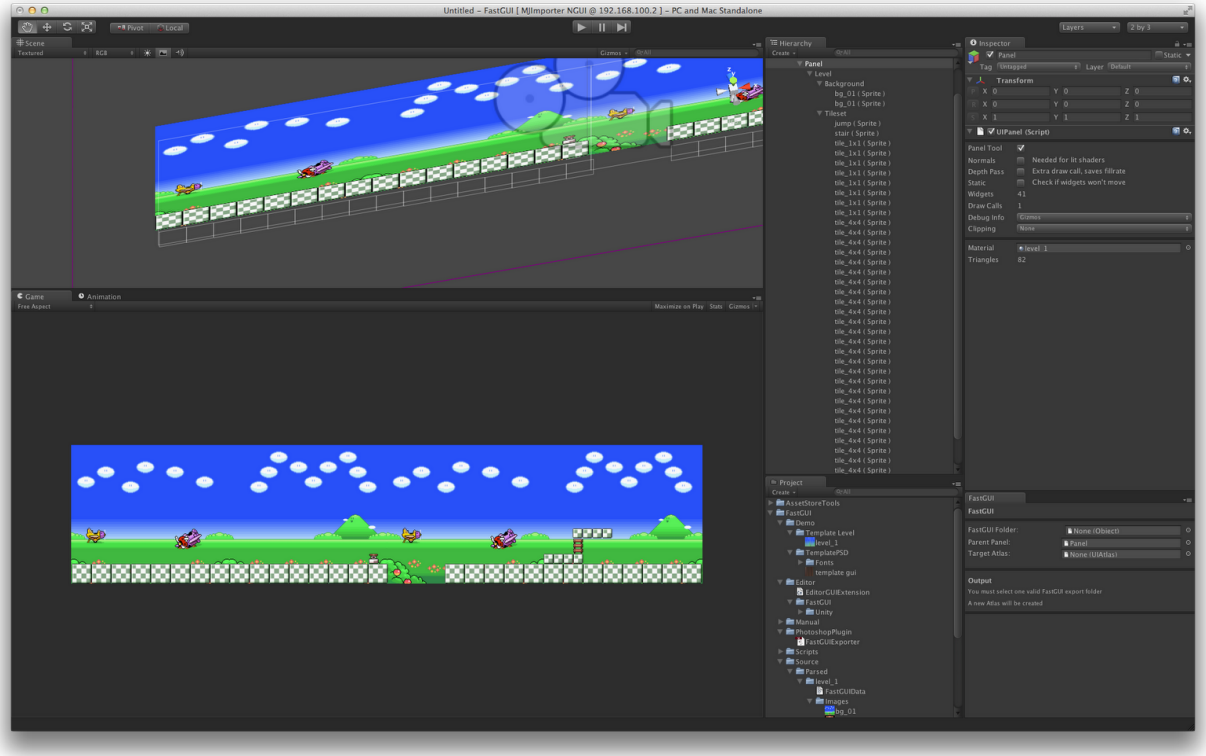
Now, besides the **Source** folder, we have a new one, called **Output**, and this is one of the best things about this version, because every single object FastGUI creates is linked to its specific layer on Photoshop, meaning that if you edit your PSD file and export it again, it will be already updated.
So, if we move things on Photoshop like this:

Before:



After:

Export the the PSD again, and just reimport the object on **FastGUI**



Isn’t it amazing? Simply because of the newly added **Output** folder, the references between **Photoshop** and **Unity** are kept, allowing you to edit everything on Photoshop, and still keep your objects in Unity with any custom scripts, components and anything else you need and the names can be the same!