A big **thank you** for purchasing my

I hope you find this pack useful to create a great game!

If you have any questions, please contact me here**.**
Please make sure to include your **invoice number**. Thank you!

ricimi designs

Halloween GUI Pack Terms of Use & License

The Halloween GUI Pack can only be used under the standard Unity Asset Store End User License Agreement. A Copy of the Asset Store EULA is available here.

The copyright of the Halloween GUI Pack and all of its contents belongs to ricimi ©.
After purchasing the Halloween GUI Pack, you have the right to use it only for the purposes of developing and publishing a game.

You are NOT allowed to redistribute or resale the Halloween GUI Pack or any of its contents for any purpose. To distribute or resale this product is NOT permitted under any circumstances and is strictly prohibited.

Thank you for respecting my work.

Documentation

The Halloween GUI Pack is a customizable mobile-friendly game UI pack containing many elements that can be used to create a complete game with a creepy halloween style. While the sprites themselves do not depend on any specific Unity version, the accompanying demo project requires **Unity 2019.4.0** or higher.

Asset structure

After importing the asset package into your Unity project, you will see all the resources provided live in the **GUIPack-Halloween** folder. This folder is further subdivided into the following subfolders:

- **Demo:** Contains all the assets of an example project that makes use of all the sprites included the pack via Unity's UI system. Please note the demo and its accompanying source code are only intended as an example showcasing what you can build with this game UI pack. Feel free to use it as a starting point and extend it as you see fit for your own game.

- **Documentation:** Contains this documentation.

- **Editor:** Contains one color swatch.

- **Settings:** Contains the original *TagManager.asset* file that contains the layer information. This is **important**, because the demo project contained in this GUI pack makes **extensive use of custom sorting layers** in Unity to define the order of the UI elements on the screen.

- **Sources:** Contains .psd sources and .svg files of the sprites included in this pack. The .svg files are very useful if you need to re-export the sprites from Photoshop to have a bigger size.

- **Sources/MockupsPSD:** Contains the original art source files in a layered Photoshop format for any customisation you may need.

- **Demo/Sprites:** Contains all .png files used for building the demo.

Some notes regarding the demo project

This game UI pack contains a complete demo project with full C# source code that you can use as a starting point for your own game. While the source code is not intended to be a universal framework, it can be a very useful reference when it comes to learning how to approach the implementation of a game UI using Unity's built-in UI system.

All the scenes in the demo project make use of Unity's Canvas to display their contents. The Canvas render mode is set to *Screen Space – Camera* and the canvas scaler is set to *Scale With Screen Size* UI scale mode. This, together with extensive use of anchors when positioning UI elements, makes it possible to automatically scale the UI across multiple resolutions (please note we have optimized the demo for panoramic aspect ratios).

You can find more details about how this works in the official Unity documentation here. Note that I am using a **reference resolution of 1280x1920**, which works well across a wide range of aspect ratios. This is particularly useful for mobile development, where screen sizes vary wildly between devices.

We have tested the demo using a **resolution of 540x960** in the game tab.

Prefabs and Canvas

The prefabs are UI elements that are intended to be positioned as **children of a Canvas object**; trying to place them outside a Canvas will usually produce scaling differences like the ones you mention. Additionally, if using the prefabs outside the context of the provided demo, you want to make sure the canvas settings are the same as those in the demo.

Sorting layers in the demo project

The demo project contained in this GUI pack makes **extensive use of custom sorting layers** in Unity to define the order of the UI elements on the screen.
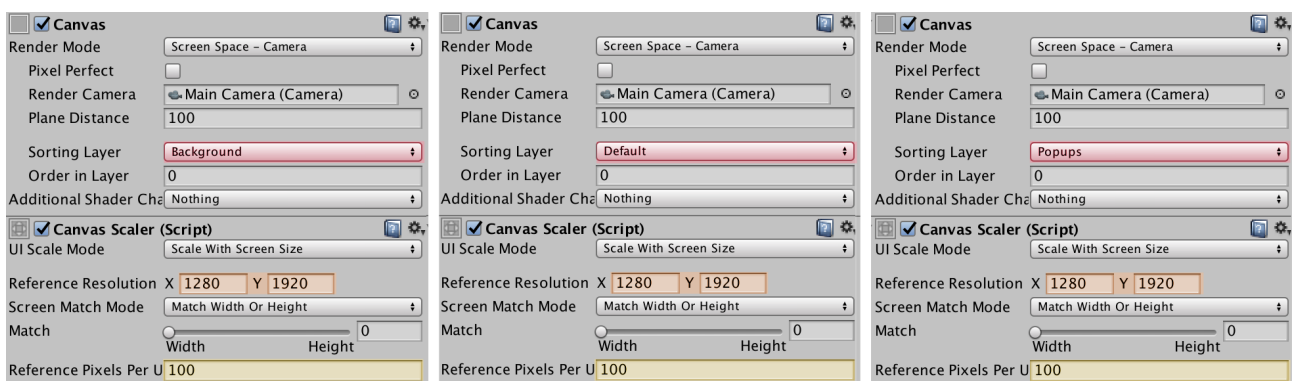
When generating the final package to submit to the Asset Store, Unity does not export this custom sorting layer information. The end result is that, after importing the GUI pack into a new or existing Unity project, you may see the rendering order is wrong (e.g., the background or the particles may be on top of everything else). This is only due to the fact that the custom sorting layer is not available.

In order to alleviate this situation, We have provided the original *TagManager.asset* file that contains the layer information in the *Demo/Settings* folder.

With Unity closed, you may go to your project path and replace the *TagManager.asset* contained in the *ProjectSettings* directory with the one provided by us. After re-launching your project, you will see the rendering order is as intended.

Important: if you have imported the Halloween GUI pack into an existing project that already defines its own set of custom layers, this will replace your layers with the ones in our GUI pack! Proceed with caution.
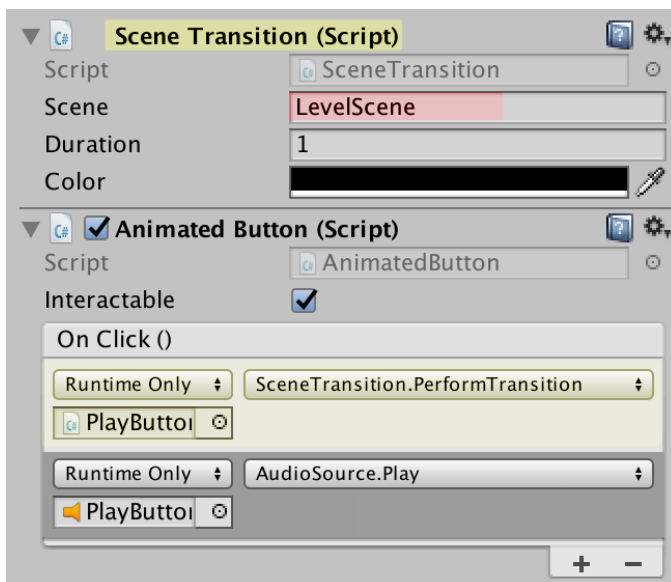
The *SceneTransition* component

The demo project makes extensive use of Unity built-in UI features, but also provides some useful extensions. Two of the most notable ones are the **SceneTransition** component and the **PopupOpener** component.

The *SceneTransition* component provides functionality to transition from one scene to another. Using it is very simple. Consider the following example, where we have a *PlayButton* in the home scene that should transition to the levels scene when clicked.

We only need to add a *SceneTransition* component to the *PlayButton* button game object:

The **SceneTransition** component carries out the logic needed to smoothly fade out from the current scene into the new one.

You can specify the destination scene name and the duration and color of the transition.

Note how the very same *PlayButton* calls the *SceneTransition*'s *PerformTransition* method in order to start the transition when clicked. You can make this call in code, too.
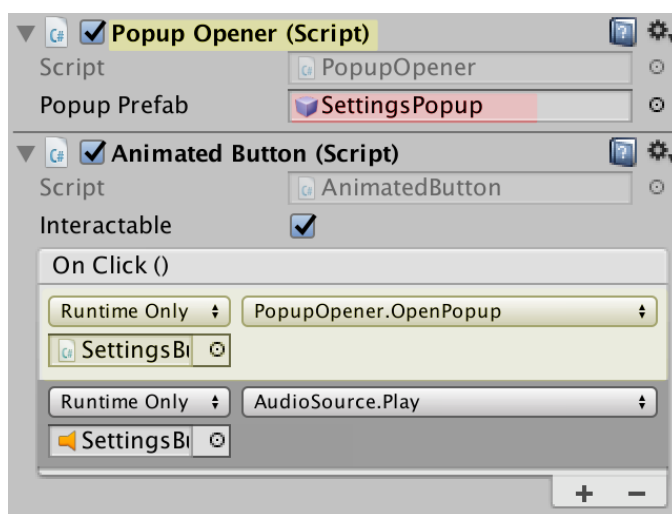
The *PopupOpener* component

The **PopupOpener** component provides functionality to open a popup and darkening the

background behind it. Using it is, again, very simple. Consider the following example, where we have a *SettingsButton* in the home scene that should open a settings popup when clicked:

1. We need to add a *PopupOpener* component to the *LoginButton* game object. The *PopupOpener* component carries out the logic needed to open a popup in the current scene.

2. We need to add a *Popup* component to the *Popup* prefap to open (a game object that contains a *Popup* component).
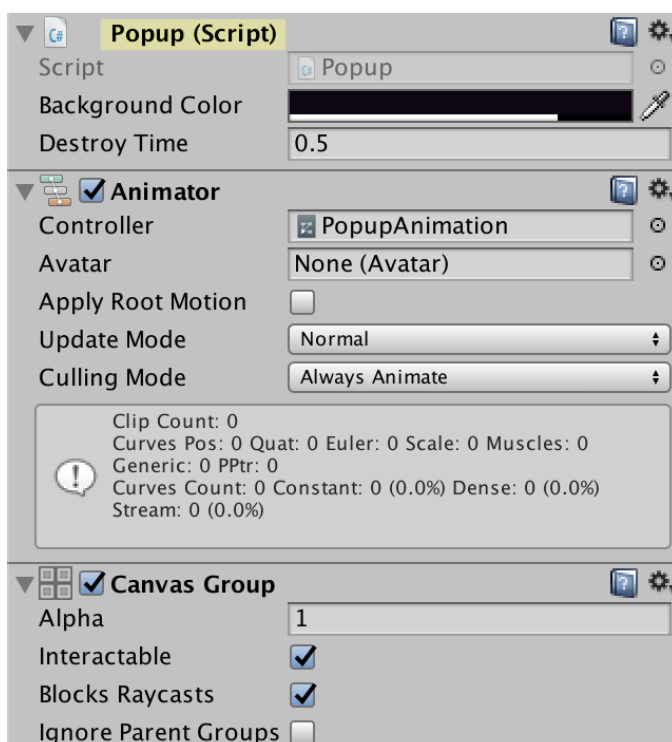
Note how the very same Settings*Button* calls the *PopupOpener*'s *OpenPopup* method in order to open the popup when clicked. You can make this call in code, too.

We also need to add the Animator with the popup animation to the popup prefab. The canvas group is used to perform a fade out animation while closing.

That's it! Thank you. :)

Halloween GUI Pack

© **game**vanilla

www.gamevanilla.com

· ·

Follow us:

@gamevanilla   @ricimiart