A big **thank you** for purchasing our

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

If you have any support questions, please contact us **here**.
Please make sure to include your **Invoice number**.

The **Clean & Minimalist GUI Pack** can only be used under the
Unity Asset Store License Agreement which you can find here.

ricimi
designs

Clean & Minimalist GUI Pack

The **Clean & Minimalist GUI Pack** is a customizable, mobile-friendly game UI pack containing many elements that can be used to create a complete game with a nice clean and minimalist look. While the sprites themselves do not depend on any specific Unity version, the accompanying demo project requires Unity 2018.4.0 or higher.

Asset structure

After importing the asset package into your Unity project, you will see all the resources provided live in the GUIPack-Clean&Minimalist 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 the documentation of this pack.

- **Sprites:** Contains the .png files as well as .psd and vectorial .svg files of all the sprites included in this pack. The vectorial .svg files are very useful if you need to re-export the sprites to have a bigger size or if you want to tweak them.

- **UI-Artwork-PSD:** Contains the original art source files (in Photoshop format).

Some notes regarding the demo project

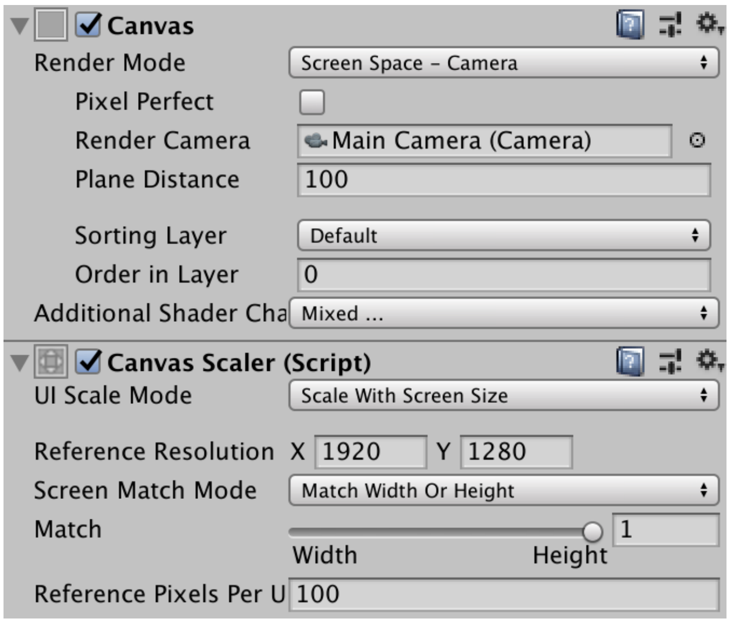
This game UI pack contains a complete demo project with full C# source code that you can use as a starting point for your own game. While the source code is not intended to be a universal framework, it can be a very useful reference when it comes to learning how to approach the implementation of a game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

Reference resolution

Please note that we are using a reference resolution of 1920x1280, which works well across a wide range of aspect ratios. This is particularly useful for mobile development, where screen sizes vary wildly between devices.



Canvas ✓

Render Mode	Screen Space – Camera
Pixel Perfect	☐
Render Camera	Main Camera (Camera)
Plane Distance	100
Sorting Layer	Default
Order in Layer	0
Additional Shader Cha	Mixed ...

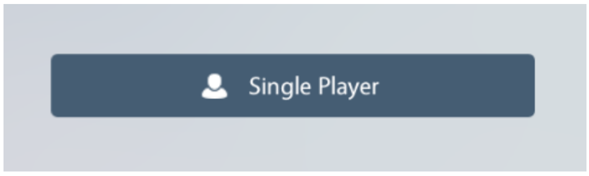
Canvas Scaler (Script) ✓

UI Scale Mode	Scale With Screen Size
Reference Resolution	X 1920 Y 1280
Screen Match Mode	Match Width Or Height
Match	1 (Width – Height)
Reference Pixels Per U	100

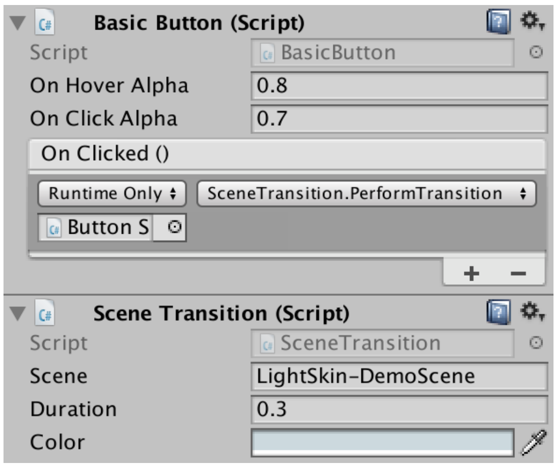
The *SceneTransition* component

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 *Single Player* button in the home scene that should transition to the levels scene when clicked:



We only need to add a *SceneTransition* component to the *Single Player* button game object.



Basic Button (Script)

Script	BasicButton
On Hover Alpha	0.8
On Click Alpha	0.7

On Clicked ()

Runtime Only — SceneTransition.PerformTransition — Button S

Scene Transition (Script)

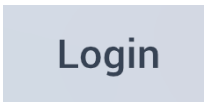
Script	SceneTransition
Scene	LightSkin-DemoScene
Duration	0.3
Color	

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.

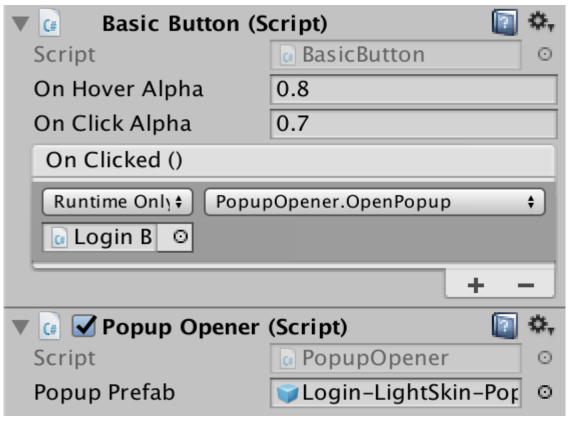
Note how the very same *Single Player* button calls the *SceneTransition*'s *PerformTransition* method in order to start the transition when clicked. You can make this call in code, too.

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 *Login* button in the home scene that should open a login popup when clicked:



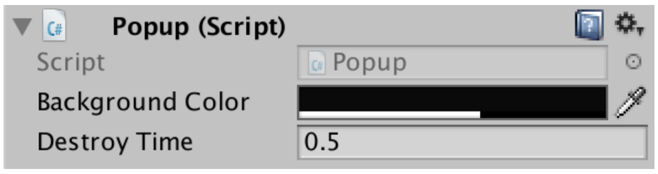
1. We need to add a *PopupOpener* component to the *LoginButton* game object. The *PopupOpener* component carries out the logic needed to open a popup in the current scene.



Basic Button (Script)
Script: BasicButton
On Hover Alpha: 0.8
On Click Alpha: 0.7
On Clicked ()
Runtime Only | PopupOpener.OpenPopup
Login B

Popup Opener (Script)
Script: PopupOpener
Popup Prefab: Login-LightSkin-Pop

2. We need to add a *Popup* component to the *Popup* prefap to open (a game object that contains a *Popup* component).



Popup (Script)
Script: Popup
Background Color
Destroy Time: 0.5

Note how the very same *LoginButton* calls the *PopupOpener*'s *OpenPopup* method in order to open the popup when clicked. You can make this call in code, too.

Thank you for supporting us by buying this pack. You're helping us to create **more assets** and **exciting products** for game developers.

Clean & Minimalist GUI Pack

©gamevanilla

www.gamevanilla.com

Follow us:

@ricimidesigns @gamevanilla