PLATINMODS.COM

MOD MENU MAKER FOR UNITY

Peek-A-Boo

IMPORTANT NOTICES

Introduction

As more and more visitors of our site asked for a tutorial on mod menus, we decided to develop an easy-to-use library that does all the necessary calculations and coding for the modder, without requiring C# or Unity knowledge.

Conditions of Use

The only permitted use of this tool is the one explained in the attached documentation and in the relative videos, beyond which no person is allowed to share or alter in any way the libraries, their source code, related videos or documents. This tool was created by Peek-A-Boo for exclusive distribution on Platinmods.com: every generated menu has to keep the proper credits.

Support Disclaimer

No support will be given to questions that could find answer in the documentation, already asked, off-topic or below the needed requirements of the tool. In any other matter, the author reserves the right not to give support if the workaround is concerning Unity's libraries, game's source or dnSpy and not the menu itself; this includes cases in which the menu is not visible due to missing code parts in the game's libraries.

Features

- easy and ready to use tool
- quick referencing thanks to self-generated toggles
- guided positioning and layout creation
- automatic reshaping with cross-device compatibility
- built-in styles with customization options
- support for string in any (human) language
- future updates will not require any recoding
- other minor additions, like dragging the open button

Requirements

- Unity modding experience
- being familiar with dnSpy environment

Preliminaries

Before attempting to make any menu, it is important to know already what can be modded in the chosen game: edit the source as if making a regular mod and write down any change for later use. It is also good practice to test the cheats, not to have to rework the menu.

ATTACHED FILES

PMT.MenuMaker_test

The testing file is the only library to reference when editing the game's source in dnSpy. It contains the most important members needed by the release dll and a couple testing methods, used to find candidate classes.

PMT.MenuMaker_release

The release library is the one to include in the final step of mod menu creation, right when testing the game and the cheats. It has never to be referenced from dnSpy: use for that purpose the other given one.

Code.cs

The source code necessary to be added into the game's source, it can be edited with any text editor to be copied and pasted in the target class.

Samples

Some files to show how the menu can be customized. See also the next section.

CODE DETAILS

Initialization

MakeGUI.PreInit and MakeGUI.Init are necessary parts of the menu creation process: PreInit ensures that any mistake in the initialization is avoided and Init completes that process.

Strings

The menu first strings to customize are the opening and closing one, called openString and closeString, which meaning is easy to understand. The close button's auto-placement is done considering the length of closeString: "Close" and "X" produce different layouts.

Labels are the only strictly needed customization, as the menu won't work without any label. It is possible to add as many labels as needed, the menu will be recalculated to best fit them. For each label will be created a boolean toggle, to be used to switch the cheats.

The credits for the mod can be written inside creditString, which will be displayed at the bottom of the menu.

Positioning

The menu can be repositioned by setting menuXPos and menuYPos, which are (x;y) coordinated calculated from the top left corner of a 1280x720px screen. For precision works, it is possible to load a screenshot of the game into any graphic editor and find the desired position; otherwise just use an approximate value, as the menu will be moved if needed: it will always be as close as possible to the set numbers, but will never be placed offscreen.

The open button can be moved using openXPos and openYPos.

The close button can also be moved with closeXPos and closeYPos, but, if those are not set, it will be placed considering the menu position, to always be nearby it.

Giving -1 to any position value will make it be calculated automatically.

Resizing

The menu is automatically resized to best fist the length of the labels and their number, so that all of them are always visible. It is possible to change with buttonSize the maximum height of toggles, in a range from 40 to 60, where 50 is the default value. If there is not enough space, a lower height will be used.

Style

The menu has 4 built-in styles, that can be set with menuStyle from 0 to 3. Each of those styles is by default solid, but can be turned into semi-transparent using menuAlpha, a float value between 0.5 and 1.

It is also possible to not have space between the toggles by setting hasMargin to false.

GUI

MakeGUI.OnGUI is the real method that builds the menu, it is entirely calculated by the MenuMaker library.

MENU CREATION SUMMARY

Load all game's assemblies into dnSpy.

Find MonoBehaviour classes looking for Start, Awake or Update methods.

Place in a few of those classes TestGUI.Spawn(n,"text"), referencing the test PMT.MenuMaker.

Add the modified assembly and the test PMT.MenuMaker to the game.

Launch the game and check which buttons are always visible in menu, pause and all game modes.

Once found and picked one of the candidates, refresh the assemblies.

Add the menu code, obtained from the code.cs attached to this release.

Write the buttons labels and optionally some settings for the menu.

Link the modding methods to the menu through GetGUI.toggles[n], referencing the test PMT.MenuMaker.

Save the produced assembly and include it in the game together with the release PMT.MenuMaker.

Test the game and the effect of toggles.