MENU KIT $_{1.2}$

Documentation

This package was developed by Wilgner Fábio

Thanks to my friend Paulo Camacan for making the UITransition script

Contact

wilgner.fabio@gmail.com

Made with <3

Summary

Please leave a review on Assets Store

This gives feedback to me on what to add / improve and others.

Thanks.

MMK (Main Menu Kit)

Introduction

The Menu Kit is a package that contains a complete menu for your game, is simple, flat and easy to customize to your taste, contains 4 ready menus, **play**, **options**, **credits** and **exit**. All working perfectly and with animations included, plus a dialog box, languages, transitions, animations that can be done easily with a few clicks. Functional options and is saved in **json** file.

Features

- Demonstration scene
- Easy to use and customize
- Includes options settings (VIDEO, AUDIO and GAME)
- Language Control (EN, PT_BR)
- Audio System
- Custom Input (Suggestion sent in my Email)
- Pause Menu (RESUME, OPTIONS and EXIT) (Suggestion in the presentation video of the Main Menu Kit)
- Simple Player
- GUI flat and beautiful
- 2 Backgrounds
- Animation Transitions (Fade In, Fade Out, Swap In, Swap Out, Custom)
- Save settings to JSON file
- Custom Cursor
- 2 play type (Start direct play and game modes, level select under development)
- Animations in the Main Menu

- Scale animations, directions or both on the credits screen and in game modes
- Organized structure
- All C # code and commented in English (soon Brazilian Portuguese)
- Prefabs ready for use
- Custom Editor
- Organization of menus and modes according to position in the inspector
- Documentation in PDF
- Improvements in inspector and codes
- Work with Unity +2017

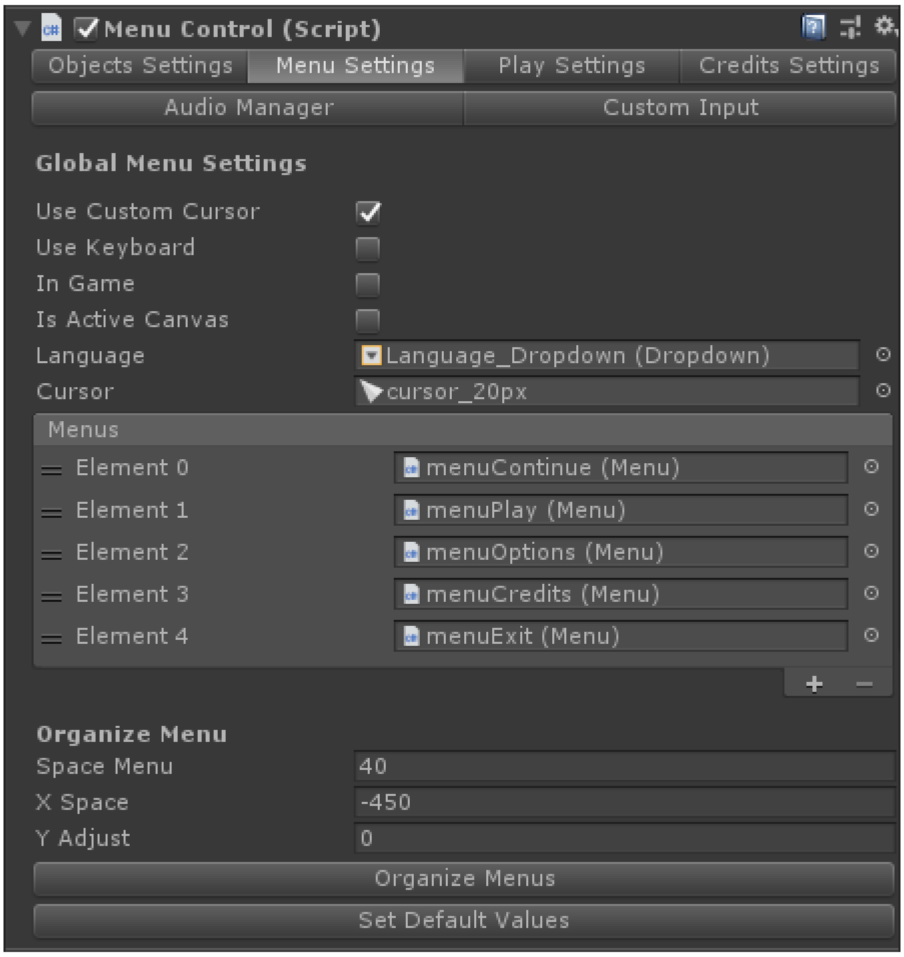
What to do first?

Before importing the Kit menu I recommend you make a backup in your project and should create a new scene for the menu or create a new project to follow, execute the settings and not lose the project.

After that, you will not appear in the menu, next to the **Wilgner's Studio > MenuKit > Add > Main Menu** window or press CTRL + SHIFT + M with your own already running menu.

You can choose the prefabricated patterns above and make changes so you do not have to make 0.

Cofiguring the Control Menu



Menu Control (Script)

Objects Settings | Menu Settings | Play Settings | Credits Settings

Audio Manager | Custom Input

Global Menu Settings

Use Custom Cursor	✓
Use Keyboard	
In Game	
Is Active Canvas	
Language	Language_Dropdown (Dropdown)
Cursor	cursor_20px

Menus

Element 0	menuContinue (Menu)
Element 1	menuPlay (Menu)
Element 2	menuOptions (Menu)
Element 3	menuCredits (Menu)
Element 4	menuExit (Menu)

Organize Menu

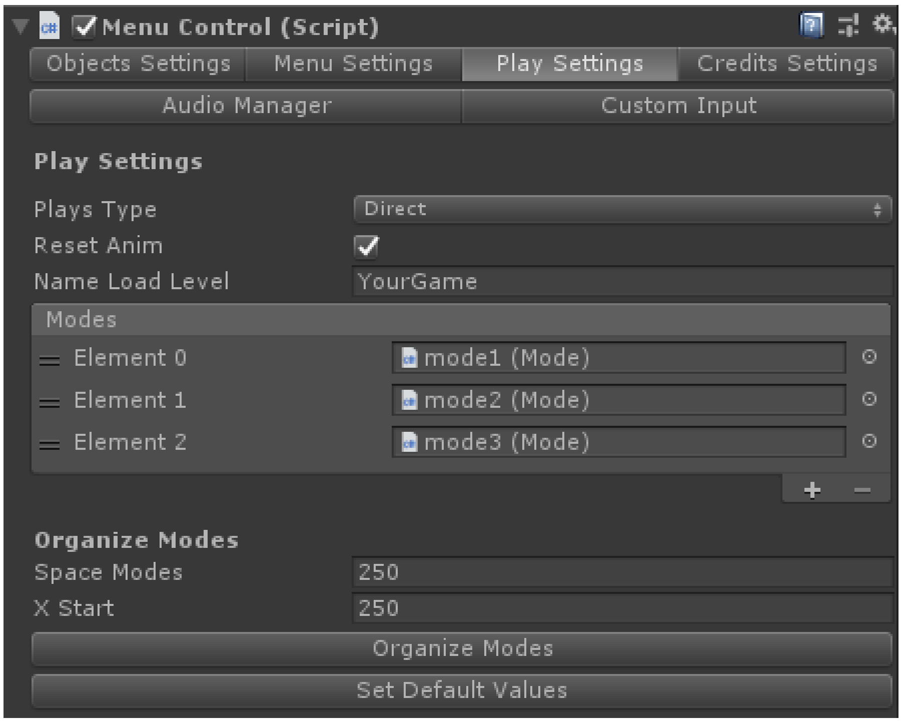
Space Menu	40
X Space	-450
Y Adjust	0

Organize Menus

Set Default Values

(In Order)

1. Use a custom cursor
2. By checking this option you can move through the menu using the keys (in development, it works only in the main menu)
3. Check if this is the pause menu
4. Dropdaw for change language
5. Image shown on the cursor if 1 is enabled (I recommend 15-20px size)
6. Menus, you can arrange as you want and then click on "Organize Menus"


Menu Control (Script)

Objects Settings | Menu Settings | Play Settings | Credits Settings

Audio Manager | Custom Input

Play Settings

Plays Type	Direct
Reset Anim	✓
Name Load Level	YourGame

Modes

Element 0	mode1 (Mode)
Element 1	mode2 (Mode)
Element 2	mode3 (Mode)

Organize Modes

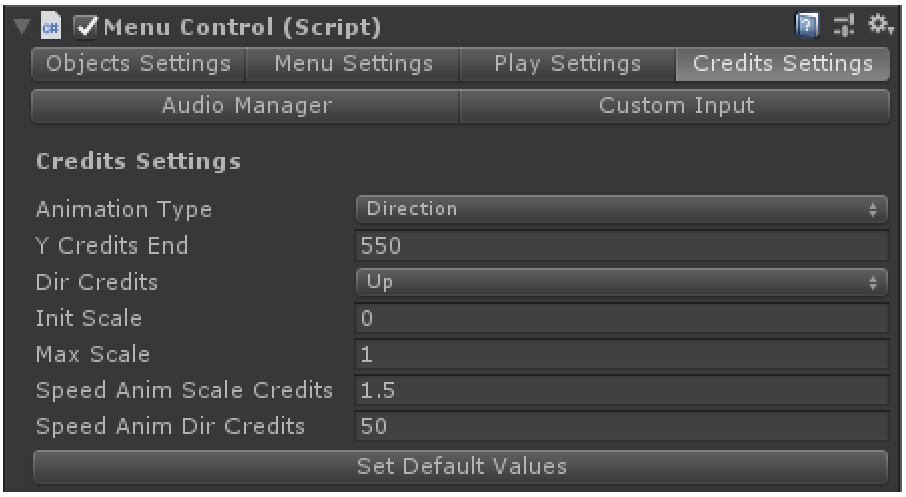
Space Modes	250
X Start	250

Organize Modes

Set Default Values

(In Order)

1. Type of plays, can be direct (game starts directly), Modes (if you have game mode, example: singleplayer, training, casual)
2. Restart the animation modes every time you go to that menu
3. Name of the scene to be loaded
4. Just like the menu, you can organize the modes and click on "Organize Modes"

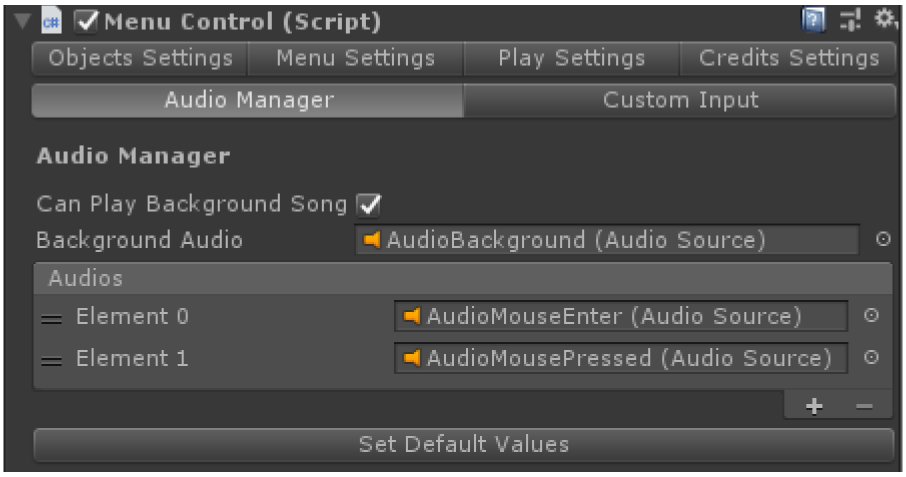


Credits Settings

Animation Type	Direction
Y Credits End	550
Dir Credits	Up
Init Scale	0
Max Scale	1
Speed Anim Scale Credits	1.5
Speed Anim Dir Credits	50

(In Order)

1. Type of animation (Direction: Like a movie; Scale: Can be static or scale the size)
2. Position in which the credit ends
3. Direction it will move (by modifying here it is important to change item 2 as well)
4. Initial Size
5. Max Size
6. Speed (Init Scale to Max Scale)
7. Speed at which credits move (50 unit unity)

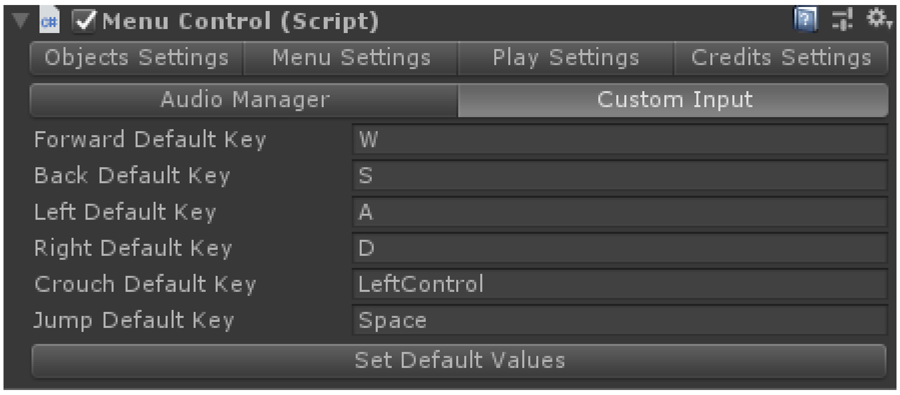


Audio Manager

Can Play Background Song	✓
Background Audio	AudioBackground (Audio Source)
Audios	
Element 0	AudioMouseEnter (Audio Source)
Element 1	AudioMousePressed (Audio Source)

(In Order)

1. Background audio can play when you start the game / menu
2. Audio when playing when 1st is marked
3. List of Audios that the menu uses (can add one for screen change for example)


Menu Control (Script)

Forward Default Key	W
Back Default Key	S
Left Default Key	A
Right Default Key	D
Crouch Default Key	LeftControl
Jump Default Key	Space

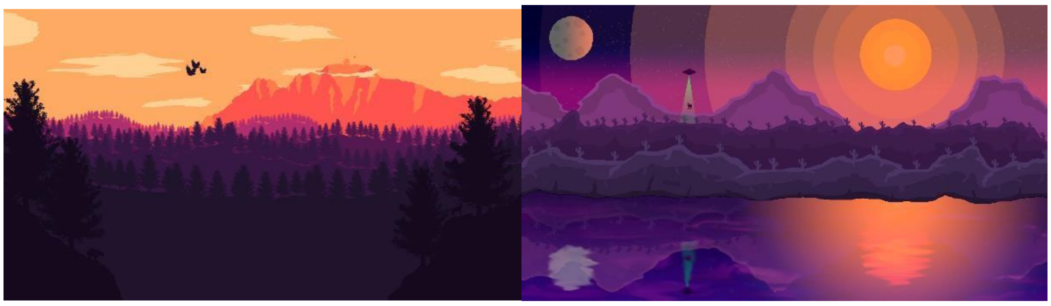
Custom input controls (basic)

Changing the Background

First let's change the background, it's nothing less than an image, its path is:

MenuKit > Canvas > MenuKit_Canvas > backgroundImage

Select the Source Image in inspetor and select the desired image, I created two flat backgrounds:



You can delete / deactivate the background image to use a (like PUBG, Overwatch).

Basic Settings

Basic settings is where the title and version are all screens (except for credits) contains these same settings, just access it and change in the inspector or add an image instead of the text.

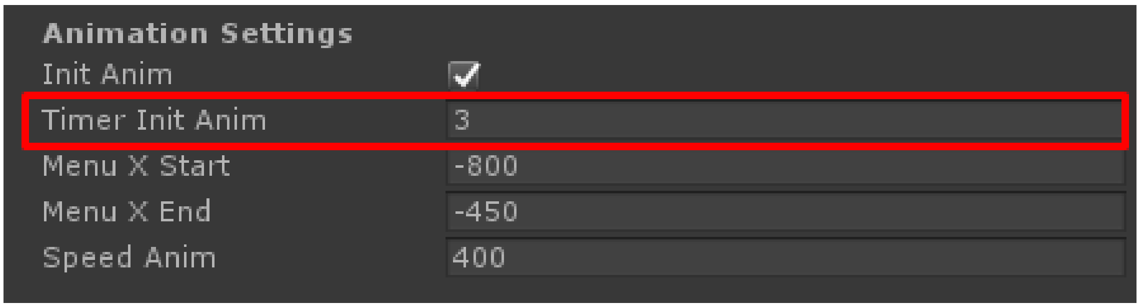
I chose to repeat the elements to have greater control in case you want to animate them or make other configurations.

Adding a menu

Do you want a new menu? No problem, to get the process you can go through the tab **Wilgner's Studio** right click in the hierarchy and add a New Menu or take the prefab and drag until **MenuKit > Canvas > MenuKit_Canvas > MainScreen > Menus**, you can also copy and paste a ready menu.

Now in the MenuControl click on more, add (drag) the new menu created and click on "Organize Menus".

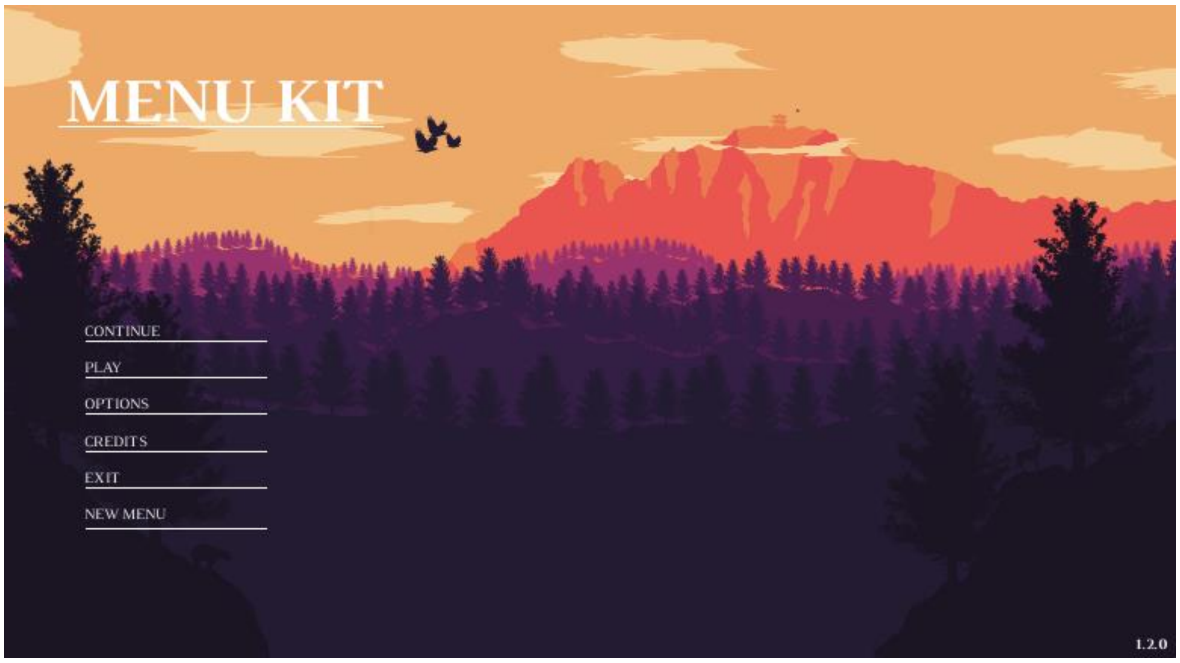
In this example menu I am using the initial animation of 0.5 of difference, so in the new menu of time being 2.5 you change to 3 and so on and after that you can change in the TEXT inspector.



Animation Settings

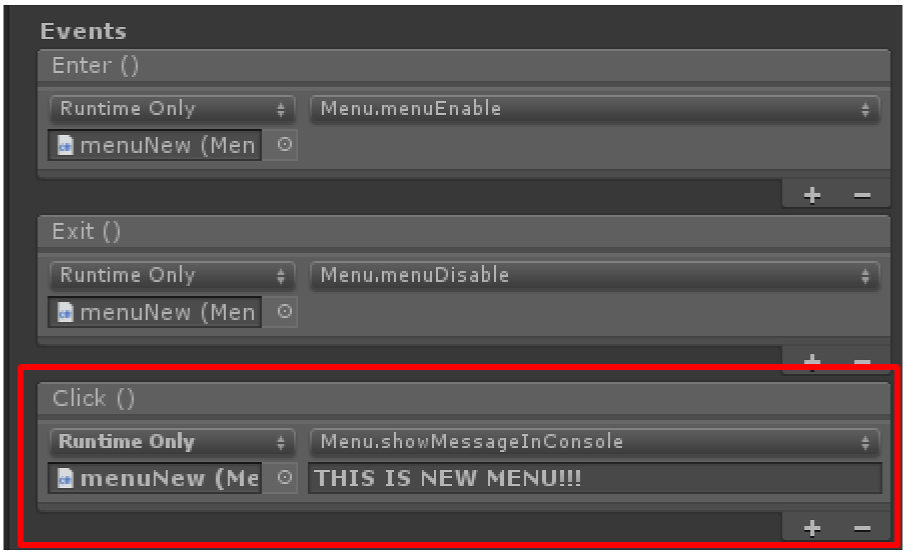
Init Anim	✓
Timer Init Anim	3
Menu X Start	-800
Menu X End	-450
Speed Anim	400

Simple, right? We already have a new menu:



For now it does the action of prefab, menu copied or none, to change we look for the **Events** in the **Menu.cs** itself, the names of the events already suggestive of the action that it will do.

For example, I will add a message in the console, in the event **Click ()** drag **Menu.cs** (above the inspector) to the click and we go in **No Function> Menu> showMessageInConsole** and we typed some text.



Events

Enter ()

Runtime Only — Menu.menuEnable

menuNew (Men

Exit ()

Runtime Only — Menu.menuDisable

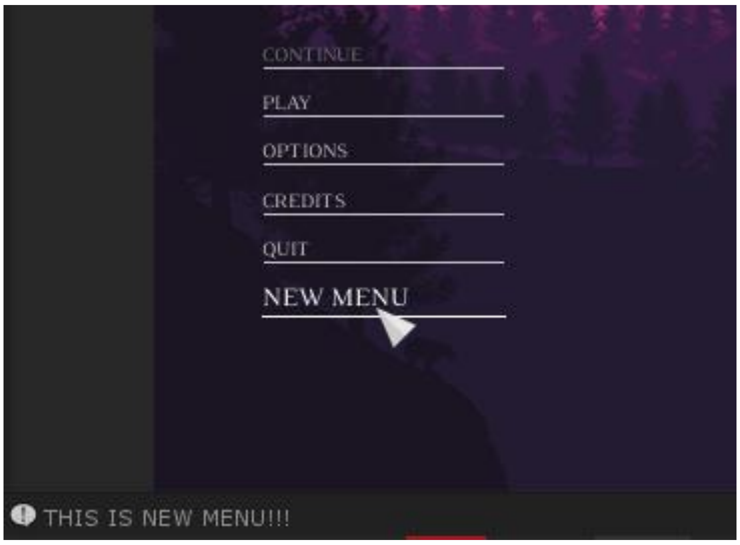
menuNew (Men

Click ()

Runtime Only — Menu.showMessageInConsole

menuNew (Me — THIS IS NEW MENU!!!

Now when we run the project and click on our new menu:

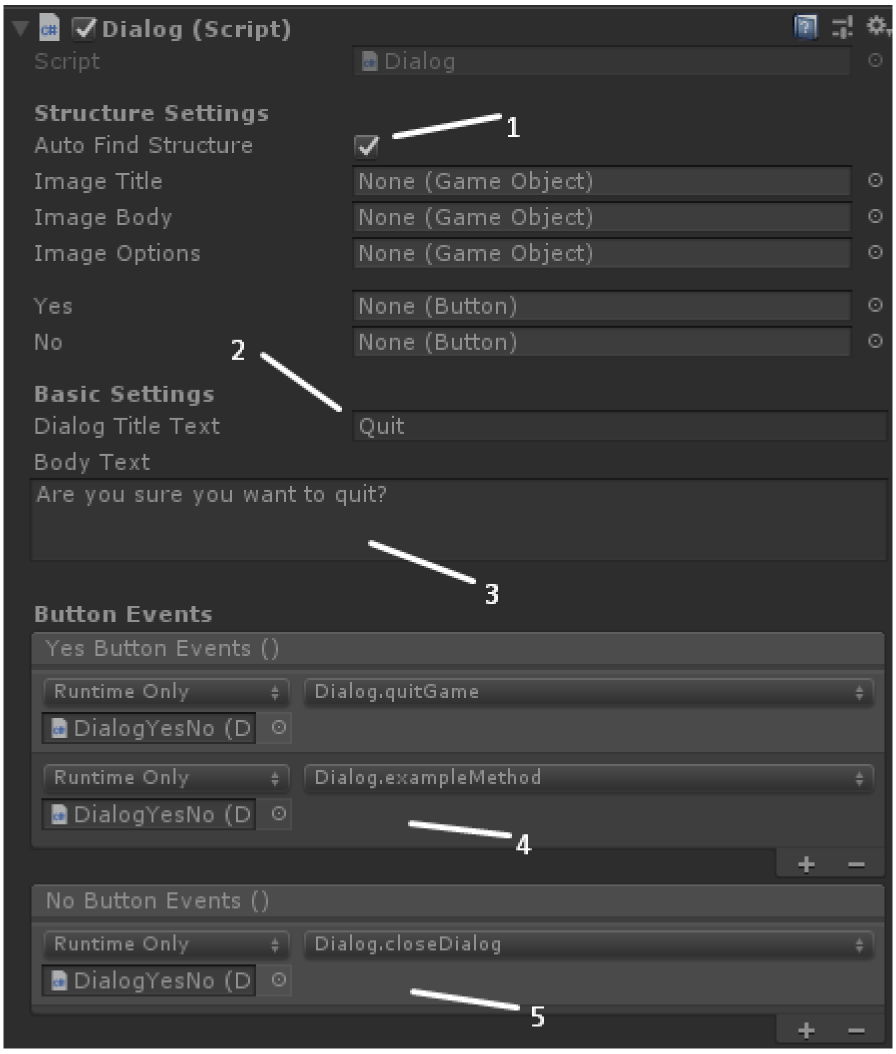


THIS IS NEW MENU!!!

Very simple. Congratulations, you've already created a new menu item.

Creating a Dialog Box

To create a dialog box with yes and no buttons, take the prefab and drag it to the scene or use the custom tab, some important settings:


Dialog (Script)

Script: Dialog

Structure Settings

Auto Find Structure: ✓

Image Title: None (Game Object)

Image Body: None (Game Object)

Image Options: None (Game Object)

Yes: None (Button)

No: None (Button)

Basic Settings

Dialog Title Text: Quit

Body Text: Are you sure you want to quit?

Button Events

Yes Button Events ()

Runtime Only — Dialog.quitGame — DialogYesNo (D

Runtime Only — Dialog.exampleMethod — DialogYesNo (D

No Button Events ()

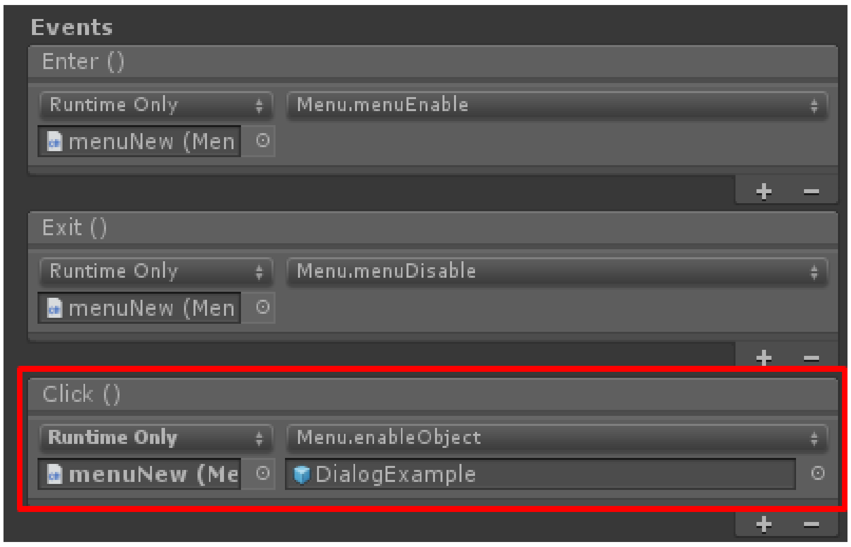
Runtime Only — Dialog.closeDialog — DialogYesNo (D

1. This option automatically finds the entire hierarchy required for the Dialog working. You can turn it off and manually.
2. This is the title of the Dialog Box
3. The body of text / message
4. All events that occur when you click the **yes** button
5. All events that occur when you click the **no** button

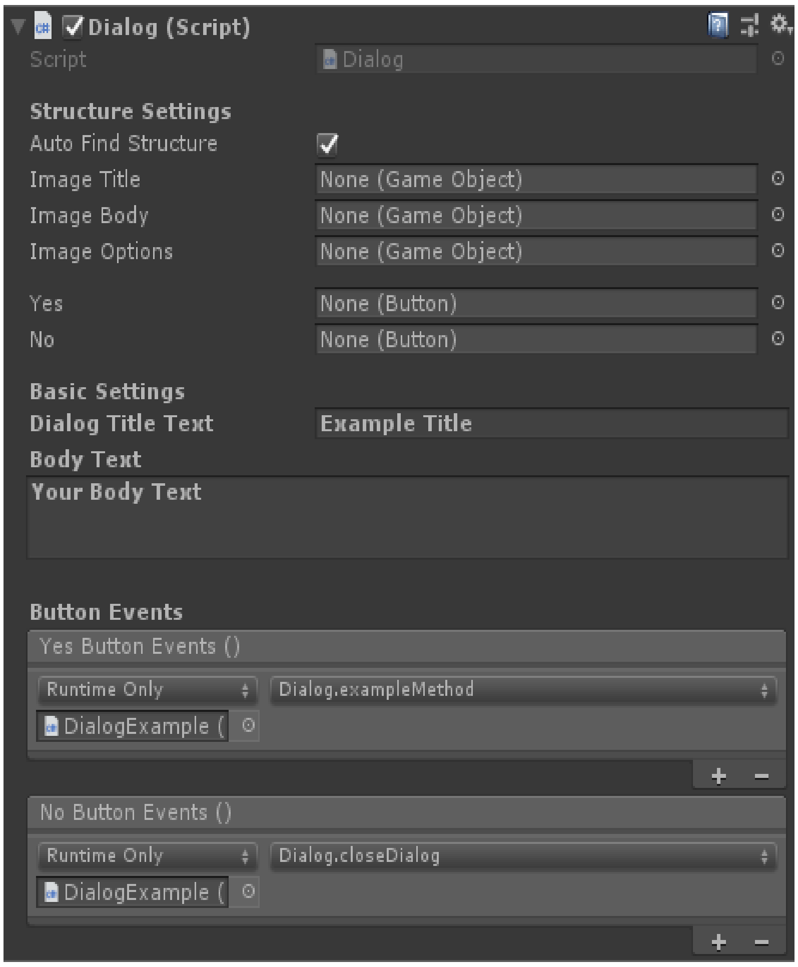
After copying the existing dialog box I will change the title and message and delete the quitGame event.

Remember our menu created let's change the click event, I'll add it myself in the event and select the function

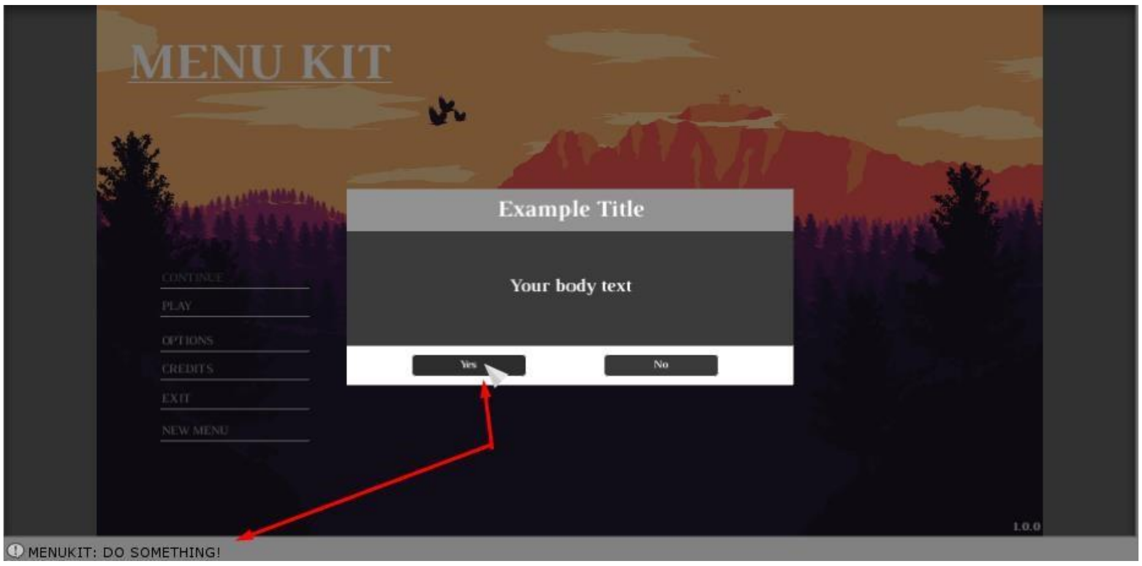
Menu> EnableObject and I will define our new Dialog, thus:


Events
Enter ()
Runtime Only | Menu.menuEnable
menuNew (Men
Exit ()
Runtime Only | Menu.menuDisable
menuNew (Men
Click ()
Runtime Only | Menu.enableObject
menuNew (Me | DialogExample

Our new dialog box looks like this:

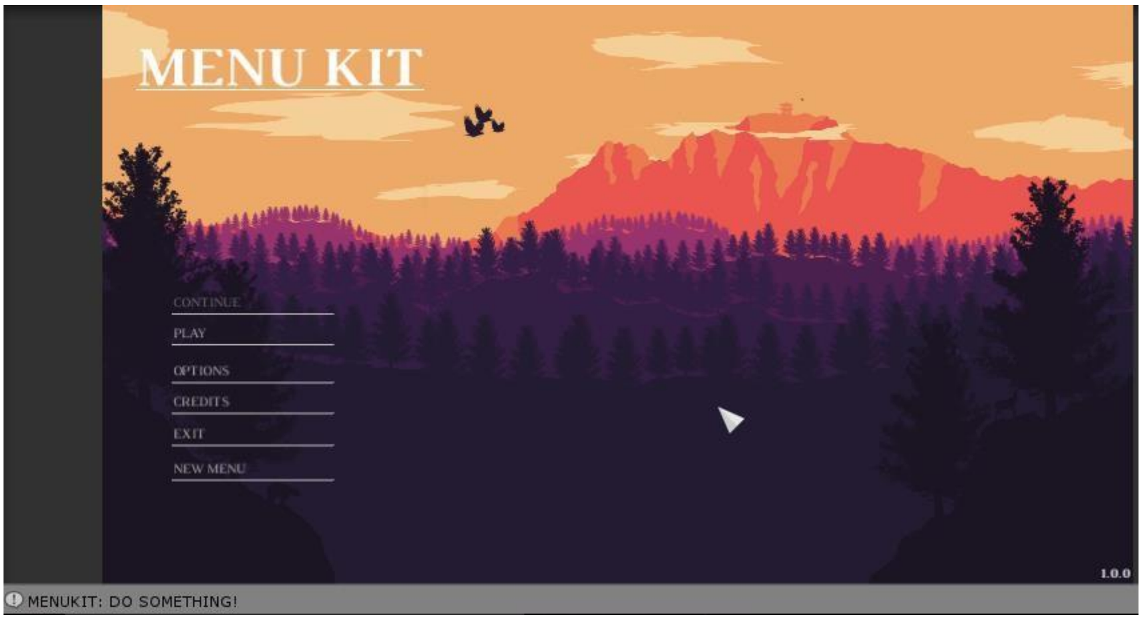

Dialog (Script)
Script: Dialog
Structure Settings
Auto Find Structure: ✓
Image Title: None (Game Object)
Image Body: None (Game Object)
Image Options: None (Game Object)
Yes: None (Button)
No: None (Button)
Basic Settings
Dialog Title Text: Example Title
Body Text
Your Body Text
Button Events
Yes Button Events ()
Runtime Only | Dialog.exampleMethod
DialogExample (
No Button Events ()
Runtime Only | Dialog.closeDialog
DialogExample (

The exampleMethod method shows only one message in the console, after everything configured you can start your project, clicking on the new menu see what happens: Gif (Folders)



MENUKIT: DO SOMETHING!

When clicking on the yes we have a message in the console, clicking on the:



MENUKIT: DO SOMETHING!

The dialog box closes.

Very simple too, with that we finished the tour on the main screen. You can add as many functions as you want in Events.

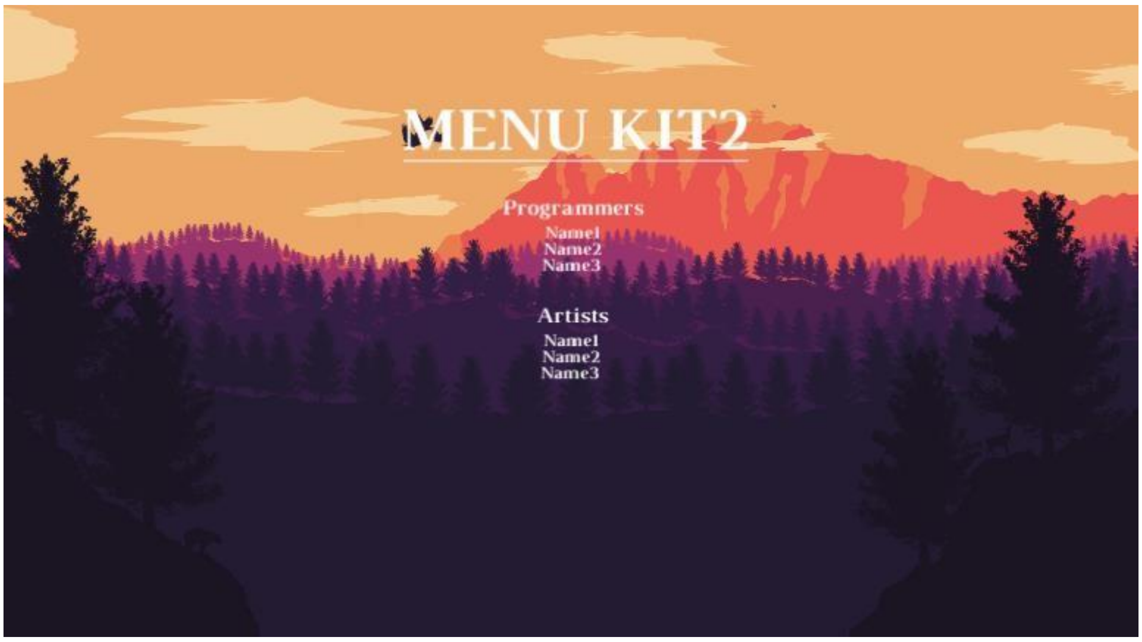
Modifying the Credits Screen

To start with another screen I recommend that you disable the others, in this case I deactivated the MainScreen (which was active) in the inspector and it becomes cleaner / easier / more practical to work on, come on.

A CreditsScreen is found in **MenuKit> Canvas> MenuKit_Canvas> CreditsScreen**, to change the title let's object title_c and change the TEXT, I put it as "MENU KIT2".

In the hierarchy we have Subtitle_structure1 and 2, are two examples very used in the credits, feel free to change, in my case I will use them as an example and I will disable the object wilgner_credits.

I'll make a small change in his animation, by default will come the direction animation, I'll position the new CreditsScreen in the center:

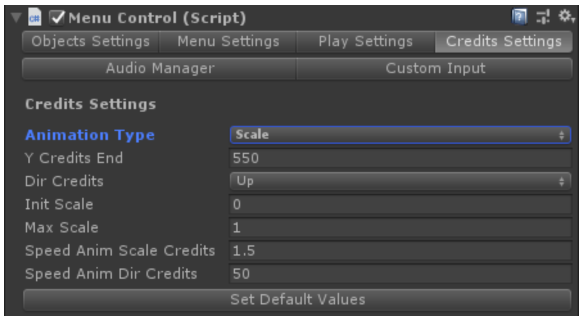


That way, and now we go to MenuControl, in

Objects Settings I'll put my new

CreditsScreen, I will disable the Anim Direction Credits and activate Anim Credits Scale by doing this:

You can set the Initial Scale, Maximum Scale and the speed of the scale to do the scale animation (Already set).


Menu Control (Script)

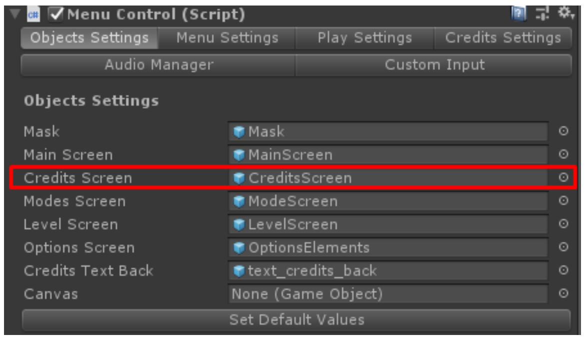
Objects Settings | Menu Settings | Play Settings | Credits Settings

Audio Manager | Custom Input

Credits Settings

Animation Type	Scale
Y Credits End	550
Dir Credits	Up
Init Scale	0
Max Scale	1
Speed Anim Scale Credits	1.5
Speed Anim Dir Credits	50

Set Default Values


Menu Control (Script)

Objects Settings | Menu Settings | Play Settings | Credits Settings

Audio Manager | Custom Input

Objects Settings

Mask	Mask
Main Screen	MainScreen
Credits Screen	CreditsScreen
Modes Screen	ModeScreen
Level Screen	LevelScreen
Options Screen	OptionsElements
Credits Text Back	text_credits_back
Canvas	None (Game Object)

Set Default Values

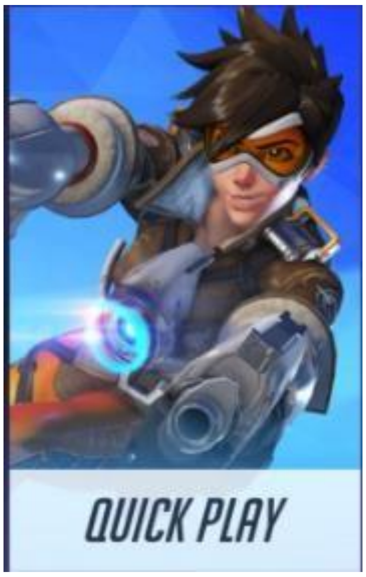
You can start the project or disable the CreditsScreen in the inspector (all screens are disabled at startup, minus the MainScreen).

Simple.

Modifying the ModeScreen

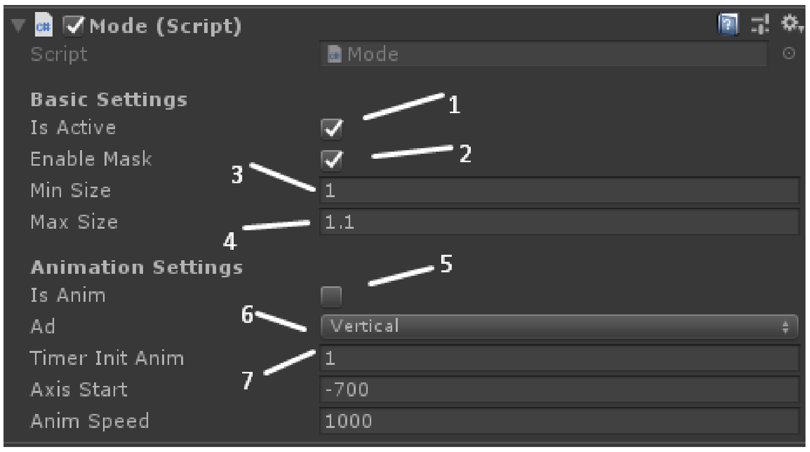
Again, it deactivates the other screens and only activates the ModeScreen to work better without the others getting in the way. **ModeScreen** is in **MenuKit > Canvas > MeniKit_Canvas > ModeScreen**

Again we have the BasicSettings, you can change the title and version. I will copy the mode3 (name mode4) and position as I want, I will not change any information in it, we go to Modes, in background_mode we can change the background image of our background, I will leave default, a good example is the game Overwatch, we can see:



To change the title is simple, we go in the title_mode_text and change the TEXT component in the inspector the same thing in the description_text, the maskDisable object is if the mode is unavailable the maskDisable is activated and leaves the mode dark.

Be sure to check that all masks are in the correct mode position they belong to.



Mode (Script)

Script	Mode
Basic Settings	
Is Active	✓
Enable Mask	✓
Min Size	1
Max Size	1.1
Animation Settings	
Is Anim	
Ad	Vertical
Timer Init Anim	1
Axis Start	-700
Anim Speed	1000

1. Active case means the mode is active and can be interactive.
2. Activate the mask in case it is disabled there will be a darkening in the mode.
3. Standard Scale
4. Scale when the mouse is on top
5. Initial animation
6. Direction of the animation
7. Time to start the animation (I'm using 0.5 difference of each mode)

I will change the animation from mode4 to 1.5. In the EventTrigger in Pointer Click I will change the playmode3 to playmode4.

I go into mode2 and change the isActive to false.

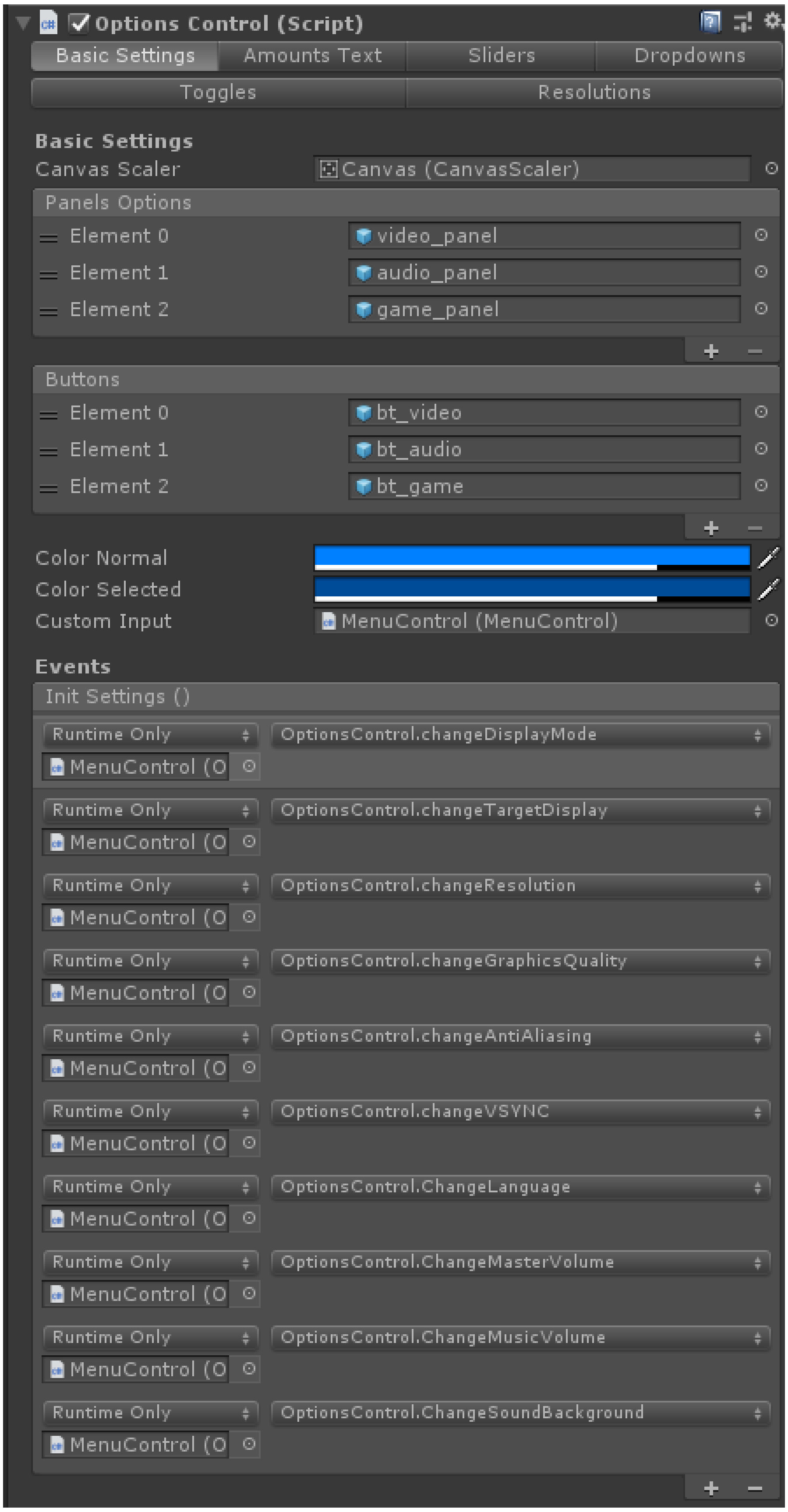
Let's configure the MenuControl:

If you have another screen of modes and you are not changing the one of the prefab do not forget to define the new one in Objects Settings as we did with Credits Screen.

In Play Settings disables No Modes.

Okay, test the project and see what happens:

Options Screen



Options Control (Script)

Basic Settings | Amounts Text | Sliders | Dropdowns
Toggles | Resolutions

Basic Settings

Canvas Scaler	Canvas (CanvasScaler)

Panels Options

Element 0	video_panel
Element 1	audio_panel
Element 2	game_panel

Buttons

Element 0	bt_video
Element 1	bt_audio
Element 2	bt_game

Color Normal	🟦
Color Selected	🟦
Custom Input	MenuControl (MenuControl)

Events

Init Settings ()

Runtime Only	MenuControl (O	OptionsControl.changeDisplayMode
Runtime Only	MenuControl (O	OptionsControl.changeTargetDisplay
Runtime Only	MenuControl (O	OptionsControl.changeResolution
Runtime Only	MenuControl (O	OptionsControl.changeGraphicsQuality
Runtime Only	MenuControl (O	OptionsControl.changeAntiAliasing
Runtime Only	MenuControl (O	OptionsControl.changeVSYNC
Runtime Only	MenuControl (O	OptionsControl.ChangeLanguage
Runtime Only	MenuControl (O	OptionsControl.ChangeMasterVolume
Runtime Only	MenuControl (O	OptionsControl.ChangeMusicVolume
Runtime Only	MenuControl (O	OptionsControl.ChangeSoundBackground

Canvas Scaler: Canvas scaler set to canvas object

Panels Options: They are all options menu panels (video, audio and game) and Buttons are the buttons of these respective panels.

Color nomal: This is the default color (not selected) of the button

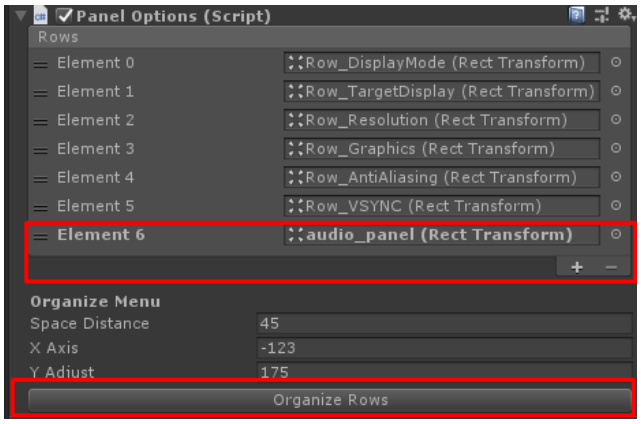
Color Selected: Is the color of when the respective button panel is selected.

Events: All events started along with the menu, for example changeResolution, this event changes the menu resolution according to json when starting the menu. You can add or remove actions to run at the beginning.
(Suggestion sent in my email)

Now let's create a sample toggles (checkbox) and learn how to save in JSON.

You can copy a Toggle ready or create from the tab.

You can organize in Panel by adding a new element and clicking on "Organize Rows", this applies to all panels



Panel Options (Script)

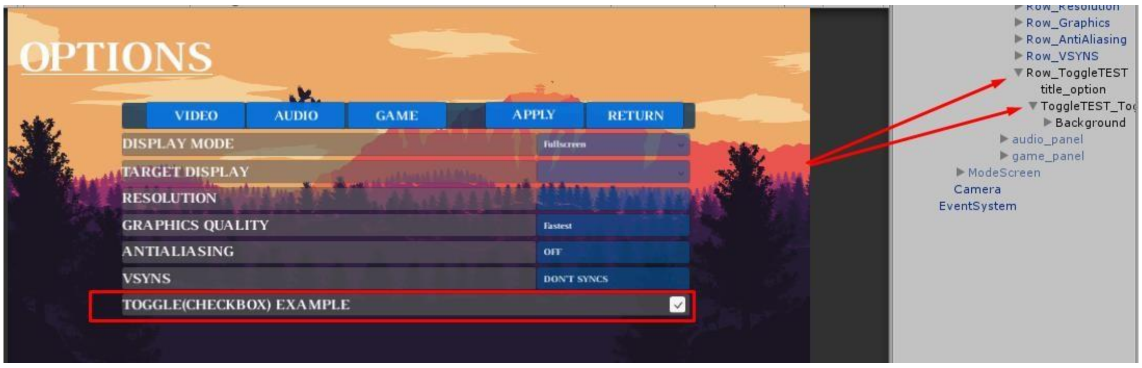
Rows

Element 0	Row_DisplayMode (Rect Transform)
Element 1	Row_TargetDisplay (Rect Transform)
Element 2	Row_Resolution (Rect Transform)
Element 3	Row_Graphics (Rect Transform)
Element 4	Row_AntiAliasing (Rect Transform)
Element 5	Row_VSYNC (Rect Transform)
Element 6	**audio_panel (Rect Transform)**

Organize Menu

Space Distance	45
X Axis	-123
Y Adjust	175

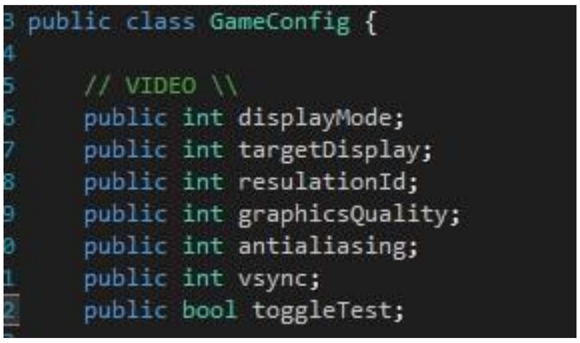
Organize Rows



Remember to change the names to stay organized, let's open the **Script > GameConfig**, this is the class where json is created/saved/stored, in //VIDEO below the VSYNC we will create a new variable:

public bool toggleTest;

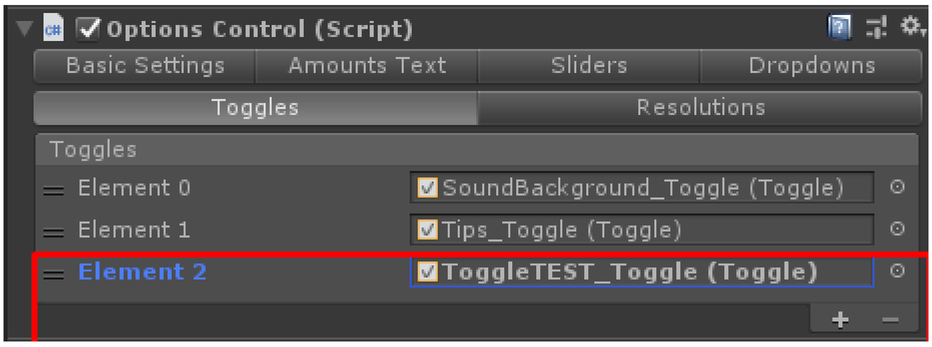
Type bool because that is the value that the toggle stores.



```
public class GameConfig {

    // VIDEO \\
    public int displayMode;
    public int targetDisplay;
    public int resulationId;
    public int graphicsQuality;
    public int antialiasing;
    public int vsync;
    public bool toggleTest;
```

Now let's go to the MenuControl object (OptionsControl.cs) in toggles let's add the toggle we created:



Options Control (Script)

Toggles	
Element 0	SoundBackground_Toggle (Toggle)
Element 1	Tips_Toggle (Toggle)
Element 2	ToggleTEST_Toggle (Toggle)

Now let's go to the Script > OptionsConfig and go to the saveGameConfig method. In //Graphics let's type:gameConfig.toggleTest = toggles [2].isOn;

The 2 is the position of it in the vector, in case 2 (Element 2), now it is already saving the value in the JSON file, now let's do it when starting the game again loading the selected option. Let's in the setValues method in // Graphics let's type:

```
toggles [2].isOn = _gameConfig.toggleTest;
```

With this, the value of it is defined according to the value in our JSON.

To change something in the game when you click Apply you must create a change method, work the value of the toggle (example) inside and call the method in changeSettingsGame ().

With just that you already have a new option on the OptionsScreen running and saving, loading from a JSON file.

You can retrieve these values in other scenes by reading the json file (example in the other scene, YOURGAME).

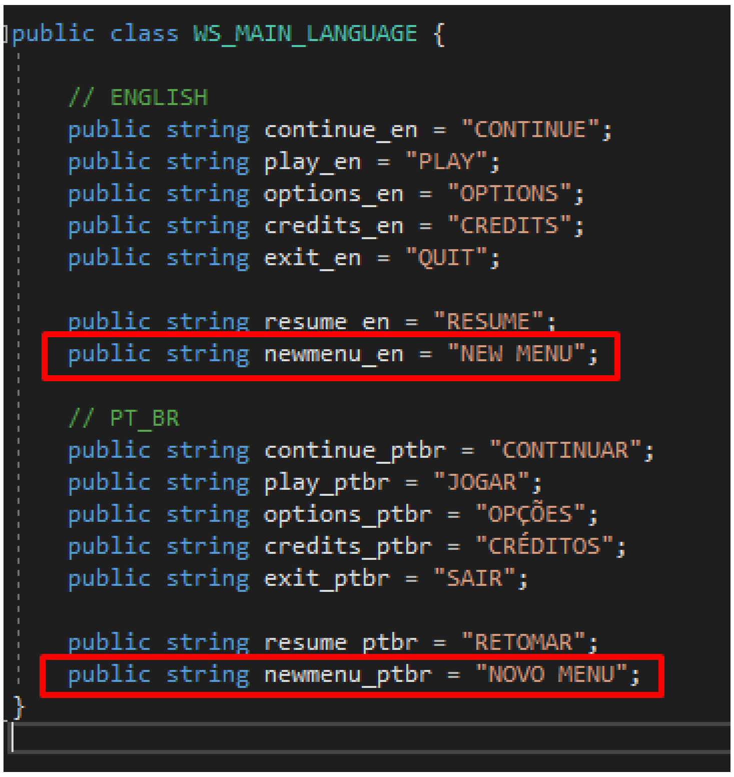
Language Control

The new version of MMK allows multiple languages.
Let's start. Previously I taught how to create a new menu item, now let's translate.
First the menu items are referenced in Menu Settings (MenuControl.cs) and not in LanguageControl.cs.
We already referenced it in creation.

Let's go in the folder **Scripts** > **Json Class** > **Languages** > **WS_MENUKIT**
WS_MAIN_LANGUAGE.cs > This file contains the translations of the menu item (CONTINUE, PLAY, OPTIONS, etc).
WS_OPTIONS_LANGUAGE.cs > This file contains the translations for all elements present in the options screen

I am still developing, but the fundamental is already ready, for example missing translations within the dropdowns.

As we create a menu item we will use the **WS_MAIN_LANGUAGE.cs** as explained above.
After opening it we will insert two variables to represent the text in English and Portuguese (example, you can add another language), it will look like this:



```
public class WS_MAIN_LANGUAGE {

    // ENGLISH
    public string continue_en = "CONTINUE";
    public string play_en = "PLAY";
    public string options_en = "OPTIONS";
    public string credits_en = "CREDITS";
    public string exit_en = "QUIT";

    public string resume_en = "RESUME";
    public string newmenu_en = "NEW MENU";

    // PT_BR
    public string continue_ptbr = "CONTINUAR";
    public string play_ptbr = "JOGAR";
    public string options_ptbr = "OPÇÕES";
    public string credits_ptbr = "CRÉDITOS";
    public string exit_ptbr = "SAIR";

    public string resume_ptbr = "RETOMAR";
    public string newmenu_ptbr = "NOVO MENU";
}
```

After that save and we will open the LanguageControl (scripts folder or can open from the MenuControl object with right click and edit).
This script is used to check which language is selected and from there it updates the text according to our languages bank (The two scripts containing the languages).

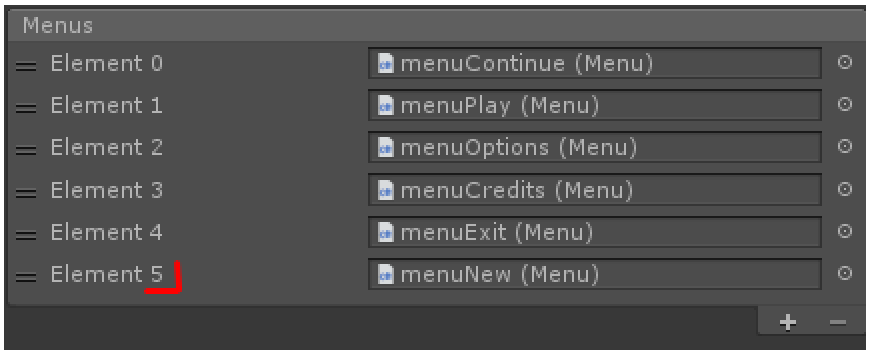
Let's find two functions:

SetMainLanguage: This function is used to update texts on the main menu.

SetOptionsLanguage: Updates the text of the options.

As this new menu item is in the main menu and not the pause menu we will put inside the IF inGame (or isPause).

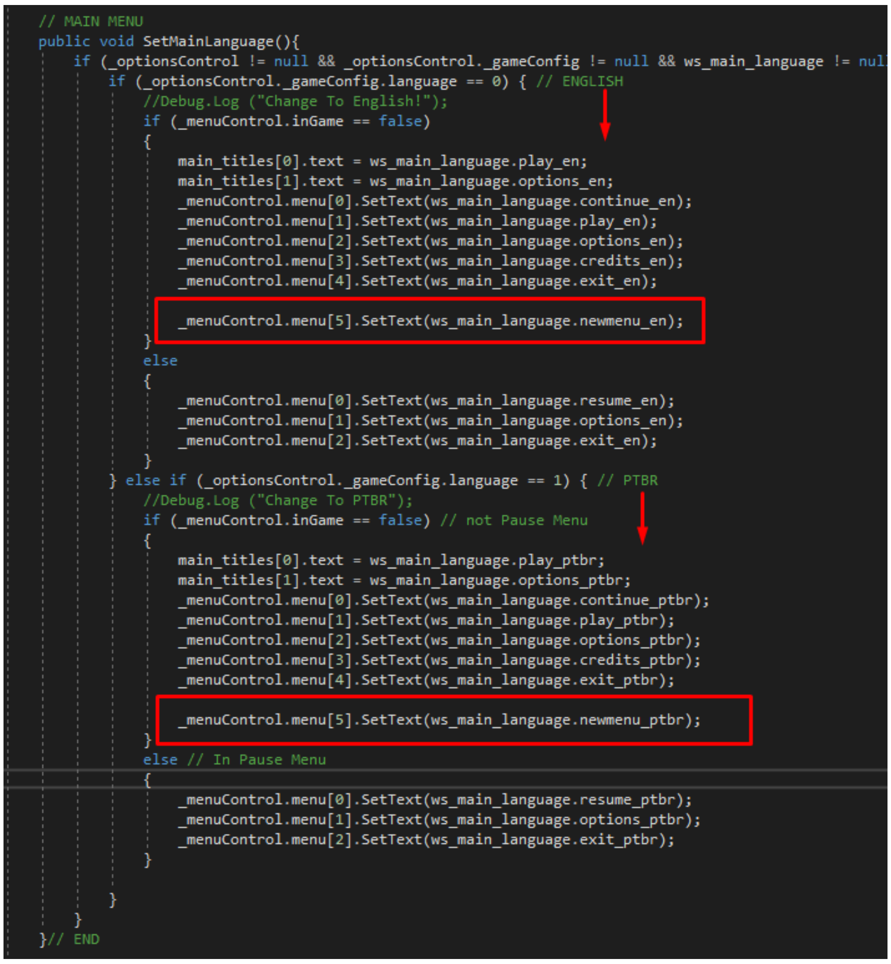
Let's reference the new menu index we created (in my case it's 5)



Menus

Element 0	menuContinue (Menu)
Element 1	menuPlay (Menu)
Element 2	menuOptions (Menu)
Element 3	menuCredits (Menu)
Element 4	menuExit (Menu)
Element 5	menuNew (Menu)

Referencing the script in which the translation of this menu and the variable that we created above to reference it, like this:



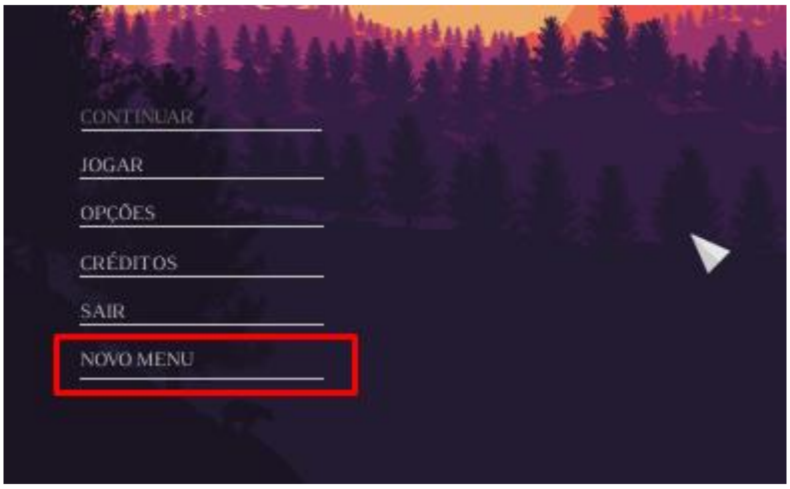
```
// MAIN MENU
public void SetMainLanguage(){
    if (_optionsControl != null && _optionsControl._gameConfig != null && ws_main_language != nul
        if (_optionsControl._gameConfig.language == 0) { // ENGLISH
            //Debug.Log ("Change To English!");
            if (_menuControl.inGame == false)
            {
                main_titles[0].text = ws_main_language.play_en;
                main_titles[1].text = ws_main_language.options_en;
                _menuControl.menu[0].SetText(ws_main_language.continue_en);
                _menuControl.menu[1].SetText(ws_main_language.play_en);
                _menuControl.menu[2].SetText(ws_main_language.options_en);
                _menuControl.menu[3].SetText(ws_main_language.credits_en);
                _menuControl.menu[4].SetText(ws_main_language.exit_en);

                _menuControl.menu[5].SetText(ws_main_language.newmenu_en);
            }
            else
            {
                _menuControl.menu[0].SetText(ws_main_language.resume_en);
                _menuControl.menu[1].SetText(ws_main_language.options_en);
                _menuControl.menu[2].SetText(ws_main_language.exit_en);
            }
        } else if (_optionsControl._gameConfig.language == 1) { // PTBR
            //Debug.Log ("Change To PTBR");
            if (_menuControl.inGame == false) // not Pause Menu
            {
                main_titles[0].text = ws_main_language.play_ptbr;
                main_titles[1].text = ws_main_language.options_ptbr;
                _menuControl.menu[0].SetText(ws_main_language.continue_ptbr);
                _menuControl.menu[1].SetText(ws_main_language.play_ptbr);
                _menuControl.menu[2].SetText(ws_main_language.options_ptbr);
                _menuControl.menu[3].SetText(ws_main_language.credits_ptbr);
                _menuControl.menu[4].SetText(ws_main_language.exit_ptbr);

                _menuControl.menu[5].SetText(ws_main_language.newmenu_ptbr);
            }
            else // In Pause Menu
            {
                _menuControl.menu[0].SetText(ws_main_language.resume_ptbr);
                _menuControl.menu[1].SetText(ws_main_language.options_ptbr);
                _menuControl.menu[2].SetText(ws_main_language.exit_ptbr);
            }

        }
    }
}// END
```

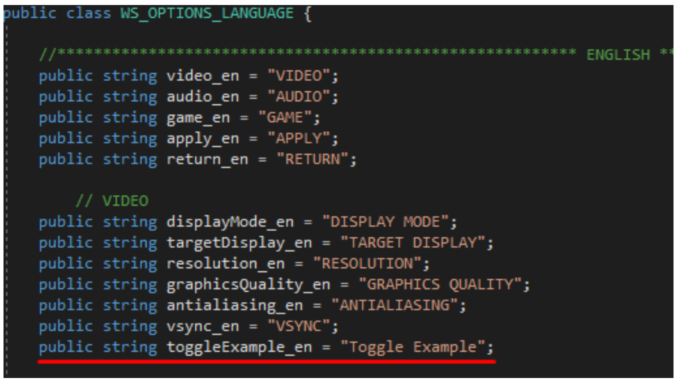
Now, just save and you're done. (teste).



Let's now translate the toggle we created. We follow almost our own steps.

Now let's use the: **WS_OPTIONS_LANGUAGE.cs**

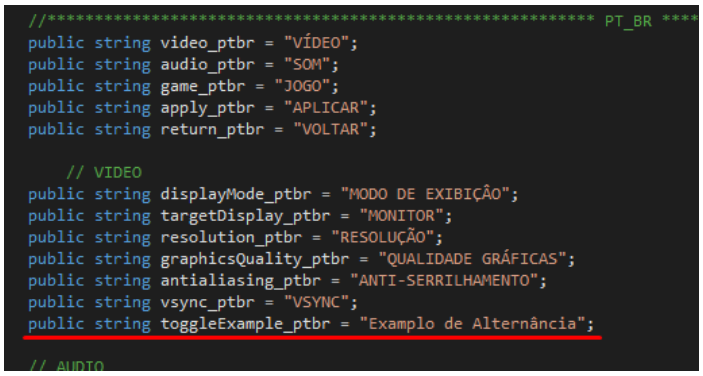
The toggle is in the video panel, then:



```
public class WS_OPTIONS_LANGUAGE {

    //********************************************************* ENGLISH **
    public string video_en = "VIDEO";
    public string audio_en = "AUDIO";
    public string game_en = "GAME";
    public string apply_en = "APPLY";
    public string return_en = "RETURN";

        // VIDEO
    public string displayMode_en = "DISPLAY MODE";
    public string targetDisplay_en = "TARGET DISPLAY";
    public string resolution_en = "RESOLUTION";
    public string graphicsQuality_en = "GRAPHICS QUALITY";
    public string antialiasing_en = "ANTIALIASING";
    public string vsync_en = "VSYNC";
    public string toggleExample_en = "Toggle Example";
```

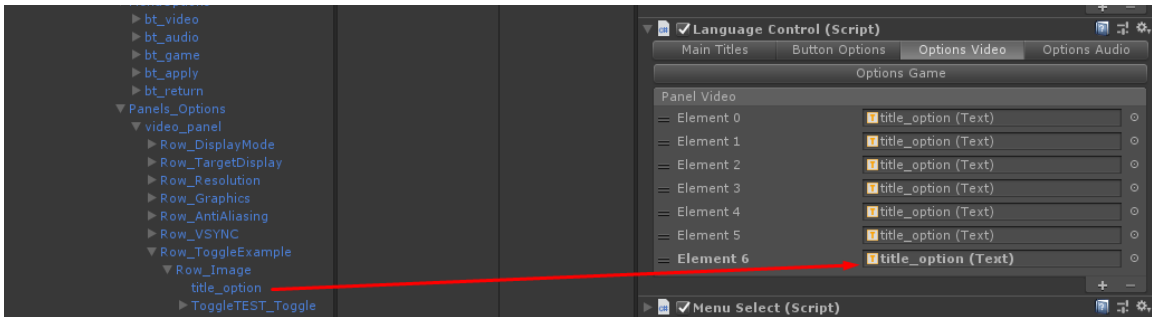


```
    //********************************************************* PT_BR ****
    public string video_ptbr = "VÍDEO";
    public string audio_ptbr = "SOM";
    public string game_ptbr = "JOGO";
    public string apply_ptbr = "APLICAR";
    public string return_ptbr = "VOLTAR";

        // VIDEO
    public string displayMode_ptbr = "MODO DE EXIBIÇÂO";
    public string targetDisplay_ptbr = "MONITOR";
    public string resolution_ptbr = "RESOLUÇÃO";
    public string graphicsQuality_ptbr = "QUALIDADE GRÁFICAS";
    public string antialiasing_ptbr = "ANTI-SERRILHAMENTO";
    public string vsync_ptbr = "VSYNC";
    public string toggleExample_ptbr = "Examplo de Alternância";

    // AUDIO
```

We already have the language bank ready, now we need to reference the text of the toggle for the MMK to know that it exists and for us to use in the codes.



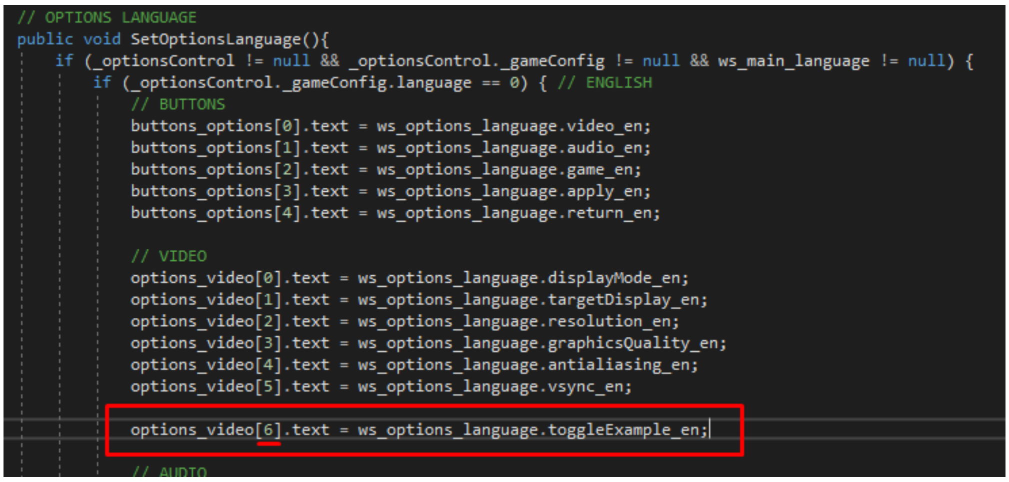
```
▶ bt_video
▶ bt_audio
▶ bt_game
▶ bt_apply
▶ bt_return
▼ Panels_Options
  ▼ video_panel
    ▶ Row_DisplayMode
    ▶ Row_TargetDisplay
    ▶ Row_Resolution
    ▶ Row_Graphics
    ▶ Row_AntiAliasing
    ▶ Row_VSYNC
    ▼ Row_ToggleExample
      ▼ Row_Image
          title_option
        ▶ ToggleTEST_Toggle
```

```
▼ ☑ Language Control (Script)
Main Titles | Button Options | Options Video | Options Audio
Options Game
Panel Video
  Element 0    title_option (Text)
  Element 1    title_option (Text)
  Element 2    title_option (Text)
  Element 3    title_option (Text)
  Element 4    title_option (Text)
  Element 5    title_option (Text)
  Element 6    title_option (Text)
▶ ☑ Menu Select (Script)
```

Let's reference our new variable (languages) to update the text correctly:

LanguageControl.cs and the method **SetOptionsLanguage()** this time.

Remembering that his index is 6 as the image above, then:



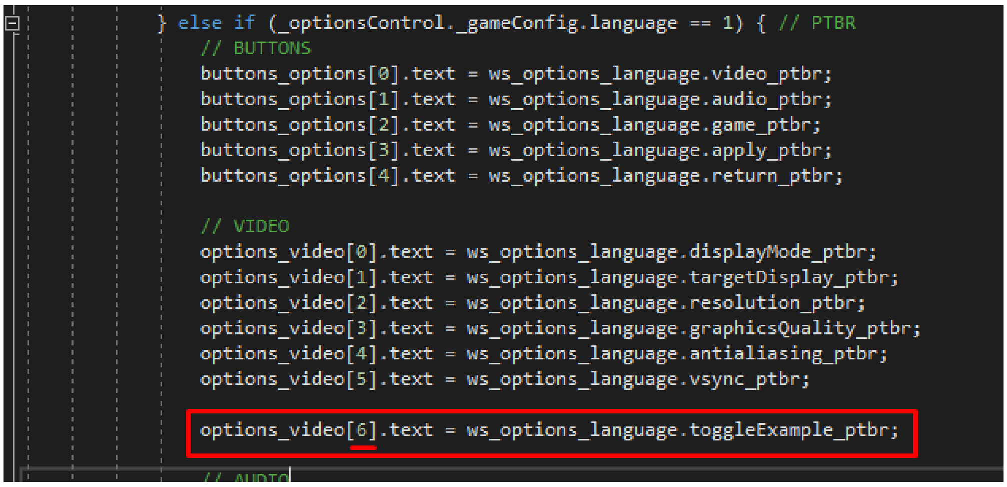
```
// OPTIONS LANGUAGE
public void SetOptionsLanguage(){
    if (_optionsControl != null && _optionsControl._gameConfig != null && ws_main_language != null) {
        if (_optionsControl._gameConfig.language == 0) { // ENGLISH
            // BUTTONS
            buttons_options[0].text = ws_options_language.video_en;
            buttons_options[1].text = ws_options_language.audio_en;
            buttons_options[2].text = ws_options_language.game_en;
            buttons_options[3].text = ws_options_language.apply_en;
            buttons_options[4].text = ws_options_language.return_en;

            // VIDEO
            options_video[0].text = ws_options_language.displayMode_en;
            options_video[1].text = ws_options_language.targetDisplay_en;
            options_video[2].text = ws_options_language.resolution_en;
            options_video[3].text = ws_options_language.graphicsQuality_en;
            options_video[4].text = ws_options_language.antialiasing_en;
            options_video[5].text = ws_options_language.vsync_en;

            options_video[6].text = ws_options_language.toggleExample_en;

            // AUDIO
```

And now the other language (in case pt_br):



```
        } else if (_optionsControl._gameConfig.language == 1) { // PTBR
            // BUTTONS
            buttons_options[0].text = ws_options_language.video_ptbr;
            buttons_options[1].text = ws_options_language.audio_ptbr;
            buttons_options[2].text = ws_options_language.game_ptbr;
            buttons_options[3].text = ws_options_language.apply_ptbr;
            buttons_options[4].text = ws_options_language.return_ptbr;

            // VIDEO
            options_video[0].text = ws_options_language.displayMode_ptbr;
            options_video[1].text = ws_options_language.targetDisplay_ptbr;
            options_video[2].text = ws_options_language.resolution_ptbr;
            options_video[3].text = ws_options_language.graphicsQuality_ptbr;
            options_video[4].text = ws_options_language.antialiasing_ptbr;
            options_video[5].text = ws_options_language.vsync_ptbr;

            options_video[6].text = ws_options_language.toggleExample_ptbr;

            // AUDIO
```

Done.

You can reference the **GameConfig.language** in the script **OptionsControl** to be able to translate other parts of your game and into new languages.

Please leave a review on the Assets Store
This gives feedback to me on what to add / improve and others.
Thank you.

F.A.Q

Frequently asked questions on youtube, forum, email or the Unity Assets Store will appear here.

Credits & Contact

This package was developed by Wilgner Fabio
and is just a package of something bigger.
Special thanks to Paulo Camacan for offering
and doing the UITransition script
and to you for buying and supporting me <3.
Luthier Reg by Source.

wilgner.fabio@gmail.com
or
wilgner.fabio2@gmail.com
https://forum.unity.com/threads/released-main-menu-kit-simple-flat-and-complete.518549/