Memory Game

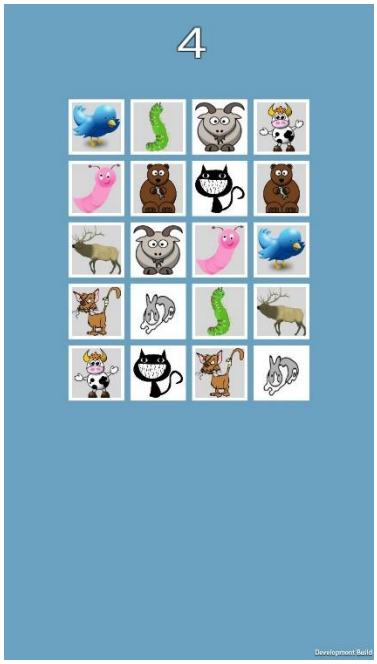
By WESoft Soluções

Version: 2.0

Last Updated: 11 April 2017

Website: http://www.wesoft.com.br

Contact: assets@wesoft.com.br



Overview

First, thank you for purchase the Memory Game project.

Here you get a complete game project that can be easily published into the principal game marketplaces.

Features:

- Tile animations (rotation to show / hide the “card” content)
- Portrait / Landscape modes
- 6 themes (and you can add your own themes)
- 5 dificulty levels (customizable)
- Multilanguage, using XML (English US and Brazilian Portuguese. And you can add others)

- Unity Ads (Android & iOS)
- Vungle Ads (UWP, but you can use for Android / iOS too). See the Official Documentation: https://support.vungle.com/hc/en-us/articles/204311244

Assets

Scenes (folder: /Scenes):

- Menu – the initial game screen. Here the user can easily access some game information, change the game language and select to play the game.
- Game – This is the scene where the "action" occurs.

Classes (folder: /Scripts):

- GameManager.cs – this class is responsible for the game initialization and some actions used by the menu UI buttons, like select the game theme, change language, open the developer website and so on. The class is instancied in the Menu scene. It's attached to GameManager component.
- Settings.cs – controls the some runtime game settings / user preferences, like the developer's info, select language, current theme…
- LanguageManager.cs – this is the base class for controlling the game language. It loads the selected language using the corresponding XML file (see the /Resources/Languages folder).
- GameSceneManager.cs – controls the transition between scenes and the ads presentation.
- ScreenManager.cs – controls the animation between the game menus. See https://docs.unity3d.com/Manual/HOWTO-UIScreenTransition.html
- GameGrid.cs – this is basically, the game core. Initialize the game according to the selected theme and level, instantiate copies of the Tile prefab and controls all the game actions. Is instancied in the Game Scene, attached to the Grid component.
- Util.cs – used to provide some helper functions. For now, only only the suffle code is used here.
- Tile.cs – conrols the Tile prefab (status, rotate).

Prefabs (folder: /Prefabs)

- Tile.prefab – this

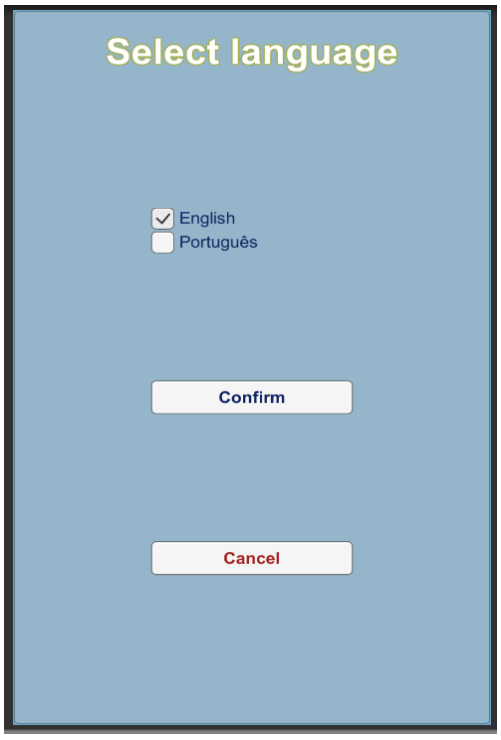
Resources (folder: /Resources)

- Languages – here you can put your own language files. Use the existing xml as example.
- Themes – each theme has your own folder. You can add more themes, but need to make some code / menu changes.

Other folders:

- Animations – used in the UI
- Audio – the sound effects
- Textures – used in the Tile prefab
- Textures/Backgrounds – used in the Tile prefab
- Textures/UI – used in the UI (images for buttons)

How the language selection works and how to add your own language



To add more languages, first, create a xml file in the /Resources/Languages/ folder. Name it as you want, but we recommend to use names in the same format as the existent files (language-variation.xml).

In the **Menu** scene, add a new Toggle (see the **languageSelect/PanelLanguage/Panel**) in the scene. Don't forget to use the same ToggleGroup in the Toggle. Name the Toggle. Also, set the toggle's title. We prefer to use the language name in your native translation (like "Português" for the Brazilian Portuguese Toggle).

Now, open the GameManager class and reference the new Toggle(s). See the lines 51-52 as reference:

```
  Toggle englishToggle;
  Toggle brazilianToggle;
```

Now, go to the **ConfirmLanguage** method (line 134) and change it to check the select Toggle (remember to pass the xml name - without the extension – in the SetLanguage method):

```
public void ConfirmLanguage()
{
	if (englishToggle.isOn)
		SetLanguage("en-us");
	else
		SetLanguage("pt-br");
}
```

***note: may be better if you change the if / else code to a switch statement, as the number of languages can be increased.**

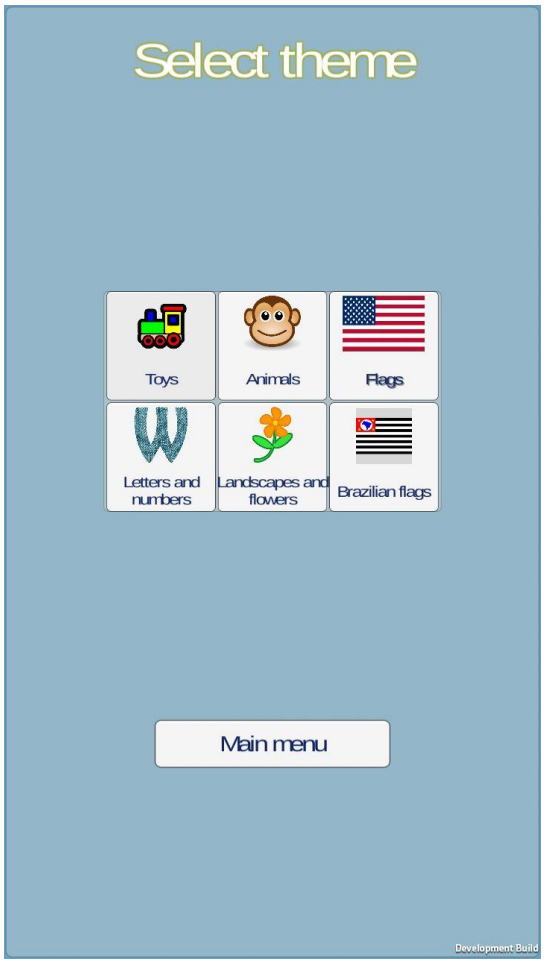
Also, go to the **SetLanguage** method and add your language to the If statement (or change to a switch statement) in the line 153:

```
if (newLanguage == "en-us")
	englishToggle.isOn = true;
else
	brazilianToggle.isOn = true;
```

To change the default language from English to some other, go to the line 123 and change it to call the language of your choice:

```
SetLanguage(PlayerPrefs.GetString("language", "en-us"));
```

How the theme selection works and how to add your own theme



The theme selection works a bit different from the language select. First, see the **ThemeSelect/ PanelThemeSelect/Panel** component in the **Menu** scene. Here you can see six buttons, that, on clicked, calls the **SetTheme** method in the **GameManager** class. This method requires a int, which is the theme index defined in the string array “themes”, defined in the **GameGrid** class. See the line 51:

```
string[] themes = {
                            "Toys",
                            "Animals",
                            "Flags",
                            "Letters",
                            "Landscapes",
                            "Brazilian_Flags",
                        };
```

To add a new theme, simply create a new subfolder in the **/Resources/Themes/** folder and put your theme images into this subfolder. Then, go to the code above (**GameGrid class**) and add the folder name to the array. Go to the Menu scene and **add a new button** as a child of the **ThemeSelect/ PanelThemeSelect/Panel** component. Set the image and the text for the button, and define the **On Click** event like the other buttons, calling the **SetTheme** method (passing the

index for this theme defined in the array above). **Don't forget to** call the ScreenManager.OpenPanel (PanelDificulty) too.

How the difficulty selection works

The difficulty select uses Toggles and a Toggle Group. See the **DificultySelect/ PanelDificulty/ Panel** gameobject. The difficulties are defined in the code (see below). Each difficulty toggle's names ends with "_index" (_0, _1, _2 and so on).

How it works:

When the user clicks the **Start game** button, the **StartGame** method (from the **GameManager** class) is called (line 202):

```
public void StartGame()
{

      Settings.Dificulty =
int.Parse(dificultyToggleGroup.ActiveToggles().FirstOrDefault().name.Split('_')[1
]);
```

```
        GameSceneManager.StartGame();
}
```

First, we get the selected dificulty index and pass it to the **Settings** class. Then, we call the **StartGame** method from the **GameSceneManager** class. This method simply call the **LoadScene**, loading (wow) the **Game** scene.

The **Game** scene, uses the **GameGrid** class. Here the difficulties are defined. First, whe have the DifficultyStruct struct (line 68) and the whichDifficulty variable:

```
public struct DifficultyStruct
{
       public string Name; //to help identify only
       public int NumberOfTiles;
       public int TilesPerRow;
       public float TimeToMemorize;
}

DifficultyStruct whichDifficulty;
```

Then, in the awake method (line 118), we define all the difficulties, using a **List**:

```
difficulties = new List<DifficultyStruct>();

DifficultyStruct diff = new DifficultyStruct();
diff.Name = "Very easy";
diff.NumberOfTiles = 4;
diff.TilesPerRow = 2;
diff.TimeToMemorize = 10;

difficulties.Add(diff);


diff = new DifficultyStruct();
diff.Name = "Easy";
diff.NumberOfTiles = 6;
diff.TilesPerRow = 2;
diff.TimeToMemorize = 8;

difficulties.Add(diff);
```

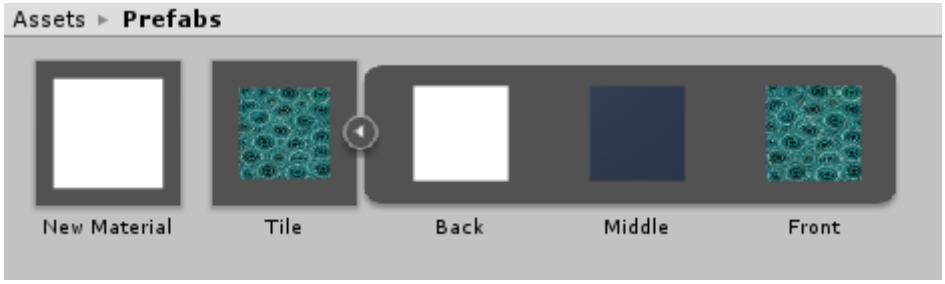
And so on.

In the line 169 we get the selected difficulty from the **Settings** class:

```
whichDifficulty = difficulties[Settings.Dificulty];
```

Then, in the **Start** method the code calls the **newGame** method, that creates the tiles and initialize the game.

Changing details of the Tile prefab



The Tile prefab is composed by 3 Quads: **Back**, **Middle** and **Front**.

The **Back** quad is where the "face" that displays the theme image for that Tile. The **Middle** quad is the tile border. And the **Front** quad is the image displayed when the card/title is hidden. You can change the Tile border color by changing the **Tint** of the **Middle** quad "New Material" component. And you can change the **Front** image selecting another **Texture** in their "FrontPiece" component.

About the sounds and graphic assets used in the game

All of the assets are public domain. Those can be found in the https://openclipart.org and http://freesound.org websites.

Other questions

If you have other questions, please contact-us: assets@wesoft.com.br