2D Complete Game

ToPlay Studio

2D Complete Game's Documentation

Thank you for purchasing 2D Complete Game!
If you wish give us any feedback don't hesitate, we will appreciate it so much. In case you have any questions let us know too, we will answer so soon as possible.
Contact: contato@toplaystudio.com.br

Contents

1- Change Log

Version	Date	Changes
1.0	03. 30.14	Creation of document

2- Overview

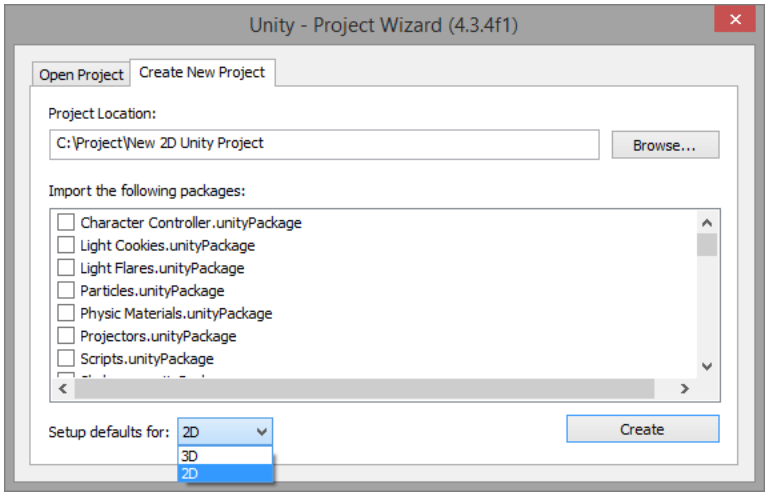
2D Complete Game is a template based on Boopha game that was in Windows Phone Marketplace. The original code was been made in XNA. We would like to share it with the community, we've made some changes and portability to use it on Unity. We hope you learn a lot of things with it and can create your own games. Enjoy! =)

Many people have difficulties to use Unity's native 2d feature. We decide to use a lot of this feature and we include other functions that you could take to other projects too.

Project Setup is part of learning. Although being boring, if you don't know yet, you will learn a bit more. Feel free to modify to your preferences, because we made easier for beginners.

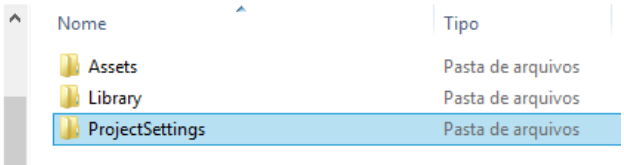
3 - Project Setup

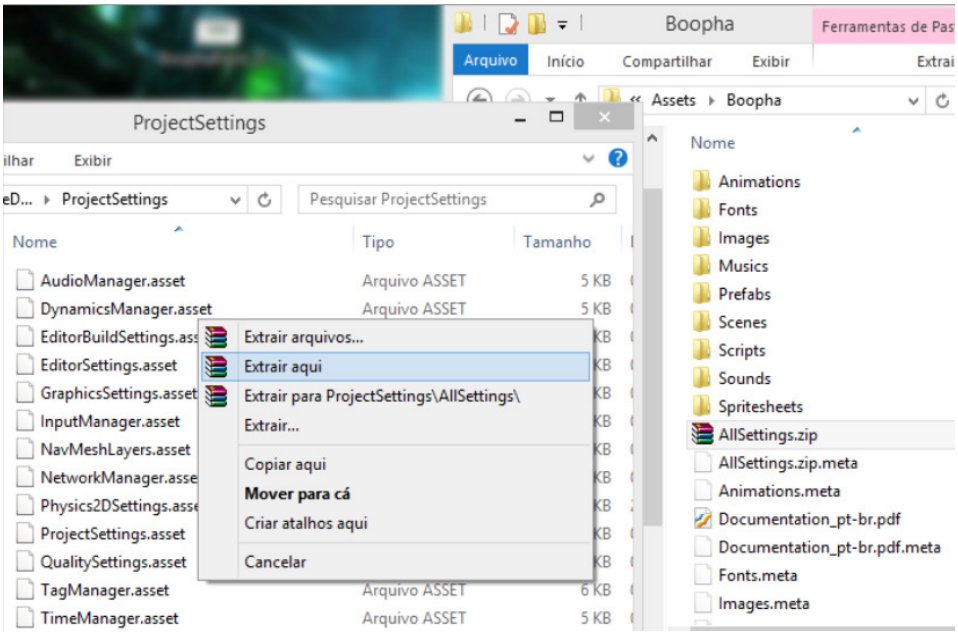
Create a new project on Unity and select "2D" at this window below.



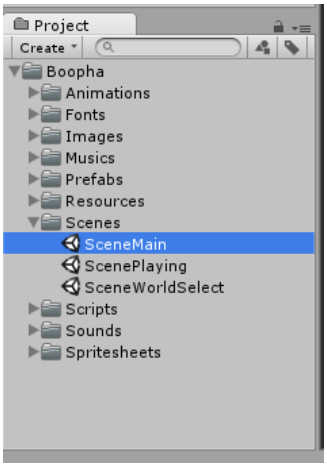
You can do this by two ways, simple and quick or step by step.

1- Simple and Quick: Just close Unity. Go to your project asset's folder, extract all files from Settings.zip into Unity's ProjectSettings folder. This folder is at your project root folder. Like this:

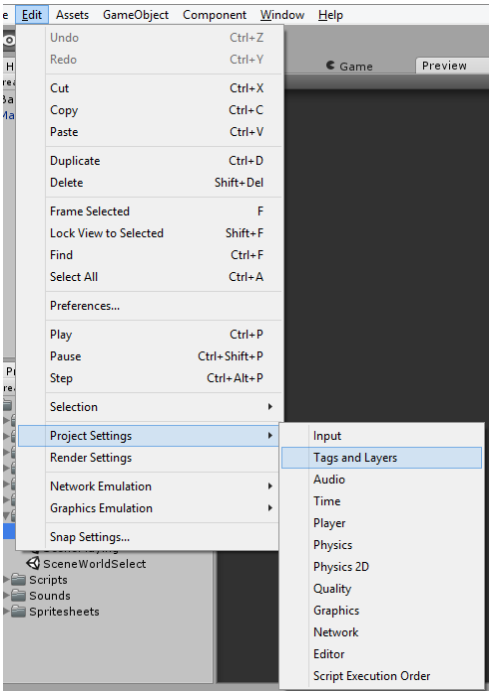




2- Step by step: Follow next steps.

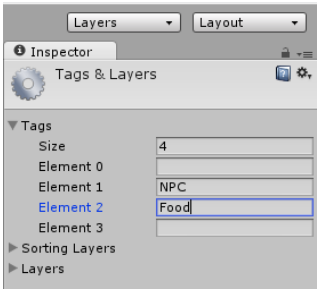


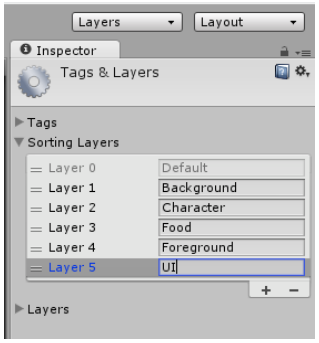
Once project was created, you have to import the 2d Complete Game package from Unity's Asset Store, then open SceneMain by double click. The game starts at this scene, but don't press play now. Go to menu ***Edit > Project Settings > Tags and Layers.***

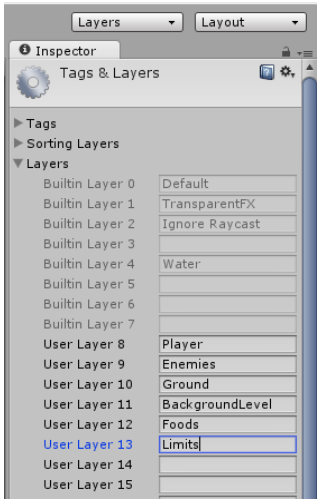


Now we will configure Tags and Layers. By default unity does not export these settings so we have to do it. It's cool, you will understand layers concept.

Create new layers and tags like images below.
Tags: NPC e Food.
Sorting Layers: Background, Character, Food, Foreground e UI.
Layers: Player: Enemies, Ground, BackgroundLevel, Foods e Limits.
Please, use same numeric positions that are showed in this images. This is very important.

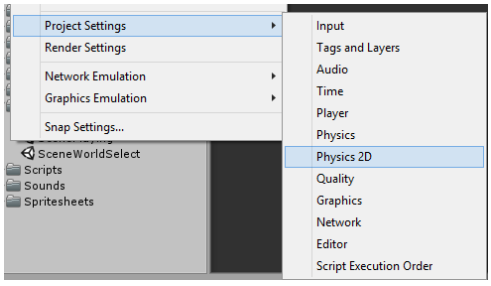


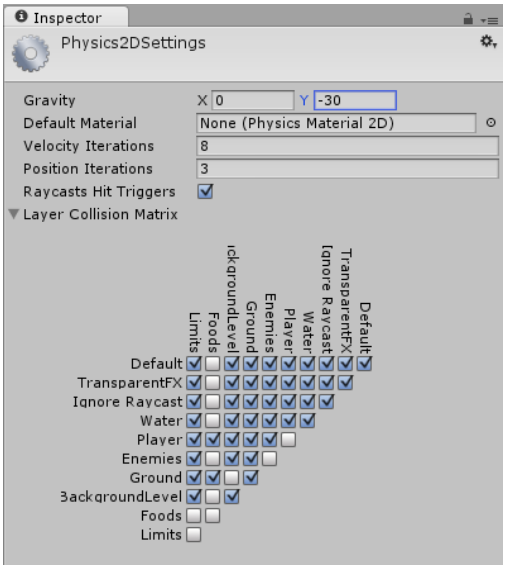




Next lets configure wich layers will combine/touch others. This setting is very important, because here we will know which layers will collide. Do the same as step showed before. Go to Physic2D at menu ***Edit > Project Settings > Physics 2D.***

Then, the matrix with layers will open at Inspector window. Configure it conform the image and don't forget to adjust gravity just like the images below.

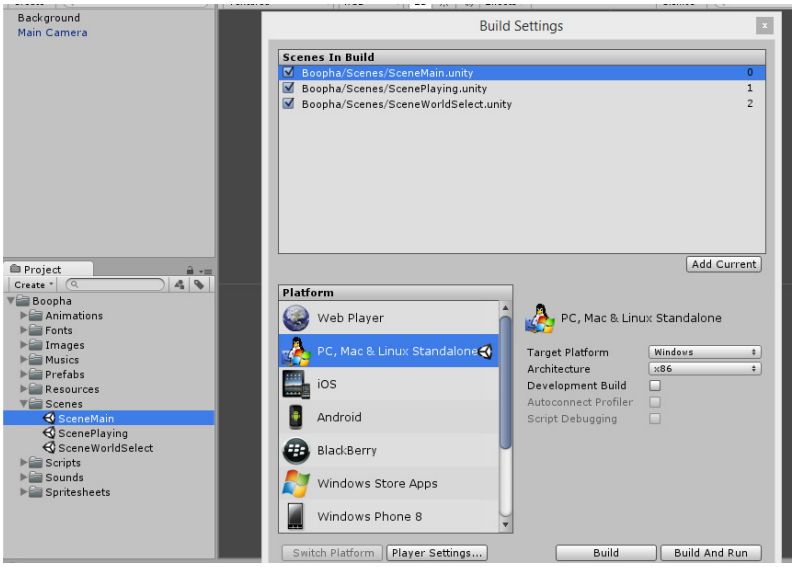




If you execute the game before this moment, probably a lot of errors will appear. Let's go to put it to work!

Open menu **File > Build Settings** and next drag and drop all three scenes into Scenes to Build. Must be like image below:

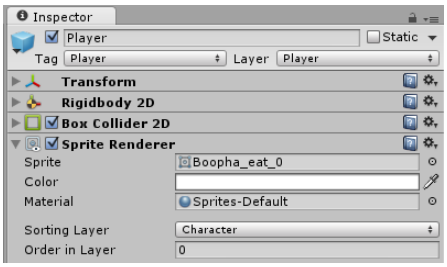
Next, just close these windows.



3.1 - Prefabs Adjusts

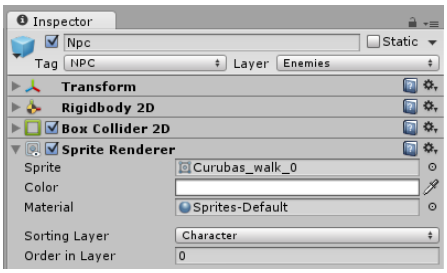
If you do all right until now the project is still running. But you will have physics problems if you don't adjust prefabs using layers and tags.

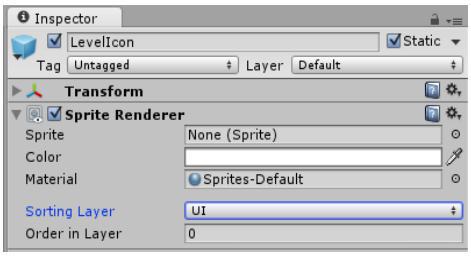
As I said, Unity doesn't export layers and tags settings, but we have created it. Let's configure our prefabs. Open Prefabs's folder and select our Player prefab. You will see something like this image below. Configure the prefab just like the next instructions.

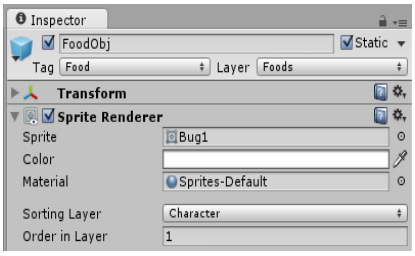


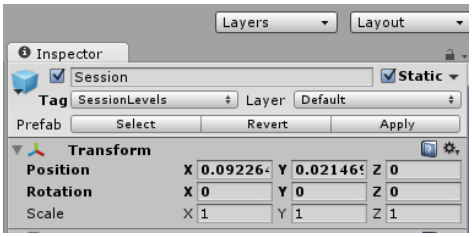
Check the tags Player and Layer Player. Adjust also *SpriteRenderer* component. In option *Sorting Layer* select Character. Check *Order In Layer* that must be same as image aside.

Here we will put others prefab's image that you have to adjust too. Be carefully here, do it all right :p

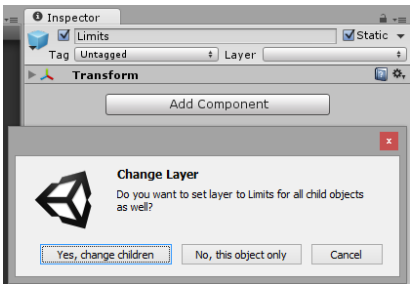








Select Limits prefab and change Layer to layer with same name. Probably your limits prefab have no layer selected. When you try to change it, Unity will ask you to change his children too. Answer Yes.

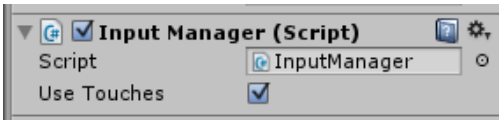


Ok, now you can press play and have fun =)

4 - Input

All scenes contain one InputManager component for one simple reason: it controls button to back scene by back or esc. At SceneMain the component is attached on "Background" object. All inputs are centralized in only one class named **InputManager**.

InputManager contains only one option at Inspector window. Check or uncheck to use touches or keyboard. Open script and you will find more details about this control. The default controls use horizontal axes at Unity's Input.



Be free to use other input control that you like. This is very simple to ease for beginners.

5 - Expanding the game

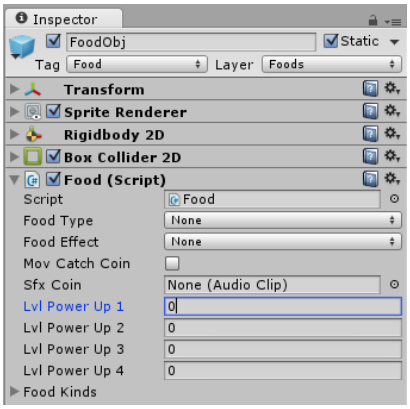
5.1 - Resolution Manager

The project is adaptable to the screen size of many devices. The component that controls that is ScreenAdapt at all OnGUI events. All OnGUI event must begin and end

calling the adapt methods. This may seems cumbersome at first but it is not and you will like the results. ScreenAdapt guarantees a centralized control of screen size functions at only one place, which affect the whole game, situation that makes future maintenance easier.

ScreenAdapt is a class that provides direct access through Singleton instance. You don't need to include it at scene or object. Just call for it at OnGUI event by this way **ScreenAdapt.Instance**.

5.2 - Foods and Behavior



To expand your game maybe you want new behaviors. Use foods for this. Foods can easily carry some behaviors to your player. It's possible to create powerUps, powerDowns and other cool things that are in you head.

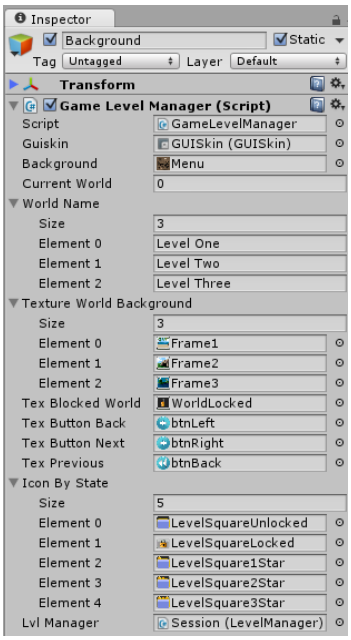
There are some options to configure The FoodObj. Use code to create several kinds at runtime. At **NpcController** class exists a method that creates a new instance of food. The food class is self managed, so it creates itself by random its kinds and behaviors. You can change this logic.

Exists two enums that help Food class. They are FoodType and FoodEffect. Although we don't use all options, I left then for example for you to create your own.

Check the Food class, because I left more explanations about the functions.

5.3 - Creating new Worlds and Levels

One way to expand your game is creating new worlds and levels. This process is simple and visual. At scene **SceneWorldSelect** these controls can be found at Background object and Session object. Configure both, and don't forget that Session object will be taken to playingScene.

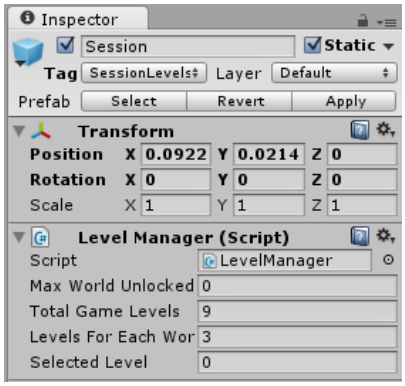


At **GameLevelManager**, a component that is attached to Background object that is in SceneWorldSelect, configure the worlds names. Increasing array size you can create new worlds. Increase equal size of Texture World Background and fill it with frames to be shown and select your world.

You must select the frames that will represent your worlds at world selection menu. This setting can be changed at **LevelManager** that is attached to Session object.

At Session object, as same step as before, we will just increase values of Total Game Levels, Levels for Each World and your levels and worlds will be shown at selection menu.

You also can change how many worlds will start unlocked by changing the MaxWorldUnlocked option.



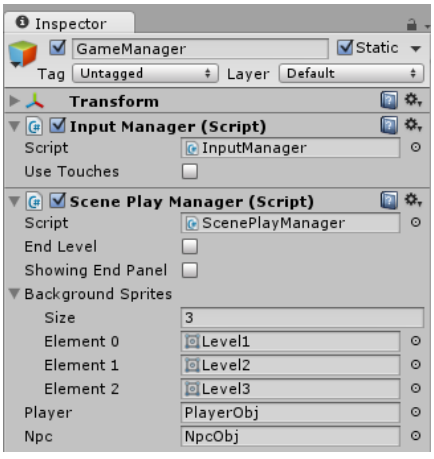
After you create your worlds you will aslo want to create his levels. To do that you will need to change **LevelManager** class. In it you can find a function that create levels automatically. You can change it to create infinite levels. Other way you can personalize it more by creating your own "create levels" method.

```
private void CreateLevels()
{
    //We dont want to recreate all levels. So check it.
    if(isLevelsCreated == false)
    {
        int worldNum =0;
        int lvCount =0;
        List<Level> lst = new List<Level>();

        for(int i =0; i < totalGameLevels; i++) // 9 tot
        {
            if(i >1 && i % levelsForEachWorld == 0) // 3
            {
                worldNum++;
                lvCount = 0;
            }

            Level lvl = new Level();
```

Check the LevelManager class and change what you think is necessary. There are more explanations.



Now go to ScenePlaying and select at Inspector window, GameManager object. Attached to it there is a ScenePlayManager component where you will configure backgrounds for each level. Increase Background Sprites array to fit your size of worlds and then drag and drop your background sprites from Images folder (or from other that you are using).

At this object you also can configure if you will use touches or keyboard to control your player.

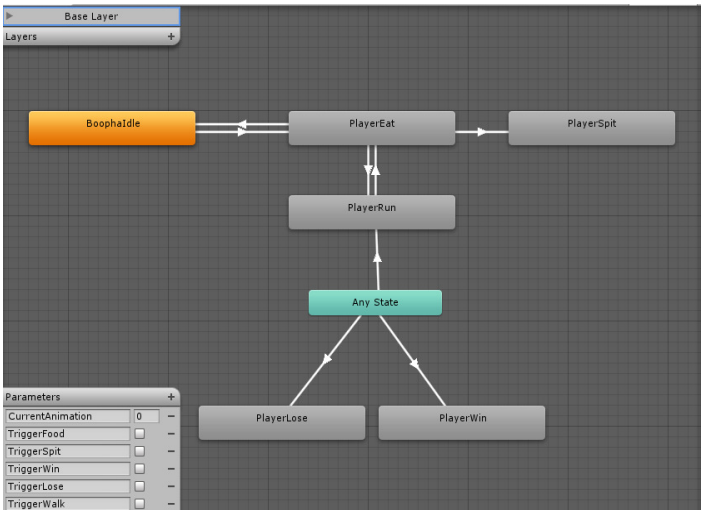
All done, your levels are ready To Play!

5.4 - Player and Animation

The player prefab has two components that define it: the **PlayerController** and **PlayerAnimMachine**. This second class is responsible to simplify animation controls. This code was taken out from PlayerController's main class.

Both objects, Player and NPC have animations based on Mechanim. They have trigger controls and current status control. These controls are at Animations folder.

All animations were created by spritesheets. These spritesheets and his unity's cut are included at SpriteSheets folder.



Thank you! If you have troubles or questions, just contact us:

www.facebook.com/ToPlayGameStudio

http://www.toplaystudio.com.br/

contato@toplaystudio.com.br