Flappy Bird Devkit/source

This is a complete development kit for Unity game engine for creating “Flappy Bird style” games. It is working with Android, IOS, Windows Phone, and all other Unity target platform. But I only attached an adMob advertizing client, which is for Android only, you can add more.

Unity version needed: 4.3.x, tested on 4.3.4.

This is a complete working game, but I used NGUI for the menu (start/score buttons), so I can not include it in this project, but there is help to do it with your code.

The entire game is written in C#, clean well commented (full) **source code** is part of the project. For saving the player data I prefer to use this Unity plugin: http://u3d.as/content/icedev576/secure-store

All assets are custom made but I suggest changing with your own.

Support:

- Developer homepage and issue tracking: https://github.com/icedev576/flappybird-devkit
- Email: icedev576@gmail.com

Table of Contents

Source (C#) files:

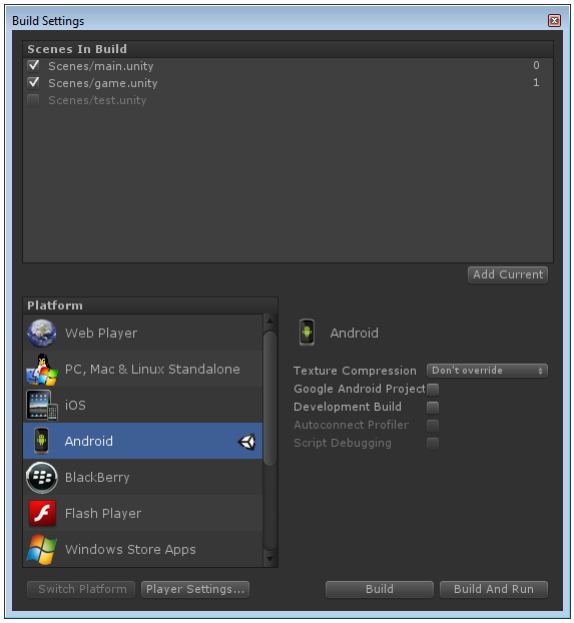
All files are under the Assets folder

- **autokill**.cs: attach this sript to the pipe prefab and it will be automaticly destroyed if it reached the left world limit.
- **Limits**.cs: world limit constant, edit this file to redifine a new world.
- **main_menu**.cs: this is very simple main menu, if user touches the screen or any button is down we simply load the game level.
- **mainLogic**.cs: this controls the game, procedurally generates the pipes while the player moving forward the scene.
- **pause**.cs: a very simple pause script, the gui will not be paused (IgnoreTimeScale)
- **playerLogic**.cs: this class controls the player. It checks the input, if the player is touched the screen or is any key down (for testing in the editor). It checks the world limits. There is two main event for the player: collided with a pipe, player should die, or flied through the pipes, in that case the score will be incremented by one.
- **reset**.cs: A very simple reset script, loads the last loaded map.

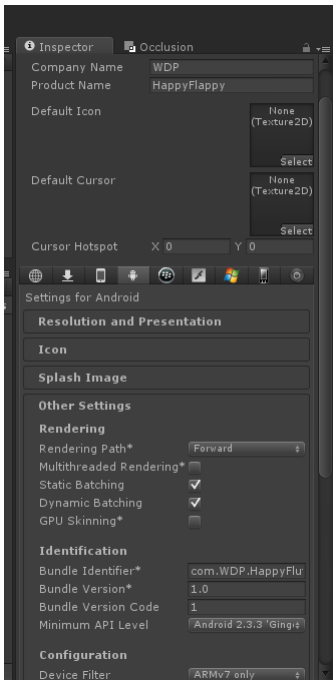
Start Guide:

1. Import the Unity package into your editor, see Importing Unity Package section
2. Load the Main scene: File/Open Scene, then find the main.unity under the Assets/Scenes folder
3. Press Play button to start testing the game
4. If you want to tune the game mechanics please read the following sections:
    a. Using the adMob asset
    b. Changing the background image in the main scene
    c. Game Logic object
    d. Player Logic
    e. Changing the pipes appearance

5. Building the Android Package: after you successfully followed the instructions in Importing Unity Package section you can go to File/Build Settings, then click on the Player Settings:



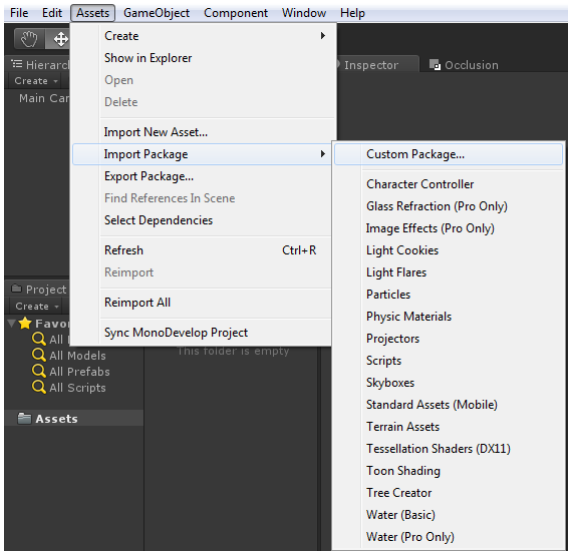
6. The Player Setting is found in the inspector window:



7. You should define: Company name and Product Name, the Bundle Identifier is automatically generated from these. Define the minimum API level. Other improvements can be found under the “Resolution and Presentation”, “Icon”, “Splash Image” tabs
8. Go back to File/Build Settings and click on Build
9. Hopefully you got an APK file, you can upload it to Store.

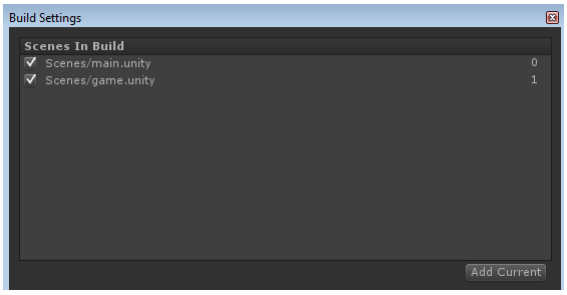
Importing Unity Package:

Asset/Import Package/Custom Package:



Find your downloaded FlappyBird_devkit.unitypackage file. Then open, then Import.

After that you should add scenes to you build. File/Build Settings:



And you may want to choose your platform (e.g: switch platform -> Android).

I suggest changing the Game view from “Free aspect” to “16:10 portrai”.

Scenes:

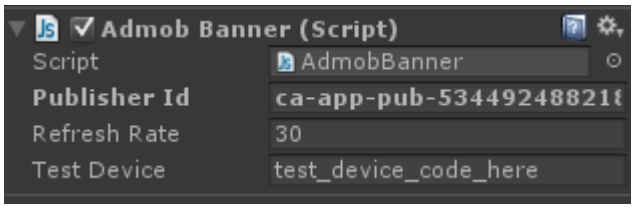
- Main: this is the starting screen, with instructions, next level is the game
- Game: the actual game, the map is generated procedurally; also the background image is randomly chosen when starting a new game.

Asset folders:

- Animations: currently only contains one animation for the flying bird
- Images: contains all images used in the game
- Prefabs: contains the pipes and the player as a prefab. There is also an ingame_menu prefab what I used with NGUI
- Scenes: the two scene file
- Sounds: all sound effects
- Admobbanner and Plugins: an adMob plugin for Android version. Other platform may use other advertizing plugins.

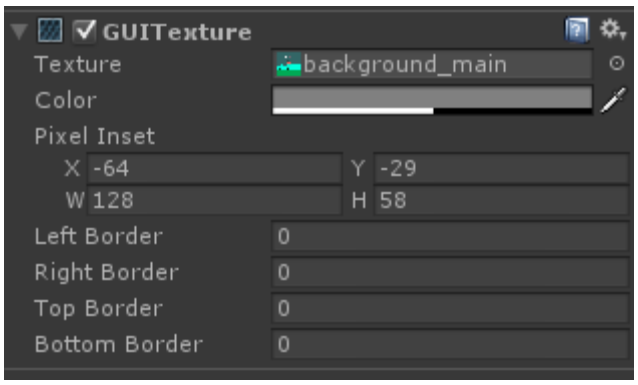
Using the adMob asset:

Simply put the prefab to the scene, but don`t forget to setup the Pubilsher Id, what is actually the Ad unit ID, it is generated via the http://www.google.com/ads/admob/ if you add a new application to you profile.



Changing the background image in the main scene:

It can be simply done by assigning a new gui texture for background_main game object:



The Game Scene:

You can found some game objects in the Hierarchy:

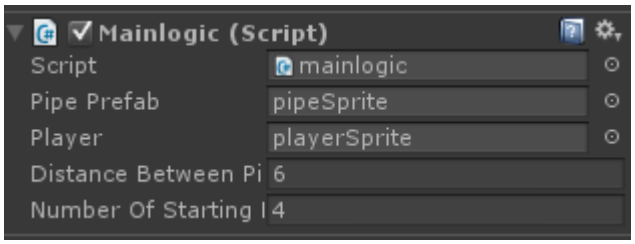
- adBanner: the adMob client
- background_day: a background image with a suny image
- background_night: a background image with a night scene
- Background Camere: used for rendering the background image
- **gameLogic**: this controls the game, only one script is attached to it: the mainLogic.cs, , setup instructions in a separate paragraph
- Main Camera: renders the scene in the center with the player
- **playerSprite**: the actual player, setup instructions in a separate paragraph

Game Logic object:

There are two objects assigned to this script: the player instance and the pipe prefab.

- The player should be an instanced game object what you can find in the hierarchy view.
- The pipePrefab should be chosen from the prefabs folder.
- The Distance Between Pipes: is the minimum horizontal distance for between pipes while generating them randomly

- Number of staring pipes: how many pipes should be when starting a new scene



The player:

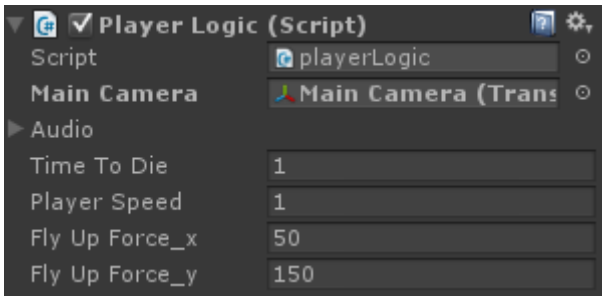
The playerSprite game object has a lot of scipts attached to it:

- Sprite Renderer: you can change the bird image here
- Rigidbody 2D: with the gravity scale you can define how fast should be dropping down the bird when the user not touching the screen
- Animation: called when the user touched the screen
- Polygon Collider 2D: for detecting the collisions
- Player Logic: this will control the player, setup instructions in a separate paragraph
- Several Audio Sources: all sound effects should be placed here

Player Logic:

The main camera should be assigned, because this will follow the player.

- Time To Die: in seconds how many time we should wait before we kill a dying player (for avoiding stucked player)
- Player Speed: player horizontal velocity
- Fly Up Force_x: player horizontal acceleration when touched the screen
- Fly Up Force_y: player vertical acceleration when touched the screen



Changing the pipes appearance:

In the prefab folder you can find the pipeSprite prefab edit this prefab for changing the pipe image and the distance between the vertical pipes.

Background images:

Create as many background image as you want, simply duplicate one of the given (e.g: background_nigh game object), and change the image. The mainLogic script will automatically detect all objects which have a **background tag** on it, and will randomly choose from them

Scripts:

In the root of the asset folder you can find the scripts:

- **autokill**.cs: attach this sript to the pipe prefab and it will be automaticly destroyed if it reached the left world limit.
- **Limits**.cs: world limit constant, edit this file to redifine a new world.
- **main_menu**.cs: this is very simple main menu, if user touches the screen or any button is down we simply load the game level.
- **mainLogic**.cs: this controls the game, procedurally generates the pipes while the player moving forward the scene.
- **pause**.cs: a very simple pause script, the gui will not be paused (IgnoreTimeScale)
- **playerLogic**.cs: this class controls the player. It checks the input, if the player is touched the screen or is any key down (for testing in the editor). It checks the world limits. There is two main event for the player: collided with a pipe, player should die, or flied through the pipes, in that case the score will be incremented by one.
- **reset**.cs: A very simple reset script, loads the last loaded map.

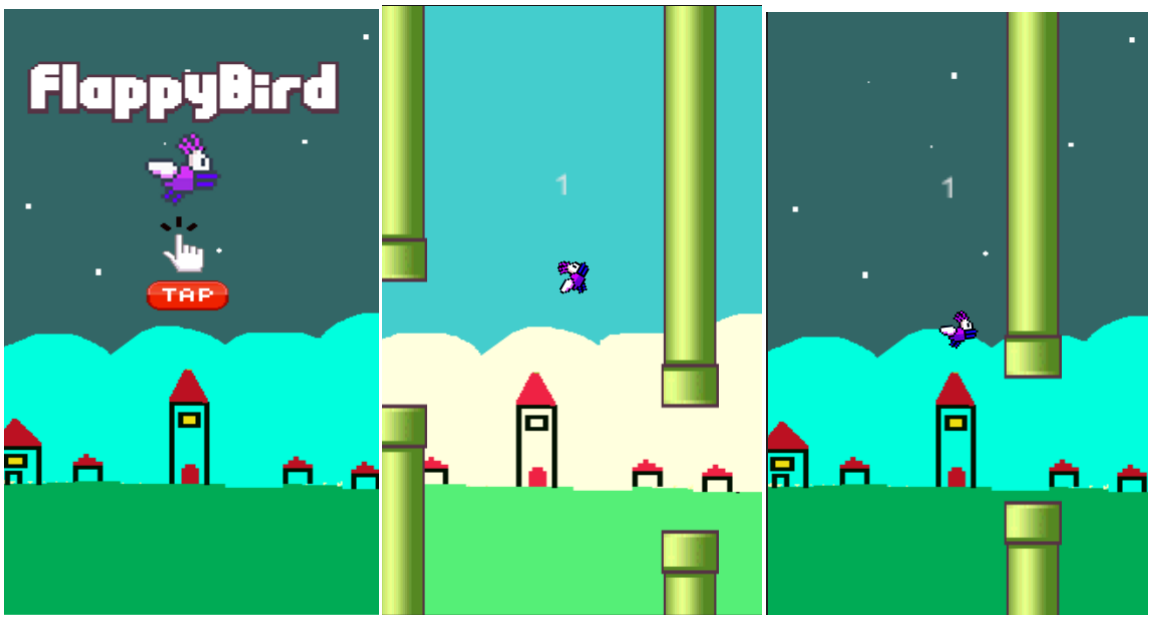
NGUI integration:

The NGUI is a separate Unity package it can purchase via the asset store. The mainLogic and playerLogic scripts contains some NGUI specific data, If you are using NGUI you can uncomment the second lines in these scripts (*//#define USE_NGUI*). In that you should manually assign the NGUI assets whit the scripts (pause, reset etc).

Alternative GUI:

You could choose any other GUI systems it will not interfere with the game. Although there is a fallback system: the example project in this package is using the default Unity GUI.

```
// for a fallback if no NGUI is defined, use the default gui:
#if !USE_NGUI
void OnGUI()
{
	var temp = GUI.matrix;
	GUI.matrix = Matrix4x4.TRS(Vector3.zero, Quaternion.identity,new Vector3(Screen.width / WIDTH * scale,
Screen.height / HEIGHT  * scale, 1));
	GUI.Label(new Rect(65, 70, 150, 100), "" + score );
	GUI.matrix = temp;
}
#endif
```



Questions:

If you have any other questions or issues don`t hesitate to contant me: icedev576@gmail.com, or https://github.com/icedev576/flappybird-devkit