LP2B – Multimedia: Digital Representation

- LP2B Practicum Class n°3 -

Final Project : the 3-in-1 App !

Putting it all together with transfers and a menu

This Practicum is about applying knowledge from ALL the Tutorials

What are we going to Make ?

- We want to regroup THREE games within a single Unity App
  Two previous ones *(Bricker Breaker, Apple Catcher)* , and a new one *(Flappy Bird)*

- The app initializes with a menu that has 3 buttons. Clicking one of those buttons will initialize the matching game by loading the corresponding scene

- We will slightly alter our 3 games so that pushing the « Escape » key during any of them will return the user to the menu

- And the new Game, the Flappy Bird clone, has its own list of requirements !

- And now you will probably say…

This is CRAZY ! How can we do so much in only one Practicum ?!

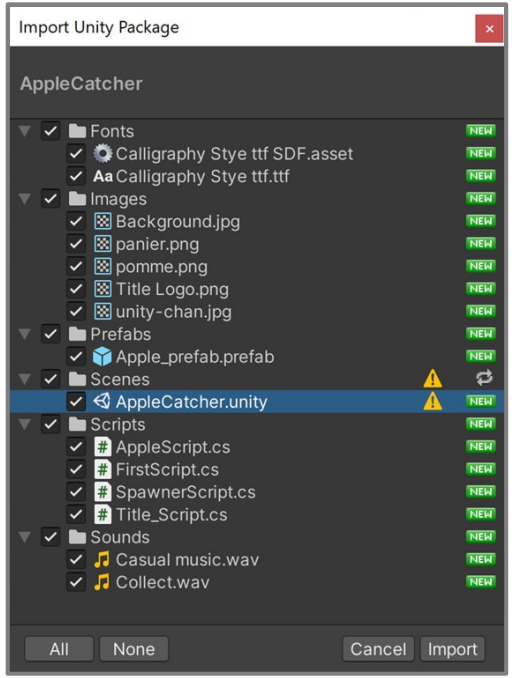
…But don't panic. We are going to « recycle » !

A Menu…

…To access 3 games !

Importing a Pre-Made Unity Package

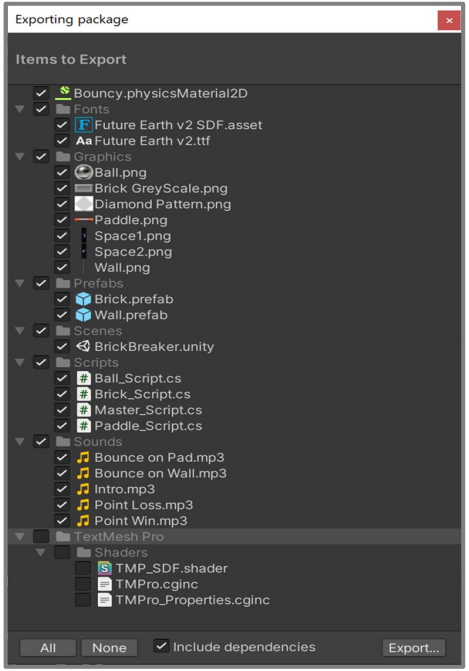
- Unity allows the transfer of data between projects with Asset Packages
  Those packages are not just about code. They can also contain assets, scenes…
- And good news : I already prepared a package with « Apple Catcher » in it !
  Just import it, and you'll have the whole scene and assets ready to go !
- To import the file : *(Assets => Import Package => Custom Package)*
- Unity will then show you the contents of the package, allowing you to review what you're about to import.
  You can even choose to only import parts of the package. Here, please import all *(default)*
- A warning should display in the Console. Don't worry, it's just because the Scene you imported was « number 0 » in the original project
  And the project you have here probably also has a scene 0. The conflict is solved automatically, but you'll have to add the « Apple catcher » scene to the build settings menu, in order to give it a new number



The import Window allows you to review what you are about to import.
Unity Notifies you of any conflict or potential problem with warnings and/or errors

Exporting Brick Breaker as a Package

- You can create your own packages to export any part of a project you want
  You should have done it already as a way to send me your Practicum Projects
- Make an export of your own version of « Brick breaker »
  Export the WHOLE ASSET FOLDER except textMeshPro in your package
- Import your package into « big final project », like we did for Apple Catcher
  You should likely get the same warning for the scene. But this is is fine
- If you have problems, it's probably because some entities in your different projects share the same name
  The package manager will point that problem out and help you to solve it
- Once the Import is Successful, open the scene with the brick breaker game and play-test it to check it works like it did before
- If you have trouble with this, the brute force solution is to copy-paste the files in the new project. *(But this is more likely to cause issues. Try to use packages.)*



The Package export Window allows you to review what you are about to export
TextMeshPro doesn't have to be part of your export : it's added automatically by Unity when needed

The Game Selection Menu

- The most important part of this final project is the GAME SELECTOR MENU

  There, you will use Video, Canvas and Animations. The big concepts of the last tutorials

- <u>Requirements for this menu :</u>

  1) The « planets » are, in fact, buttons. Clicking one loads the corresponding game
  2) The Planet buttons MUST use the Unity Button class. Meaning they must also be part of a Canvas
  3) The Planet Buttons use animations so the user can easily understand that they are interactive
  4) The background is a video. Make sure it will ALWAYS appear behind the planets
  5) The video and images necessary to make this menu are all available on Moodle
  6) There are no « extra objectives » here. Remember you will need time to make the Flappy Bird clone !

Part 2 : The Flappy Bird Clone

- Because the menu was a bit too short for a final project, you also have to make another mini-game. But don't worry, not from scratch !

- Flappy Bird is a simple mobile game where the user controls a bird and has to dodge obstacles that come from the right

  ...Those obstacles look awfully familiar, don't you think ?

- The bird only has one action : going up, as he naturally goes down over time

  This is why that game is very suited for mobile devices : simple controls !

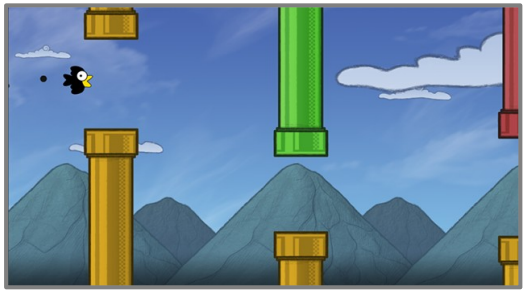
- There are two lose conditions : hitting a pipe, or falling down the screen

  The goal of the game simply is to last as long as possible. So we'll put a timer.

- Because there is a lot to do in this project, I helped you a little bit :

There is a Unity Package on Moodle that contains some pre-created elements, which will help you save time !

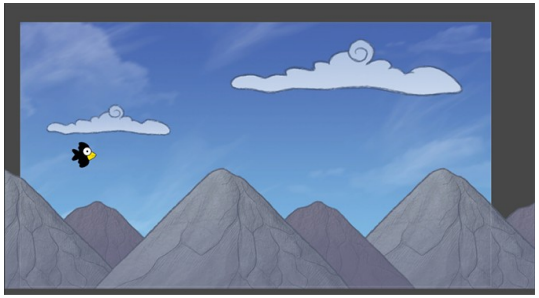
The original Flappy Bird



The clone you're about to make

The Contents of the Package

- The Package I am providing you contains a scene with graphics in place and a prefab for a « pipe obstacle »
- The « pipe obstacle » carries a RigidBody2D, colliders on its children, and a basic movement script which moves it to the left
  This script can be modified freely, you can add, remove, edit as you wish.
- The bird *(player character)* already has a RigidBody and Collider, but no script
  This is why that game is very suited for mobile devices : simple controls !
- <u>But then, what is missing in order to get the desired Flappy Bird game ?</u>
  1) A « Game Master » that will spawn pipe obstacles at different heights and intervals
  2) A score counter that either displays the distance scrolled, or the time the player managed to survive *(both work as valid ways to evaluate performance here)*
  3) Proper detection of the defeat conditions (impact with a pipe / falling)
  4) Some animation and graphic work



The Package contains a Scene called FurapiBird, which looks like this. It is incomplete, but having the images in position will save you time !

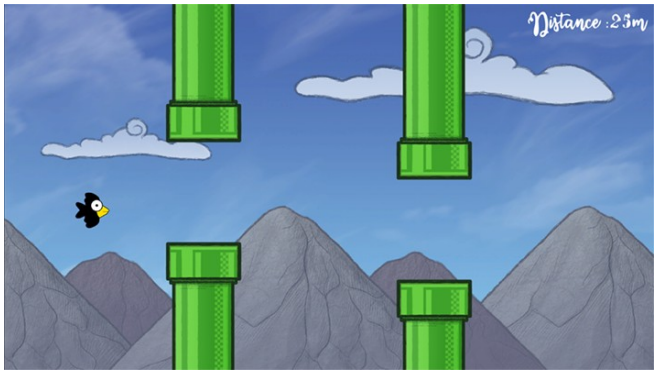
The package also comes with an Obstacle Prefab, which already has a basic script

Importing the package also imports its folder structure. If you don't like it, feel free to reorganize !

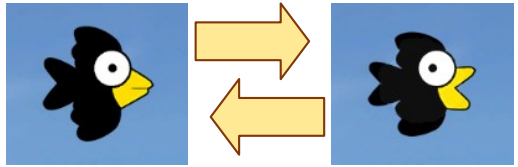
Requirements for « Furapi Bird »

Here are the features you must implement in Furapi Bird :

(There is a video on Moodle that shows how the game looks with those done)



- Pushing the « up » key should make the bird rise
- Obstacles must spawn at random time intervals, with different heights
- Give the bird an animator and a default animation as shown here
- Add a text field to display the distance travelled to act as score
- If you want, you can use the time survived as score instead
- The text field should use a custom font *(you can re-use the « apple catcher » one)*
- When the bird collides an obstacle or falls down the screen, it dies
- When the bird dies, you must block inputs so the player can no longer act
- When the bird dies, you must switch it to another animation, as shown here
- Play a constant music loop, and « impact sound » when the player dies
- After player dies, wait 2 seconds, then return to the game select menu



Alternating between those two drawings is enough to do the default animation
Make sure that animation is set to loop

Use this drawing as the death animation

Extra Objectives for this Project

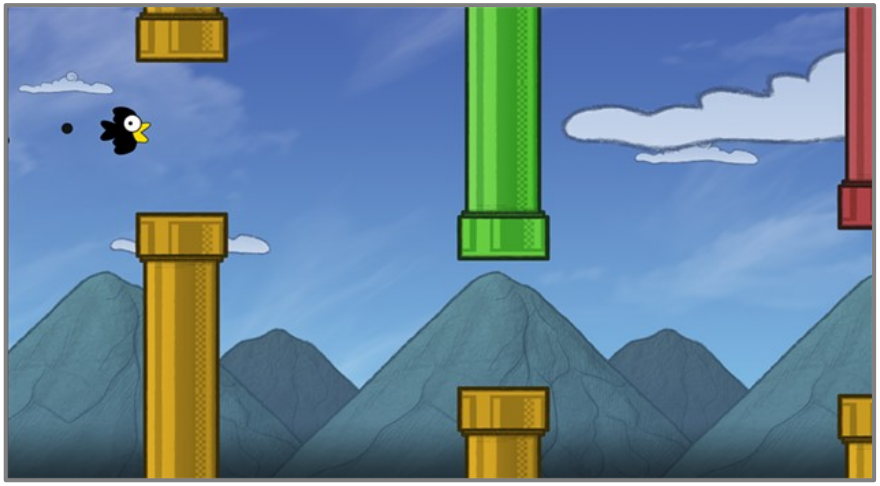
Here are extra objectives you can do for this final App :
(Doing them is what will allow you to go above 16/20)

IN THE MENU PART :

- Add music and a sound when clicking the button
- Have the music loop stop when clicking a button
- Try to make the animations look and feel enjoyable

IN FURAPI BIRD :

- Animate the background to give the illusion of forward movement
- Have the speed of the pipes increase slightly every 30 seconds
- Randomize the colors of the pipes for variety
- Stop the main music loop when the player dies
- Play the « death jingle » sound after the « impact sound ».
- Tweak timing so the game returns to menu after both sounds play
- Stop all movement of the pipes when the player dies



Feel free to take initiative !
Any extra feature you want to implement is acceptable,
As long as it doesn't contradict other previous goals !

Returning to Game Select Menu

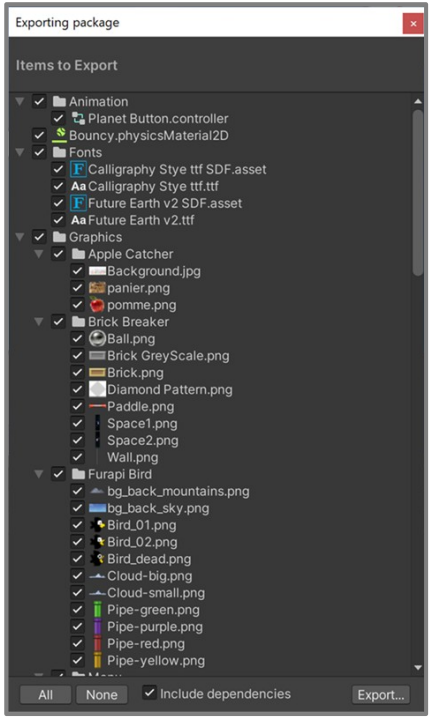
- To really tie our App together, we want give the user the ability to exit a minigame at any time, by pressing « Escape »
  This will allow our player to try a game, and then another, without having to exit the program
- Doing this is not any different than what you had to do for the menu : you will need to use a variant of the « Scene loading Coroutine »
  This particular Coroutine was given to you explicitely in tutorial class. You can just copy it !
- Just remember to implement this functionality in all 3 minigames
  I don't mind if there are several copies of the coroutine in different scripts
- It would also be nice if pressing « Escape » on the menu exited the application
  The line of code to achieve that is given below. Just place it in a correct « if...then » test
- The line of code to close the application is : `Application.Quit();`

  Warning : it doesn't work in the test environment. Only in your final build !

Press « Escape »

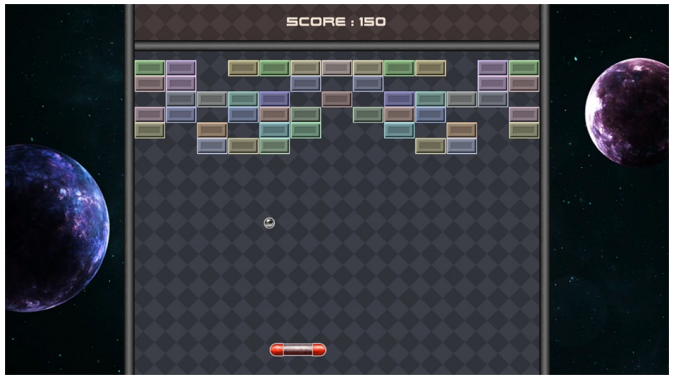
How to Return this project

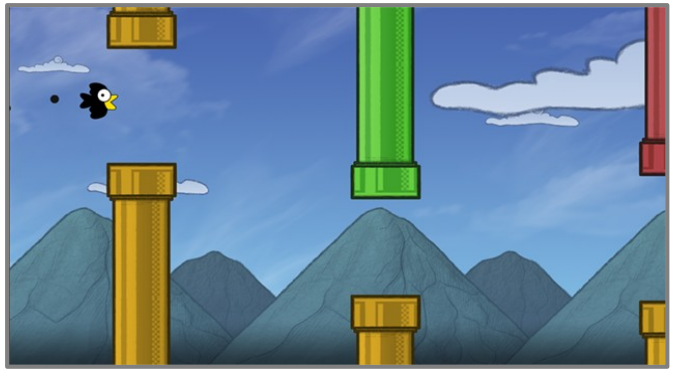
- This project represents a whole semester of work. A big accomplishment !
  And that's why I think it deserves a proper exe build for me to evaluate it !
- As usual, I will need a copy of your Assets folder. Either as a raw ZIP or as a package. If you go for package, please still place it in a ZIP file anyways
  Even I have a build for playtesting, I will still need your Asset folder to review your scripts
- Make a build of your App and join it as a ZIP, separate from the rest
  This means there are two ZIP files to send : the build ZIP and the Asset folder ZIP
- This will result in fairly large files. Please use moodle to transfer them
  They will be rejected as e-mail attachment due to their size, so we have to do this
- You have until …… to return the project.



Try to give (another?) shot at making a package. Test it by importing it in a blank project before sending it to me !

Godspeed, Unity Creators !





Good luck for this final Project !

It is pretty ambitious indeed ! But it also allows me to review everything we did together in one neat project !

Thanks to the power of « recycling », it won't even be that much work to compile everything together.

Since this project has « a little bit of everything », use it as an opportunity to play on your strengths !