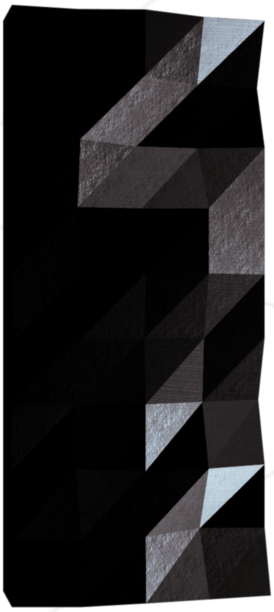

Piscine Unity - Day 03

Advanced inputs and 2D GUI

Staff staff@staff.42.fr

Summary: This document contains the subject for Day 03 for the „Piscine Unity" from 42

Contents

Chapter I

General Instructions

- The Unity bootcamp has to be made entirely, exclusively and mandatorily in `C#`. No `Javascript/Unityscript`, `Boo` or any other horrors.

- The use of functions or namespace not explicitely authorised in the exercise header or ini the rules of the day will be considered cheating.

- For a optimal usage of Unity, you have to work on `~/goinfre`, which is on the local drive of your computer. Remember to make appropriate backup on your own, the local goinfre can be purged.

- Unlike any other bootcamps, each day doesn't require a folder `ex00/`, `ex01/`, . . . , `exn/`. Instead you'll have to submit your project folder which will be name like the day: `d00/`, `d01/`, ... However, a project folder, by default, contains a useless folder: the `"projet/Temp/"` sub folder. Make sure to **NEVER** try to push this folder on your repository.

- In case you're wondering about it, there is no imposed norme at `42` for `C#` during this bootcamp. You can use whatever style you like without restrictio. But remember that code that can't be read or understood during peer-evaluation is code that can't be graded.

- You must sort your project's assets in appropriate folders. For every folder correspond one and only one type of asset. For exemple: `"Scripts/"`, `"Scenes/"`, `"Sprites/"`, `"Prefabs/"`, `"Sounds/"`, `"Models/"`, . . .

- Make sure to test carefully prototypes provided every day. They'll help you a lot in the understanding of the subject as well as what's requested of you.

- The use of the Unity Asset Store is forbidden. You are encouraged to use the daily provided assets (when necessary) or to look for additional ones on the Internet if you don't like them, exception made of scripts obviously because you have to create everything you submit (excluding scripts provided by the staff). The Asset Store is forbidden because everything you'll do is available there in one form or another. However the use of Unity Standard Assets is authorised and event advised for some exercises.

- From d03 for peer-evaluation you'll be required to build the games to test them. `The corrector` will have to build the game, you must therefore always push projects/sources. You project must always be properly configured for the build. `No` last minute tweaks will be tolerated.

- Warning: You'll not be corrected by a program, except if stipulated in the subject. This imply a certain degree of liberty in the way you can do exercises. However keep in mind the instructions of each exercise, don't be LAZY, you would miss a lot of very interesting things.

- It isn't a problem to had additional or useless files in your repository. You can choose to separate your code in different files instead of one, except if the exercise's header stipulate a list of files to submit. One file must define one and only one behaviour, so no namespace. Those instructions don't apply to the `"projet/Temp/"` sub-folder which isn't allowed to exist in your repositories.

- Read carefully the whole subject before beginning, really, do it.

- This document could potentially change up to 4 hour before submission.

- Even if the subject of an exercise is short, it's better to take a little bit of time to understand what's requested to do what's best.

- Sometimes you'll be asked to give specific attention on the artistic side of your project. In this case, it'll be mentioned explicitely in the subject. Don't hesitate to try a lot of different things to get a good idea of the possibilities offered by Unity.

- By Odin, by Thor ! Use your brain !!!

Chapter II

Foreword

Today we are gonna play this.

Chapter III

Exercice 00 : A simple menu

	Exercise 00
Exercice 00 : A simple menu	
Files to turn in : `An ex00 scene file and everything that seems relevant`	
Forbidden functions : `None`	
Remarks : `n/a`	

`Do not hesitate to play the provided prototype for 5 min if you have doubts about what is expected from one of the exercises.`

Create a scene with a menu for your tower defense. Let's keep it simple for now, a Play or Start or Whatever button that will launch a game and a Quit or Exit button that will close the app.

The Play button must direct to the "ex01" scene which doesn't yet exist but will be created in the next exercise.

It is up to you to find a nice background and a font that generates passion!

`Today is quite a big day, don't lose too much time on making your menu !`

Chapter IV

Exercice 01 : Drag and drop

	Exercise 01
Exercice 01 : Drag and drop	
Files to turn in : `An ex01 scene file and everything that seems relevant`	
Forbidden functions : `None`	
Remarks : `n/a`	

`http://docs.unity3d.com/Manual/Layers.html is your friend! Both to manage the drag and drop and to avoid the ennemies/missiles/etc to go under your towers, for example.`

Create a bar in the bottom of the screen in which will the 3 basic towers a played can buy be visible.

To buy a tower you need to drag and drop it from the bar to the wanted location. Be careful to have the UI elements stay on the foreground and don't go under another sprite!

The tower is placed only if the square location is empty and if the player has enough remaning energy. In this case the tower's cost is take out of the player's energy reserve.

The bar must display the damages, cost, the range and the cool down of every tower as well as the player's life and energy. You cannot display hard coded values but must fetch them from the GameManager and the different tower's prefabs (In today's provided assets).

There must be a visual feedback - like a change of color of the tower for example - to see at first glance if there is enough energy in stock to buy the tower. It cannot be possible to move a tower if the energy available isn't sufficient.

The color attribute is very interesting because it can be SET but can be GET as well. It is therefore possible to test the color of a sprite which happens to be really useful in some cases. Go read the documentation for more infos.

Chapter V

Exercice 02 : Pause menu

	Exercise 02
Exercice 02 : Pause menu	
Files to turn in : `The same ex01 scene file and everything that seems relevant`	
Forbidden functions : `None`	
Remarks : `n/a`	

You will now add a pause menu when the player press the "esc" key. A function is already available in the GameManager to manage pause. You just need to use true or false as an argument to break and get back to the game:

Protoype :

```
public void pause(bool paused);
```

This menu must propose to resume the game or quit.

If the player quits, a second confirmation menu must be displayed. If the player confirms he will go back to the main menu.

Since we're working on time management note that a function to increase or decrease the game speed is also available:

Protoype :

```
public void changeSpeed(float speed);
```

You are free to choose your implementation but, you will at least need at least a button to increase the speed and another one to decrease the speed or pause the game. You will then win a lot of precious time during your test and evaluation phases. Both pause and changeSpeed functions are compatible with each other so you will automatically get the right speed when resuming the game even if you played with an increase speed.

A game with its own cursor is a little more stylish. Find a way to change it! Go through Unity's documentation should solve the problem in less than 5 min.

`A cursor is provided in the attachments today, but you are allowed to find it ugly and look for another one!`

Chapter VI

Exercice 03 : Rox or sux ?

	Exercise 03
Exercice 03 : Rox or sux ?	
Files to turn in : `An ex03 scene file and everything that seems relevant`	
Forbidden functions : `None`	
Remarks : `n/a`	

You must create the screen that displays teh score as well as the rank achieved by the player at the end of the game.

A score is already calculated in the GameManager so you can get it or create your own.

You can create ranks from F to SSS+ or find a little more creative names! The only constraint is to give to the player a rank that follows a clear logic depending on the HP and energy left, with at least 5 different ranks.

The screen must have a button for a replay if the player lost, or to go to the next level if the player did kill all waves of ennemies.

You must create a new map with a higher difficulty.

> ⚠ `If you want to change difficulty, be careful not to change the ennemies' attributes on the original prefabs else their attributes will be changed on EVERY map (prefabs are independant of scenes). You must update attributes at a later stage in a script or update some other game parameters.`

Chapter VII

Exercice 04 : More work?

	Exercise 04
Exercice 04 : More work?	
Files to turn in : `An ex04 scene file and everything that seems relevant`	
Forbidden functions : `None`	
Remarks : `n/a`	

> `Every tower upgrades are already ready to use in the prefab folder to make you save some time.`

To position tower is nice and great but, now let's upgrade them!

Create a radial menu that appear when the player right click on a tower.

The menu must propose to upgrade the tower. It is only possible to upgrade a tower from level 1 to level 2 and from level 2 to level 3. The interface must display the cost of the upgrade. Obviously, it must be impossible to upgrade a tower if the remaining energy is insufficient. When a tower is upgraded, the appropriate energy is deducted from the remaining energy bar.

The radial interface also allows to downgrade a tower from level 3 to level 2 and from level 2 to level 1 as well as sell the tower if it's level 1. The interface must show how much energy the player will get back when downgrading or selling a tower. It is up to you to decide how much energy back the player gets but usually it is around half the buying cost.

A third button must allow to simply close the menu except if you already manages it with a right click somewhere else for example. You are rather free in the way you implement this menu as long as it doesn't crash..

You must also create a more difficult map to test out towers and their hability to be destroyed.

Chapter VIII

Exercice 05 : For the progamers

	Exercise 05
Exercice 05 : For the progamers	
Files to turn in : `An ex05 scene file and everything that seems relevant`	
Forbidden functions : `None`	
Remarks : `n/a`	

To finalise this game a last thing is missing: keyboard shortcuts! Instead of stupidly drag and dropping a tower one by one it would be a lot better to select them from the keyboard.

Create a keyboard mapping system to select towers. For example when clicking on "q" or "1" the mouse cursor would be replaced by a cannon tower, all that would be left would be to select an available location on the map to put the tower.

Final touch, an emergency destruction spell if an ennemy is able to sneak through. Create a fireball icon in the bottom bar that can be dragged and dropped anywhere on the map to create an explosion. There must be a keyboard shortcut for this fireball spell as well as a cooldown to avoid the player using it over and over again.

You will find images that have not been used in the previous exercises, like range templates. It is up to you to play with them to finalise the interface and give the player a maximum of information to be able to increase difficulty and have a clear and rigorous game.

Don't forget to make a proper and stylish last screen if the player is able to finish your last level! You must always think about something to conclude a game and not a simple redirection to the title screen.

This time there is no prefab or script provided for the last features but if you are able to reach this exercise, you should be able to know how to manage.