FlyPort WorkTest Report

Facundo David Farall
November 6, 2020

1 INTRODUCTION

This report summarises the ideas and thinking process I went through when solving the given tasks. The main focus of the following sections is to provide additional information, not necessarily found when reading the code, about the approach I took on each exercise.

2 EXERCISE 1 SUMMARY: SINGLETON

The main challenge of this task was, in my experience, adapting what the Singleton Pattern stands for, to the Unity way of doing things.

My initial approach was to have the *Singleton* class inherit from *MonoBehaviour*, since that is usually the case in Unity. However, that implied two things: first, the user could have instantiated a *GameObject* with a *Singleton* component from the editor or with the method *Instantiate<>()*, which was against the whole Singleton idea; and second, even though I could take measures to prevent having more than one instance, even if the user did such things, since *MonoBehaviour* hides the class constructor, I had to have one *GameObject* with a *Singleton* component instantiated by me via scripting during run-time. Even though this does not violate the Singleton Pattern, it was not in strict agreement with the instructions given, so there were some of compromises.

So that brought me to the question "Do I really need *MonoBehaviour*?", and the answer was no. *MonoBehaviour* gave me the ability to add classes as components to *GameObjects*, having proprieties exposed to the editor, and handle Unity events. The first two were against what was asked, and the third one I did not need. Hence, the implementation you will find in my code does not inherit from *MonoBehaviour*.

3 EXERCISE 2 SUMMARY: POLYMORPHISM

Regarding the second exercise, the only thing I wanted to add was that in the given solution you will find a fourth script called *TriggerManagerAlt*, besides the three asked. That scripts stands as an alternative solution to the structure proposed.

I believed that instead of having one class with a fixed size of colour alternatives (two), and then have a child class redefine that fixed amount of variables, the whole thing could be replaced by one single class that exposes a list of colours, and then the user can set the size and elements of the list.