Tappy Plane

Tappy Plane is a *Flappy Bird* style endless runner in which the player taps the screen to avoid obstacles, thus earning coins. The project was created for educational purposes as part of James' **#GameInAWeek** challenge and is released freely under an MIT license.

Project Structure

This project utilizes the following project structure:

- Assets
    - Animations
    - Audio
    - Editor
    - Fonts
    - Imported
    - Localization
    - Prefabs
    - Scenes
    - Scripts
    - Sprites
- Project Settings
- UnityPackageManager

in which folders are discriptly named. **Import/DeFuncArt** contains assets which are utilized between James' various projects. This folder has a similar structure to the Assets folder, that is, containing Animations, Editor, Fonts, Other, Prefabs, Scenes and Scripts subfolders. *Scripts* contains a number of principles explained in more detail in various **#50-Unity-Tips** articles

Resolution

The game is optimized for mobile (Android and iOS), supporting the landscape orientation and aspect ratios 9:16 (iPhone) through to 3:4 (iPad). The game scene is cropped between these aspect ratios, with a background equal to the size of the largest aspect ratio (3:4). All UI elements are contained within a scalable UI Canvas. Larger aspect ratios, such as iPhoneX or 9:18, should be playable, but not optimized for.

Scenes

The game consists of simply two scenes *LoadingScene* and *GameScene*. GameScene is subdivided into five screens: *Menu*, *Settings*, *Game*, *Results* and *Shop*. Each screen is rendered on a separate UI Canvas.

Controls

The player simply taps the screen to move their plane vertically higher. For debugging within Unity, the developer can use the SPACE key.

Coding Principles

The C# language was used throughout the game, adopting the following principles:

- Classes are named with PascalCase, i.e. MyClass
- Methods are named with PascalCase, i.e. MyClass.MyMethod()
- Properties are named with camelCase, i.e. MyClass.myVariable
- XML Documentation with the <summary> tag on all classes, methods and properties.
- Unless designated public, adhering to encapsulation, all properties are private or protected and exposed to the inspector via **[SerializeField]**.