Guess The Number

Adrián Núñez Garrido – Innovamat Technical Test

Access to the project repository:

https://github.com/anugaDev/Innovamat_TechnicalTest

Includes:

Build for Windows *(Builds/Windows)*

Build for Android *(Builds/Android)*

What I've Implemented

Board

The main flow of the game is contained within the *Board* environment. For dividing the responsibilities of displaying and controlling the game loop, the environment is implemented using the MVC architecture.

First, in the editor, the initial data is set up in the *BoardConfiguration* Scriptable Object, which allows the developer to easily modify settings as display time or number of panels. At the start of the execution, *BoardInstaller* is called by the *MainInstaller* to declare and inject the architecture with data. This approach is done to ensure the avoidance of crossed references and to make a clear structure initialization. *BoardInstaller* will also set up the array of number panels used during the game described in the *BoardConfiguration*.

BoardModel class is responsible of holding the environment data. It stashes an array of possible number texts, an array of numbers chosen to display on the current round and the index of the correct number within it. It also saves properties for current fails and successes of the user.

BoardView contains the properties related to the game display. This includes both the text panel containing the current number, a list of the intractable panels that will contain the numbers and text components for the user fails and successes. It also contains *UnityEvents* for starting and ending the game, called when entering the game from the UI menu.

BoardController manages the game flow and holds both *Boardview* and *BoardModel*. For a clean and readable code, this class is set up using a reactive approach with *UnityEvents*. When declared, the controller sets callbacks listening the start/end game events and the intractable panels from the *BoardVIew*.

For controlling wait times, the *BoardController* uses action callbacks setting corroutines, for example with the open and close animations.

Display Panel

To allow modulability, *DisplayPanelView* compresses the properties related to an intractable panel, instanced multiple times during the game. This includes text, button and open and close animations.

The subclass *NumberDisplayPanelView*, contains the exclusive properties related to an intractable number in the game window: the success and fail animations.

The animations are created using *Timelines*, which allow to create and organize the sequence of the animations in the editor.

Extras:

Localization

As an extra, a small localization system has been created. This system allows the user to change the language in which the menu and the numbers are displayed within the game. For a clean and consistent pattern through the project, the Localization environment is implemented using the same MVC architecture with an installer and a configuration scriptable object.

The *GameLocalization* allows the developer to easily setup different languages by allowing creating different *LanguageLocalization* scriptable objects.

Menu

Finally, a navigation menu has been added to allow the player to start the game, switch between languages and quit the application. This part of the application is mainly implemented using Unity UI Components and *UnityEvents* to set active the different menu interfaces within the scene. *ExitGameUtil* contains the calls necessary to exit the application.