Game manager: Implemented using the singleton pattern, due to it being adequate for organizing displays, transitions and motorizing the state of the game, and making all the rest of the game elements act accordingly.

Game sate: My first idea was to implement a state machine, but it would be too much since the application will never receive more than one input at a time, and there are really few cases where the user input is taken into account (when he decides the number to click).

Ease in – outs / color transitions: Simple structure that automatically takes into account child ui components, and enables – disables button input.

Victory/defeat counters: Instead of being notified by the game manager (and thus adding more functionality to it) they get the values every frame and display it.

Number table: Look up table that translates int numbers to their text translation (5 → five), really simple method and quick to execute and implement, but too rigid (I considered making a script that dynamically translates them was out of the scope).

About error handling: I considered special attention to the code as if it was going to be used in another parts of a larger project, deactivating and doing logs if the scripts / prefabs were not used properly to reduce possible crashes.
Despite that, in game objects that could not be deactivated (game manager) constant null checks are used to reduce crash probability.

Value to select: Script used to store values and read input from the user, and pass it to the game manager.