Development Report: Higher or Lower Game

DANIEL HERRERO

Contents

Overview

The Higher or Lower Game started as a simple command-line interface (CLI) game and later evolved into a more interactive graphical user interface (GUI) using Tkinter. This transition was motivated by the desire to enhance user experience and usability while ensuring the code was modular and readable.

Key Decisions and Rationale

1. STARTING WITH THE COMMAND LINE INTERFACE

Developing the game as a CLI provided a straightforward way to implement the core logic, such as card handling, game rules, and score management. It allowed me to focus entirely on functionality without the added complexity of GUI elements. However, working in the CLI was tedious and time-consuming. Debugging certain features also felt more abstract without visual elements.

2. TRANSITION TO A GUI WITH TKINTER

Moving to Tkinter made the game more interactive and user-friendly. Adding visuals, toggleable settings, and improved feedback mechanisms enhanced the overall experience.

Implementation Highlights:

Rules Display and Settings Toggles:

I gave users the ability to customise game rules, such as:

- Including or excluding Jokers.
- Assigning varying values to King, Queen, and Jack.
- Deciding whether Ace should be valued as 1 or 11.

These options made the game dynamic and appealing to different preferences.

Visual Aesthetics:

Card images were downloaded online, and their naming format made it easy to generate file paths dynamically based on the card's number and suit.

3. USE OF OBJECT-ORIENTED PROGRAMMING (OOP)

While the original CLI game was largely procedural, it already featured Card and Deck classes. These turned out to be incredibly useful during the transition to the GUI. As the project grew, the code became more object-oriented than I had initially planned, which helped organise components and keep things modular.

Benefits

- Encapsulation of card and deck operations made the logic reusable.
- The modularity of the OOP design simplified interactions between game logic and GUI elements.

Challenges

I initially relied on returning values between functions but found that using global variables was more practical for shared states. Although this added some complexity, it helped in coordinating the logic and UI.

4. HANDLING ASYNCHRONOUS BEHAVIOUR IN TKINTER

Implementing animations and timed actions using canvas.after and managing callbacks with lambda functions was a key challenge. Initially, keeping UI updates in sync with game logic was tricky, but these tools eventually allowed me to maintain control over timing and flow.

5. FOCUS ON READABILITY AND MODULARITY

I prioritised breaking down the code into smaller, purpose-driven functions, making the project more maintainable and easier to read. Separating code by function rather than creating large, monolithic chunks ensured that future updates or debugging efforts would be more manageable.

Ideas for Improvement

1. FURTHER DECOUPLE LOGIC AND UI

To reduce the reliance on global variables, I could implement a centralised controller class to manage the interaction between the game logic and the UI. This would improve modularity and scalability.

2. ENHANCE USER FEEDBACK

Adding animations or visual cues for correct or incorrect guesses could improve the user experience. Implementing a score history or leaderboard might also provide players with a sense of progression.

3. PACKAGE THE GAME

Using a tool like PyInstaller to create an executable version of the game would allow others to play without needing Python installed on their systems.

Reflections on the Development Process

LESSONS LEARNED

Starting with the CLI game helped me focus on the core functionality before adding complexity with the GUI.

Using canvas.after and lambda in Tkinter taught me how to handle asynchronous behaviour effectively and synchronise logic with user interactions.

CHALLENGES OVERCOME

Managing timing and state synchronisation using canvas.after was initially difficult but became easier with practice.

Transitioning to a more object-oriented design highlighted the importance of planning and flexibility when refactoring.

This report summarises the decisions I made during the development of the Higher or Lower game, the challenges I encountered, and areas for future improvement. Overall, this project was a valuable learning experience, especially in improving GUI development skills and balancing functionality with readability.