Guessing Game

CS 329E: Bulko
Programming Assignment 4
Table View Controllers

INF385T: McQuaid
Design Assignment A
Game screens

September 23, 2025

IMPORTANT NOTE: The first four sections below are for CS329E students. The last section is for INF385T students.

1 Problem Definition

There is a simple guessing game called ”high-low” where one person chooses a number in a specified range, and a second person tries to guess it in as few guesses as possible. After each guess, the first person states whether the guess was correct, too high, or too low. The game ends when the second player correctly guesses the number.

In this assignment, you will be implementing a variation on this game where, instead of a number, your app will choose a random card from a standard deck of playing cards, and the user attempts to guess the card by specifying a pip value (from low to high: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, or K) and a suit (clubs, diamonds, hearts, or spades). After each guess, the app will display three messages. The first message states whether the pip value was correct, too high, or too low. The second message states whether the suit was correct or incorrect. The third message will either indicate the number of guesses taken so far, or state that the user correctly guessed the card.

You will be working on this assignment with the UX Prototyping student(s) on your project team. Note that this is still an individual assignment, so you are NOT to collaborate with any of your other team members. The detailed instructions below only provide you with minimal guidelines for creating an app with a simple user interface. The UX students are tasked with creating a clean design for the app, and you must implement the design they provide you with. This means *you are expected to meet with each other and collaborate*. You should read section 5 below to help you understand what the UX students are being asked to do.

2 Detailed Instructions

1. On the starting screen, create the following:
   (a) A label containing the message, ”I’m thinking of a card. Guess what it is in as few guesses as possible!” Use the Attributes Inspector to set the number of lines in the label to two, so there will be enough room for the message to wrap around.
   (b) Add labels for ”Pip value” and ”Suit”.
   (c) Add buttons next to the labels. You do not have to change the text in the buttons on the storyboard, since you will doing that programmatically.
   (d) Add a button ”Submit guess”.
   (e) Add three labels, one for pip messages, one for suit messages, and one for guess messages. Initially, all three labels will contain no text.
   (f) Create connections.
2. Create a new view controller for selecting a pip value to guess.
   (a) Add a Table View to the default View Controller. Pin the edges so that it occupies the entire screen.
   (b) Set the number of Prototype Cells to 1, and the Table View cell style to `Basic`.
   (c) Create a segue by ctrl-dragging from the pip button to the new VC.
   (d) Create a custom class (in a new file) to handle things in the new VC.
   (e) Add the `UITableViewDataSource` and `UITableViewDelegate` protocols to the View-Controller class.
   (f) Identify the delegate and data source in the `viewDidLoad()` method.
   (g) Create a ”data source”: an array containing the strings ”A”, ”2”, ”3”, ”4”, ”5”, ”6”, ”7”, ”8”, ”9”, ”10”, ”J”, ”Q”, and ”K”.
   (h) Implement the two methods required for a `UITableViewDataSource.`
   (i) Add code that, when a user selects one of the pip values from the table, remembers the value in a global variable and then dismisses the table view.
3. Create a new view controller for selecting a suit to guess. Follow steps similar to those in the previous view controller, using an array containing ”clubs”, ”diamonds”, ”hearts”, and ”spades” instead.
4. Add the key functionality to the starting VC.
   (a) In `viewDidLoad()`, generate a random card.
      i. Hint 1: generate a random index into the pip array, and a random index into the suit array.
      ii. Hint 2: the method `Int.random(in:  n1...n2)` generates a random integer in the range `n1` to `n2` inclusive.
   (b) In `viewWillAppear()`, put the pip value and suit selected by the user into the buttons. (Note that this is also a good place to set them to the initial values ”select pip” and ”select suit”.)
   (c) In the action function for the ”Submit Guess” button, write code that generates the appropriate text in the three status message labels.
      i. Messages that should appear in the pip message label include ”Correct pip value”, ”Your pip value is too low”, and ”Your pip value is too high”.

ii. Messages that should appear in the suit message label include ”Correct suit” and ”Incorrect suit”.

iii. Messages that should appear in the guess message label include ”You guessed correctly in XXX tries!”, and ”Guesses so far: YYY”. Also, if the user clicks the ”Submit guess” button before having selected both a pip and a suit, the message ”select a pip value and suit first” should appear in the guess message label. This will not count as a guess.

3 Grading criteria

1. You have UI components and a data model as defined. (20%)
2. The Table View screens look like the ones pictured by the UX Prototyping student. (30%)
3. The main screen looks like the one pictured by the UX Prototyping student. (30%)
4. The application behaves as expected. (20%)
5. **If the app does not build and run, ZERO points will be given.**
6. The Coding Standard is followed. One point deducted for each violation.

4 General criteria

1. Set the simulator to the newest model supported by the version of Xcode you have installed.
2. I will be looking for good documentation, descriptive variable names, clean logical structure, and adherence to all coding conventions expected of an experienced programmer, as well as those outlined in the Coding Standard document. There will be penalties for failure to meet these standards.
3. Xcode will automatically generate standard headers to your .swift files. Add lines to each Swift file so that the header includes the following:

```
// Project: LastnameFirstname-HW4
// EID: xxxxxx
// Course: CS329E
```

5 Instructions specific to UX Prototyping students

You will develop seven screens for the CS non-majors in CS329E: Elements of Mobile Computing. They will use these screens to do their part, described in the first four sections above.

The seven screens you will design must be designed with the recognition that the developers have only been learning Swift for a few weeks. Therefore, you must make screens that are feasible for a beginner to implement.

Each UX Prototyping student will have four clients. These are the same students who will form the group you will support in the semester-long project, but this homework assignment is individual, so there is no need for coordination beyond ensuring that your screens are feasible and the CS students understand your designs.

Each UX Prototyping student will deliver initial screens to the CS students on 24 September, the day the CS students are given the assignment. The CS students will turn in the full package on 3 October, so there can be some back and forth about feasibility and desirability in the interim.

The seven screens are as follows:

1. Starting screen
2. After clicking ”select pip”
3. After choosing ”3”, returning, and clicking ”Submit” early
4. After clicking ”select suit”
5. After choosing ”Hearts”, returning, and clicking ”Submit”
6. After choosing ”8”, choosing ”Spades”, and clicking ”Submit”
7. After choosing ”Diamonds” and clicking ”Submit”

Note that the last three screens are similar but slightly different. The fifth and sixth screens are incorrect guesses, but different incorrect guesses. The fifth screen will say that the pip value is too low, while the sixth screen will say that the suit is incorrect. Both the fifth and sixth screens will show the number of guesses so far. The seventh screen will show a correct guess and the number of tries it took to guess correctly.

Grading criteria for the UX Prototyping students:

1. The CS students will be able to implement the screens you design using the tools they have already learned. (30%)
2. The screens are visually appealing. (30%)
3. The screens are clearly labeled. (20%)
4. The screens are consistent with the directions given to the CS students under *Detailed Instructions* above. (20%)