Homework #2: Matching

Due: Tuesday, February 3 @ 12:30 PM

Submission:

Please turn in all files on Canvas before the deadline. You should compress your submission into a single file, do not submit a large number of individual files. If you know you are going to miss a deadline, contact the TA **before** the deadline. Canvas has been known to be quirky, so it is not advised to wait until 5 minutes before it is due to make your submission.

Please include a text file called “README” at the top level of your main project directory. Include the following:

- Your name
- Your email address
- How long this project took you to complete
- Any comments or notes for the grader

Overview:

This is not a group assignment. It is acceptable to consult with other class members, **but your code must be your own**.

You will expand upon homework 1 to create a simple matching game.

Specifications:

Part 1: Model

Reuse your card related classes from homework 1.

Add a method to class Card that compares the card against another card. Return 1 if the “contents” of both cards match, else return 0.

Override this method in class PlayingCard. Use the following list to determine the match score. Return the value equal to the highest matching category.

- Different colors and different ranks: 0

- Color of both cards match: 1
- Suit of both cards match: 2
- Rank of both cards match: 8
- Rank and color of both cards match: 16
- Rank and suit of both cards match (i.e. they are the same card): 32

Part 2: View

Display 30 cards on the screen in a grid (face down). The cards should be chosen randomly from a PlayingCardDeck.

The game will have two modes, 2 card matching and 3 card matching. Include a **UISwitch** to switch back and forth between these modes of operation.

Include a button labeled “Peek”. The purpose of this button will be explained below.

Include a reset button.

Include a display somewhere on the screen to show the current score.

Include a display somewhere on the screen to show how many games have been played.

Include a display somewhere on the screen to show the player’s average score over previous games.

Include a display somewhere on the screen that shows the point value of the most recent match.

When cards are taken out of play (i.e. when they are matched) leave them face up but disable the button that represents the card. Change the button in some visual manner to indicate that it is out of play (example: make the text gray).

Part 3: Control

When the user taps on a card it should turn face up. Add -1 to the current score.

If the user taps on the same card again turn it face down. (No penalty score.)

The reset button will start the game again. The cards should be re-shuffled and turned face down. (In fact, you should have a different subset of the original playing deck.) Scores should also be reset.

When the user presses the Peek button, turn all cards face up. Add -15 to the current score. Disable interaction with the cards until the user presses the Peek button again,

at which time the cards are put back the way they originally were. If there were un-matched cards that were face up before the Peek action, make sure they are still face up after the Peek action.

If the game is in two card matching mode, when the user taps a second card (so that there are two face up cards), compute the score of that match and take those cards out of play. Add the score of the match to the player's overall score.

If the game is in three card matching mode, when the user taps a third card (so that there are three face up cards), compute the score of that match and take those cards out of play. Add the score of the match to the player's overall score.

3-card matching rules

Matching in three card mode should function as follows:

Compare each of the three cards against each other card. Match these pairs individually. The final score is the sum of the matches between all three cards.

In other words, suppose we are attempting to match three cards: C1, C2, and C3.

Compute the matching score between C1 and C2.
Compute the matching score between C2 and C3.
Compute the matching score between C3 and C1.

The resulting score is the sum of these three scores.

Notes:

You may use any code presented in class. Please type the code yourself (as opposed to copy-paste) as it is a good learning experience.

To make 30 cards fit, you'll have to make them smaller (maybe 40x60 or so). The cards will strongly want to be their natural size (based on the background image sizes) while working with them in Xcode, so you'll have to hold down the Command key while resizing them.