Mini-project 5 - “Memory”

申恒恒

February 27, 2016

Mini-project description - Memory

Memory is a card game in which the player deals out a set of cards face down. In Memory, a turn (or a move) consists of the player flipping over two cards. If they match, the player leaves them face up. If they don't match, the player flips the cards back face down. The goal of Memory is to end up with all of the cards flipped face up in the minimum number of turns. For this project, we will keep our model for Memory fairly simple. A Memory deck consists of eight pairs of matching cards.

Mini-project development process

As usual, we suggest that you start from the program template for this mini-project.

1. Model the deck of cards used in Memory as a list consisting of 16 numbers with each number lying in the range [0,8) and appearing twice. We suggest that you create this list by concatenating two list with range [0,8) together. Use the Docs to locate the list concatenation operator.

2. Write a draw handler that iterates through the Memory deck using a for loop and uses draw_text to draw the number associated with each card on the canvas. The result should be a horizontal sequence of evenly-spaced numbers drawn on the canvas.

3. Shuffle the deck using random.shuffle(). Remember to debug your canvas drawing code before shuffling to make debugging easier.

4. Next, modify the draw handler to either draw a blank green rectangle or the card's value. To implement this behavior, we suggest that you create a second list called exposed. In the exposed list, the ith entry should be True if the ith card is face up and its value is visible or False if the ith card is face down and it's value is hidden. We suggest that you initialize exposed to some known values while testing your drawing code with this modification.

5. Now, add functionality to determine which card you have clicked on with your mouse. Add an event handler for mouse clicks that takes the position of the mouse click and prints the index of the card that you have clicked on to the console. To make determining which card you have clicked on

easy, we suggest sizing the canvas so that the sequence of cards entirely fills the canvas.

6. Modify the event handler for mouse clicks to flip cards based on the location of the mouse click. If the player clicked on the ith card, you can change the value of exposed[i] from False to True. **If the card is already exposed, you should ignore the mouseclick.** At this point, the basic infrastructure for Memory is done.

7. You now need to add game logic to the mouse click handler for selecting two cards and determining if they match. We suggest following the game logic in the example code discussed in the Memory video. State 0 corresponds to the start of the game. In state 0, if you click on a card, that card is exposed, and you switch to state 1. State 1 corresponds to a single exposed unpaired card. In state 1, if you click on an unexposed card, that card is exposed and you switch to state 2. State 2 corresponds to the end of a turn. In state 2, if you click on an unexposed card, that card is exposed and you switch to state 1.

8. Note that in state 2, you also have to determine if the previous two cards are paired or unpaired. If they are unpaired, you have to flip them back over so that they are hidden before moving to state 1. We suggest that you use two global variables to store the index of each of the two cards that were clicked in the previous turn.

9. Add a counter that keeps track of the number of turns and uses set_text to update this counter as a label in the control panel. (BTW, Joe's record is 12 turns.) This counter should be incremented after either the first or second card is flipped during a turn.

10. Finally, implement the new_game() function (if you have not already) so that the "Reset" button reshuffles the cards, resets the turn counter and restarts the game. All cards should start the game hidden.

11. (Optional) You may replace the draw_text for each card by a draw_image that uses one of eight different images.

Once the run button is clicked in CodeSkulptor, the game should start. You should not have to hit the "Reset" button to start. Once the game is over, you should hit the "Reset" button to restart the game. While this project may seem daunting at first glance, our full implementation took well under 100 lines with comments and spaces. If you feel a little bit intimidated, focus on developing your project to step six. Our experience is that, at this point, you will begin to see your game come together and the going will get much easier.
For more helpful tips on implementing this mini-project, please visit the Clinic tips page for this mini-project.

Grading Rubric - 11 pts total (scaled to 100)

- 1 pt - The game correctly draws 16 cards on the canvas (horizontally or as a grid). Using images in place of textual numbers is fine. **However,**

it is the submitter's responsibility to ensure that custom images load during peer assessment.

- 1 pt - The cards appear in eight unique pairs.
- 1 pt - The game ignores clicks on exposed cards.
- 1 pt - At the start of the game, a click on a card exposes the card that was clicked on.
- 1 pt - If one unpaired card is exposed, a click on a second unexposed card exposes the card that was clicked on.
- 1 pt - If two unpaired cards are exposed, a click on an unexposed card exposes the card that was clicked on and flips the two unpaired cards over.
- 1 pt - If all exposed cards are paired, a click on an unexposed card exposes the card that was clicked on and does not flip any other cards.
- 1 pt - Cards paired by two clicks in the same turn remain exposed until the start of the next game.
- 1 pt - The game correctly updates and displays the number of turns in the current game in a label displayed in the control area. The counter may be incremented after either the first or second card is flipped during a turn.
- 1 pt - The game includes a "Reset" button that resets the turn counter and restarts the game.
- 1 pt - The deck is also randomly shuffled each time the "Reset" button is pressed, so that the cards are in a different order each game.

References

[1] IIPP, Joe Warren, Scott Rixner, John Greiner, Stephen Wong, `https://class.coursera.org/interactivepython2-010`

[2] `http://www.mohu.org/info/lshort-cn.pdf`