Memory Matching Challenge

By Natalie Huffman
Under the direction of Susan Rodger
Duke University
June 2017

Game

• This game is a memory matching game
• The user can click on two cards at a time, making them turn over
• If they match, they will vanish
• If not, they both turn back over
• Try to match up all the cards in as few moves as possible!
To Add

- We have three methods we want to edit:
  - myFirstMethod
    - Move an element from the list `cards` up 2.4 meters
    - Set an element from the list `nums` to -1
  - Restart
    - Move all the cards to this.billboard
    - Reset the property `gone` to 0
  - initializeEventListeners
    - Turn a card .5 revolutions right when it is clicked on
    - Update number of tries, change the text to reflect it
    - Run the restart procedure when restart button is clicked

- The comments in the code will tell you where your additional code should go
- Make sure that anything you add goes directly under the corresponding comment
- The code has to be in the right order to work
Go to myFirstMethod

• We have generated a random number, which is called \textit{random}
• We want to move the random-th billboard in the list \textit{cards} UP 2.4 meters
• It should look like the \texttt{UP} and \texttt{LEFT} move statements in the following do in order block

• Then we want to set the random-th number in the list \textit{nums} to -1 so we never move the same card twice (use the assign block)
• This should look like the assignment statement in the following do in order block
Go to restart

- Look for the do together block with the comment
- We want to use the local variable `card2` to move all of the items in the list `cards` to this.billboard
- The each in __ together block uses `card2` to represent one item in the list `cards`
- If you perform an action on `card2`, Alice will perform that action on all the items in `cards`

- We need to reset all of our variables so we can restart the game
- The variable `gone` holds the number of successful matched cards, so we can know when the game is over
- We want to reset `gone` to 0, so the score is reset to 0
  - `Gone` is a property of the scene
Go to initializeEventListeners

• First, some definitions:
• **cardsTurned** is the number of cards that have been turned over
• **Board1** is a property that stores the value of the first card clicked, so that when the second card is clicked we know what the first one was
• **win** is a boolean property that holds the value of whether or not the user has won (since we don’t want them to turn over cards if they’ve already won)
• **getModelAtMouseLocation** stores the value of whatever was just clicked on

Go to initializeEventListeners

• First, we will edit the first `addMouseClickOnObjectListener` block, which is for the cards
• We want to put a statement where the comment is to turn the card the user clicks RIGHT .5 revolutions
  – Remember—what variable holds the value of whatever was just clicked on?
• At the end of the “else” block, we want to add 2 statements
• One will update tries by one
  – Tries is a property of score, the TextModel seen above the white rabbit
• The other will set the value of score to tries

Second
addMouseClickOnObjectListener

• Now we will edit the second addMouseClickOnObjectListener, which is for the restart button
• When the user clicks on the restart button, we want the restart procedure to run
  – Restart is a scene procedure
Run the game!

• As you click cards, they should turn over
• If they match, they will vanish
• Try to win in as few moves as possible!