CISC 110 Assign 5: Die-Rolling Simulation

Assignment 5 Marking Scheme (2% of final mark)

Marked out of 10:

1 mark: All input values are read from text fields and result is displayed in a text field

4 marks: Weights added to allow “unfair advantage” when rolling die

3 marks: Die-rolling animation controlled by Timer

1 mark: Random roll value displayed in die-rolling animation on each roll

1 mark: At least one function is defined that is called from within another function

In this assignment, you will create an app that simulates throwing a six-sided die (only one die, rather than a pair of dice) many times in order to do a statistical analysis of how random Math.random actually is.

Here is a summary of the portion completed during the lab:

The user chooses the number of trials and specifies a particular die value (1, 2, 3, 4, 5, or 6). During each trial, the app simulates rolling the die until the user’s specified die value is rolled. After completing all of the trials, it calculates the average number of rolls it took to roll the user’s die value over all of the trials. Since there are six numbers on the die, if there is an even probability of rolling any one of them, it should take an average of six rolls to roll a desired value.

Here is a summary of what you need to add for the assignment portion:

- Allow the user to specify weights for each side of the die, so that the probability of it landing on one side could be greater than another - a weighted die to allow an unfair advantage.
- Include an animation of a die rolling and stopping on a random side that is controlled by a Timer.
- Organize your code into steps, with at least one step implemented as a function that you call (use) within another function, for instance you might call the function within your button-handler function.

Below are the main steps for your assignment.
1. Add six input text fields to allow the user to input six weights, one for each side of the die, which add up to 1. For instance, if the user wanted to have approximately even probability that any of the six sides would be the result of a roll, the six weights could be: .16, .16, .17, .17, .17, and .17. Read those values into variables. Since you may need to refer to those variables within more than one function, create global variables, defined inside your class but prior to function definitions. Alternatively, you could create custom properties for your die. HINT: Set default values for the weights in your input text fields, so that you don’t have to type them in every time you run it!

2. Modify how you assign the die’s roll value according to the weights specified by the user. For instance, if the weights are the example values given in step 2, you could assign a roll value of 1 for a result from Math.random() between 0 and .16, a roll value of 2 for a result from Math.random between .16 and .32, etc. Your code must work no matter what weights the user enters, as long as they add up to 1. You may assume that the user will only enter valid input values.

3. Add an animation of a die rolling several times to simulate the rolls of the dice that are being calculated. It should roll and display a random side each time and stay there for a second before it rolls again. Use a Timer to achieve this. For how to use a Timer, look at the Week 6 Timer Events and Timer Events Plus examples on the course website.

At the start of your Timer event handler function, set the value of a text field in your die randomly with Math.random. Then play your die-rolling animation.

Put the variable declaration for the Timer inside your class and just before the start of your constructor function, so that it’s a global variable that you can refer to anywhere in your class. Add the listener for the Timer inside your constructor function. Start your Timer inside the button handler, so the dice-rolling animation only starts playing when the button is pressed. Right before you start the Timer, you could also reset it, so that your Timer will start over if the user presses the button a second time. If your Timer variable is called timer, the command to reset it is: timer.reset();

Remember that you can use a second parameter when you create a new Timer, which specifies the number of times the Timer will go off before stopping, instead of running indefinitely. See the lecture notes for an example. For instance, you could play your die-rolling animation three times, once every two seconds, by specifying 2000 as the number of milliseconds for your Timer and specifying 3 as the second parameter. If you design your dice-rolling animation to be two seconds long, then it will finish at the end of each Timer time period.

You can have a simple die-rolling animation, as long as the value of the die is set randomly with Math.random and as long as a Timer controls it. For instance, you could have the die be a simple square with a text field that shows the same random value for its entire roll. Then the next roll it would show another random value for its entire roll!
4. (Optional) Once your die-rolling simulation is working the way you’d like, publish it and upload it to the CISC 110 web space.