CS 106 Winter 2018

Lab 08: Randomness and Noise

Due: Wednesday, March 13th, 11:59pm

SAVE each sketch as “L08Q01”, “L08Q02, and L08Q03.

QUESTION ONE: Coin Flipping

In the demo code there is an example coinFlip_newGame.pde. It flips a coin 8 times and shows the results of either Heads or Tails. We went over this code in class. It uses a function playGame() which is in a separate tab in Processing. You are to write a variation of coinFlips_newGame to flip a coin 100,000,000 times and show the resulting count of the number of Heads and the Number of Tails, as well as an image of the “winner”.

Use coinFlips_newGame as the starter code. You need to rewrite the function playGame().

Your result must look similar to the images shown below.

Note: To format a number to contain commas we can use the nfc() function.

For example,

```
text(nfc(headsCount), 10, 30);
```

prints the variable headsCount to the sketch window with commas as appropriate.

QUESTION Two: Coin Flipping with Bias

Modify your code in Question One above in two ways.

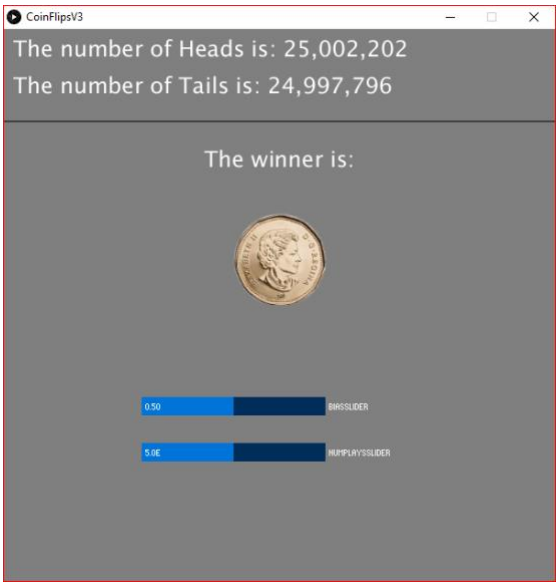
You are to include a ControlP5 slider to control the bias for the probability of each outcome being a Head. The slide goes from 20% to 80%. If the slider is at the left then then 20% of the outcomes, on average, are Heads and 80%, on average, are Tails.

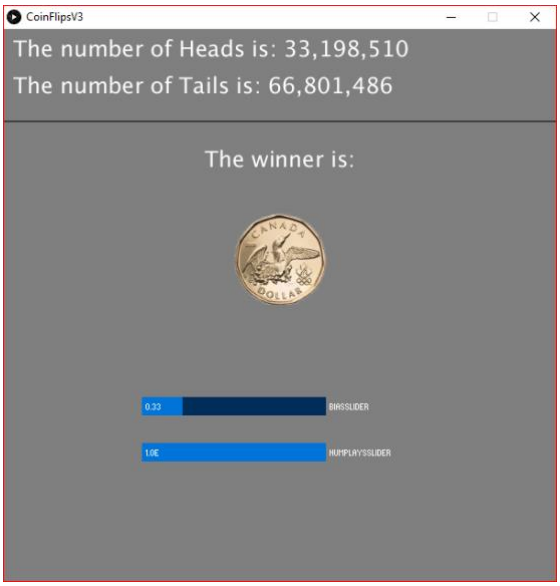
You are to include another ControlP5 slider to control the number of plays which can vary from 10 to 100,000,000. If the slider is at the left then each play is 10 flips of the coin. If the slider is at the right then each play is 100,000,000 flips of the coin.

You may need to remind yourself about ControlP5 by looking at the lecture 05 slides and by looking at the slider in busyBox.pde demo code from the week we did ControlP5.

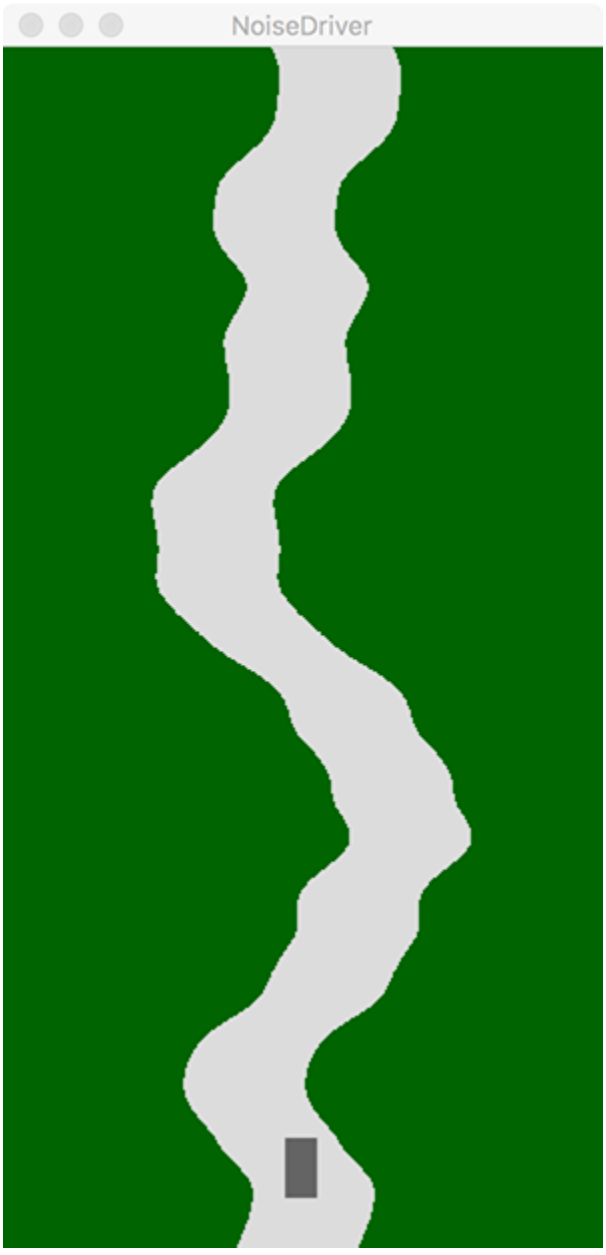
Screen shots are shown below.

A video is at: https://youtu.be/szw_jO0RzzQ





QUESTION THREE: Driving the noise function

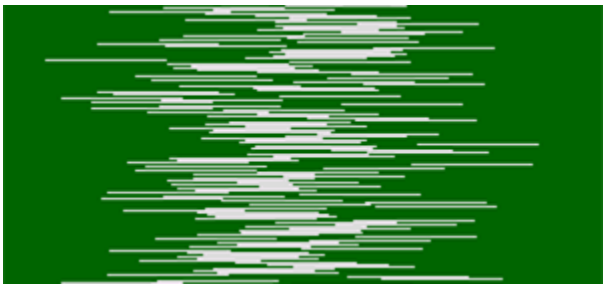


Create a simple driving game in which the road is procedurally generated using (the one-dimensional version of) the `noise()` function. Follow these steps:

1. Create a sketch of size 300×600. In the `draw()` function, give it a green background.
2. Now draw a horizontal one-pixel-wide light grey line in every row of pixels in the sketch. Make the line 60 pixels long. Use the one-dimensional `noise()` function to choose the *X* position of the midpoint of the line, giving it the *Y* coordinate of the line as input. That way, the shape of the road is

fully determined by noise. The `noise()` function produces values between 0 and 1; map that to the largest possible range of *X*positions in the sketch window for which no part of the road moves off-screen.

3. If you do this, you'll probably see a "road" that looks something like this:



The problem is that you're taking values of the `noise()` function that are spaced too far apart, and that are therefore changing too fast. To compensate, define a global floating-point constant that acts as a scaling factor for the *Y* values you pass into `noise()`. Experiment with values of that variable until you get a road that's the right "twistiness" for a driving game.

4. Now let's make the road scroll down the screen. Define a global variable that controls the vertical translation of the game. Before passing *Y* values into the `noise()` function, add the current value of this global variable to them. Then, at the end of every frame, add a small amount to the translation amount. If you do this right, the road will scroll downward forever, with new segments of road always coming in from the top of the screen. Adjust the per-frame translation amount so that the road sweeps by at a reasonable rate for a driving game.
5. Let's add a car to the sketch. Draw a small vertically oriented rectangle with its centre 40 pixels from the bottom of the sketch window. When the sketch starts, set the *X* position of the car so that it sits in the centre of the road (use a global variable to remember the car's *X* position). The car will always remain at this *Y* position in the sketch, and the road will scroll past it.
6. Add the ability to control the *X* position of the car via the left and right arrow keys. We want the car to respond every frame a key is pressed without worrying about key repeat, and so the `keyPressed()` hook function isn't the best way to go. Instead, we'll use a style of key event handling you saw in CS 105 but haven't used this term. Add code like this to the `draw()` function:

```
  if (keyCode == LEFT) {
    // Change car X location here.
  } else if (keyCode == RIGHT) {
    // Change car X location here.
  }
}
```

7. Finally, add code to detect whether the car has gone off the road. We'll say that the car is on the road as long as the *X* coordinate of its centre lies somewhere within the line segment of the road at its *Y* position. Every frame, check whether the car has gone off the road. If it has, stop the road from scrolling. When the road stops, you know the game is over.

The entire sketch can be written with fewer than 50 lines of code, not counting comments.

Submission

Submit all sketch directories from this lab as one ZIP file called L08.zip to the lab dropbox on Learn.

It is your responsibility to submit to the correct dropbox with the correct files before the deadline. Otherwise you will receive a mark of 0.