CS106 W20 - Lab 06

Geometric Context

Due: Wednesday, February 12, 11:59 PM

Question 1: Singles

In your L06 folder, create a new sketch called Singles. In that sketch, write the following functions.

1. Write a function reverseArray() that takes an array of integers as input and returns an array of integers of the same length as the input, in which the elements of the original array are given in reverse order. For example, if you passed in [1, 2, 5, 6, 3] as input, the function would return [3, 6, 5, 2, 1]. The original array must not be modified.
2. Write a function partialSums() that takes an array of integers as input and modifies it so that each element is the sum of all the original numbers up to that point. That is, the code …

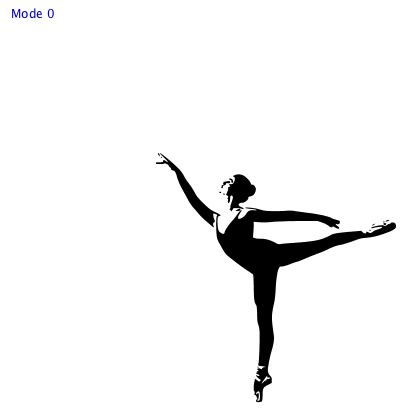
   ```
   let arr = [1, 2, 5, 6, 3];
   partialSums(arr);
   print(arr);
   ```

   … will print 1, 3, 8, 14, 17 to the console (because 3=1+2, 8=1+2+5, and so on).

Question 2: Dancers

In a series of exercises, you will use combinations of geometric context functions to draw different configurations of dancers. The goal is to do everything with geometric context functions. Every dancer will be drawn using image(dancer, 0, 0), surrounded by suitable calls to push(), pop(), translate(), rotate(), and scale().

1. In the provided starter code, run the Dancers sketch. You'll see a single dancer in the centre of the sketch window:



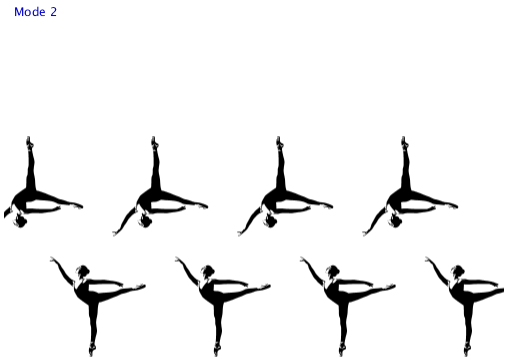
Press the space bar to cycle through a number of drawing "modes". You'll see that only Mode 0 is implemented, and the others display blank screens. You will write code for the other modes.

2. Fill in the function drawMode1() so that it draws a 5×5 grid of dancers, as in the image below. All transformations must be based on the built-in functions translate(), rotate(), scale(), push (), and pop (). To draw each dancer, you *must* use image(dancer, 0, 0)—*no other arguments to image() are permitted here, or in any of the other problems below!*

   In this case, the most natural approach is to use two nested loops, one in *x* and one in *y* (in either order). Inside the nested loops, translate, scale and draw a single dancer.

3. Fill in the function drawMode2() so that it draws alternating upright and upside-down dancers, with their arms all pointing to the same side of the screen. Again, make sure that

every dancer is drawn using image(dancer, 0, 0). Remember that you can use the call scale(1, -1) (inside a push)/ pop() pair) to flip a dancer upside-down.



4. Fill in the function drawMode3() so that it draws a single large dancer near the middle of the sketch, with at least three smaller dancers perched on her. The smaller dancers must have their lower legs touching the large dancer, and must fit entirely in the sketch window. The image below gives one example.

Unlike the other exercises in this lab, there's no "abstract" way to draw this configuration, for instance based on loops. The best solution is simply to "eyeball" it: draw the large dancer, and experiment with the scaling and translation of the smaller dancers until you're happy with the result.

5. Fill in the function drawMode4() so that it draws two scaled copies of the configuration of dancers from drawMode3(), where the large dancers are facing each other:

Mode 4

Unlike the other modes, for this question *you should not call the* image() *function directly*. Instead, using the geometric context functions to set up appropriate translations and scales, and then call drawMode3() to draw precisely the same dancers you created in the previous step. The surrounding context must transform those two sets of dancers so that you get a picture like this one.

Tips

- You don't have to create exact copies of the screenshots shown here. The important thing is to create similar patterns. You may need to experiment with different scales and translations to get something similar.
- You should avoid writing functions that simply draw each dancer explicitly (e.g., 25 calls to image() for Mode 1). When possible, use for loops to draw the dancers you need.

Save your sketch, titled Dancers, in your L06 folder.

Submission

When you are ready to submit, please follow these steps.

1. Please ensure that any sketches you submit compile and run. It's better to submit a sketch that runs smoothly but implements fewer required features than one that has broken code for all features. If you get partway into a feature but can't make it work, comment it out so that the sketch works correctly without it.
2. If necessary, review the Code Style Guide and use Processing's built-in auto format tool. You do not need to use the precise coding style outlined in the guide, but whatever style you use, your code must be clear, concise, consistent, and commented.
3. If necessary, review the How To Submit document for a reminder on how to submit to LEARN.

4. Make sure to include a comment at the top of all source files containing your name and student ID number.
5. Create a zip file called L06.zip containing the entire L06 with its subfolders Singles and Dancers.
6. Upload L06.zip to LEARN. Remember that you can (and should!) submit as many times as you like. That way, if there's a catastrophe, you and the course staff will still have access to a recent version of your code.
7. If LEARN isn't working, and ***only*** if LEARN isn't working, please email your ZIP file to the course account (see the course home page for the address). In this case, you *must mail your ZIP file before the deadline.* Please use this only for emergencies, not "just in case". Submissions received after the deadline may receive feedback, but their marks will not count.