CS106 W20 - Lab 07

Animating Geometric Context

Due: Wednesday, February 26, 11:59 PM

Question 1: Singles

In your L07 folder, create a new sketch called Singles. In that sketch, write the following functions.

1. Write a function splitter() that takes a string as a parameter and converts the string into an array by splitting the string using tokens ‘ ’ and ‘,’ (space and comma). The function then simply returns the array. For example if you pass “Hey, how are you?”, the function returns: ["Hey", "how", "are", "you?"]

2. Write a function arrayAddMultiply() that takes an array of integers as a parameter and finds the sum and product of its elements. The function also prints the sum and product on the console. For example, if you pass [1, 2, 3, 4], the function prints 10 and 24.

You are encouraged, but not required, to test your functions by writing a setup() function that calls them and checks whether the correct answers are returned.

Save your work in the sketch Singles.

Question 2: Dancers, Revisited

> *Make sure you are using the Lab07 starter code. Even though Lab06 starts out similarly, the Lab06 starter disables animation, which you need working in Lab07.*

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.

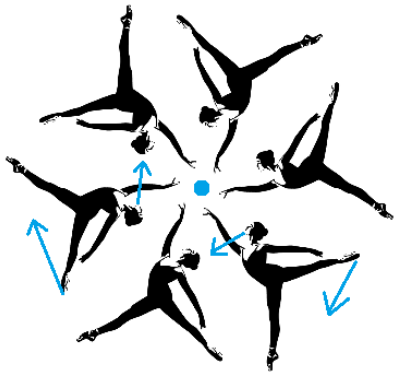
2. Fill in the function drawMode1() so that six dancers get arranged in a circle, rotated symmetrically.

There are lots of different patterns that are correct solutions to this question; you don't have to copy the image here. What's important is that you have a ring of six dancers twirling around the centre of the screen.

Hint: experiment with image(dancer, 0, something) within a suitable rotation context. Then, to comply with the main restriction (and to set you up for success in part 4), you can switch that to:

```
push();
translate(0, something);
image(dancer, 0, 0);
pop();
```

3. Fill in the function drawMode2() to animate the twirling. Make your dancers proceed around the centre of the screen, each one following the one in front. Recall that our key to these animations is transforming by the built-in variable frameCount (times a constant that you pick to look good.)



Hint: this is the much easier animation enhancement. The centre of rotation that you are already using in drawMode1 is the only one you need. You just have to make its amount depend on frameCount.

4. (Challenge, optional.) Fill in the function drawMode3() to do the twirling differently. Here, have your dancers' centres remain fixed, and make each dancer spin around her centre.

Hint: You need six centres of rotation. Remember that rotations always happen around (0, 0). Think about the relationship between a given dancer and the point (0, 0). Use the helper code below to show that on-screen. Run the helper code in with the earlier draw modes. Change the (0, 0) point so that it's in the centre of the dancer, by adjusting imageMode. (Make the same change to rectMode so that the helper shows the change accurately.) You'll then have to modify some translate amounts to put the dancer back in the right place.

```
image( dancer, 0, 0 );
noFill();
rect(0, 0, dancer.width, dancer.height);
fill(0, 0, 255);
```

```
    ellipse(0, 0, dancer.width/20, dancer.height/20);
```

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., 6 calls to image() for Mode 1). When possible, use for loops to draw the dancers you need.

Save your sketch, titled Dancers, in your L07 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 L07.zip containing the entire L07 with its subfolders Singles and Dancers.
6. Upload L07.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.