JavaFX Animation

November 25, 2014

1 Introduction

This assignment will cover some basic Animation techniques in JavaFX! Please read through the entire assignment description before beginning your assignment. I also particularly suggest reading the Oracle Tutorials for some of the concepts presented in this homework. They are extremely helpful.

2 Problem Description

Your task is to create a to-scale version of the solar system using the planets described below. You need to animate the planets in a way that makes them revolve in a circular path around the sun. The planets and their requirements/dimensions are included below so you don't actually need to do any research to complete the assignment.

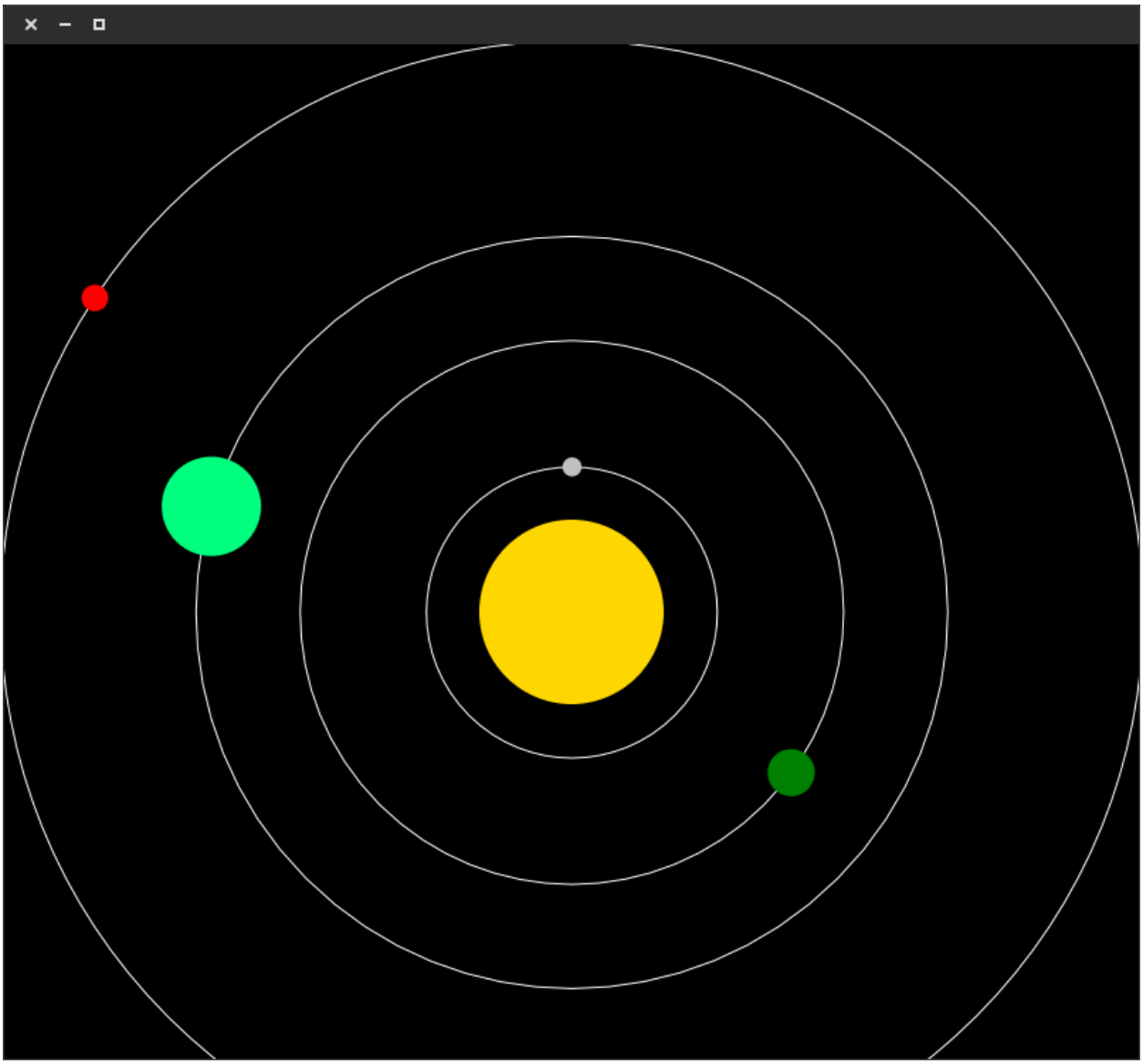
2.1 Provided Files

We have provided the following files for you. You must edit these files as outlined below. Do not change their names. You are free to add additional files to augment your application.

- `Planet.java`
- `Planetarium.java`

2.2 Sample Output

Here's an example of what your application might look like (with the planets actually moving, of course!)



2.3 `Planet` Requirements and Guidelines

Your `Planets` should meet the following requirements:

1. The sun should be at the center of the display
    - The sun shouldn't be a planet, but it needs to be a circle.
    - The sun's radius should be 65.
    - You will not be animating the sun. The sun is a solid rock for the planets to cling to in the storms of the universe. It doesn't move.
2. Planets should be an enumeration

- You have been provided with a skeletal Planet.java file for you to fill in.
- Each planet has a unique `Color`. You can be creative and choose what colors you make them.
- Each planet has a

  `distance` - how far from the sun the planet is

  `radius` - how big the planet is

  `period` - how long it takes the planet to revolve around the sun

  These properties should be final and of type `double`.
- Each planet also has a `Path` that it follows when orbiting around the sun. The `Path` should be white.
- Each Planet should be represented as a `Circle`
- Planets should have a way to get their `Circle` and `Path`

3. Planets should revolve around the sun in a circular orbit. This means you need to animate!
   - There are some helpful tools you can use to animate your planets One of them being `ArcTo`. `ArcTo` forms an arc from the previous coordinates to specified x and y coordinates using a provided radius.
   - `Transitions` are one way to animate objects in Java. In order to get the planets to actually move, you need to create a `PathTransition` for the Planet. Again, the API is your friend.

2.4 Planet Information

The planets' information is as follows:

1. The Earth's information is provided for you in Planets.java
2. Mercury
   - `Color`: Silver
   - Period: `Earth`'s period * 0.24
   - Radius: `Earth`'s radius * 0.1915
   - Distance from sun: `Earth`'s distance * 0.387
3. Venus
   - `Color`: Green
   - Period: `Earth`'s period * 0.62
   - Radius: `Earth`'s radius * 0.4745
   - Distance from sun: `Earth`'s distance * 0.723

4. Mars
    - `Color`: Red
    - Period: `Earth`'s period * 1.88
    - Radius: `Earth`'s radius * 0.266
    - Distance from sun: `Earth`'s distance * 1.52

2.5 The Application: `Planetarium`

- The height and width of the `Scene` should be declared final. Try to get it to be somewhere in the ballpark of 700-800. Depending on how big your monitor is, you may have to play with this number to get it to show up on your screen correctly. If your monitor or display can't handle 700-800 it is ok to change it to something reasonable (everything should be able to handle 600). If you make your application size smaller, and you are having trouble fitting the solar system in the window, you are free to change the values of earth's provided information to get it to fit.
- The background color should be black. This is the universe after all! Devoid of all light and hope! Ok that's a bit of an exaggeration, but... it should be black.
- When you run your application you should see a really cool output that automatically starts playing. There should be no glitches and it should run at a decent speed (fast enough to actually see animation... no tortoise universes).
- All of the paths should be completely closed. No gaps. Planets should go around in an actual orbit at constant speed. If your planets go around, stop, and then reverse direction go figure out how to make it not do that!

3 Hints and Tips

This assignment might seem "free form" but there are a lot of resources to help you animate this! Some such things that might be paeticularly useful are:

- `Path`
- `PathTransition`
- The Oracle tutorials are super helpful as well. Particularly for javafx since there hasn't been a lot of time for other sources to be created. Take advantage of this resource.

If you aren't super comfortable with enumerations, there is a really helpful tutorial provided by Oracle that explains the idea really well.

Do not procrastinate on this homework. There aren't a lot of requirements, but time needs to be spent messing around with things to fully understand how to animate this!

3.1 Sample Output

4 Javadocs

We are going to have you do Javadocs for this assignment (and for all assignments here on out). Javadocs are a clean and useful way to document your code's functionality. For more information on what Javadocs are and why they are awesome, the online documentation for them is very detailed and helpful. The relevant tags that you need to have are `@author`, `@version`, `@param`, and `@return`. Here is an example of a properly Javadoc'd class:

```
import java.util.Scanner;

/**
 * This class represents a Dog object.
 * @author George P. Burdell
 * @version 13.31
 */
public class Dog {

   /**
    * Creates an awesome dog (NOT a dawg!)
    */
   public Dog(){
      ...
   }

   /**
    * This method takes in two ints and returns their sum
    * @param a first number
    * @param b second number
    * @return sum of a and b
    */
   public int add(int a, int b){
      ...
   }
}
```

5 Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **100** points. Review the Style Guide and download the Checkstyle jar and associated XML file. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by piping the output of Checkstyle through `wc -l` and subtracting 2 for the two non-error lines printed above (which is how we will deduct points). For example:

```
$ java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java | wc -l
      2
```

Alternatively, if you are on Windows, you can use the following instead:

```
C:\> java -jar checkstyle-6.0-all.jar -c cs1331-checkstyle.xml *.java | findstr /v "Starting
    audit..." | findstr /v "Audit done" | find /c /v "hashcode()"
0
```

> Food for thought: is there a one-liner like above that shows you only the number of errors? Hint: `man grep`.

The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **100** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

6 Turn-in Procedure

Submit all of the Java source files you modified and resources your program requires to run to T-Square. Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, or the `cs1331-checkstyle.xml` file. When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code with Checkstyle!

Verify the Success of Your Submission to T-Square
Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.

(a) It helps insure that you turn in the correct files.

(b) It helps you realize if you omit a file or files. [1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

(c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight. Do not wait until the last minute!