Advanced Programming
Homework Assignment 4

Due Wednesday, February 13, at 3PM

Preminimaries

1. Read chapters 4, 6, 8, 10, and 13 of SOE. (It's fine to skim 4, 6, 8, and 10, but read 13 carefully: it's more challenging. The details of the kaleidoscope example at the end of 13 are not important, but you may want to try to figure it out as a test of comprehension.)

2. Find a partner, if you don't already have one. The rest of the assignment should be done "shoulder to shoulder," with both of you working at the same screen at the same time.

Warm-up

1. The code in the file `hw4template.hs` on the course web site constructs a very simple planetary system containing just a single object called `sun`. Put this code in a file `Main.hs` and make sure you can execute it.

2. Use the provided function `translateB` to write a function

```
orbit :: Behavior Picture     -- the satellite
      -> Behavior Picture     -- the fixed body
      -> Float                -- the frequency of the orbit
      -> Float                -- the x-radius of the orbit
      -> Float                -- the y-radius of the orbit
      -> Behavior Picture
```

   that takes two picture behaviors and makes the first orbit around the second at the specified distance and with the specified radii. That is, the two pictures will be overlayed (using `over`) and, at each time t, the position of the satellite will be translated by `xradius` $\times\ cos(t \times$ `frequency`$)$ in the x dimension and by `yradius` $\times\ sin(t \times$ `frequency`$)$ in the y dimension.

   Test your function by creating another circle, `mercury`, colored red and with radius 0.1, and making it orbit around the `sun` with a frequency of 2.0, and with radii of 2.0 and 0.2 in the x and y axes, respectively.

3. Use your `orbit` function again to create an earth with a moon orbiting around it. The earth should be a blue circle of radius 0.2. The moon should be a white circle of radius 0.08. The orbit of the moon around the earth should be characterized by a frequency of 2.5 and radii of 0.5 and 0.15. Use `orbit` once again to make this whole system orbit around the sun (in addition to `mercury`).

4. A problem you might have noticed is the overlay behavior of planets. For this part modify orbit to put planets over or under each other. Hint: you might find the lifted conditional *cond* from SOE useful for this part.

5. Modify your functions (and write any support functions that you find necessary) to make the orbital distances and planet sizes shrink and grow by some factor (you can pass this factor as parameter to the `orbit` function), according to how far the planets are from the observer. For example, the earth and moon should look a little smaller when they are going behind the sun, and the orbital distance of the moon from the earth should be less.

   Choose the scaling factor so that the solar system simulation looks good to you.

6. **Optional:** Add some other planets, perhaps with their own moons. If you like, feel free to adjust the parameters we gave above to suit your own aesthetic or astronomical tastes. Make sure, though, that the features requested in parts (c) and (d) — growing, shrinking, occlusion, etc. — remain clearly visible.

Main assignment

Design and implement your own Haskell animation.

Submission instructions

- The warm-up exercise does not need to be handed in.
- Put all of your code in a file `Main.hs`. We will build this file in the `SOE/src` directory, so you can assume that the `SOE` library is available.

  (I'm sure there are going to be teams that want to modify the SOE code. In this case, please submit a single tar file containing both your `Main.hs` and all of the `SOE/src` files—both modified and unmodified ones.)
- Put both of your names in a comment at the top of `Main.hs`.

  Also, please put the approximate number of hours that you spent on this assignment. Give separate numbers for time spent reading (individually) and time spent programming (together).
- Email the file to both `jschorr@seas.upenn.edu` and `bcpierce@cis.upenn.edu`.