Programming Assignment 2

Henry Z. Lo

Due: Tuesday, June 17 at midnight

1 Goals

This assignment will help you with the following:

1. Review priority queues.
2. Review event-driven simulations.
3. Work with inner classes.

2 Overview

You will simulate an infinite game of pong. The simulation involves one ball and two paddles, one on the left and one of the right. The ball bounces off the horizontal walls, and off the paddles on either side.

The simulation is already written in the time-driven style. You will need to convert it to the event-driven style. To do so, you need to do add the following event classes, which will be inner classes in `Simulation`:

1. `WallBounceEvent`, to simulate bouncing off a wall.
2. `PaddleBounceEvent`, to simulate bouncing off a paddle.

These should subclass `Event`; see `RedrawEvent` as an example. Every `Event` must do the following:

- Take a time t as an argument. t must be an absolute time, not relative; e.g. 100th second of the simulation, not 100 seconds from now.
- Have a `process` method, which adds future event(s) to the priority queue `pq`, and updates the simulation as needed.

All other changes should happen in the `simulate()` function. The goal is to remove all the checks, redraws, etc. with events and `Event.process()` calls.

The resulting priority-queue based simulation should behave exactly as the original time-based simulation. Moving the elements should still happen at every event occurrence, and there should be `RedrawEvent`s scheduled to animate the screen.

3 Instructions

1. Initialize the priority queue and add a `RedrawEvent` to it.
2. Replace the `while(true)` loop with one that continuously polls from the priority queue.
3. Replace the drawing code with a call to `e.process()`.
4. Change the time t to be the event time.
5. If you did this right, the simulation should still animate.
6. The `move()` calls should stay in the loop, but items should move depending on how much time has elapsed. Everything else should be replaced with calls to `e.process()`
7. Now create a `WallBounceEvent` class. Its `process` method should do the following:
   (a) Bounce the ball off the wall.
   (b) Make the correct paddle follow the ball.
   (c) Calculate when the next `WallBounceEvent` will be, and schedule it.
8. Create a `PaddleBounceEvent` class and do the same for it. You may want to `stop()` the paddle that the ball just bounced from.
9. Classes must compile.
10. Put all deliverables in your cs310/pa2 folder. Deliverables include:
    - Paddle.java (do not modify)
    - Ball.java (do not modify)
    - StdDraw.java (do not modify)
    - Simulation.java
    - memo.txt
11. Answer the questions in section 4, put answers in memo.txt.

4 Questions

1. In this assignment, we don't need to worry about events becoming invalidated. If we had two balls, we would. How would this complicate your code?
2. Because each event only spawns off one other event, we don't have to worry about the priority queue growing infinitely. If we did, how could you prevent the priority queue from eating up all your memory?
3. Why was Event an inner class of Simulation? Can you think of a better way to organize this code?