Program #4 - Rect210

Prof Bill - May 2018

Program #4 logistics:

- Due: **Thu May 31, 2018** at the beginning of class (2 weeks)
- Worth: **8 points** (8% of your grade)
- Learn: Priority Queue, Heap, BST, algorithm design, OOP, JavaFX GUI, program args in Java/Netbeans

1. Description

P4 plays with rectangles. We'll develop three programs:

- ➢ **P4Generate** - generate files of random rectangle data (for us to use later)
- ➢ **P4Intersect** - runs two algorithms to find intersecting rectangles
- ➢ **Program4** - a GUI that graphically finds intersecting rectangles, step by step

Rect210 is an interface on the k: drive that gets it all started.
Last program means… **Finish strong!**
thanks… yow, bill

Source: en.wikipedia.org/wiki/Composition_with_Red_Blue_and_Yellow

2. Features

There are 3 programs: 2 console, 1 GUI.

2.1 P4Generate

This is a console program that generates a file of random rectangles. We'll use this data in our next two programs.

P4Generate queries the user for 7 parameters which guide the random rect generator:

- **num** - the number of random rectangles to create [default=100]
- **prefix** - the name prefix for all rectangles, ex: "Bill" means rectangles are named "Bill1", "Bill2", etc. [default=R]
- **paneWidth** - the width of the XY pane holding all rects, i.e. the max value of any x coordinate [def=800]
- **paneHheight** - the height of the XY pane, max y value [def=600]
- **minSize** - min size (width or height) of rect [def=5]
- **maxSize** - max size of rect [def=200]

Once queries are complete, generate the random rectangles to meet user specifications and save them to a file.

2.2 P4Intersect

This console program does the heavy lifting. It determines all the rectangle intersections. The intersections are written to a file. Go!

- ❖ Input: a rect file
- ❖ Output: a rect intersection file
- ❖ Processing: determine rect intersections using two algorithms: brute force, and sweep line
- ❖ More output: Give the run time for each algorithm

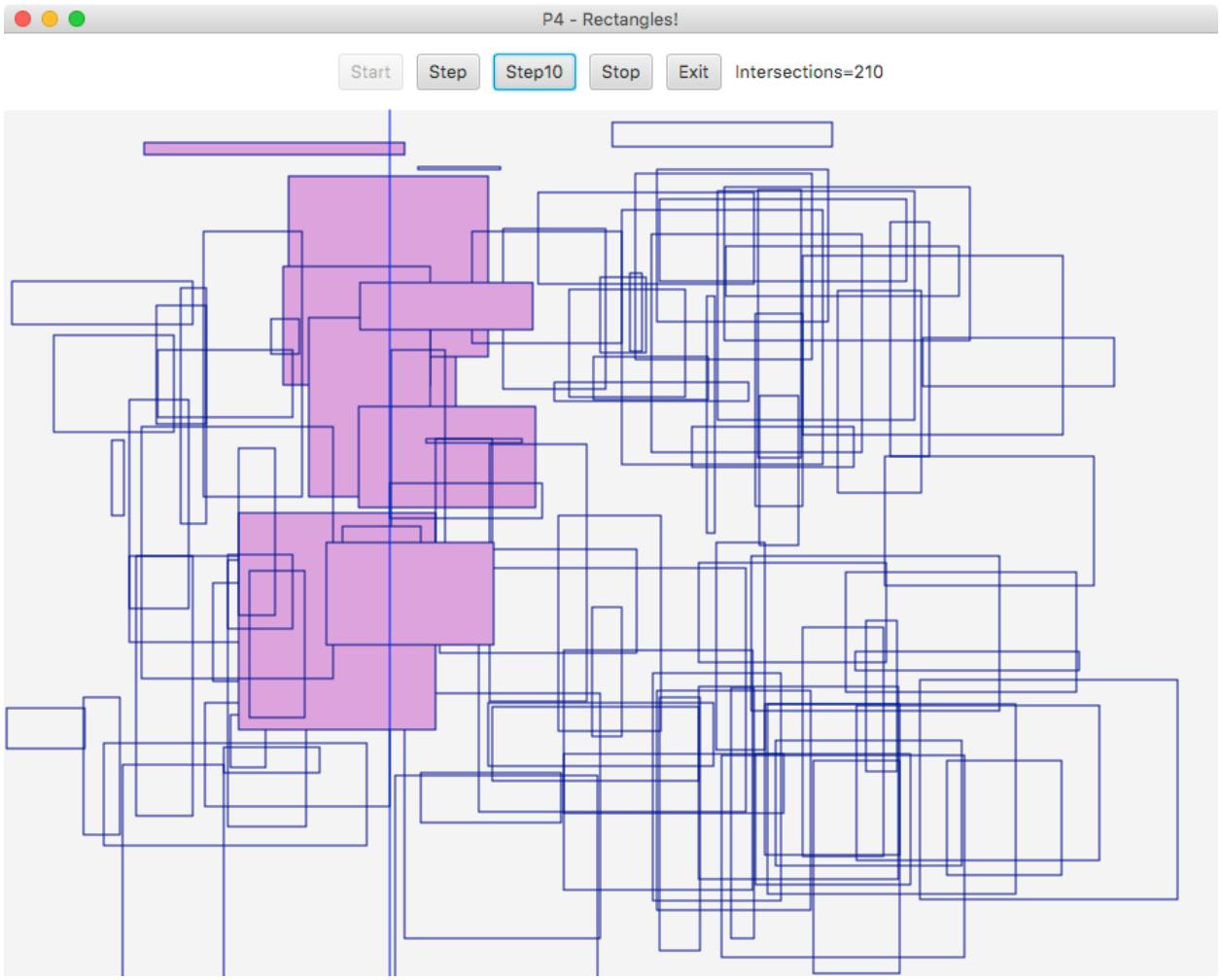
2.3 Program4

Program4 is a (fun) GUI that shows the sweep algorithm working, step-by-step. Features should include:

- ➔ Args - Let user specify a **rect file** as a program argument: main(String[] args)

- ➔ No args - If no args are specified, you can just quit or generate some random rects on your own
- ➔ Show - after reading your rect file, draw all the rectangles in your main window
- ➔ Start - Start your sweep, showing your sweep line at x=1,
- ➔ Step - move sweep line to the next rect (removeMin!), highlight “current” rects
- ➔ Step10 - do 10 steps in one button press
- ➔ Stop - end the algorithm
- ➔ Exit - exit the program
- ➔ Num intersections - show the number of rect intersections as your working

Of course, you are in charge of your own GUI (and creativity). Here’s my simple offering. Can you see the blue sweep line? (under the Start button) Current rectangles are highlighted in purple. The total number of intersections is updated with each step.



3. Design details

Three design areas: algorithms, file formats, and code.

3.1. Algorithms

The **brute force algorithm** determines the rectangle intersections by looking at each pair of rectangles. This is a double loop... the dreaded O(n^2), where n = # rects. Pseudocode:

```
for r = each rectangle 1 to #rectangles
   for s = each rectangle r to #rectangles
      if r intersect s then save it
```

The sweep line algorithm is described in the Princeton text/lecture notes. You'll find the algorithm discussed, starting at slide 37.

algs4.cs.princeton.edu/lectures/99GeometricSearch.pdf

The basic idea is to load the rects into a heap, using the x coordinate. We can use the JCF PriorityQueue as our heap. Rects are added twice, actually. Once using their x coordinate and once using the xMax (x + width) coordinate. It's like - each rect has two events, entering the sweep and exiting it. We look for intersections when the rect leaves the sweep.

3.2 File Formats

Our P4 files are simple text files.

The **rect file** format is:

```
<num-rects>
<name> <x> <y> <width> <height>
...
```

The **rect intersection file** format is:

```
<num-intersections>
<rect1 name> <rect2 name>
…
```

Please sort your intersections by rect1 name first, then rect2 name. This will be important so you can automate verification of your results versus others (like me).

3.3 Code

My code is on the k: drive. It should be sufficiently commented. (cough)
Classes are:

- **Rect210** - our primary (simple) rectangle interface; Rect210 has-a JavaFX Rectangle, so we can draw it
- **Rect210Factory** - interface specifying methods to create Rect210 objects
- **RectIntersect** - a simple class that holds two intersecting rectangle
- **RectIO** - read and write rect files; write rect intersect files; these just return null right now, you'll have to code them up

I'll start up a P4 Helper document soon.

How to succeed (writing any program):

1. Start early!
2. Don't be shy. Ask a question in class. Email me. Come to office hours.
3. Small bites. Divide and conquer your program into small, manageable tasks.
4. ABW. Always be working. Your program should always compile and run. Use the debugger. Never leave your work in disarray.

4. Grading

Create a **program4** folder on your k: drive.This folder should contain:

- All your Java source files
- Your program4 executables
- Any test input and output files that you have
- A **README.txt** file where you describe the status of your program and the creative command that you added

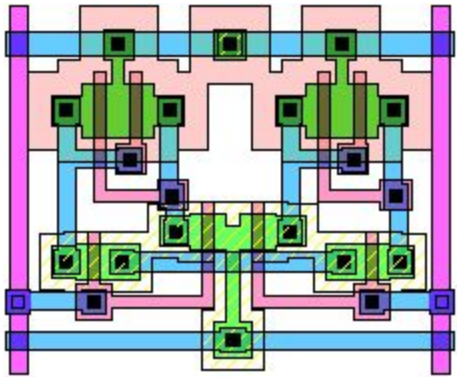
All your code must follow our class **Coding Guidelines**. Ugly code will be severely penalized. A program that doesn't even compile is probably worth 0 points.

Remember our **plagiarism** guidelines as well. Getting help from google or stackoverflow or a friend is OK, but:

1. You must acknowledge any help you receive with a comment in your code
2. You must understand any code in your solution
3. Get help on program components, not the assignment (the tic tac toe philosophy)
4. If you have any questions in this area, contact me **before** you turn in your work, not after (when it's too late)

thanks… yow, bill

PS - Rectangles and VLSI chip design



Source: www.rulabinsky.com/cavd/text/chap11-6.html