CSC 210 Program #3

Quadtrees
Jan 28, 2008

Logistics

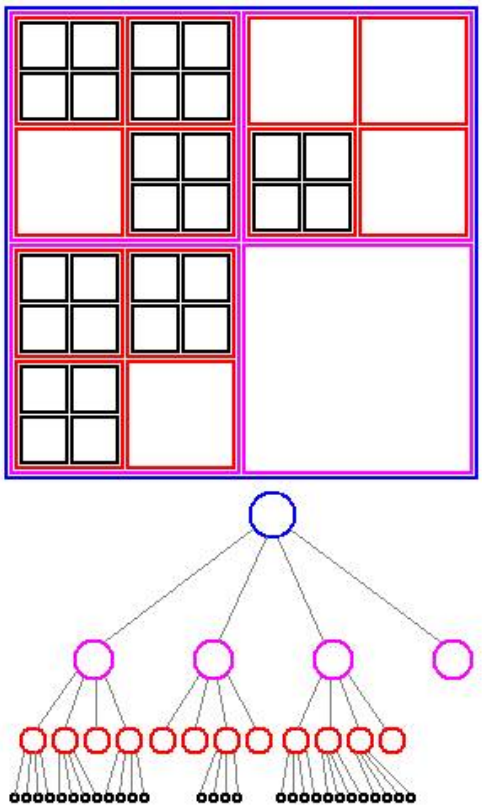
Program #3 Quadtrees is:

- **Worth** 5 points, or 5% of your grade
- **Due** Monday Feb 11, 2008
- **Covers** a cool tree structure, recursion, algorithm efficiency, mouse input, Java enums, and more!

Description

Program #3 deals with a fun, simple ADT called a quadtree. This ADT is popular in graphics and imaging applications for improving performance on operations like region searching and nearest finding the nearest neighbor to an object.

You can read a nice summary of quadtrees on Wikipedia:
http://en.wikipedia.org/wiki/Quadtree



I have created a starting point for you, a quadtree interface, defining the ADT: QuadtreeInt.java. The basic quadtree operations are:

- Insert an object at some point
- Remove an object
- Find an object at a point
- Find all objects in a region

What's my take on all this? Well, a quadtree is like a binary tree, except it has four children. The "data" held in a quadtree node is typically some object… we'll just use a Point object. You can get fancier than that if you like.

The four children of a Quadtree node represent the division of a square into four parts, one for each direction: NorthWest, NorthEast, SouthEast, and SouthWest.

Of course, the structure is recursive in that each node of the quadtree subdivides its parent into four equal parts.

Perhaps a picture will help… like that one right over there.

In Program #3, I want you to implement the `QuadtreeInt` interface, slap it into a Panel, and create some fun GUI controls so that we can play with it. My `QuadtreeInt` code is also available on the k: drive.

Hints

I have lots of hints on this one.

1. Read Chapter 8 on Trees. A lot of it is applicable to quadtrees: nodes, recursive methods, etc.
2. Please keep your ADT separate from your Panel and your GUI. Some of you had a problem with this in Program #2. I have separate `Quadtree` and `QuadtreePanel` classes. I haven't done my GUI yet, but that will probably include a number of new, separate classes. The question to answer is "Can someone else easily use my ADT without modifying my code?"
3. I did an `enum` for my directions. In fact, I did a number of utility functions that made my implementation a lot easier to code and to understand.
4. I didn't store the (x,y) or bounding box of each node. I start at the top and keep track of it as I'm traveling down the hierarchy. I pondered that one for a bit, but I like how it turned out. I'm lean and mean.
5. I used the following Java API classes: `List`, `ArrayList`, `Dimension`, `Point`, and `Rectangle`. There are a couple nice `Rectangle` methods in there like:

```
boolean contains( Point p);
boolean intersects( Rectangle r);
```

6. Look at page 422-424 of our book. That's pretty much how my insert() works. I have a public `insert()` method. It calls a private method that I called `doInsert()`. Also, I did my pseudo-code and drawings on paper before starting the code… just like the book.
7. My `Quadtree` constructor creates an empty node at the root. My root is never `null`.
8. There are nice examples on mouse input in Appendix C, pages 813-823.
9. I want you to add some way to insert many points to your quadtree, either through a file or randomly. We'll work this out in lecture.
10. For simplicity, you can assume that your bounding box, the size of your quadtree's universe, is a power of 2.

Here's someone else's cool Quadtree browser:

http://donar.umiacs.umd.edu/quadtree/points/prquad.html

I have attached a page of crappy drawings in the hardcopy version of this handout. See me if you missed it.

For your creative flair, I would like to see you add an algorithm to (quickly) find the nearest neighbor to any point in your quadtree. By “quickly”, I mean without looking at each node to do it. We’ll see how it goes though.

Grading

As you did for Program #1… by the due date, please place your work for Program #2 in your folder on the k: drive. I'll be looking for:

1. **Your README file** – describing the state of your program... what works, what doesn't, what your “flair” is, etc.
2. **Your Net Beans folder** – including your Java source code, class files, etc.
3. **Your Javadoc** – generate using the “Build/Generate Javadoc” menu
4. **Your applet** – create a web page at w:/index.htm with an Applet of one of your favorite trees. Please include a link to your Javadoc.
5. **Your printout** – Please print one source file… your most important one, so that I have someplace where I can scratch my comments.

I expect your code to be beautiful. “No crappy code”™

I think this is a fun assignment.

good luck… yow, bill

Additional Notes

I’ll post additional notes directly on the blog.