CS102

Introduction to data structures, algorithms, and object-oriented programming

May 4, 2016
**BreakOut game**

- Problems?
  
  Some people reported problems with jagged edges.

  It's not possible to build a circle out of squares, but there is a technique that can eliminate some of the jaggedness of images:

  antialiasing: based on transparency.

  If 50% of a pixel is covered by the geometric figure that you are trying to draw, then color that pixel with a color that is 50% transparent. If 25% of the pixel is covered, use a color that is 75% transparent (25% opaque). If the entire pixel is covered by the figure, of course, use a color that is 100% opaque -- antialiasing only affects pixels that are only partly covered by the geometric shape.
BreakOut game

To make images and text more smooth when drawn on the scene, you can cast the Graphics object as a Graphics2D object in the drawFrame method (or inside any method in which you use the graphics object) and then use the setRenderingHint method as shown below to make the images sharper.

```java
Graphics2D g2 = (Graphics2D)g;
g2.setRenderingHint(
    RenderingHints.KEY_ANTIALIASING,
    RenderingHints.VALUE_ANTIALIAS_ON);
```
Improvements to BreakOut game?

List 3 changes you could make to the BreakOut game from lab on Monday that would make it more challenging/fun.

Change your current implementation such that each row of bricks has a different color. *Hint: Use a parallel array of colors that has the same number of elements as the number of rows in your wall.*

Change the implementation so that the user is allowed to play (and lose the ball) three times before they lose. Otherwise, increase their score by 10 each time they take out a brick.

More?
Parallel Arrays

- At times you may need to declare arrays that work together and have the same size. This may be more convenient than making objects with multiple fields.

- These types of arrays are often called "parallel" arrays because they can be used in synchrony or solo for different effects.
Multithreading

• You may have heard of the term *multithreading*. This refers to using different processors on your machine at the same time, or programming in parallel.

• You have been using multithreading without really knowing it when you set up multiple listeners in a graphics application. Each listener is in its own thread and each is operating concurrently.
Parallel Programming using multi-core processors

• If your machine has a multi-core processor, it may be able to run many different threads at the same time, or in parallel.

• We did not cover threads in this class, but chapter 12 of our book covers this topic and you should know enough about Java to make sense of the code examples.
Analysis of Algorithms

• There are 2 main quantities that we measure when talking about the efficiency of algorithms:
  
  **Time:** How many steps does an algorithm take on an input of size n as n grows larger?

  **Space:** How much memory is used by the program on an input of size n?

We express these terms using asymptotic analysis which specifies the change in running time as n grows to infinity.
Analysis of Algorithms

• We use a notation system called "Big-Oh" notation to express running time and space used, both in terms of the size of the input.

• A single for loop that goes through all the elements in an array runs in $O(n)$—linear time

• A nested for loop that goes through rows and columns of an array runs in $O(n^2)$ time—quadratic time
Analysis of Algorithms

• There is an area of computer science that is concerned entirely with measuring the running time of different algorithms, particularly to measure the running time of algorithms that solve the same problem (e.g., sorting).  


Where do we go from here?

• We will have lecture and lab on Monday, May 9th

In this lab, I may give you some pseudocode and ask you to implement the code. You will also learn about how to mechanically time your programs to get an idea of how fast they run on large data sets.

Usually, if the pseudocode for one algorithm takes more steps than another, it will take a longer time to run on large data sets. We will test this with some algorithms designed to sort "comparable" data in increasing order.
Final Exam Reminders

- Read Chapters 1-10, and 13 of our on-line textbook.

- Final exam: Friday, May 20th from 5-7 pm, SP 309

You will be allowed to bring 2 letter-sized sheets of paper with notes or printing on both sides. No other notes are allowed for the exam.