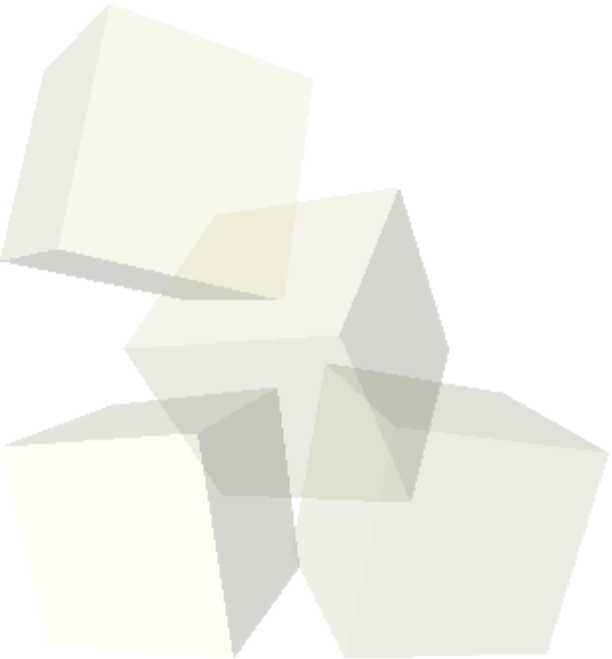
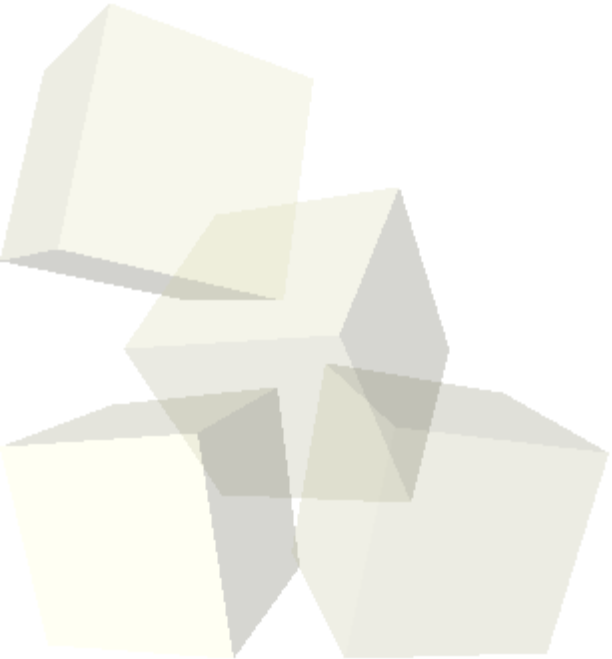
Refactoring (and maybe some Recursion)

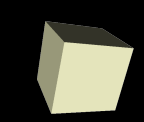
3/26/2009



Opening Discussion

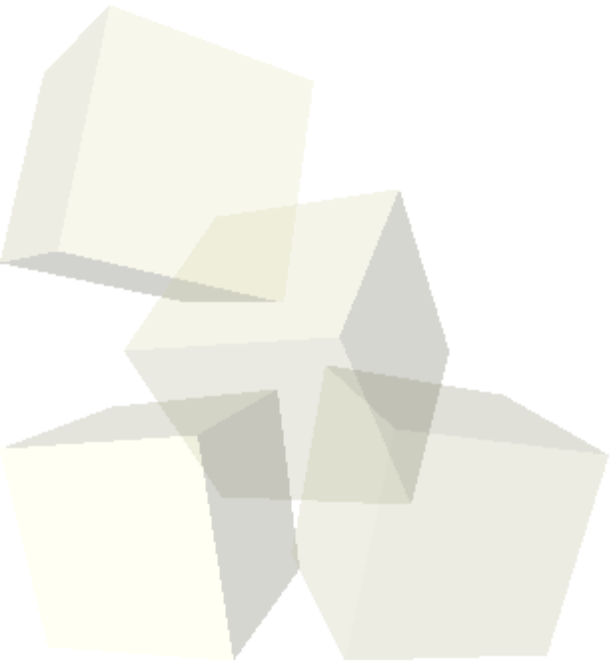
- Let's look at solutions to the interclass problem.
- Do you have any questions about the assignment?
- Do you have any questions about the reading?





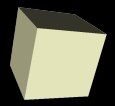
Code

- I want to get our Matrix animation working.
- We want a button in the the properties panel that lets us star and stop the animation.



Refactoring

- This is something that you do when you don't want to change the functionality of your code, but you want to change how it does something.
- You typically refactor your code when it “smells.” Your book lists many different smells.
  - Long method
  - Large class
  - Duplicate code
  - Shotgun surgery
  - Switch statements
- There are several different types refactoring. Eclipse has automatic tools that do a number of these.



Refactoring Code

- There is an aspect of our drawing program that fits with one of the “smells” that you read about. It is related to how we add objects into our tree. I'd like to use a mechanism that doesn't have a switch statement and uses polymorphism instead.
- To do this we need a type that represents objects that know how to create Drawables so we can make an array of those.

Recursion

- You should have learned about recursive functions in PAD1. A recursive function is simply a function that calls itself.
- You can use recursion to imitate loops, but we won't do that very often in C or Java. Where recursion comes in really handy is when a function needs to test more than one alternative at a time.
- This works nicely because the call stack remembers where you are in a given function so when you return back, you can take off from that point again.

Maze Solving

- One of my favorite recursive algorithms is maze solving. This is a special case of graph traversals which are common problems in CS.
- We'll use a 2D array of ints as our maze and we can even put this into our drawing program.
- A simple warm-up piece of code is flood fill like you would have in a drawing program.
- Once we have that we can see how to convert it to do things like find the shortest path through a maze or count all paths through a maze.
- We can try to make this nice and graphical as well so it fits properly into our drawing program.

Formula Parsing

- Another one of my favorite recursive algorithms is formula parsing. This allows us to have the user type in a function and our code can evaluate it.
- We do this through “divide and conquer”. We split the formula in two across the lowest precedence operator then recursively evaluate the two halves.
- We can use this to put function plotting into our program if we give it the ability to handle a variable.

Minute Essay

- When is recursion better than loops and what makes it more powerful?
- The assignment #4 is due today.
- We have a quiz on Tuesday.
- Interclass Problem - Write a recursive function to find the longest path through a maze.