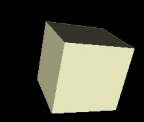
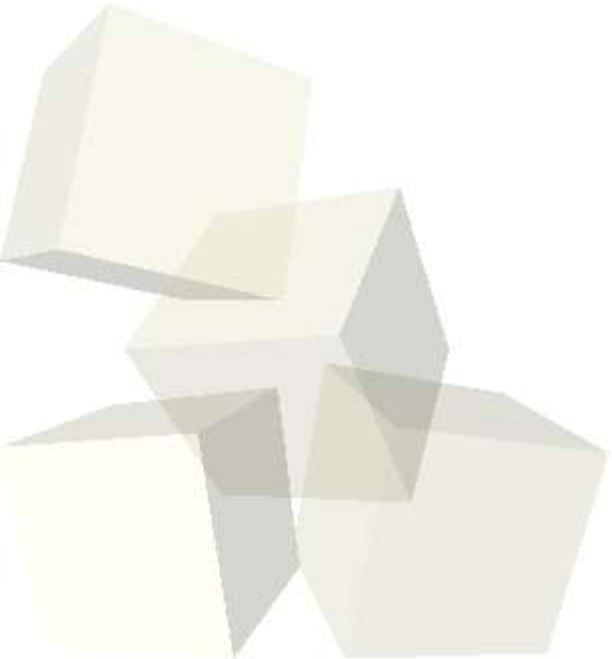


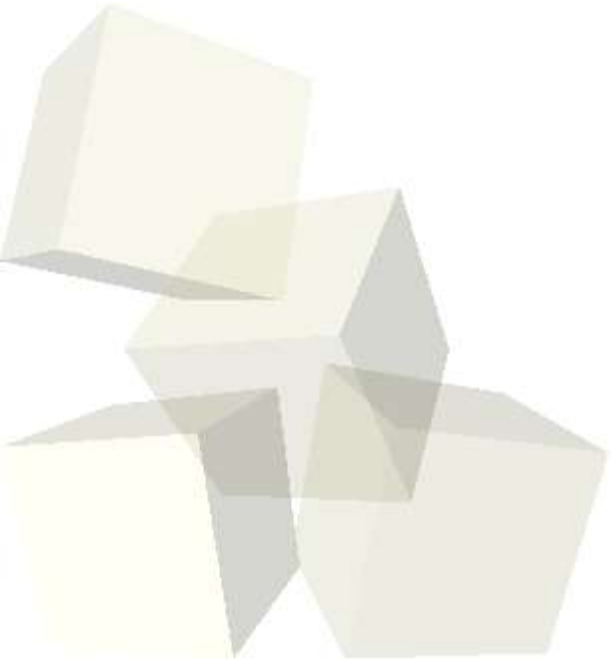
Problem Decomposition

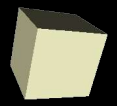
8-31-2005



Opening Discussion

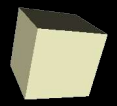
- What did we talk about last class? Did anyone explore any of the problems we looked at last time?
- How do we break up problems in imperative languages? Why do we like to break up problems in all languages?





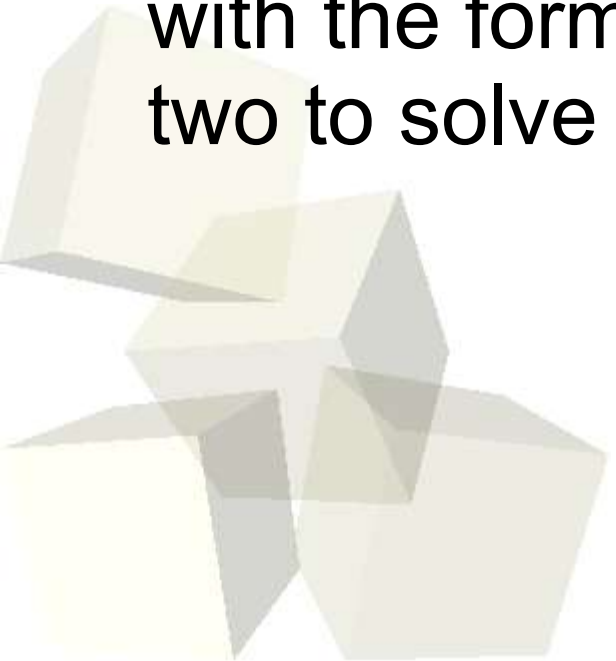
Functions

- In nearly every programming language, the primary construct for breaking up problems is the function. Some languages will use the terms procedure (if nothing is returned) or method, but the general idea is the same.
- The idea of a function is that it is a piece of code that we can call from someplace else and pass it certain data. The function then returns zero or more values.
- How do you define functions in the languages that you are familiar with?



Manageability

- The most basic reason we like to break our code up into functions is to keep things manageable. Large problems are very hard to solve if we try to tackle them all at once. We have to break them into smaller pieces that we can actually deal with.
- Problems can be decomposed in either a top-down or a bottom-up manner. We generally start with the former and often use a combination of the two to solve large problems.



Code Reuse

- From a software engineering point of view we want to break problems up into functions because a lot of the time we can use those functions in multiple places so the code is effectively reused.
- Though we don't write them, libraries are a perfect example of this. Most scientific computing depends very heavily on good libraries of functions because not every practitioner is going to have time or ability to write optimal code for every task.
- Let's try to make our quadratic equation code from last time more modular by working in a function or two.

Collections of Data

- Another aspect that is critical to making imperative programming workable is the ability to collect data together.
- In most standard imperative languages we do this with arrays or structures/records.
- Arrays all hold the same type and we specify which element with a numeric index.
- Structures/records can hold different types and each element is specified by a name.
- We need this to break the programs from last time into functions. Why do we need to have data grouped? What approaches can we take to do it?

Minute Essay

- This was our last class talking about imperative programming. Did this give you enough of a review? Is there anything you want to spend a while talking about next class before we move onto Matlab?
- You should read the second chapter of the Matlab book for next class. If you don't have your book yet, you can read the Wikipedia entry for “floating point”.