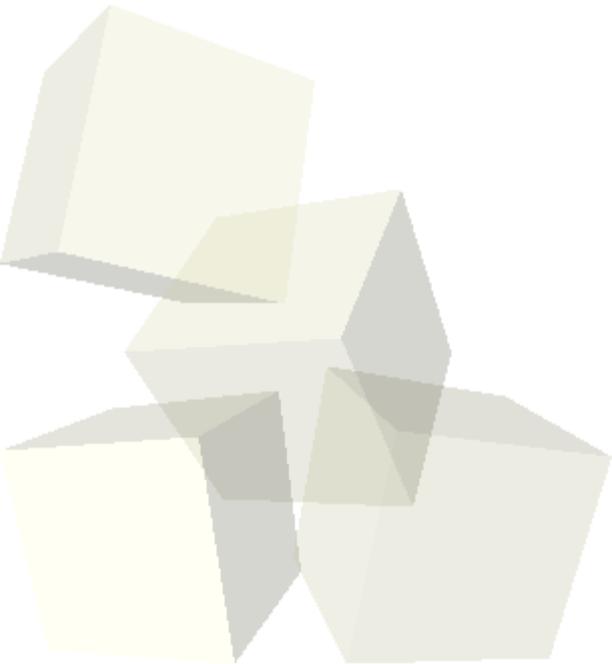
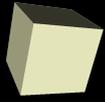




# Flow Control and Basic RegExs

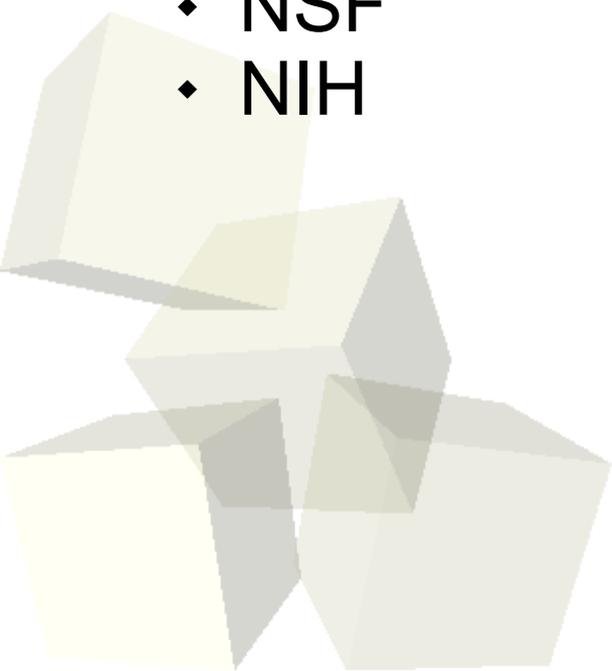
3/31/2008





# Opening Discussion

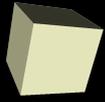
- Do you have any questions about anything?
- I've redone the schedule to reflect content in the book we are using and spread it out to fit our needs.
- Notes on peer review panels and different systems.
  - ◆ NASA
  - ◆ NSF
  - ◆ NIH





# Conditionals in Perl

- Perl has the standard if statement that you are used to.
  - ◆ Use the normal comparison operators for numbers.
  - ◆ For string data you use letter symbols. Use eq for equality.
- The if can follow a single statement.
- Format with if in front requires curly braces so they have elsif.
- There is also an unless that works like if, but the action happens when it is false. This exists because Perl programs don't like to use not.
- Boolean operators and, or, not are in English, not symbols. Short circuit so can be used for flow control.



- Perl has a full compliment of loops. Most are just like what you are used to in C family language.
- while loop is the same.
- do-until instead of do-while.
- for loop is the same.
- for each loop is different and goes through the elements of a list. We saw how to use that one last class.





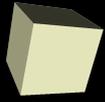
- We saw how to use `open` to open a file for reading and then how to read from it.
- If you precede the file name with a “>” the handle will be for writing. (It's like directing output of a program to file in Linux.)
- That deletes an existing file. Use “>>” to append. (Also like Linux.)
- The file handle becomes the first argument to `print`. There is no comma after the handle.





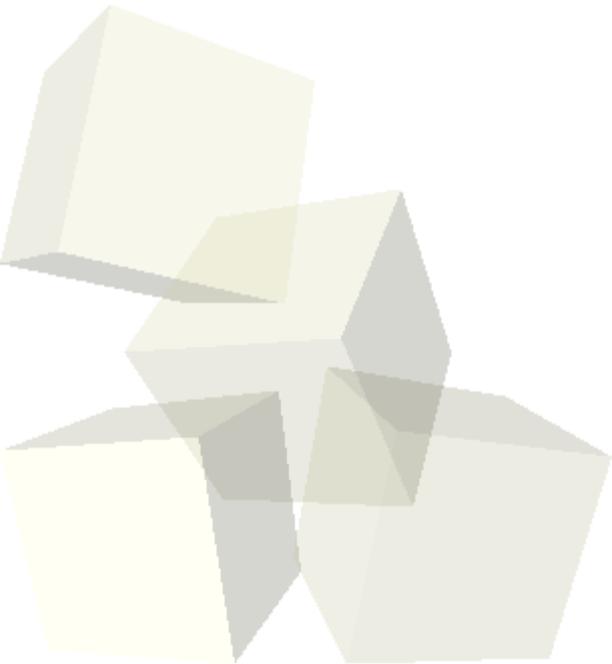
# Regular Expressions

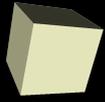
- Regular Expressions are a really big deal in Perl. They are a significant part of why so many people use Perl.
- Regular expressions can be used with several types of operations.
- Matching – put the matching expression between matched symbols, typically `//`.
- Substituting – The normal format is `s///` where after the first `/` you put the expression to match and after the second `/` is what to replace it with. Put a `g` at the end to substitute multiple. An `i` to ignore case.
- Transcription – replace chars using `tr///`.



# Binding for RegExs

- By default a regular expression will happen on the variable `$_`. To make it happen on something else use the binding operator, `=~`.





# Closing Remarks

- I think you know enough Perl to take a quiz now so we will have quiz #4 next class.

