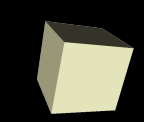
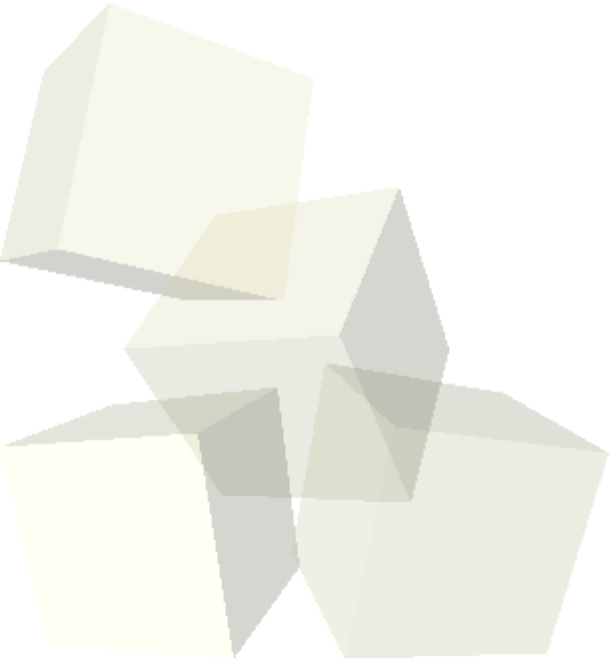


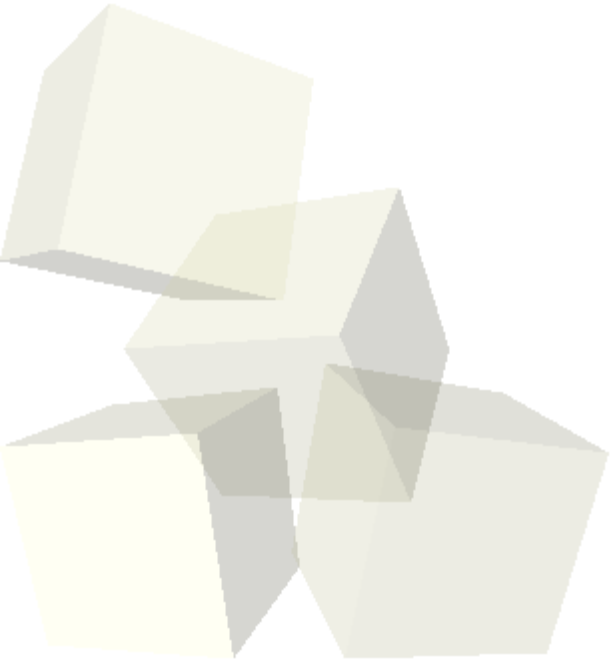
Introduction to Perl

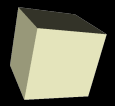
3/14/2008



Opening Discussion

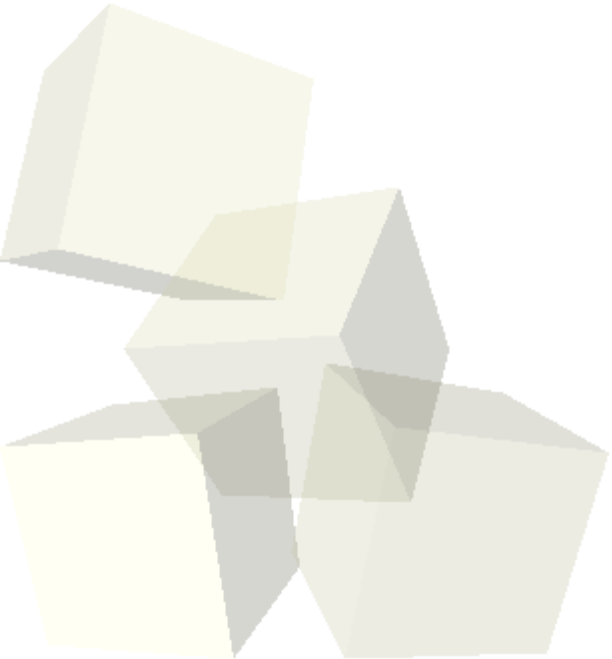
- How is the project coming?





Eclipse

- To start with let's all open eclipse. To do this bring up a terminal and type eclipse.
- We want to go to the Perl perspective instead of the default Java perspective.
- We can also create a Perl project where we will put the scripts that we write for this class.



Basics of Perl

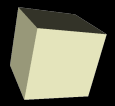
- Perl is a scripting language so the scripts aren't compiled. You simply run the scripts.
- Perl does have a JIT compiler to improve performance.
- Scripts can be written with any text editor.
- We have installed the Perl environment under Eclipse on these machines that you can use if you want nice syntax highlighting. You will get syntax highlighting with vi as well if your file ends in .pl.

Simplest Program

- Let's write the standard first program for a new language: hello world.
- This really illustrates what defines scripting languages. They are very efficient for writing short programs. They don't have the overhead of languages like C or Java.
- We use the print command for printing.
- From the command line you can do “perl script.pl” for whatever script you have written to run it. If you make it executable with chmod you should be able to simply execute the script.

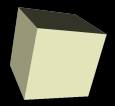
Perl Variables

- We don't have to declare variables in Perl unless we use the strict modifier on a file.
- Even when we do declare then we don't really specify a type. Perl basically has three types: scalars, lists, and hashes.
- One of the hardest things to get used to in Perl is the extra characters that have to be attached to variable names.
- For standard variables you put a $ in front of them in their general usage.
- For lists we will put @ and for a hash (something we will get to later) you put a %.



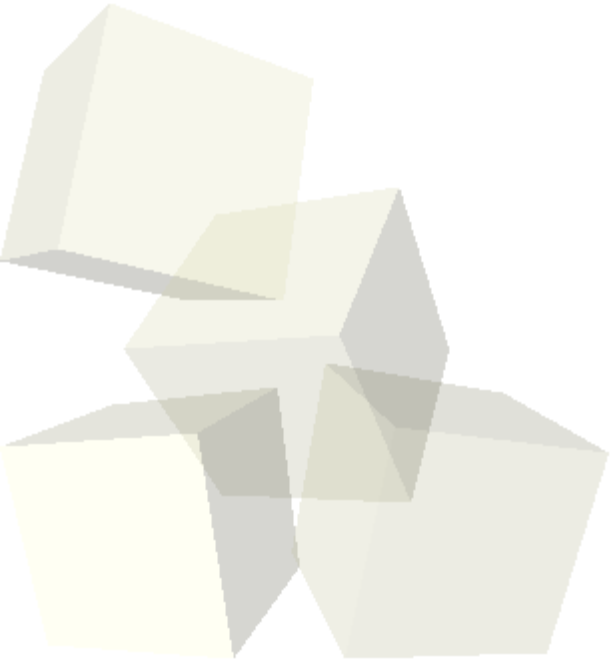
Type Conversion

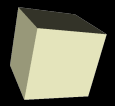
- Perl is very loose and easy with types and does implicit type conversions.
- For this reason, operators aren't overloaded. The operator tells Perl what type it should be using.
- Numbers in Perl are doubles.
- If you use a string in a place where a number is wanted, Perl tries to convert it.
- If you use a number where a string is wanted, Perl will convert it.



Comments

- Comments in Perl are made with the # sign. Everything after the # is a comment.
- Using #! tells the system what program should be used to run a script and is generally found on the top line of Perl scripts.



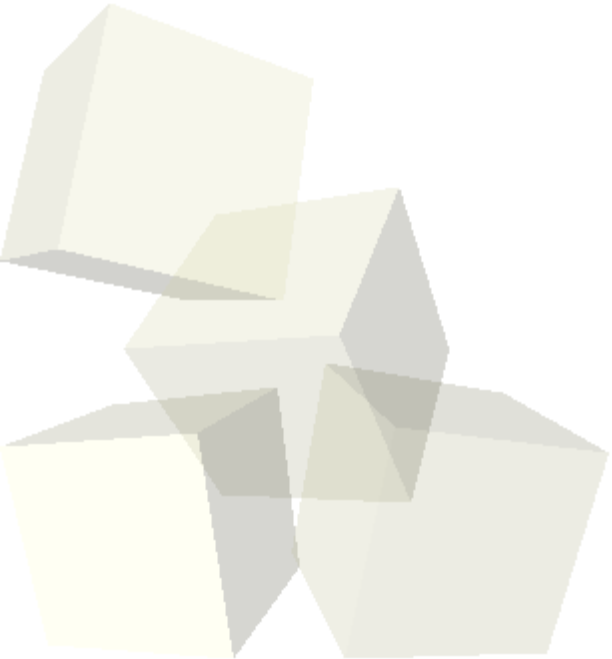


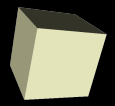
Basic Syntax

- On the surface, the basic syntax of Perl looks a lot like C. At least for things like assignments, conditionals, and loops.
- Functions look a bit different, but we'll get to those later.
- You use + to add numbers and . To concatenate strings.
- Strings with double quotes will substitute values for variables that appear in them. Strings in single quotes do not do this.

Files

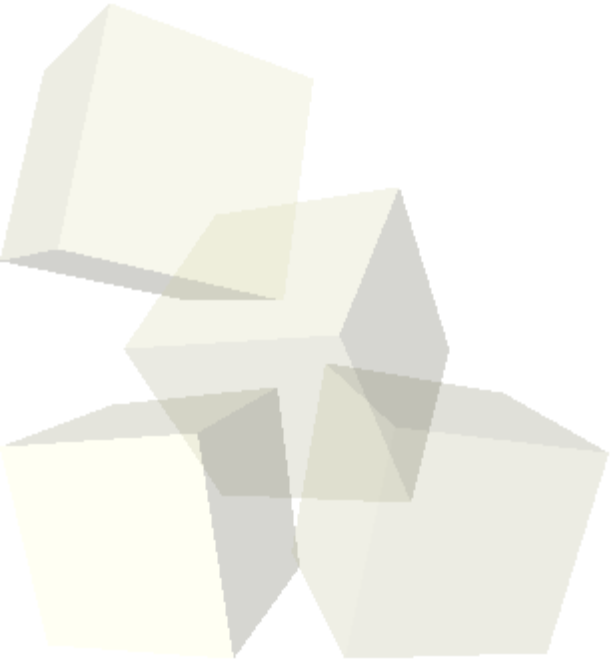
- The open and close commands are used to open and close files.
  - open(FILE_HANDLE,$filename);
  - close FILE_HANDLE;
- Putting a file handle in angle braces, <...>, will read from that file.

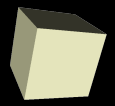




Arrays/Lists

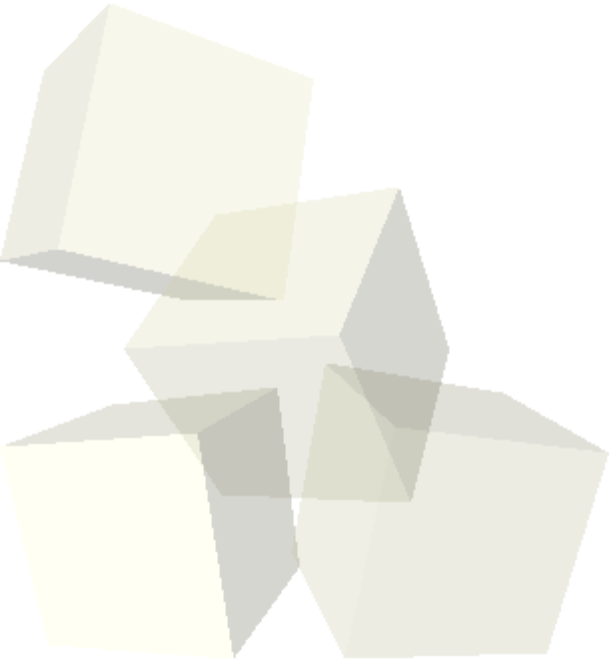
- List literals are specified with parentheses and have the values inside separated by commas.
- Array variables are prefixed with @, but only when referring to the whole list.
- We get out elements with []. Because we are pulling out a single scalar the variable is preceded with a $.

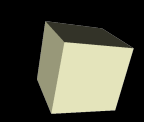




Context

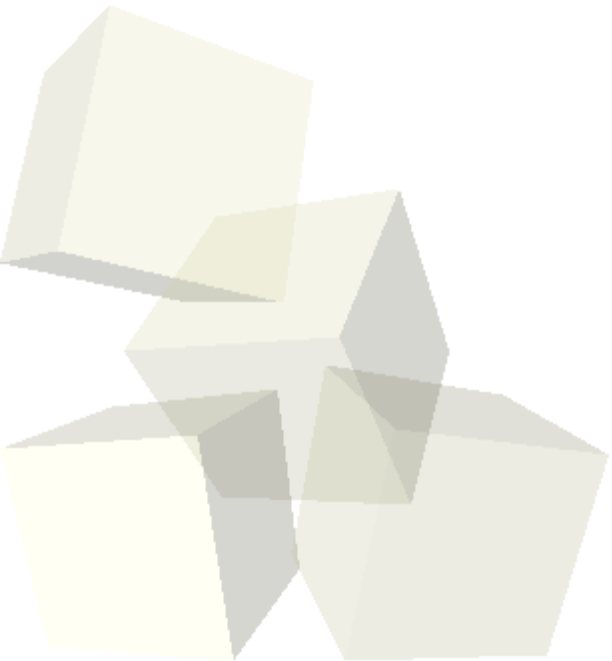
- How Perl interprets many statements depends on the context. Normally scalar vs. list context.
- If you read a file in a scalar context it will give you back one line from the file. If you read it in a list context it will give you a list with all the lines.





Problems

- Let's continue to play around in Perl and do some reading from files to see how different things work.



Reminders

- Enjoy your spring break! We have class on Monday after spring break, but won't on Wednesday or Friday.