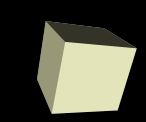
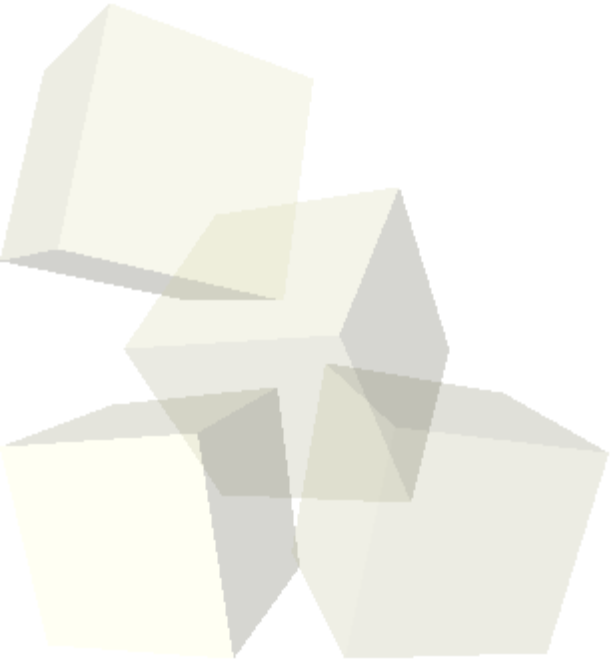
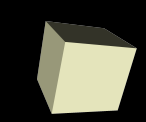


Bitwise Operators

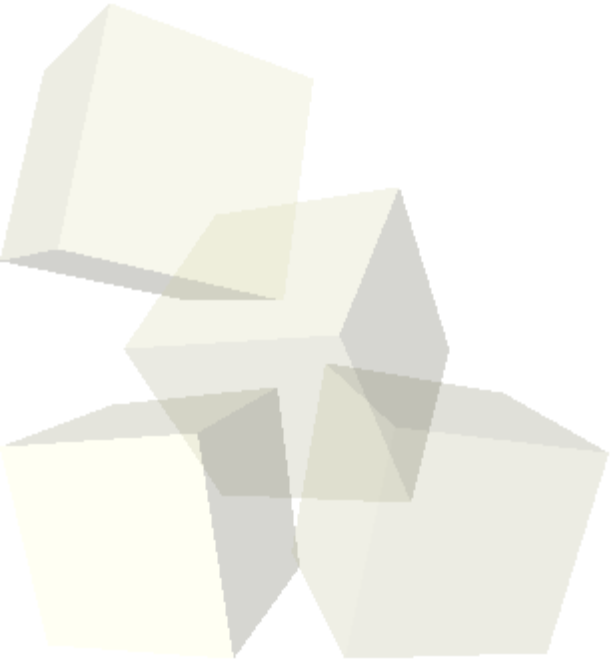
11/19/2007





Opening Discussion

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



Motivating Bitwise Operators

- There are a number of situations in which being able to play directly with the bits in numbers can be helpful.
- Consider an application where you must keep a large number of boolean values.
- Another is when you need to build a single number from several other values, often booleans.
  - Packing booleans
  - Packing RGB values
  - Passing multiple flags to a function

&, |, and ^

- There are three binary bitwise operators. That is, three operators that work on the individual bits in numbers and take two operands.
- & is bitwise and. A bit is only turned on in the result if the bit in that position in both operands is on.
- | is bitwise or. A bit will be on in the result if either of the bits in that position in the operands are on.
- ^ is bitwise xor (exculsive or). A bit in the result will be on if it is on in only one of the two operands, but not both.

- This is the bitwise not operator.
- It is unary.
- Applied to an integer value it flips the states of all the bits in that value.
- Note that ~0 gives you a number with all the bits on, regardless of the number of bits.

>> and <<

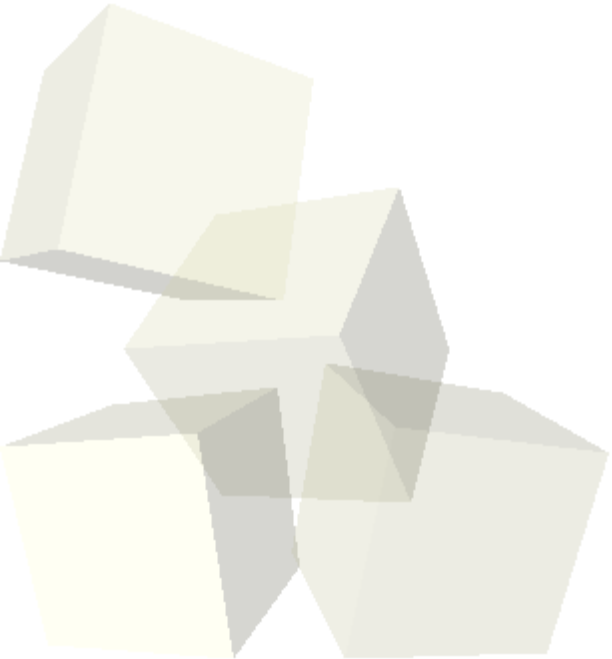
- These are bit shifting operators.
- They are binary and move the bits in the first argument either up or down by the number of places specified in the second argument.
- The upshift, <<, basically works like multiplying by a power of two while downshift, >>, is basically dividing by a power of two.

Combining Operators/Masks

- Combining the different bitwise operators allows us to specifically check or set any combination of bits in a number that we want to.
- This is the real power of bitwise operators.
- These are often combined with hex literals for masking off different parts of numbers.
- ^ allows you to swap ints without a temporary. Don't bother doing this. It's just a fun trick to know.

Assignment Operators

- All the binary bitwise operators can be used in an assignment form by putting them before = in an assignment.
- So for integers, a*=2 is the same as a<<=1.
- I find that I use &= and |= fairly often when going through loops that build values through bitwise operations.



Minute Essay

- For the fun class would you rather do cover ASCII based graphics or X11 based graphics?
- Interclass Problem – Do problem 46 on page 922.