Announcements for This Lecture

Today's Material

- Section 2.3.8 (first use of loops in the text)
- All of Chapter 7
- Two topics covered today
  - Elementary graphics
  - For-loops

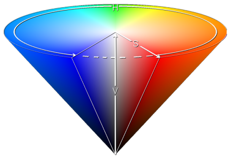
Both used on A5

Assignment A4

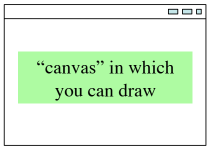
- Assignment due Tonight!
  - **Remember to report your time in the comments!**
- Rounding in assignment
  - **Do not** use roundTo5 in your conversion methods
  - Rounding only happens in the A4Tester and toString()
- New code files posted
  - A4.java, A4.jar are fixed

A4: One Last Time (I Promise)

- Color Ranges
  - R, G, B should be 0 to 255
  - C, M, Y, K should be 0 to 100
  - H should be 0 to 360
  - S, V should be 0 to 1
- Files updated online
  - A4.java
  - a4.jar
- If you did it right, does not effect you



Drawing Canvases



- GUIs often treat window interior as a canvas
- Each pair (x,y) is a "pixel"
  - Position you give a color to
- For A5, understand that
  - x-coords increase to right
  - y-coords increase down

(0,0)	(1,0)	(2,0)	…
(0,1)	(1,1)	(2,1)	…
(0,2)	(1,2)	(2,2)	…
…			

Graphics Programs with ACM Package

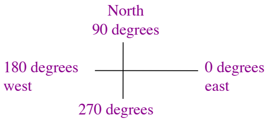
```
import acm.graphics.*;
import java.awt.*;
import acm.program.*;
/** An instance maintains graphics
  * window on the monitor */
public class GDemo extends
                        GraphicsProgram {
   /** Constructor: an instance with
    *   canvas of size   (500, 500) */
   public A5() {
       super();
       start(sizeArgs);
   }
  ...
}
```

Usage Examples

- Creating a turtle
  - GDemo demo = new GDemo();
  - GTurtle t = demo.getTurtle();
- Drawing with the turtle
  - t.forward(200);
  - t.left(125);
  - t.forward(250);
- Can also draw w/ pen

A5: Drawing with the Turtle

- Features of class GTurtle
  - point (x, y): where the "Turtle" is
  - angle: direction the Turtle faces
  - color: the Turtle pen color
  - whether the pen is up or down.
- Draw equilateral triangle:
  - t.forward(30); t.left(120);
  - t.forward(30); t.left(120);
  - t.forward(30); t.left(120);



- Use all of this in A5
  - Draw spirals and shapes
  - Most procedures will be recursive in some way

Important Concept in CS: Doing Things Repeatedly

1. Perform *n* trials or get *n* samples.
   - A5: draw a triangle six times to make a hexagon
   - Run a protein-folding simulation for 10^6 time steps
2. Process each item in a String, Vector, or "list"
   - Compute aggregate statistics for a dataset, such as the mean, median, standard deviation, etc.
   - Send everyone in a Facebook group an appointment time
3. Do something an unknown number of times
   - CUAUV team, vehicle keeps moving until reached its goal

From Recursion to Loops

- Recursion can do all this
  - Do something
  - Call method to do again
  - But how do you stop?
- Iteration is an alternative
  - while-loops
  - for-loops

```
<set things up>;
while (stuff still to do) {
    <process current item>;
    <prepare for next item>;
}
```

Recursion can do anything iteration can, and vice versa

- Some problems easier with recursion, other with iteration
- You will understand which more as you gain experience

For Loops: Processing Ranges of Integers

```
int x;
x= 0;

// add the squares of ints
// in range 2..200 to x
x= x + 2*2;
x= x + 3*3;
…
x= x + 200*200;
```

- For each number i in the range 2..200, add i*i to x

The for-loop:

```
for (int i= 2; i <= 200; i= i +1) {
    x= x + i*i;
}
```

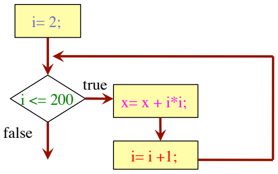
- **loop counter**: i
- **initialization**: int = 2;
- **loop condition**: i <= 200;
- **increment**: i= i + 1
- **repetend**: { x= x + i*i; }
  - Also called the **body**

For Loops: Processing Ranges of Integers

The for-loop:

```
for (int i= 2; i <= 200; i= i +1) {
    x= x + i*i;
}
```

loop counter: i
initialization: int = 2;
loop condition: i <= 200;
increment: i= i + 1
repetend: { x= x + i*i; }



To execute the for-loop.

1. Execute initialization.
2. If loop condition false, terminate execution.
3. Execute repetend.
4. Execute increment, repeat from step 2.

Note on Ranges

- m..n is a range containing n+1-m values

2..5 contains 2, 3, 4, 5.	Contains 5+1 – 2 = 4 values
2..4 contains 2, 3, 4.	Contains 4+1 – 2 = 3 values
2..3 contains 2, 3.	Contains 3+1 – 2 = 2 values
2..2 contains 2.	Contains 2+1 – 2 = 1 values
2..1 contains ???	

- The notation m..n, always implies that m <= n+1
  - So you can assume that even if we do not say it
  - If m = n+1, the range has 0 values

Application: URL Analysis for Search Engines

- How does Google rank its web pages?
  - (Part of the Answer): Use clues from the URL
- “Deep” URLs are usually less important
  - Example:
    www.fake.com/this/that/other/minor/tiny/detail.htm
  - Count number of slashes in URL (given as String s)
- We need a loop to count number of ‘/’ in String s
  - so we need a loop to look at s[0], ..., s[s.length()-1]
  - so we need a loop to process integers in 0..s.length()-1

Patterns for Processing Integers

range a..b-1

```
for (int i= a;  i < b;  i= i + 1) {
    Process integer i;
}
```

```
// store in count # of '/'s in  String s
// inv: count is # of '/'s in s[0..i-1]
count= 0;
for (int i= 0; i < s.length();  i= i +1) {
    if (s.charAt(i) == '/')
        { count= count+1;  }
}
// count is # of  '/'s in s[0..s.length()-1]
```

range c..d

```
for (int i= c;  i <= d;  i= i + 1) {
    Process integer i;
}
```

```
// Store in double var. v the sum
// 1/1   + 1/2 + ...+ 1/n
v= 0; // call this 1/0 for today
// inv: v is 1/1   + 1/2 + ...+ 1/(i-1)
for (int i= 1; i <= n;  i= i +1) {
    v= v + 1.0 / i;
}
// v= 1/1   + 1/2 + ...+ 1/n
```