CS1110, 20 Oct. 2009, Lec 14
Elementary graphics; intro to loops and for-loops

Reading: Sec. 2.3.8 and chapter 7 on loops.
The lectures on the ProgramLive CD can be a big help.

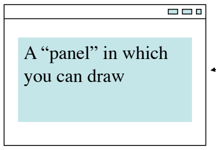
A5 is out: graphics, loops, recursion. Due Sat. Oct 30th.

Prelim 2: Tu Nov 9th, 7:30-9pm.
Conflicts? Submit CMS "assignment" "P2 conflicts" by Oct 26th.

Talks on Thursday Oct 21:
Adaptive Drama Management:
Bringing Machine Learning to Interactive Entertainment
4:15pm, Upson B17
(computer science, game design, social psychology)

Computational Advertising, 7pm, Phillips 203
(intersection of computer science and econ)

Graphical User Interfaces (GUIs): graphics.



A JFrame, with a "panel" on which you can draw

On the panel, each pair (x,y) indicates a "pixel" or picture element.

For Assignment 5, you need to understand that:

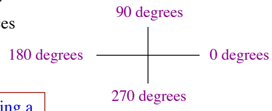
x-coordinates increase rightward
y-coordinates increase downward.

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

Assignment A5: drawing with a Turtle

We'll use ACM's GraphicsProgram, which supplies a "GTurtle":

- point (x, y): where the "Turtle" is
- angle: the direction the Turtle faces
- a pen color
- whether the pen is up or down



Class GTurtle has methods for moving a GTurtle around, drawing as it goes.

Draw equilateral triangle with side lengths 30; turtle t ending up at starting point and facing the same direction:

```
t.forward(30);   t.left(120);
t.forward(30);  t.left(120);
t.forward(30);  t.left(120);
```

In A5, write methods to draw shapes and spirals, and draw things using recursive procedures.

From recursion to loops: doing things repeatedly

We write programs to make computers do things.
We often want to make them do things *multiple times*.

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 given String, Vector, or other "list"
   - Compute aggregate statistics for a dataset, such as the mean, median, standard deviation, etc.
   - Send everyone in a certain (Facebook) group an individual appointment time
3. Do something an unknown number of times
   - ALVINN, the van that *learned* to drive itself, continuously watched human driving behavior and adjusted its model accordingly

From recursion to loops: doing things repeatedly

We've talked about *recursion*.
Alternatives:

while-loops

for-loops (special syntax for common special cases)

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

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

The for loop, for processing a range of integers

```
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.

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

The for-loop:

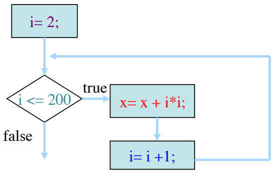
```
for (int i= 2; i <= 200; i= i +1) {
   x= x + i*i;
}
```

Execution of the for-loop

The for-loop:

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

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

Called a "flow chart"

Application: URL analysis for search engines

Problem: how does a search engine (e.g., Google) decide which webpages are the most important to present?

(Small) part of the answer: use URL cues

- "Deep" URLs are usually less important, e.g., www.fake.com/this/that/other/minor/tiny/detail.htm

This requires counting the number of slashes in a URL (given as a String).

You know a recursive solution; next slide: loop solution.

The pattern for processing range of integers:

range a..b-1

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

range c..d

```
for (int i= c;  i <= d;  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]
```

```
// 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
```

Note on ranges
(later, will make reasoning about loops easier)

2..5	contains 2, 3, 4, 5.	It contains 5+1 – 2 = 4 values
2..4	contains 2, 3, 4.	It contains 4+1 – 2 = 4 values
2..3	contains 2, 3.	It contains 3+1 – 2 = 2 values
2..2	contains 2.	It contains 2+1 – 2 = 1 values
2..1	contains .	It contains 1+1 – 2 = 0 values

The number of values in **m..n** is **n**+1 – **m**: "follower minus first"

In the notation m..n, we require always, without saying it, that

m <= n + 1 (so, "2..1" is OK but not "2..0")

If m = n + 1, the range has 0 values.

Application: Some Personalized Email (SPEM)

Problem: how can we get people to read our mass email messages?

One answer: make it personal.

- Only one recipient
- Customized message ("*Hi Lisa, great seeing you at the talk yesterday*. Don't forget the meeting tomorrow"; "*Hail Batman. This course needs a better class of Criminal*. Don't forget the meeting tomorrow")
- We don't want to add duplicate recipients to the list (people notice and *hate* getting redundant emails).

This requires storing individualized information, iterating over the items we stored, and figuring out msg/mail output.

Some Personalized Email (SPEM): design decisions

How shall we represent a group of recipients (e.g., TAs vs. students)?

The usual design problem: how should we lay everything out?

We want the functionality of Vectors (so we can add recipients), … but we want to modify that functionality to be suitable for our purposes (no adds of duplicate recipients, ability to mail each recipient).

- new class **MailRecip** with appropriate equals method [remember last lecture?], personalization and mailing methods
- new class **MailGroup** extending Vector, using MailRecip's equals method to prevent addition of duplicates