cis3.5 spring2009 lecture II.2

topics:

- understanding processing
- "many eyes" preview

programming basics

- each line contains a *statement*
- statements end with a semi-colon (;)
- comments are contained within /** and */
- variables provide a way to save information within your sketch and use it to control the position, size, shape, etc of what you are drawing
  - variables have a *data type*, e.g., int, char
  - and a name
  - and a value

control structures: branching

```
if ( test ) {
  statements
}
```

```
if ( test ) {
  statements
}
else {
  statements
}
```


control structures: looping

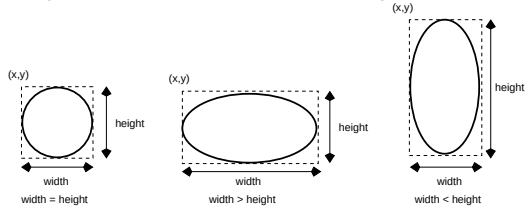
```
for ( init; test; update ) {
  statements
}
```

```
while ( expression ) {
  statements
}
```

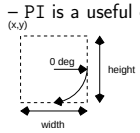
shapes

- `point( x, y )`
  - draws one point (looks like a dot...)
- `line( x1, y1, x2, y2 )`
  - connects two points
- `triangle( x1, y1, x2, y2, x3, y3 )`
  - connects three points
- `quad( x1, y1, x2, y2, x3, y3, x4, y4 )`
  - connects four points
- `rect( x, y, width, height )`
  - origin + extent; square if width=height

- `ellipse( x, y, width, height )`
  - origin: upper left corner of ellipse's *bounding box*; circle if width=height



- `arc( x, y, width, height, start, stop )`
  - origin: upper left corner of arc's bounding box (see ellipse)
  - `start` and `stop`: can be whole numbers (`int`) or real numbers (`float`); expressed in degrees or radians, depending on current angle mode; `0` is due east; measured clockwise
  - `PI` is a useful constant



attributes

- `strokeWeight()`
  - line thickness
- `strokeJoin()`
  - square (MITER, default), blunt (BEVEL), rounded (ROUND)
- `strokeCap()`
  - SQUARE, PROJECT, ROUND (default)

functions

- provide a way to *modularize* code
- makes it easier to read and re-use
- also allows you to specify content for functionality built in to Processing
- for example:
  `void draw() {...}`
  - `void` keyword that indicates a function which returns nothing
  - `draw()` = the name of the function
  - curly brackets (`{` and `}`) delineate the beginning and end of the function
  - with Processing, your sketch has to use no functions or all functions

animation

- use `setup()` function to specify things to do once, when the sketch first opens
- use `draw()` function to specify things to do repeatedly
- use `frameRate()` function to specify how often things should be repeated
  – default is 60 frames per second
  – call to `frameRate()` should be done inside `setup()` function

keyboard interaction

- `keyPressed()`
  – handles behavior when user presses a key down
- `keyReleased()`
  – handles behavior when user releases a key
- `keyTyped()`
  – handles behavior when user types a key (press and release)
- `key`
  – indicates which key was pressed/released/typed
  – equals `CODED` when special key is pressed/released/typed, like an arrow key, shift, control, alt, etc.
- `keyCode`
  – indicates special key: UP, DOWN, LEFT, RIGHT, ALT, CONTROL, SHIFT

mouse interaction

- `mousePressed()`
  – handles behavior when user presses mouse button
- `mouseReleased()`
  – handles behavior when user releases mouse button
- `mouseClicked()`
  – handles behavior when user clicks mouse button (press and release)
- `mouseMoved()`
  – handles behavior when user moves mouse (moves it without pressing button)
- `mouseDragged()`
  – handles behavior when user drags mouse (moves it with button pressed)
- `mouseButton`
  – indicates which button was pressed, on a multi-button mouse (not a Mac!)
- `mouseX` and `mouseY`
  – indicate (x, y) location of mouse pointer

data visualization

- it's not just your boring old excel plots any more (scatter, x-y, bar, pie)!
- "many eyes" preview
  `http://manyeyes.alphaworks.ibm.com/manyeyes/`
- Ben Fry
  `http://benfry.com/salaryper/`
  `http://benfry.com/projects/`
- Edward Tufte
  `http://www.edwardtufte.com/tufte/`