cis3.5 fall2009 lecture II.2

topics:

- more about programming with "Processing"

resources:

- Processing web site: `http://www.processing.org/`
- linear motion: `http://www.processing.org/learning/topics/linear.html`
- sequential animation: `http://www.processing.org/learning/topics/sequential.html`
- reference: `http://www.processing.org/reference/index.html`

variables

- 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*, a name and a value
- valid data types are:
  `int` — for storing integers (whole numbers)
  `float` — for storing floating point (real) numbers
  `boolean` — for storing `true` or `false` values
  `char` — for storing single characters
  `String` — for storing multiple (strings of) characters
- example:

```
int x1 = 10;
int y1 = 10;
int x2 = 20;
int y2 = 20;
line( x1, y1, x2, y2 );
```


looping

- loops are used for doing things repeatedly
- there are two basic types of loops:
  - **for** loops
  - **while** loops
- loops are handy for animation, because you typically want to display things repeatedly when you are doing animation

for loops

- *for* loops repeat things for a fixed number of times
- syntax:

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

- example:

```
int x = 10;
int y1 = 10;
int y2 = 20;
for ( int i=0; i<10; i++ ) {
  line( x, y1, x, y2 );
  x = x + 10;
}
```

while loops

- *while* loops repeat things as long as a condition holds true
- syntax:

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

- example:

```
int x = 10;
int y1 = 30;
int y2 = 40;
while ( x < width ) {
  line( x, y1, x, y2 );
  x = x + 10;
}
```


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
- basic animation involves the following steps:
  1. drawing initial frame
  2. waiting some amount of time (e.g., $1/60$th of a second)—Processing does that automatically
  3. erasing the background and drawing the next frame
  4. and doing the previous step repeatedly, until you are ready to stop animating
- there are two basic ways to implement animation in Processing:
  - drawing your own shapes, text, etc.
  - displaying a GIF or other image file

- example of drawing your own shapes (from `http://www.processing.org/learning/topics/linear.html`):

```
float a = 100;

void setup() {
  size( 640, 200 );
  stroke( 255 );
}

void draw() {
  background( 51 );
  a = a - 0.5;
  if ( a < 0 ) {
    a = height;
  }
  line( 0, a, width, a );
}
```


- example of displaying an image (from `http://www.processing.org/learning/topics/sequential.html`):

```
int numFrames = 5;  // The number of frames in the animation
int frame = 0;
PImage[] images = new PImage[numFrames];

void setup() {
  size( 200, 200 );
  frameRate( 30 );
  images[0]  = loadImage("PT_anim0000.gif");
  images[1]  = loadImage("PT_anim0001.gif");
  images[2]  = loadImage("PT_anim0002.gif");
  images[3]  = loadImage("PT_anim0003.gif");
  images[4]  = loadImage("PT_anim0004.gif");
}

void draw() {
  frame = ( frame + 1 ) % numFrames;  // Use % to cycle through frames
  image( images[frame], 50, 50 );
}
```

mouse interaction

- `mouseX` and `mouseY`
  – indicate (x, y) location of mouse pointer
- `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 (on a Mac, use Cntl-click for left mouse button, Alt-click for middle mouse button and Apple-click for right mouse button)

- mouse location example:

```
void setup() {
  size( 200, 200 );
}

void draw() {
  background( #cccccc );
  fill( #000099 );
  rect( mouseX, mouseY, 20, 20 );
}
```


- mouseMoved example:

```
void setup() {
  size( 200, 200 );
}

void draw() {
  background( #cccccc );
  fill( #990000 );
  rect( mouseX, mouseY, 20, 20 );
}

void mouseMoved() {
  fill( #000099 );
  rect( mouseX, mouseY, 20, 20 );
}
```

- how does this behave differently from the mouse location example?

- mouseDragged example:

```
void setup() {
  size( 200, 200 );
}

void draw() {
  background( #cccccc );
  fill( #990000 );
  rect( mouseX, mouseY, 20, 20 );
}

void mouseMoved() {
  fill( #000099 );
  rect( mouseX, mouseY, 20, 20 );
}

void mouseDragged() {
  fill( #009900 );
  rect( mouseX, mouseY, 20, 20 );
}
```

- how does this behave differently from the previous two examples?

- mouseClicked example:

```
int r = 0;
int g = 0;
int b = 0;

void setup() {
  size( 200, 200 );
}

void draw() {
  background( #ffffff );
  fill( r, g, b );
  rect( 50, 50, 20, 20 );
}

void mouseClicked() {
  r = r + 51;
  if ( r > 255 ) {
    r = 0;
    g = g + 51;
    if ( g > 255 ) {
      g = 0;
      b = b + 51;
      if ( b > 255 ) {
        b = 0;
      }
    }
  }
  println( "r=" + r + " g=" + g + " b=" + b );
}
```


- mouseButton example:

```
void setup() {
  size( 200, 200 );
}

void draw() {
  background( #cccccc );
  rect( mouseX, mouseY, 20, 20 );
}

void mousePressed() {
  if ( mouseButton == LEFT ) {
    fill( #990000 );
  }
  else if ( mouseButton == CENTER ) {
    fill( #009900 );
  }
  else if ( mouseButton == RIGHT ) { // Ctrl-click on mac
    fill( #000099 );
  }
}
```