CIS3.5 Spring 2010 Lecture II.2

More 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
- looping is a type of:
  - repetition (required element of imperative programming)
  - iteration (same thing as repetition)

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;
}
```

Standard Processing Program

1. Setup any variables or classes you are going to use.
2. Use setup() function to specify things to do once, when the sketch first opens
3. Use draw() function to specify things to do repeatedly
    - use frameRate() function to specify how often things should be repeated in draw();
    - default frame-rate is 60 (60 frames per second)
    - NOTE: call to frameRate() should be done inside setup() function
4. Declare and event-listeners that you are going to use.
5. Declare any custom made functions you are going to use.
6. Declare any classes that you are going to use.

Note: I have created a processing template that you can use to start your programs.

Animation

<u>Basic animation involves the following steps:</u>

1. Drawing initial frame - perhaps in setup().
2. Waiting some amount of time (e.g., 1/60th of a second)
    - Processing does that automatically
3. Erasing the screen.
    - Usually be reapplying the background (draw does this automatically).
4. Drawing the next frame.
5. Repeating steps 2-4, until you are ready to stop animating.

<u>There are two basic ways to implement animation:</u>

1. Drawing your own shapes, text, etc.
2. Displaying a GIF or other image file

Vector Animation (drawing 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 );
}
```

Bitmap Animation (using pictures)

http://www.processing.org/learning/topics/sequential.html

```
int numFrames = 4; // 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");
}
void draw() {
      frame = ( frame + 1 ) % numFrames; // Use % to cycle through frames
      image( images[frame], 50, 50 );
}
```

Movement and Animation

```
int xPos = 0;
int yPos = 50;
    ....
void draw() {
  xPos = (xPos + 2) % width;
  frame = ( frame + 1 ) % numFrames; // Use % to cycle through frames
  image( images[frame], xPos, yPos );
}
    ...
void keyPressed() {
  if (key == CODED) {
    if (keyCode == UP) {
      yPos = yPos - 2;
    } else if (keyCode == DOWN) {
      yPos = yPos + 2;
    }
  }
}
```

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)

Example 1 (mouse location)

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

void draw() {
    background( #cccccc );
    // What happens if you remove the line above?
    fill( #000099 );
    rect( mouseX, mouseY, 20, 20 );
}
```

Example 2 (mouseMoved)

```
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? */
```

Example 3 (mouseDragged)

```
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? */
```

Example #4 (mouseClicked)

```
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 );
}
```

Example #5 (mouseButton)

```
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 );
      }
}
```