Review

• What is Computing?
• Occupations in CS?
• What can be Programmed?
• Creative Computing
• Processing
• Downloading Processing
• Dropbox
• Sketchpad
• Assignment #1

• Primitive Shapes
  – point
  – line
  – triangle
  – quad
  – rect
  – ellipse
• Processing Canvas
• Coordinate System
• Shape Formatting
  – Colors
  – Stroke
  – Fill
Comments

• Used to explain your source code
• Ignored by Processing

/* This is a comment that spans multiple lines */

// This is a comment that is restricted to a single line

line(0, 0, 10, 10); // Can start anywhere, continue to line end

Note the color of the various items in the processing editor.
random(high);
random(low, high);

Generate a random number in the range
low (or 0) to high

print(something);
println(something);

Print something to the Processing console.
mouseX
mouseY
    Built-in predefined variables that hold the current mouse X and Y locations.

key
    Always contains the value of the most recent key pressed on the keyboard.

keyCode
    Always contains a number that codes for the most recent key pressed, even keys that cannot be printed.
void setup()
{
    // Called once when program starts
}

void draw()
{
    /* Called repeatedly
        while program runs */
}
randomEllipse

void setup()
{
    size(300, 300);
    smooth();
}

void draw()
{
    fill(random(255), random(255), random(255));
    ellipse(mouseX, mouseY, 30, 30);
}
Controlling draw()

`frameRate(fps);`
  Sets number of frames displayed per second. i.e. the number of times draw() is called per second. Default = 60.

`noLoop();`
  Stops continuously calling draw().

`loop();`
  Resumes calling draw().
void mousePressed() {
    // Called when the mouse is pressed
}

void mouseReleased() {
    // Called when the mouse is released
}

void mouseClicked() {
    // Called when the mouse is pressed and released
    // at the same mouse position
}

void mouseMoved() {
    // Called while the mouse is being moved
    // with the mouse button released
}

void mouseDragged() {
    // Called while the mouse is being moved
    // with the mouse button pressed
}
void keyPressed() {
    // Called each time a key is pressed
}

void keyReleased() {
    // Called each time a key is released
}

void keyTyped() {
    // Called when an alpha-numeric key is pressed
    // Called repeatedly if the key is held down
}
**keyCode vs. key**

**key**
- A built-in variable that holds the character that was just typed at the keyboard

**keyCode**
- A built-in variable that holds the numeric code for the keyboard key that was touched

All built-in keyboard interaction functions …
- Set `keyCode` to the integer that codes for the keyboard key
- Set `key` to the character typed
- All keyboard keys have a `keyCode` value
- Not all have a `key` value
### ASCII - American Standard Code for Information Interchange

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>30</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>'</td>
<td>]</td>
<td>]</td>
</tr>
<tr>
<td>40</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>,</td>
<td>-</td>
<td>.</td>
<td>/</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>60</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>70</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>80</td>
<td>N</td>
<td>O</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td>90</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>]</td>
<td></td>
<td>^</td>
<td>_</td>
<td>\</td>
<td>a</td>
</tr>
<tr>
<td>100</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
</tr>
<tr>
<td>110</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>o</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
</tr>
<tr>
<td>120</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td>}</td>
</tr>
<tr>
<td>130</td>
<td>~</td>
<td>•</td>
<td>€</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>140</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>150</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>160</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>170</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>180</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>190</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>210</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>220</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>230</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>240</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>250</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
More Graphics

arc(…)
curve (…)
bézier(...)
shape(...)
Arcs

```
arc( x, y, width, height, start, stop );
```

*An arc is a section of an ellipse*

- `x, y, width, height`: location and size of the ellipse
- `start, stop`: arc bounding angles (in radians)
Arcs

arc( x, y, width, height, start, stop );
Spline Curves

curve( x1, y1, x2, y2, x3, y3, x4, y4 );

_Spline_: A smooth line drawn through a series of points

A _curve_ is a Catmull-Rom (cubic Hermite) spline defined by four points

x2, y2 and x3, y3

beginning/end points of visual part of curve

x1, y1 and x4, y4

control points that define curve curvature
Spline Curves

curve( x1, y1, x2, y2, x3, y3, x4, y4 );

curveEditor.pde
Bézier Curves

bezier( x1, y1, cx1, cy1, cx2, cy2, x2, y2 );

A smooth curve defined by two anchor points and two control points

x2, y2 and x2, y2
anchor points of bézier curve
cx1, cy1 and cx2, cy2
control points that define curvature
Bézier Curves

```plaintext
beziers( x1, y1, cx1, cy1, cx2, cy2, x2, y2 );
```
Custom Shapes

• Composed of a series of vertexes (points)
• Vertexes may or may not be connected with lines
• Lines may join at vertexes in a variety of manners
• Lines may be straight, curved, or bézier splines
• Shapes may be closed or open
Custom Shapes

\[
\text{beginShape( [option] );}
\]

\[
\text{vertex( } x, y \text{ );}
\]

\[
\text{curveVertex( } x, y \text{ );}
\]

\[
\text{bezierVertex( cx1, cy1, cx2, cy2, x, y );}
\]

\[
\text{endShape( [CLOSE] );}
\]
strokeJoin()
Example Sketches...

- LadyBug1
- Monster1
- Ndebele
- Penguin1
- SouthParkCharacter1
- Sushi
- GiorgioMorandi
OpenProcessing

http://www.openprocessing.org/

– Bryn Mawr and SMU student sketches