Obamicon

Drawing Primitives

• \texttt{point}(x, y);
• \texttt{line}(x_1, y_1, x_2, y_2);
• \texttt{triangle}(x_1, y_1, x_2, y_2, x_3, y_3);
• \texttt{rect}(x, y, \text{width}, \text{height});
• \texttt{ellipse}(x, y, \text{width}, \text{height});

Modes

• \texttt{rect}(x, y, \text{width}, \text{height});
• \texttt{ellipse}(x, y, \text{width}, \text{height});
• \texttt{rectMode} (\texttt{CENTER});
• \texttt{ellipseMode} (\texttt{CORNER});

Programming Principle

• Syntax is important!

Odds and Ends

• Processing programs carry the extension .pde
• must be in a folder with the same name
  – \texttt{myProgram.pde} must be inside a folder called \texttt{myProgram}
• Code block
  – The curly braces {}
• Comments
  – //
  – /* and */
• Naming convention

Review

• What is Computing?
• What can be Programmed?
• Creative Computing
• Processing
• Downloading Processing
• Dropbox
• Primitive Shapes
  – point
  – line
  – triangle
  – quad
  – rect
  – ellipse
• Processing Canvas
• Coordinate System
• Shape Formatting
  – Colors
  – Stroke
  – Fill

Review

• point\( (x, y); \)
• line\( (x_1, y_1, x_2, y_2); \)
• triangle\( (x_1, y_1, x_2, y_2, x_3, y_3); \)
• rect\( (x, y, \text{width}, \text{height}); \)
• ellipse\( (x, y, \text{width}, \text{height}); \)

Modes

• rect\( (x, y, \text{width}, \text{height}); \)
• ellipse\( (x, y, \text{width}, \text{height}); \)
• rectMode\( (\text{CENTER}); \)
• ellipseMode\( (\text{CORNER}); \)

Programming Principle

• Syntax is important!

Odds and Ends

• Processing programs carry the extension .pde
• must be in a folder with the same name
  – \texttt{myProgram.pde} must be inside a folder called \texttt{myProgram}
• Code block
  – The curly braces {}
• Comments
  – //
  – /* and */
• Naming convention
Basic Processing Program

```java
void setup() {
    // Called once when program starts
}

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

The Event Loop

- Any code in `draw()` is executed 60 times per second
- Put code that you only want executed once in `setup()`
  - defaults
- `noLoop()`
- `loop()`

Mouse Interaction

- Built-in predefined variables that hold the mouse X and Y locations
  - current `mouseX` `mouseY`
  - previous (last) `pmouseX` `pmouseY`
  - 0 if mouse is not in window

More Graphics Primitives

```java
arc(...)  
curve(...)  
bezier(...)  
shape(...)  
```

Arcs

```java
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)
Spline Curves

\[ \text{curve}(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4); \]

\textit{spline}: A smooth curve drawn defined by four points

- \( x_2, y_2 \) and \( x_3, y_3 \)  
  beginning/end points of visual part of curve
- \( x_1, y_1 \) and \( x_4, y_4 \)  
  control points that define curve curvature

\[ \text{curveEditor}(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4); \]

Bézier Curves

\[ \text{bezier}(x_1, y_1, cx_1, cy_1, cx_2, cy_2, x_2, y_2); \]

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

- \( x_1, y_1 \) and \( x_2, y_2 \)  
  anchor points of bézier curve
- \( cx_1, cy_1 \) and \( cx_2, cy_2 \)  
  control points that define curve curvature

\[ \text{bezierEditor}(x_1, y_1, cx_1, cy_1, cx_2, cy_2, x_2, y_2); \]

Custom Shapes

- Composed of a series of vertexes (points)
  - Vertices may or may not be connected with lines
  - Lines may join at vertexes in a variety of manners
  - Lines may be straight, curves, or bézier splines
- Shape may be closed or open

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

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

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

\[ \text{bezierVertex}(cx_1, cy_1, cx_2, cy_2, x, y); \]

\[ \text{endShape}([\text{CLOSE}]); \]

\[ \text{Custom Shapes}; \]
void keyPressed() {
  // Called each time a key is pressed
}

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

void keyTyped() {
  // Called when a 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 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

Example Sketches...
  – LadyBug
  – Monster
  – Ndebele
  – Penguin
  – SouthParkCharacter
  – Sushi
  – GiorgioMorandi