Processing: Basic Shapes

Elements of Graphics
CS324e
Processing Language

- Java-based syntax for achieving graphics functionality
- Incorporates usual programming language features:
  - Functions
  - Comments
  - Expressions
Primitive Data Types

- boolean
- byte
- char
- int
- float
- color
Example Processing Setup

```java
void setup() {
    size(200, 200);
}
```
void draw() {
    background(102);
}

- Code inside `setup()` runs once
- Code inside `draw()` runs as a continuous loop

**Draw Loop**
Variable scope

- Variables declared within a block are local to that block
- Global variables are declared outside of all blocks
- What is the relationship between global variables, setup() and draw()?
Consider...

```cpp
int x = 0;
void setup() {
    x += 3;
}

void draw() {
    x++;
}
```
Coordinate Systems

- Coordinate systems define the “space” of the scene within the computer
- Common coordinate systems:
  - World coordinate system
  - Object coordinate system
  - Camera coordinate system
  - Screen coordinate system
- Multiple coordinate systems allow for multiple levels of interaction
  - Multiple coordinate systems also require conversion between systems
Screen Coordinate System

- 2-D, pixel-based coordinate system
- Based on the size (resolution) of the screen/window
- Pixel position defined using \((x, y)\) coordinate notion
Defining Geometry in Processing

- Function `point(x, y)` defines a pixel within the window
- Function `line(x1, y1, x2, y2)` defines a line of pixels between `(x1, y1)` and `(x2, y2)`

```
point(x, y)
```

```
line(x1, y1, x2, y2)
```
Shape Primitives

❖ Other shape primitives in Processing:

❖ `rect(a, b, c, d)`
❖ `ellipse(a, b, c, d)`
❖ `triangle(x1, y1, x2, y2, x3, y3)`
❖ `quad(x1, y1, x2, y2, x3, y3, x4, y4)`

triangle(x1, y1, x2, y2, x3, y3)  quad(x1, y1, x2, y2, x3, y3, x4, y4)  ellipse(a, b, c, d)
Curves

- \texttt{arc(a,b,c,d,start,stop)}
- \texttt{bezier(x1,y1,x2,y2,x3,y3,x4,y4)}
• **arc** models elliptical arcs

• **arc** expects radians (0 to $2\pi$) rather than degrees (0 to 360) by default

• **bezier** models cubic Bezier curves

• Bezier curves are
  • Smooth
  • Scalable
  • Parametric

• **bezierVertex** can model higher order Bezier curves

  • We will come back to this concept later in the semester
Hands-on: Creating Geometry

❖ Today’s activities:

1. Create a Processing sketch
2. Use the point, line, rect, ellipse, triangle, and quad methods at least two times each
3. Create at least one shape with the arc method
4. Create at least one shape with the bezier method

• Consider: What makes bezier challenging to work directly with in code?