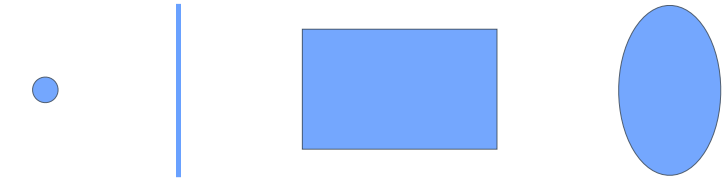
Pixels and Processing

ISE 108: Introduction to Programming
Stony Brook University

Outline of Topics

- Pixels and Coordinate Systems
- Drawing Simple Shapes
- Introduction to the Processing Environment
- Color

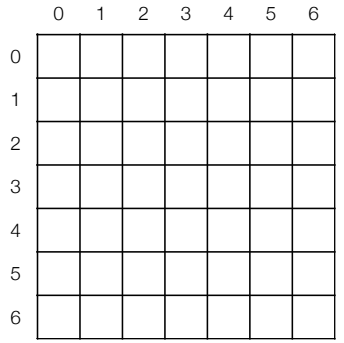
Primitive Shapes



Point Line Rectangle Ellipse

Coordinates

- The screen is like a piece of graph paper
  - Each cell is a *pixel* (“picture element”)
- The *origin* (0, 0) is at the *top left*
  - x-axis: + is to the right, - is to the left
  - y-axis: + is down, - is up



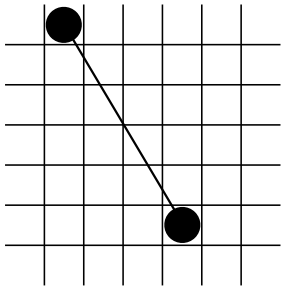
Points

- Use the `point()` command to draw a single pixel
- `point()` takes two integers as input values:
  - The x and y coordinates of the pixel
- `point()` draws a point at the indicated location in the current drawing color
- Use a semicolon to end the command
  - e.g., `point(3, 4);` draws a point at x: 3 and y: 4

Drawing Lines

- To draw a line, use the `line()` command
- Tell `line()` where the line should begin and end
- Format:

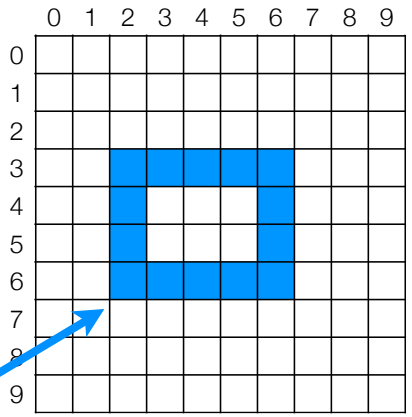
```
line(start_x, start_y,
     end_x, end_y);
```



```
line(1,0,4,5);
```

Rectangles

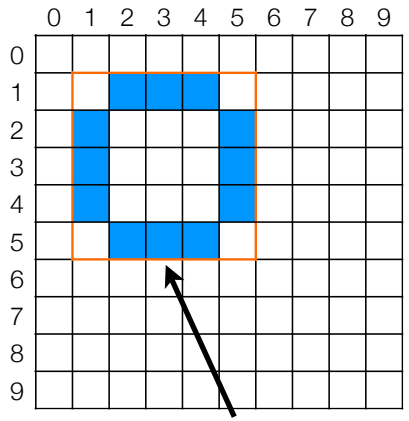
- To draw a rectangle, you need to know:
  1. the coordinates of its top-left corner
  2. its width (in pixels)
  3. its height (in pixels)



rect (2, 3, 5, 4);

Ellipses

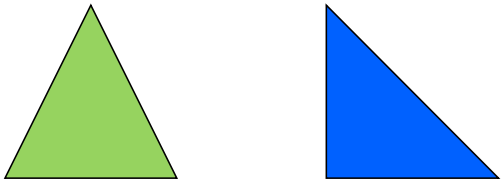
- Ellipses are drawn in the center of a *bounding box* (a rectangle that encloses all of the ellipse's points)
- To draw an ellipse, you need to know:
  1. its center coordinates
  2. its width in pixels
  3. its height in pixels



ellipse (3, 3, 5, 5);

Triangles

- Use the `triangle()` function to draw a triangle on the screen
- `triangle()` takes six values: the x and y coordinates of each vertex
  - ex. `triangle (100, 50, 75, 100, 125, 100);`



Canvas Size

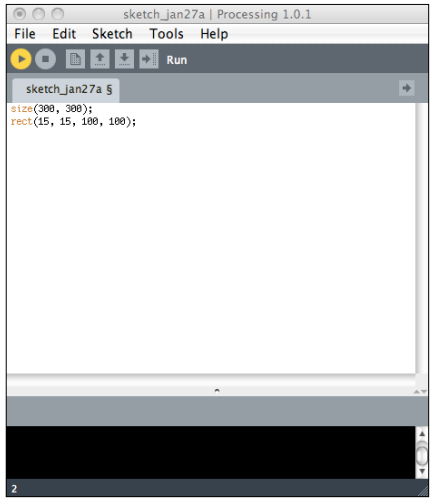
- By default, Processing only gives you a small amount of drawing space (known as the “canvas”)
- The `size()` command tells Processing how much drawing space you want to use

  `size (`*width_in_pixels* `,` *height_in_pixels*`);`

- You must set the canvas size ***before*** you draw anything!
  - This will normally be the first Processing command in any program you write

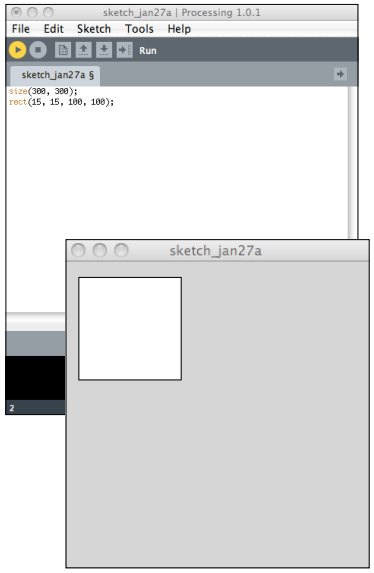
The Processing Sketchpad

- Use the Sketchpad to write Processing code



The Processing Sketchpad

- To run your code, click the Play button
  - A new window will display the result



Adding Color

- By default, Processing draws black lines and white objects on a gray background
- We can change the color of the background (canvas)
- We can also change the drawing (pen) color, and the fill color

Color Types

- Color comes in two types: grayscale and RGB
- Grayscale is indicated by an integer between 0 (pure black) and 255 (pure white)
- RGB (Red-Green-Blue) uses three integers between 0 and 255 (one for each color component)
  - e.g., 255 red, 125 green, 8 blue = pumpkin orange
- You can also use Processing's Color Selector (in the Tools menu)

Grayscale and RGB

Grayscale Value		Color	Red	Green	Blue		Color
40	=		30	30	30	=	
80	=		100	30	30	=	
120	=		50	200	50	=	
160	=		50	50	150	=	
200	=		120	100	75	=	
240	=		175	50	255	=	

Canvas Color

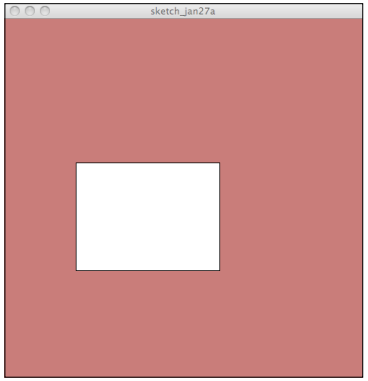
- Use the `background()` command to set the color of the canvas
- In the parentheses, put one integer for grayscale color or three (with commas) for RGB color
  - e.g., `background (50);`
  - e.g., `background (255, 0, 255);`

My First Program

```
size(500, 500);

background(200, 100, 100);

rect(100, 200, 200, 150);
```



Other Color Changes

- `stroke()` changes line color
  - e.g., `stroke(255, 0, 0);`
- `fill()` changes the color inside a shape
  - e.g., `fill(153);`

- Note 1: do this **BEFORE** drawing a shape!
- Note 2: Changing the stroke or fill color affects every line or shape you draw from that point on (at least, until the next color change)

Comments

- Comments are notes that describe how a program works
  - They are only useful for humans; the computer ignores them
- To add a comment, insert two consecutive forward slashes (//) in front of the comment text
- The comment extends from the slashes until the end of the line
  - e.g., `// This is a comment`