Drawing in Java Using the StdDraw Library: MyHouse.java

CIS 110

Penn Engineering

Explanatory Comment

```
10 public class MyMouse {
11     public static void main(String[] args) {
12         //set the size of the window to 500 pixels by 500 pixels
13         StdDraw.setCanvasSize(500, 500);
14
15         StdDraw.clear(StdDraw.BLUE); // draw a blue sky
```

Set Window Size

```
10 public class MyHouse {
11     public static void main(String[] args) {
12         //set the size of the window to 500 pixels by 500 pixels
13         StdDraw.setCanvasSize(500, 500);
14
15         StdDraw.clear(StdDraw.BLUE); // draw a blue sky
```

Color the entire window blue

```
10 public class MyHouse {
11     public static void main(String[] args) {
12         //set the size of the window to 500 pixels by 500 pixels
13         StdDraw.setCanvasSize(500, 500);
14
15         StdDraw.clear(StdDraw.BLUE); // draw a blue sky
```

Comment indicates *purpose*

Can replace BLUE with BLACK, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, or YELLOW

Color the entire window blue

```
10 public class MyHouse {
11     public static void main(String[] args) {
12         //set the size of the window to 500 pixels by 500 pixels
13         StdDraw.setCanvasSize(500, 500);
14
15         StdDraw.clear(StdDraw.BLUE); // draw a blue sky
```

Comment indicates *purpose*

Set the color to grass green

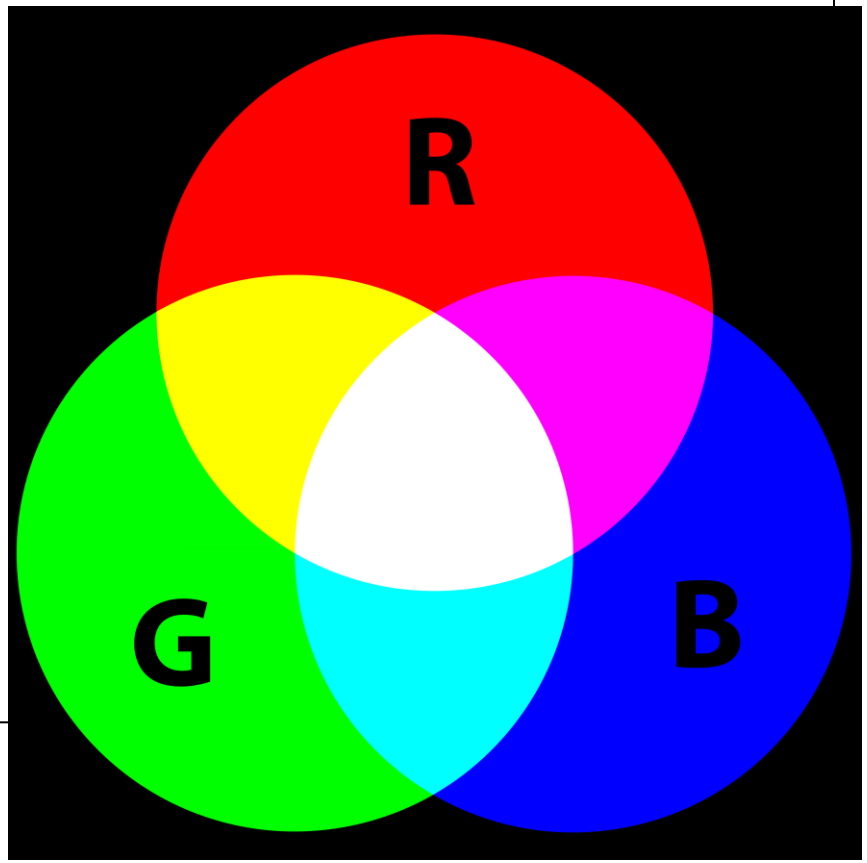
```
17      //draw a green field
18      StdDraw.setPenColor(0, 170, 0);
19      StdDraw.filledRectangle(0.5, 0.25, 0.6, 0.3);
```

Colors

Composed of three elements:

1. Red
2. Green
3. Blue

Values from 0 .. 255



Set the color to grass green

```
17      //draw a green field
18      StdDraw.setPenColor(0, 170, 0);
19      StdDraw.filledRectangle(0.5, 0.25, 0.6, 0.3);
```

Solid rectange

```
//draw green field
StdDraw.setPenColor(0, 170, 0);
StdDraw.filledRectangle(0.5, 0.25, 0.6, 0.3);
```

y center

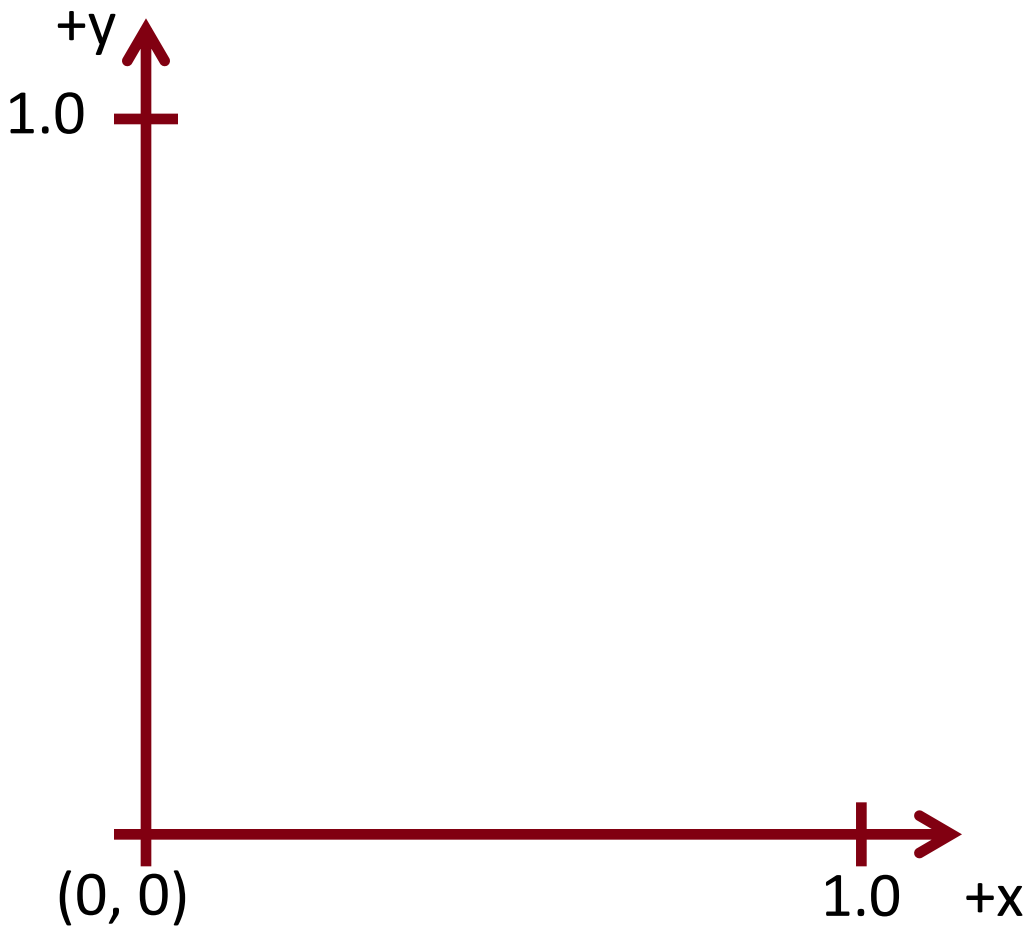
half height

```
//draw a green field
StdDraw.setPenColor(0, 170, 0);
StdDraw.filledRectangle(0.5, 0.25, 0.6, 0.3);
```

x center

half width

Coordinate System



Lists of 3 x- and y-coordinates

```
double[] x = {0.255, 0.745, 0.49};
double[] y = {0.70, 0.70, 0.90};
StdDraw.filledPolygon(x, y);
```

Draw a solid triangle with corners at (0.255, 0.7), (0.745, 0.7), (0.49, 0.9)

```
double[] x = {0.255, 0.745, 0.49};
double[] y = {0.70, 0.70, 0.90};
StdDraw.filledPolygon(x, y);
```

Set line thickness (default is 0.002)

```
26        StdDraw.setPenRadius(0.005); // thicken the pen for outline drawing
```

Draw a rectangle outline

```
34        StdDraw.rectangle(250 / 500.0, 260 / 500.0, 120 / 500.0, 90 / 500.0);
```

Keep repeating the instructions in this block forever

```
41        // draw a circular cloud at the mouse location as long
42        // as the mouse is within bounds
43        while (true) {
44            double cloudX = StdDraw.mouseX();
45            double cloudY = StdDraw.mouseY();
46            StdDraw.setPenColor(StdDraw.WHITE);
47            if (cloudY > 0.55) {
48                StdDraw.filledCircle(cloudX, cloudY, 0.005);
49            }
50            StdDraw.show(30);
51        }
```

Check the mouse's x- and y-coordinates. Call them cloudX and cloudY.

```
41        // draw a circular cloud at the mouse location as long
42        // as the mouse is within bounds
43        while (true) {
44            double cloudX = StdDraw.mouseX();
45            double cloudY = StdDraw.mouseY();
46            StdDraw.setPenColor(StdDraw.WHITE);
47            if (cloudY > 0.55) {
48                StdDraw.filledCircle(cloudX, cloudY, 0.005);
49            }
50            StdDraw.show(30);
51        }
```

Draw a circle centered at the cursor with radius 0.005, *only if the y-coordinate is greater than 0.55!*

```
41          // draw a circular cloud at the mouse location as long
42          // as the mouse is within bounds
43          while (true) {
44              double cloudX = StdDraw.mouseX();
45              double cloudY = StdDraw.mouseY();
46              StdDraw.setPenColor(StdDraw.WHITE);
47              if (cloudY > 0.55) {
48                  StdDraw.filledCircle(cloudX, cloudY, 0.005);
49              }
50              StdDraw.show(30);
51          }
```

Show the changes we just made; Wait to show any further changes until we encounter StdDraw.show() again and *at least 30 milliseconds have past.*

```
41          // draw a circular cloud at the mouse location as long
42          // as the mouse is within bounds
43          while (true) {
44              double cloudX = StdDraw.mouseX();
45              double cloudY = StdDraw.mouseY();
46              StdDraw.setPenColor(StdDraw.WHITE);
47              if (cloudY > 0.55) {
48                  StdDraw.filledCircle(cloudX, cloudY, 0.005);
49              }
50              StdDraw.show(30);
51          }
```

StdDraw.show() controls the animation speed, or "frame rate."

Keyboard input

- StdDraw.hasNextKeyTyped() – check to see if the user has pressed key
- If the user presses a key, StdDraw.hasNextKeyTyped() is true until and unless you write a line that processes the input

- c = StdDraw.nextKeyTyped();

```
public class KeyBoardInput {
    public static void main(String[] args) {
        char c = 0;
        double radius = 1 / 500.0;
        StdDraw.setCanvasSize(600, 600);
        while (c != 'q') {
            if (StdDraw.hasNextKeyTyped()) {
                c = StdDraw.nextKeyTyped();
            }
            StdDraw.circle(0.5, 0.5,
                                radius);
            radius = radius + 1 / 500.0;
            StdDraw.show(10);
        }
    }
}
```

Penn Engineering

Using StdDraw.show for animation

- StdDraw.show()
  - Display on-screen and turn off animation mode:
  - subsequent calls to drawing methods such as line(), circle(), and square() will be displayed on screen when called
- StdDraw.show(t)
  - Display on screen, pause for t milliseconds, and turn on *animation mode*:
  - subsequent calls to drawing methods such as line(), circle(), and square() will not be displayed on screen until the next call to show().