CS324e - Elements of Graphics and Visualization

Java2D Graphics

Java2D Graphics

- Render
  - Heating animal remains to extract fat
  - generating an image from a (mathematical) model
- Java2D can handle primitive shapes, text, images
- The Graphics object is an abstraction used to perform the rendering

Graphics Object Properties

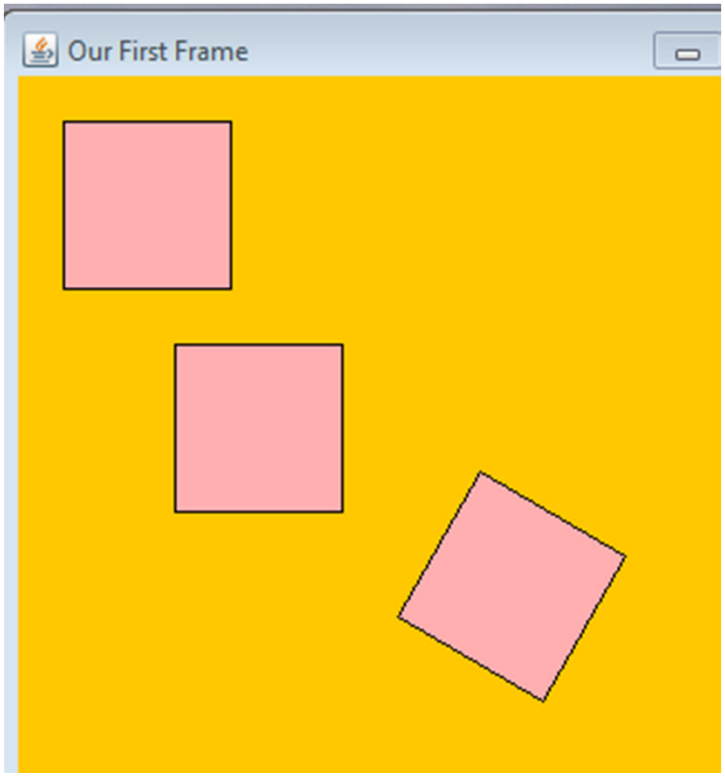
- foreground color - setColor(Color c)
- background color - setBackground(Color c)
- font - setFont(Font f)
- stroke - how wide is brush plus end joints - setStroke(Stroke s)
- rendering hints - for example anti-aliasing (demo bugs)
- clip - region to which drawing is constrained

Graphics Object Properties

- Composite - what method to use to combine color data from a new object being drawn to the objects already on the screen
- Paint - similar to color, allows the creation of gradients
- Transform - size, position, and orientation of drawing primitives
  - translate, rotate, scale, shear

Transforms

- Methods to apply an *affine transformation*
  - affine transformations alter the coordinates of an object but preserve straight lines and proportions between points on a straight line



Examples

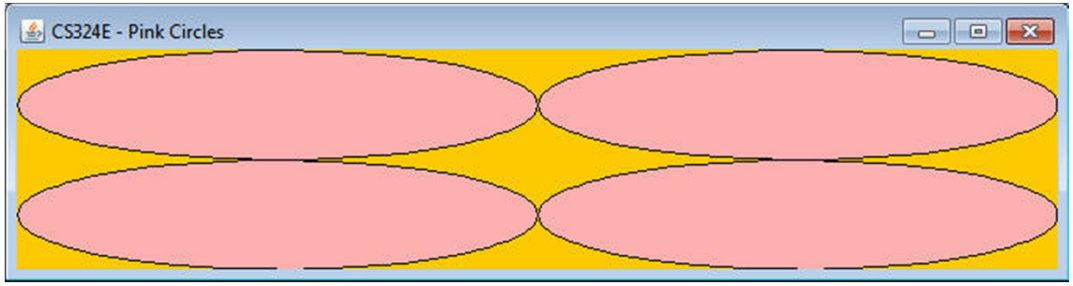
- Translate
- Rotate
  - translate after a rotation
- Scale
- Shear
  - If you specify a non-zero x shear, then x values will be more and more shifted to the right the farther they are away from the **y** axis. For example, an x shear of 0.1 means that the x value will be shifted 10% of the distance the point is away from the y axis. Y shears are similar: points are shifted down in proportion to the distance they are away from the **x** axis.
  - http://www.apl.jhu.edu/~hall/java/Java2D-Tutorial.html#Java2D-Tutorial-Transforms-Shear

Graphics Primitives

- the Graphics2D fill and draw methods accept parameters of type Shape
  - polymorphism
- Built in Shapes
  - Arc2D
  - Ellipse2D
  - Line2d
  - Polygon
  - Rectangle2D
  - and more!

Ellipse2D

- Specify x and y of upper corner of bounding box, width and height
- abstract class
- nested classes
  - Ellipse2D.Double
  - Ellipse2D.Float



Arc2D

- Multiple constructors
- ellipse bounds, start angle, extent angle, type
  - angles in degrees
  - types: CHORD, OPEN, PIE
  - "The angles are specified relative to the non-square framing rectangle such that 45 degrees always falls on the line from the center of the ellipse to the upper right corner of the framing rectangle."

Arc2D

CHORD PIE OPEN

start = 30, extent = 110

Fonts

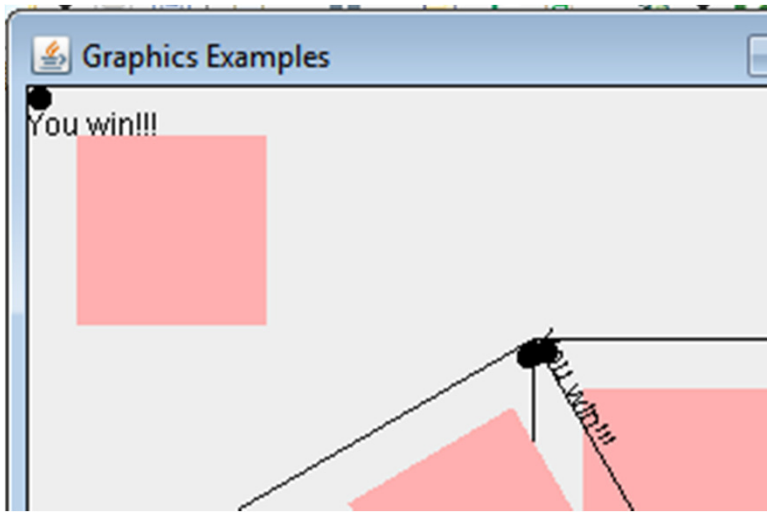
- Recall the drawString method from the Graphics class:

drawString

```
public abstract void drawString(String str,
                                int x,
                                int y)
```

Renders the text of the specified `String`, using the current text attribute state in the `Graphics2D` context. The baseline of the first character is at position (*x*, *y*) in the User Space. The rendering

- Property of the graphics object is the current font



Fonts

- Font can be changed

setFont

```
public abstract void setFont(Font font)
```

Sets this graphics context's font to the specified font. All subsequent text operations using this graphics context use this font. A null argument is silently ignored.

- Font constructors

Constructor Detail

Font

```
public Font(String name,
    int style,
    int size)
```

Creates a new `Font` from the specified name, style and point size.

The font name can be a font face name or a font family name. It is used together with the

Font Properties

- Font family names:
  - DIALOG, DIALOG_INPUT, MONOSPACED, SANS_SERIF, SERIF
- Styles:
  - BOLD, ITALIC, PLAIN
- Size: point size

Font Example

```
g2.setFont(new Font(Font.SANS_SERIF, Font.BOLD | Font.ITALIC, 50));
g2.setColor(Color.BLACK);
g2.drawString("You win!!!", 0, 50);
```

- | operator

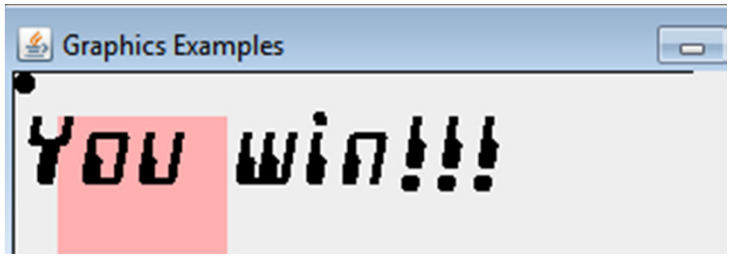
```
g2.setColor(Color.BLACK);
g2.setFont(new Font(Font.SERIF,
        Font.BOLD | Font.ITALIC, 50));
g2.drawString("You win!!!", 0, 20);
```

Checking Available Fonts

```
public static void main(String[] args) {
    GraphicsEnvironment ge
            = GraphicsEnvironment.getLocalGraphicsEnvironment();
    String[] fontFamilies = ge.getAvailableFontFamilyNames();
    System.out.println(fontFamilies.length);
    for(String fontName : fontFamilies)
        System.out.println(fontName);
```

- Last time I ran it, 302 fonts
- Handles nonexistent fonts gracefully

```
g2.setFont(new Font("Future",
        Font.BOLD | Font.ITALIC, 50));
g2.setColor(Color.BLACK);
g2.drawString("You win!!!", 0, 50);
g2.setColor(Color.PINK);
```



Stroke

- line width
- end caps of lines
- join style for multi-segment lines

The *join style* is the decoration that is applied where two line segments meet.
`BasicStroke` supports the following three join styles:


`JOIN_BEVEL`


`JOIN_MITER`


`JOIN_ROUND`

The *end-cap style* is the decoration that is applied where a line segment ends.
`BasicStroke` supports the following three end-cap styles:


`CAP_BUTT`


`CAP_ROUND`


`CAP_SQUARE`

Stroke

```
private static final Stroke originStroke
        = new BasicStroke(3, BasicStroke.CAP_ROUND,
        BasicStroke.JOIN_ROUND, 1.0f,
        new float[]{10f, 5f}, 0);

private void drawOrigin(Graphics2D g2) {
    Stroke oldStroke = g2.getStroke();
    g2.setStroke(originStroke);
```

- parameters: width, cap, join, miterLimit, dash, dash_phase
- Also constructor with just new line width

Sample Stroke

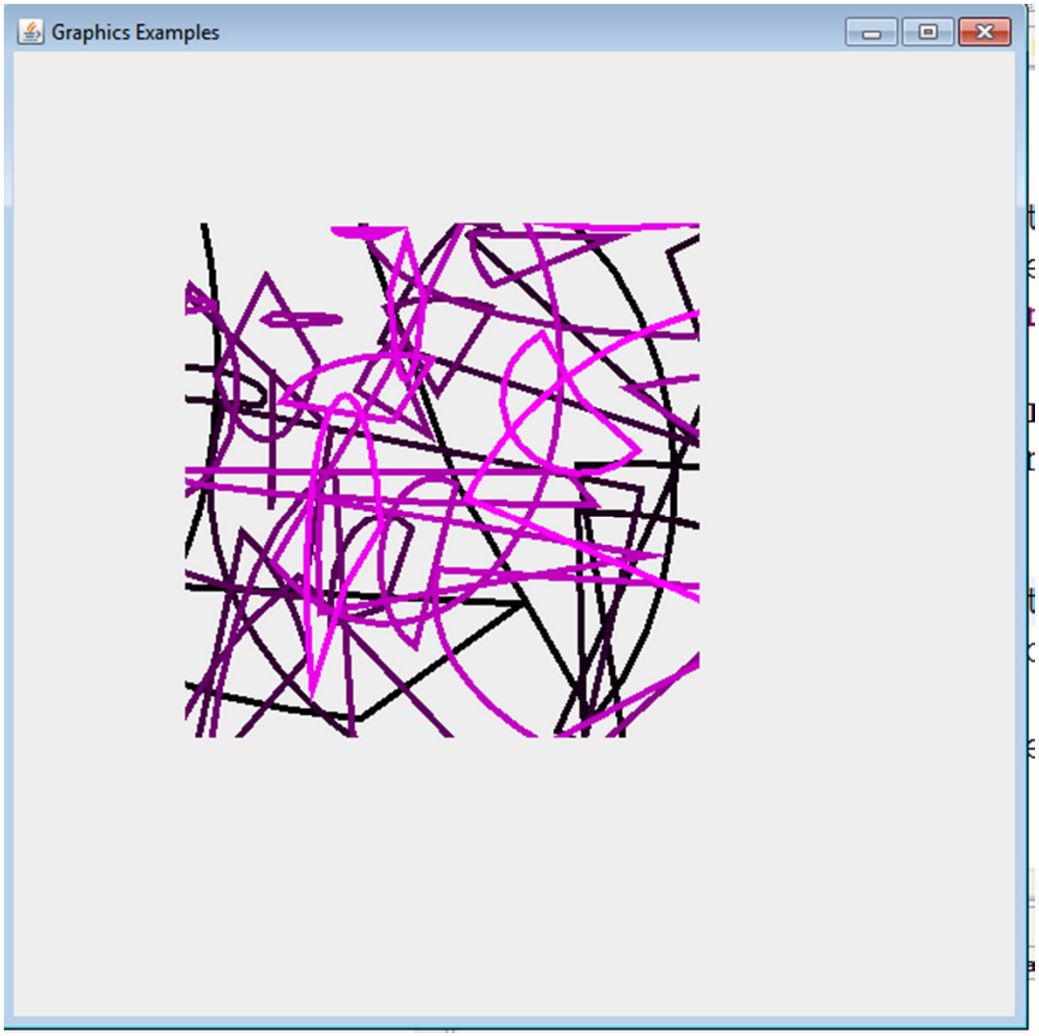
- From previous slide,
  - dashed (10, 5), round caps

The Clip

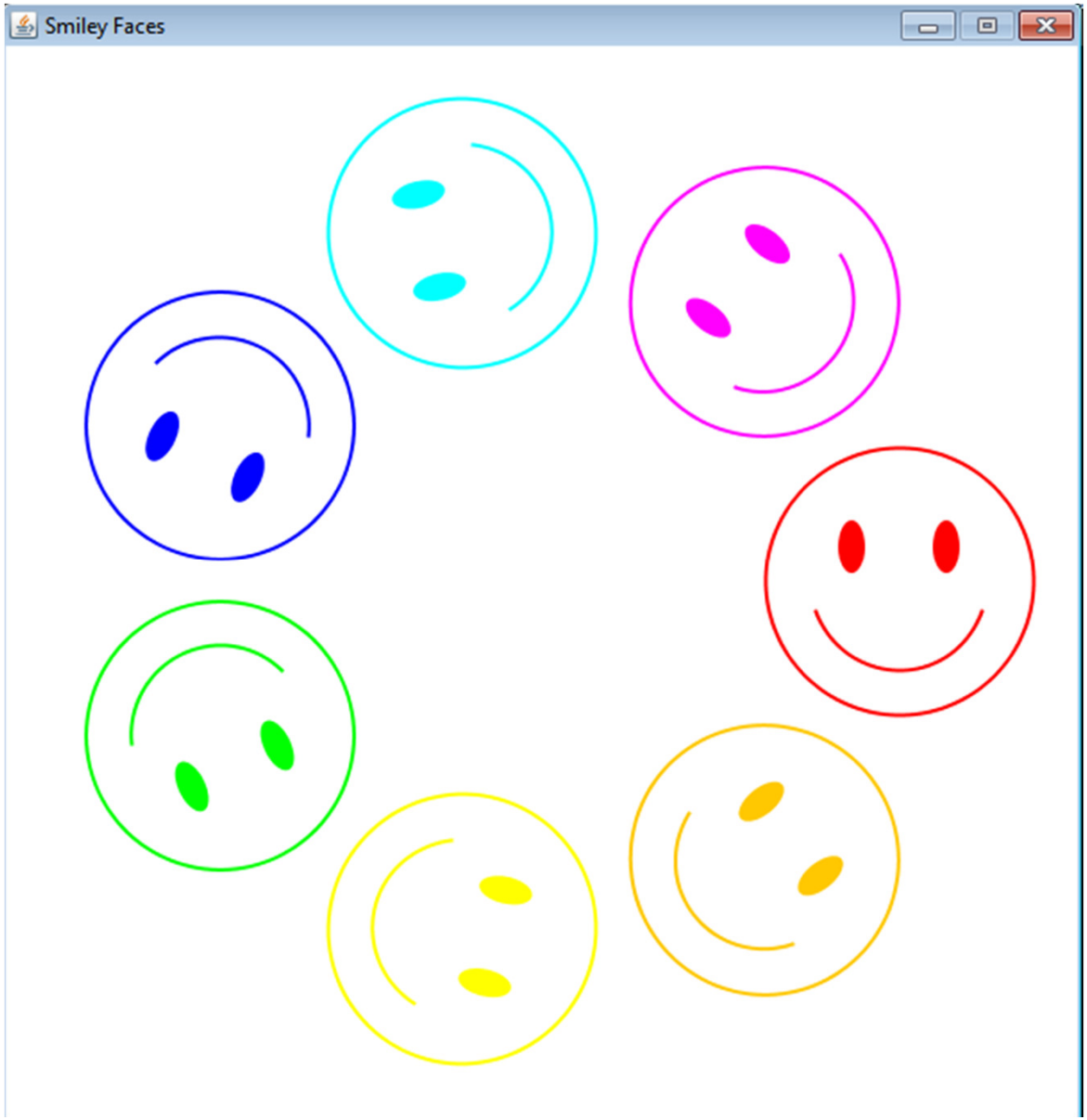
- Restrict area the graphics object will draw on
- Can be any *Shape*
  - so far just primitives, ellipses, rectangles, arcs
  - More advanced possible

The Clip

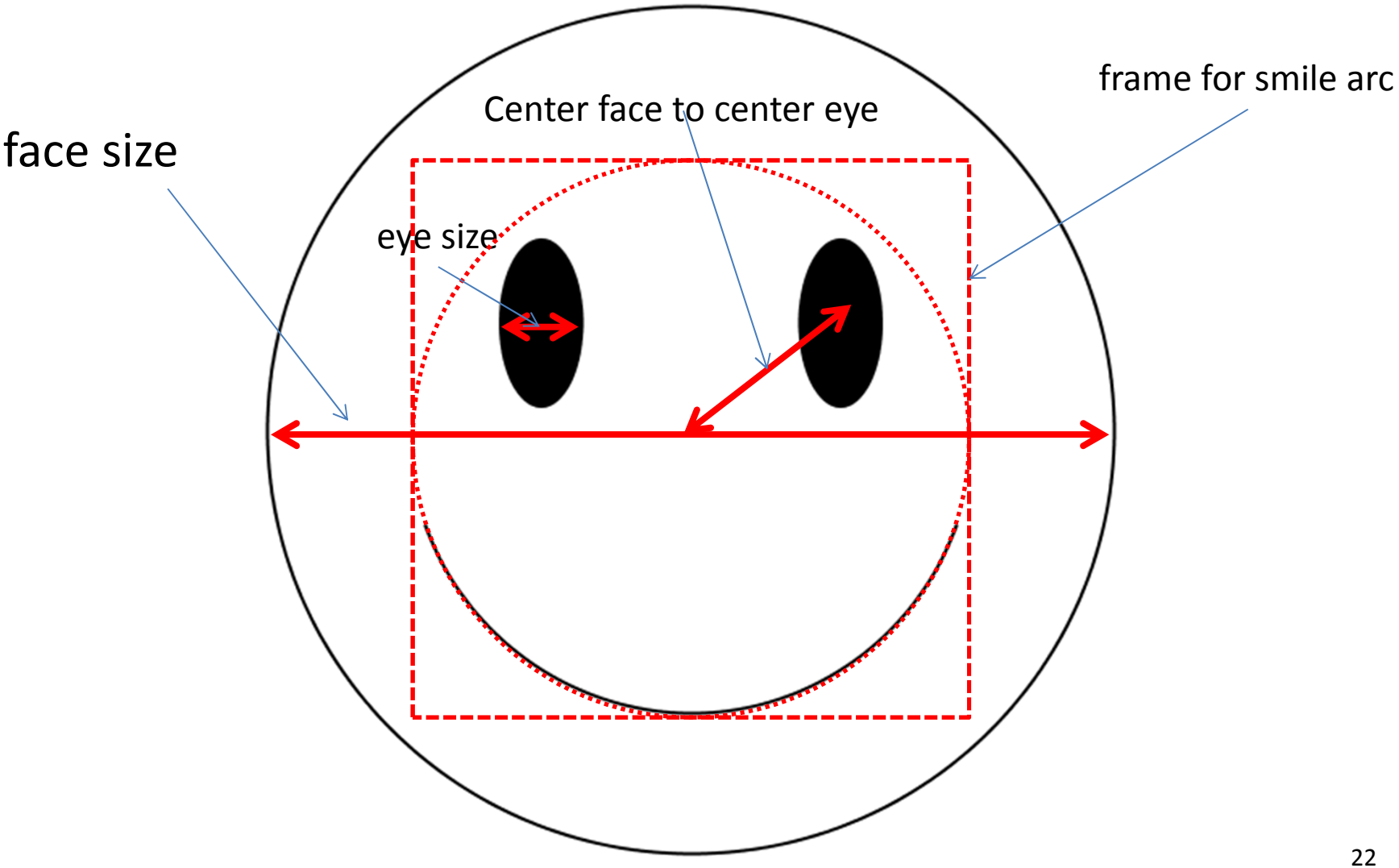
```
Stroke oldStroke = g2.getStroke();
g2.setStroke(new BasicStroke(4));
g2.setClip(new Rectangle2D.Double(100, 100, 300, 300));
```



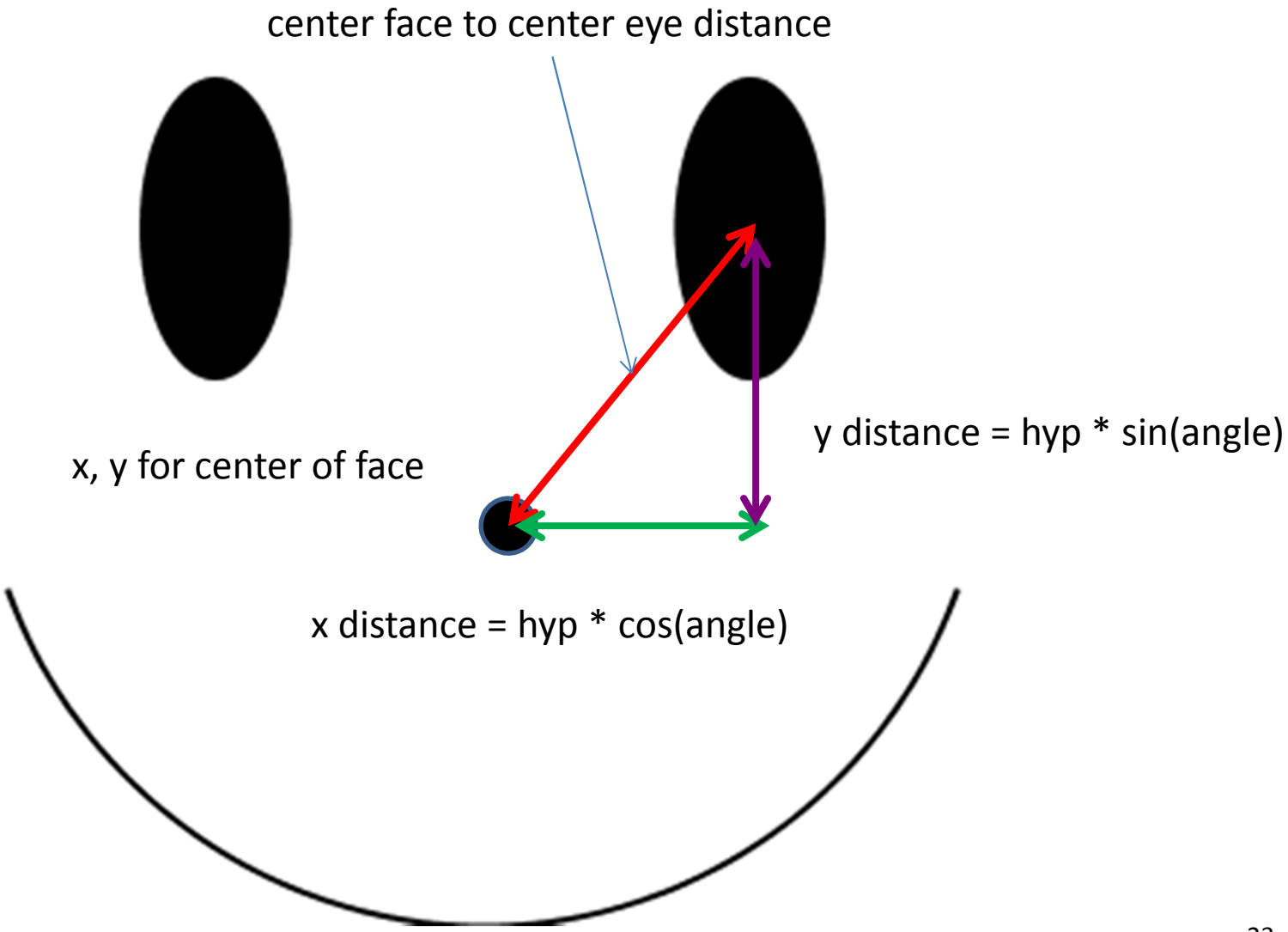
Write a program to draw a circle of Smiley Faces



One Smiley Face



Calculate Eye Distance



Use Constants!

- Constants for various dimensions
- Could even have constants for various ellipses and arcs
  - no need to recalculate each time if the graphics object is translated and rotated correctly
- Complete code to draw one face at a given location and angle
  - generalize the solution!!

Drawing Many

- To draw many smiley faces translate graphics to middle of panel
- determine angle increment for each face
- calculate center point for each face
  - trigonometry again
- array of colors for faces

manyColor