Graphics
The Plan

- Review/Questions
- Hardware
- Coordinate System
- Built-in Shapes
- User-defined Shapes
- Sprites and Shapes
- Making a Sprite
- Play Pong Deathmatch
Review/Questions
Hardware

• Monitor
  – Resolutions (640x480, 800x600, 1280x1024)
  – Bit depth (8, 15, 16, 24, 32)
  – Refresh rate (75-85 Hz)

• Video Card
  – Assists monitor
  – Optimizes graphics
Coordinate Systems

• Cartesian
• Polar
• Screen (Graphics)
• Java 2D (Graphics2D)
Coordinate Systems

- Cartesian
  - Rectangular
  - X increases to the right
  - Y increases as you go up
  - Origin typically at center
  - Real valued
Coordinate Systems

• Polar
  – r increases as distance from the origin increases
  – theta increases in the counterclockwise direction
  – grid lines make concentric circles and sectors
  – Origin typically at center
  – r is real valued
  – theta is from 0 to 2*PI
Coordinate Systems

• Conversion between Cartesian and Polar
  – \([x, y] = [r\cos(\theta), r\sin(\theta)]\)
  – \(r = \sqrt{x^2+y^2}\)
  – \(\theta = \arccos(x/r) \text{ if } y>0\)
  – \(\theta = -\arccos(x/r) \text{ if } y<=0\)
  – No need to memorize this, but you may see it in the code
Coordinate Systems

• Screen (Graphics)
  – Rectangular
  – X increases to the right
  – Y increases as you go down
  – Origin at upper left
  – Non-negative integer valued
Coordinate Systems

• Java 2D (Graphics2D)
  – Rectangular
  – X increases to the right
  – Y increases as you go **down**
  – Origin at upper left
  – Real valued (approximated)
Coordinate Systems

• **Java2D to Screen conversion**
  – Simple – round the floating point to an integer (or just truncate)

• **Screen to Java2D conversion**
  – None needed because integers are approximated reals
Coordinate Systems

Why use Java2D coordinate system?

• Smoother motion

• Integer values often need to be rounded which can lead to more calculation error

• Simpler to rotate and expand
Built-in Shapes

In `java.awt.geom` package

- Ellipse2D.Double
- Rectangle2D.Double
- RoundRectangle2D.Double
- All constructed with `(x, y, width, height)`
- What about circles and squares?
User-defined Shapes

Also in java.awt.geom

• GeneralPath
  – Lines
  – Curves
    • Quadratic
    • Cubic
  – Can be transformed via AffineTransform

• Area
  – Constructive Area Geometry
  – Useful tool for finding intersections
Shapes

All classes so far are all Shapes
• Can draw them using a Graphics2D
• Can get boundary information
• Can be used to make a Sprite…
Sprites and Shapes

Sprites have

• Size
• Shape
• Orientation
• Location
• Color
• Optionally a Tracker
Making a Sprite

How to make a Sprite:

1. Extend Sprite
2. In the constructor
   a. Call super()
   b. Make any Shape
   c. Initialize shape=new GeneralPath(yourShape)
   d. Call normalize()
Making a Sprite

package tipgame;

import java.awt.geom.*;

public class SquareSprite
  extends Sprite

public SquareSprite()
  {
    super();
    Rectangle2D.Double rectangle;
    rectangle=new Rectangle2D.Double(0, 0, 1, 1);
    shape=new GeneralPath(rectangle);
    normalize();
  }

How to make a Sprite:
1. Extend Sprite
2. In the constructor
   a. Call super()
   b. Make any Shape
   c. Initialize shape=new GeneralPath(yourShape)
   d. Call normalize()
Making a Sprite

In the constructor of CAGSprite:

```java
super();
Area area=new Area();
Rectangle2D.Double rectangle;
rectangle=new Rectangle2D.Double(0, 0, 1, 1);
Ellipse2D.Double circle;
circle=new Ellipse2D.Double(0, 0, 1, 1);
area.add(new Area(rectangle));
area.subtract(new Area(circle));
shape=new GeneralPath(area);
normalize();
```
PONG DEATHMATCH