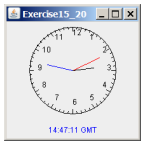
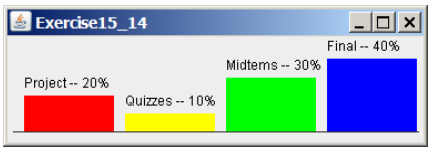
Java Graphics

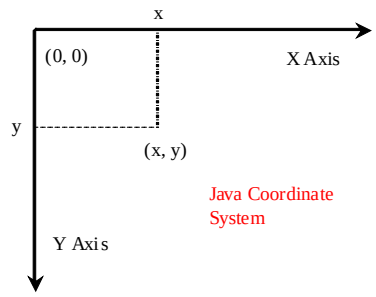
- Drawing shapes in Java such as lines, rectangles, 3-D rectangles, a bar chart, or a clock utilize the Graphics class
  - Drawing Strings
  - Drawing Lines
  - Drawing Rectangles
  - Drawing Ovals
  - Drawing Arcs
  - Drawing Polygons



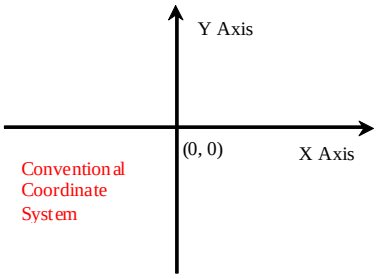


 1

Java Coordinate System

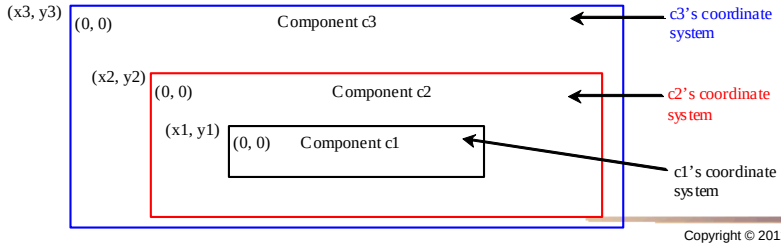


Java Coordinate System



Conventional Coordinate System

Each GUI Component Has its Own Coordinate System



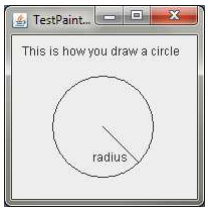
 2

JPanel class paintComponent method

- paintComponent method used to draw graphics
  - JPanel is canvas upon which you paint components
  - paintComponent method utilized by Java Virtual Machine to draw things on a component as needed
  - JPanel class paintComponent method should never be invoked directly in program, let JVM do it
- Subclass needs to be created to inherit JPanel
  - Define a class that extends JPanel
  - Override JPanel paintComponent method so it can be utilized to specify what to draw
  - paintComponent method is set to protected to prevent user invocation

 3

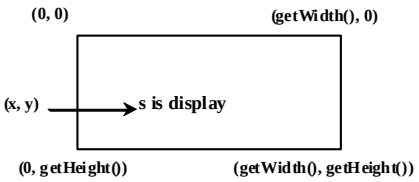
Graphics TestPaint Example



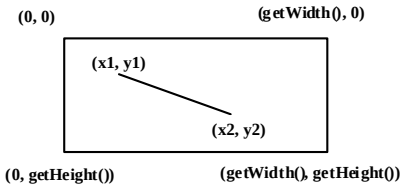

```
import javax.swing.*;
import java.awt.Graphics;
public class TestPaint extends JFrame {
  public TestPaint() {
    GraphicsPanel gpnMain = new GraphicsPanel();
    add(new GraphicsPanel());
  }
  public static void main(String[] args) {
    TestPaint fraMain = new TestPaint();
    fraMain.setTitle("TestPaintComponent");
    fraMain.setSize(200, 200);
    fraMain.setLocationRelativeTo(null); // Center the frame
    fraMain.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraMain.setVisible(true);
  }
}
// Overriding the JPanel class paintComponent method
// requires creating new class and inherit JPanel
class GraphicsPanel extends JPanel {
  // protected visibility to prevent user from invoking
  protected void paintComponent(Graphics gfxDraw) {
    // Call superclass paintComponent to clear canvas
    super.paintComponent(gfxDraw);
    gfxDraw.drawString("This is how you draw a circle", 10, 20);
    gfxDraw.drawOval(40, 40, 100, 100);
    gfxDraw.drawLine(90, 90, 125, 125);
    gfxDraw.drawString("radius", 80, 125);
  }
}
```

Drawing Strings

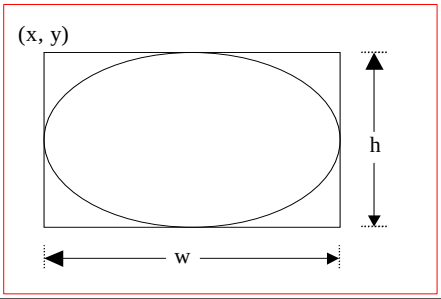
drawString(String s, int x, int y);



drawLine(int x1, int y1, int x2, int y2);



drawOval(int x, int y, int w, int h);



 5

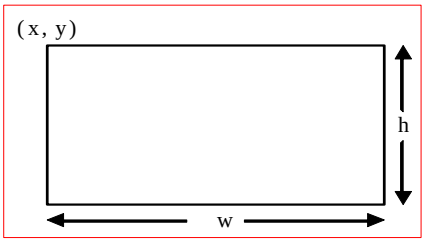
Graphics Class

You can draw strings, lines, rectangles, ovals, arcs, polygons, and polylines, using the methods in the <u>Graphics</u> class

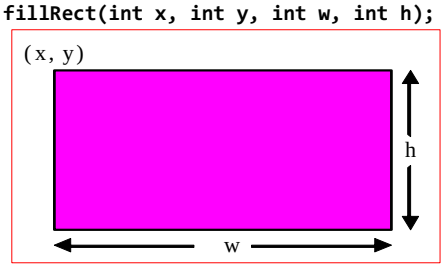
java.awt.Graphics	
+setColor(color: Color): void	Sets a new color for subsequent drawings.
+setFont(font: Font): void	Sets a new font for subsequent drwings.
+drawString(s: String, x: int, y: int): void	Draws a string starting at point (x, y).
+drawLine(x1: int, y1: int, x2: int, y2: int): void	Draws a line from (x1, y1) to (x2, y2).
+drawRect(x: int, y: int, w: int, h: int): void	Draws a rectangle with specified upper-left corner point at (x, y) and width w and height h.
+fillRect(x: int, y: int, w: int, h: int): void	Draws a filled rectangle with specified upper-left corner point at (x, y) and width w and height h.
+drawRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void	Draws a round-cornered rectangle with specified arc width aw and arc height ah.
+fillRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void	Draws a filled round-cornered rectangle with specified arc width aw and arc height ah.
+draw3DRect(x: int, y: int, w: int, h: int, raised: boolean): void	Draws a 3-D rectangle raised above the surface or sunk into the surface.
+fill3DRect(x: int, y: int, w: int, h: int, raised: boolean): void	Draws a filled 3-D rectangle raised above the surface or sunk into the surface.
+drawOval(x: int, y: int, w: int, h: int): void	Draws an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+fillOval(x: int, y: int, w: int, h: int): void	Draws a filled oval bounded by the rectangle specified by the parameters x, y, w, and h.
+drawArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void	Draws an arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+fillArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void	Draws a filled arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+drawPolygon(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a closed polygon defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.
+fillPolygon(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a filled polygon defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.
+drawPolygon(g: Polygon): void	Draws a closed polygon defined by a Polygon object.
+fillPolygon(g: Polygon): void	Draws a filled polygon defined by a Polygon object.
+drawPolyline(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a polyline defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.

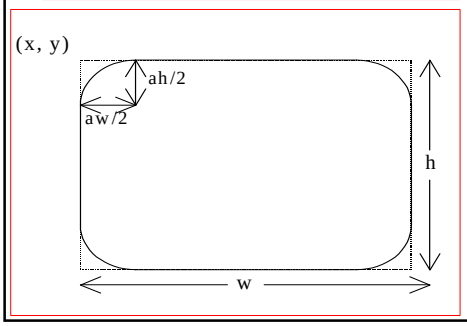
Drawing Rectangles

```
drawRect(int x, int y, int w, int h);
```




```
fillRect(int x, int y, int w, int h);
```






```
drawRoundRect(int x, int y,
   int w, int h, int aw, int ah);

fillRoundRect(int x, int y,
   int w, int h, int aw, int ah);
```

 7

Stacking Order is Last On Top and Color and Font

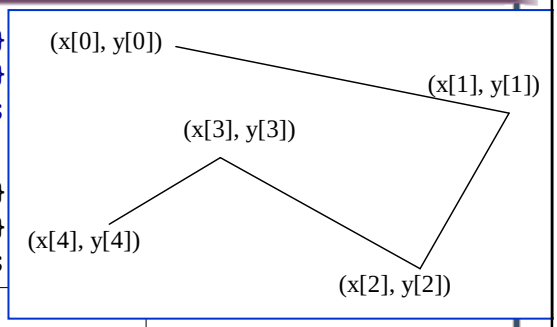
Running as an Applet

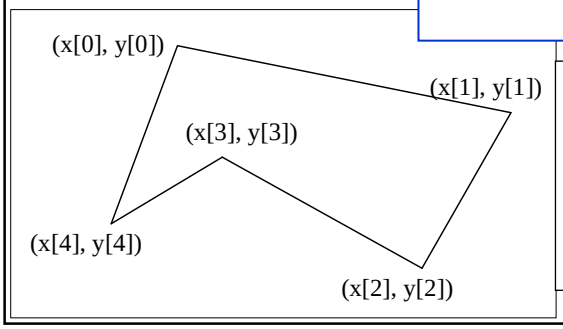

```
import javax.swing.*;
import java.awt.*;
public class PaintInColor extends JApplet {
  // Applet extends JApplet not JFrame and delete main()
  public PaintInColor() {
    add(new PaintCanvas());
  }
}
class PaintCanvas extends JPanel {
  protected void paintComponent(Graphics gfxPaintColors) {
    super.paintComponent(gfxPaintColors);
    // For color changes set before draw method
    // Stacking order is Last draw on Top
    setBackground(new Color(204, 255, 204));  // Change JPanel background color
    gfxPaintColors.setColor(new Color( 255, 255, 153)); // Change brush color
    gfxPaintColors.fillRect(0, 0, 600, 40);
    gfxPaintColors.setColor(new Color( 153, 0, 0));  // Change brush color
    gfxPaintColors.setFont(new Font("Arial", Font.BOLD, 20)); // Change Font
    gfxPaintColors.drawString("Setting Colors is Easy", 30, 30);
    gfxPaintColors.setColor(Color.BLACK); // Change brush color
    gfxPaintColors.drawRoundRect(10, 50, 240, 100, 30, 30);
    gfxPaintColors.setColor(Color.DARK_GRAY); // Change brush color
    gfxPaintColors.fillRoundRect(20, 60, 220, 80, 26, 26);
    gfxPaintColors.setColor(new Color( 255, 220, 153)); // Change brush color
    gfxPaintColors.setFont(new Font("Consolas", Font.BOLD, 14)); // Change Font
    gfxPaintColors.drawString("Change color or font", 40, 90);
    gfxPaintColors.drawString("before you apply the", 40, 110);
    gfxPaintColors.drawString("draw or fill methods!", 40, 130);
  }
}
```

Drawing Polylines and Polygons

```
int[] x = {40, 70, 60, 45, 20}
int[] y = {20, 40, 80, 45, 60}
g.drawPolyline(x,y, x.length);

int[] x = {40, 70, 60, 45, 20}
int[] y = {20, 40, 80, 45, 60}
g.drawPolygon(x, y, x.length);
```



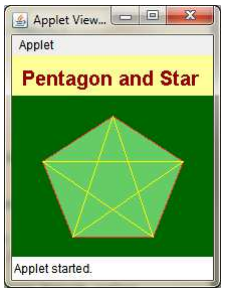



```
// Using Polygon Class to draw
Polygon polygon = new Polygon();
polygon.addPoint(40, 59);
polygon.addPoint(40, 100);
polygon.addPoint(10, 100);
g.drawPolygon(polygon);
```

 9

Drawing Polygons Example

Running as an Applet




```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  public class PentagonStar extends JApplet {
4.    // Applet by extends JApplet not JFrame and delete main()
5.    public PentagonStar() {
6.      add(new PentagonCanvas());
7.    }
8.  }
9.  class PentagonCanvas extends JPanel {
10.   protected void paintComponent(Graphics gfxPentagon) {
11.     super.paintComponent(gfxPentagon);
12.     setBackground(new Color(0, 102, 0));  // Change JPanel background color
13.     // For color changes set before draw method
14.     // Stacking order is Last draw on Top
15.     gfxPentagon.setColor(new Color( 255, 255, 153)); // Change brush color
16.     gfxPentagon.fillRect(0, 0, 600, 40);
17.     gfxPentagon.setColor(new Color( 153, 0, 0));  // Change brush color
18.     gfxPentagon.setFont(new Font("Arial", Font.BOLD, 20)); // Change Font
19.     gfxPentagon.drawString("Pentagon and Star", 10, 30);
20.     gfxPentagon.setColor(new Color( 102, 204, 102)); // Change brush color
21.     int nXpenta[] = {100, 170, 140, 60, 30};
22.     int nYpenta[] = {60, 105, 180, 180, 105};
23.     gfxPentagon.fillPolygon(nXpenta, nYpenta, nXpenta.length);
24.     gfxPentagon.setColor(new Color(255, 51, 51)); // Change brush color
25.     gfxPentagon.drawPolygon(nXpenta, nYpenta, nXpenta.length);
26.     int nXstar[] = {100, 140, 30, 170, 60};
27.     int nYstar[] = {60, 180, 105, 105, 180};
28.     gfxPentagon.setColor(new Color(255, 255, 0)); // Change brush color
29.     gfxPentagon.drawPolygon(nXstar, nYstar, nXpenta.length);
30.   }
31. }
```

Graphics2D class has more controls


```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.geom.*; // Access 2D shapes
4.  public class PentagonStar2D extends JApplet {
5.    public PentagonStar2D() {
6.      add(new PentaStar2DCanvas());
7.    }
8.  }
9.  class PentaStar2DCanvas extends JPanel {
10.   protected void paintComponent(Graphics gfxPentaStar) {
11.     super.paintComponent(gfxPentaStar);
12.     setBackground(new Color(204, 255, 204));
13.     // Graphics2D object is cast from Graphics object
14.     Graphics2D g2dPntStr = (Graphics2D)gfxPentaStar;
15.     gfxPentaStar.setColor(new Color(255, 255, 153));
16.     gfxPentaStar.fillRect(0, 0, 600, 40);
17.     gfxPentaStar.setColor(new Color( 153, 0, 0));
18.     gfxPentaStar.setFont(new Font("Arial", Font.BOLD, 20));
19.     gfxPentaStar.drawString("Pentagon and Star", 10, 30);
20.     gfxPentaStar.setColor(new Color( 51, 153, 51));
21.     int nXpenta[] = {100, 170, 140, 60, 30};
22.     int nYpenta[] = {60, 105, 180, 180, 105};
23.     gfxPentaStar.fillPolygon(nXpenta, nYpenta, nXpenta.length);
24.     gfxPentaStar.setColor(new Color(153, 103, 0)); // Change brush color
25.     g2dPntStr.setStroke(new BasicStroke(10.0f, BasicStroke.CAP_ROUND,
26.         BasicStroke.JOIN_MITER));
27.     g2dPntStr.setBackground(Color.RED);
28.     Polygon shpPentagon = new Polygon(nXpenta, nYpenta, nXpenta.length);
29.     g2dPntStr.draw(shpPentagon);
30.     int nXstar[] = {100, 140, 30, 170, 60};
31.     int nYstar[] = {60, 180, 105, 105, 180};
32.     g2dPntStr.setStroke(new BasicStroke(4.0f, BasicStroke.CAP_ROUND,
33.         BasicStroke.JOIN_MITER));
34.     gfxPentaStar.setColor(new Color(255, 255, 0)); // Change brush color
35.     g2dPntStr.draw(new Polygon(nXstar, nYstar, nXstar.length)); // Draw Star
36.   }
37. }
```

Displaying Image Icons

- Can display image icons in labels and buttons
  - An image icon displays a fixed-size image
  - java.awt.Image class provides resize flexibility
  - Image can be created from an image icon using the getImage()

```
1.  import java.awt.*;
2.  import javax.swing.*;
3.  public class DisplayImage extends JApplet {
4.    public DisplayImage() {
5.      add(new PlayPanel());
6.    }
7.  }
8.  class ImagePanel extends JPanel {
9.    private ImageIcon icnDuck = new ImageIcon("resources/BalloonDuck.png");
10.   private Image imgDuck = icnDuck.getImage();
11.   /** Draw image on the panel */
12.   protected void paintComponent(Graphics gfxMain) {
13.     super.paintComponent(gfxMain);
14.     setBackground(new Color(204, 204, 255));
15.     if (imgDuck != null)
16.         gfxMain.drawImage(imgDuck, 0, 0, 143, 200, this);
17.   }
18. }
```

 12

Event ActionListener ZoomIn & ZoomOut


```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class BalloonGame extends JApplet {
  private JButton btnZoomIn = new JButton("Zoom In");
  private JButton btnZoomOut = new JButton("Zoom Out");
  private BalloonPanel panCanvas = new BalloonPanel();
  public BalloonGame() {
    JPanel panControls = new JPanel();
    panControls.add(btnZoomIn);
    panControls.add(btnZoomOut);
    this.setLayout(new BorderLayout(5, 5));
    this.add(panCanvas, BorderLayout.CENTER);
    this.add(panControls, BorderLayout.NORTH);
    btnZoomIn.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent eventZmIn) {
            panCanvas.zoomIn();
          }
        } );
    btnZoomOut.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent eventZmOut) {
        panCanvas.zoomOut();
      }
    } );
  }
}
```

Content from Section 16.3

 13

Event ActionListener ZoomIn & ZoomOut

Content from Section 16.3

```java
import java.awt.*;
import javax.swing.*;
class BalloonPanel extends JPanel {
  private int nH = 200, nW = 143;
  private ImageIcon icnDuck = new ImageIcon("resources/BallonDuck.png");
  private Image imgDuck = icnDuck.getImage();
  public void zoomIn() {
    nH *= 1.11111;
    nW *= 1.11111;
    repaint();
  }
  public void zoomOut() {
    nH *= 0.9;
    nW *= 0.9;
    repaint();
  }
  protected void paintComponent(Graphics gfxBalloon) {
    super.paintComponent(gfxBalloon);
    setBackground(new Color(204, 204, 255));
    if (imgDuck != null)
        gfxBalloon.drawImage(imgDuck, 0, 0, nW, nH, this);
  }
}
```

 14

MouseEvent

java.awt.event.InputEvent	
+getWhen(): long	Returns the timestamp when this event occurred.
+isAltDown(): boolean	Returns whether or not the Alt modifier is down on this event.
+isControlDown(): boolean	Returns whether or not the Control modifier is down on this event.
+isMetaDown(): boolean	Returns whether or not the Meta modifier is down on this event
+isShiftDown(): boolean	Returns whether or not the Shift modifier is down on this event.

java.awt.event.MouseEvent	
+getButton(): int	Indicates which mouse button has been clicked.
+getClickCount(): int	Returns the number of mouse clicks associated with this event.
+getPoint(): java.awt.Point	Returns a Point object containing the x and y coordinates.
+getX(): int	Returns the x-coordinate of the mouse point.
+getY(): int	Returns the y-coordinate of the mouse point.

java.awt.event.MouseListener	
+mousePressed(e: MouseEvent):	Invoked when mouse button pressed on source component.
void +mouseReleased(e: MouseEvent):	Invoked when mouse button released the source component.
void +mouseClicked(e: MouseEvent):	Invoked when mouse button clicked (pressed and released)
void +mouseEntered(e: MouseEvent): void	Invoked when the mouse enters the source component.
+mouseExited(e: MouseEvent): void	Invoked when the mouse exits the source component.

With Timer and MouseListener Events

Decent and Flame

```
1.  import java.awt.*;
2.  import java.awt.event.*;
3.  import javax.swing.*;
4.  import java.net.URL;
5.  import java.applet.*;
6.  public class BalloonGame1 extends JApplet {
7.    private JButton btnZoomIn = new JButton("Zoom In");
8.    private JButton btnZoomOut = new JButton("Zoom Out");
9.    private JButton btnFlame = new JButton("Flame");
10.   private BalloonPanel panCanvas = new BalloonPanel();
11.   public BalloonGame1() {
12.     JPanel panControls = new JPanel();
13.     panControls.add(btnZoomIn);
14.     panControls.add(btnZoomOut);
15.     panControls.add(btnFlame);
16.     this.setLayout(new BorderLayout(5, 5));
17.     this.add(panCanvas, BorderLayout.CENTER);
18.     this.add(panControls, BorderLayout.NORTH);
19.     btnZoomIn.addActionListener(new ActionListener() {
20.           public void actionPerformed(ActionEvent eventZmIn) {
21.             panCanvas.zoomIn();
22.           }
23.         } );
24.     btnZoomOut.addActionListener(new ActionListener() {
25.       public void actionPerformed(ActionEvent eventZmOut) {
26.         panCanvas.zoomOut();
27.       }
28.     } );
29.     btnFlame.addMouseListener(new MouseAdapter() {
30.       public void mousePressed(MouseEvent eventFlame) {
31.         panCanvas.flameOn();
32.       }
33.       public void mouseReleased(MouseEvent eventFlame) {
34.         panCanvas.flameOff();
35.       }
36.     } );
37.   }
38. }
```

Content from Section 16.10

 16

Graphics Panel and GUI Event Methods

Content from Section 16.10

```java
39. class BalloonPanel extends JPanel {
40.   private int nH = 200, nW = 143, nX = 20, nY = 0, nTickX, nTickY = 100;
41.   private int nHalfWidth, nHeight;
42.   private double dH = 200, dW = 143, dTrig, dVelY, dY;
43.   private boolean bFlameOn;
44.   private String sMessage = "";
45.   private URL urlBalloonDuck = getClass().getResource("resources/BalloonDuck.png");
46.   private ImageIcon icnDuck = new ImageIcon(urlBalloonDuck);
47.   private Image imgDuck = icnDuck.getImage();
48.   URL urlFlame = getClass().getResource("resources/flame.wav");
49.   private AudioClip audioFlame = Applet.newAudioClip(urlFlame);
50.   BalloonPanel() {
51.     Timer tmr30mSec = new Timer(33, new TimerListener());
52.     tmr30mSec.start();
53.   }
54.   public void zoomIn() {
55.     dH *= 1.11111;
56.     dW *= 1.11111;
57.     nH = (int)dH;
58.     nW = (int)dW;
59.     repaint();
60.   }
61.   public void zoomOut() {
62.     dH *= 0.9;
63.     dW *= 0.9;
64.     nH = (int)dH;
65.     nW = (int)dW;
66.     repaint();
67.   }
68.   public void flameOn() {
69.     bFlameOn = true;
70.     audioFlame.play();
71.     nTickY = 500;
72.   }
73.   public void flameOff() {
74.     bFlameOn = false;
75.     audioFlame.stop();
76.   }
77.
```

Timer ActionListener Class and paintComponent method

Content from Section 16.10

```java
78.   protected void paintComponent(Graphics gfxBalloon) {
79.     super.paintComponent(gfxBalloon);
80.     setBackground(new Color(204, 204, 255));
81.     gfxBalloon.drawString(String.format("X = %d", nX), 10, 15);
82.     gfxBalloon.drawString(String.format("Y = %d", nY), 10, 30);
83.     gfxBalloon.drawString(String.format("dY/dt = %6.3f", dVelY), 10, 45);
84.     gfxBalloon.drawString(String.format("%s", sMessage), 10, 60);
85.     if(imgDuck != null)
86.       gfxBalloon.drawImage(imgDuck, nX, nY, nW, nH, this);
87.   }
88.   class TimerListener implements ActionListener {
89.     public void actionPerformed(ActionEvent eventTimer) {
90.       nHalfWidth = (getWidth()/2 - nW/2);
91.       nHeight = getHeight();
92.       if(nY < nHeight - nH) {
93.         nTickX++;
94.         nTickY++;
95.         dTrig = Math.sin((double)nTickX/100);
96.         nX = nHalfWidth - (int)((double)nHalfWidth * dTrig);
97.         dVelY = (0.0005 * (double)nTickY/100) + dVelY;
98.         dY = (dVelY * (double)nTickY/100) + dY;
99.         nY = (int)dY;
100.        sMessage = "";
101.      }
102.      else
103.      {
104.        if(dVelY > 0.1 && dVelY != 0)
105.          sMessage = String.format("You Crashed: dY/dt = %6.3f > 0.1", dVelY);
106.        else if(dVelY <= 0.1 && dVelY != 0)
107.          sMessage = String.format("Soft Landing: dY/dt = %6.3f > 0.1", dVelY);
108.        dVelY = 0;
109.      }
110.      if(bFlameOn) {
111.        dVelY = -(0.002 * (double)nTickY/100) + dVelY;
112.        dY = (dVelY * (double)nTickY/100) + dY;
113.        nY = (int)dY;
114.      }
115.      repaint();
116.    }
117.  }
118. }
```