Timers, Animation & Interface Design
Timers

• Java’s Timer class (from javax.swing package) generates a sequence of action events, spaced at equal intervals

• Timer constructor requires two arguments:
  – delay time (in milliseconds)
  – action listener to handle the event triggered by the Timer

• Useful for animation
Example: creating a simple clock

• A Timer object used in conjunction with a Date object can be used to display and update the current time

• The code on the next slide illustrates this
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.Timer;
// supercedes util's Timer class

public class TimerTest {
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        final int FIELD_WIDTH = 20;
        final JTextField textField = new JTextField(FIELD_WIDTH);
        Container contentPane = frame.getContentPane();
        contentPane.setLayout(new FlowLayout);
        contentPane.add(textField);
        ActionListener listener = new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                Date now = new Date();
                textField.setText(now.toString());
            }
        };
        final int DELAY = 1000;
        // milliseconds between timer ticks
        Timer t = new Timer(DELAY, listener);
        t.start();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}

The ActionListener is associated with the Timer via the Timer's constructor; once the Timer starts, it spawns an ActionEvent once per second.

ActionListener object performs this action: translate the current date & time into a readable String
Animation

• Animation involves painting and repainting the same scene, giving the viewer the illusion of a moving picture

• Two repeated actions can accomplish this
  – draw a shape
  – move the shape
Animation

- To draw the picture, we can begin with an object that implements the Icon interface, and define the paintIcon method
- We can place the resulting Icon object in a JLabel, and place the label in the content pane of a frame for display
- We add a timer to automate the process, passing it an ActionListener that redraws the shape
- In the following example, a series of random pictures are drawn, changing once every second
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import javax.swing.Timer;

public class Transformer extends JFrame implements ActionListener, Icon {
    public static void main(String[] args) {
        JFrame f = new Transformer();
    }

    private Icon[] imageArray;  // collection of random images
    private int index = 0;      // current array index
    private Container win;      // content pane of the frame
    private JLabel pic;         // JLabel used to display current image
    private Timer t;            // spawns new action event every 2 seconds
    private Random rg;          // used for choosing images and generating
                                 // background colors
public Transformer() {
    setTitle("It changes!");
    setAlwaysOnTop(true);  // window always visible, if present
    imageArray = new ImageIcon[8];
    win = this.getContentPane();
    win.setBackground(Color.WHITE);
    win.setLayout(new FlowLayout());
    imageArray[0] = new ImageIcon("3x2x2connector.jpg");
    // some images omitted to fit on this slide ...
    imageArray[7] = new ImageIcon("motor.jpg");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    rg = new Random();
    pic = new JLabel(imageArray[rg.nextInt(8)]);
    win.add(pic);
    setVisible(true);
    t = new Timer(1000, this);
    t.start();
}
public int getIconWidth() {return imageArray[index].getIconWidth();}
public int getIconHeight() {return imageArray[index].getIconHeight();}

public void paintIcon (Component c, Graphics g, int x, int y) {
    index = rg.nextInt(8);
    pic = new JLabel(imageArray[index]);
}

public void paint (Graphics g) {
    win.setBackground(new Color(rg.nextInt(256), rg.nextInt(256), rg.nextInt(256)));
    paintIcon(this, g, 0, 0);
}

public void actionPerformed(ActionEvent event) {
    pic = new JLabel(imageArray[index]);
    win.removeAll(); // erases current image
    win.add(pic); // adds new image to content pane
    repaint(); // calls paint (resetting background &
    pack(); // painting image)
}
Shapes that move

• In the next example, a more typical form of animation is used

• A picture is drawn, then redrawn with the “moving” part of the image in a different location

• The repaint() method takes care of erasing and redrawing the picture

• The next several slides provide an example
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import javax.swing.Timer;

public class Traveller extends JFrame {
    public static void main(String[] args) {
        JFrame f = new Traveller();
    }

    private Container win;  // frame’s content pane
    private Timer t;        // changes picture once per second
    private Random rg;      // provides random image locations
    private ImageIcon bouncer;  // the “moving” part
public Traveller() {
    setTitle("It moves!");
    setSize(200,200);
    setAlwaysOnTop(true);
    win = this.getContentPane();
    win.setBackground(Color.WHITE);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    rg = new Random();
    bouncer = new ImageIcon("barbiehd.gif");
    setVisible(true);
    t = new Timer(1000, new ActionListener () {
        public void actionPerformed(ActionEvent event) {
            repaint();
        }
    });
    t.start();
}
public void paint (Graphics g) {
    g.setColor(Color.RED);
    g.fillRect(0,0,200,200);
    bouncer.paintIcon(win, g, rg.nextInt(150), rg.nextInt(150));
}

Result: an image that moves around randomly within a window
MoveableShape Interface

• In the previous example, an image was drawn, then redrawn in another location
• Since the actions described above can apply to any image, we can define an interface for generic shape animation
• The interface has two methods:
  – draw() draws the shape
  – translate() moves the position of the top left corner of the shape
public interface MoveableShape
{
    void draw(Graphics2D g2);
    void translate(double dx, double dy);
}

The next several slides illustrate the use of this interface
import java.awt.*;
import javax.swing.*;
import java.awt.geom.*;
import java.awt.event.*;

public class BouncingBall implements MoveableShape {
    private double x, y;         // position of ball
    int size;                    // size of ball

    public BouncingBall(double x, double y, int size) {
        this.x = x;
        this.y = y;
        this.size = size;
    }
}
// Ball “moves” according to the arguments passed to this method
public void translate (double dx, double dy) {
    x += dx;
    y += dy;
}

// Paints ball at its current position (defined by x & y)
public void draw(Graphics2D g2) {
    g2.setColor(Color.red);
    Ellipse2D.Double ball = new Ellipse2D.Double(x,y,size,size);
    g2.fill(ball);
}
import java.awt.*;
import java.util.*;
import javax.swing.*;

// This class contains a MoveableShape; because it implements Icon,
// we can use it to put a BouncingBall object into a JLabel for
// display (as we’ll see in the driver program)

public class ShapeIcon implements Icon {
    private int width;
    private int height;
    private MoveableShape shape;

    public ShapeIcon(MoveableShape shape, int width, int height) {
        this.shape = shape;
        this.width = width;
        this.height = height;
    }
}
public int getIconWidth() {
    return width;
}

public int getIconHeight() {
    return height;
}

public void paintIcon(Component c, Graphics g, int x, int y) {
    Graphics2D g2 = (Graphics2D) g;
    shape.draw(g2);
}
import java.awt.*;
import javax.swing.*;
import java.awt.geom.*;
import java.awt.event.*;

public class DropTheBall extends JFrame {
    private JLabel space;   // space ball occupies
    MoveableShape bb;       // BouncingBall object
    ShapeIcon shape;        // Icon container for BouncingBall
public DropTheBall () {
    setSize(300,300);
    setTitle("oops");
    setAlwaysOnTop(true);
    bb = new BouncingBall(135,0,20);
    shape = new ShapeIcon(bb, 300,300);
    space = new JLabel(shape);
    Container contentPane = getContentPane();
    contentPane.setLayout(new FlowLayout());
    contentPane.add(space);
    setVisible(true);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Timer t = new Timer(10, new ActionListener () {
        public void actionPerformed(ActionEvent event) {
            bb.translate(0,1);
            space.repaint();
        }
    });
    t.start();
}
public static void main (String [] args) {
    DropTheBall b = new DropTheBall();
}

Action shots: