Today's Lecture

Java Applets, Animation, Interaction

1. Java Applets

2. Writing Solid Code (WSC)

3. More Java Applets

GraphApplet.java

```
import java.awt.Graphics;

public class GraphApplet extends java.applet.Applet {
    double f(double x) {
        return (Math.cos(x/5) + Math.sin(x/7) + 2) * size().height / 4;
    }

    public void paint(Graphics g) {
        for (int x = 0 ; x < size().width ; x++) {
            g.drawLine(x, (int)f(x), x + 1, (int)f(x + 1));
        }
    }
}
```


JackHammerDuke.java

```
/*
 * @(#)JackhammerDuke.java      1.2 95/10/13 Bob Weisblatt
 *
 * Copyright (c) 1994-1995 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL or COMMERCIAL purposes and
 * without fee is hereby granted.
 * Please refer to the file http://java.sun.com/copy_trademarks.html
 * for further important copyright and trademark information and to
 * http://java.sun.com/licensing.html for further important licensing
 * information for the Java (tm) Technology.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 *
 * THIS SOFTWARE IS NOT DESIGNED OR INTENDED FOR USE OR RESALE AS ON-LINE
 * CONTROL EQUIPMENT IN HAZARDOUS ENVIRONMENTS REQUIRING FAIL-SAFE
 * PERFORMANCE, SUCH AS IN THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT
 * NAVIGATION OR COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL, DIRECT LIFE
 * SUPPORT MACHINES, OR WEAPONS SYSTEMS, IN WHICH THE FAILURE OF THE
 * SOFTWARE COULD LEAD DIRECTLY TO DEATH, PERSONAL INJURY, OR SEVERE
 * PHYSICAL OR ENVIRONMENTAL DAMAGE ("HIGH RISK ACTIVITIES").  SUN
 * SPECIFICALLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR
 * HIGH RISK ACTIVITIES.
 */

import java.io.InputStream;
import java.awt.*;
import java.net.*;

/**
 * A simple java.applet.Applet class to play an image loop.  The "img" tag parameter
 * indicates what image loop to play.
 *
 * @author      Bob Weisblatt (major thievery from jag, maybe wholesale is a better word.)
 * @version     1.07f, 19 Jan 1995
 */
```



```
public class JackhammerDuke extends java.applet.Applet implements Runnable {
    /**
     * The current loop slot.
     */
    int seqslot = 0;

    /**
     * The number of images
     */
    int nimgs = 4;

    /**
     * A strip of 4 images.
     */
    Image imgs;

    /**
     * The directory or URL from which the images are loaded
     */
    String dir;

    /**
     * The thread animating the images.
     */
    Thread kicker = null;

    /**
     * The length of the pause between revs.lkj
     */
    int pause;

    /**
     * The current x position.
     */
    double x;

    int imgsWidth = 328;
    int imgsHeight = 90;

    /**
     * Run the image loop. This method is called by class Thread.
     * @see oak.lang.Thread
     *
     * getDocumentBase() is the URL of the document in which the applet is
```

```
     * embedded. It shouldn't be changed.
     *
     * dir is in net.www.html.Tag but is undocumented.
     */
    public void run() {
        Thread.currentThread().setPriority(Thread.MIN_PRIORITY);

        imgs = getImage(getCodeBase(), "images/jack.gif");

        if (imgs != null) {
            x = (size().width - imgsWidth/nimgs) / 2;
            while (size().width > 0 && size().height > 0 && kicker != null) {
                if (seqslot == 0) {
                    play(getCodeBase(), "audio/jackhammer-short.au");
                }

                repaint();
                try {
                    Thread.sleep((seqslot == sequence.length -1) ? 500 : 100);
                } catch (InterruptedException e){}
                seqslot = (seqslot + 1) % sequence.length;
            }
        }

    }

    private int sequence[] = { 2, 1, 0, 1, 0, 1, 0, 1, 2, 1, 0, 2, 0, 1, 0, 2, 3};

    /**
     * Paint the current frame.
     */
    boolean erase;

    public void update(Graphics g) {
        if (erase || (sequence[seqslot] == 3)) {
            g.setColor(Color.lightGray);
            g.fillRect(0, 0, size().width, size().height);
            erase = false;
        }
        paint(g);
    }
    public void paint(Graphics g) {
        int loopslot = sequence[seqslot];
        if ((imgs != null) && (loopslot < nimgs)) {
            int w = imgsWidth / nimgs;
```



```
            x = Math.max(0, Math.min(size().width - w, x + Math.random() * 4 - 2));

            g.clipRect((int)x, 0, w, imgsHeight);
            g.drawImage(imgs, (int)x - loopslot * w, 0, this);
            erase = (loopslot == 3);
        }
    }

    /**
     * Start the applet by forking an animation thread.
     */
    public void start() {
        if (kicker == null) {
            kicker = new Thread(this);
            kicker.start();
        }
    }

    /**
     * Stop the applet. The thread will exit because kicker is set to null.
     */
    public void stop() {
        kicker = null;
    }
}
```


Writing Solid Code

1. `ASSERT`

2. “Hungarian”

3. Code to check code

4. Misc.

`Clock2.java`


```
/*
 * Copyright (c) 1994-1995 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL or COMMERCIAL purposes and
 * without fee is hereby granted.
 * Please refer to the file http://java.sun.com/copy_trademarks.html
 * for further important copyright and trademark information and to
 * http://java.sun.com/licensing.html for further important licensing
 * information for the Java (tm) Technology.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 *
 * THIS SOFTWARE IS NOT DESIGNED OR INTENDED FOR USE OR RESALE AS ON-LINE
 * CONTROL EQUIPMENT IN HAZARDOUS ENVIRONMENTS REQUIRING FAIL-SAFE
 * PERFORMANCE, SUCH AS IN THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT
 * NAVIGATION OR COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL, DIRECT LIFE
 * SUPPORT MACHINES, OR WEAPONS SYSTEMS, IN WHICH THE FAILURE OF THE
 * SOFTWARE COULD LEAD DIRECTLY TO DEATH, PERSONAL INJURY, OR SEVERE
 * PHYSICAL OR ENVIRONMENTAL DAMAGE ("HIGH RISK ACTIVITIES").  SUN
 * SPECIFICALLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR
 * HIGH RISK ACTIVITIES.
 */

// author: Rachel Gollub, 1995
// Time!

import java.util.*;
import java.awt.*;
import java.applet.*;

public class Clock2 extends Applet implements Runnable {
  Thread timer = null;
  int lastxs=0, lastys=0, lastxm=0, lastym=0, lastxh=0, lastyh=0;
  Date dummy = new Date();
  String lastdate = dummy.toLocaleString();
```

```
public void init() {
  int x,y;
  resize(300,300);              // Set clock window size
}

  // Plotpoints allows calculation to only cover 45 degrees of the circle,
  // and then mirror

public void plotpoints(int x0, int y0, int x, int y, Graphics g) {

  g.drawLine(x0+x,y0+y,x0+x,y0+y);
  g.drawLine(x0+y,y0+x,x0+y,y0+x);
  g.drawLine(x0+y,y0-x,x0+y,y0-x);
  g.drawLine(x0+x,y0-y,x0+x,y0-y);
  g.drawLine(x0-x,y0-y,x0-x,y0-y);
  g.drawLine(x0-y,y0-x,x0-y,y0-x);
  g.drawLine(x0-y,y0+x,x0-y,y0+x);
  g.drawLine(x0-x,y0+y,x0-x,y0+y);
}

  // Circle is just Bresenham's algorithm for a scan converted circle

public void circle(int x0, int y0, int r, Graphics g) {
  int x,y;
  float d;

  x=0;
  y=r;
  d=5/4-r;
  plotpoints(x0,y0,x,y,g);

  while (y>x){
    if (d<0) {
      d=d+2*x+3;
      x++;
    }
    else {
      d=d+2*(x-y)+5;
      x++;
      y--;
    }
    plotpoints(x0,y0,x,y,g);
  }
}
```



```
  // Paint is the main part of the program

public void paint(Graphics g) {
  int xh, yh, xm, ym, xs, ys, s, m, h, xcenter, ycenter;
  String today;
  Date dat = new Date();

  s = dat.getSeconds();
  m = dat.getMinutes();
  h = dat.getHours();
  today = dat.toLocaleString();
  xcenter=80;
  ycenter=55;

  // a= s* pi/2 - pi/2 (to switch 0,0 from 3:00 to 12:00)
  // x = r(cos a) + xcenter, y = r(sin a) + ycenter

  xs = (int)(Math.cos(s * 3.14f/30 - 3.14f/2) * 45 + xcenter);
  ys = (int)(Math.sin(s * 3.14f/30 - 3.14f/2) * 45 + ycenter);
  xm = (int)(Math.cos(m * 3.14f/30 - 3.14f/2) * 40 + xcenter);
  ym = (int)(Math.sin(m * 3.14f/30 - 3.14f/2) * 40 + ycenter);
  xh = (int)(Math.cos((h*30 + m/2) * 3.14f/180 - 3.14f/2) * 30 + xcenter);
  yh = (int)(Math.sin((h*30 + m/2) * 3.14f/180 - 3.14f/2) * 30 + ycenter);

  // Draw the circle and numbers

  g.setFont(new Font("TimesRoman", Font.PLAIN, 14));
  g.setColor(Color.blue);
  circle(xcenter,ycenter,50,g);
  g.setColor(Color.darkGray);
  g.drawString("9",xcenter-45,ycenter+3);
  g.drawString("3",xcenter+40,ycenter+3);
  g.drawString("12",xcenter-5,ycenter-37);
  g.drawString("6",xcenter-3,ycenter+45);

  // Erase if necessary, and redraw

  g.setColor(Color.lightGray);
  if (xs != lastxs || ys != lastys) {
    g.drawLine(xcenter, ycenter, lastxs, lastys);
    g.drawString(lastdate, 5, 125);
  }
  if (xm != lastxm || ym != lastym) {
    g.drawLine(xcenter, ycenter-1, lastxm, lastym);
```



```
    g.drawLine(xcenter-1, ycenter, lastxm, lastym); }
  if (xh != lastxh || yh != lastyh) {
    g.drawLine(xcenter, ycenter-1, lastxh, lastyh);
    g.drawLine(xcenter-1, ycenter, lastxh, lastyh); }
  g.setColor(Color.darkGray);
  g.drawString(today, 5, 125);
  g.drawLine(xcenter, ycenter, xs, ys);
  g.setColor(Color.blue);
  g.drawLine(xcenter, ycenter-1, xm, ym);
  g.drawLine(xcenter-1, ycenter, xm, ym);
  g.drawLine(xcenter, ycenter-1, xh, yh);
  g.drawLine(xcenter-1, ycenter, xh, yh);
  lastxs=xs; lastys=ys;
  lastxm=xm; lastym=ym;
  lastxh=xh; lastyh=yh;
  lastdate = today;
}

public void start() {
  if(timer == null)
    {
      timer = new Thread(this);
      timer.start();
    }
}

public void stop() {
  timer = null;
}

public void run() {
  while (timer != null) {
    try {
        Thread.sleep(100);
    } catch (InterruptedException e){}
    repaint();
  }
  timer = null;
}

public void update(Graphics g) {
  paint(g);
}
}
```


ImageTape.java


```
import java.awt.*;

public class ImageTape extends java.applet.Applet implements Runnable {
    Image imgs[];
    int nimgs;
    int imgwidth;
    int x, newx;
    int dist = 5;
    int timeout = 200;
    Thread scroller;
    String dir;

    public void init() {
        String at = getParameter("img");
        dir = (at != null) ? at : "doc:/demo/images/duke";
        at = getParameter("speed");
        timeout = 1000 / ((at == null) ? 4 : Integer.valueOf(at).intValue());
        at = getParameter("dir");
        dist = (at == null) ? 5 : Integer.valueOf(at).intValue();
        at = getParameter("nimgs");
        nimgs = (at == null) ? 16 : Integer.valueOf(at).intValue();
        newx = x = size().width;

        imgs = new Image[nimgs];
        for (int i = 0 ; i < nimgs ; i++) {
            imgs[i] = getImage(getDocumentBase(), dir + "/T" + (i+1) + ".gif");
        }
    }

    public void start() {
        if (scroller == null) {
            scroller = new Thread(this);
            scroller.start();
        }
    }
    public void stop() {
        if (scroller != null) {
            scroller.stop();
            scroller = null;
        }
    }

    public void run() {
```

```
        while (true) {
            try {
                Thread.currentThread().sleep(timeout);
            } catch (InterruptedException e){}
            scroll(dist);
        }
    }

    synchronized void scroll(int dist) {
        newx += dist;
        repaint();
    }

    public void update(Graphics g) {
        g.setColor(Color.lightGray);

        if (newx != x) {
            //g.clipRect(1, 1, size().width-2, size().height-2);
            int dist = newx - x;
            if (dist > 0) {
                g.copyArea(1, 1, (size().width-2) - dist, size().height-2, dist, 0);
                for (x=newx ; x>size().width ; x -= Math.max(size().width-2, imgwidth));
                paint(g, 1, dist + 1);
            } else {
                g.copyArea(1 - dist, 1, (size().width-2) + dist, size().height-2, dist, 0);
                for (x = newx ; x < 0 ; x += Math.max(size().width - 2, imgwidth));
                paint(g, (size().width-1) + dist, size().width-1);
            }
        } else {
            paint(g);
        }
    }

    public boolean imageUpdate(Image img, int flags, int x, int y, int w, int h) {
        if ((flags & WIDTH) != 0) {
            imgwidth += img.getWidth(this);
        }
        return super.imageUpdate(img, flags, x, y, w, h);
    }

    public void paint(Graphics g, int fromx, int tox) {
        int x = this.x;
        newx = x;

        g.setColor(Color.lightGray);
```



```
        g.fillRect(fromx, 1, tox - fromx, size().height-2);
        g.clipRect(fromx, 1, tox - fromx, size().height-2);
        g.setColor(Color.black);

        for (int i = 0 ; i < nimgs ; i++) {
            if (imgs[i] == null) {
                continue;
            }

            int w = imgs[i].getWidth(this);
            int h = imgs[i].getHeight(this);
            if ((w > 0) && (h > 0)) {
                if ((x + w > fromx) && (x < tox)) {
                    g.drawImage(imgs[i], x, size().height - (h+1), this);
                }
                if ((x + w) > size().width) {
                    x -= Math.max(size().width - 2, imgwidth);
                    if ((x + w > fromx) && (x < tox)) {
                        g.drawImage(imgs[i], x, size().height - (h + 1), this);
                    }
                }
                x += w;
            }
        }
    }
    public void paint(Graphics g) {
        g.draw3DRect(0, 0, size().width, size().height, true);
        paint(g, 1, size().width-1);
    }
}
```


ThreeD.java


```
/* A set of classes to parse, represent and display 3D wireframe models
   represented in Wavefront .obj format. */

import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Event;
import java.io.StreamTokenizer;
import java.io.InputStream;
import java.io.IOException;
import java.net.URL;

class FileFormatException extends Exception {
    public FileFormatException(String s) {
        super(s);
    }
}

/** The representation of a 3D model */
class Model3D {
    float vert[];
    int tvert[];
    int nvert, maxvert;
    int con[];
    int ncon, maxcon;
    boolean transformed;
    Matrix3D mat;

    float xmin, xmax, ymin, ymax, zmin, zmax;

    Model3D () {
        mat = new Matrix3D ();
        mat.xrot(20);
        mat.yrot(30);
    }
    /** Create a 3D model by parsing an input stream */
    Model3D (InputStream is) throws IOException, FileFormatException {
        this();
        StreamTokenizer st = new StreamTokenizer(is);
        st.eolIsSignificant(true);
        st.commentChar('#');
      scan:
```



```
        while (true) {
            switch (st.nextToken()) {
              default:
                break scan;
              case StreamTokenizer.TT_EOL:
                break;
              case StreamTokenizer.TT_WORD:
                if ("v".equals(st.sval)) {
                    double x = 0, y = 0, z = 0;
                    if (st.nextToken() == StreamTokenizer.TT_NUMBER) {
                        x = st.nval;
                        if (st.nextToken() == StreamTokenizer.TT_NUMBER) {
                            y = st.nval;
                            if (st.nextToken() == StreamTokenizer.TT_NUMBER)
                                z = st.nval;
                        }
                    }
                    addVert((float) x, (float) y, (float) z);
                    while (st.ttype != StreamTokenizer.TT_EOL &&
                           st.ttype != StreamTokenizer.TT_EOF)
                        st.nextToken();
                } else if ("f".equals(st.sval) ||
                           "fo".equals(st.sval) ||
                           "l".equals(st.sval)) {
                    int start = -1;
                    int prev = -1;
                    int n = -1;
                    while (true)
                        if (st.nextToken() == StreamTokenizer.TT_NUMBER) {
                            n = (int) st.nval;
                            if (prev >= 0)
                                add(prev - 1, n - 1);
                            if (start < 0)
                                start = n;
                            prev = n;
                        } else if (st.ttype == '/')
                            st.nextToken();
                        else
                            break;
                    if (start >= 0)
                        add(start - 1, prev - 1);
                    if (st.ttype != StreamTokenizer.TT_EOL)
                        break scan;
                } else {
                    while (st.nextToken() != StreamTokenizer.TT_EOL
```

```
                        && st.ttype != StreamTokenizer.TT_EOF);
            }
        }
    }
    is.close();
    if (st.ttype != StreamTokenizer.TT_EOF)
        throw new FileFormatException(st.toString());
}

/** Add a vertex to this model */
int addVert(float x, float y, float z) {
    int i = nvert;
    if (i >= maxvert)
        if (vert == null) {
            maxvert = 100;
            vert = new float[maxvert * 3];
        } else {
            maxvert *= 2;
            float nv[] = new float[maxvert * 3];
            System.arraycopy(vert, 0, nv, 0, vert.length);
            vert = nv;
        }
    i *= 3;
    vert[i] = x;
    vert[i + 1] = y;
    vert[i + 2] = z;
    return nvert++;
}
/** Add a line from vertex p1 to vertex p2 */
void add(int p1, int p2) {
    int i = ncon;
    if (p1 >= nvert || p2 >= nvert)
        return;
    if (i >= maxcon)
        if (con == null) {
            maxcon = 100;
            con = new int[maxcon];
        } else {
            maxcon *= 2;
            int nv[] = new int[maxcon];
            System.arraycopy(con, 0, nv, 0, con.length);
            con = nv;
        }
    if (p1 > p2) {
        int t = p1;
```



```
        p1 = p2;
        p2 = t;
    }
    con[i] = (p1 << 16) | p2;
    ncon = i + 1;
}
/** Transform all the points in this model */
void transform() {
    if (transformed || nvert <= 0)
        return;
    if (tvert == null || tvert.length < nvert * 3)
        tvert = new int[nvert*3];
    mat.transform(vert, tvert, nvert);
    transformed = true;
}

/* Quick Sort implementation
 */
private void quickSort(int a[], int left, int right)
    {
        int leftIndex = left;
        int rightIndex = right;
        int partionElement;
        if (right > left)
            {

                /* Arbitrarily establishing partition element as the midpoint of
                 * the array.
                 */
                partionElement = a[ ( left + right ) / 2 ];

                // loop through the array until indices cross
                    while( leftIndex <= rightIndex )
                        {
                            /* find the first element that is greater than or equal to
                             * the partionElement starting from the leftIndex.
                             */
                            while((leftIndex < right) &&
                                  (a[leftIndex] < partionElement))
                                ++leftIndex;

                            /* find an element that is smaller than or equal to
                             * the partionElement starting from the rightIndex.
                             */
                            while( ( rightIndex > left ) &&
```



```
                                  ( a[rightIndex] > partionElement ) )
                                --rightIndex;

                            // if the indexes have not crossed, swap
                                if( leftIndex <= rightIndex )
                                    {
                                        swap(a, leftIndex, rightIndex);
                                        ++leftIndex;
                                        --rightIndex;
                                    }
                        }

                /* If the right index has not reached the left side of array
                 * must now sort the left partition.
                 */
                if( left < rightIndex )
                    quickSort( a, left, rightIndex );

                /* If the left index has not reached the right side of array
                 * must now sort the right partition.
                 */
                if( leftIndex < right )
                    quickSort( a, leftIndex, right );

            }
    }

private void swap(int a[], int i, int j)
    {
        int T;
        T = a[i];
        a[i] = a[j];
        a[j] = T;
    }


/** eliminate duplicate lines */
void compress() {
    int limit = ncon;
    int c[] = con;
    quickSort(con, 0, ncon - 1);
    int d = 0;
    int pp1 = -1;
    for (int i = 0; i < limit; i++) {
        int p1 = c[i];
```



```
        if (pp1 != p1) {
            c[d] = p1;
            d++;
        }
        pp1 = p1;
    }
    ncon = d;
}

static Color gr[];

/** Paint this model to a graphics context.  It uses the matrix associated
  with this model to map from model space to screen space.
  The next version of the browser should have double buffering,
  which will make this *much* nicer */
void paint(Graphics g) {
    if (vert == null || nvert <= 0)
        return;
    transform();
    if (gr == null) {
        gr = new Color[16];
        for (int i = 0; i < 16; i++) {
            int grey = (int) (170*(1-Math.pow(i/15.0, 2.3)));
            gr[i] = new Color(grey, grey, grey);
        }
    }
    int lg = 0;
    int lim = ncon;
    int c[] = con;
    int v[] = tvert;
    if (lim <= 0 || nvert <= 0)
        return;
    for (int i = 0; i < lim; i++) {
        int T = c[i];
        int p1 = ((T >> 16) & 0xFFFF) * 3;
        int p2 = (T & 0xFFFF) * 3;
        int grey = v[p1 + 2] + v[p2 + 2];
        if (grey < 0)
            grey = 0;
        if (grey > 15)
            grey = 15;
        if (grey != lg) {
            lg = grey;
            g.setColor(gr[grey]);
        }
```

```java
            g.drawLine(v[p1], v[p1 + 1],
                       v[p2], v[p2 + 1]);
        }
    }

    /** Find the bounding box of this model */
    void findBB() {
        if (nvert <= 0)
            return;
        float v[] = vert;
        float xmin = v[0], xmax = xmin;
        float ymin = v[1], ymax = ymin;
        float zmin = v[2], zmax = zmin;
        for (int i = nvert * 3; (i -= 3) > 0;) {
            float x = v[i];
            if (x < xmin)
                xmin = x;
            if (x > xmax)
                xmax = x;
            float y = v[i + 1];
            if (y < ymin)
                ymin = y;
            if (y > ymax)
                ymax = y;
            float z = v[i + 2];
            if (z < zmin)
                zmin = z;
            if (z > zmax)
                zmax = z;
        }
        this.xmax = xmax;
        this.xmin = xmin;
        this.ymax = ymax;
        this.ymin = ymin;
        this.zmax = zmax;
        this.zmin = zmin;
    }
}

/** An applet to put a 3D model into a page */
public class ThreeD extends Applet implements Runnable {
    Model3D md;
    boolean painted = true;
    float xfac;
    int prevx, prevy;
```



```java
    float xtheta, ytheta;
    float scalefudge = 1;
    Matrix3D amat = new Matrix3D(), tmat = new Matrix3D();
    String mdname = null;
    String message = null;

    public void init() {
        mdname = getParameter("model");
        try {
            scalefudge = Float.valueOf(getParameter("scale")).floatValue();
        }catch(Exception e){};
        amat.yrot(20);
        amat.xrot(20);
        if (mdname == null)
            mdname = "model.obj";
        resize(size().width <= 20 ? 400 : size().width,
               size().height <= 20 ? 400 : size().height);
    }
    public void run() {
        InputStream is = null;
        try {
            Thread.currentThread().setPriority(Thread.MIN_PRIORITY);
            is = new URL(getDocumentBase(), mdname).openStream();
            Model3D m = new Model3D (is);
            md = m;
            m.findBB();
            m.compress();
            float xw = m.xmax - m.xmin;
            float yw = m.ymax - m.ymin;
            float zw = m.zmax - m.zmin;
            if (yw > xw)
                xw = yw;
            if (zw > xw)
                xw = zw;
            float f1 = size().width / xw;
            float f2 = size().height / xw;
            xfac = 0.7f * (f1 < f2 ? f1 : f2) * scalefudge;
        } catch(Exception e) {
            md = null;
            message = e.toString();
        }
        try {
            if (is != null)
                is.close();
        } catch(Exception e) {
```



```java
        }
        repaint();
    }
    public void start() {
        if (md == null && message == null)
            new Thread(this).start();
    }
    public void stop() {
    }
    public boolean mouseDown(Event e, int x, int y) {
        prevx = x;
        prevy = y;
        return true;
    }
    public boolean mouseDrag(Event e, int x, int y) {
        tmat.unit();
        float xtheta = (prevy - y) * 360.0f / size().width;
        float ytheta = (x - prevx) * 360.0f / size().height;
        tmat.xrot(xtheta);
        tmat.yrot(ytheta);
        amat.mult(tmat);
        if (painted) {
            painted = false;
            repaint();
        }
        prevx = x;
        prevy = y;
        return true;
    }
    public void paint(Graphics g) {
        if (md != null) {
            md.mat.unit();
            md.mat.translate(-(md.xmin + md.xmax) / 2,
                             -(md.ymin + md.ymax) / 2,
                             -(md.zmin + md.zmax) / 2);
            md.mat.mult(amat);
            //      md.mat.scale(xfac, -xfac, 8 * xfac / size().width);
            md.mat.scale(xfac, -xfac, 16 * xfac / size().width);
            md.mat.translate(size().width / 2, size().height / 2, 8);
            md.transformed = false;
            md.paint(g);
            setPainted();
        } else if (message != null) {
            g.drawString("Error in model:", 3, 20);
            g.drawString(message, 10, 40);
```



```java
        }
    }
    private synchronized void setPainted() {
        painted = true;
        notifyAll();
    }
    //    private synchronized void waitPainted() {
        //      while (!painted)
            //      wait();
        //      painted = false;
        //    }
}
```

Next Lecture

Guest: Guy Jacobson, Bell Labs

Data Compression (`compress`, `zip`, *etc.*)