GRAPHICAL CONVOLUTION IN JAVA

Erik S. Wheeler

EE 4012 -- Senior Design Project
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, Mississippi 39762
wheeler@isip.msstate.edu

ABSTRACT

Convolution is a concept that escapes many undergraduate engineering students. For years we have been forced to try and visualize the process with only limited success, but, with the advent of the World Wide Web and programming languages like Java, a tool for performing graphical convolution for anyone with an internet connection is now possible. Imagine wondering what the convolution of a Pi and a triangle function is and going to your computer to see it performed right before your eyes. This tool will not solve all of the problems of learning or teaching convolution but it should help some.

1. INTRODUCTION

A linear system is often described by the output obtained when an impulse is placed on the input. This is called the impulse response of that system, denoted h(t). Using this impulse response, the expected output of the system to any input may be determined by convolving that input with the system's impulse response. This process is represented by the convolution integral:

$$y(t) = \int_{-\infty}^{\infty} x(\lambda)h(t-\lambda)d\lambda$$

Convolution is often taught using a graphical convolution approach, in which the impulse response or input signal is reversed and moved to the left until there is no overlap of the two curves. The reversed curve is then slid to the right, positive time, across the other and the

area of the product of the overlapping curves is calculated.

This sliding and summing of curves is often hard for students to visualize, so a tool that can perform this animation would probably help in understanding the process.

This tool would have to allow the user to draw the input and impulse response curves, then show the result of the convolution as the impulse response is moved across the input, or visa versa.

2. IMPLEMENTATION APPROACHES

There were several possible implementation approaches considered, including Matlab, C or some other standard programming language, and Java.

Matlab would have been very convenient in that it has a built in convolution routine and supports limited animation, but it could only be used by people who have access to Matlab and have a basic knowledge of Matlab programming. Matlab would also require multiple copies of the code for individuals without accounts on the MSU Electrical Engineering server.

A standard programming language would similarly require multiple copies of the code. It would also not be platform independent. Although the code should be able to be run on any machine it is compiled on, this is often not the case. A completely platform independent program could be run on any computer architecture.

This left the immerging object-oriented programming language called Java. Java is platform independent because the compiler generates bytecode instructions that are accepted equally on any computer. It only requires one copy of the executable and Java is currently the only way to provide animation to web pages. This last fact is the reason it is so popular right now. This means that a program written in Java and put on a homepage would

be accessible to anyone with an internet connection.

3. JAVA OVERVIEW

According to “The Java Language, A White Paper,” Java is “A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language” [2]. But, what does this really mean?

Like the html language, Java is largely intended to be a program-by-example language in which the developer finds some suitable code and incorporates it into his or her program. This requires that the language be relatively simple.

What the developers meant by simple is that Java is based on C++ and has a small interpreter and class support. The entire language takes up about 40 kilobytes of memory, but to someone who has never written in an object-oriented programming language, Java may not appear all that simple.

The term “object-oriented”, along with “surfing”, may be one of the most overused phrases in the computer community, but for Java, object-oriented is not just another empty buzzword. Java was designed from the beginning as an object-oriented language. Unlike C++, which found its origins in C, Java was not required to conform to many of the non-object-oriented aspects of C.

What Sun means by distributed is that Java has a set of built-in classes that allow communication over the internet via URL’s. This feature is the main reason for Java’s huge popularity.

Interpreted means that the programming environment does not go through the standard compile and link process. Instead, it interprets the code to bytecodes which can execute directly on any machine. According to Sun, this should allow for faster development of usable code.

To avoid common problems associated with the pointer and pointer arithmetic used in C, the entire ability to use pointers is eliminated from the Java language. The memory management in Java is also completely handled by the language itself, with no chance of code related errors. These two features add to the language's robustness.

The developers of Java spent extensive amounts of time making sure that Java would be a secure tool for communication over the internet. There would be untold havoc if someone were able to write a Java virus that contaminates computer systems via their homepages. Therefore, Java has a set of built-in restrictions that attempt to circumvent such acts. One of the main restrictions is the ability of Java applets, an applet is the term used to describe Java programs intended for internet use, to write only to the file system of the host computer. Although a hinderance, it is easy to understand why this is an essential feature.

The reason Java is perfect for the internet is the fact that it is architecture neutral. That is, a compiled Java program can be run on any machine with the Java run-time environment. It does this by generating bytecode instructions which are handled equally in any computer architecture. Architecture neutrality is a key feature to being a portable language. To be truly portable, there can be no implementation dependant aspects of the language. In other words, an integer is always the same length on any machine.

The developers at Sun claim that Java performance is comparable to C or C++, but anyone who has run a Java program may disagree with this claim. In its base form as an independent application, Java may be as fast as C++, but when running Java applets on html pages, Java can be mind-numbingly slow.

If you have ever run more than one Netscape window at a time, you know what multithreaded means. Java is a multithreaded

language in that several separate processes can occur simultaneously and completely independent of each other.

The Java language and run-time system are dynamic during linking. Code is only linked when necessary and updating of programs may be performed transparently without ever requiring upgrades for the user.

4. JAVA ENVIRONMENT

Sun provides the Java Development Kit (JDK v1.0) free for anyone interested in programming in Java. The Java Development Kit, which was still in beta release at the beginning of this project, contains all of the necessary tools for developing Java programs. Some of the more important features of the package are the Java language compiler, class libraries, an applet viewer, and a debugger.

The debugger turned out to be almost worthless, and after consultation with other new Java programmers who agreed that the debugger itself was so full of bugs that it would cause more problems than solve, I decided to scrap it and debug the old fashioned way using the text editor. Borland has put out a graphical Java debugger, but it was not used, largely due to my lack of funds.

Sun also provides two very valuable programming aids in the Java language tutorial [1] and the Java API documentation [3], both available at the Java web site, java.sun.com. The API documentation contains a list of all of the standard classes and the methods and variables of those classes. Therefore, if any information about the interaction between these classes and their methods was desired, it could be found by checking this API.

Like many hot topics in the technical and non-technical world today, Java has its own newsgroup, comp.lang.java. Several of the programming problems I encountered were solved by helpful subscribers to this newsgroup. The book, *Teach Yourself Java in*

21 Days [4], also came highly recommended by the newsgroup readers and was my main source of information while learning the Java language.

5. DESIGN PROCESS

The design process was taken one step at a time in order to assure that something would be working by the end of the semester. This process went through several steps from the most basic considerations of discrete convolution to the addition of a graphical user interface.

5.1. Discrete Convolution Considerations

The basic element used in Java animation is the screen pixel. Using this as a discrete increment of time and amplitude, several parameters for discrete convolution could be determined, including time and amplitude scaling factors.

A width of 50 pixels for the time unit and 50 pixels for the amplitude unit was deemed appropriate. This allowed for adequate resolution while still leaving room on the screen. The arrays used to represent the input and impulse response curves would be arrays of integers in which each element represented a pixel location. This definition gave rise to the need for scaling of the input and impulse response signals during the convolution routine in order to attain proper output width and amplitude.

If the scaling factors are not taken into consideration, as was the case in my first attempt, the convolution of two square waves 50 pixels high and 50 pixels long would result in a triangle wave 50^3 pixels high.

5.2. Convolution Animation

The next step was to display the animated convolution of the two curves and updating the result as one curve is moved across the other. Several problems resulted from this endeavour, including determining the requirements and restrictions of displaying animation in Java. In order to reduce flicker a technique called

double buffering was used. Double buffering is the process in which screen images are first painted in memory on an off-screen “canvas”, then displayed when needed. The representation of the moving, time reversed input signal was displayed in this manner. Unfortunately, if one canvas is placed on top of another, the first canvas displayed is no longer visible, so another approach was required for the displaying of the overlapping curves. This was accomplished using the polygon drawing method of the Java graphics class. The polygon representation, which displays multiple line segments based on array elements passed to it, was also helpful in displaying the updated output signal, since the amount of the polygon displayed could easily be manipulated.

5.3. Drawing Tool

Once the problems with displaying the animation were solved, the drawing tool had to be created. Sami Shaio’s drawing demo, DrawTest.java v1.14, provided with the JDK, was used as a basis for the drawing tool. The program had to be modified to not allow infinite slope curves or defining two points for the same time increment. It also had to convert the curves drawn into pixel locations to be stored in their appropriate arrays. The drawing areas were then labeled to indicate axes and other important information.

5.4. Linking

After the animation program and the drawing tool were working, they had to be linked together. This link had to communicate the animation start command, the contents of the arrays representing the curves, and the command to pause the animation. Once the basic mechanism for communicating between applets was established, the rest was trivial.

5.5. User Interface

The final step in developing the tool was the addition of user interface. This included the buttons to allow activities such as clearing curves, pausing animation, and a group of

predefined curves buttons for some hard to draw functions.

The manner in which the convolution program was written did not allow for easy addition of user interface, so all buttons and checkboxes had to be implemented in the drawing tool, then passed to the animation program.

6. VERSION 1.0 FEATURES

The predefined curves described above include a cosine wave, an exponential decay, a one minus exponential decay, an approximation of an impulse, and a Sinc function.

The freehand drawing tool allows all of the considerations listed earlier plus the ability to draw outside the bounds of the drawing area. This feature is useful for approximating impulses of unit area without making them too wide. That is, an amplitude much higher than the drawing limits can be obtained by releasing the mouse button outside of the drawing limits.

The convolution area receives the curve information in the form of an array, computes the convolution and displays the animation loop of the graphical convolution process. If any changes are made, like new input or impulse response curves, or a request for a pause in the animation, the applet responds accordingly.

7. EVALUATION

All tests performed on the program using the Solaris or Sun versions of Netscape 2.0 were successful without any problems, but when run on a Pentium 120 computer for the demonstration, the convolution area would often not display correctly. The source of this bug was not able to be determined, but may be due to either a bug in the release version of Netscape 2.0 or in the convolution applet itself. Clearing the cache and reloading the applet would usually alleviate this problem.

8. SUMMARY

The program accomplished all objectives set forth in the initial scope of the project. The

latest version of the program may be tested by connecting to the Institute for Signal and Information Processing (ISIP) homepage, at http://isip.msstate.edu/, and clicking on “Fun Stuff”. The code is available for viewing or copying at the same address.

REFERENCES

1 M. Campione and K. Walrath, “The Java Language Tutorial: Object-Oriented Programming for the Internet,” available at http://java.sun.com/, Sun Microsystems, January,1996.

2 J. Gosling and H. McGilton, “The Java Language Environment: A White Paper,” available at http://java.sun.com/, Sun Microsystems, 1996.

3 Java API Documentation, available at http://java.sun.com/, Sun Microsystems, 1996.

4 L. Lemay and C.L. Perkins, *Teach Yourself Java in 21 Days*, Sams.net, Indianapolis, IN 1996.

APPENDIX A: CONVOLUTION.JAVA

```
/*
 *    Convolution.java
 *    Erik S. Wheeler
 *    Mississippi State University
 *    EE 4012 -- Senior Design Project
 *
 *    Graphical Convolution in Java
 *
 */

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

public class Convolution extends java.applet.Applet implements Runnable {
  final int resol = 1;
  final int timeres = 2;
  final static int maxwidth  = DrawCurves.maxwidth;
  final static int maxheight = DrawCurves.maxheight;
  final int pstart = 150;
  final double divconst     = DrawCurves.divconst;
  final double timedivconst = DrawCurves.timedivconst;

  boolean pauseFlag = false;
  boolean startAnimation = false;
  boolean newCurves = false;
  int quarterwidth = (int)(maxwidth/4);
  int pend = pstart + maxwidth;
  int l1 = pstart + quarterwidth;      // 1/4 thru
  int l2 = pstart + 2*quarterwidth;    // 1/2 thru
  int l3 = pstart + 3*quarterwidth;    // 3/4 thru
  int xarray[] = new int[(2*quarterwidth)];
  int harray[] = new int[(2*quarterwidth)];
  int yarray[] = new int[maxwidth];
  int revxarray[] = new int[(2*quarterwidth)];
  int hposh[] = new int[(2*quarterwidth)];
  int vposh[] = new int[(2*quarterwidth)];
  int hposy[] = new int[maxwidth];
  int vposy[] = new int[maxwidth];
  double fxarray[] = new double[(2*quarterwidth)];
  double fharray[] = new double[(2*quarterwidth)];
  double fyarray[] = new double[maxwidth];
  int currtime;

  Image RevXImg, fullImg;
  Graphics osG;
  Thread runner;

  public Convolution() {
  }
```

```
public void init() {
  RevXImg=createImage(2*quarterwidth, maxheight);
  fullImg = createImage(4*maxwidth, 4*maxheight);
}

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

public void stop() {
  if (runner != null) {
    runner.stop();
    runner = null;
  }
}

public void run() {
  int i;

  osG = fullImg.getGraphics();

  setBackground(Color.white);

  while (true) {
    if (startAnimation) {
                createX();
                createH();
                createRevX();
                createY();
                newCurves = false;

                while (!newCurves) {
                  for (currtime=pstart;
                      currtime<(pend-timeres);currtime+=timeres) {
                    repaint();
                    if (newCurves) currtime=pend;
                    while(pauseFlag) pause(500);
                    pause(100);
                  }
                  pause(1000);
                }
    }
    else {
                pause(1000);
    }
  }
}
```

```
 void updateParam(int[] xa,
                  int[] ha) {
  startAnimation = true;
  newCurves = true;
  xarray = xa;
  harray = ha;
 }

 void createX() {
  int i;

  for (i = 0; i < 2*quarterwidth; i++) {
   fxarray[i] = xarray[i] / divconst;
  }
 }

 void createH() {
  int i;

  for (i = 0; i < 2*quarterwidth; i++) {
   fharray[i] = harray[i] / divconst;
   hposh[i] = l1 + i;
   vposh[i] = 20 + maxheight/2 - harray[i];
  }
 }

 void createRevX() {
  int i;
  Graphics offscreenG = RevXImg.getGraphics();

  offscreenG.clearRect(0,0,2*maxwidth,maxheight);
  offscreenG.setColor(Color.blue);

  for (i=0;i<(2*quarterwidth);i++) {
   revxarray[i] = xarray[(2*quarterwidth-i-1)];
  }

  for (i=0;i<(2*quarterwidth-resol);i+=resol) {
   offscreenG.drawLine(i,maxheight/2-revxarray[i],
                       i+resol,maxheight/2-revxarray[(i+resol)]);
  }
  offscreenG.setColor(Color.red);
  offscreenG.drawLine(quarterwidth,maxheight/2-6,quarterwidth,maxheight/2+6);
 }

 void createY() {
  int i;

  Convolve();

  for (i=0;i<maxwidth;i++) {
   hposy[i] = i+pstart;
   vposy[i] = 20 + 2*maxheight - yarray[i];
  }
```

```
}

public void Convolve() {
   int i,j;
 for (i = 0;i < maxwidth;i++) fyarray[i]=0.0;
 for (i = 0;i < (2*quarterwidth);i++) {
   for (j = 0;j < (2*quarterwidth);j++) {
              fyarray[i+j]=fyarray[i+j]+(fxarray[i]*fharray[j]);
   }
 }
 for (i = 0;i<maxwidth;i++) {
   fyarray[i] = fyarray[i] / timedivconst;
   yarray[i] = (int)(fyarray[i] * divconst);
 }
}

public void pause(int time) {
 try { Thread.sleep(time); }
 catch (InterruptedException e) { }
}

public void update(Graphics g) {
 g.clipRect(0,0,2*maxwidth,4*maxheight);
 prePaint();
 paint(g);
}

void prePaint() {
 Font f = new Font("TimesRoman",Font.PLAIN,10);
 int i;

 osG.setFont(f);
 osG.clearRect(0,0,2*maxwidth,4*maxheight);

 osG.setColor(Color.red);
 osG.drawImage(RevXImg,(currtime-quarterwidth),20,this);

 osG.drawLine(pstart,20+maxheight/2,pend,20+maxheight/2);
 osG.drawLine(l2,20,l2,20+maxheight);

 // ***** TIME *****
 osG.drawLine(pstart,20+2*maxheight,pend,20+2*maxheight);
 for (i=pstart;i<=pend;i+=timedivconst) {
  osG.drawLine(i,16+2*maxheight,i,24+2*maxheight);
  osG.drawString(String.valueOf((int)((i-l2)/timedivconst)),i,14+2*maxheight);
 }

 // ***** AMPLITUDE ******
 osG.drawLine(l2,60+maxheight,l2,3*maxheight);
 for (i=(int)(20+2*maxheight+divconst);i<=3*maxheight;i+=divconst) {
  osG.drawLine(l2-4,i,l2+4,i);
  osG.drawString(String.valueOf((int)((20+2*maxheight-i)/divconst)),
                l2+6,i);
 }
```

```
    for (i=(int)(20+2*maxheight-divconst);i>=60+maxheight;i-=divconst) {
      osG.drawLine(l2-4,i,l2+4,i);
      osG.drawString(String.valueOf((int)((20+2*maxheight-i)/divconst)),
                     l2+6,i);
    }

    osG.setColor(Color.blue);
    osG.drawPolygon(hposy,vposy,currtime-pstart);
    osG.drawPolygon(hposh,vposh,vposh.length);
  }

  public void paint (Graphics g) {
    g.drawImage(fullImg,0,0,this);
  }

  public boolean handleEvent (Event e) {
    switch(e.id) {
    case Event.WINDOW_DESTROY:
      System.exit(0);
      return true;
    default:
      return false;
    }
  }
  public static void main (String args[]) {
    Frame f1 = new Frame("Convolution");
    Convolution conv = new Convolution();
    conv.init();
    conv.start();
    f1.add(conv);
    f1.resize(2*maxwidth, 4*maxheight);
    f1.show();
  }
}
```

APPENDIX B: DRAWCURVES.JAVA

```
/*
 *    DrawCurves.java
 *    Erik S. Wheeler
 *    Mississippi State University
 *    EE 4012 -- Senior Design Project
 *
 *    Graphical Convolution in Java
 *
 * DrawCurves and DrawPanel are modified versions of:
 *** @(#)DrawTest.java1.14 95/09/01 Sami Shaio                    ***
*** Copyright (c) 1994-1995 Sun Microsystems, Inc. All Rights Reserved. ***
 *
 *
 */
import java.lang.Math;
import java.awt.*;
import java.applet.*;
import java.util.Vector;

public class DrawCurves extends java.applet.Applet {
  public static final int maxwidth  = 600;
  public static final int maxheight = 200;
  public static final double divconst     = 50.0;
  public static final double timedivconst = 50.0;
  public DrawPanelArea xdpa;
  public DrawPanelArea hdpa;

  public void init() {
    Panel panelC = new Panel();
    xdpa = new DrawPanelArea();
    hdpa = new DrawPanelArea();

    setLayout(new BorderLayout());
    add("Center", panelC);

    panelC.setLayout(new GridLayout(1,2,5,5));
    panelC.add(xdpa);
    panelC.add(hdpa);

    add("South", new ControlPanel(this));
  }

  public boolean handleEvent (Event e) {
    switch(e.id) {
    case Event.WINDOW_DESTROY:
      System.exit(0);
      return true;
    default:
      return false;
    }
  }
```

```
  void startOverX () {
    xdpa.dp.lines.removeAllElements();
    xdpa.dp.xlast = 0;
    xdpa.dp.ylast = xdpa.dp.zeroY;
    for (int i=0;i<xdpa.dp.curveArray.length;i++) xdpa.dp.curveArray[i]=0;
    xdpa.dp.repaint();
  }

  void startOverH () {
    hdpa.dp.lines.removeAllElements();
    hdpa.dp.xlast = 0;
    hdpa.dp.ylast = hdpa.dp.zeroY;
    for (int i=0;i<hdpa.dp.curveArray.length;i++) hdpa.dp.curveArray[i]=0;
    hdpa.dp.repaint();
  }

  void execAnimation () {
    // get a hold of the receiver applet (Convolution)
    Convolution receiver=(Convolution)getAppletContext().getApplet("receiver");

    // update parameters and start animation of Convolution
    receiver.updateParam(xdpa.dp.curveArray, hdpa.dp.curveArray);
  }

  void handlePause (boolean pauseState) {
    Convolution receiver=(Convolution)getAppletContext().getApplet("receiver");
    receiver.pauseFlag = pauseState;
  }

  public static void main (String args[]) {
    Frame f1 = new Frame("Draw Curves");
    DrawCurves dCurves = new DrawCurves();
    dCurves.init();
    dCurves.start();
    f1.add(dCurves);
    f1.resize(maxwidth+20, maxheight+20);
    f1.show();
  }
}

class DrawPanelArea extends Panel {
  DrawPanel dp;

  public DrawPanelArea() {
    dp = new DrawPanel();
    setBackground(Color.white);
    setLayout(new BorderLayout());
    add("Center",dp);
    add("South", new DrawPanelControls(this));
  }
}
```

```
class DrawPanel extends Panel {
  final int maxwidth  = DrawCurves.maxwidth; // Class Variables
  final int maxheight = DrawCurves.maxheight;
  final double divconst     = DrawCurves.divconst;
  final double timedivconst = DrawCurves.timedivconst;
  final int zeroY = (int)(maxheight / 2);
  final int zeroX = (int)(maxwidth / 4);

  int curveArray[] = new int[(int)(maxwidth/2)];
  // no need to init array b/c values default to zero
  Vector lines = new Vector();
  int x1, y1;
  int x2, y2;
  int xl, yl;
  int xlast = 0;
  int ylast = zeroY;


  public DrawPanel() {
    setBackground(Color.white);
  }

  public boolean handleEvent(Event e) {
    switch(e.id) {
    case Event.MOUSE_DOWN:
      x1 = xlast;
      y1 = ylast;
      x2 = -1;
      return true;
    case Event.MOUSE_UP:
      if (e.x > x1) {
                    xlast = e.x;
      } else {
                    xlast = x1 + 1;
      }
      x2 = xl = -1;
      ylast = e.y;
      lines.addElement(new Rectangle(x1, y1, xlast, ylast));
      fillArray();
      repaint();
      return true;
    case Event.MOUSE_DRAG:
      if (e.x > x1) {
                    x2 = e.x;
      } else {
                    x2 = x1 + 1;
      }
      xl = x2;
      yl = y2;
      y2 = e.y;
      repaint();
      return true;
    case Event.WINDOW_DESTROY:
      System.exit(0);
```

```
    return true;
  default:
    return false;
  }
 }

 public void paint(Graphics g) {
  Font f = new Font("TimesRoman",Font.PLAIN,10);
  int i;
  int np = lines.size();

  g.setFont(f);
  g.setColor(Color.blue);

  // AMPLITUDE
  for (i=0;i<=maxheight;i+=divconst) {
    g.setColor(Color.lightGray);
    g.drawLine(0,i,maxwidth,i);
    g.setColor(Color.blue);
    g.drawLine(zeroX-4,i,zeroX+4,i);
    //      g.drawString(String.valueOf((int)((zeroY-i)/divconst)),zeroX+3,i+10);
  }

  // TIME
  for (i=0;i<=maxwidth;i+=timedivconst) {
    g.setColor(Color.lightGray);
    g.drawLine(i,0,i,maxheight);
    g.setColor(Color.blue);
    g.drawLine(i,zeroY-4,i,zeroY+4);
    //      g.drawString(String.valueOf((int)((i-zeroX)/timedivconst)),i+3,zeroY+10);
  }

  g.drawLine(0,zeroY, maxwidth, zeroY);
  g.drawLine(zeroX,0,zeroX,maxheight-1);

  // draw the current lines
  g.setColor(getForeground());
  g.setPaintMode();
  for (i=0; i<np; i++) {
    Rectangle p = (Rectangle)lines.elementAt(i);
    g.setColor(Color.red);
    if (p.width != -1) {
                g.drawLine(p.x, p.y, p.width, p.height);
    } else {
                g.drawLine(p.x, p.y, p.x, p.y);
    }
  }
  g.setXORMode(getBackground());
  if (xl != -1) {
    // erase last line
    g.drawLine(x1, y1, xl, yl);
  }
  g.setColor(getForeground());
  g.setPaintMode();
```

```
  if (x2 != -1) {
    g.drawLine(x1, y1, x2, y2);
  }
 }

 void fillArray() {
  double slope = (double)(y1 - ylast) / (double)(xlast - x1);
  for (int i=x1; i<=xlast; i++) {
    if ((i>=0)&&(i<(int)(maxwidth/2))) {
                curveArray[i]=(int)(slope*(i-x1)+(zeroY-y1));
    }
    else {
                // ERROR !!!!
    }
  }
 }

 void drawSine() {
  int i;

  for (i=0;i<(maxwidth/2);i++) {
    curveArray[i] = (int)(divconst *
                  Math.cos((double)(2*Math.PI*i/timedivconst)));
  }

  lines.removeAllElements();
  for (i=0;i<(maxwidth/2-1);i++) {
    lines.addElement(new Rectangle(i, zeroY-curveArray[i],
                   i+1, zeroY-curveArray[i+1]));
  }
  xlast = maxwidth/2;
  repaint();
 }

 void drawSinc() {
  int i;

  for (i=0;i<zeroX;i++) {
    curveArray[i] = (int)(divconst *
                  (Math.sin((double)(2*Math.PI*(i-zeroX)/timedivconst))) /
                  (2*Math.PI*(i-zeroX) / timedivconst));
  }
  curveArray[zeroX] = (int)divconst;
  for (i=zeroX+1;i<(maxwidth/2);i++) {
    curveArray[i] = (int)(divconst *
                  (Math.sin((double)(2*Math.PI*(i-zeroX)/timedivconst))) /
                  (2*Math.PI*(i-zeroX) / timedivconst));
  }
  lines.removeAllElements();
  for (i=0;i<(maxwidth/2-1);i++) {
    lines.addElement(new Rectangle(i, zeroY-curveArray[i],
                   i+1, zeroY-curveArray[i+1]));
  }
  xlast = maxwidth/2;
```

```
  repaint();
 }


 void drawExp() {
  int i;

  for (i=0;i<zeroX;i++) curveArray[i] = 0;
  for (i=zeroX;i<(maxwidth/2);i++) {
   curveArray[i] = (int)(divconst *
                    Math.exp((double)((zeroX-i)/timedivconst)));
  }

  lines.removeAllElements();
  for (i=0;i<(maxwidth/2-1);i++) {
   lines.addElement(new Rectangle(i,zeroY-curveArray[i],
                    i+1,zeroY-curveArray[i+1]));
  }
  xlast = maxwidth/2;
  repaint();
 }

 void draw1_Exp() {
  int i;

  for (i=0;i<zeroX;i++) curveArray[i] = 0;
  for (i=(int)maxwidth/4;i<(maxwidth/2);i++) {
   curveArray[i] = (int)(divconst *
                   (1-Math.exp((double)((zeroX-i)/timedivconst))));
  }

  lines.removeAllElements();
  for (i=0;i<(maxwidth/2-1);i++) {
   lines.addElement(new Rectangle(i,zeroY-curveArray[i],
                    i+1,zeroY-curveArray[i+1]));
  }
  xlast = maxwidth/2;
  repaint();
 }

 void drawImpulse() {
  int i;

  for (i=0;i<maxwidth/2;i++)
   curveArray[i] = 0;
  for (i=(int)(maxwidth/4-2);i<=maxwidth/4+2;i++)
   curveArray[i] = 500;

  lines.removeAllElements();
  lines.addElement(new Rectangle(0,zeroY,(int)(maxwidth/4-2),zeroY));
  lines.addElement(new Rectangle((int)(maxwidth/4-2),zeroY,
                   (int)(maxwidth/4-2),0));
  lines.addElement(new Rectangle((int)(maxwidth/4-2),0,
                   (int)(maxwidth/4+2),0));
```

```
    lines.addElement(new Rectangle((int)(maxwidth/4+2),0,
                    (int)(maxwidth/4+2),zeroY));
    lines.addElement(new Rectangle((int)(maxwidth/4+2),zeroY,
                    (int)(maxwidth/2),zeroY));
    xlast = maxwidth/2;
    repaint();
   }

}

class DrawPanelControls extends Panel {
  DrawPanelArea target;

  public DrawPanelControls(DrawPanelArea target) {
    this.target = target;

    setLayout(new FlowLayout(FlowLayout.CENTER));
    setBackground(Color.lightGray);
    target.setForeground(Color.red);
    add (new Button("Cos(2*pi*t)"));
    add (new Button("exp(-t)"));
    add (new Button("1-exp(-t)"));
    add (new Button("Impulse"));
    add (new Button("Sinc"));
  }

  public void paint(Graphics g) {
    Rectangle r = bounds();

    g.setColor(Color.lightGray);
    g.draw3DRect(0,0,r.width, r.height, false);
  }

  public boolean action (Event e, Object arg) {
    if (e.target instanceof Button) {
      if (((String)arg).equals("Cos(2*pi*t)")) {
                    target.dp.drawSine();
      } else if (((String)arg).equals("exp(-t)")) {
                    target.dp.drawExp();
      } else if (((String)arg).equals("1-exp(-t)")) {
                    target.dp.draw1_Exp();
      } else if (((String)arg).equals("Impulse")) {
                    target.dp.drawImpulse();
      } else if (((String)arg).equals("Sinc")) {
                    target.dp.drawSinc();
      } else {
                    // error
      }
    }
    return true;
  }
}
```

```
class ControlPanel extends Panel {
  DrawCurves target;

  public ControlPanel(DrawCurves target) {
    this.target = target;

    setLayout(new FlowLayout(FlowLayout.CENTER));
    setBackground(Color.lightGray);
    target.setForeground(Color.red);
    add (new Button("Clear Input Signal (Left)"));
    add (new Button("Start Animation"));
    add (new Button("Clear Impulse Response (Right)"));
    add (new Checkbox("Pause Animation", null, false));
  }

  public void paint(Graphics g) {
    Rectangle r = bounds();

    g.setColor(Color.lightGray);
    g.draw3DRect(0,0,r.width, r.height, false);
  }

  public boolean action(Event e, Object arg) {
    if (e.target instanceof Button) {
      if (((String)arg).equals("Start Animation")) {
                target.execAnimation();
      } else if (((String)arg).equals("Clear Input Signal (Left)")) {
                target.startOverX();
      } else if (((String)arg).equals("Clear Impulse Response (Right)")) {
                target.startOverH();
      } else {
                // Unexpected Error
      }
    } else if (arg instanceof Boolean) {
      if (((Checkbox)e.target).getLabel().equals("Pause Animation")) {
                target.handlePause(((Boolean)arg).booleanValue());
      }
    }
    return true;
  }
}
```