

California State University, San Bernardino

## CSUSB ScholarWorks

---

Theses Digitization Project

John M. Pfau Library

---

2001

### World Wide Graphics

Alysha Marie Timmons

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Digital Communications and Networking Commons](#), and the [Instructional Media Design Commons](#)

---

#### Recommended Citation

Timmons, Alysha Marie, "World Wide Graphics" (2001). *Theses Digitization Project*. 2089.  
<https://scholarworks.lib.csusb.edu/etd-project/2089>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact [scholarworks@csusb.edu](mailto:scholarworks@csusb.edu).

WORLD WIDE GRAPHICS

---

A Project  
Presented to the  
Faculty of  
California State University  
San Bernardino

---

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science  
in  
Computer Science

---

by  
Alysha Marie Timmons

June 2001

WORLD WIDE GRAPHICS

---

A Project  
Presented to the  
Faculty of  
California State University  
San Bernardino

---


by  
Alysha Marie Timmons


June 2001

Approved by:

  
George Georgiou, Computer Science

6/8/01  
Date

  
Richard Botting

  
Kerstin Voigt

© 2001 Alysha Marie Timmons

## ABSTRACT

The integration of computers and the Internet in the educational process of today has led to a need for distance presentation tools. There are software products that address this subject; however, they are costly or complicated. The development and implementation of this software product provides an easy to follow tool for instructors to capitalize on the popularity and availability of the Internet.

World Wide Graphics (WWG) is a software package that provides instructors with the tools needed to present a web-based presentation to a group of students while having the ability of enhancing the prepared HTML slide with user-drawn graphics and highlighting. The students have convenient access to the presentation with the use of an Internet browser, such as Internet Explorer. The instructor has complete control over the presentation and what is displayed to the students.

## ACKNOWLEDGMENTS

I would like to thank my friend and cooperator, Sandy Hengstebeck, for the many long hours, phone calls, e-mails, and brainstorming. To my advisor, Dr. Georgiou, I express special thanks for his time, patience, and expertise, for always being just a phone call or e-mail away, for being ready to listen and "advise", and most of all, for being Dr. Georgiou. In addition, I would like to thank my mother, Marie, and my sister, Anitra, for hours of searching through any book available for a helpful answer and for their unending love and support.

To Mom and Sis

## TABLE OF CONTENTS

|   |      |
|---|------|
| ABSTRACT .....  | iii  |
| ACKNOWLEDGMENTS .....                                   | iv   |
| LIST OF TABLES .....                                    | vii  |
| LIST OF FIGURES .....                                   | viii |
| CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION        |      |
| 1.1 Introduction .....                                  | 1    |
| 1.1.1 Background Research .....                         | 4    |
| 1.1.2 Purpose .....                                     | 7    |
| 1.1.3 Scope .....                                       | 7    |
| 1.1.4 Definitions, Acronyms,<br>and Abbreviations ..... | 7    |
| 1.1.5 Overview .....                                    | 9    |
| 1.2 Overall Description                                 |      |
| 1.2.1 Product Perspective .....                         | 10   |
| 1.2.2 Product Functions .....                           | 10   |
| 1.2.3 User Characteristics .....                        | 10   |
| 1.2.4 Constraints .....                                 | 11   |
| 1.2.5 Assumptions and Dependencies .....                | 11   |
| CHAPTER TWO: SOFTWARE DESIGN                            |      |
| 2.1 Preliminary Design Pseudocode .....                 | 12   |
| 2.2 Detailed Design .....                               | 14   |



|  |    |
|--|----|
| 2.2.1 Choice Panel Design .....  | 16 |
| 2.2.2 Drawing Panel Design .....   | 19 |
| CHAPTER THREE: SOFTWARE QUALITY ASSURANCE                                    |    |
| 3.1 Unit Test Plan .....   | 24 |
| 3.2 Integration Test Plan .....  | 26 |
| 3.3 System Test Plan .....   | 28 |
| CHAPTER FOUR: MAINTENANCE .....  | 30 |
| CHAPTER FIVE: USERS MANUAL .....   | 31 |
| CHAPTER SIX: CONCLUSION .....  | 33 |
| APPENDIX A: SOURCE CODE OF THE WORLD WIDE<br>GRAPHICS SYSTEM .....           | 35 |
| APPENDIX B: EXAMPLES OF THE WORLD WIDE GRAPHICS<br>GRAPHICAL INTERFACE ..... | 63 |
| REFERENCES .....   | 67 |

## LIST OF TABLES

|                                       |    |
|---------------------------------------|----|
| Table 1. Choice Panel .....           | 16 |
| Table 2. Function Action .....        | 17 |
| Table 3. Function SetSave .....       | 17 |
| Table 4. Function SetClear .....      | 18 |
| Table 5. Drawing Panel .....          | 19 |
| Table 6. Function DrawLine .....      | 20 |
| Table 7. Function DrawRectangle ..... | 21 |
| Table 8. Function DrawEllipse .....   | 22 |
| Table 9. Function SaveImg .....       | 23 |

## LIST OF FIGURES

|   |    |
|---|----|
| Figure 1. Class Diagram .....                                     | 14 |
| Figure 2. Overview of the Integration<br>of the Two Systems ..... | 15 |

# CHAPTER ONE

## SOFTWARE REQUIREMENTS SPECIFICATION

### 1.1 Introduction

Years ago, it was rare for a classroom to have one computer. Today, schools have computer labs that provide a computer for each student enrolled in the course. In addition, computers have become a product that is readily found in American households. Due to this evolution, the concept of telecommuting and televised classes is now more than a notion.

Many times, students may find it difficult to attend classes, due to circumstances such as time constraints, commuting costs, or health-related problems. With the availability of computers and the Internet, it is not a far-fetched idea that students could "attend" class from their own home or workplace.

Viewing a lecture through the web not only benefits those students who cannot attend, but also those who are in the classroom. Most teachers still lecture with the aid of chalkboards, whiteboards, and projector screens. In this situation, students at the back of the room may find it difficult to see what is being presented. During a

lecture, few teachers or students are able to utilize the computers made available. The student may actually find the computer a hindrance while trying to follow the lecture. The computer takes up desk space and obstructs the view of the front of the room. Additionally, teachers find computers to be a distraction for many students during lectures. If the teachers were able to present their lecture on the computer, this would alleviate some of the problems and utilize the computers. In addition, it would be a great benefit if the teacher could draw attention to key points in their presentation and know that students could easily distinguish important elements.

With so many presentation services on the web, this would seem to be a moot situation. However, there are some disadvantages to these services. The presenter is required to upload their files to a third party's site for the presentation. Once uploaded, no changes can be made to the files. Another major disadvantage is the fact that these services are expensive and dependent upon the number of viewers. Often, educational institutions and individual teachers are not financially capable of meeting these expenditures. Therefore, these services are no longer a viable solution.

World Wide Graphics (WWG) is a system that was built upon the application Presentations World Wide (PWW). [4] PWW is an application that provides educators an interface in which to run a web-based presentation. An instructor sets up the presentation using PWW. Then, when desired, the presentation begins using PWW. The instructor controls what slide to shown at what time. Anyone connected to the given URL will view exactly the same slide that the instructor has chosen.

World Wide Graphics adds to the PWW interface the tools to draw graphical shapes on the web page. During the lecture, the presenter can draw lines, rectangles, or ellipses of various colors. There is also the option to draw filled rectangles or ellipses, as well as lines of various thicknesses. All graphics utilize semi-transparent colors so as to allow any text or images beneath to show through. Then, with a click of a button, the presenter can post the drawn graphics to the web, so the students at the other end can see them.

World Wide Graphics is created using Java 2, an object-oriented programming language with extended graphical abilities, and the features of the Internet. WWG provides instructors and students an easy to use method

that makes lectures more effective. The instructor simply installs an executable program, which is started before launching a presentation. Students, on the other hand, are merely required to point their Internet browser to the given URL.

This project documentation explains the development of World Wide Graphics, a interactive tool that allows a presenter to draw the attention of a student to a particular element during a web-based presentation via graphical tools.

#### 1.1.1 Background Research

Research was done on the currently available web presentation services. It was found that all had advantages and disadvantages. Examples of presentation services on the web are WebEx [8], Astound Conference Center [1], and PlaceWare Web Conferencing [7].

PC Magazine chose WebEx as Editor's Choice in the December 17, 1999 issue.[6] The WebEx is a proprietary service that includes whiteboards, annotation tools, polling functions, chatting, and one-way presentations. If you only have up to four participants in your meeting and keep it within 30 minutes, then you can almost use it for free. The disadvantages of WebEx appear to be the

requirement of scheduling a conference for a one-time presentation. If the user has more than four attendees or the presentation is longer than 30 minutes, then there is a substantial associated cost. The lowest plan is \$25 per month for the first four users and \$10 for each additional user, with a \$.15 per user per minute cost. [8]

Astound Conference Center has services similar to WebEx; however, it does not offer free-form drawing tools. One advantage of Astound is the ability to meet immediately or schedule a meeting for later. A disadvantage of Astound is the fact that the links provided to the viewer are not live. The links appear in the viewer's browser; however, if the viewer clicks on the links, nothing will happen. Only the presenter can use them. The pricing requires a contract for a monthly rate. The shortest contract is a 3-month contract at \$25 per user. There is a one-time meeting available for \$.30 per minute per person if the meeting is less than 30 minutes. [1]

PlaceWare Web Conferencing offers most of the services that WebEx and Astound offer. It provides exceptional 3-D graphics. The presenter is also given the ability to add slides on the fly; however, it requires three separate downloads by the presenter and was indicated in PC Magazine



to be difficult to set up. [7] While trying to determine a price, on the low end, PlaceWare indicated that they have a separate service called MyPlaceWare that allows presentations with up to five attendees for free. The PlaceWare Web Conferencing Service is different from the other two web services, because it doesn't charge by the minute. It has a one-time set up expense of \$3,000 and a \$600 per person per year fee.

Other projects involving the Internet as a teaching resource have been developed in the past. Examples of past projects include WebCT [9], and Dohyon Donte Kim's Internet Instructional Aid [5]. However these products do not meet our specifications. This could be for several reasons.

WebCT is an aid to the instructor who wishes to set up an "on-line classroom information center". It provides a template for syllabi, homework assignments, discussion groups, quizzes, etc. However, it does not provide a method to deliver live presentations.

Donte Kim's Internet Instructional Aid (IIA) is a software package that provides "an environment in which students can have additional group and individual contact with the instructor and convenient access to prior lectures and instructional materials" [5]. It provides two applets.

One applet, cyboard, allows the instructor and students to draw in the applet and save the images created. It also allows the instructor and students to post questions using a chat program. However, it only displays images drawn on the whiteboard and does not open existing images or text files. The program is best used as an accompaniment to a lecture, as opposed to delivering a full-blown lecture.

#### 1.1.2 Purpose

The purpose of this section is to present the specifications of World Wide Graphics. WWG was created as the author's M.S. project.

#### 1.1.3 Scope

The software project that was produced is the World Wide Graphics system. WWG works in conjunction with the Presentation World Wide system to provide the presenter the ability to conduct a classroom lecture as a web presentation. While the presenter gives the presentation, he/she will be able to draw on and highlight the current slide of the presentation. Once the viewer has connected to the web site, via a browser, the viewer will not need to interact with the computer to view the presentation.

#### 1.1.4 Definitions, Acronyms, and Abbreviations

WWG: World Wide Graphics

PWW: Presentations World Wide

HTML: Hypertext Markup Language, a set of codes that can be inserted into text files to indicate special typefaces, inserted images, links to other documents, etc.

Java: A programming language developed to enable networked computers to transmit computations to each other, not just data. Java programs are compiled not into machine code, which would not be portable, but into concise code, known as Java byte code

Browser: A computer program that enables the user to read Hypertext in files or in the World Wide Web

Internet: A cooperative message forwarding system linking computer networks all over the world

Web Server: A computer that is attached to the Internet and contains web pages (HTML files) that can be viewed using a web browser

URL: Universal Resource Locator, a way of specifying the location of publicly available information on the Internet, in the form `http://www.csci.csusb.edu`

GUI: Graphical User Interface, a way of communicating with the computer by manipulating icons and windows with a mouse

Mouse: A computer input device that is used by moving it around and pressing one or more buttons

GIF: A file format for storing bitmap images

User/Presenter: The user of the system, generally an educator, who will create and run the presentation. This person will be in charge of advancing to the next slide during the presentation, as well as drawing any free-form graphics on the slides.

Viewer: The person, most likely a student, who will be viewing the presentation after it is created and posted to the web server.

#### 1.1.5 Overview

The remainder of the SRS pertains to the overall description of the product. It explains in detail the functions, characteristics, constraints, and dependencies related to the use of WWG. In addition, the requirements of the presenter and viewer using WWG are presented.

## 1.2 Overall Description

### 1.2.1 Product Perspective

The WWG system is a software program that works in conjunction with the PWW system. The product consists of a Java executable. It is required that Java is installed in order to run the executable. The presenter does not install the program files, since the program files do not need to be compiled in order to run. Only the executable is installed and ran. The viewers of the presentation are required to have Internet access, with a Java-enabled browser, such as Internet Explorer 5.0, Netscape 4.0, or better.

### 1.2.2 Product Functions

A professor creates and begins a presentation using the PWW interface. Each student will open a browser window points to the appropriate URL. Each will see the same page that the presenter is currently viewing. The presenter will be able to draw figures and highlight areas on the page as the presentation proceeds.

### 1.2.3 User Characteristics

In order to use World Wide Graphics, a presenter needs to have previously created and loaded the HTML pages to the web server. While running the presentation, the mouse will

be used (click and drag) to draw the desired graphics. When the mouse button is pressed, the beginning point of the graphic is set. The user then drags the mouse to indicate the desired size of the graphic. When the mouse button is released, the graphic is drawn.

#### 1.2.4 Constraints

In order to run WWG, the instructor must have access to a computer with Java capabilities. WWG is designed to communicate with input from a mouse connected to the computer running the program.

#### 1.2.5 Assumptions and Dependencies

In order for WWG to properly function, it is assumed that the server on which it is run has Java 1.3 installed. The presenter needs to have a Unix operating system installed in order to properly run PWW. It is also required that the persons viewing the presentation have access to an Internet browser, Internet Explorer 5.0, Netscape Navigator 4.0, or better.

CHAPTER TWO

SOFTWARE DESIGN

2.1 Preliminary Design Pseudocode

Create a visual panel in which to draw

Create buttons and menu to choose graphic shapes

Create two images in which to draw

    One for visual purposes

    One for future saving purposes

Create the interface to handle inputs from a mouse

If a shape is chosen

    Update the type of shape to draw

    If chosen to highlight

    Set the flag to highlight

    Set the shape to "Line"

    Set the color to yellow

    If mouse button is pressed

    Record the initial coordinates of the point chosen

    If mouse is dragged

        Continually update the coordinates of the next  
        point

        Draw the desired figure beginning at start point  
        and ending at this point

If mouse button is released

Record the final coordinates of the destination  
point

Draw the final figure beginning at start point  
and ending at this point

If the option to fill is chosen, then draw a  
filled shape.

If clear button is pressed

Clear (fill the entire image with the background  
color) both the shown graphics image and the  
image for saving

If save button is pressed

Call a class to translate the graphics to GIF  
format

Pass the image created for saving to the encoding  
class



## 2.2 Detailed Design

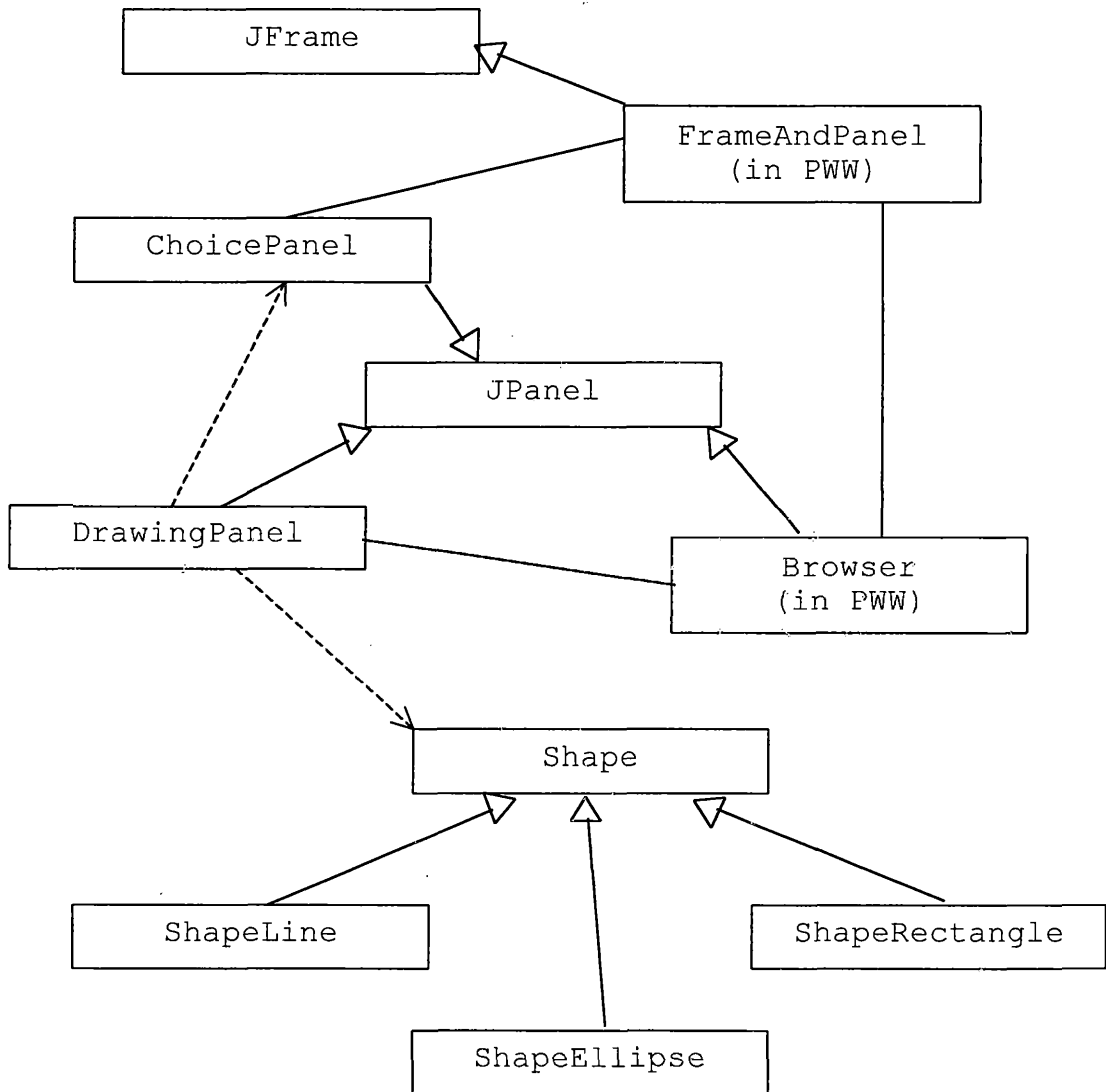


Figure 1. Class Diagram

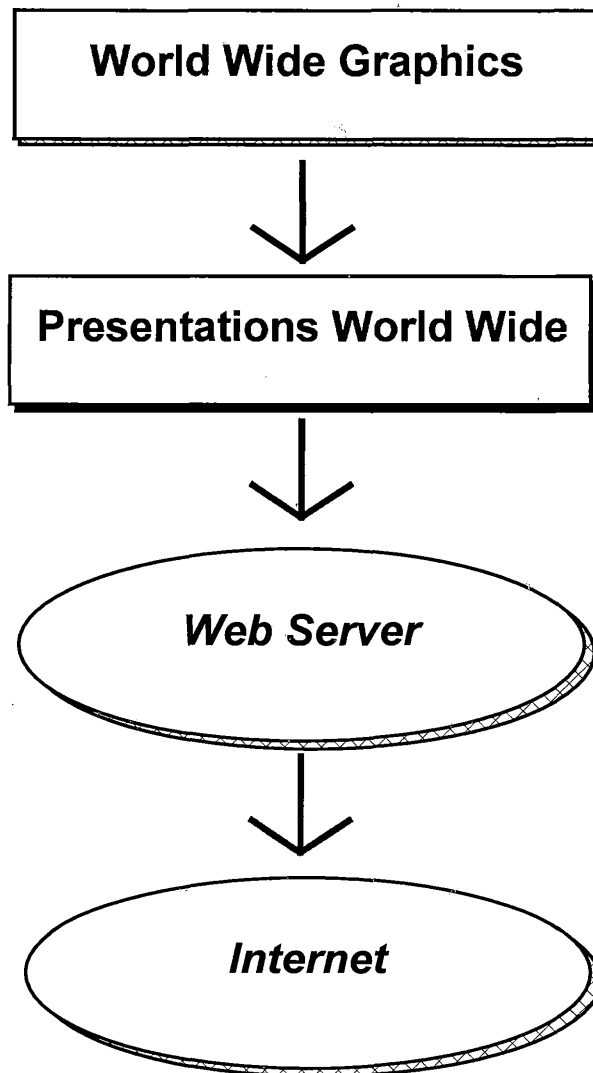


Figure 2. Overview of the Integration of the Two Systems

### 2.2.1 Choice Panel Design

Table 1. Choice Panel

|               |  |
|---------------|--|
| 1. Class Name | ChoicePanel  |
| Where Used    | WWG  |
| Purpose       | Button interface                                   |
| Sub Items     | Transmits user's decisions<br>to the Drawing Panel |
| Note          | Main button panel of WWG                           |

#### Procedure ChoicePanel

Begin

    Create panel for buttons

    Add choices for the shape

    Add choices for the color

    Add option to fill the shape (checkbox)

    Add option to highlight (checkbox)

    Add choices for the size of the highlight

    Add button for clearing

    Add button for saving

End

Table 2. Function Action

|                  |  |
|------------------|--|
| 2. Function Name | Action   |
| Member Of        | ChoicePanel  |
| Purpose          | To set the appropriate variables and flags based upon the user's input |
| Sub Items        | Transmits user's decisions to the DrawingPanel                         |

Procedure action

Begin

If any choice is made (the shape, color, fill,  
highlighting),

Update the decision variables in the DrawingPanel

End

Table 3. Function SetSave

|                  |   |
|------------------|---|
| 3. Function Name | setSave                                       |
| Member Of        | ChoicePanel                                   |
| Purpose          | To call the save function                     |
| Sub Items        | Transmits user's decision to the DrawingPanel |

Procedure setSave

Begin

Call the save function in the DrawingPanel

End

Table 4. Function SetClear

|                  |  |
|------------------|--|
| 4. Function Name | SetClear   |
| Member Of        | ChoicePanel                                      |
| Purpose          | To call the clear function                       |
| Sub Items        | Transmits user's decision to<br>the DrawingPanel |

Procedure setClear

Begin

Call the clear function in the DrawingPanel

End

### 2.2.2 Drawing Panel Design

Table 5. Drawing Panel

|               |   |
|---------------|---|
| 1. Class Name | Drawing Panel   |
| Where Used    | WWG   |
| Purpose       | To provide the panel for drawing and call the appropriate functions for drawing |
| Sub Items     | Calls the appropriate class to draw the graphic                                 |

#### Procedure DrawingPanel

Begin

Create the panel for drawing

Create two blank images for the graphics - One for display and one for saving

Initialize variables to draw black lines

Create method for executing mouse input

End

Table 6. Function DrawLine

|                  |   |
|------------------|---|
| 2. Function Name | drawLine                                      |
| Member Of        | DrawingPanel                                  |
| Purpose          | To draw the final line to the image displayed |
| Sub Items        | Draws image to saving image                   |

Procedure drawLine

Begin

    Get the coordinates of the beginning and end points

    If the option to highlight is chosen

        Change the Graphics to Graphics2D

        Set the Stroke size of the line

        Draw the line

    Else

        Draw the line

    End if

End

Table 7. Function DrawRectangle

|                  |   |
|------------------|---|
| 3. Function Name | DrawRectangle   |
| Member Of        | DrawingPanel  |
| Purpose          | To draw the final rectangle<br>to the image displayed |
| Sub Items        | Draws image to saving image                           |

Procedure drawRectangle

Begin

Get the coordinates of the beginning and end points

If the option to fill is chosen

Draw the filled rectangle

Else

Draw the outline of the rectangle

End if

End



Table 8. Function DrawEllipse

|                  |  |
|------------------|--|
| 4. Function Name | drawEllipse                                      |
| Member Of        | DrawingPanel                                     |
| Purpose          | To draw the final ellipse to the image displayed |
| Sub Items        | Draws image to saving image                      |

Procedure drawEllipse

Begin

    Get the coordinates of the beginning and end points

    If the option to fill is chosen

        Draw the filled ellipse

    Else

        Draw the outline of the ellipse

    End if

End

Table 9. Function SaveImg

|                  |   |
|------------------|---|
| 5. Function Name | saveImg   |
| Member Of        | DrawingPanel  |
| Purpose          | To save the image to a file   |
| Sub Items        | Converts the graphics image<br>to GIF format  |
| Note             | Code for the GifEncoder was<br>downloaded from<br><br>acme.com/java and is<br><br>Copyright (C)1996,1998 by<br><br>Jef Poskanzer<br><br><jef@acme.com>. |

Procedure saveImg

Begin

    Initialize the file to write to

    Convert the image to GIF format

    Save to file

End

CHAPTER THREE  
SOFTWARE QUALITY ASSURANCE

3.1 Unit Test Plan

In order to test the various units of the DrawingPanel, each type of shape will be individually tested. In addition, various color choices will be made to ensure that the colors are correctly drawn. The drawing of highlighting in various widths will, also, be tested.

|                  |   |
|------------------|---|
| Test Name/Number | 1 - Draw a line   |
| Test Objective   | Observe whether line is correctly drawn   |
| Test Description | Click in area, drag mouse, release<br>button  |
| Test Conditions  | Change color choice to another color  |
| Expected Results | Draws a line of chosen color from<br>initial point to end point                     |
| Actual Results   | Correctly drew line from beginning<br>point to end point, using color choice<br>red |
| Test Name/Number | 2 - Draw a Rectangle  |
| Test Objective   | Observe whether rectangle is correctly  |

|                  |   |
|------------------|---|
|                  | drawn   |
| Test Description | Click in area, drag mouse, release button   |
| Test Conditions  | Change color choice to another color  |
| Expected Results | Draws a rectangle of chosen color from initial point to end point                   |
| Actual Results   | Correctly drew rectangle from beginning point to end point, using color choice blue |
| Test Name/Number | 3 - Draw an ellipse   |
| Test Objective   | Observe whether ellipse is correctly drawn  |
| Test Description | Click in area, drag mouse, release button   |
| Test Conditions  | Change color choice to another color  |
| Expected Results | Draws an ellipse of chosen color from initial point to end point                    |
| Actual Results   | Correctly drew ellipse from beginning point to end point, using color choice green  |
| Test Name/Number | 4,5,6,7 - Highlighting  |

|                  |  |
|------------------|--|
| Test Objective   | Observe whether highlighting lines are correctly drawn                                 |
| Test Description | Click in area, drag mouse, release button  |
| Test Conditions  | Test using each choice of line size  |
| Expected Results | Correctly draws each size highlighting line from initial point to end point            |
| Actual Results   | Correctly drew lines from beginning point to end point, testing each of the four sizes |

### 3.2 Integration Test Plan

The integration of the various shapes available combined with the option to fill the shape creates possibility of even more test cases. In addition, the drawing of multiple images, overlapping will be tested.

|                  |   |
|------------------|---|
| Test Name/Number | 8 - Test the changing of the choice values  |
| Test Objective   | Observe whether choice of shape is automatically changed when option to highlight is chosen |
| Test Description | Choose to draw an ellipse, then click   |

|                  |   |
|------------------|---|
|                  | the highlight checkbox  |
| Test Conditions  | Ellipse is chosen first, then highlighting  |
| Expected Results | Automatically changes the shape choice to "Lines"   |
| Actual Results   | Correctly changes the shape choice from "Ellipse" to "Lines"  |
| Test Name/Number | 9 and 10 - Draw filled shapes   |
| Test Objective   | Observe whether filled shapes are correctly drawn   |
| Test Description | Click in area, drag mouse, release button   |
| Test Conditions  | Test using each choice of shape and two different colors  |
| Expected Results | Draws a filled rectangle and ellipse  |
| Actual Results   | Correctly drew a filled rectangle and a filled ellipse from beginning point to end point, using color choices red and green |
| Test Name/Number | 11 - Draw overlapping graphics  |
| Test Objective   | Observe whether various shapes are  |

|                  |   |
|------------------|---|
|                  | correctly drawn when overlapped   |
| Test Description | Click in area, drag mouse, release button. Change shape and color, repeat process while overlapping the previous image. |
| Test Conditions  | Test with all choices of shape as well as highlighting  |
| Expected Results | Draws the various images overlapped   |
| Actual Results   | Correctly drew all the shapes from beginning point to end point overlapping, using various colors                       |

### 3.3 System Test Plan

In order to test the entire system, the first thing to test is the ability of the system to correctly draw all of the various shapes in use, and then clear the screen. Then, an important element to test is the efficiency of saving the graphics to a file in the GIF format. Since the system will prospectively be used with both the Internet Explorer and the Netscape Navigator, the system will be tested with both browsers.

|                   |  |
|-------------------|--|
| Test Name/Number. | 12 - Test the clear button   |
| Test Objective    | Observe whether screen is correctly cleared after various graphics are drawn                                     |
| Test Description  | Click the clear button.  |
| Test Conditions   | Draw various shapes before clearing.   |
| Expected Results  | Draws the shapes. Then, when clear button is pressed, clears the screen of all of the previously drawn graphics. |
| Actual Results    | Correctly drew all of the shapes and erased the entire image when chosen to clear.                               |

|                  |   |
|------------------|---|
| Test Name/Number | 13 - Save image in GIF format   |
| Test Objective   | Observe whether image is correctly saved  |
| Test Description | Click the save button   |
| Test Conditions  | Draw various shapes before saving.<br>After saving, open file in browser to test if it is viewable. |
| Expected Results | Image is correctly saved and displayed.   |
| Actual Results   | Correctly saved and displayed the image.  |



## CHAPTER FOUR

### MAINTENANCE

Location: /pool/www/public/csci/atimmons/project/WWG.tar

Code and documentation on PWWS CD-ROM: On file in the  
Computer Science Department of California State University,  
San Bernardino, JB-307.

Compiler: JDK1.3 or higher

Library: JDK1.3 Packages

Operating System: Linux Kernel v2.0.27

Decompress file:

1. Place the file in the desired directory.
2. Create a directory called "WWG".
3. Change to the new directory.
4. Type "tar -xvf WWG.tar".

Running the executable:

1. Change to the WWG directory created in decompression.
2. Type "java WWG".

## CHAPTER FIVE

### USERS MANUAL

Create and begin a presentation using PWW with the WWG features.

To draw a shape:

1. Choose the desired shape ("Rectangles" or "Ellipses") from the "Shape:" drop-down menu.
2. Choose the desired color from the "Color:" drop-down menu.
3. If desired, choose the fill option by clicking the checkbox next to "Fill".
4. Click the mouse where you would like the beginning point of the shape.
5. Drag the mouse until you reach the desired size of shape.
6. Release the mouse button.

To draw a line:

1. Choose "Lines" from the "Shape:" drop-down menu.
2. Choose the desired color from the "Color:" menu.
3. Click the mouse where you would like the beginning point of the line.

4. Drag the mouse until you reach the desired length of the line.
5. Release the mouse button.

To highlight:

1. Choose to highlight by clicking the checkbox next to "Highlight". The shape is automatically changed to "Lines" and the color is changed to "Yellow".
2. If desired, change to a different color from the "Color:" drop-down menu.
3. Click the mouse where you would like the beginning point of the line.
4. Drag the mouse until you reach the desired length of the line. (When highlighting, lines are only drawn horizontally.)
5. Release the mouse button.

To clear the screen:

Click on the "Clear" button.

To save the image:

Click on the "Save" button.

## CHAPTER SIX

### CONCLUSION

With the World Wide Graphics and the Presentations World Wide systems, the instructor will no longer be limited by the standard methods of presenting material to a group of students. Instead of using chalkboards that can be messy, whiteboards that require special markers, or projector screens that are not always visible to the entire class, WWG allows the instructor to highlight text, point to certain areas via lines, or draw shapes for emphasis.

Although there are other presentation software products available, they are not commonly used by educational institutions. Reasons for the lack of use vary among the different software. Often times, the product uses technology that is not easily available or installed without difficulty. Other times, the software does not provide an intuitive interface and requires extensive training before use.

In comparison, World Wide Graphics is comprised of an intuitive and easy to understand graphics interface. The majority of input to this system is made via a mouse. A major advantage of using WWG, is the expected cost of such

a system. It is feasible that WWG is provided for a one-time fee, rather than per person or per minute fee. For this reason, it is highly feasible that an educational institution could employ WWG.

It should be stated that the image saved and posted to the browser using WWG is opaque. That is, any image included in the HTML page may obscure the drawn graphics. In the future, it would be ideal if the image was saved as semi-transparent. Then, the drawing could be placed on a layer above the HTML page allowing both the drawing saved by the presenter and any graphics included in the HTML page to be viewable by the student. It would also be beneficial to the presenter if the system provided a method for including text on the graphics. Nonetheless, this product is useful for instructors of all levels of computer knowledge, as well as various fields of study.

APPENDIX A:  
SOURCE CODE OF THE WORLD  
WIDE GRAPHICS SYSTEM

```

/*****
/* File: DrawingPanel.java
/*
/* This class defines the shapes used and the images
/* used for display and saving. It also handles the
/* drawing of the images when the mouse is released
*****/

import java.awt.*;
import java.awt.geom.*;
import java.awt.image.*;
import java.awt.Canvas;
import java.io.*;
import javax.swing.*;
import com.sun.image.codec.jpeg.*;
import java.awt.event.MouseEvent;
import Shape;
import ShapeRectangle;
import ShapeEllipse;
import ShapeLine;

public class DrawingPanel extends JPanel{
    private Shape shape;
    private Shape rbLine, rbRect, rbEllipse;
    private Color color;
    private Color transColor;
    int trans = 80;
    private boolean fill;
    private boolean highlight;
    private boolean saving;
    private boolean firstentered = true;
    private boolean exited = false;
    private int lineSize = 1;
    private String filePath, filePath2, dirPath;
    JTabbedPane jtab;
    Image Im;
    BufferedImage BI;
    Graphics g;
    Graphics ImG;
    boolean drawing;

    String drawCmds;
    String tempCmd;

    public DrawingPanel() {

```

```

g = getGraphics();

setBackground(Color.white);
this.setBackground(Color.white);

rbLine    = new ShapeLine      (this);
rbRect    = new ShapeRectangle(this);
rbEllipse = new ShapeEllipse   (this);

setColor(Color.black);
setBackground(new Color(255,255,255,255));

setShape(rbLine);
setLine(1);

drawCmds = new String("");
tempCmd  = new String("");

addMouseListener(new java.awt.event.MouseListener()
{
    public void mouseMoved(java.awt.event.MouseEvent
        e){}
    public void
        mouseEntered(java.awt.event.MouseEvent e){
        if (firstentered & drawing)
            createbuff();

        firstentered = false;
    }
    public void mouseExited
        (java.awt.event.MouseEvent e){}
    public void mousePressed
        (java.awt.event.MouseEvent e)
    {
        g = getGraphics();
        g.setColor(color);

        setColor(color);

        ImG.setColor(color);

        shape.anchor(new Point(e.getX(),e.getY()));
    }
}

```



```

    public void mouseReleased
        (java.awt.event.MouseEvent e)
    {
        ImG = Im.getGraphics();
        ImG.setColor(color);

        shape.end(new Point
            (e.getX(),e.getY()), transColor);

        Shape rb = getShape();

        if(rb == rbLine)        drawLine (rb, g, ImG);
        else if(rb == rbRect) drawRectangle(rb, g, ImG);
        else if(rb == rbEllipse) drawEllipse (rb, g,
            ImG);
    }
    public void mouseClicked
        (java.awt.event.MouseEvent e)
    {}
    }); // MouseListener

    addMouseMotionListener(new
        java.awt.event.MouseMotionListener()
    {
        public void mouseMoved
            (java.awt.event.MouseEvent e){}
        public void mouseDragged
            (java.awt.event.MouseEvent e)
        {
            shape.stretch(new Point(e.getX(),e.getY()),
                transColor);
        }
    }); // MouseMotionListener

    setOpaque(false);
} //cons

/*****
Function: createbuff is responsible for creating the image
needed in order to save the graphics drawn
Note: a separate off-screen Image, Im, and Graphics,
ImG, associated with that image was required in order
to save the drawings to a gif.
*****/

```

```

public void createbuff()
{
    drawing = true;
    Im = createImage(800,600);
    ImG = Im.getGraphics();
} // createbuff (image)

/*****
Function: setImagePath is called by the parent component
in order to set the path in which to save the image.
*****/

public void setImagePath(String fName)
{
    dirPath = fName;
    filePath = fName + "myImg.jpg";
    filePath2 = fName + "myImg.gif";
} //setImagePath

public void redraw()
{
    if (drawing)
    {
        Image oldImage, newImage;
        ImageProducer filtered;

        oldImage = java.awt.Toolkit.
            getDefaultToolkit().getImage(filePath2);

        filtered = new FilteredImageSource
            (oldImage.getSource(),
            new TransFilter());

        newImage = createImage(filtered);

        g.drawImage(newImage, 0,0,this);
        String newPath = dirPath + "newImg.gif";

        try{

            File filen = new File(newPath);
            FileOutputStream outn = new
                FileOutputStream(filen);
            GifEncoder encode2 = new

```

```

        GifEncoder(newImage, outn);
        encode2.encode();
    }
    catch(Exception e3)
    {
        System.out.println
            ("ERROR*****saving newImage");
    }
    clear();
}
} //redraw

/*****
Function: setShape sets the shape that is chosen, either a
        line, rectangle or ellipse.
*****/

    public void setShape(Shape shape) {
        this.shape = shape;
    }

    public Shape getShape() {
        return shape;
    }

    public void drawLines      () { setShape(rbLine);      }
    public void drawRectangles() { setShape(rbRect);      }
    public void drawEllipses   () { setShape(rbEllipse);   }

/*****
Function: setColor sets the color chosen to a semi-
        transparent color by changing the alpha value.
*****/

    public void setColor(Color color) {
        System.out.println("setting transcolor");
        transColor = new Color(color.getRed(),
                                color.getGreen(),color.getBlue(), trans);
        this.color = transColor;
    }

    public Color getColor()          { return color;
}

    public void setFill(boolean b) { fill = b;

```

```

        System.out.println("Setting fill to: " + b); }

public boolean getFill()          { return fill; }

public void    setHighlight(boolean h) {
    highlight = h;
    rbLine.setHighlight(h);
}

public boolean getHighlight()      {return highlight;}

public void setLine(int s)
{
    lineSize = s;
    rbLine.setLineSize(s);
}

public void    setSave(boolean s)      {
    saveImg();
}

/*****
Note: In order for the graphics to be drawn correctly
(including the correct color with the changed alpha value),
it was necessary to pass the Graphics for the screen and the
Graphics for the saved image to the functions performing the
final drawing of a shape.
*****/

/*****
Function: drawLine draws the line using the beginning and
end points selected by the mouse input. If the
highlighting flag is set to true, then using the
stroke function draws the line.
*****/

protected void drawLine(Shape rb, Graphics g,
    Graphics ImageG)    {
    Point anchor = rb.getAnchor(), end = rb.getEnd();
    System.out.println("in drawLine");
    System.out.println("get color: " + g.getColor());

    Graphics2D G2 = (Graphics2D)g;
    Graphics2D ImageG2 = (Graphics2D)ImageG;

```

```

        ImageG2.setStroke(new
            BasicStroke(lineSize, BasicStroke.CAP_BUTT,
                BasicStroke.JOIN_MITER));
        G2.setStroke(new
            BasicStroke(lineSize, BasicStroke.CAP_BUTT,
                BasicStroke.JOIN_MITER));

        if(highlight) {
            ImageG2.drawLine(anchor.x, anchor.y, end.x,
                anchor.y);
        }
        else
        {
            ImageG.drawLine(anchor.x, anchor.y, end.x, end.y);
        }
    }

    /*****
    Function: drawRectangle draws the rectangle using the
    mouse input points set as the beginning and end
    points. It checks if the fill flag is set to
    determine if the shape drawn is a filled rectangle or
    an outline.
    *****/

    protected void drawRectangle(Shape rb, Graphics g,
        Graphics ImageG) {
        Rectangle r = rb.getBounds();
        System.out.println("In rect, fill= " + getFill());
        if(fill) {
            g.fillRect(r.x, r.y, r.width, r.height);

            ImageG.fillRect(r.x, r.y, r.width, r.height);

            System.out.println("Filled rect ??? ");
        }
        else {
            System.out.println("In not filled");
            g.drawRect(r.x, r.y, r.width, r.height);

            ImageG.drawRect(r.x, r.y, r.width, r.height);
        }
    }
}

```

```

/*****
Function: drawEllipse draws an ellipse using similar
methods of the drawRectangle function.
*****/

```

```

protected void drawEllipse(Shape rb, Graphics g,
Graphics ImageG) {
    System.out.println("Entered Drawing Ellipses");
    Rectangle r = rb.getBounds();
    Graphics2D g2d = (Graphics2D)g;
    g2d.setColor(g.getColor());
    System.out.println("set color-g2D: " +
        g2d.getColor());
    System.out.println("color alpha: " +
        g2d.getColor().getAlpha());

    Ellipse2D.Double E2 = new Ellipse2D.Double(r.x, r.y,
        r.width, r.height);

    if(fill)
    {
        g2d.fill(E2);
        ImageG.fillArc(r.x, r.y, r.width, r.height, 0,
            360);
    }
    else
    {
        g.drawArc(r.x, r.y, r.width, r.height, 0, 360);
        ImageG.drawArc(r.x, r.y, r.width, r.height, 0,
            360);
    }
}

protected void clear()
{
    if(drawing)
    {
        Image blank = createImage(1,1);
        g = getGraphics();
        g.drawImage(blank, 0, 0, this);
    }

    makeBlankImg();
    System.out.println("already 'cleared'");
}

```

```

protected void makeBlankImg()
{
    System.out.println("in make blank image");
    try
    {
        if (drawing)
        {
            ImG.clearRect(0,0,this.getSize().width,
                this.getSize().height);
        }
        Image blank = createImage(1,1);
        File fileBlank = new File(dirPath+"myImg.gif");
        FileOutputStream outBlank = new
            FileOutputStream(fileBlank);
        GifEncoder encode = new GifEncoder(blank, outBlank,
            true);
        encode.encode();
    }
    catch(Exception ee)
    {
        System.out.println("Exception in makeBlankImg: " +
            ee);
    }
}

/*****
Function: saveImg saves the images drawn to a file named
    "myImg.gif" in the path provided. It uses the
    GifEncoder provided by acme.com to save the image in
    the gif format.
*****/

protected void saveImg() {
    System.out.println("Entered saving");

    try{

        if (Im == null){
            throw new Exception("Im = null");
        }
        try{

            File file2 = new File(filePath2);
            FileOutputStream out2 = new FileOutputStream(file2);

```

```

        resized.drawImage(Im, 0,0,810,560, this);
        GifEncoder encode = new GifEncoder(Im, out2);
        encode.encode();
    }
    catch(Exception e3){
        System.err.println("Error saving to file");
        e3.printStackTrace();
    }
    }
    catch(Exception what)
    {
        System.err.println("trying to save error");
        what.printStackTrace();
    }
    System.out.println("finished saving to "+
        filePath2);
}
static class TransFilter extends RGBImageFilter
{
    public TransFilter()
    {
        canFilterIndexColorModel = true;
    }

    public int filterRGB(int x, int y, int rgb)
    {
        int alpha, r, g, b;

        alpha = (rgb >> 24);
        r = (rgb >> 16);
        g = (rgb >> 8);
        b = (rgb >> 0);

        return alpha | r | g | b;
    }
} // class TransFilter
}

```



```

/*****
/* File: ChoicePanel.java
/*
/* This class defines the buttons, checkboxes, and
/* pull-down menus that provide the choices to the
/* user such as the shape to draw, the color of the
/* shape, and the options to fill the shape or
/* highlight an area.
*****/

import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import javax.swing.*;
import DrawingPanel;

import ShapeRectangle;
import ShapeLine;
import ShapeEllipse;

class ChoicePanel extends JPanel
{
    private DrawingPanel drawingPanel;
    private Color color;
    private Checkbox fillCheckbox = new
        Checkbox("Fill");
    private Checkbox highlightCheckbox = new
        Checkbox("Highlight");
    public JButton saveButton = new JButton("Post
        Image");
    public JButton clearButton = new JButton("Clear
        Image");
    Choice geometricChoice = new Choice();
    Choice colorChoice = new Choice();
    Choice lineChoice = new Choice();

    private JPanel Example;
    private int lineSIZE, trans;
    private boolean HIGHLIGHT, FILL;
    String SHAPE = "";
    Graphics EXG;
    Graphics EX;
    Rectangle Rect;

```

```
Line2D.Double Line2;
```

```
public ChoicePanel(DrawingPanel drawingPanel) {
    JPanel    choicePanel    = new JPanel();
    Example = new JPanel();

    lineSIZE = 1;
    trans = 80;
    makeTrans(Color.black);
    HIGHLIGHT = FILL = false;
    SHAPE = "Line";

    Example.setBorder (BorderFactory.
        createLoweredBevelBorder());

    this.drawingPanel = drawingPanel;
    EXG = Example.getGraphics();
    EX = P.getGraphics();

    Example.setBackground(Color.white);
    geometricChoice.addItem("Lines");
    geometricChoice.addItem("Rectangles");
    geometricChoice.addItem("Ellipses");

    colorChoice.addItem("Black");
    colorChoice.addItem("Red");
    colorChoice.addItem("Blue");
    colorChoice.addItem("Green");
    colorChoice.addItem("Yellow");

    lineChoice.addItem("X-Thin");
    lineChoice.addItem("Thin");
    lineChoice.addItem("Medium");
    lineChoice.addItem("Thick");
    lineChoice.addItem("X-Thick");

    saveButton.addActionListener(new java.awt.event.
        ActionListener() {
            public void actionPerformed (ActionEvent e)
            {
                callSave(true);
            }
        });
};
```

```

clearButton.addActionListener(new java.awt.event.
    ActionListener()
    {
        public void actionPerformed (ActionEvent e)
        {
            callClear(true);
        }
    });

choicePanel.setLayout(new GridLayout(5,0));

choicePanel.add(new Label("Shape:"));
choicePanel.add(geometricChoice);
choicePanel.add(new Label("Color:"));
choicePanel.add(colorChoice);
choicePanel.add(fillCheckbox);

choicePanel.add(highlightCheckbox);
choicePanel.add(new Label("Line Size:"));
choicePanel.add(lineChoice);
choicePanel.add(clearButton);
choicePanel.add(saveButton);

setLayout(new GridLayout(2,1));
add(choicePanel);
add(Example);

Example.setVisible(true);

System.out.println("String of checkbox: " +
    highlightCheckbox.toString());
}

public void drawExample()
{
    EX = Example.getGraphics();

    EX.setColor(color);
    System.out.println("SHAPE: " + SHAPE);
    if (SHAPE.equals("Line"))
    {
        Graphics2D g2d = (Graphics2D)EX;
        Line2D.Double L2;

```

```

        if (HIGHLIGHT)
        {
            L2 = new Line2D.Double(10, 75, 175, 75);
        }
        else
        {
            L2 = new Line2D.Double(12, 123, 173, 13);
        }

        g2d.setStroke(new
        BasicStroke(lineSIZE, BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_MITER));

        g2d.draw(L2);

    }
    else if (SHAPE.equals("Rect"))
    {
        if (FILL)
            EX.fillRect(10,10,165,115);
        else
            EX.drawRect(10,10,165,115);
    }
    else if (SHAPE.equals("Ellipse"))
    {
        if (FILL)
            EX.fillArc(10, 10, 165, 115, 0, 360);
        else
            EX.drawArc(10, 10, 165, 115, 0, 360);
    }
}

public void clearExample()
{
    EX = Example.getGraphics();
    EX.setColor(Color.white);
    EX.fillRect(2,2,180,180);
}

public void makeTrans(Color CLR)
{
    Color T = new Color(CLR.getRed(), CLR.getGreen(),
        CLR.getBlue(),
        trans);
}

```

```

        color = T;
        System.out.println("In make Trans, color alpha: " +
            color.getAlpha());
    }

    /**
     * Functions: callSave and callClear call the appropriate
     * function in its drawingPanel.
     */

    public void callSave(boolean b)
    {
        drawingPanel.setSave(true);
    }

    public void callClear(boolean b)
    {
        drawingPanel.redraw();
    }

    /**
     * Function: action overrides the action function so as
     * to call the appropriate functions in the drawing
     * panel, such as setting the color, shape, or flags.
     */

    public boolean action(Event event, Object what) {

        if(event.target instanceof Checkbox) {
            // if highlight box is checked, then set
            // geometric choice to lines

            drawingPanel.setHighlight
            (highlightCheckbox.getState());
            drawingPanel.setFill(fillCheckbox.getState());

            HIGHLIGHT = highlightCheckbox.getState();
            FILL = fillCheckbox.getState();

            if (highlightCheckbox.getState()) {
                geometricChoice.select(0);
                colorChoice.select(4);
                lineChoice.select(2);
                drawingPanel.setColor(Color.yellow);
            }
        }
    }

```

```

        drawingPanel.setHighlight
            (highlightCheckbox.getState());
        drawingPanel.setLine(15);
        drawingPanel.drawLines();
        System.out.println("Finished drawing
            high");
        makeTrans(Color.yellow);
        lineSIZE = 15;
        SHAPE = "Line";
    }
    else if(highlightCheckbox.getState() == false)
    {
        System.out.println("Unselected
            highlighting");
        highlightCheckbox.setState(false);
        drawingPanel.setHighlight(false);
        drawingPanel.setCursor(new
            Cursor(Cursor.DEFAULT_CURSOR));
    }
}
else if(event.target instanceof Button) {
    saveButton.addActionListener(new
        java.awt.event.ActionListener() {
            public void actionPerformed (ActionEvent e)
            {
                drawingPanel.setSave(true);
            }
        });
    clearButton.addActionListener(new
        java.awt.event.ActionListener() {
            public void actionPerformed (ActionEvent e)
            {
                callClear(true);
            }
        });
}
else if(event.target instanceof Choice) {
    drawingPanel.setCursor(new
        Cursor(Cursor.DEFAULT_CURSOR));

    if (highlightCheckbox.getState())
        drawingPanel.setCursor(new
            Cursor(Cursor.TEXT_CURSOR));
}

```

```

if(((String)what).equals("Lines")) {
    fillCheckbox.setState(false);
    drawingPanel.drawLines();
    SHAPE = "Line";
}
else if(((String)what).equals("Rectangles")) {
    highlightCheckbox.setState(false);
    System.out.println("Rectangles");
    drawingPanel.setHighlight(false);
    drawingPanel.drawRectangles();
    SHAPE = "Rect";
}
else if(((String)what).equals("Ellipses")) {
    highlightCheckbox.setState(false);
    drawingPanel.setHighlight(false);
    drawingPanel.drawEllipses ();
    SHAPE = "Ellipse";
}
else if(((String)what).equals("Black"))
    color = Color.black;
else if(((String)what).equals("Red"))
    color = Color.red;
else if(((String)what).equals("Blue"))
    color = (Color.blue);
else if(((String)what).equals("Green"))
    color = (Color.green);
else if(((String)what).equals("Yellow"))
    color = (Color.yellow);
else if(((String)what).equals("X-Thin"))
    lineSIZE = (1);
else if(((String)what).equals("Thin"))
    lineSIZE = (10);
else if(((String)what).equals("Medium"))
    lineSIZE = 15;
else if(((String)what).equals("Thick"))
    lineSIZE = (20);
else if(((String)what).equals("X-Thick"))
    lineSIZE = (25);

    drawingPanel.setColor(color);
    drawingPanel.setLine(lineSIZE);
    makeTrans(color);

```

```

}

```

```
        clearExample();  
        drawExample();  
  
        return true;  
    }  
    public Insets insets() { return new Insets(5,0,5,0); }  
}
```



```

/*****
/* File: Shape.java defines abstract functions to      */
/* draw subclasses of this class. Subclasses include */
/* ShapeLine, ShapeRubberband, and ShapeEllipse      */
/* Some of the code has been revised and altered     */
/* from code out of David Geary's Mastering the      */
/* AWT [3]                                           */
*****/

```

```

import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics.*;
import javax.swing.*;

```

```

abstract public class Shape {
    protected Point anchor    = new Point(0,0);
    protected Point stretched = new Point(0,0);
    protected Point last      = new Point(0,0);
    protected Point end       = new Point(0,0);

    private Component component;
    private boolean  firstStretch = true;
    boolean highlight = false;
    int lineSize = 10;

    abstract public void drawLast(Graphics g);
    abstract public void drawNext(Graphics g);
    abstract public void setHighlight(boolean b);
    abstract public void setLineSize(int s);

    public Shape(Component component) {
        this.component = component;
    }

    public Point getAnchor    () { return anchor;    }
    public Point getStretched() { return stretched; }
    public Point getLast      () { return last;      }
    public Point getEnd       () { return end;       }

    public void anchor(Point p) {
        firstStretch = true;
        anchor.x = p.x;
        anchor.y = p.y;

        stretched.x = last.x = anchor.x;
        stretched.y = last.y = anchor.y;
    }
}

```

```

    }

    public void stretch(Point p, Color t) {
        last.x      = stretched.x;
        last.y      = stretched.y;
        stretched.x = p.x;
        stretched.y = p.y;
        Graphics g = component.getGraphics();
        g.setColor(t);
        System.out.println("in stretch, get color: " +
            g.getColor());

        if(g != null) {
            g.setXORMode(component.getBackground());
            if(firstStretch == true) firstStretch = false;
            else
                drawLast(g);

            drawNext(g);
        }
    }

    public void end(Point p, Color t)
    {
        last.x = end.x = p.x;
        last.y = end.y = p.y;

        Graphics g = component.getGraphics();
        g.setColor(t);
        System.out.println("in end, g.getColor: " +
            g.getColor());

        if(g != null) {
            g.setXORMode(component.getBackground());
            drawLast(g);
        }
    }

    public Rectangle bounds()
    {
        return new Rectangle(stretched.x < anchor.x ?
            stretched.x : anchor.x,
            stretched.y < anchor.y ?
            stretched.y : anchor.y,
            Math.abs(stretched.x -
                anchor.x),
            Math.abs(stretched.y -

```

```

        anchor.y));
    }

    public Rectangle lastBounds()
    {
        return new Rectangle(
            last.x < anchor.x ? last.x : anchor.x,
            last.y < anchor.y ? last.y : anchor.y,
            Math.abs(last.x - anchor.x),
            Math.abs(last.y - anchor.y));
    }
} // Shape

```

```

/*****
/* File: ShapeEllipse.java defines functions to draw */
/* an ellipse by defining functions declared in its */
/* parent class - Shape.java */
/* Some of the code has been revised and altered */
/* from code out of David Geary's Mastering the */
/* AWT [3] */
*****/

import java.awt.*;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.Rectangle;

public class ShapeEllipse extends Shape {
    private final int startAngle = 0;
    private final int endAngle = 360;

    public ShapeEllipse(Component component) {
        super(component);
    }

    public void setHighlight(boolean b){}
    public void setLineSize(int s){}
    public void drawLast(Graphics graphics) {
        Rectangle r = lastBounds();
        graphics.drawArc(r.x, r.y,
            r.width, r.height, startAngle, endAngle);
    }

    public void drawNext(Graphics graphics) {
        Rectangle r = bounds();
        graphics.drawArc(r.x, r.y,
            r.width, r.height, startAngle, endAngle);
    }
} // Shape Ellipse

```

```

/*****
/* File: ShapeLine.java defines functions to draw      */
/*   a line by defining functions declared in its      */
/*   parent class - Shape.java                        */
/*   Some of the code has been revised and altered    */
/*   from code out of David Geary's Mastering the     */
/*   AWT [3]                                           */
*****/

import java.awt.*;
import java.awt.geom.*;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.Graphics.*;
import java.lang.Math;

/**
 * A Shape that does lines.
 *
 * @version 1.0, 12/27/95
 * @author David Geary
 * @see Shape
 * @see gjt.test.ShapeTest
 */
public class ShapeLine extends Shape {

    //boolean highlight = false;
    //int lineSize = 10;

    public ShapeLine(Component component) {
        super(component);
    }

    public void setHighlight(boolean b) {highlight = b;}

    public void setLineSize(int s)      {lineSize = s;}

    public void drawLast(Graphics graphics) {
        System.out.println("In Line drawLast, highlight= " +
            highlight);
        System.out.println("set color: " +
            graphics.getColor());
        System.out.println("color alpha: " +
            graphics.getColor().getAlpha());
    }
}

```

```

Graphics2D g2d = (Graphics2D)graphics;
g2d.setColor(graphics.getColor());
System.out.println("set color-g2D: " +
g2d.getColor());
System.out.println("color alpha: " +
g2d.getColor().getAlpha());

if(highlight) {
    Line2D.Double L2 = new
    Line2D.Double(anchor.x, anchor.y,
        stretched.x, anchor.y);

    g2d.setStroke(new
    BasicStroke(lineSize, BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_MITER));

    g2d.draw(L2);
}
else
{
    Line2D.Double L2 = new Line2D.Double(anchor.x,
    anchor.y, stretched.x, stretched.y);

    g2d.setStroke(new
    BasicStroke(lineSize, BasicStroke.CAP_BUTT,
        BasicStroke.JOIN_MITER));

    g2d.draw(L2);
}
}

public void drawNext(Graphics graphics) {
    System.out.println("In DrawNext");
    Graphics2D g2d = (Graphics2D)graphics;

    if (highlight)
    {
        Line2D.Double L2 = new
        Line2D.Double(anchor.x,
            anchor.y, stretched.x, anchor.y);

```

```

        g2d.setStroke(new
            BasicStroke(lineSize, BasicStroke.CAP_BUTT,
                BasicStroke.JOIN_MITER));
        g2d.draw(L2);
    }

    else
    {
        Line2D.Double L2 = new
            Line2D.Double(anchor.x,
                anchor.y, stretched.x, stretched.y);

        g2d.setStroke(new
            BasicStroke(lineSize, BasicStroke.CAP_BUTT,
                BasicStroke.JOIN_MITER));
        g2d.draw(L2);
    }
}

```

```

/*****
/* File: ShapeRectangle.java defines functions to draw */
/*   a rectangle by defining functions declared in its */
/*   parent class - Shape.java                        */
/*   Some of the code has been revised and altered    */
/*   from code out of David Geary's Mastering the    */
/*   AWT [3]                                           */
*****/

import java.awt.*;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.Graphics.*;
import java.awt.Rectangle;

public class ShapeRectangle extends Shape
{
    public ShapeRectangle(Component component) {
        super(component);
    }

    public void setHighlight(boolean b){}

    public void setLineSize(int s){}

    public void drawLast(Graphics graphics) {
        Rectangle rect = lastBounds();
        graphics.drawRect(rect.x, rect.y,
                           rect.width, rect.height);
    } // drawLast

    public void drawNext(Graphics graphics) {
        Rectangle rect = bounds();
        graphics.drawRect(rect.x, rect.y,
                           rect.width, rect.height);
    } // drawNext
} // ShapeRectangle

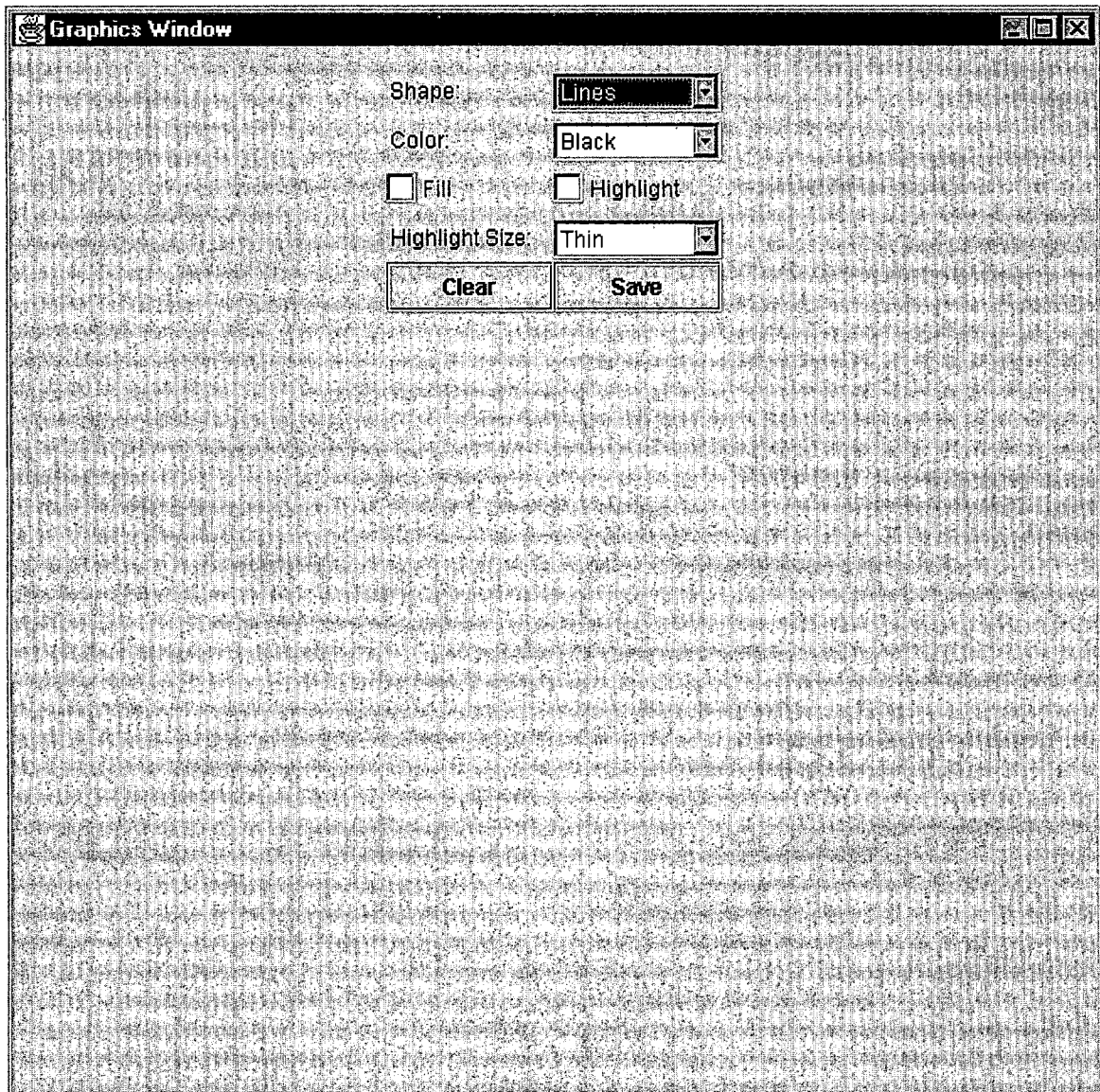
```



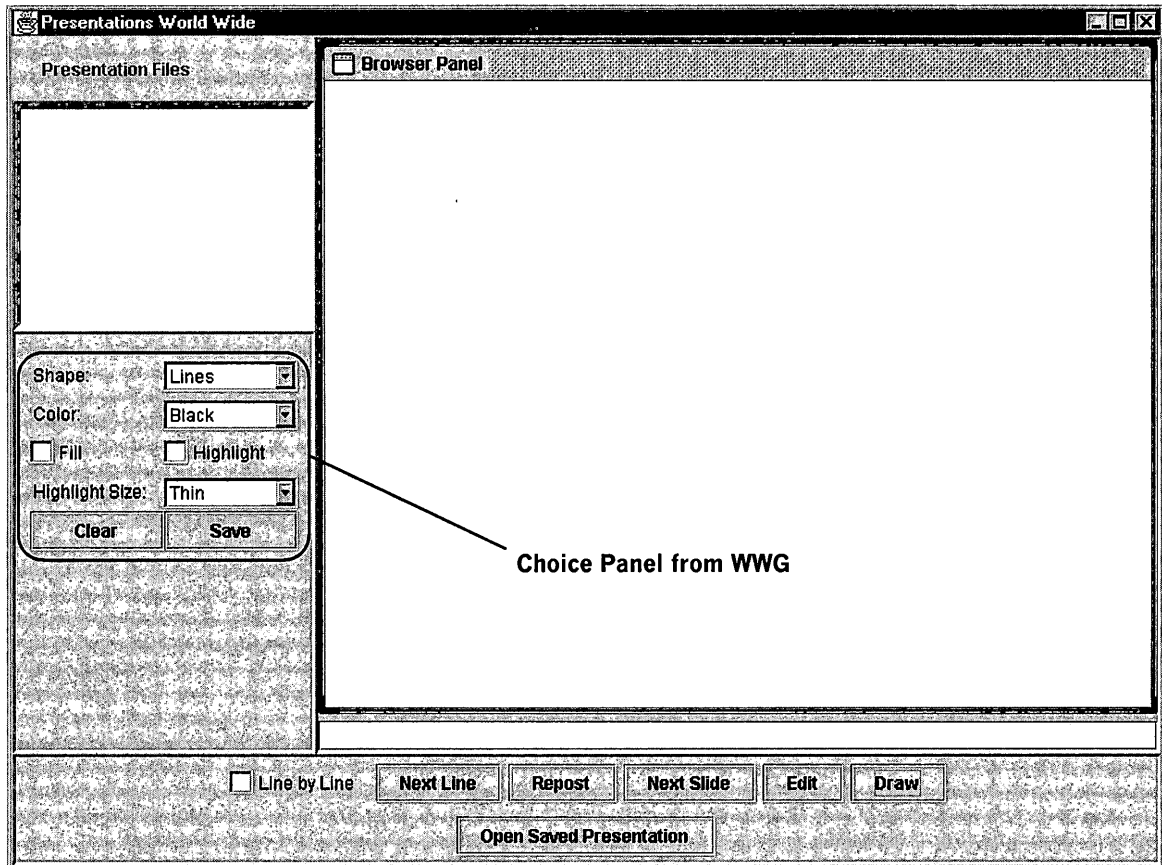
APPENDIX B:  
EXAMPLES OF THE WORLD WIDE GRAPHICS  
GRAPHICAL INTERFACE

APPENDIX B:  
EXAMPLES OF THE WORLD WIDE GRAPHICS  
GRAPHICAL INTERFACE

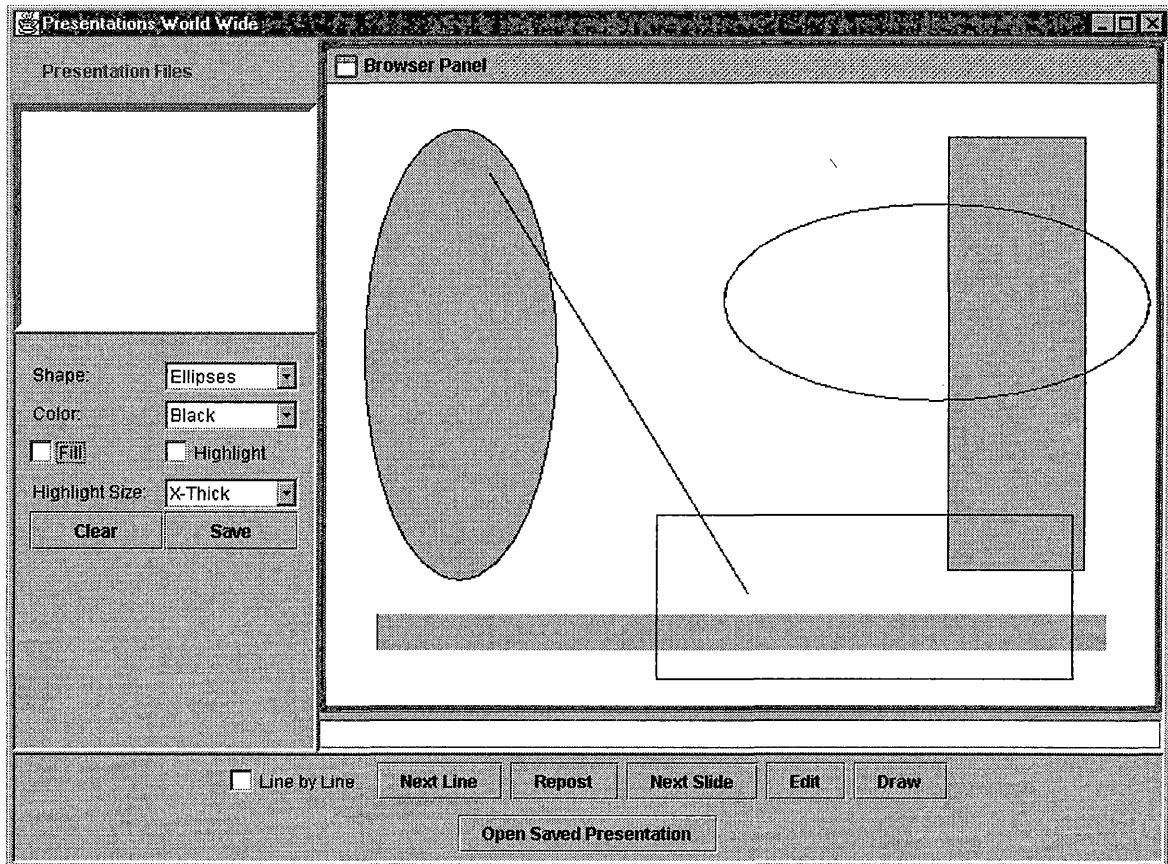
## World Wide Graphics Interface



World Wide Graphics Integrated  
into Presentations World Wide



## Example of Drawing Several Graphics



## REFERENCES

- [1] Astound Conference Center.  
  
[http://astound.com/wc/prod/prod\\_002.html](http://astound.com/wc/prod/prod_002.html)
- [2] Downing, Douglas A., Covington, Michael A., Covington, Melody Mauldin, Dictionary of Computer and Internet Terms, 6th Edition. Barron's Educational Series, Inc. Hauppauge, New York. 1998.
- [3] Geary, David. Graphics Java 1.1 Mastering the AWT. Sun Microsystems Press A prentice Hall. Mountain View, California. 1997.
- [4] Hengstebeck, Sandra Marie. Presentations World Wide System. California State University, San Bernardino. San Bernardino, California. June 2001.
- [5] Kim, Dohyon Donte. The Internet Instructional Aid. California State University, San Bernardino. San Bernardino, California. March 1999.
- [6] Simone, Luisa. "Editor's Choice". PC Magazine. December 17, 1999.  
  
<http://www.zdnet.com/pcmag/stories/reviews/0,6755,2408782,00.html>.
- [7] Simone, Luisa. "MyPlaceWare/PlaceWare 3.5 Conference Center". December 17, 1999.

- <http://www.zdbet.com/pcmag/stories/reviews/0,6755,2408779,00.html>.
- [8] Simone, Luisa. "WebEx.com/WebEx Meeting Center".  
December 17, 1999.  
<http://www.zdbet.com/pcmag/stories/reviews/0,6755,2408781,00.html>.
- [9] WebCT Helping Educators Transform Education.  
<http://www.webct.com>.