A TRANSPARENT OVERLAY FOR COMPUTER-BASED PRESENTATIONS

Major Professor	: Dr. Timothy Budd
Minor Professor	: Dr. Gregg Rothermel
Committee member	: Dr. Bruce D'Ambrosio

BY
CHANDRAKANTH L. MODEM

DEPARTMENT OF COMPUTER SCIENCE
OREGON STATE UNIVERSITY

August 1999

A TRANSPARENT OVERLAY FOR COMPUTER-BASED PRESENTATIONS

Prepared in partial fulfillment of requirements for the Masters of Science (MS) program in Computer Science

BY
CHANDRAKANTH L. MODEM

DEPARTMENT OF COMPUTER SCIENCE
OREGON STATE UNIVERSITY

August 1999

TABLE OF CONTENTS

ABSTRACT

The project aims at building an application that would simulate a transparency that can either be overlaid on top of the graphical display of another application or used as a stand-alone by accepting input from a keyboard or a mouse to enhance a presentation. The project provides an environment to accentuate the presentation by allowing the*presenter* to add on to the slide content dynamically using this transparency. This application could be used not only in seminars and academic presentations, but also in the emerging concept of computer-based teaching and remote teaching via Virtual Network Computing (VNC) wherein audience logon to the presenter's computer from a wide variety of machine architectures. This application is a step towards complete automation of a slide presentation. It has been developed using the Win32 API (Application Programming Interface) in the Microsoft Windows environment. This stand-alone application supplements any slide display – on the web or any other application. The transparency is essentially a transparent window that provides options to write and draw with the same screen on the background.

Preface

A picture is worth thousand words – This ancient adage is illustrated and proven true every day, by the use of graphical display by computer applications. Perhaps nothing is more critical, challenging and fascinating than the drive towards a computing environment that provides effective communication and user-friendliness. It is the solutions to these challenges that have made computers a common place in today's world and enabled their integration into all fields and walks of life. It is from this challenge to make computers more efficient and an effective communication tool that I derive the motivation for this project. It is my belief that my efforts in this regard will further effective communication within the medium of the computer. This would not constrain itself just to the usual presentations in the form of seminars or academic presentations but also will pave its way as a potential tool in Virtual Network Computing (VNC).

1. Introduction

The main goal of the project was to build an application that would simulate a transparency that could be used as an electronic overlay to enhance a slide presentation and serve as a potential tool in the scenario of Virtual Network Computing. The application aims at supplementing the presenter's slide content and thus enabling a better understanding of the presentation.

My major professor, Dr. Timothy Budd, initiated the idea. There was a need for an application that could replace*plastic slides* and *markers* during a presentation. There existed the technology of using computers to directly display slides on an overhead projector but this could not provide a feature that could enable the presenter to add to the slide contents on the fly. This disability gave rise to a novel idea of developing an electronic transparency that could be used with any slide presentation. This application can either be overlaid on top of the graphical display of another application or used independently, simulating a traditional slide on a computer, by instilling dynamism into the presentations. And as an additional benefit, this brings about ecological equilibrium to some extent, for it discourages the use of the plastic slides and markers that have a role to play in destabilizing the ecological balance.

This application would not only enhance seminars and academic presentations but would also supplement the emerging concept of computer-based teaching and remote teaching via Virtual Network Computing (VNC).

The VNC system is, in essence, a remote display system which allows an audience to view a computing *desktop* environment not only on the machine where it is running, but from anywhere on the Internet and from a wide variety of machine architectures. The VNC system allows the audience to access the same desktop from a wide variety of platforms.

This application provides a mode to add comments on the fly, thus furnishing an interactive environment, which would especially benefit in remote-teaching environments where there is no other means of addressing the queries of an audience.

The initial stages of the project began with the requirements analysis of the application. A paint program was a good start. I envisioned a few options that could be provided in this application. There was a need to build a *transparent* window, which on overlapping another window (slide show window) would not hide the contents of that window. A comparative study of different programming languages and environments was conducted. Though *Java* was considered a good choice because of its portability, it did not support some of the features required of this application. TheWin32 API in the C++ programming language was the final choice because of its extensive support for windows programming. The Win32 API provides functions to construct a *transparent window* and also capture various window messages. The decision about the programming environment was one of the major challenges in the initial phase of the project.

The project was completed in a period of two months with the design and requirement phase occupying a significant portion of the time. The implementation phase was a good learning experience in terms of the programming language and the *Windows Message Driven Architecture*.

Applications of the Project:

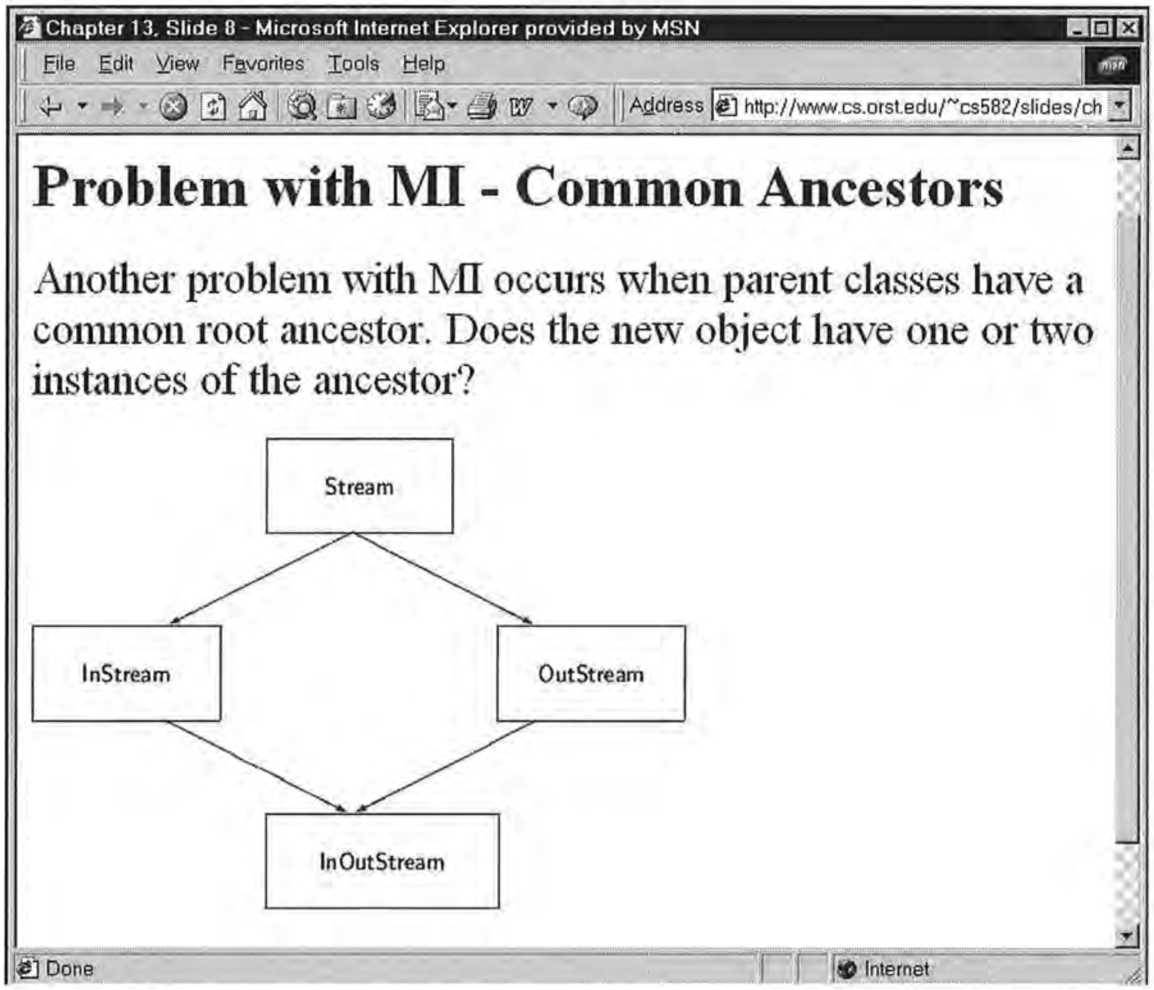
In the recent past, there emerged various ways of giving presentations through the electronic medium. These presentations were an alternative to the traditional one that used the plastic slides and markers. Now, besides using applications like PowerPoint for making slides, we can also employ languages like Hypertext Markup Language (HTML) for similar purposes. HTML slides could be either placed on the web or pulled out from an application and used for presentation purposes, as needed.

The mechanism of creating slides, using applications like PowerPoint not only eases the operations in making a slide presentation, but also allows the addition of various attributes such as setting the timer for each slide display, incorporating fancy slide transitions, and adding narration. Though these were better and easy ways of making the slide-shows that could be used during presentations, they did not provide any means for adding comments dynamically to the slide content. This was considered one of the major drawbacks in using PowerPoint slides and web pages independently, for they did not provide any mode of inserting comments dynamically during the presentations.

The prime motive of instilling a dynamic behavior into presentations through electronic medium was instrumental in the development of this *overlay,* which could be used in association with PowerPoint slides or web pages. This transparent window is a stand-alone application that can either be overlaid on top of the graphical display of another application to supplement the content of any presentation or invoked as a plain scribble pad, simulating a traditional slide on an overhead projector. The need for such an application arises in situations that involve the understanding of complex topics. Moreover, it creates an interactive environment during presentations and remote teaching environments too.

This program helps create a better dialogue between the presenter and the audience, and serves as a teaching tool in the Virtual Network Computing environments. For example, in the following snapshot, we see that the *Diamond Problem* (a problem encountered during multiple inheritance when the parent classes have a common ancestor) presented in the slide has been supplemented with an example for better comprehension among the audience. This serves as a supplementing tool by not only allowing the addition of some explanation in the form of plain text but also permits various figures to be drawn so as to emphasize certain facts in ways which would otherwise not be possible.

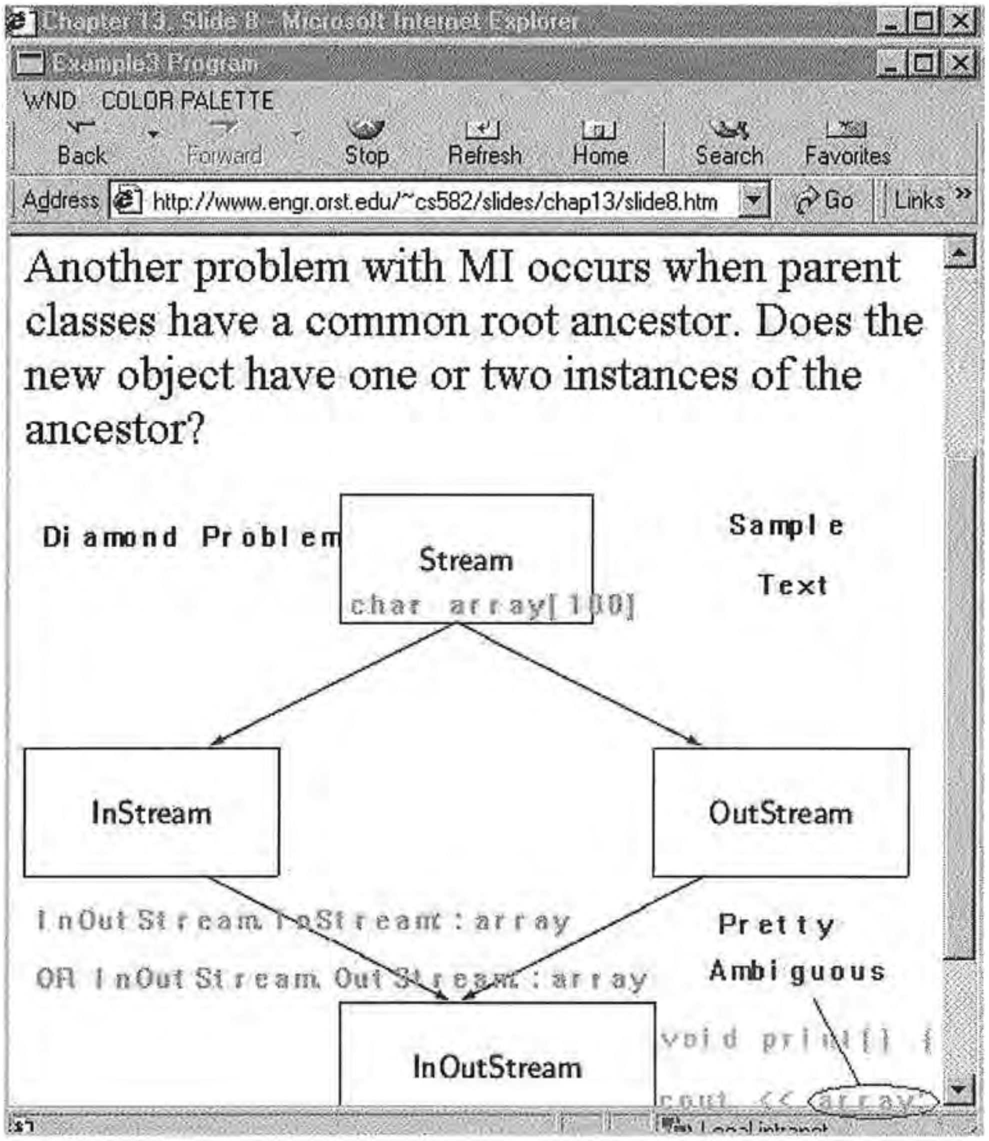
To illustrate the usage of the overlay, we can take a slide which contains some information about the *Diamond problem.*


Chapter 13, Slide 8 - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Address http://www.cs.orst.edu/~cs582/slides/ch
Stream
InStream
OutStream
InOutStream
Done
Internet

Problem with MI - Common Ancestors

Another problem with MI occurs when parent classes have a common root ancestor. Does the new object have one or two instances of the ancestor?

The slide content here just gives an overview of the concept. However, this could be explained better by giving an example with the help of the overlay during the presentation. The following slide depicts the usage of the transparent overlay.

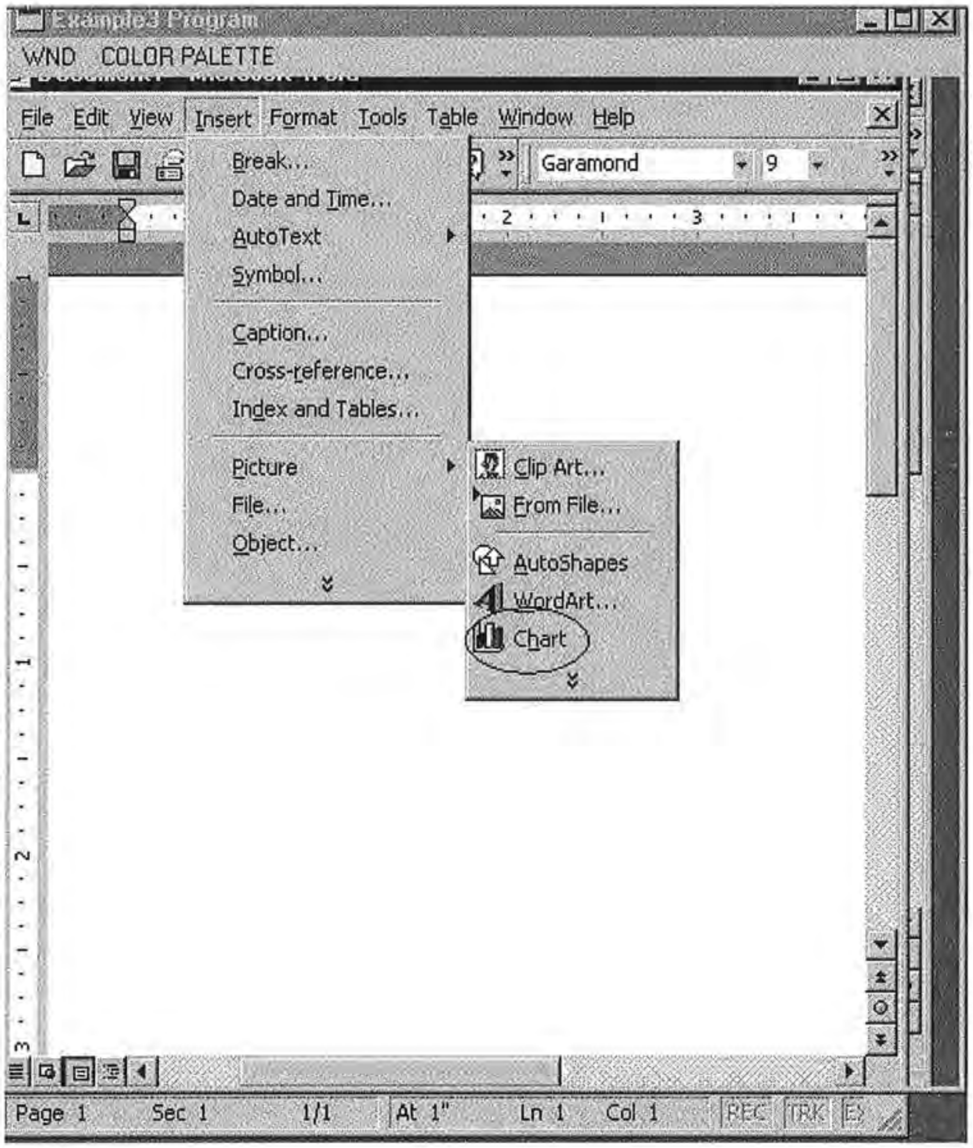


Another problem with MI occurs when parent classes have a common root ancestor. Does the new object have one or two instances of the ancestor?

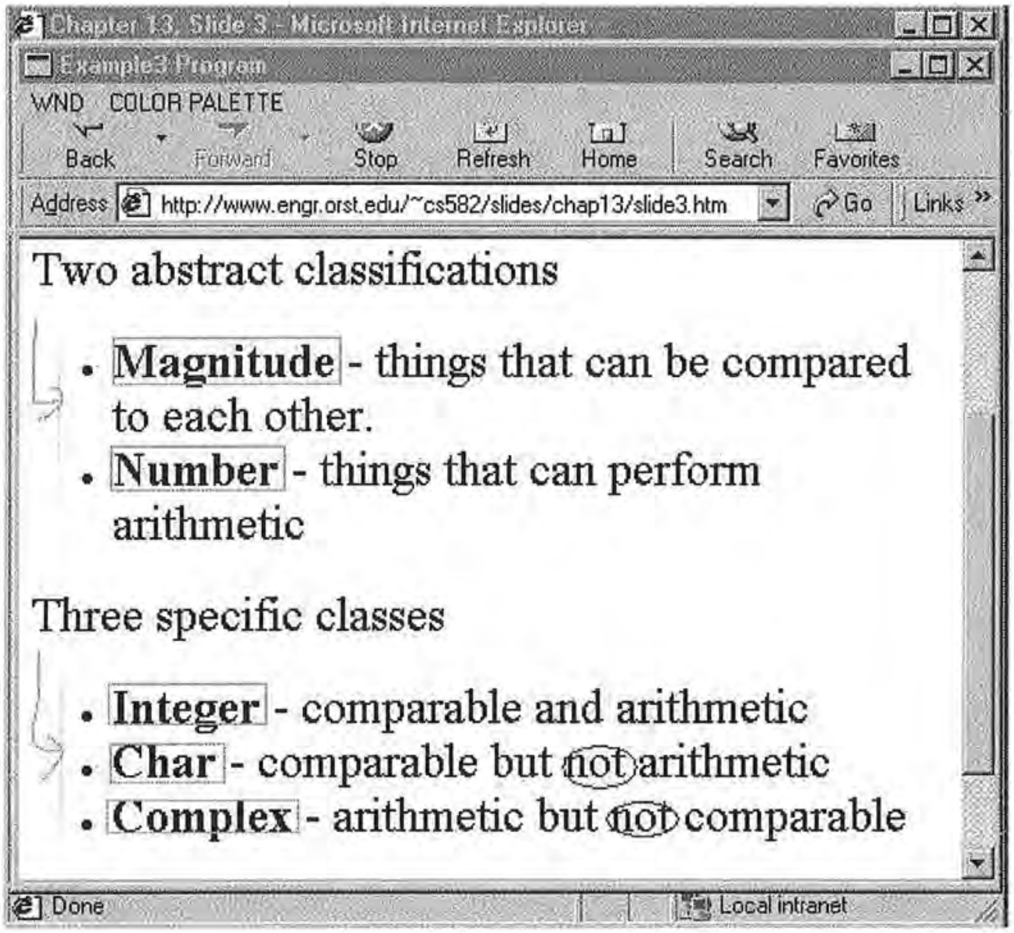
```
InOutStream::InStream::array
OR InOutStream::OutStream::array
```

```
void print() {
cout << array
```

Here, some of the comments by the way of example are added with the help of the application developed. They give us a better picture in understanding the content of the slide and also help emphasize certain facts.



One of the other primary purposes in developing this application is to use it as an aid in Virtual Networking Computing environments. Here again, the application can be used to bring about a better comprehension of ideas to be conveyed. For instance, when teaching the intricacies of Microsoft Word software, the fact of how a chart can be inserted into a document can be better conveyed dynamically by highlighting the menu options as depicted above.


Chapter 13, Slide 3 - Microsoft Internet Explorer
Example3 Program
WND COLOR PALETTE
Back
Forward
Stop
Refresh
Home
Search
Favorites
Address http://www.engr.orst.edu/~cs582/slides/chap13/slide3.htm
Go
Links
Done
Local intranet

Two abstract classifications

- **Magnitude** - things that can be compared to each other.
- **Number** - things that can perform arithmetic

Three specific classes

- **Integer** - comparable and arithmetic
- **Char** - comparable but not arithmetic
- **Complex** - arithmetic but not comparable

One other benefit of this application is in using it as a blank transparency on the desktop.

The snapshot provided above depicts the usage of the application window as a stand-alone(as a blank slide) along with some text entered using the interface to the application. Here, it has been used in place of slide sheets and markers during the course of a presentation. This serves as a supplementing tool by not only allowing the addition of some plain text but also permitting various figures to be drawn so as to emphasize certain facts.

2. Requirement Analysis

- One of the unique features required for the project was to make the window's background transparent. In this manner, the contents of the window beneath the application window are not hidden from the viewer.
- This program needs to provide a means for entering textual content. It also needs to provide the functionality to draw shapes such as a rectangle, a circle, and an ellipse and freehand drawing.
- It would be a better idea to provide a color palette so as to highlight important concepts to achieve better visualization effects.
- There is a need to choose a suitable environment and language that would help build this application, which could provide for the creation of a transparent window and also contribute for better portability and speed.

Initial Study

A comparative study of the available programming languages was initiated to find the one that would provide the features to achieve the requirements enlisted above.

To begin with, *Java* was an obvious choice because of the portability feature. But it was soon realized that it would not meet the main requirement of providing a transparent window. Also there was a need to capture all window messages and take appropriate actions. Java's event driven architecture does not support all the necessary interfaces to capture events. *Speed* is a major requirement in any Graphical User Interface application, since the effect of any user action should be reflected as soon as possible. Java being architectural-neutral, needs to execute the byte-code directly on any machine onto which interpreter has been installed, making it slower in execution when compared to a windows program. Moreover, any Java application requires the Java interpreter on the machine the program needs to be executed on. Whereas, an executable can be created using Win32 SDK, which would not need any host software on the machine to run the program.

Win32 SDK was a good option since it contained features that enabled the development of the application and also provides portability to some extent. Most importantly, it helps achieve transparency by providing a NULL Background, NULL Brush, and NULL Pen.

The application can also be ported to a Visual C++ (VC++), if developed using Win32 SDK, which would in turn use Win32 API function calls. The main advantage here is to create an *executable*(.exe) file of the application that could run in any Microsoft Windows environment(PC) where SDK/VC++ is not installed. This gives portability to some extent and helps achieve better speed than a Java application.

Win32 API window functions are based on a message-driven architecture and enable better event handling than Java API. For example, window resize, create, move, refresh, and destroy messages can be handled better in Win32 SDK than in Java. Also it provides an easier mode for creating short-cut menus and design windows with custom attributes that would allow us to specify the menu, cursor and background.

Thus, Win32 SDK was the final choice for implementing this application.

3. Design and Interface

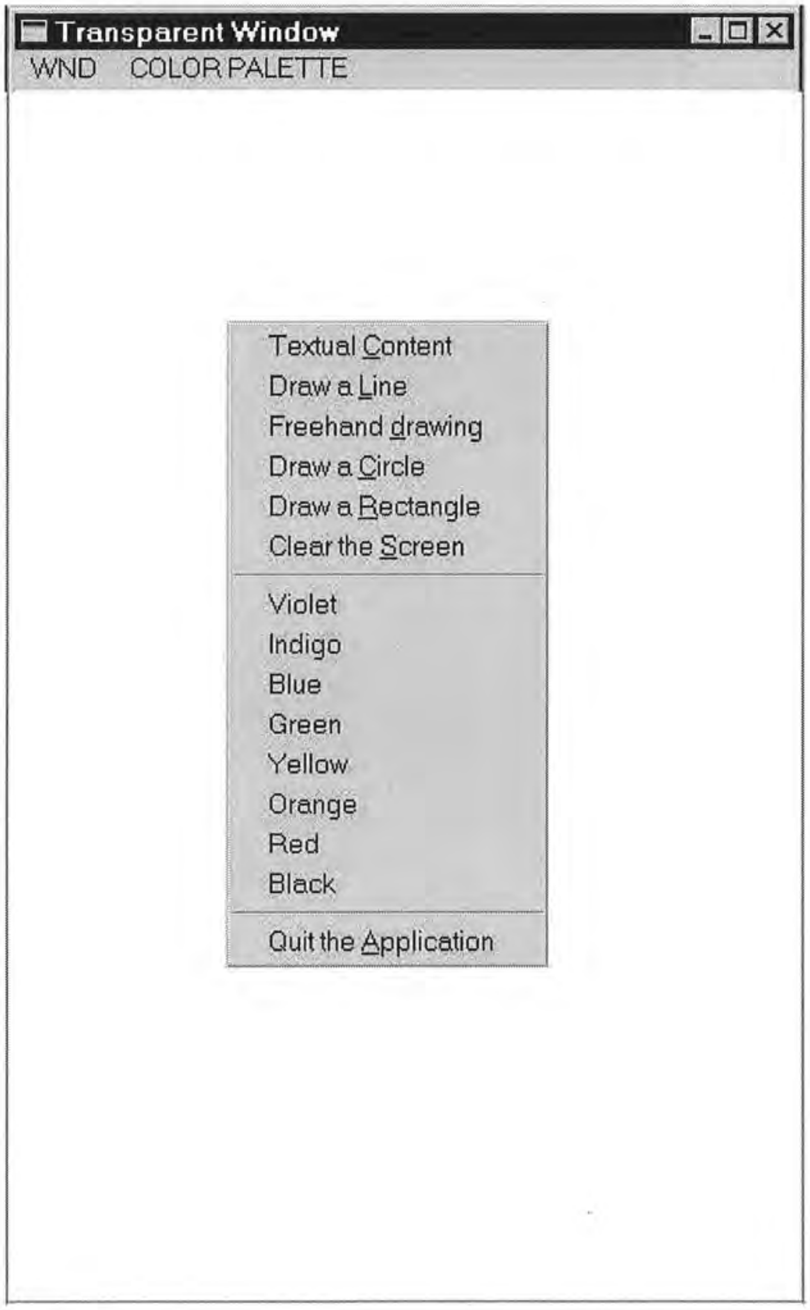
3.1 Overview

The application is available as an *executable*(.exe) file that can be invoked either from the DOS prompt or by double-clicking on the application's icon. When the application is started, a transparent window is displayed with a drop-down menu with the following options: Text, Circle, Line, Freehand, Rectangle, Color Palette, Clear, and Quit.

On selecting an appropriate option, the user can perform the corresponding action. For example, the *Rectangle* option allows the user to draw a rectangle. The user can close the application using the *Quit* option. The user can also access the above-mentioned options on the click of the *Right* mouse button that serves as a shortcut to accessing the main menu options.

The application window also provides a default control menu along with maximize, minimize, and close options.

The following figure depicts the interface to the application. It shows the short-cut menu which can be invoked by the click of a right mouse button. The options displayed within the short-cut menu can be accessed through the main menu too.



3.2 Interface

The application window is essentially a transparent window that grabs the contents within the window coordinates as its background.

The drop-down menu provides the following options:

Menu Option	Purpose
Text	This allows the user to type text where the left mouse button is depressed
Line	This enables the user to draw straight lines
Freehand	This allows free hand drawing
Circle	This enables the user to draw a circle
Rectangle	This enables the user to draw a rectangle
Ellipse	This allows the user to draw an ellipse
Color Palette	This provides a color palette to choose a color for highlighting or drawing purposes
Clear	This clears the contents of the application window
Quit	This closes the application

4. Coding and Implementation

4.1 Creation Of Transparent Window

An important feature of the application is the creation of a transparent window. The following code snippet further shows how this was achieved.

```
1   Wndclass.style           = CS_HREDRAW | CS_VREDRAW;
2  Wndclass.lpfnWndProc      = WndProc;
3   Wndclass.cbClsExtra      = 0;
4   Wndclass.cbWndExtra      = 0;
5   Wndclass.hInstance       = hInstance;
6   Wndclass.hIcon           = LoadIcon (NULL, IDI_APPLICATION);
7   Wndclass.hCursor         = LoadCursor (NULL, IDC_ARROW);
8   Wndclass.hbrBackground   = NULL;
9   Wndclass.lpszMenuName    = MAKEINTRESOURCE(IDR_MENU1);
10    Wndclass.lpszClassName          = "AClass";
```

(The Window Class' Structure)

Here, one of the attribute values that deal with the background of the window(Line 10) is set to be null (without background). This would typically mean that the window would grab the contents lying beneath it (whether it is another window or a screen if nothing lies between the*overlay* and the screen.)

4.2 Rubberbanding Effect

Rubberbanding is a technique for selecting items on the canvas. The drawing mechanism starts once the left mouse button is held down. A figure of our choice forms on the screen as the cursor is moved (within the boundary of the canvas). As the cursor is moved, the figure with the old co-ordinates is erased and the one with new co-ordinates is drawn. Once the left mouse button is released, the figure of our choice is drawn finally.

This technique has been employed to implement all the figure options: rectangle, circle, ellipse and line.

The generalized algorithm for drawing various figures using the Rubberbanding technique can be given as follows:

Start the function

(i) Choose the respective option for drawing any figure from the menu bar.

(ii) Wait until the LButtonDown event (the depression of the left mouse button) occurs.

(iii) In the handler for LButtonDown, set four variables to the current position: startx, starty, endx, and endy.

(iv) In the handle to the MouseMove (the movement of the mouse within the context of the window) message, do the following:

(a) Check if the left mouse button has been pressed.

(b) Access the existing Device Context. (A device context is an abstraction of all the devices that can be drawn onto, such as PostScript file, canvas, printer, metafile, and bitmap. Instead of drawing directly on one of these devices, the application programmer can write a function that writes to a device context, and then pass any device context to that function.)

(c) Create a Pen (a device context which is used for drawing purposes) with the drawing color as the background color (style used here is a solid pen)

(d) Draw the respective figure corresponding to the option chosen from (startx, starty) to (endx, endy).

(e) Create and select another pen with the user-defined color, and draw with the new pen from startx, starty to the current mouse position.

(f) Restore the old pen to the device context.

(v) Repeat step (iv) until the LButtonUp event(the release of the left mouse button) is triggered.

(vi) In the handle to LButtonUp, do the following:

(a) Create a new pen with the user-defined color. Select the pen and draw the corresponding figure for the option chosen from startx, starty to the current cursor position.

(b) Restore the old pen in the device context.

End the function

4.3 Clear the screen

The following code snippet shows how the window contents are cleared. Since the window captures the screen within the window coordinates for its background, it is necessary to minimize and then clear and restore it. When the window is minimized, the contents are erased and the window grabs the recent contents within the coordinates as it gets restored. This is based on the Z-Order technique followed in windows.

Z-Order is the 3D depth order in which components get drawn on the panel. Windows draws the components in the order in which they were added to the layout manager. Thus, the last component added to the layout manager will end up drawing on top of previous components if they overlap.

Here, in the following code, the handle of the window beneath the transparent overlay is acquired(Line 1). If the handle is successfully acquired i.e if there existed a window beneath, then the window beneath is brought onto the top and the overlay is minimized so as to make it lose all its contents. In the mean time, the priority of the transparent overlay is changed so as to suit its idle state(Lines 5-6). Once the window is restored, it pops up all refreshed and with its priority being set to normal(Lines 8-9).

```
1     hWndNext.GetNextWindow(GW_HWNDNEXT);
2     if(hWndNext){
3           hCurrProcess=GetCurrentProcess();
4           hWndNext.BringWindowToTop();
5     SetPriorityClass(hCurrProcess,IDLE_PRIORITY_CLASS);
6           ShowWindow(SW_MINIMIZE);
```

```
            }
SetPriorityClass(hCurrProcess,NORMAL_PRIORITY_CLASS);
ShowWindow(SW_RESTORE);
```

4.4 Color Palette

The Color Palette provides a submenu that lists the colors of the rainbow – VIBGYOR (Violet, indigo, blue, green, yellow, orange and red) along with the default black color and allows the user to choose one option. When one of these colors is chosen, then all the following drawings appear in that particular color.

Testing

The usefulness and the efficiency of this overlay were validated by using this *overlay* in association with other applications and by checking to see if it supported the features desired of it to enhance the presentations. Moreover, all the various menu options were tested by verifying the fact that they supported the functionality desired of them.

5. Conclusions

This *overlay* can either be used as a stand-alone application simulating an electronic *scratch pad* or can be used in association with other slide presentations to supplement their contents dynamically. It has a great potential in computer-based teaching and remote teaching via VNC. It is an innovative tool that can be utilized in presentations, removing the overhead of *slides* and *markers*.

The idea of *Distance learning* has made a great impact, especially with everything becoming *network-centric*. In the days ahead, there will be greater demand for tools that would help in making computer-based teaching more efficient and user-friendly. I envisage a great need for this application in the academic and industrial arena, in providing an effective means of communication.

6. Future Directions

- Window repainting can be improved so as to erase the background captured when the window is moved.
- When the window is resized, the contents of the window should be able to automatically readjust to the new window size.
- Window refreshing can be made more robust.
- More features could be added to the existing application to make it more comprehensive by providing erasure and modification of the contents within the window.
- This program can be ported to Java (once the desired features are provided) so as to enable it to run on all the environments, thus improving on its portability.
- Add sockets-based access to this application to enable remote control.

BIBLIOGRAPHY

[1] Cay S. Horstmann and Gary Cornell.: *Core java 1.2 series,* Sun Microsystems Press, California: Prentice Hall, 1998.

[2] David J. Kruglinski.: *Inside Visual C ++,* Microsoft Press, Washington: 1997.

[3] Herbert Schildt, Chris H.Pappas, William H. Murray.: *Osborne Windows Programming Series,* California: McGraw-Hill, 1994

[4] Kate Gregory.: *Using Visual C++ 6.0,* Indiana: Que, 1998.

[5] Nathan Gurewich and Ori Gurewich.: *Visual C++ 5.0 in 21 days,* Indiana: Sams Publishing, 1997.

[6] William H. Murray III and Chris H. Pappas.: *Application Programming for Windows NT,* California : McGraw-Hill, 1993.