A GUI IN VC++(6.0) FOR CREATING OpenGL OBJECTS AND ANIMATION

A THESIS

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

MASTER OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

TO
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY(DEEMED UNIVERSITY)
PATIALA. 147 001.

SUPERVISORS
B.N.Gajbhiye(DRDL,Hyd.),
DR.Renu Vig.

SUBMITTED BY
J.Ravi Kiran Prasad.
ME(R)-132/99(1).

DEPARTMENT OF COMPUTER SCIENCE,
TECHNICAL TEACHERS' TRANING INSTITUTE,
SECTOR 26,
CHANDIGARH. 160 019.
2000-01.

CERTIFICATE

This is to certify that Mr.**J.Ravi Kiran Prasad**, a student of M.E.(Comp.Sci.&Engg.), TTTI, Chandigarh, has done the Project work entitled " **A GUI in VC++(6.0) for Creating OpenGL Objects and Animation**" during the period 2000-01 in partial fulfilment for the award of the degree of **Master of Engineering in Computer Science and Engineering** of Thapar Institute of Engineering and Technology, Patiala, under my guidence and supervision.

Internal Guide:

Dr. Renu Vig,
Asst. Professor,
Department of Computer Science,
TTTI,
Chandigarh.

Project Guide:

24/1/2001

B.N.Gajbhiye,
Scientist 'F',
Head,SQAG,
CCRS,
DRDL,
Hyderabad.

Prof. V.P.Puri,
विभागाध्यक्ष, कम्प्यूटर विज्ञान,
तकनीकी शिक्षक प्रशिक्षण संस्थान,
सेक्टर-26, चण्डीगढ़-160019
Head, Deptt. of Computer Science,
Technical Teachers Training Institute,
Sector-26, Chandigarh-160019

DEDICATED
TO
MY PARENTS

ACKNOWLEDGEMENTS

I express my deep sense of gratitude to Sri B.N.Gajbhiye, Scientist 'F', Head SQAG, DRDL, Hyderabad, for his constant supervision and inspiring guidance throughout the progress of this work.

I express my gratefulness to my internal guide Dr.Renu Vig, Asst. Professor, Dept. of Computer Science, TTTI, Chandigarh, for her constant encouragement and help.

I express my profound gratitude to my youngest brother J.Murali Mohan, M.Sc.,M.Tech., Systems Manager, Wipro Systems, Hyderabad, for his invaluable help and suggestions without which this work could not have been completed.

It gives me immense pleasure to express my deep sense of gratitude to my beloved parents for their encouragement, affectionate help and support which made this work possible.

I express my heartfelt thanks to my younger sister K.R.L.Padmaja, brother-in-law K.S.R.Murthy, and their children, K.Vidya Madhuri and K.Krishna Chaitanya, for their moral help and encouragement.

I express my thankfulness to B.Rajesh Kumar Reddy, M.Tech., R.E.C., Warangal, Project student at D.R.D.L., for his helpful suggestions and discussions.

I express my gratefulness to my cousin, J.Rama Krishna Sarma, TA'B', DRDL,Hyderabad, for his cooperation and help at DRDL.

I express my thankfulness to The Head , Dept. of Computer Science, TTTI, Chandigarh for his help and support.

Finally, my sincere thanks are due to the authorities of DRDL, for providing the necessary facilities for the completion of this project work.

J.Ravi Kiran Prasad.

ABSTRACT

This project entitled " A GUI in VC++(6.0) for Creating OpenGL Objects and Animation" is developed for creating Opengl objects and animation.

OpenGL is a high quality 3D graphics Application Program Interface. By using this OpenGL, high-quality still and animated 3- Dimensional color objects with graphic effects such as shading, lighting, texture mapping are created. OpenGL is a unique choice for creating these objects mainly because of it's portability and fast rendering features.

The main focus is to provide necessary tools for drawing purposes and for viewing the developed model with various visualization effects like lighting, different views, dynamic rotation, dynamic pan, zooming option and so on.

The GUI for creating these OpenGL Objects and animation, is developed in VC++(6.0).

CONTENTS

ORGANIZATION PROFILE

1. ORGANIZATION PROFILE

Brief History

Soon after the independence, a body called DEFENCE SCIENCE ORGANIZATION was formed in 1948 for the developing science & technology in the areas of defence and its related activities. Along with this, other technical development establishments were also formed. In 1958,there was a major re-organization, by which the present DRDO (Defense Research Development Organization) was formed. Amalgamating the defense science organization with some of the existing technical development establishments lead to the formation of DRDO. Since then, DRDO has undergone a phased build up in the areas of infrastructure and expansion of units in various fields of Defence Science & Technology. At the time of formation, DRDO had under its umbrella, only 11 Laboratories all over the country. But now the number of laboratories and small Cells attached to DRDO expanded practically to all the scientific and technological disciplines of defence interest. Among these laboratories, DRDL(Defence Research and Development Labs) is one of the major Laboratories.

ROLE OF DRDL

As a supporting organization for armed forces, the main functions of DRDL are as follows :

- To design , develop systems and equipment based on the operational requirements as defined by services and to help in indigenous production.
- To render scientific advice to service head quarters.
- To carry out basic and applied research to solve the problems of the services.
- To evaluate and conduct the technical trials of new weapons and equipment that are designed and developed by the laboratory.
- To render technical support to civilian from the indigenously developed technology.

DRDL plays a vital role in technical development for National security. DRDL is one of the biggest laboratories of DRDO and is engaged in the development of Missiles. The infrastructure of DRDL includes large

manpower, sophisticated machines and modern computer systems with advanced management system. State of the art technology has been utilized where ever possible for achieving high performance.

Computer communications and range systems(CCRS) is a special body in DRDL that deals with all aspects of the above mentioned services. CCRS is well equipped with sophisticated and advanced computer systems as per the requirements of the projects, it handles. The latest achievements of the CCRS are development and maintenance of the DNET, an official Intranet application of the DRDL, Hyderabad.

Software Quality Assurance Group(SQAG) is responsible for quality assurance of the software developed for the weapon system.

INTRODUCTION

2. INTRODUCTION

2.1 OpenGL - A 3D GRAPHICS API

The project, "A GUI in VC++(6.0) for Creating OpenGL Objects and Animation", itself indicates that the graphics part of this project is achieved through OpenGL, a high-quality 3D graphics Application Program Interface. Then a common question arises, if Microsoft Foundation Classes are adopted as they provide a good user interface, then why not make use of MFC graphics library itself, Why to adopt OpenGL?

The reason for this lies in the intention in the development of OpenGL itself, which is to provide a thin software interface to underlying graphics hardware, for fast rendering 3D applications and portability on different hardware and operating system platforms, thus resulting in a unique choice for this software. OpenGL for windows98 and windows NT even had a promise to produce such excellent 3D renderings as found in the movies Jurassic Park, Toy Story, Twister, Terminator Two, Judgement Day and the like, as they are the applications of OpenGL at high end work stations.

2.11 About OpenGL

OpenGL technology is an open industry 3D graphics Application Program Interface to support 3D graphics capabilities. OpenGL was designed to be the easily transportable API, being available on a variety of H/W platforms and Operating Systems, including Microsoft's windows98 and windows NT, IBM's Os/2,DEC's AXP, Unix, Open VMS and Xwindows. It includes approximately 120 commands to draw graphics primitives such as points, lines, and polygons. However, unlike the Windows GDI (graphics device interface), these primitives are drawn in three dimensions. OpenGL also includes support for shading, texture mapping, anti-aliasing, lighting, and animation.

2.12 History of OpenGL

OpenGL uses algorithms carefully developed and optimized by **Silicon Graphics Incorporation (SGI),** an acknowledged world leader in

Computer Graphics and animation. The forerunner of OpenGL is SGI's GL, developed for SGI's high-end graphics workstations. But when SGI tried porting IRIS GL to other H/W platforms, problems occurred. Hence a new language, Open GL, offering the power of GL and being "open" to be adaptable to other platforms introduced.

2.13 Control of OpenGL

The OpenGL architecture is designed and maintained by the OpenGL Architecture Review Board (ARB), whose founding members are SGI, Digital Equipment Corporation, IBM, Intel and Microsoft. In December 1995, the ARB ratified the final specification for version 1.1 of OpenGL.

2.2 OpenGL and Windows

Microsoft's intention to introduce a GUI, resulted in operating systems with a Graphics Device Interface (GDI), a 2D API to translate GDI calls to H/W instructions. It made possible to write hardware-independent graphics - but at the cost of speed. Then graphic card makers began writing optimized GDI drivers to considerably speed up GDI. Then Microsoft introduced Win G API to lure game developers. Win G consisted of little more than a few functions that got bitmaps to the display much faster, but it was still too *slow*.

To give game developers more direct access to 2D graphics H/W, Microsoft devised a set of APIs now known as DirectX, which includes Direct Draw for screen updates, Direct Sound for fast sound and MIDI streaming, Direct Play for networked multi player games, and Direct Input for better responsiveness to Joy sticks and other I/O devices. Now to provide better 3D graphics, Microsoft added a new interface to Direct X, Direct 3D. Direct 3D is tightly integrated with Direct Draw and 3D accelerated Hardware.

As a part of Microsoft's intention to become a serious competitor for the workstation market and SGI's wish to expand into the PC market place, Open GL DLLs got integrated into the latest versions of Windows and Windows NT operating systems. To achieve it's task, Microsoft even purchased Render Morphics, Ltd., creators of the Reality

(a) In generic implementation of OpenGL all of the pixel format management, double buffering and rendering context management is handled by the generic OpenGL module and GDI. It makes use of 3DDI of Windows which communicates with Graphics Card device driver, which then accesses graphics hardware to render efficiently 3D graphical pictures.

(b) In the case of Hardware supported implementation, H/w vendors with specific hardware acceleration for OpenGL such as the GLINT chipset, may install their own OpenGL client drivers along with specialized device-driver interfaces. More seriously, the installable client driver intercepts Windows application calls. The client driver packages these OpenGL and window management (WGL) commands and sends them to the graphics display driver. The graphics display driver is linked with libraries that contain dispatch functions, OpenGL code, and some portable low-level drawing support functions. The big win with OpenGL support with OpenGL support in the display driver and appropriate hardware is **speed.** Rendering can be accelerated tremendously.

2.21 Benefits of OpenGL with windows

(1) The addition of OpenGL to Windows OS can now serve as a low-cost platform for developing and running high-end graphics applications, including CAD/CAM, and thus can replace more expensive graphics workstations.
(2) Windows OS now provides a way to add 3-D effects to applications. You won't need to buy a third-party library, because OpenGL is included with the system.
(3) OpenGL simplifies development of 3-D applications. The number of 3-D drawing and rendering applications should increase, because programmers no longer have to develop their own 3-D graphics libraries.
(4) It creates the potential for new, exciting 3-D games for windows, there by providing a standard 3-D graphics library for the Windows environment.
(5) It sets a standard that hardware manufacturers can use to create high-performance video cards. Graphics card manufacturers can improve performance by performing many OpenGL operations in hardware. In the

near future, 3-D graphics will be available on the desktop for a reasonable price.

2.22 Minimum System Requirements

Processor	:	486 processor (Pentium)
Monitor	:	SVGA
Memory	:	8 MB of RAM
OS	:	Windows 95(latest version),Windows 98, Windows NT 3.5 or above
Software	:	VC++/VRML/C++/C/VB/4GL
Display card	:	8-bit color card

2.23 OpenGL Platform Compatibility

OpenGL functions are completely portable from one platform to another. However, porting an OpenGL application from one implementation to another involves some rewriting to accommodate differences in each platform running Windows. In other words, OpenGL programs are not compatible at the binary level but are compatible at the source code level. Therefore these programs written and tested in one platform must be recompiled in others for it to run properly. Currently, OpenGL for Windows NT runs on Intel, Alpha, and MIPS platforms.

2.24 OpenGL - A Set of DLLs

OpenGL functionality is made available through a set of system Dynamic Link Libraries (DLLs). The three main DLLs include:

(1) OpenGL's Main Library is opengl32.dll, with approximately 117 functions. These functions are used to manage object shape description, matrix transformation, lighting, coloring, texture, clipping, bitmaps, fog, and antialiasing. The names for these core functions have a "gl" prefix. Required header file is gl.h

(2) OpenGL's Utility Library is glu32.dll, with approximately 43 utility functions. The commands manage texture support, coordinate transformation, polygon tessellation, rendering spheres, cylinders and disks, NURBS (Non-Uniform Rational B-Spline) curves and surfaces,

and error handling. The names for these core functions have a "glu" prefix. Required header file is glu.h

(3) OpenGL's Auxiliary Library is glaux.lib, is only a library, since it's a system-dependant wrapper about some of the operating system, containing 31 functions with a prefix aux. This is a library of functions for managing windows, handling input events, drawing classic 3-D objects, managing a background process, and running a program. The window management and input routines provide a base level of functionality with which you can quickly get started programming in OpenGL.

As OpenGL is purely a graphics and portable API, it does not include functions to manage windows and to provide user interaction. The host environment should provide such functions. As a glue between OpenGL code and host environment a set of 6 functions are included in win32 API, called wiggle functions prefixed by wgl.

(4) The WGL functions manage rendering contexts, display lists, extension functions, and font bitmaps.

(5) New Win32 functions for pixel formats and double buffering :These functions support per-window pixel formats and double buffering (for smooth image changes) of windows. These new functions apply only to OpenGL graphics windows.

2.241 OpenGL Data types

To make it easier to port OpenGL code, OpenGL defines it's own data types.

GLint, GLsizei -- 32-bit integer
GLfloat, GLclampf (color amplitude) -- 32-bit floating point
GLdouble,GLclampd -- 64-bit floating point
GLubyte, GLboolean -- 8-bit unsigned integer
GLushort -- 16-bit unsigned integer
GLuint, GLenum, GLbitfield -- 32-bit unsigned integer.

2.242 Rendering Context

An OpenGL *rendering context* is a port through which all commands pass. Every thread that makes Open GL calls must have a current rendering context. This rendering context is suitable for drawing on the device that the specified device context references. In particular, the rendering context has the same pixel format as the device context. Despite this relationship, a rendering context is not the same as a device context. A

device context contains information pertinent to the graphics component (GDI) of Windows. A rendering context contains information pertinent to OpenGL. A device context must be explicitly specified in a GDI call. A rendering context is implicit in an OpenGL call. You should set a device context's pixel format before creating a rendering context.

2.243 Graphical features of OpenGL

(1) Drawing in 3D & 2D : Lines, Points & Polygons
(2) Drawing of Quadrics : Disks, Partial disks, Cylinders & Spheres
(3) Curves & Surfaces
(4) Transformation Matrices to change the location, size and perspective of an object.
(5) Double Buffering for smooth animation.
(6) Z-Buffering to calculate the distance from the viewer's location for hidden line or hidden surface removal.
(7) Lighting effects provides the ability to calculate the effects on the lightness of a surface's color from one or more light sources.
(8) Smooth shading provides the ability to calculate the shading effects that occur when light hits a surface at an angle.
(9) Material properties provide the ability to specify the material properties of a surface to achieve shining or dullness effects.
(10) Alpha blending provides the ability to specify opacity.
(11) Raster graphics rendering.
(12) Texture Mapping provides the ability to apply an image to a graphics surface.
(13) Display lists provides the ability to reuse the calculations of a set of OpenGL commands repeatedly which improves speed.
(14) Selection and Feed back modes provide the ability to select a particular entity on the screen and to store it's related information respectively.
(15) Error Message provides the user to know the type of OpenGL error occurred that disabled the application.
(16) Information availability and many varying features.

2.3 Limitations of Generic Implementation of OpenGL

(1) There are printing limitations. An application cannot directly print an OpenGL image to a monochrome printer. An application can directly

print an OpenGL image to a color printer that offers four or more bits of color information per pixel.

(2) OpenGL and GDI graphics cannot be mixed in a double-buffered window.

(3) There are no per-window hardware color palettes. Windows has a single system hardware color palette, which applies to the whole screen. An OpenGL window cannot have its own hardware palette. It can have its own logical palette. To do so, it must become a palette-aware application.

(4) There is no direct support for the Clipboard, DDE, metafiles, or OLE.

(5) The Inventor 2.0 C++ class library is not included in the current version of Windows. The Inventor class library, built on top of OpenGL, provides higher-level constructs for programming 3-D graphics.

(6) There is no support for the following pixel format features: stereoscopic images, alpha bit planes, and auxiliary buffers.

ANALYSIS

3. ANALYSIS

3.1 Problem Specification

OpenGL is a high quality 3D graphics Application Program Interface. By using this OpenGL, high-quality still and animated 3- Dimensional color objects with graphic effects such as shading, lighting, texture mapping are created. OpenGL is a unique choice for creating these objects mainly because of it's portability and fast rendering features. The main focus is to provide necessary tools for drawing purposes and for viewing the developed model with various visualization effects like lighting, different views, dynamic rotation, dynamic pan, zooming option and so on.

3.11 Incorporating OpenGL into MFC

Since OpenGL is portable , it is available on a variety of hardware platforms and operating systems, including Windows 95, Windows NT, OS/2, DEC's AXP and Open VMS, and X Windows.

To use OpenGL in our programs, we need either Microsoft Windows NT 3.5 or later (preferably 4.0) or a recent version of Windows 95. If our version of Windows 95 doesn't have OPENGL32.DLL and GLU32.DLL in the Windows system directory, we have to download them from Microsoft. Once we have the DLLs and LIBs in place, then we can start programming.

A new OpenGL program can be obtained by creating a totally reusable MFC class with a little effort that would work right away, and could have features that make OpenGL programming a simple matter of adding a few lines of code to the program rather than creating a basic OpenGL view class from scratch, remembering the details one by one and implementing them, a somewhat difficult task to achieve.

The first step is to create a new project using the Developer Studio. We have to perform the following steps:

- Select the new file as Project Workspace.

- Select the target as an MFC AppWizard EXE.
- Name the project "OpenGL" in the location of your choice.
- Select Single Document Interface (SDI).
- Optionally, we can turn off all other options,no OLE, ODBC, 3D controls, and so on.
- Continue through the last dialog box .Then Press OK

Then the Developer Studio creates the COpenGLView class file skeleton and a framework where we can test it. At this point, we have to compile the application just to make sure that there are no errors in the build process. If the project builds successfully, we can proceed to the next step:incorporating all of the steps necessary to make the view class capable of displaying OpenGL images.

3.12 Logistics

Before we can add OpenGL code to our MFC application, we need to include header files for functions. These headers provide function prototypes for all of the OpenGL functions and auxiliary functions.

Add the following lines to STDAFX.H:

- #include "gl\gl.h" // Include the standard OpenGL headers
- #include "gl\glu.h" // Add the utility library
- #include "gl\glaux.h" // Add in the auxiliary library (optional)

When building our application, it is necessary to link the OpenGL static libraries to our application. We can do this by linking the following files:
OPENGL32.LIB
GLU32.LIB
GLAUX.LIB(Optional).

While using Visual C++,

- Select Project from the Main Menu Bar.
- Select Settings from project menu
- Select the Link tab
- Link the 3 libraries by entering OPENGL32.LIB, GLU32.LIB, and GLAUX.LIB to the Object/Library Modules edit control.

The last library is only necessary if we are using the auxiliary library.

3.13 Customizing the view for OpenGL

If we open up the class wizard and select the COpenGLView class, we can see that the message handlers for **OnDraw( )** and **PreCreateWindow( )** are already passed on to the class. We have to add functions for the following additional messages:

- WM_CREATE (for **OnCreate**)
- WM_DESTROY (for **OnDestroy**)
- WM_ERASEBACKGROUND (for **On EraseBkground**)
- WM_SIZE (for **OnSize**)

3.2 To Create an OpenGL Program

The first step in creating an OpenGL program is to set up the rendering window so that OpenGL calls will work. OpenGL is a totally different rendering procedure than GDI. The OpenGL specification supports things such as double buffering, stereo displays, and additional hardware video planes that aren't possible under GDI (at least, so far). For OpenGL to render to a Windows window, we have to make the window format acceptable to OpenGL. The steps that we need to take care of include :

- Setting up the correct window style.
- Getting a device context for the rendering location.
- Selecting and setting a pixel format for the device context.
- Creating a rendering context associated with the device context.
- Responding to the WM_SIZE message.
- Responding to the WM_ERASEBACKGROUND message.
- Creating and drawing objects using OpenGL commands.
- Deleting the RC and DC.

3.21 Setting up the correct window style

Before the window is created, we have to set the window style to include WS_CLIPSIBLINGS and WS_CLIPCHILDREN to prevent OpenGL from trying to draw into any other windows. If this is not set up properly, then the pixel format for the window cannot be set up.

This can be achieved by simply overriding PreCreateWindow function and add the following line:

```
BOOL COpenGLView::PreCreateWindow(CREATESTRUCT& cs)
{
      cs.style |= WS_CLIPSIBLINGS | WS_CLIPCHILDREN ;
      return CView::PreCreateWindow(cs);
}
```

3.22 Creating a device context

Before setting a PIXEL FORMAT we have to get a pointer to the window's Device Context(DC) which can be achieved by writing the following code in the On Create( ) function:

```
CDC* pDC;
pDC=GetDC();
```

3.23 Setting up a pixel format

Before OpenGL can render images on a drawing surface (window or bitmap), the drawing surface must be initialized. A pixel format specifies several properties of a drawing surface, mainly those dealing with the organization and layout of the color bits. A pixel format is specified with the PIXELFORMATDESCRIPTOR structure. Each window has its own current pixel format. Different windows can have different pixel formats. and a single device can support several pixel formats.

To describe a pixel format as per our desire, we fill in the fields of a PIXELFORMATDESCRIPTOR structure. The code below to be added in the On Create() function of COpenGLView class:

```
int CGPView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
      if (CView::OnCreate(lpCreateStruct) == -1) return -1;
      PIXELFORMATDESCRIPTOR pfd; int ipf;
      CDC* pDC;
      pDC=GetDC(); //Obtains the Device Context's Handle
      memset(&pfd,0,sizeof(pfd));
      //Allocates Memory for PixelFormatDescriptor object
      pfd.nSize=sizeof(pfd);//sets the size
      pfd.nVersion=1;// Sets the Version to 1
      pfd.dwFlags=PFD_DRAW_TO_WINDOW|PFD_SUPPORT_OPENGL;
     //Render the Picture on the window & provide Opengl support on the
        surface
      pfd.iPixelType=PFD_TYPE_RGBA;
      pfd.iLayerType=PFD_MAIN_PLANE;
      pfd.cDepthBits=24;
```

```
    //sets the bpp(bits per pixel) value for depth information
    ipf=ChoosePixelFormat(pDC->m_hDC,&pfd);
    // Chooses a PixelFormat that matched to user specification
    SetPixelFormat(pDC->m_hDC,ipf,&pfd); //Sets the PixelFormat
    m_hglrc=wglCreateContext(pDC->m_hDC);
    //Creates a Rendering Context for DC
    wglMakeCurrent(pDC->m_hDC,m_hglrc); //Makes the RC current
    return 0;
}
```

3.24 Pixel Format Functions

- **ChoosePixelFormat( )** : Obtains a DC's pixel format that's the closest match to a pixel-format template we've provided. Windows tries to match our requirements to what's available.
- **SetPixelFormat( )** : Sets a device context's current pixel format to the pixel-format index specified.
- **GetPixelFormat( )**: Returns the pixel format index of a device context's current pixel format.
- **DescribePixelFormat( )**: Given a device context and a pixel format index, fills a PIXELFORMATDESCRIPTOR data structure with that pixel format's properties.

3.25 Creating a Rendering Context

Once the pixelformt is set, we will need to create an OpenGL Rendering Context (GLRC). We can think of a Rendering Context as a port through which all OpenGL commands must pass. The Rendering Context we create has the same pixel format as the device context with which it is associated. A Rendering Context(RC) is not the same as a Device Context(DC). A DC contains information for GDI, while a Rendering Context contains information for OpenGL. We can create multiple Rendering Contexts in a program.

To deal with the interface between Windows and OpenGL to get windows-specific information, such as the current DCs, into or out of the Rendering Context,the Windows implementation of OpenGL includes several Wiggle Functions, prefixed by wgl(Windows GL)
Some of the Wiggle Functions related to OpenGL are :

- **wglCreateContext( )**: Creates a new OpenGL RC suitable for drawing on the Device referenced by the associated DC. The RC has the same pixel format as the DC.

- **wglMakeCurrent( )**: Makes a specified RC, the calling thread's current RC. OpenGL calls made by the thread are then rendered on the device identified by the DC. If the RC specified is NULL, the function deselects the calling thread's current RC and releases the DC used by the RC. In this case, the DC is ignored.
- **wglDeleteContext( )**: Deletes the specified RC. It is an error to delete an RC that is another thread's current RC. However, if an RC is the calling thread's current RC, the function makes the rendering context, not current before deleting it.
- **wglGetCurrentDC( )**: If the calling thread has a current RC, the function returns a handle to the DC associated with that RC by means of wglMakeCurrent(). Otherwise, NULL is returned.
- **wglGetCurrentContext( )**: If the calling thread has a current RC, the function returns a handle to that RC. Otherwise, NULL is returned.
- **wglShareLists( )**: Enables an RC to share the display-list space of another RC in another thread. Display lists are a way of caching rendering commands. When you create an RC, it has its own display-list space. All RCs sharing a common pixel format can always share display lists.

To create a Rendering Context(RC), we use the wglCreateContext function which returns an HGLRC, which is a handle to the RC stored in a member variable, m_hglrc, of the view class.

3.26 Responding to WM_SIZE Message

If we get Windows ready to use OpenGL, then we have to tell OpenGL about the Window we're rendering to and how we want our scene to be viewed. For example, when we get a new WM_SIZE message, we have to tell OpenGL to modify its view. If we're using perspective mode, we probably want the perspective to change as the Window dimensions change. Alternatively, we might want to render different scenes or different views of the same scene into the Window.

OpenGL has something called a Viewport that lets you specify how you want the Rendering Area to relate to the actual Window dimensions.

The following functions set up the screen which are to be written in the OnSize Member function of the View class of the Project

```
void COpenGLView::OnSize(UINT nType, int cx, int cy)
{
     CView::OnSize(nType, cx, cy);
    // TODO: Add your message handler code here
    if ( 0 >= cx || 0 >= cy )
        {
            return;
        }
    wglMakeCurrent (pDC->m_hDC,m_hrc); //Make the Rcurrent,compute the
    aspect ratio,which will keep all dimension scales  equal
    GLdouble gldAspect = (GLdouble) cx/ (GLdouble) cy;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(30.0, gldAspect, 1.0, 10.0);
    glViewport(0, 0, cx, cy);
    wglMakeCurrent(NULL,NULL); //To deactivate the current RC
 }
```

3.27 Responding to the WM_ERASEBACKGROUND message

Overloading OnEraseBkgnd and returning TRUE will stop the program from painting the screen white before you render your OpenGL screen. This will make your program look much better because it will eliminate the extra screen flash.

3.28 Creating OpenGL objects

To Render the picture which is usually done by writing the OpenGL commands in the OnDraw() member function of the View Class. This process consists of the following steps:

- Make the rendering context current.
- Draw the scene using OpenGL commands.

3.29 Deleting the RC and DC

We delete the RC and DC by writing the following code in the OnDestroy( ) member function

```
void CWinView::OnDestroy()
{
     CView::OnDestroy();
     // TODO: Add your message handler code here
     wglMakeCurrent(NULL,NULL);
     wglDeleteContext(m_hglrc);
}
```

SOFTWARE REQUIREMENTS

4. SOFTWARE REQUIREMENTS

4.1 Object Oriented Concepts and Principles

In order to design any system we need to create models so that we can communicate our ideas. The idea of a model is that it represents the important aspects of a system and ignores the incidental aspects. The process of deciding what features are important and putting them in a model is called 'Abstraction'.

The development of any software system is an expensive and time-consuming process. To lower this cost as well as to achieve an abstract design and to improve the quality of the software developed, **Object Oriented development approach** has been adopted. Object Modeling starts from the assumption that the world is composed of objects that communicate with messages and perform operations (or methods) in order to complete task or series of tasks, to achieve a goal.

A Common definition of "Object Oriented" is

Object - Oriented = Objects + Classification + Inheritance + Communication

4.11 Definition of an Object

Any thing that we can identify as an individual thing, real or imaginary, is an object. An object can be distinguished from one another by their attributes, by what they do, what can be done with them and their behavior. The value of particular object's attribute at a particular time is it's state. Booch defines an object as any thing having state, identity & behavior.

4.111 Objects with Class

Objects that have the same attributes and behavior but different identities are of the same class.

4.112 Attributes, Methods and Messages

(1) Real Life entities are often described with words that indicate stable features which can have a simple value or values of

another class. Methods describe behavior of objects or services provided or to set and read each of the attributes of an object.

(2) An object will call the method of another object to modify the second or to get the second object to provide a service, is through Messages.

The essence of Object - Oriented programming is to treat data and the procedures that act on the data as a single "object" - a self-contained entity with an identity and certain characteristics of its own.

4.12 Encapsulation, Inheritance and Polymorphism

The important pillars of Object Oriented approach are:

1. **Encapsulation and Data Hiding: -** Encapsulation is the mechanism that binds together code and data, and that keeps both safe from outside interference or misuse. Further, it is encapsulation that allows the creation of an object. Simply, an object is a logical entity that encapsulates both data and the code that manipulates the data. Within an object, some of the code and/or data may be private to the object and inaccessible to anything outside the object, thus providing a significant level of protection.

2. **Inheritance:** Inheritance is the process by which one object can acquire the properties of another object. This is important because it supports the concept of Classification. Through the use of classifications, an object needs only to define those qualities that make it unique within its class. It is the inheritance mechanism that makes it possible for one object to be a specific instance of a more general case.

3. **Polymorphism:** Polymorphism is characterized by the phrase "One interface, multiple methods". Simply, polymorphism is the attribute that allows one interface to be used with a general class of actions. The specific action selected is determined by the exact nature of the situation. This helps reduce complexity by allowing the same interface to be used to specify a general class of actions.

One of the programming languages that support the basic principles of Object Oriented Programming Strategies is C++. C++ supports the properties of Encapsulation and Data Hiding through the creation of user - defined types, called Classes. Once created, a well-defined class acts

as a fully encapsulated entity and can be used as a whole unit. C++ supports the idea of Inheritance through Derived Classes and Containment. The idea of Polymorphism, that different objects do "the right thing" - through Functional Polymorphism and Class Polymorphism.

4.2 Visual C++

OpenGL, which is the basis for this project is not a programming language, rather it is a 3D API (Application Programming Interface). So, it has to be embedded in a host environment to provide user interaction and for window/screen management.

One such platform is Microsoft's Visual C++ and the Microsoft Foundation Classes as wrappers to the generic windows API, as it provides ease in interface programming.

Visual C++ is a collection of many tools, all wrapped together into one dynamic, ready-to-use package, offering a wealth of visual controls say, editors, tools, test containers, class libraries, debugging techniques and more.

The first way to create large windows programs was to use C, which used to result in a difficult and large code even to generate a simple blank window. Later C++ was introduced, which divided one long monolithic program, into neat, manageable chunks. In addition, to provide visual design Microsoft introduced the Microsoft Foundation Class(MFC) library.

The MFC is an extraordinary package, of *prewritten*, ready-to-use code. For example, instead of having to write all the code to support a new window, we can just rely on the MFC CWnd package, which handles it.

Visual C++ not only makes use of the MFC, but also makes windows programming far easier by introducing many programming tools, such as the Menu editor for designing menus, and the Dialog editor for Dialog boxes along with Object Oriented approach of C++. VC++ provides us with one integrated design environment in which we can write our programs and run them. In addition, Visual C++ organizes many files a windows program needs into projects. Projects themselves are placed in Workspaces, and a workspace can have several projects, with one being

active at an instant of time. VC++ provides various design controls like Toolbars, Pushbuttons, Radio buttons, check boxes, combo lists, display lists, and the like to provide an easy user interface for the applications.

Visual C++ makes a provision from displaying graphics to menus, from working with buttons and textboxes to handling files, from stepping into the Internet to creating our own ActiveX controls and even to create a fully functional Web Browser, including hyperlinks, graphics and so on.

The special programming tools called Visual C++ Wizards, will automatically generate code to handle the window, which later can be customized with C++ code to suit the required applications.

4.3 Windows 98 Operating System

Windows98 is a Single-User Operating System of Microsoft family, released on 26th January 1998. Windows95, though a 32-bit operating system, had a major deficiency namely, the absence of support for the 'Universal Serial Bus', which did not allow true plug-and-play. Windows98 overcomes this deficiency and adds support for virtually all the latest H/W and even includes support for TV. This can add upto 127 instruments.

Windows98 focus is on four fronts - tighter web focus, greater H/W support, increased reliability and communication & entertainment. At the base level, windows98 supports virtually, all the available modems, printers, fax machines and scanners while, at a higher level supports the 'Universal Serial Bus, multi-function cards and infrared connectivity (where data transfer is done without using cables). The highest level of H/W support is for multiple monitors and the broad architecture. With a support for up to 8 monitors on a single windows98 OS, the video game manufacturers and the graphic and television software designers are in for a dreamtime. The second edition of Windows98 even provides OpenGL DLLs, an efficient 3D graphics API.

Windows98 performs few tasks such as automatic scanning of the hard disk after an improper shutdown, proactively. The other two important tasks of windows98 are faster start up and faster shutdown. The

gain in the start-up speed comes on account of a new, optional feature called "OnNow".

The additional features include its support for 17GB DVDs and Intel's MMX instructions. This even provides for system security problems.

4.31 Features of Windows 98

1. **Easier to use**: - Navigating around the computer is easier because of the provision of a single-click to open files and Browser buttons in every window. Installing new hardware is easy because Windows98 supports the Universal Serial Bus standard allowing the user to plug in new H/w and use it immediately - without restarting the computer. With this OS, digital cameras and other digital imaging devices can also be used.

2. **More Reliable**: - Windows98 tools help the user regularly test the hard disk and check the system files - and even automatically fix some problems. The troubleshooters and the Dr.Watson diagnostic tool also help solving computer problems.

3. **Faster**: - Windows and programs open faster than ever before. By using the Maintenance wizard, we can easily improve our computer's speed and efficiency. The power management feature allows new computers to go into hibernation mode and awaken instantly, instead of requiring us to shut down and restart our computer. Also, we can use the FAT32 file system to store files more efficiently and save hard disk space, where FAT16 file system is supported by all the previous versions of Windows98.

4. **True Web Integration**: - Using the *World Wide Web* is easier - and faster - than ever. The Internet Connection wizard makes connecting to the Web simple. Using Microsoft FrontPage Express, we can create our own Web pages and then publish them using the Microsoft Web Publishing wizard. By using the Web-style Active Desktop we can view Web pages in any window - we can even make our favorite Web page as our desktop wallpaper. In Microsoft Outlook Express, we can send e-mail and post messages to Internet newsgroups. Using Microsoft NetMeeting, we can collaborate on documents and hold conferences over the Internet. Using Microsoft Windows Media Player, we can play live "streaming" media - again, over the Internet.

5. **More Entertaining**: - Windows98 supports DVD, digital audio, and VRML so that we can play high-quality digital movies and audio on our computer, as well as see the full effect of Web pages that use virtual reality features. We can also watch television broadcasts and check TV program listings by using Microsoft Web TV for Windows.

We can dramatically change the look and sound of our desktop by applying a theme. Several themes are ,included with Windows98. Each desktop theme includes unique *wallpaper,* screen savers, 3-D icons, sounds, fonts, color schemes, and mouse pointers.

TECHNICAL REQUIREMENTS

5. TECHNICAL REQUIREMENTS

To run this software, the installation requirements include

Monitor	:	SVGA
Processor	:	Pentium Processor
Operating System	:	Windows 95/98 / Windows NT version 3.5 or later(should include OpenGL DLLs).
Memory	:	32 MB RAM or above for better visual effects.
Software	:	Visual C++

DESIGN

6. DESIGN

6.1 System Design

System Design is a solution, a "how to" approach to the creation of a new system. The design of a system is essentially a blueprint or a plan for a solution for the system. The design step produces a data design, which transforms the information domain model, created during analysis into the required data structures and architectural design, which defines the relationship among major structures into a procedural description of the software.

An Object-oriented system like this package is usually developed around class hierarchies. A class hierarchy is a generalization-specialization structure in which there is a systematic refinement of ideas, concepts, attributes and operations of the idea or concept being refined gradually.

Object-Oriented Design Methodology has a top-down, progressive refinement approach. This methodology involves the following 3 phases:

1. Producing the initial design.
2. Functional refinement.
3. Object refinement.

6.11 Producing the initial design

To identify the objects and their operations as well as defining the transformation functions using these objects, a three step procedure can be adopted:

(a) Develop a strategy in English specifying the problem definition.

(b) Identify the objects and the operations on the objects. The operations can be identified by the actions specified in the strategy. The entities on which the operations can be done, refer to the objects. The operations that don't operate on any identified function are marked for refinement.

(c) Establish the interface of the objects, which include the functionality to invoke objects and their operations.

1. **Problem definition**: In this software, by making use of OpenGL API and Visual C++(6.0) as GUI, initially primitive objects are created. Next, objects are created and transformations, translation, rotation and scaling are applied. Finally, animated objects, with effects such as lighting, dynamic rotation and paning, are created.

2. **Object Identification**: The primitive objects include:

- Creation of parallelopipoid.
- Creation of coloured prism.
- Creation of disks.
- Creation of light sources.

The objects that are created for the application of transformations, translation, rotation and scaling, include:

- Creation of a solid cone.
- Creation of a solid teapot.
- Creation of a NURB's surface.
- Creation of a wire box.
- Zooming of a wired cube.

The objects that are created for animation, lighting, dynamic panning and rotation, include:

- Creation of a coloured rotating 3D cube.
- Creation of a white coloured rotating 3D cube with lighting.

3. **Interface**: For creating the objects, VC++(6.0) is used as an interface. There is a push button for each object. By, pressing a button, the concerned object is created and from the menubar, any one of the selected transformations, translation, rotation and scaling can be applied on the object. Similarly, for animation also, by pressing the button, the animated object is created. After selecting the "Pan" option from the menubar, the dynamic panning can be done by keeping the mouse right button down and dynamic rotation can be done by keeping the mouse left button down.

6.12 Functional Refinement

This is an iterative process in which the operations marked for refinement, in the previous phase are taken and an informal strategy is written and again some more objects and operations are identified. This continues till there are no functions for refinement.

6.13 Object Refinement

This phase is especially required in large systems where each object includes some more objects. This is again an iterative process in which the multilevel objects are refined until there are no more objects for refinement.

Object oriented design provides two distinct views of systems, the structural view and the behavioral view. The structural view describes the static structure and the relationship of objects, while the behavioral view describes the dynamic behavior of objects over time. The techniques used for structural view modeling include various class diagrams, object diagrams and the ER diagrams. The state-transition charts, flow charts, timing diagrams, interaction diagrams and event diagrams commonly describe the behavioral view modeling.

6.14 Programming Approach

The underlying programs for this software may be viewed broadly as consisting of two main aspects: interfacing with the end user and programs to actually carry out the designing tasks. The programs for the generation have been written in **C++**, incorporating it's object oriented features. Interfacing is done using **Visual C++.**

The programs in C++ have been written in a modular fashion with specific tasks being carried out by specific functions. The approach of object oriented design, incorporating features such as data abstraction, data encapsulation, data inheritance, polymorphism and data hiding has been implemented. The graphical user interface for this package is developed using **VC++**. The user interaction is through mouse selection, push buttons, menus, toolbars and dialog boxes, which have been incorporated using Microsoft Foundation Classes. The actual graphics display is controlled using **OpenGL,** a high standard 3D graphics library.

The C++ programs are called through header files in the relevant VC++ classes. Consequently, VC++ acts as the front end and user interface, C++ carries out the data storage, retrieval and processing in the background while Open GL takes care of graphical pictures rendering.

6.15 Data Structures

An extensive use of pointers have been made for data storage and retrieval. All the data regarding the graphical entities drawn is stored in the form of linked list. The details of each entity are stored by instantiating an object of the corresponding class of that entity.

Classes identified in the design of this project:
(a) Point: - stores the vertex.
(b) Line: - stores the endpoints.
(c) Polygon: - stores the edge information of the vertices.
(d) Curves: - stores the control points of the curve.
(e) Disk: - stores the inner and outer radii of the disk.
(f) Cone: - stores the base radius and height of the cone.
(g) Wire Box:- stores the length, breadth and height of the wire box.
(h) Wire Cube:- stores the side of the wire cube.

6.16 Coordinate Systems

(a) Model Coordinate System: - The Coordinate system used is a three-dimensional Right Handed Coordinate system. The origin of this coordinate system is at the intersection of the three axes. The default coordinate system is arranged such that the X-axis points horizontally to the right, the Y-axis points vertically upward and the Z-axis points outside the screen.
(b) Viewing Coordinate System: - Viewing Co-ordinate system is another 3 dimensional coordinate system similar to Model Coordinate system. The model is viewed through this clipping volume whose dimensions are set by default. The model data after converting to viewing coordinates is stored in the data models.
(c) Screen Co-ordinate system: The Screen Co-ordinate system is the ultimate 2 dimensional coordinate system with the positive X-axis along the horizontal from the bottom left corner of the display screen to the top right of the screen. The positive Y-axis lies along the vertical from the top left corner of the display screen to the bottom left corner of the screen. The corresponding screen coordinate value of the model is not stored in the database.

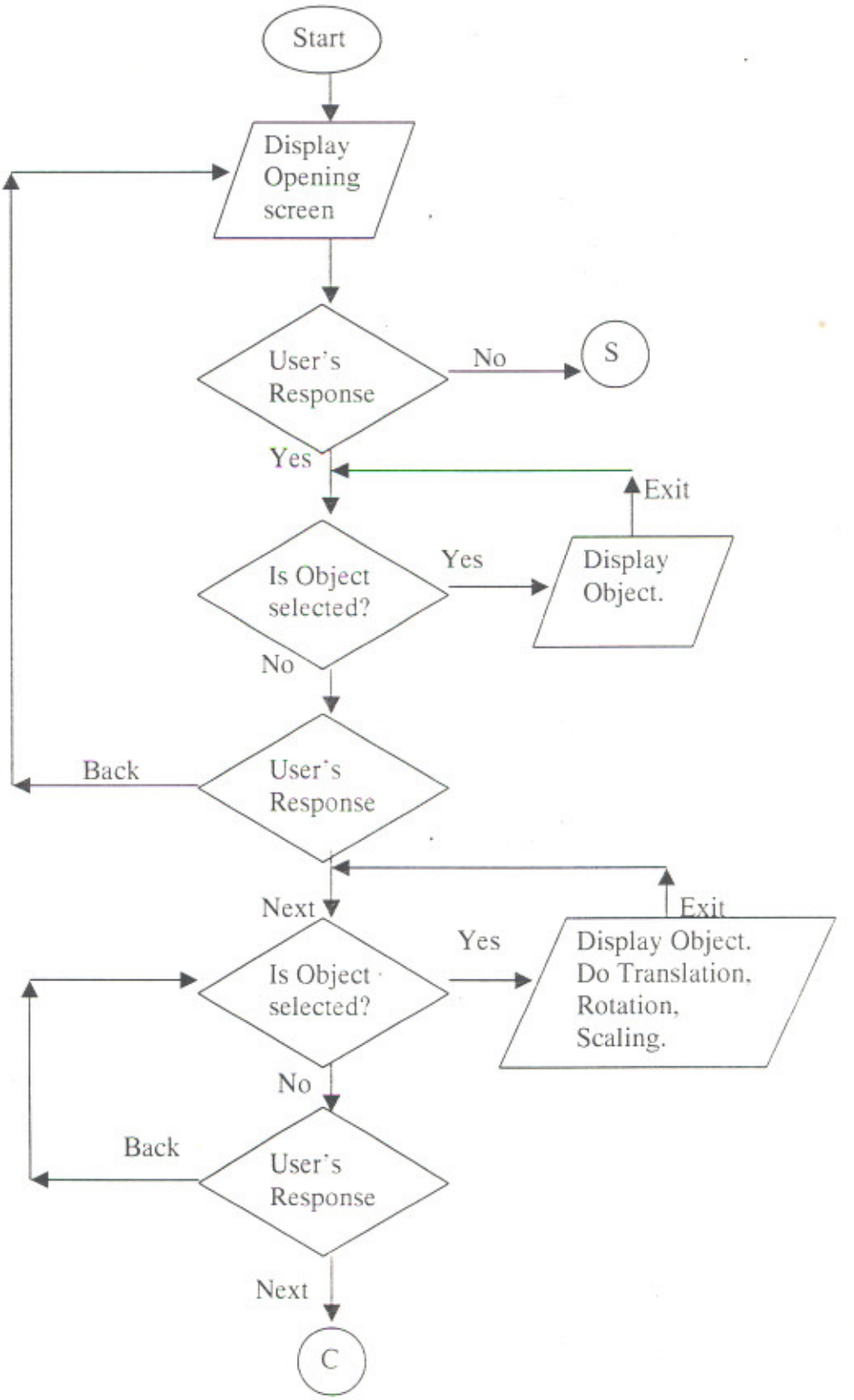
6.17 Creation of graphical primitives

To create an object, the user has to select the desired object from a list of objects shown in the screen and click the concerned button. After getting the graphical object , the user can apply translation, rotation, scaling, dynamic rotation and dynamic paning to that object by clicking the corresponding menu items.

6.2 Detailed Design

Detailed Design is the second phase in the Design of the software system. In this phase, the internal design of the objects is decided. Detailed design essentially expands the system design to contain a more detailed description of the processing logic and data structures such that the design is sufficiently complete for coding.

The behavioral or dynamic view of the system can be provided with the flowchart shown in fig.1.


Start
Display Opening screen
User's Response
No
S
Yes
Exit
Is Object selected?
Yes
Display Object.
No
Back
User's Response
Next
Exit
Yes
Is Object selected?
Display Object. Do Translation, Rotation, Scaling.
No
Back
User's Response
Next
C

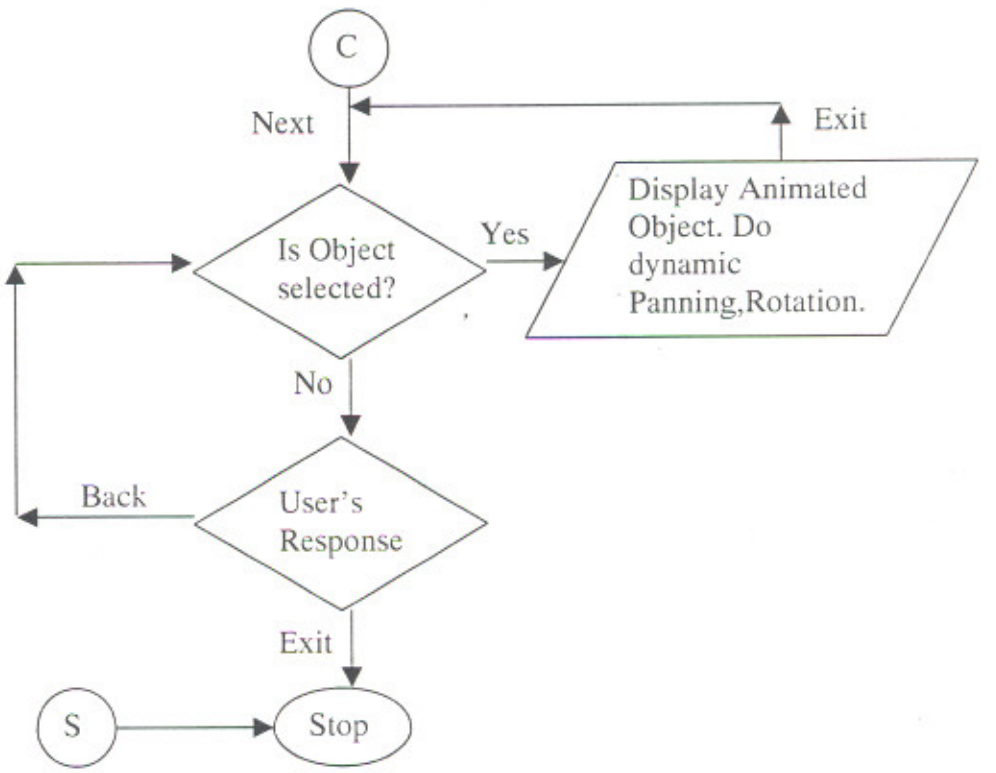


Fig.1. Generalized working and interaction Sequence

THE USER INTERFACE

7. THE USER INTERFACE

The project "A GUI in VC++(6.0) for Creating OpenGL Objects and Animation" generates five screens.

1. The first screen shows the message , "Welcome, Do you want to start the Project?" with two buttons labelled as "YES" and "NO".
   - If the button "NO" is clicked, it results in exiting from the program.
   - If the button "YES" is clicked, it results in showing second screen.

2. The second screen shows the message "Creation of OpenGL Objects and Animation" . It contains various OpenGL objects with their corresponding buttons. To create an OpenGL object, the corresponding button is to be clicked. The window on which the object is displayed contains a menu bar with menu items File,Edit,View and Help.
   The menu items in File are as follows:
   - New: Creates a new file.
   - Open: Opens an existing file.
   - Save: Saves the file.
   - Save As: Saves the file as the name given by the user.
   - Exit: Closes the window on which the object is displayed.

   The menu item Edit is used for editing
   The menu item View is used for seleting the options,Toolbar, Status Bar.
   Help contains a menu item About Object. It tells that the obect is created by using OpenGL.

The second screen also contains two more buttons , namely, "Back" and "Next".

- If the button "Back" is clicked the previous(first) screen will be shown.
- If the button "Next" is clicked, the next(third) screen will be shown.

3. The third screen contains the same heading as the second screen and also it specifies translation,rotation and scaling. By choosing an object and

clicking the concerned button, the corresponding object is created. The window on which the object is displayed contains the same menu items as described in the second screen. In addition, it contains one more menu item for that concerned object. The additional menu item contains the following menu items. Each of these items has a popup menu.

- Size : Scales the object
  - Increase: Increases the size of the object.
  - Decrease: Decreases the size of the object.
- Rotation: Rotates the object.
  - About X: Rotates the object about X- axis.
  - About Y: Rotates the object about Y- axis.
  - About Z: Rotates the object about Z- axis.
- Translation: Translates the object.
  - along x: Translates along X- direction.
  - along y: Translates along Y- direction.
  - along z: Translates along Z- direction.

The fourth screen contains two more buttons, namely, "Back"and "Next".

- If the button "Back" is clicked, the previous(second) screen will be shown.
- If the button "Next" is clicked, the next(fourth) screen will be shown.

4. The fourth screen contains the same heading as the second screen and also it specifies animation. By choosing an object and clicking the concerned button, the corresponding object is created. . The window on which the object is displayed contains the menu items, namely, File, Object name(Cube), Help.
   - The menu item File contains:
     - Exit: Closes the window on which the object is displayed.
   - Menu item Cube(Object) contains:
     - Start Rotation: Starts rotating the cube.
     - Stop Rotation: Stops rotating the cube.
     - Increase Size: Increases the size of the cube.
     - Decrease Size: Decreases the size of the cube.

- Pan: Moves the cube dynamically on the screen in any direction by keeping the mouse right button depressed and moving.

- The menu item Help contains:
  - About Animation: It shows the dialog containing the message "This is an example for OpenGL Animation".
- By clicking the mouse left button, the rotating object can be stopped.
- The object can be rotated dynamically by moving the mouse and keeping the mouse left button depressed.

The fourth screen contains two more buttons, namely, "Back"and "Next".

- If the button "Back" is clicked, the previous(third) screen will be shown.
- If the button "Next" is clicked, the next(fifth) screen will be shown.

5. The fifth screen shows the message , "Thank You". It specifies the names of both internal and external guides .This fifth screen contains two buttons, namely, "Back" and "Exit".
   - If the button "Back" is clicked, the previous(fourth) screen will be shown.
   - If the button "Exit" is clicked, it results in exiting from the program.

FUTURE SCOPE AND DEVELOPMENT

8. FUTURE SCOPE AND DEVELOPMENT

This project entitled "A GUI in VC++(6.0) for Creating OpenGL Objects and Animation " provides a platform for carrying out certain basic designing tasks. Its capabilities can be enhanced into a perfect designing package, but involves a lot of time and hard work. But it has been designed in a generalized way, thus providing an established path to enhance it's capabilities in future to be a competitive graphics package.

A categorized elucidation of the future scope and possible development areas of this software can be summarized as follows: -

Surface Modeling

As some design application areas require complicated surfaces in modeling, this software can be enhanced to include surface generation and surface modification independently.

Library

A software can be made application specific by providing a library involving certain common components required in modeling. Such a library greatly reduces the time spent on drawing on computer.

Sweep Representations

This software has not incorporated the Sweeping techniques, such as linear sweep , circular sweep and the like.

Coordinate System

This software has used absolute Cartesian Coordinate system, but can also incorporate polar and spherical coordinate systems.

To make a graphics software, a perfect CAD package, many more design techniques have to be incorporated. It's a continuous exercise to evaluate various design techniques to be incorporated in a software to make it complete for its use in various application areas.

CONCLUSION

9. CONCLUSION

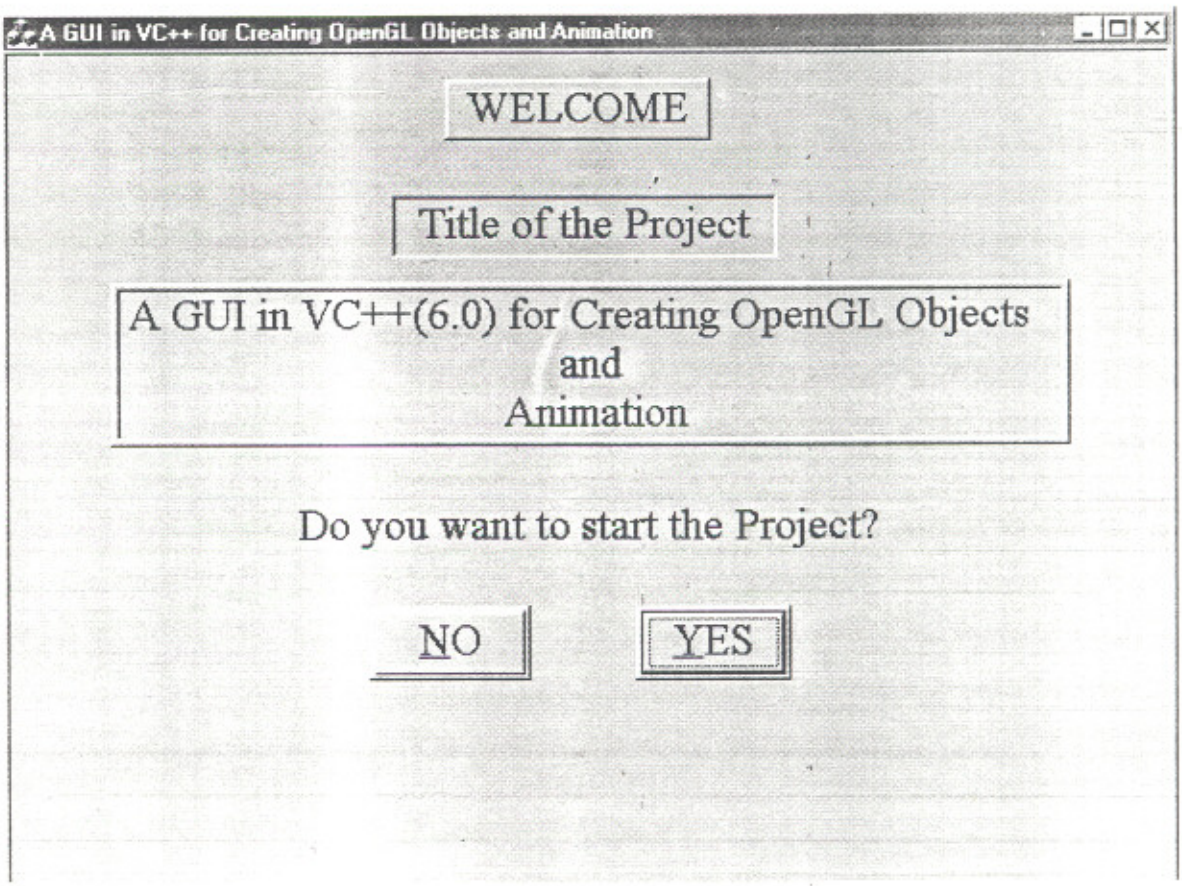
This project entitled " A GUI in VC++(6.0) for Creating OpenGL Objects and Animation" is developed for creating high quality still and animated 3- Dimensional color objects with graphic effects such as shading, lighting, texture mapping and animation. These objects are created by using OpenGL, a portable and high quality 3D graphics API . It provides necessary tools for drawing and viewing the developed model with various visualization effects such as lighting, different views, dynamic rotation, dynamic pan, zooming option and so on. The GUI is developed in VC++(6.0).

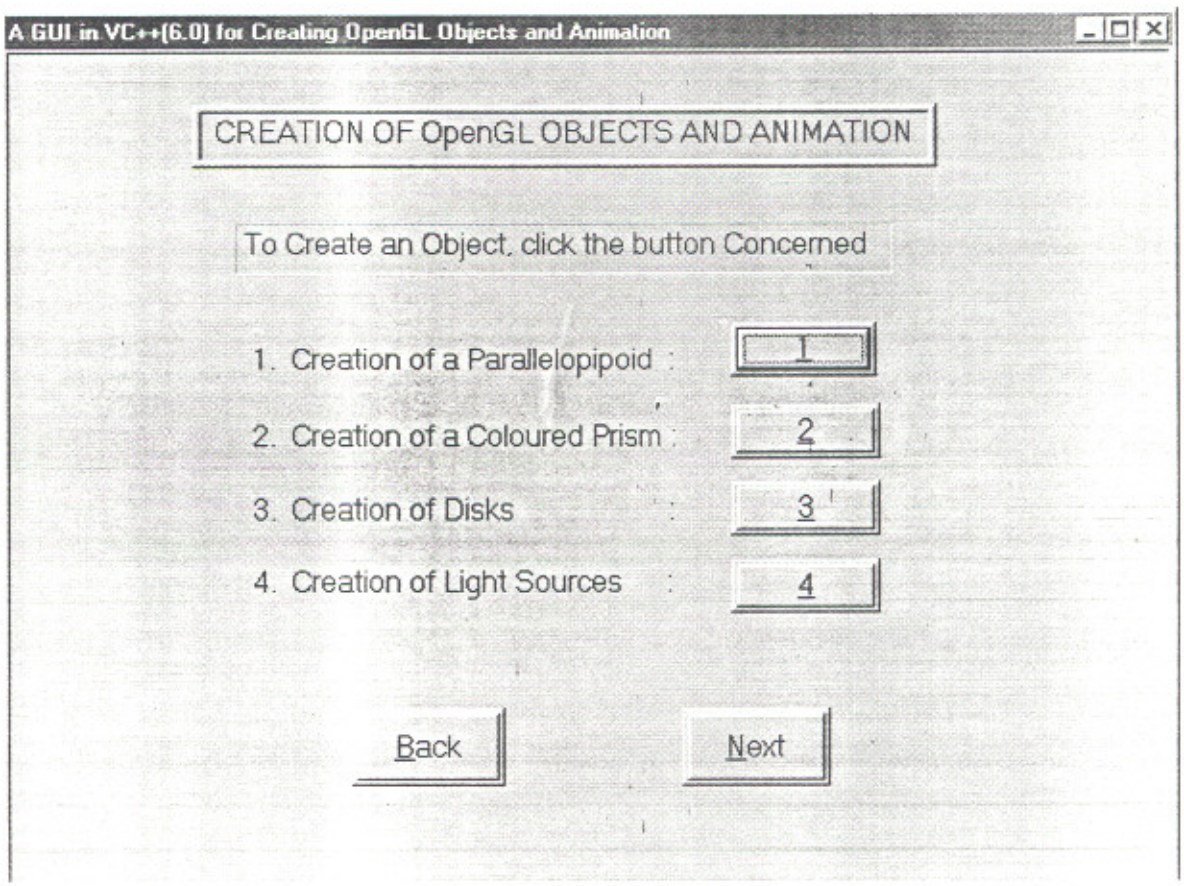
SCREENS

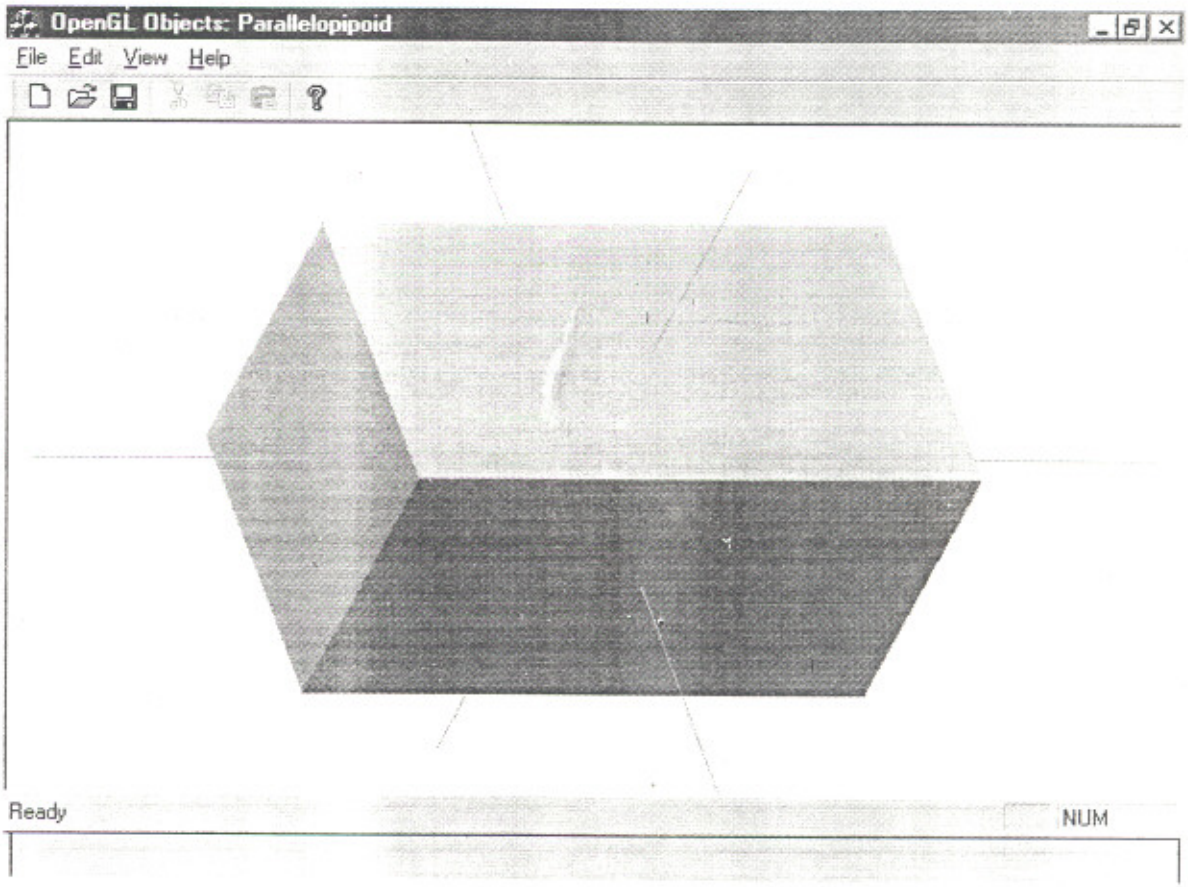
SCREENS

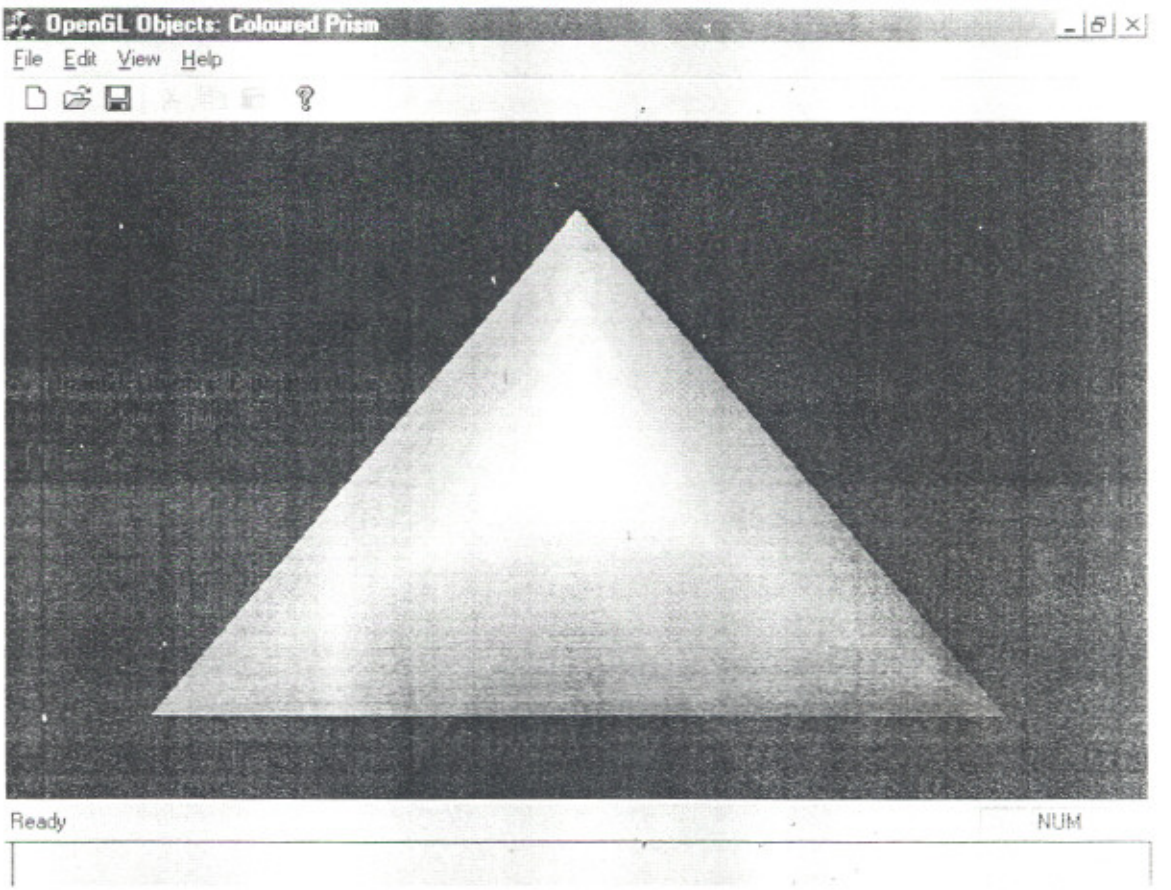

A GUI in VC++ for Creating OpenGL Objects and Animation
WELCOME
Title of the Project

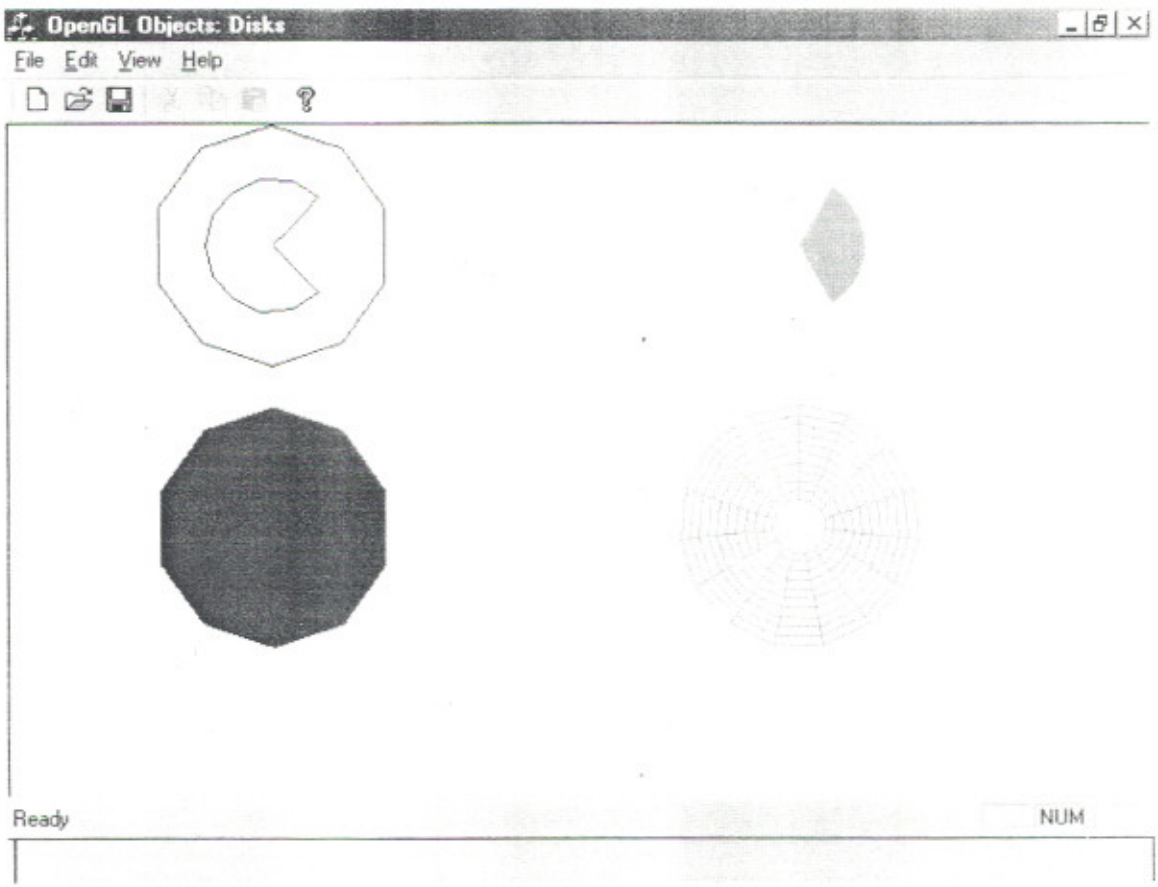
SCREENS

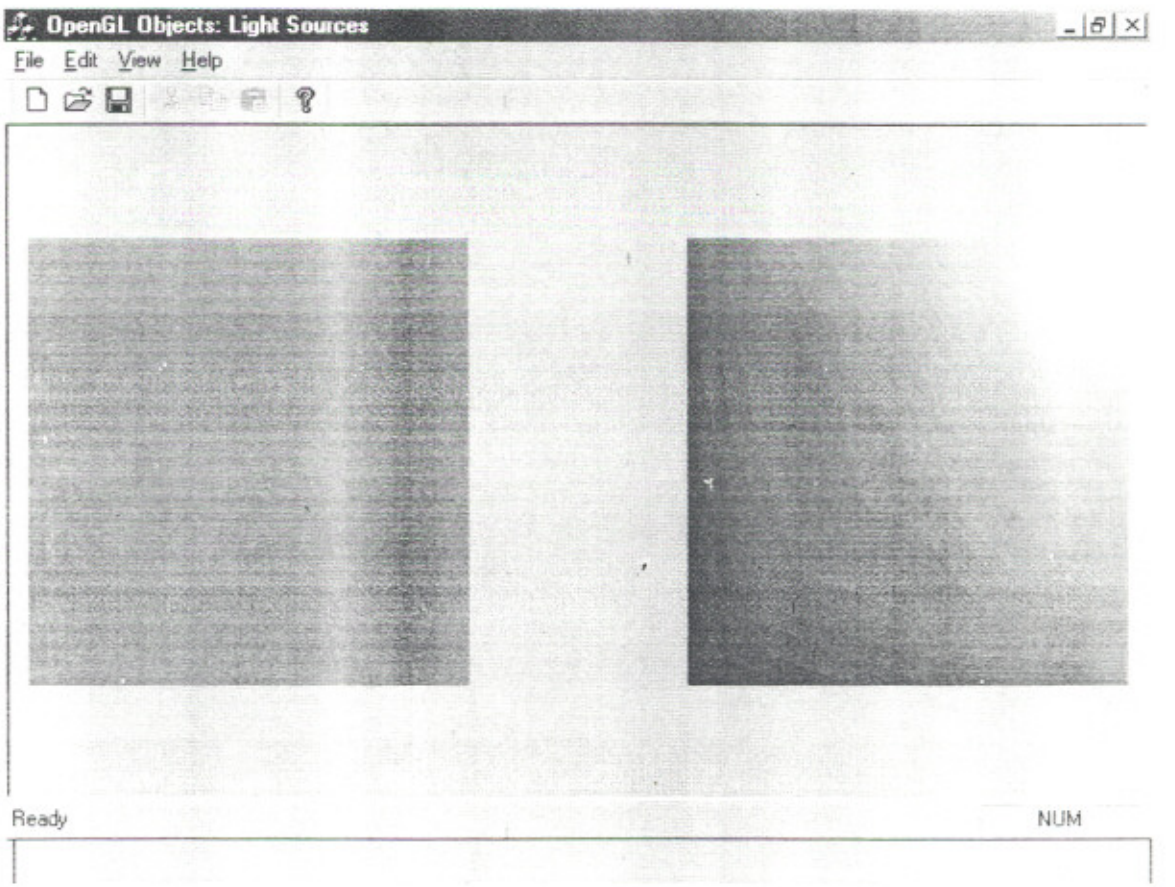

A GUI in VC++ for Creating OpenGL Objects and Animation
WELCOME
Title of the Project
A GUI in VC++(6.0) for Creating OpenGL Objects
and
Animation
Do you want to start the Project?
NO
YES

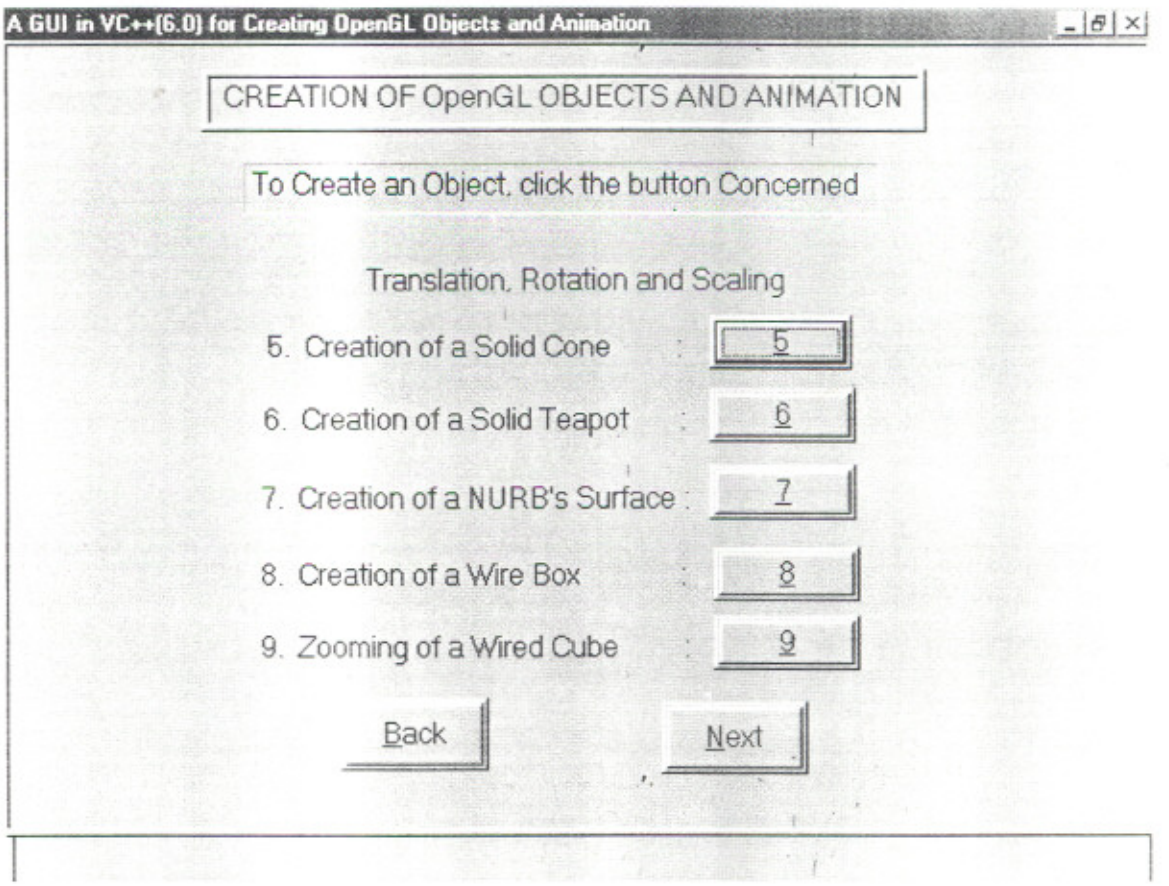


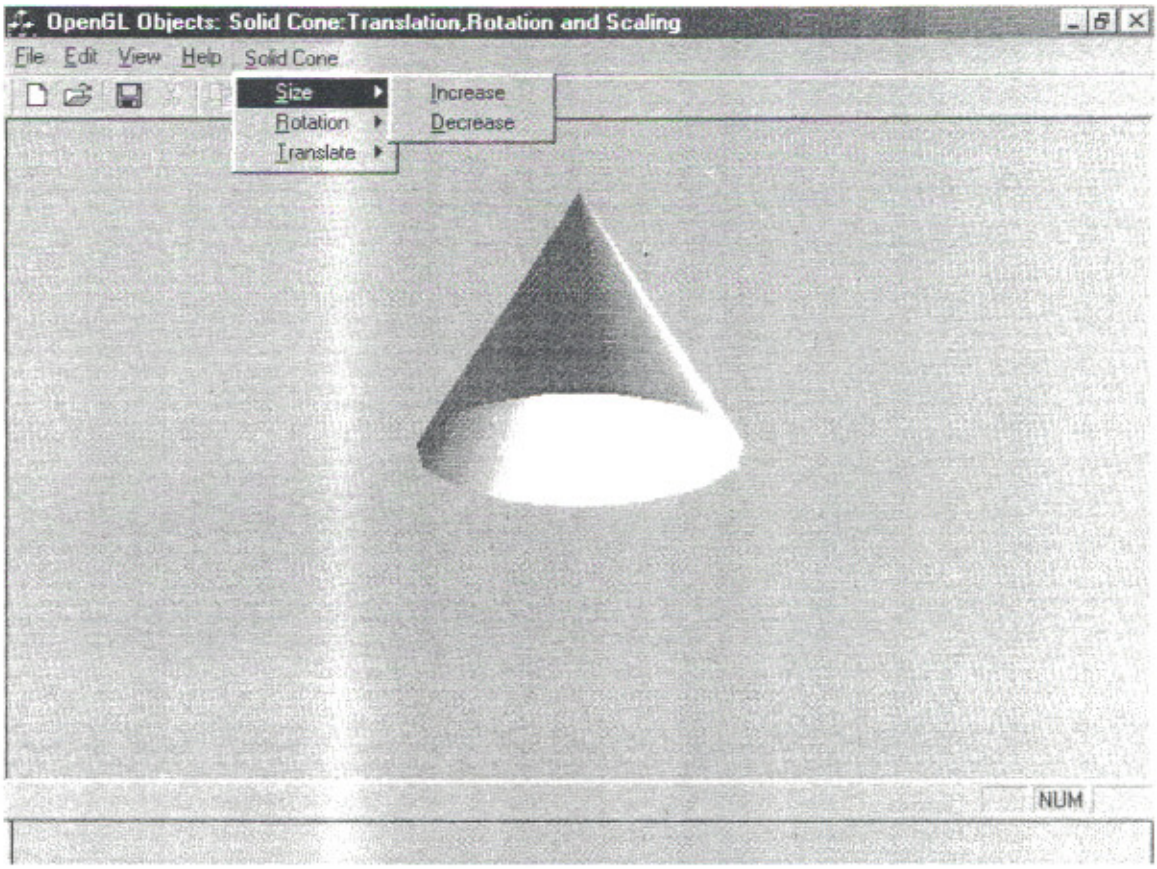

OpenGL Objects: Parallelopipoid
File Edit View Help
Ready
NUM

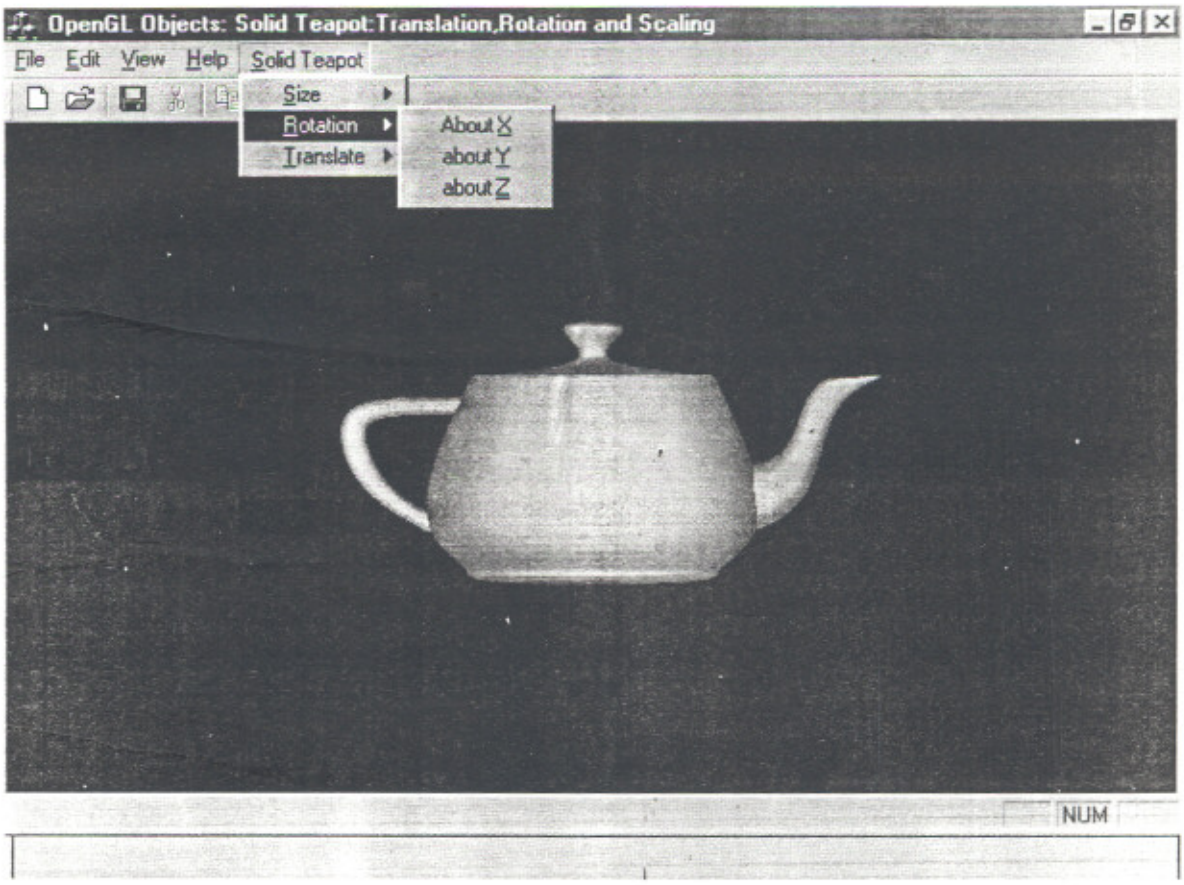

OpenGL Objects: Coloured Prism
File Edit View Help
Ready
NUM

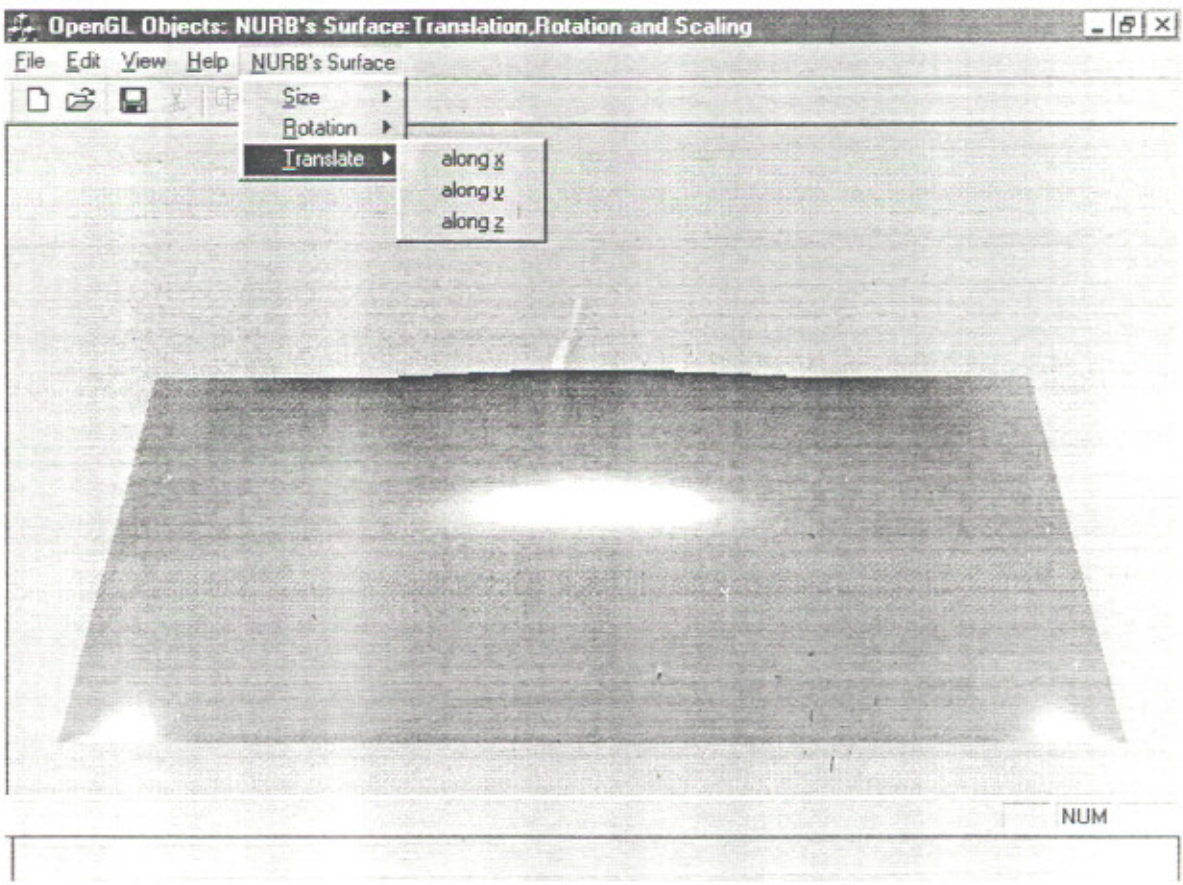

OpenGL Objects: Disks
File Edit View Help
Ready
NUM

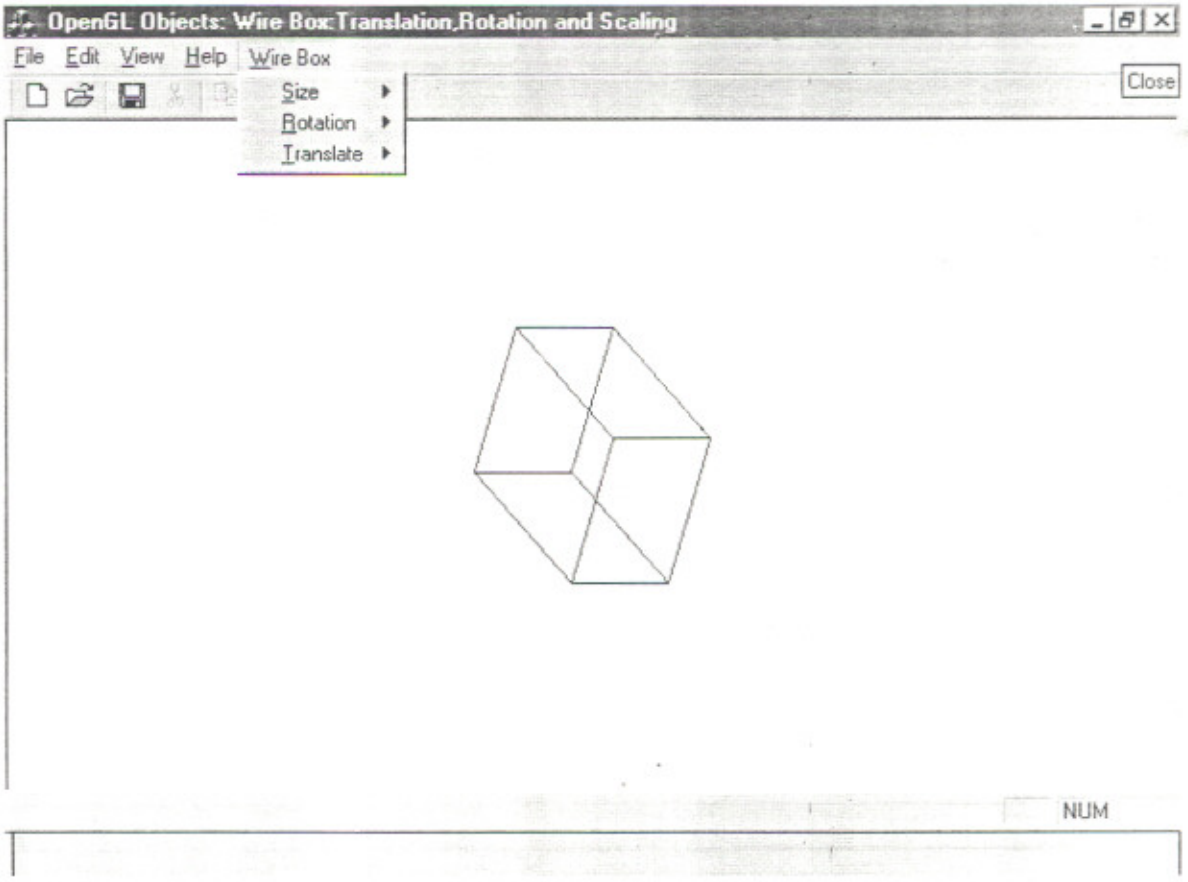

OpenGL Objects: Light Sources
File Edit View Help
Ready
NUM

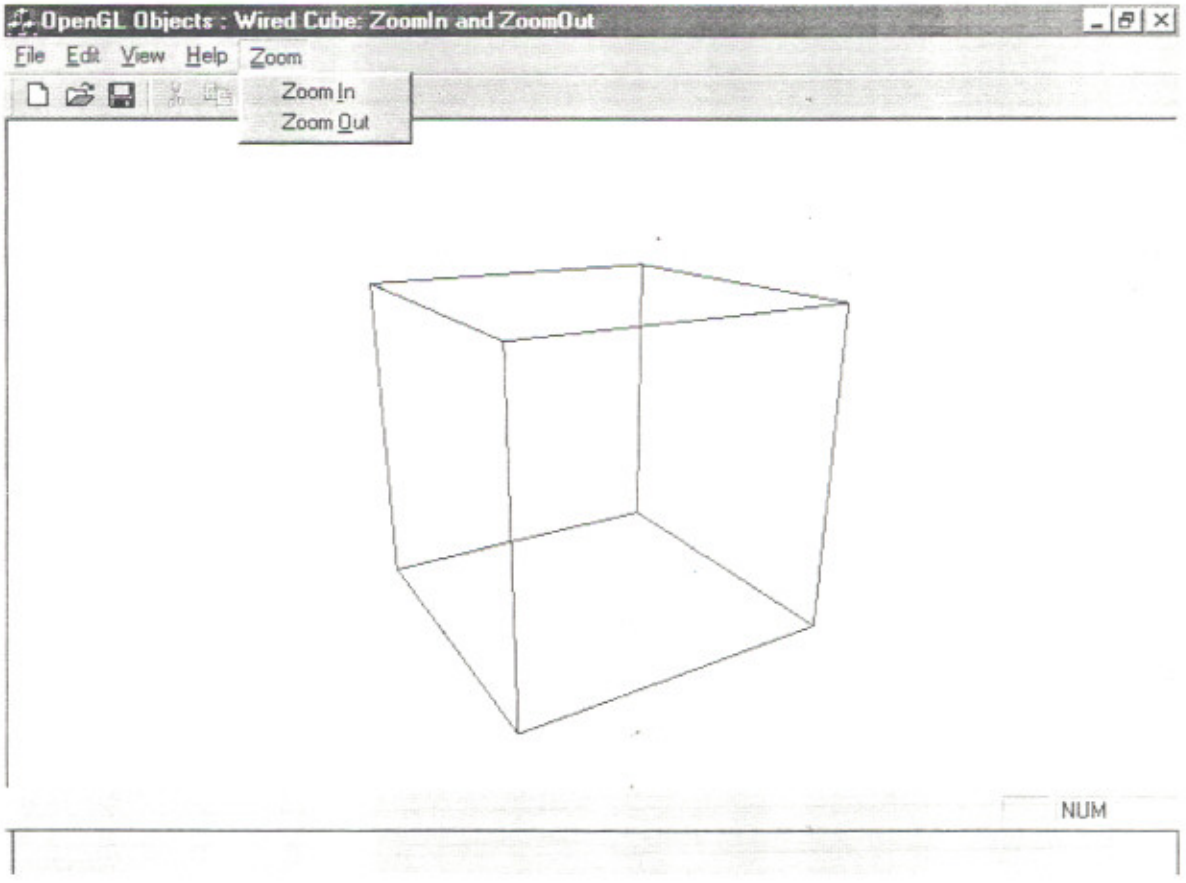

A GUI in VC++(6.0) for Creating OpenGL Objects and Animation
CREATION OF OpenGL OBJECTS AND ANIMATION
To Create an Object, click the button Concerned
Translation, Rotation and Scaling
5. Creation of a Solid Cone : 5
6. Creation of a Solid Teapot : 6
7. Creation of a NURB's Surface : 7
8. Creation of a Wire Box : 8
9. Zooming of a Wired Cube : 9
Back
Next

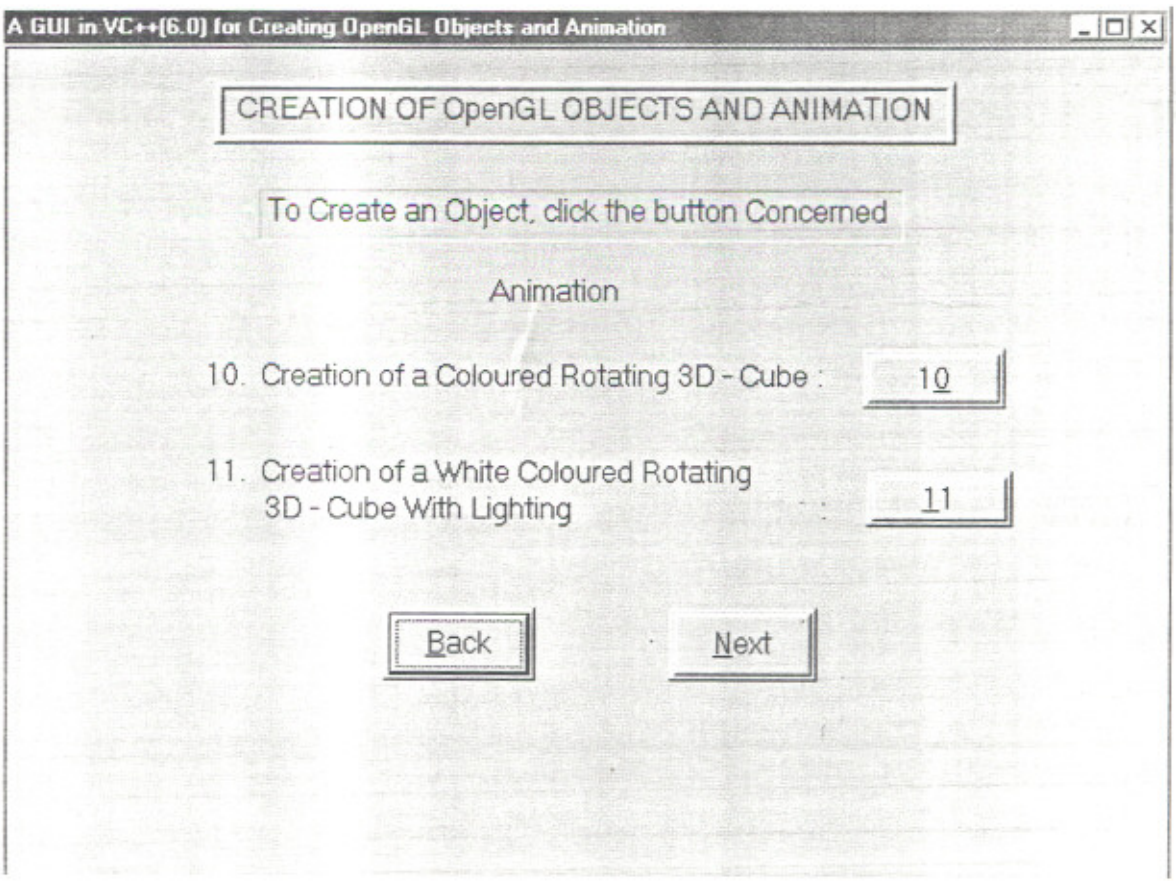

OpenGL Objects: Solid Cone:Translation,Rotation and Scaling
File Edit View Help Solid Cone
Size
Rotation
Translate
Increase
Decrease
NUM

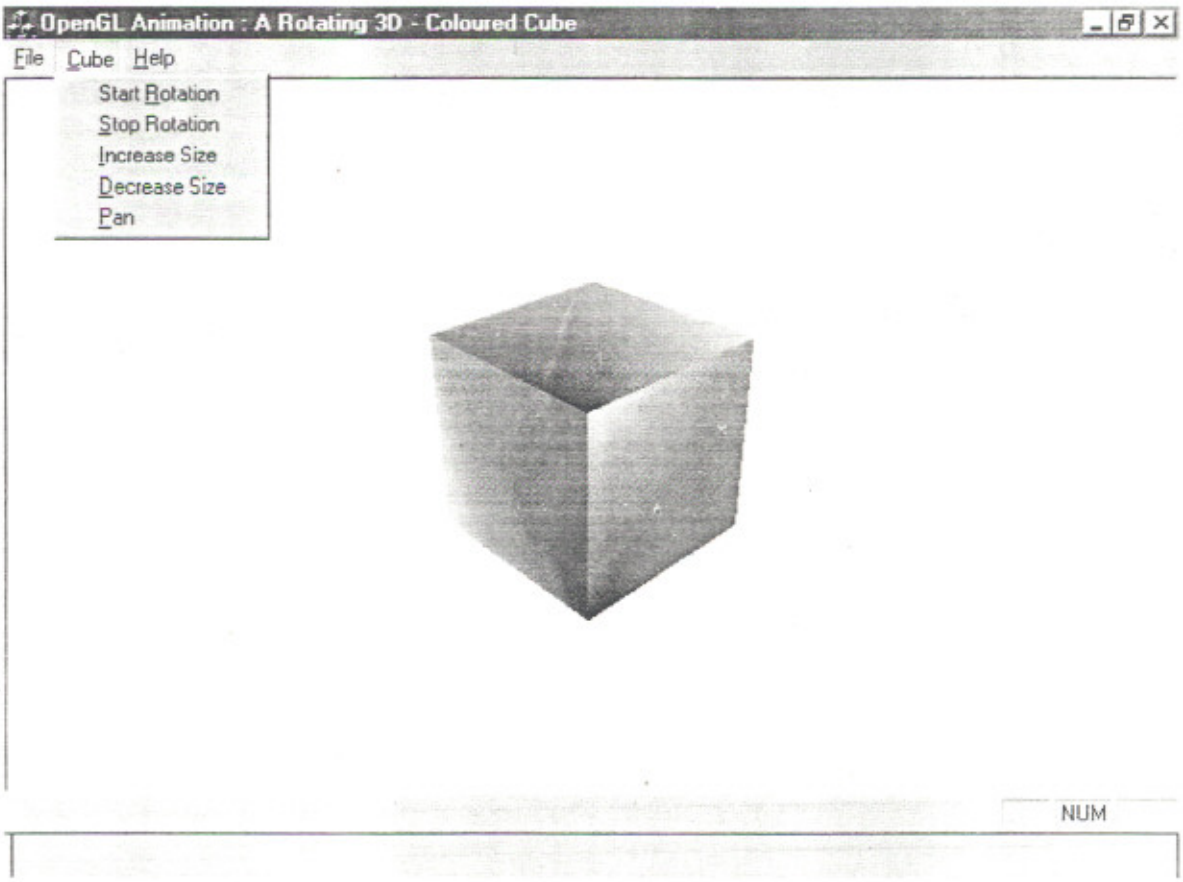

OpenGL Objects: Solid Teapot:Translation,Rotation and Scaling
File Edit View Help Solid Teapot
Size
Rotation
Translate
About X
about Y
about Z
NUM

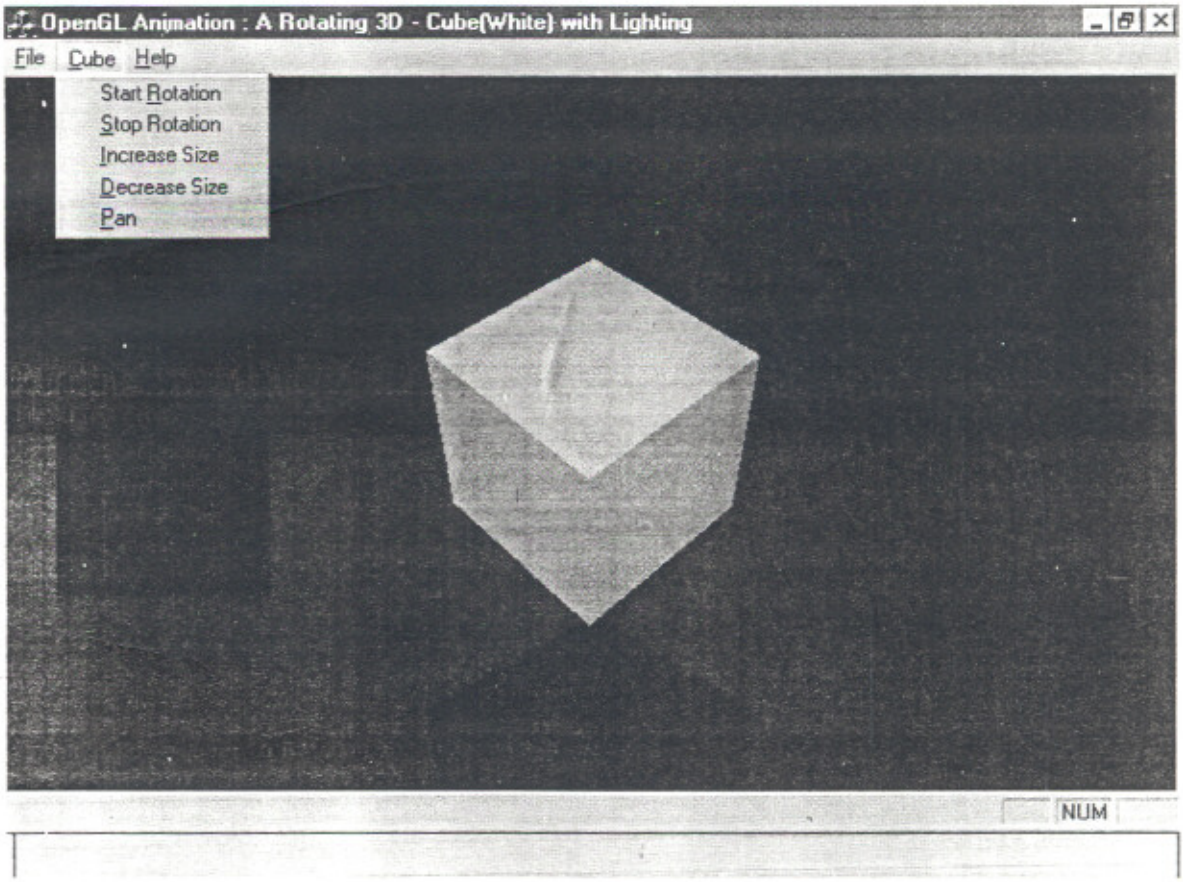

OpenGL Objects: NURB's Surface:Translation,Rotation and Scaling
File Edit View Help NURB's Surface
Size
Rotation
Translate
along x
along y
along z
NUM

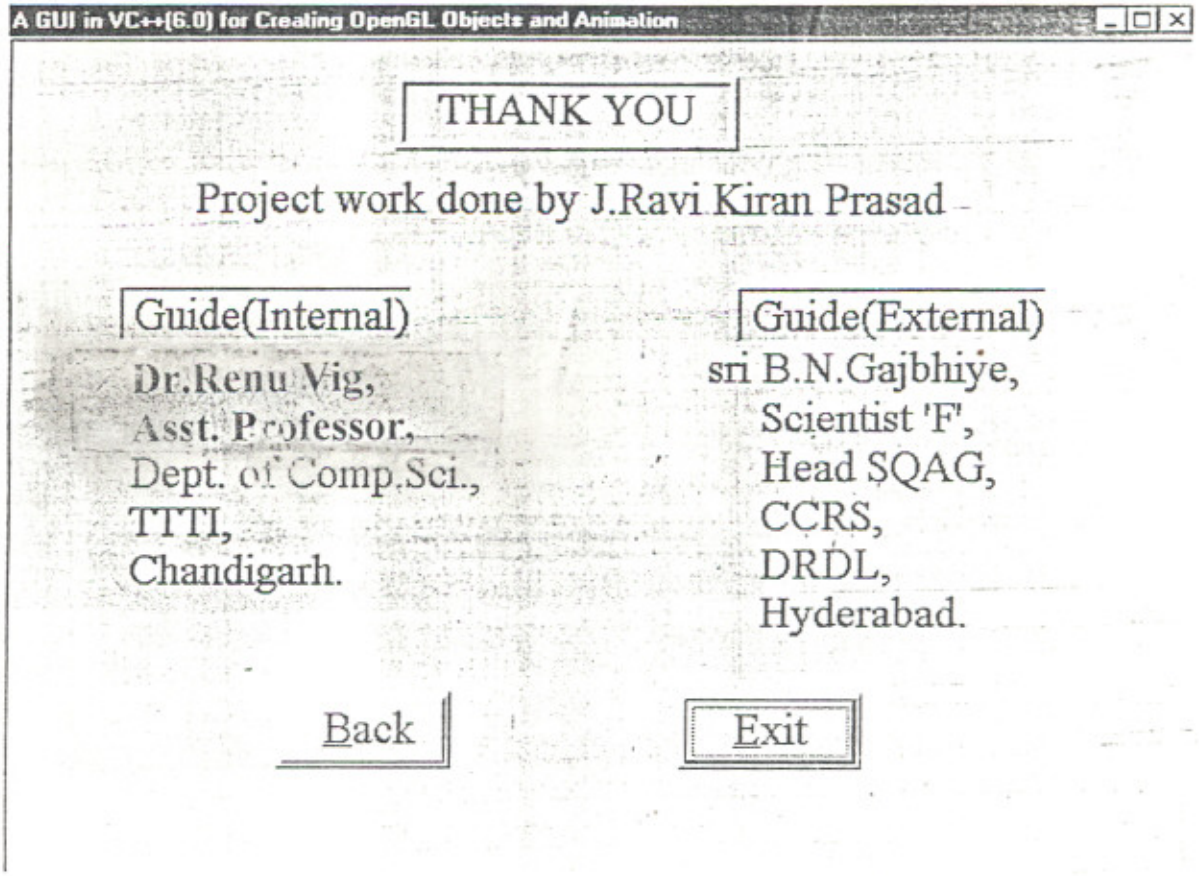

OpenGL Objects: Wire Box:Translation,Rotation and Scaling
File Edit View Help Wire Box
Size
Rotation
Translate
Close
NUM


OpenGL Objects : Wired Cube: ZoomIn and ZoomOut
File Edit View Help Zoom
Zoom In
Zoom Out
NUM


A GUI in VC++(6.0) for Creating OpenGL Objects and Animation
CREATION OF OpenGL OBJECTS AND ANIMATION
To Create an Object, click the button Concerned
Animation
10. Creation of a Coloured Rotating 3D - Cube :
10
11. Creation of a White Coloured Rotating 3D - Cube With Lighting
11
Back
Next


OpenGL Animation : A Rotating 3D - Coloured Cube
File
Cube
Help
Start Rotation
Stop Rotation
Increase Size
Decrease Size
Pan
NUM


OpenGL Animation : A Rotating 3D - Cube(White) with Lighting
File
Cube
Help
Start Rotation
Stop Rotation
Increase Size
Decrease Size
Pan
NUM


A GUI in VC++(6.0) for Creating OpenGL Objects and Animation
THANK YOU
Project work done by J.Ravi Kiran Prasad
Guide(Internal)
Dr.Renu Vig,
Asst. Professor,
Dept. of Comp.Sci.,
TTTI,
Chandigarh.
Guide(External)
sri B.N.Gajbhiye,
Scientist 'F',
Head SQAG,
CCRS,
DRDL,
Hyderabad.
Back
Exit

REFERENCES

REFERENCES

REFERENCES

1. *"Computer Graphics - A Programming Approach"*
   by Steven Harrington

2. *"Principles of Interactive Computer Graphics"*
   by William M. Newman & Robert F. Sproull.

3. *"OpenGL Super Bible"*
   by Richard S. Wright and Michael Sweet.

4. *"OpenGL programming for Windows 95 and WindowsNT"*
   by Ron Fosner

5. "*OpenGL Programming Guide*"
   by Neider, Jackie, Tom Davis and Mason Woo.

6. *"Visual C++ 6.0 - In Record time"*
   by Steven Holzner

7. *"Visual C++(6.0) in 21 days"*
   by Sams Publications

8. *"C++ -The Complete Reference"*
   by Herbert Schildt

9. "*C++ - An Introduction to Programming*"
   by Jesse Liberty & Jim Keogh

10. *"An Integrated Approach to Software Engineering"*
    by Pankaj Jalote

11. "*Power-3D, High-Speed 3D Graphics in Windows 95/NT*"
    by Kyle Ernest Lussier

Web Sites

1. www.opengl.com
2. www.sgi.com
3. www.opengl.org