**BANGALORE UNIVERSITY**

**University Visvesvaraya College of Engineering (UVCE)  
K.R CIRCLE , BENGALURU-560001.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**COMPUTER GRAPHICS PACKAGE IN OPENGL**

SUBMITTED BY

**AARSI**

**V SEMESTER ISE**

**UNDER THE GUIDANCE OF MRS.VIMALA H S**

Associate Professor Dept of CSE

**December 2016**

**University Visvesvaraya College**

**of Engineering**

**BANGALORE UNIVERSITY**

**University Visvesvaraya College of Engineering (UVCE)  
 K.R CIRCLE , BENGALURU-560001.**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

**NAME: AARSI BRANCH: ISE**  
**SEM:** V SEM (B.E) **REG NO:** 14GAEI6001  
**SUB:** COMPUTER GRAPHICS LAB

**PROJECT TITLE:** GRAPHICS PACKAGE IN OPENGL

This is to certify that I have successfully completed the mini-project of **Computer Graphics Lab** Prescribed by the Bangalore University for the academic year 2016

----------------------- -------------------------  
**Mrs. VIMALA H S** **Dr.ARUNALATHA J** Associate Professor Chairperson  
Dept. of CSE Dept. of CSE

Examiner 1:\_\_\_\_\_\_\_\_\_\_\_\_ Examiner 2:\_\_\_\_\_\_\_\_\_\_\_\_

**ACKNOWLEDGEMENT**

I take this opportunity to thank our institution **UNIVERSITY** **VISVESVARAYA COLLEGE OF ENGINEERING** for having given us the opportunity to carry out this project.

I would like to thank **Dr.Venugopal K.R, Principal, UVCE** for providing us all the facilities to work on this project. I am indebted to him for being our pillar of strength and inspiration.

I wish to place my grateful thanks to **Dr.Arunalatha J S, Chairperson, Department of Computer Science and Engineering, UVCE** who has helped me to make this project a great success.

I am grateful to acknowledge **Mrs.Vimala H S, Associate Professor** for her valuable suggestions and relentless support which have sustained me throughout the course of the project.

I also thank the lab assistants **Mr.** **Babu, Mr. Srinivas, Mr.Harish** and Office staff for their extensive support throughout the CG Laboratory.

I thank my **parents** and **friends** for their help, encouragement and support.

Last but not the least, I thank **God Almighty**, without whose blessings I could not have done anything.

**Aarsi**

**ABSTRACT**

The Dev C/C++ editor allows the user to create a document, revise and save it.

Modern day graphical editors can take inputs and commands from multiple input devices such as keyboard and mouse. Graphical editors can be classified as:

1 .Menu driven 2.Icon driven

In a menu driven editor, icons represent all functions and clicking on them performs the desired action.

The functions of this GRAPHICS PACKAGE using menu driven approach can be summarized as follows:

1. Creation of window.

2. Creation of working area.

3. Creation of icons.

4. Each icon directs to its respective program.

5. The output of each program is visible in the working area.

**CONTENTS**

**Topic Page No.**

**1. Introduction 1**

**2. Requirements Specifications 2**

**3. About OpenGL 3**

**4. To access GLUT in dev c++ 5**

**5. Snapshots for creating a project and**

**adding parameters 6**

**6. Flowchart 7**

**7. Single Program Integration 8**

**8. Stroke 12**

**9. Snap shots 16**

**10. Conclusion 22**

**11. Reference 23**

**12. Appendix 24**

**1. INTRODUCTION**

**Overview of the project**

It is developed in C using OpenGL and implemented in the WINDOWS platform. The graphics package designed here provides an interface for the users for handling display and manipulation of basic picture objects. The interface is user-friendly with icons, menus.

**General Constraints**

1. As software is being built to run on a WINDOWS platform, efficient use of the memory is very important.

2. The code should be efficient and optimal with the minimal redundancies.

3. Needless to say, the package should also be robust and fast.

**Assumptions and Dependencies**

1. It is assumed that the standard output device, namely the monitor, supports colors and users system is required to have the C compiler for the appropriate version.

2. The system is also expected to have a mouse connected since most of the drawing and other graphical operations implemented assume the presence of a mouse.

**2. REQUIREMENTS SPECIFICATION**

The requirement specifications of this project is not perfectly optimized. However the following hardware and software specifications were done to be of my best efforts. Here are the specifications:

1. **Hardware Requirements:**

The hardware requirements given here is minimal requirements for the project to run even though the project can smoothly run on almost all i3h86 machines.

* Processor Speed -300 Mhz and above
* Ram Size -64 Mb or above
* Storage Space -2MB or above

1. **Software Requirements:**

* Operating System -Windows Family
* Compiler -Dev C++/CODBLOCKS
* Graphics Library -glut.h
* Programming Language -C using OpenGL

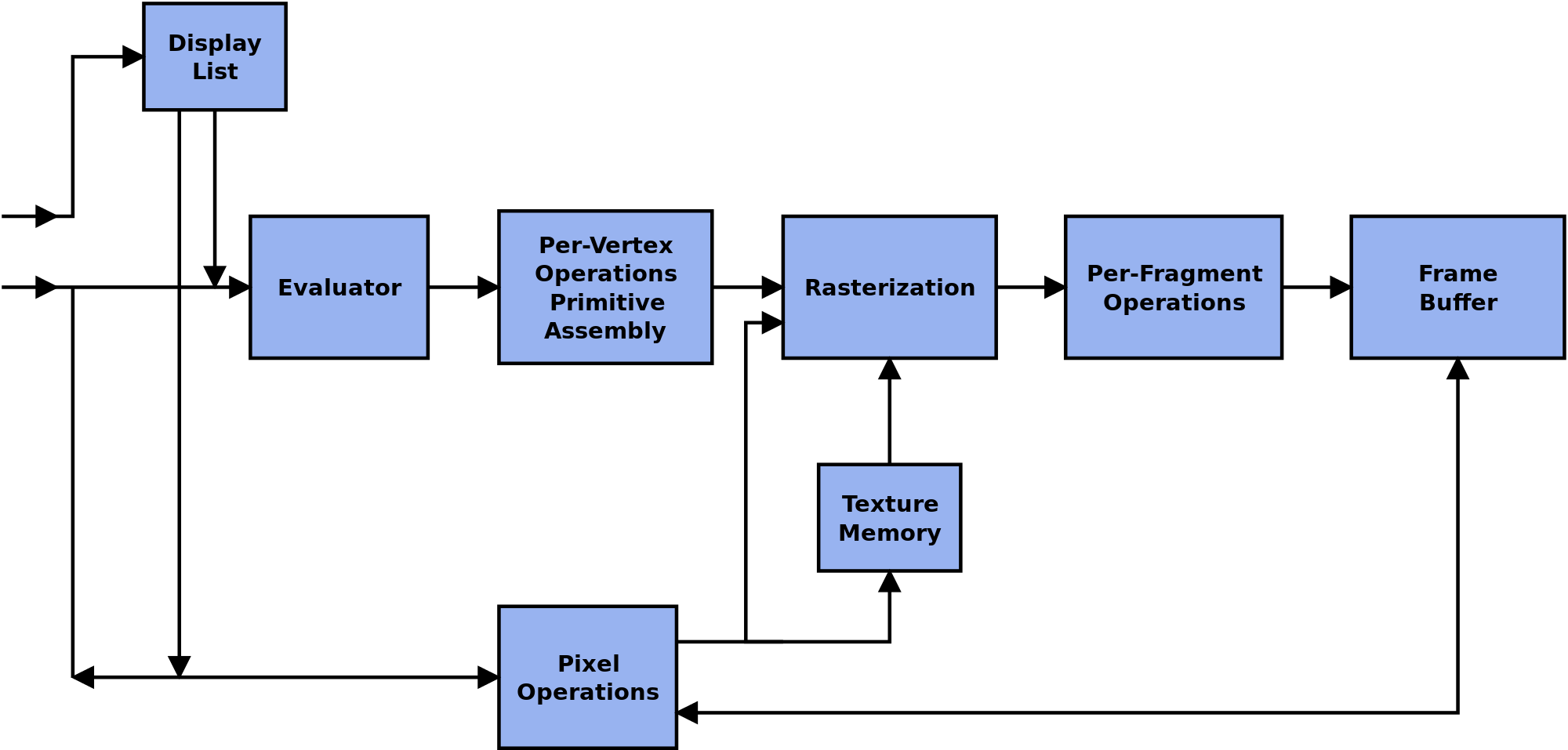
**3. ABOUT OPENGL**

The OpenGL specification describes an abstract for drawing 2D and 3D graphics.Although it is possible for the API to be implemented entirely in software,it is designed to be implemented mostly or entirely in hardware.

The API is defined as a number of functions which may be called by the client program,alongside a number of named integer constants.Although the function definition are superficially similar to those of the C programming language,they are independent.As such, OpenGl has many language bindings,some of the most noteworthy being the JavaScript binding WebGL(API based on OpenGL ES2.0);the C bindings WGL ,GLX,CGL;the C binding provided by iOS and the Java and C binding provided by the Android.

In addition to being language-independent,OpenGL is also platform independent.The specification says nothing on the subject of obtaining ,managing an OpnGL context,leaving this as a detail of the underying windwing system.For the same reason,OpenGL is purely concerned with the rendering,providing no APIs related to input,audio,or windowing.

**OPENGL GRAPHICS PIPELINE ARCHITECTURE**



**4. TO ACCESS GLUT IN DEV - C++**

**1.** Click File/New/Project. Pick a name for the project .Click "C Project", Click on "Empty Project", Click "OK".

**2.** In "Create New Project", click "save".

**3.** Click "File/New/Source File" and in "Add source file to current project" click "Yes". You now get a screen where you can get edit the source file.

**4.** Type in a simple C program. Now click "File/Save As" and save the file as "hello.c". (Be sure the file extension is .c).

**5.** Tell Dev-Cpp what libraries need to be linked. Click "Project/Project Options".

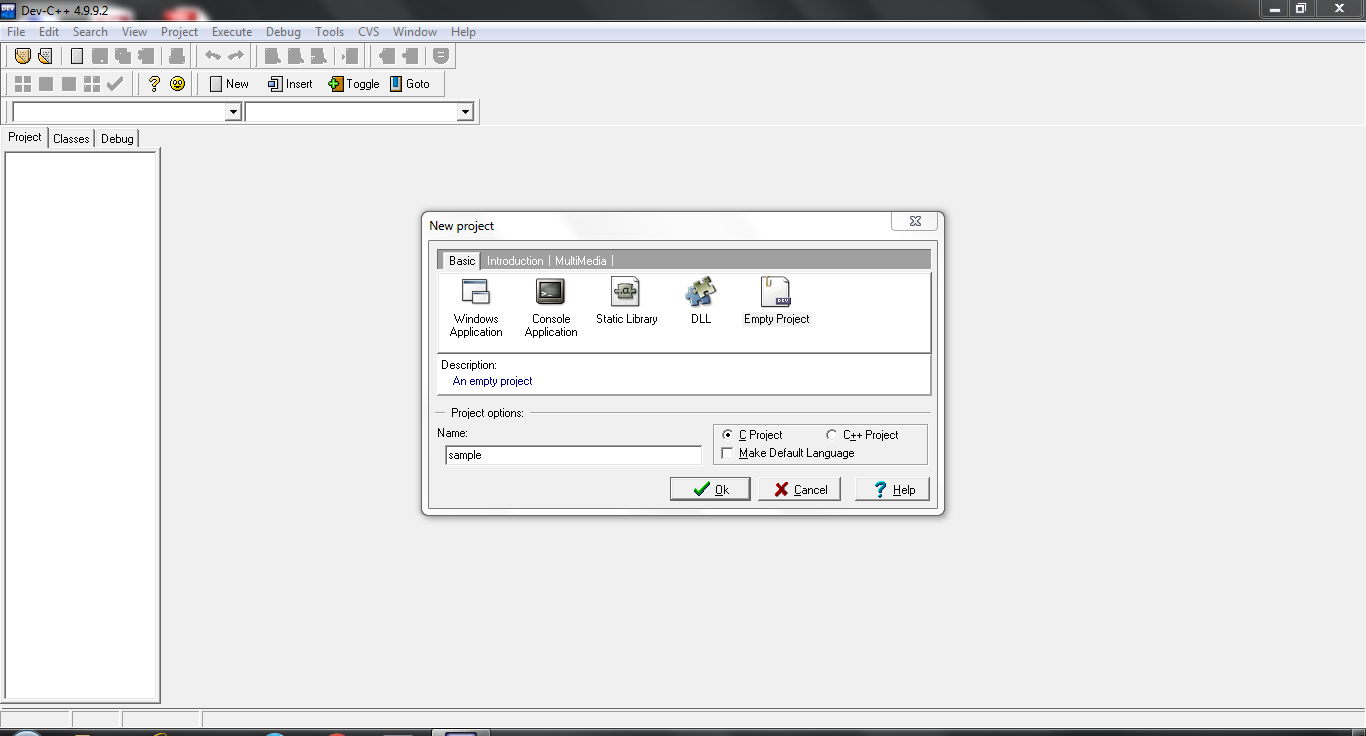
**6.** Now click "Parameters". Click the "Add Library or Object" button and navigate to the libraries that should be added, found under C:\Devv-Cpp\lib.

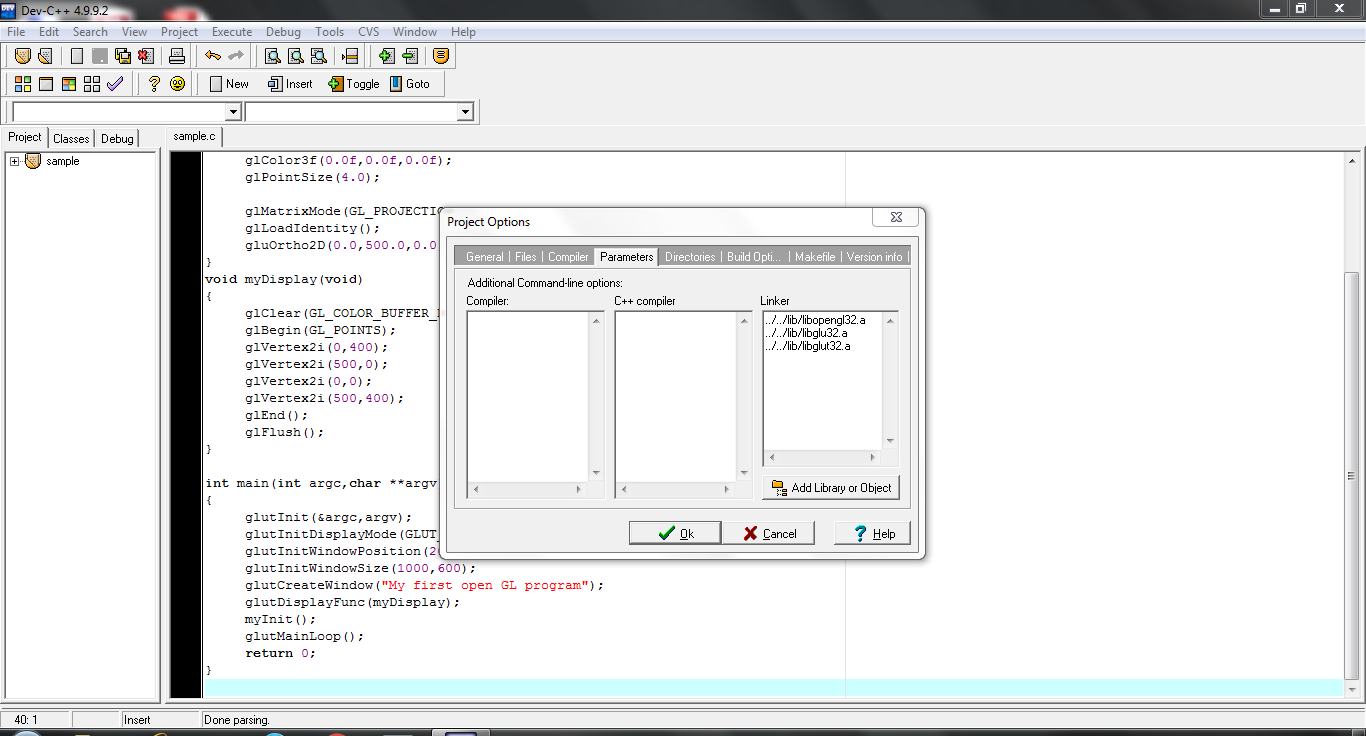
../lib/libopengl32.a

../lib/libglu32.a

../lib/libglut32.a

**5. SNAPSHOTS FOR CREATING A PROJECT AND ADDING PARAMETERS**





**6. FLOWCHART**

**INITIALZE THE WINDOW**

**CREATE THE WINDOW**

**CREATE WORKING AREA AND ICONS ON THE WINOW**

**CREATE BUTTON**

**START**

**LEFT CLICK OF MOUSE ICON**

**TETRAHEDRON**

**LIANG BARSKY**

**HOUSE**

**CUBE SPIN**

**COHEN**

**CYLLINDER**

**TEAPOT**

**CUBE VIEW**

**SCANLINE**

**MESH**

**STROKE**

**EXIT**

**STOP**

1. **SINGLE PROGRAM INTEGRATION**

**#include<stdio.h>**

**#include<process.h>**

**#include<window**

**#include<GL/glut.h**

**#include<math.h>**

**#include”program\_name.c”**

/\*all the eleven programs have to be included by writing their saved names\*/

/\* a function called write is defined, used to display the text on screen\*/

/\*glutBitmapCharacter renders a bitmap character using OpenGL.\*/

**void Write(char \*string){**

**while(\*string) {glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,\*string++);**

**}**

**}**

/\*Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font. \*/

**void OnMouseClick(int btn,int state,int x,int y){**

**if(btn==GLUT\_LEFT\_BUTTON&&state==GLUT\_DOWN){**

**printf("\n%d,%d",x,y);**

**if(x>=50&&x<=180&&y>=50&&y<=120){**

**spawnl(P\_NOWAIT, "tetra2.exe","tetra2.exe", NULL );**

**return;**

**}**

**if(x>=50&&x<=180&&y>=200&&y<=260){**

**Exit();**

**}**

**}**

**}**

/\*if the user clicks within these boundaries display function for tetrahedron i.e display is called,

/\*hence defining an if condition with boundaries of the button serve as implementation of a button

**void display()**

**{**

**glClear(GL\_COLOR\_BUFFER\_BIT); //clears the color buffer**

**glColor3f (0.45,0.0,0.45); //set the color**

**glBegin(GL\_POLYGON); //to draw the polygon**

**glVertex2f(-0.4,-1.0);**

**glVertex2f(-1.0,-1.0);**

**glVertex2f(-1.0,1.0);**

**glVertex2f(-0.4,1.0);**

**glEnd(); //to end drawing the polygon**

**glBegin(GL\_POLYGON); //to draw the polygon**

**glColor3f (0.7,0.65,0.8); //set the color**

**glVertex2f(-0.4,1.0);**

**glColor3f(0.95,0.89,1); //set the color**

**glVertex2f(1.0,1.0);**

**glColor3f (0.7,0.65,0.8); //set the color**

**glVertex2f(1.0,-1.0);**

**glColor3f(0.53,0.46,0.67); //set the color**

**glVertex2f(-0.4,-1.0);**

**glEnd();**

**glColor3f (0.45,0.0,0.45);**

**glBegin(GL\_LINE\_LOOP); //to draw the border**

**glVertex2f(-0.3,0.9);**

**glVertex2f(0.9,0.9);**

**glVertex2f(0.9,-0.9);**

**glVertex2f(-0.3,-0.9);**

**glEnd();**

**glBegin(GL\_LINE\_LOOP); //to draw the border**

**glVertex2f(-0.28,0.87);**

**glVertex2f(0.88,0.87);**

**glVertex2f(0.88,-0.87);**

**glVertex2f(-0.28,-0.87);**

**glEnd();**

**glColor3f(0.3,0.1,0.4); //set the color**

**glRasterPos2f(-0.255,0.67); //to provide position for rasterization**

**Write1("COMPUTER GRAPHICS PROJECT WITH OPENGL");**

**glBegin(GL\_LINES); //to underline**

**glVertex2f(-0.255,0.663);**

**glVertex2f(0.855,0.663);**

**glEnd();**

**glColor3f(0,0.2,0.4); //set the color**

**glRasterPos2f(0.02,0.47); //to provide position for rasterization**

**Write1("A MENU BASED PACKAGE");**

**glBegin(GL\_LINES); //to underline**

**glVertex2f(0.02,0.463);**

**glVertex2f(0.64,0.463);**

**glEnd();**

**glColor3f(0,0.2,0.7); //set the color**

**glRasterPos2f(0.08,0.17); //to provide position for rasterization**

**Write2(" BY:");**

**glColor3f(0.3,0.2,0.4); //set the color**

**glRasterPos2f(0.08,-0.07); //to provide position for rasterization**

**Write1("NAME :AARSI");**

**glRasterPos2f(0.08,-0.37);**

**Write1("REG NO:14GAEI6001");**

**glRasterPos2f(0.08,-0.47);**

**Write1("CLASS:V SEM ISE");**

**glColor3f(0.8,0.1,0.2); //set the color**

**glRasterPos2f(0.08,-0.67); //to provide position for rasterization**

**Write2(" GUIDE:");**

**glRasterPos2f(0.08,-0.77); //to provide position for rasterization**

**Write1("Mrs.VIMALA H S");**

**glRasterPos2f(-0.23,-0.87); //to provide position for rasterization**

**GLUquadricObj \* quadricObj; //Creates a quadratic object**

**quadricObj = gluNewQuadric(); //Sets a pointer to a new quadratic object**

**gluQuadricDrawStyle(quadricObj, GLU\_FILL);**

**glPushMatrix();**

**glColor3f (0.4,0.3,0.8); //Sets ellipse color**

**glTranslatef(-0.7,0.91,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.89,0.88); //to provide position for rasterization**

**Write1("TETRAHEDRON");**

**glutSwapBuffers();**

**glFlush();**

**}**

**int main(int argc,char \*\*argv) //main function**

**{**

**glutInit(&argc,argv); //initializes GLUT**

**glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB); //defines the display mode**

**glutInitWindowSize(1000,600); //choose the window size**

**glutInitWindowPosition(100,100); //establish the window position**

**glutCreateWindow("PROJECT MENU CHART"); //create the window**

**glutMouseFunc(OnMouseClick); //for mouse callback**

**glutDisplayFunc(display); //for display callback**

**glutMainLoop();**

**}**

1. **STROKE**

**a) Flowchart**

**START**

**DRAW THE WORKSPACE AREA**

**CHOOSE THE OPTIONS**

**1)ROMAN**

**2)MONO ROMAN**

**MESSAGE abcdefghi**

**ABCDEFGHI**

**READ THE DATA**

**DRAW THE WORKSPACE AREA**

**DRAW THE DATA**

**STOP**

**b) Source Code**

**#include <string.h>**

**#include <GL/glut.h>**

**#include<Windows.h>**

**void \*font = GLUT\_STROKE\_ROMAN;**

**void \*fonts[] =**

**{GLUT\_STROKE\_ROMAN, GLUT\_STROKE\_MONO\_ROMAN};**

**char defaultMessage[] = "GLUT means OpenGL.";**

**char \*message = defaultMessage;**

**int angle = 0;**

**void selectFont(int newfont)**

**{**

**font = fonts[newfont];**

**glutPostRedisplay();**

**}**

**void selectMessage(int msg)**

**{**

**switch (msg) {**

**case 1:**

**message = "abcdefghijklmnop";**

**break;**

**case 2:**

**message = "ABCDEFGHIJKLMNOP";**

**break;**

**}**

**}**

**void tick(void)**

**{**

**angle -= 2;**

**glutPostRedisplay();**

**}**

**void display11(void)**

**{**

**int len, i;**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glPushMatrix();**

**glRotatef(angle, 0.0, 0.0, 1.0);**

**glTranslatef(-750, 0, 0);**

**len = (int) strlen(message);**

**for (i = 0; i < len; i++) {**

**Sleep(20);**

**glutStrokeCharacter(font, message[i]);**

**}**

**glPopMatrix();**

**glutSwapBuffers();**

**}**

**/\*int main(int argc, char \*\*argv)**

**{**

**int i, submenu;**

**glutInit(&argc, argv);**

**for (i = 1; i < argc; i++) {**

**if (!strcmp(argv[i], "-mono")) {**

**font = GLUT\_STROKE\_MONO\_ROMAN;**

**}**

**}**

**glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**

**glutInitWindowSize(600, 600);**

**glutCreateWindow("anti-aliased stroke font");**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**gluOrtho2D(0, 2000, 0, 2000);**

**glMatrixMode(GL\_MODELVIEW);**

**glEnable(GL\_LINE\_SMOOTH);**

**glEnable(GL\_BLEND);**

**glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);**

**glLineWidth(3.0);**

**glTranslatef(1000, 1000, 0);**

**glClearColor(0.0, 0.0, 0.0, 1.0);**

**glColor3f(1.0, 1.0, 1.0);**

**glutDisplayFunc(display11);**

**glutIdleFunc(tick);**

**submenu = glutCreateMenu(selectMessage);**

**glutAddMenuEntry("abc", 1);**

**glutAddMenuEntry("ABC", 2);**

**glutCreateMenu(selectFont);**

**glutAddMenuEntry("Roman", 0);**

**glutAddMenuEntry("Mono Roman", 1);**

**glutAddSubMenu("Messages", submenu);**

**glutAttachMenu(GLUT\_RIGHT\_BUTTON);**

**glutMainLoop();**

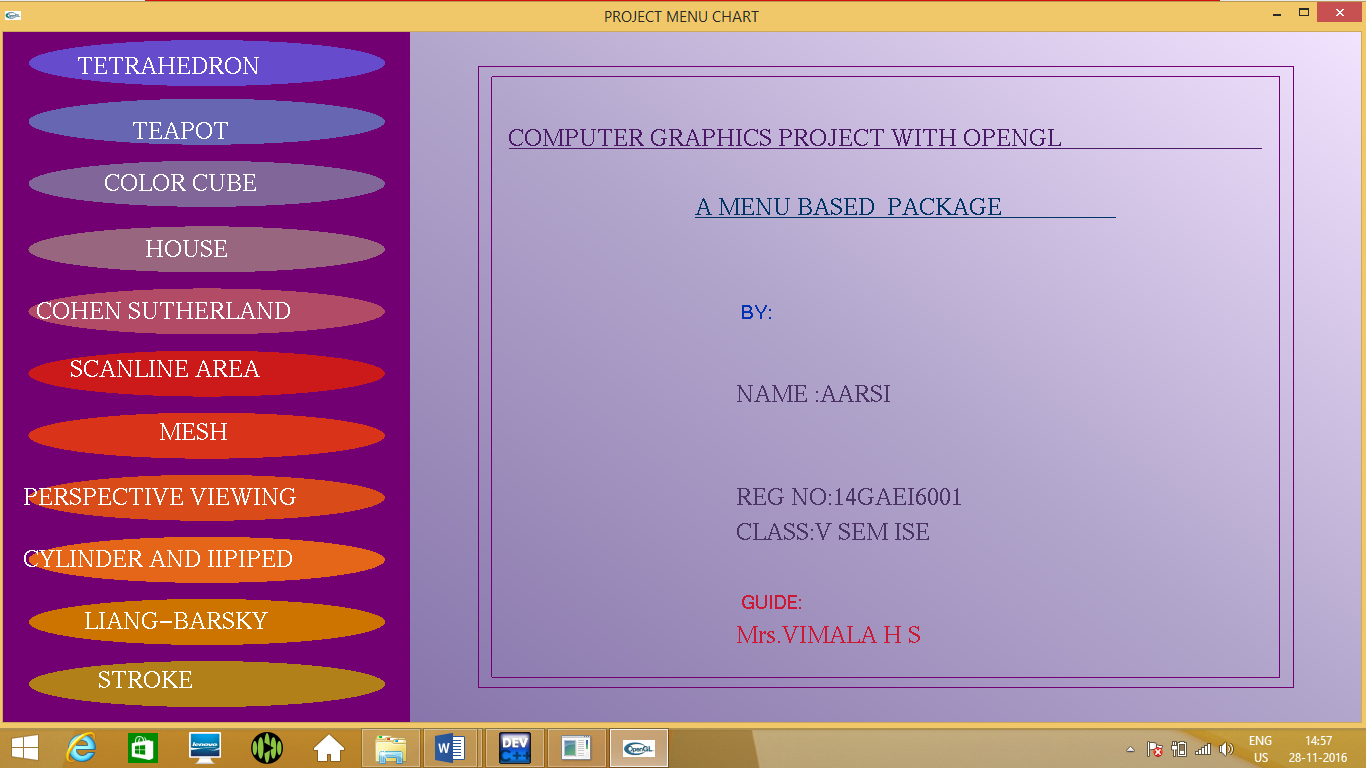
**return 0; /\* ANSI C requires main to return int. \*/**

**}**

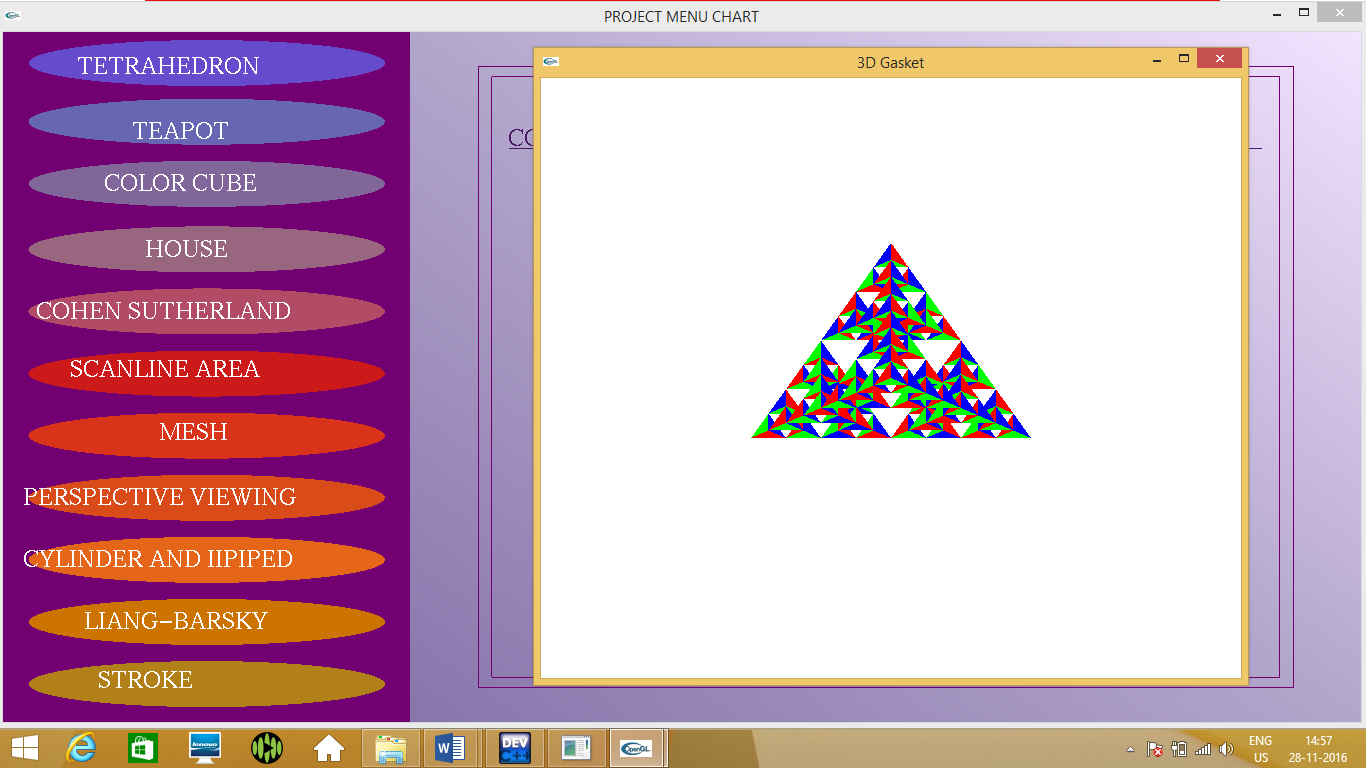
**\*/**

**9. SNAPSHOTS**

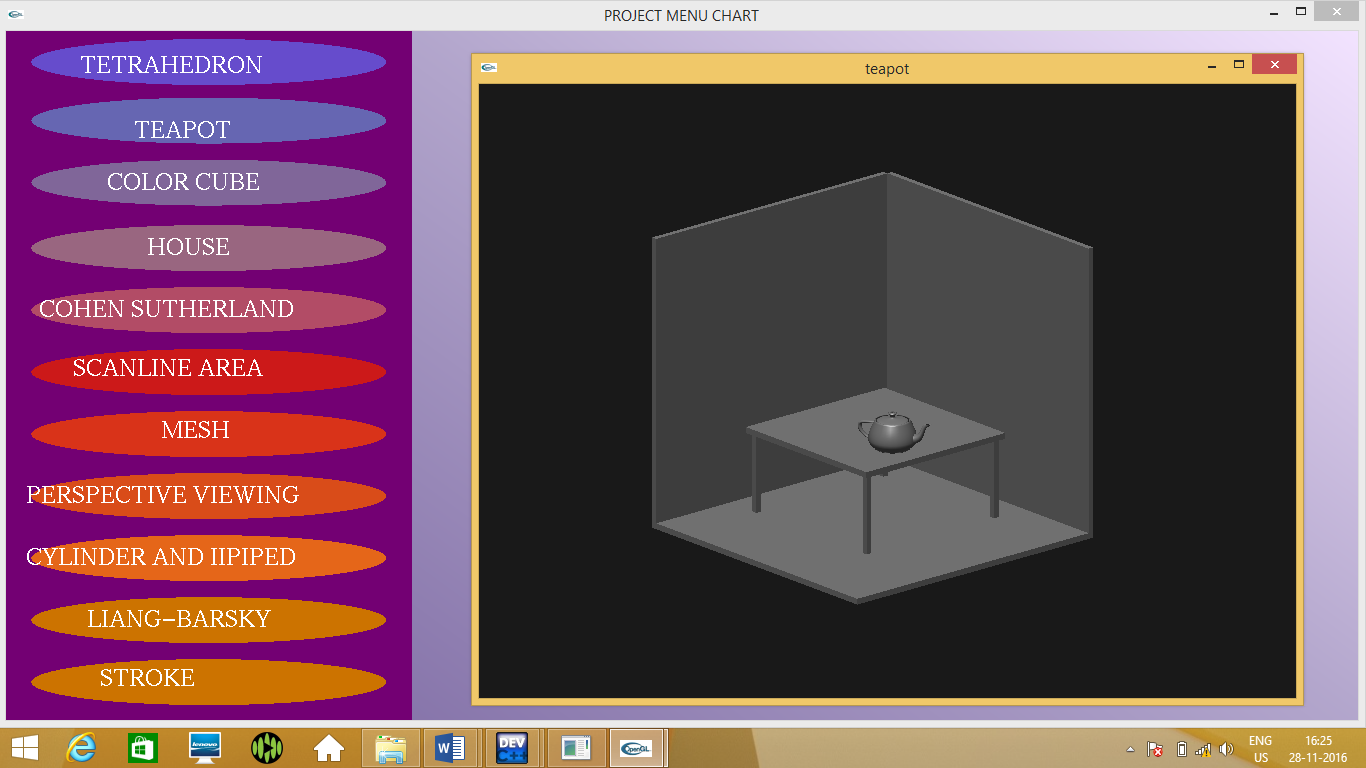
**Package Display**



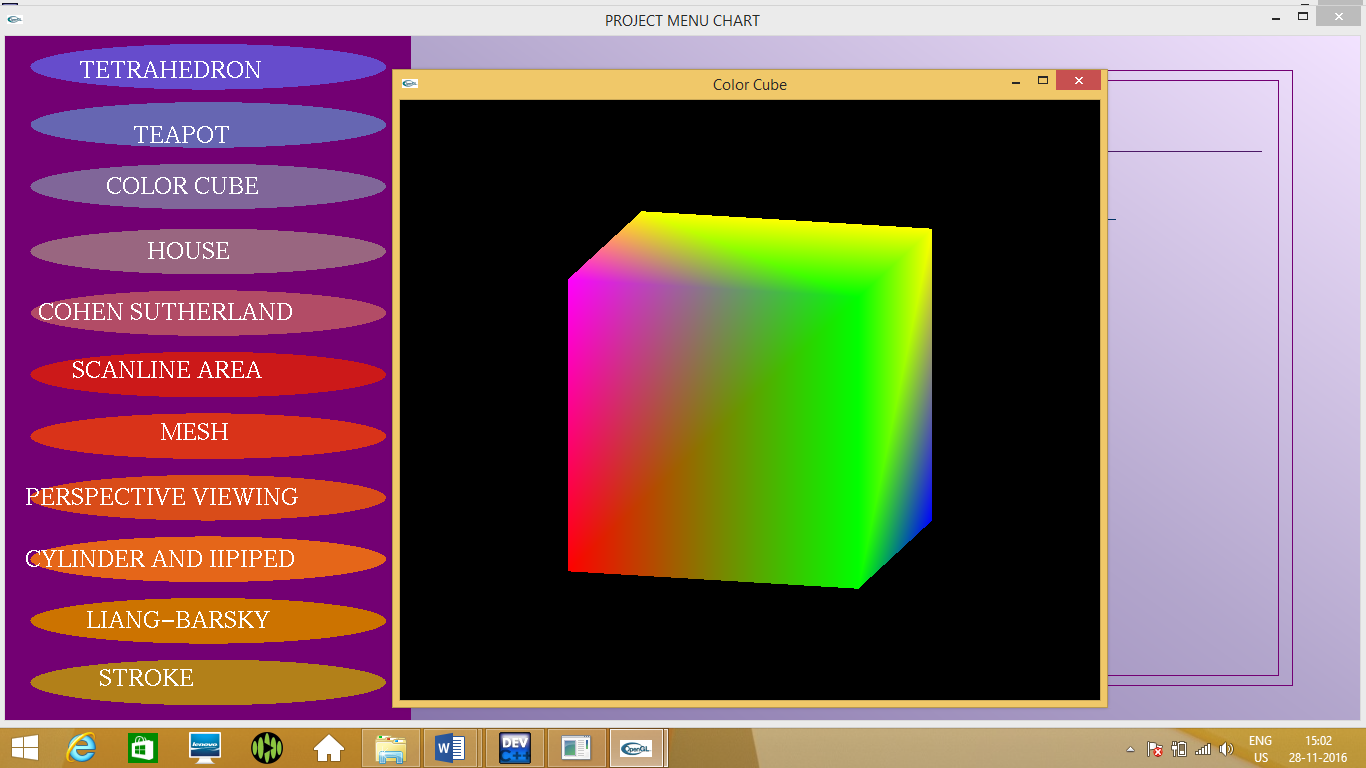
1. **TETRAHEDRON**



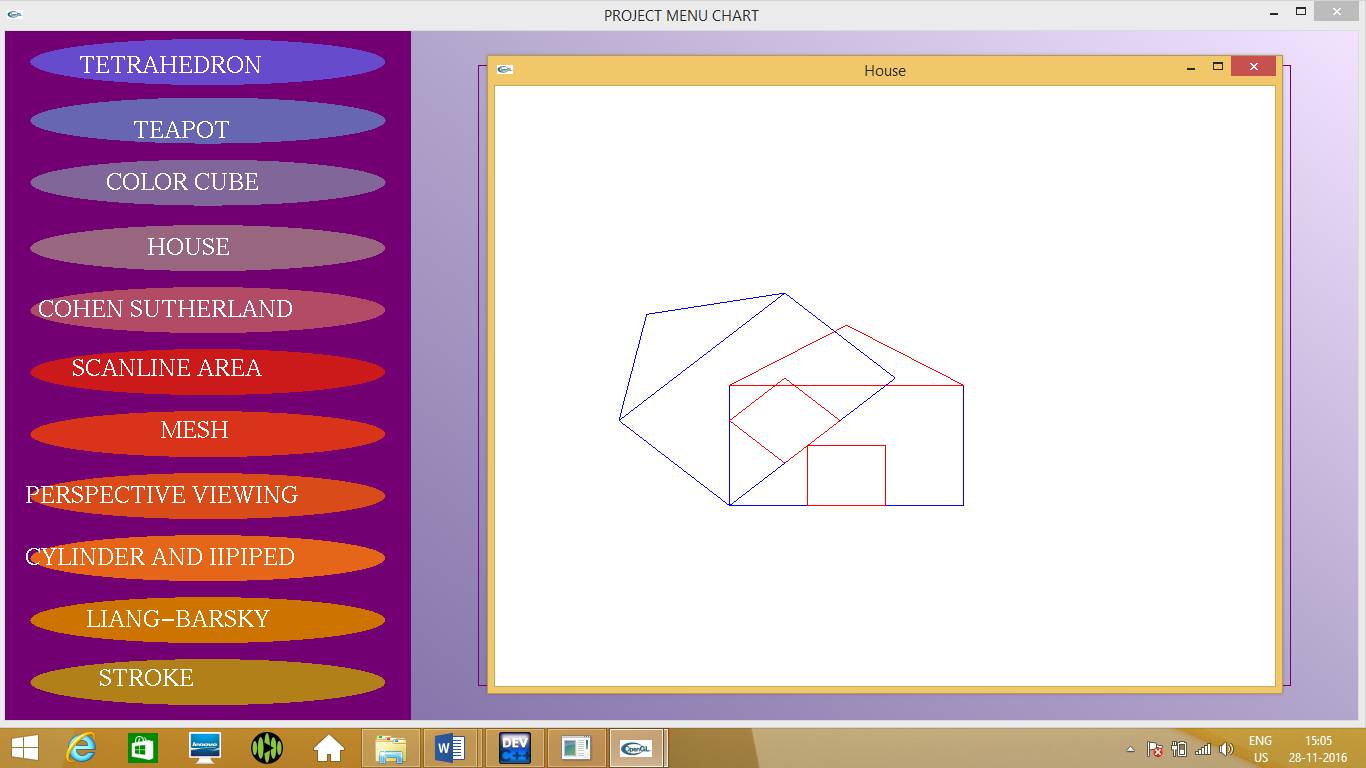
1. **TEA POT**



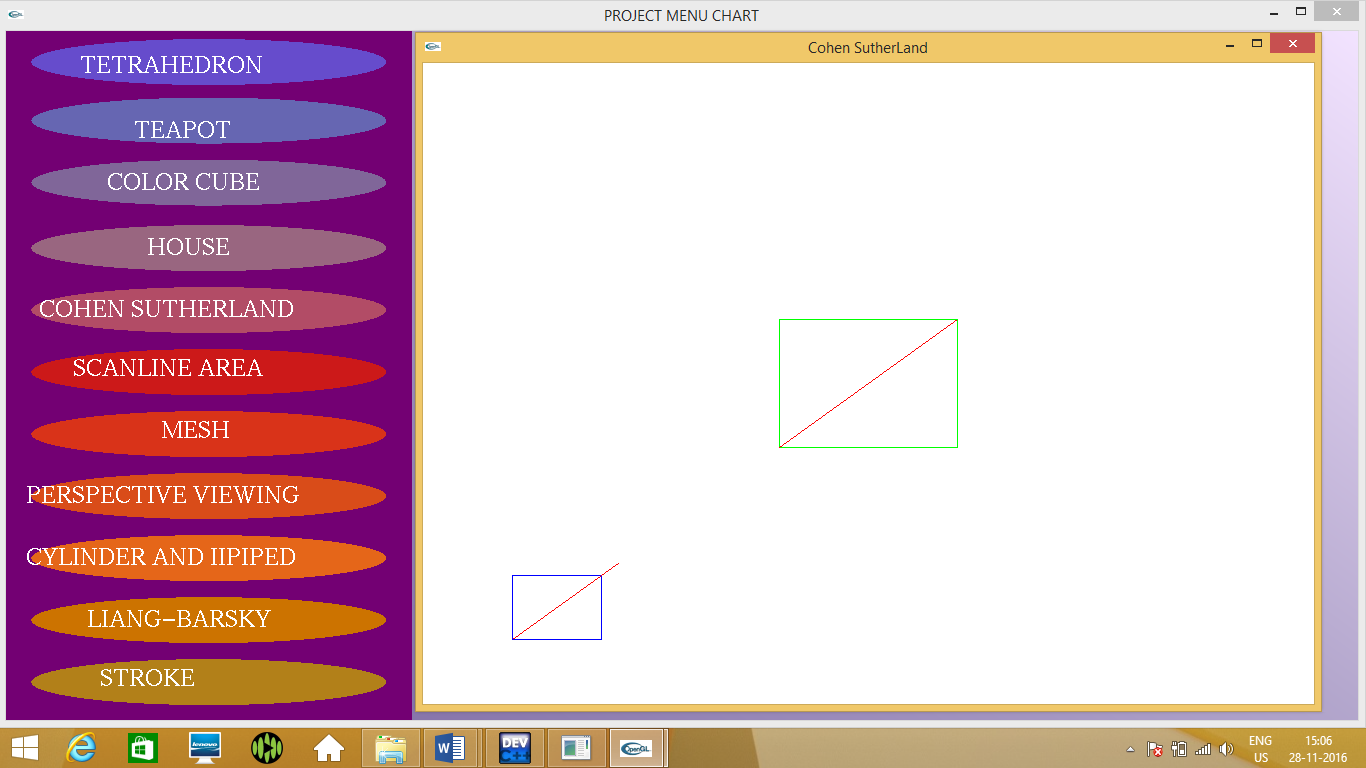
**3. COLOR CUBE**



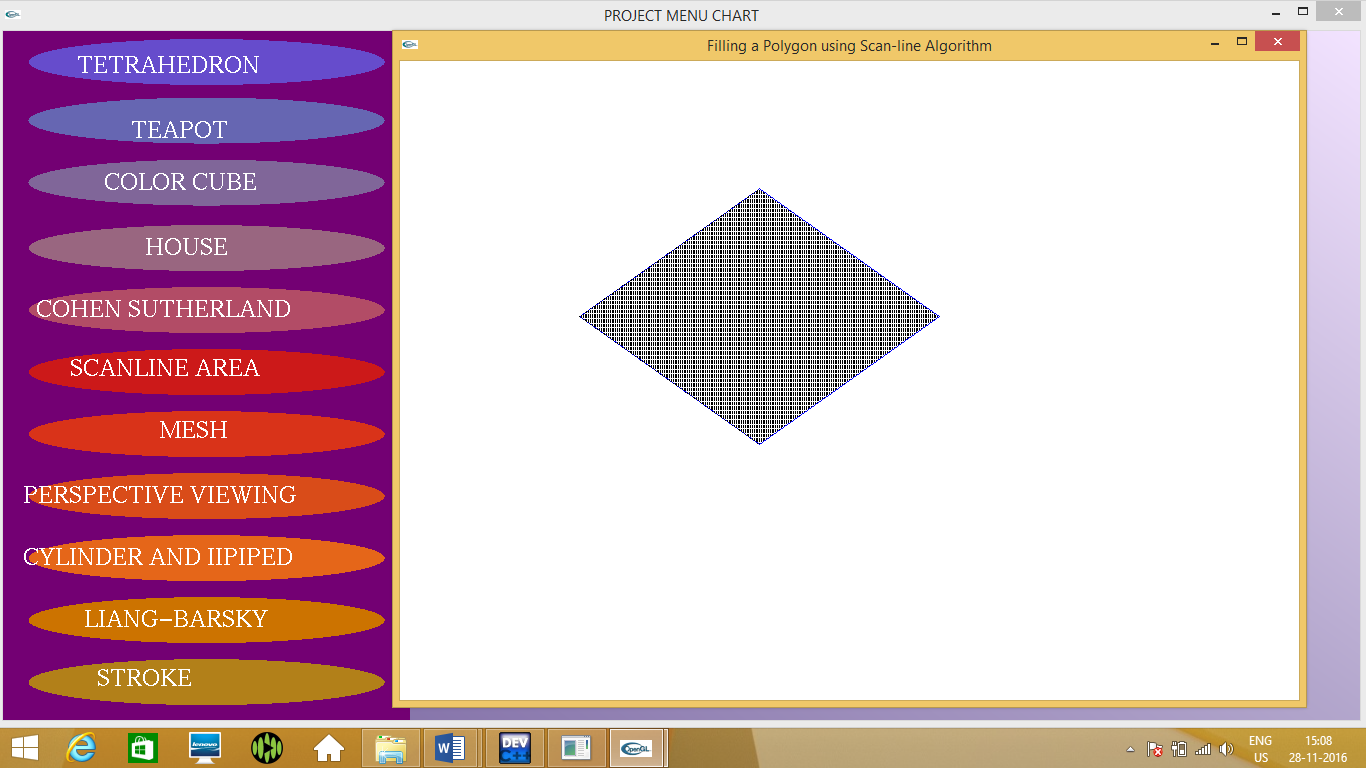
**4. HOUSE**



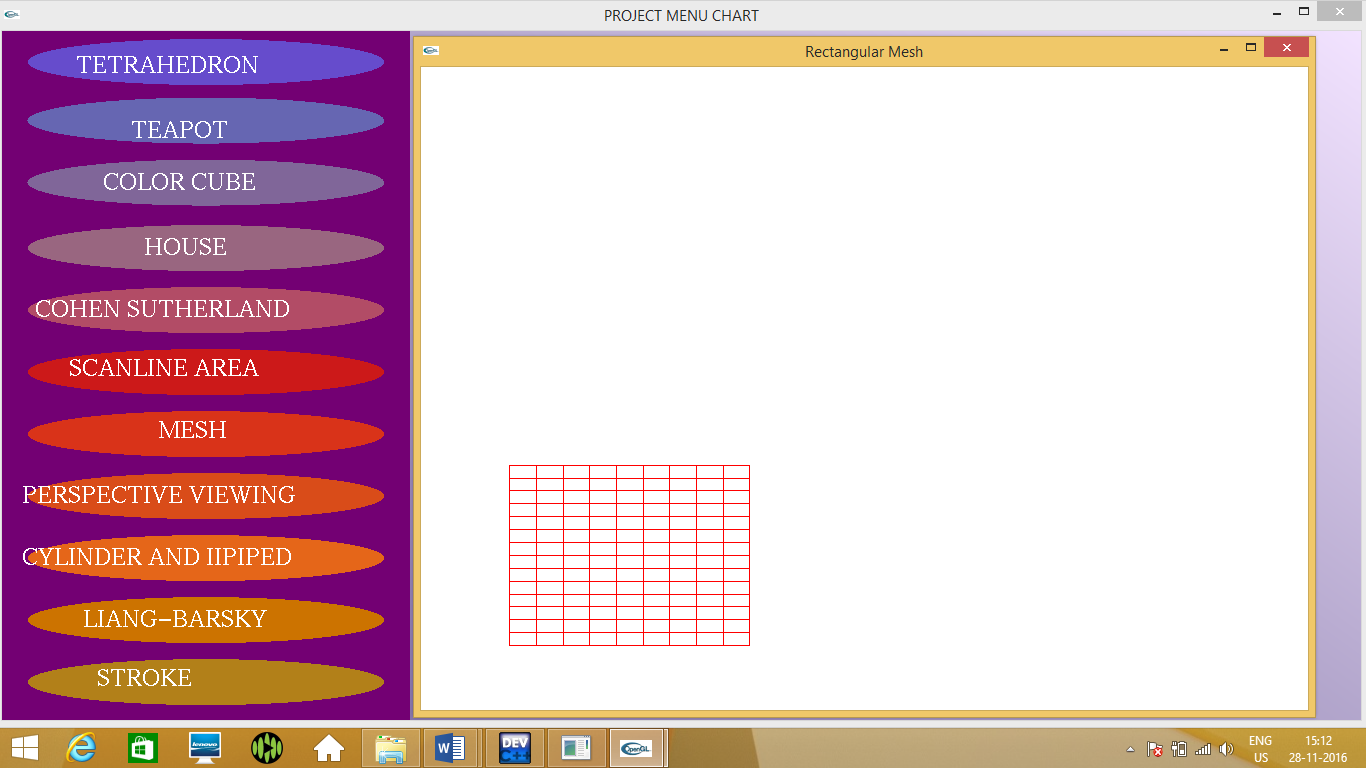
**5. COHEN SUTHERLAND**



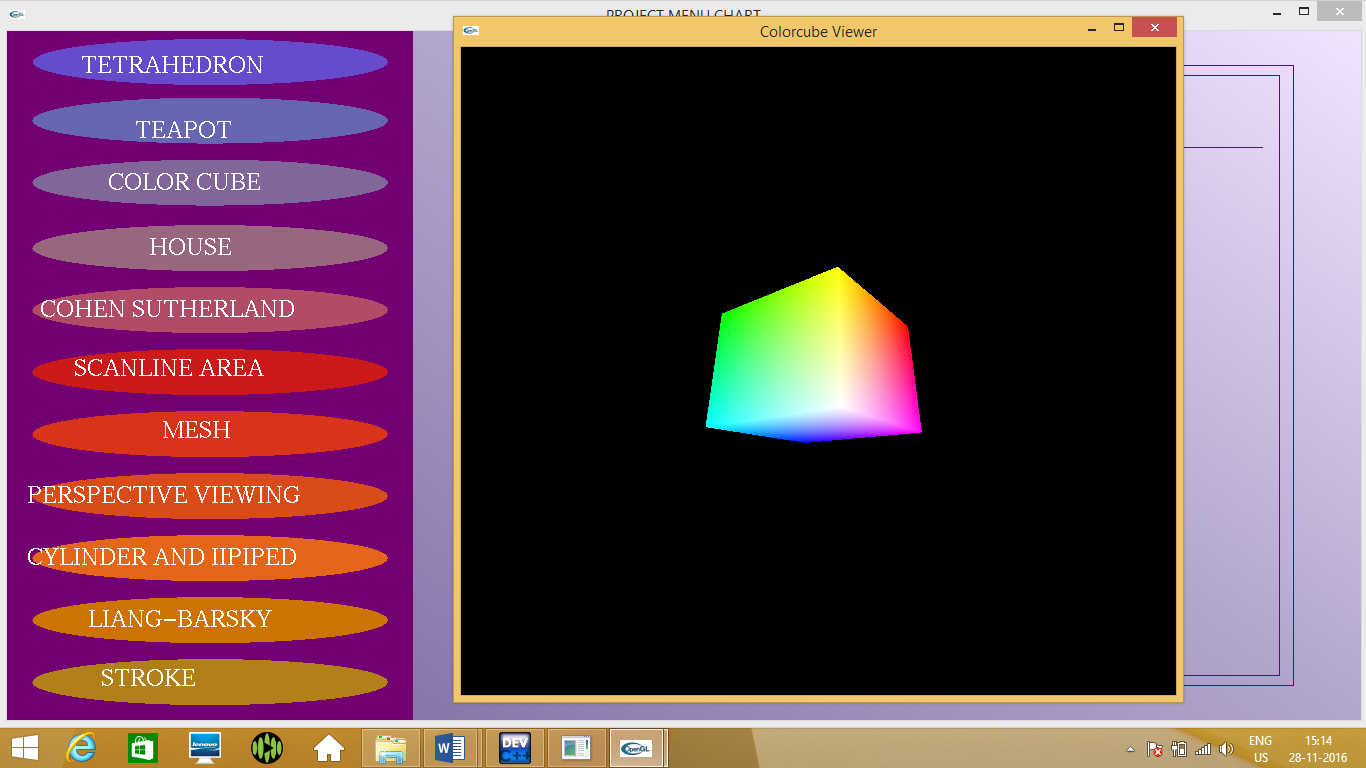
**6. SCANLINE AREA**



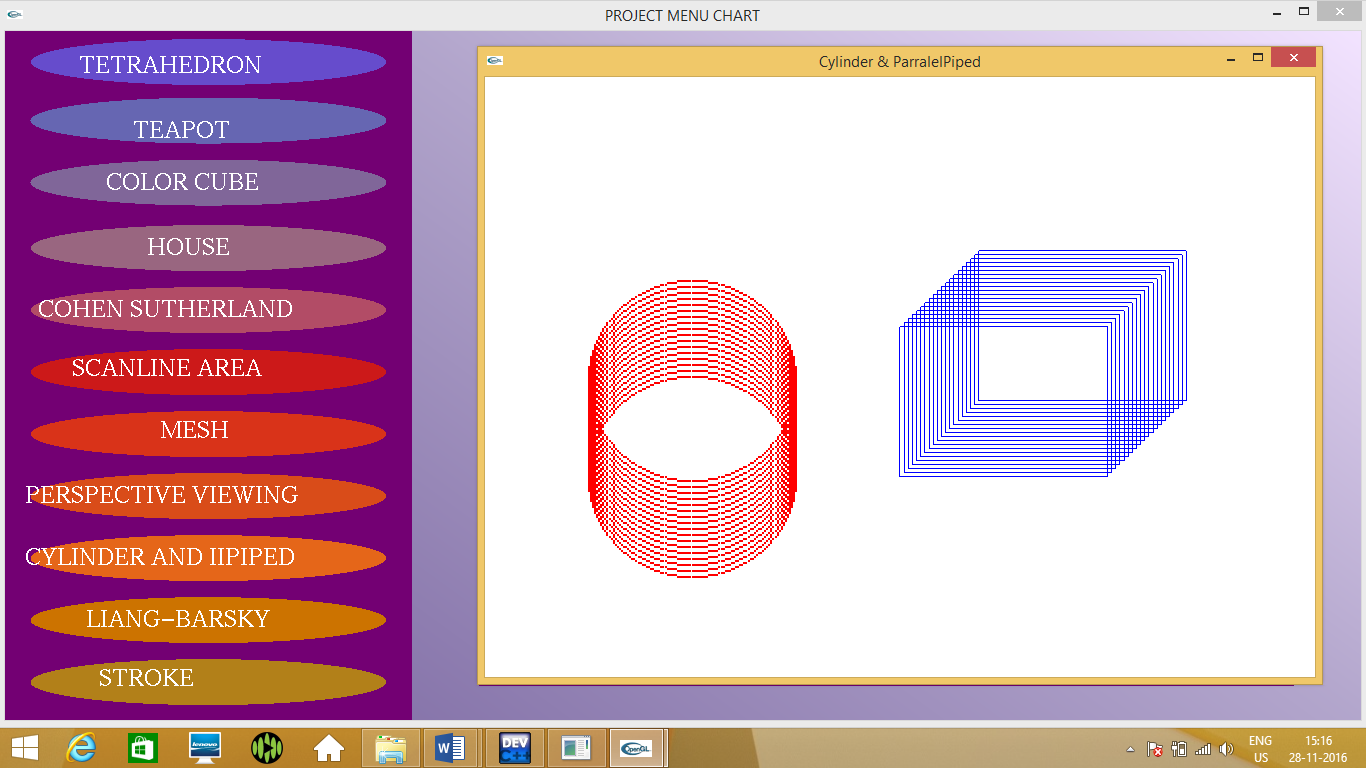
1. **MESH**



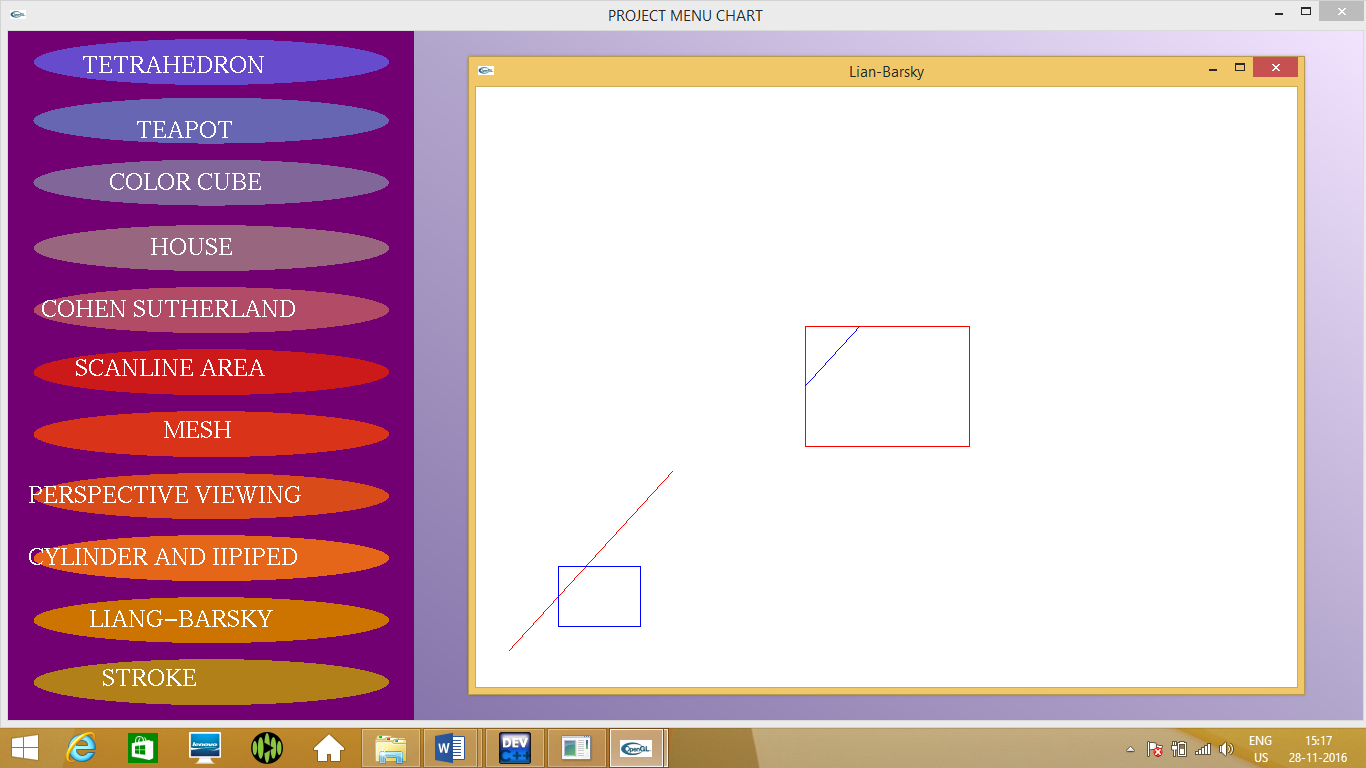
1. **CUBE-PERSPECTIVE VIEWING**



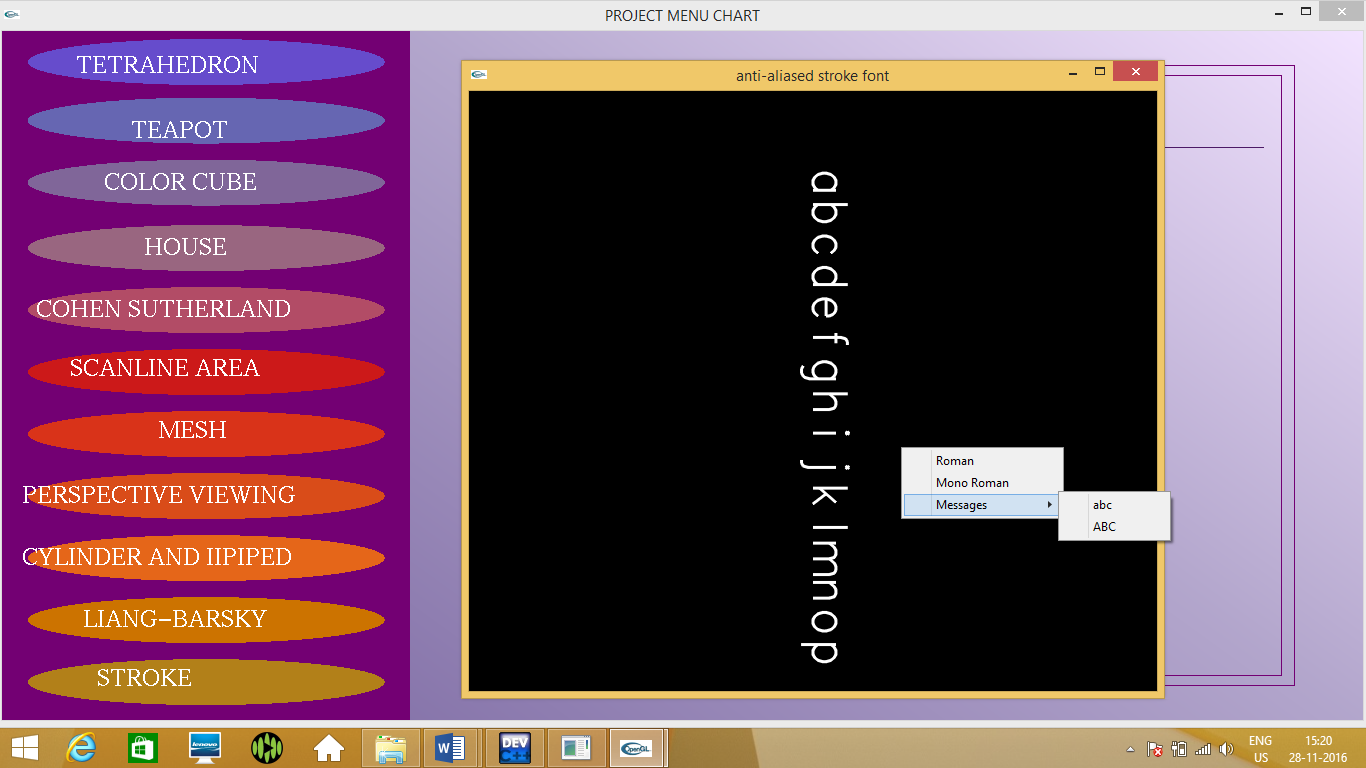
1. **CYLINDER AND IIPIPED**



**10. LIANG-BARSKY**



**11. STROKE**



1. **CONCLUSION**

The project entitled "GRAPHICS PACKAGE" is developed with the best of our effort. The project was developed using a modular approach.

The graphics creation and manipulation algorithm in the package have been implemented and tested to ensure the efficiency of operation and they were found to be quite satisfactory. The graphics editor has a good and intuitive and user friendly interface which enable the user to get better and easier interaction with the software.

After the completion of the development and study of the project we came to the conclusion that the computer graphics in C using OpenGL can be used to development a much better and complex soft wares that include 2D, 3D image processing and animation.

The program is also mainly concerned with animation and more of animation implementations. This type of implementations is now a days used in game development.

**11. REFERENCE**

* Interactive Computer graphics by Edward Angel
* Computer Graphics by Atul P Godse
* Opengl.org
* Stackoverflow.com
* Graphics Under C by Yashwanth Kanetkar

**12. APPENDIX**

**Glut functions**

**1 >>> glutDisplayFunc :-** glutDisplayFunc sets the display callback for the current window.

**Syntax :-**  void glutDisplayFunc(void(\*func)(void));

func:- The new display callback function.

**2 >>> glutReshapeFunc :-** glutReshapeFunc sets the reshape callback for the current window.

**Syntax :-** void glutReshapeFunc(void(\*func)(int width,int height));

func:- The new reshape callback function.

**3 >>> glutMouseFunc :-** glutMouseFunc sets the mouse callback for the current window.

**Syntax :-** void glutMouseFunc(void(\*func)(int button,int state,int x,int y));

func:- The new mouse callback function.

**4 >>> glutKeyboardFunc :-** glutKeyboardFunc sets the keyboard callback for the current window.

**Syntax :-** void glutKeyboardFunc(void(\*func)(unsigned char key,int x,int y));

func:- The new Keyboard callback function.

**5 >>> glutCreateWindow :-** glutCreateWindow creates a top-level window.

**Syntax :-** int glutCreateWindow(char \*name);

name:- ASCII character string for use as window name.

**6 >>> glutInit :-** glutInit is used to initalize the GLUT library.

**Syntax :-** void glutInit(int \*argc,char \*\*argv);

argc:- A pointer to the program's unmodified argc variable from main.Upon return,the value pointed to by argc will be updated because glutInit extracts any command line options intented for the GLUT library.

argv:- The program's unmodified argv variable from main.Like argc,the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library.

**7 >>> glutInitWindowPosition ,glutInitWindowSize :-** glutInitWindowPosition and glutInitWindowSize set the initial window position and size respectively.

**Syntax :-** void glutInitWindowSize(int width,int height);

void glutInitWindowPosition(int x,int y);

width: Width in pixels.

height: Height in pixels.

x: Window x location in pixels.

y: Window y location in pixels

**8 >>> glutInitDisplayMode :-** glutInitDisplayMode sets the initial display mode.

**Syntax :-**  void glutinitDisplayMode(unsigned int mode);

mode:- Display mode,normally the bitwise OR-ing of GLUT display mode bit masks.

**9 >>> glutMainLoop :-** glutMainLoop enters the GLUT event processing loop.

**Syntax :-** void glutMainLoop(void);

**10 >>> glutPostRedisplay :-** glutPostRedisplay marks the current window as rendering to be redisplayed.

**Syntax :-** void glutPostRedisplay(void);

**11 >>> glutBitmapCharacter :-** glutBitmapCharacter renders a bitmap character using OpenGL.

**Syntax :-** void glutBitmapCharacter(void \*font,int charcter);

**12 >>> glEnable :-**

**Syntax :-** void glEnable(GLenum cap);

cap:- specifies a symbolic constant indicating a GL capability.

**13 >>> glBegin and glEnd :-** The glBegin and glEnd functions delimit the vertices of a primitive or a group of like primitives.

**Syntax :-** void glBegin(GLenum mode);

mode:- The primitives that will be created from vertices presented between glBegin and the subsequent glEnd.

**14 >>> glClear :-** The glClear function clears buffers to present values.

**Syntax :-** void glClear(GLbitfield mask);

mask:- Bitwise OR operators of masks that indicate the buffers to be cleared.

**15 >>> glClearColor :-** The glClearColor unction specifies clear values for the color buffers.

**Syntax :-** void glClearColor(red,green,blue,aplha);

red:- The red value that glClear uses to clear the color buffers.The default value is zero;

green:- The green value that glClear uses to clear the color buffers.The default value is zero;

blue:- The blue value that glClear uses to clear the color buffers.The default value is zero;

alpha:- The alpha value that glClear uses to clear the color buffers.The default value is zero;

**16 >>> glColor3i :-** Sets the color.

**Syntax :-** void glColor3i(GLint red,GLint green,GLint blue);

red:- The new red value for the current value .

green:- The new green value for the current value .

blue:- The new blue value for the current value .

**17 >>> glColor3fv :-** Sets the current color from an already existing array of colors.

**Syntax :-** void glColor3fv(const GLfloat \*v);

v:- A pointer to an array that contains red,green,blue values.

**18 >>> glFlush :-** The glFlush function forces execution of OpenGL functions in finite time.

**Syntax :-** void glFlush(void);

This function has no parameters.

**19 >>> glLoadIdentity :-** The glLoadIdentity function replaces the current matrix with the identity matrix.

**20 >>> glOrtho :-** The glOrtho function multiplies the current matrix by an orthographic matrix.

**Syntax :-** void glOrtho(GLDouble left, GLDouble right, GLDouble bottom, GLDouble top, GLDouble zNear, GLDouble zFar);

left:- The coordinates of the left vertical clipping plane.

right:- The coordinates of the right vertical clipping plane.

bottom:- The coordinates of the bottom horizontal clipping plane.

top:- The coordinates of the top horizontal clipping plane.

zNear:- The distances to the nearer depth clipping plane.This distance is negative if the plane is to be behind the viewer.

zFar:- The distances to the farther depth clipping plane.This distance is negative if the plane is to be behind the viewer.

**21 >>> glPointSize :-** The glPointSize function specifies the diameter of rasterized points.

**Syntax :-** void glPointSize(GLfloat size);

Size:-The diameter of rasterized points.The default is 1.0.

**22 >>> glPushMatrix and glPopMatrix :-** The glPushMatrix and glPopMatrix functions push and pop the current matrix stack.

**Syntax :-** void WINAPI glPopMatrix(void);

**23 >>> glRotatef :-** The glRotatef function multiplies the current matrix by a rotation matrix.

**Syntax :-** void glRotatef(GLfloat angle,GLfloat x,GLfloat y,GLfloat z);

angle:-The angle of rotation.

x:-The x coordinate of vector.

y:-The y coordinate of vector.

z:-The z coordinate of vector.

**24 >>> glScalef :-** The glScaled and glScalef functions multiplies the current matrix by a general scaling matrix.

**Syntax :-** void glScalef(GLfloat x,GLfloat y,GLfloat z);

x:-Scale factors along x axis.

y:-Scale factors along y axis.

z:-Scale factors along z axis.

**25 >>> glTranslatef :-** The glTranslatef function multiplies the current matrix by a translation matrix.

**Syntax :-** void glTranslatef(GLfloat x,GLfloat y,GLfloat z);

x:- the x coordinate of a translation vector.

y:- the y coordinate of a translation vector.

z:- the z coordinate of a translation vector.

**26 >>> glVertex2d :-** Specifies a vector.

**Syntax :-** void glVertex2d(GLdouble x,GLdouble y);

x:- Specifies the x-coordinate of a vertex.

y:- Specifies the y-coordinate of a vertex.

**b) Complete Integration Program**

**#include<stdlib.h>**

**#include<GL/glut.h>**

**#include<stdio.h>**

**#include<math.h>**

**#include"1.c"**

**#include"2.c"**

**#include"3.c"**

**#include"4.c"**

**#include"5.c"**

**#include"6.c"**

**#include"teapot.c"**

**#include"8.c"**

**#include"9.c"**

**#include"10.c"**

**#include"stroke.c"**

**void Write1(char \*string) //to write the string passed as parameter**

**{**

**while(\*string)**

**glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, \*string++);**

**}**

**void Write2(char \*string) //to write the string passed as parameter**

**{**

**while(\*string)**

**glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, \*string++);**

**}**

**void OnMouseClick(int button, int state, int x, int y)**

**{**

**if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) //if left button of the mouse is pressed**

**{**

**if(x>25&&x<275&&y>15&&y<42) //if position of mouse lies between the first button coordinates**

**{**

**glBegin(GL\_POLYGON); //to draw the polygon using the following coordinates**

**glColor3f (0.9,0.9, 0.9); //set the color**

**glVertex2f(-0.45,0.85);**

**glVertex2f(-0.95,0.85);**

**glVertex2f(-0.95,0.95);**

**glVertex2f(-0.45,0.95);**

**glEnd(); //to end drawing the polygon**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB|GLUT\_DEPTH); //define the display mode**

**glutInitWindowSize(700,600); //choose the window size**

**glutInitWindowPosition(400,100); //establish the window position**

**glutCreateWindow("3D Gasket"); //create the window**

**glutDisplayFunc(display2); //for display callback**

**init();**

**glEnable(GL\_DEPTH\_TEST); //enables an openGL feature**

**glClearColor(1.0,1.0,1.0,1.0); //set the present RGBA clear color**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>69&&y<96) //if position of mouse lies between the second button coordinates**

**{**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB|GLUT\_DEPTH);**

**glutInitWindowSize(700,600);**

**glutInitWindowPosition(400,100);**

**glutCreateWindow("teapot");**

**glutDisplayFunc(displaysolid);**

**glEnable(GL\_LIGHTING);**

**glEnable(GL\_LIGHT0);**

**glShadeModel(GL\_SMOOTH);**

**glEnable(GL\_DEPTH\_TEST);**

**glEnable(GL\_NORMALIZE);**

**glClearColor(0.1,0.1,0.1,0.0);**

**glViewport(0,0,640,480);**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>123&&y<150) //if position of mouse lies between the third button coordinates**

**{ glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH); //define the display mode**

**glutInitWindowSize(700, 600); //choose the window size**

**glutInitWindowPosition(400,100); //establish the window position**

**glutCreateWindow("Color Cube"); //create the window**

**glutReshapeFunc(myReshape); //for myReshape callback**

**glutDisplayFunc(display4); //for display callback**

**glutIdleFunc(spinCube); //for spinCube callback**

**glutMouseFunc(mouse); //for mouse callback**

**glEnable(GL\_DEPTH\_TEST); //Enable hidden-surface removal**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>177&&y<204) //if position of mouse lies between the fourth button coordinates**

**{**

**theta1=theta1\*(3.14/180); //specifies the angle of rotation**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB); //define the display mode**

**glutInitWindowSize(700,600); //choose the window size**

**glutInitWindowPosition(400,100); //establish the window position**

**glutCreateWindow("House"); //create the window**

**glutDisplayFunc(display1); //for display callback**

**myinit();**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>231&&y<258) //if position of mouse lies between the fourth button coordinates**

**{**

**xz=50;yz=50;**

**xo=110;yo=110;**

**xmin=50;ymin=50;**

**xmax=100;ymax=100;**

**xvmin=200,yvmin=200;**

**xvmax=300;yvmax=300;**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB); //define the display mode**

**glutInitWindowSize(700,600); //choose the window size**

**glutInitWindowPosition(400,100); //establish the window position**

**glutCreateWindow("Cohen SutherLand"); //create the window**

**glutDisplayFunc(display5); //for display callback**

**myinit1();**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>285&&y<312) //if position of mouse lies between the fifth button coordinates**

**{**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB); //define the display mode**

**glutInitWindowSize(700,600); //choose the window size**

**glutInitWindowPosition(400,100); //establish the window position**

**glutCreateWindow("Filling a Polygon using Scan-line Algorithm"); //create the window**

**glutDisplayFunc(display6); //for display callback**

**myinit2();**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>339&&y<366) //if position of mouse lies between the sixth button coordinates**

**{**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB); //define the display mode**

**glutInitWindowSize(700,600); //choose the window size**

**glutInitWindowPosition(400,100); //establish the window position**

**glutCreateWindow("Rectangular Mesh"); //create the window**

**glutDisplayFunc(display7); //for display callback**

**init2();**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>393&&y<420) //if position of mouse lies between the sixth button coordinates**

**{**

**glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH); //define the display mode**

**glutInitWindowSize(700,600); //choose the window size**

**glutInitWindowPosition(400,100); //establish the window position**

**glutCreateWindow("Colorcube Viewer"); //create the window**

**glutReshapeFunc(myReshape8); //for myReshape callback**

**glutDisplayFunc(disp8); //for display callback**

**glutMouseFunc(mouse8); //for mouse callback**

**glutKeyboardFunc(keys8); //for keyboard callback**

**glEnable(GL\_DEPTH\_TEST); //enables an openGL feature**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>447&&y<474) //if position of mouse lies between the seventh button coordinates**

**{**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB); //define the display mode**

**glutInitWindowPosition(400,100); //establish the window position**

**glutInitWindowSize(700,600); //choose the window size**

**glutCreateWindow("Cylinder & ParralelPiped"); //create the window**

**init3();**

**glutDisplayFunc(display9); //for display callback**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>501&&y<528) //if position of mouse lies between the eighth button coordinates**

**{**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB); //define the display mode**

**glutInitWindowSize(700,600); //choose the window size**

**glutInitWindowPosition(400,100); //establish the window position**

**glutCreateWindow("Lian-Barsky"); //create the window**

**glutDisplayFunc(disp2); //for display callback**

**init5();**

**glutMainLoop();**

**}**

**if(x>25&&x<275&&y>550&&y<580)**

**{**

**int i, submenu;**

**/\* for (i = 1; i < argc; i++) {**

**if (!strcmp(argv[i], "-mono")) {**

**font = GLUT\_STROKE\_MONO\_ROMAN;**

**}**

**}\*/**

**glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**

**glutInitWindowSize(600, 600);**

**glutCreateWindow("anti-aliased stroke font");**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**gluOrtho2D(0, 2000, 0, 2000);**

**glMatrixMode(GL\_MODELVIEW);**

**glEnable(GL\_LINE\_SMOOTH);**

**glEnable(GL\_BLEND);**

**glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);**

**glLineWidth(3.0);**

**glTranslatef(1000, 1000, 0);**

**glClearColor(0.0, 0.0, 0.0, 1.0);**

**glColor3f(1.0, 1.0, 1.0);**

**glutDisplayFunc(display11);**

**glutIdleFunc(tick);**

**submenu = glutCreateMenu(selectMessage);**

**glutAddMenuEntry("abc", 1);**

**glutAddMenuEntry("ABC", 2);**

**glutCreateMenu(selectFont);**

**glutAddMenuEntry("Roman", 0);**

**glutAddMenuEntry("Mono Roman", 1);**

**glutAddSubMenu("Messages", submenu);**

**glutAttachMenu(GLUT\_RIGHT\_BUTTON);**

**glutMainLoop();**

**}**

**}**

**}**

**void display()**

**{**

**glClear(GL\_COLOR\_BUFFER\_BIT); //clears the color buffer**

**glColor3f (0.45,0.0,0.45); //set the color**

**glBegin(GL\_POLYGON); //to draw the polygon using the following coordinates**

**glVertex2f(-0.4,-1.0);**

**glVertex2f(-1.0,-1.0);**

**glVertex2f(-1.0,1.0);**

**glVertex2f(-0.4,1.0);**

**glEnd(); //to end drawing the polygon**

**glBegin(GL\_POLYGON); //to draw the polygon using the following coordinates**

**glColor3f (0.7,0.65,0.8); //set the color**

**glVertex2f(-0.4,1.0);**

**glColor3f(0.95,0.89,1); //set the color**

**glVertex2f(1.0,1.0);**

**glColor3f (0.7,0.65,0.8); //set the color**

**glVertex2f(1.0,-1.0);**

**glColor3f(0.53,0.46,0.67); //set the color**

**glVertex2f(-0.4,-1.0);**

**glEnd();**

**glColor3f (0.45,0.0,0.45);**

**glBegin(GL\_LINE\_LOOP); //to draw the border**

**glVertex2f(-0.3,0.9);**

**glVertex2f(0.9,0.9);**

**glVertex2f(0.9,-0.9);**

**glVertex2f(-0.3,-0.9);**

**glEnd();**

**glBegin(GL\_LINE\_LOOP); //to draw the border**

**glVertex2f(-0.28,0.87);**

**glVertex2f(0.88,0.87);**

**glVertex2f(0.88,-0.87);**

**glVertex2f(-0.28,-0.87);**

**glEnd();**

**glColor3f(0.3,0.1,0.4); //set the color**

**glRasterPos2f(-0.255,0.67); //to provide position for rasterization**

**Write1("COMPUTER GRAPHICS PROJECT WITH OPENGL");**

**glBegin(GL\_LINES); //to underline**

**glVertex2f(-0.255,0.663);**

**glVertex2f(0.855,0.663);**

**glEnd();**

**glColor3f(0,0.2,0.4); //set the color**

**glRasterPos2f(0.02,0.47); //to provide position for rasterization**

**Write1("A MENU BASED PACKAGE");**

**glBegin(GL\_LINES); //to underline**

**glVertex2f(0.02,0.463);**

**glVertex2f(0.64,0.463);**

**glEnd();**

**glColor3f(0,0.2,0.7); //set the color**

**glRasterPos2f(0.08,0.17); //to provide position for rasterization**

**Write2(" BY:");**

**glColor3f(0.3,0.2,0.4); //set the color**

**glRasterPos2f(0.08,-0.07); //to provide position for rasterization**

**Write1("NAME :AARSI");**

**glRasterPos2f(0.08,-0.37);**

**Write1("REG NO:14GAEI6001");**

**glRasterPos2f(0.08,-0.47);**

**Write1("CLASS:V SEM ISE");**

**glColor3f(0.8,0.1,0.2); //set the color**

**glRasterPos2f(0.08,-0.67); //to provide position for rasterization**

**Write2(" GUIDE:");**

**glRasterPos2f(0.08,-0.77); //to provide position for rasterization**

**Write1("Mrs.VIMALA H S");**

**glRasterPos2f(-0.23,-0.87); //to provide position for rasterization**

**GLUquadricObj \* quadricObj; //Creates a quadratic object**

**quadricObj = gluNewQuadric(); //Sets a pointer to a new quadratic object**

**gluQuadricDrawStyle(quadricObj, GLU\_FILL);**

**glPushMatrix();**

**glColor3f (0.4,0.3,0.8); //Sets ellipse color**

**glTranslatef(-0.7,0.91,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.89,0.88); //to provide position for rasterization**

**Write1("TETRAHEDRON");**

**glColor3f (0.4,0.4,0.7); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,0.74,0.0); //Translates the ellipse**

**glScalef(0.35,0.089,0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.81,0.69); //to provide position for rasterization**

**Write1("TEAPOT");**

**glColor3f (0.5,0.4,0.6); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,0.56,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.85,0.54); //to provide position for rasterization**

**Write1("COLOR CUBE");**

**glColor3f (0.6,0.4,0.5); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,0.37,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.79,0.35); //to provide position for rasterization**

**Write1("HOUSE");**

**glColor3f (0.7,0.3,0.4); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,0.19,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.95,0.17); //to provide position for rasterization**

**Write1("COHEN SUTHERLAND");**

**glColor3f (0.8,0.45,0.0); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,-0.71,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.88,-0.73); //to provide position for rasterization**

**Write1("LIANG-BARSKY");**

**glColor3f (0.9,0.4,0.1); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,-0.53,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Translates the ellipse**

**glRasterPos2f(-0.97,-0.55); //to provide position for rasterization**

**Write1("CYLINDER AND IIPIPED");**

**glColor3f (0.85,0.3,0.1); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,-0.35,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.97,-0.37); //to provide position for rasterization**

**Write1("PERSPECTIVE VIEWING");**

**glColor3f (0.85,0.2,0.1); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,-0.17,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.77,-0.18); //to provide position for rasterization**

**Write1("MESH");**

**glColor3f (0.8,0.1,0.1); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,0.01,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.90,0.0); //to provide position for rasterization**

**Write1("SCANLINE AREA");**

**glColor3f (0.8,0.45,0.0); //Sets ellipse color**

**glPushMatrix();**

**glTranslatef(-0.7,-0.89,0.0); //Translates the ellipse**

**glScalef(0.35, 0.089, 0.0); //Scales the ellipse**

**gluDisk(quadricObj, 0.0,0.75, 100, 100);**

**glPopMatrix();**

**glColor3f(1.0,1.0,1.0); //Sets ellipse color**

**glRasterPos2f(-0.86,-0.90); //to provide position for rasterization**

**Write1("STROKE");**

**glutSwapBuffers();**

**glFlush();**

**}**

**int main(int argc,char \*\*argv) //main function**

**{**

**glutInit(&argc,argv); //initializes GLUT**

**glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB); //defines the display mode**

**glutInitWindowSize(1000,600); //choose the window size**

**glutInitWindowPosition(100,100); //establish the window position**

**glutCreateWindow("PROJECT MENU CHART"); //create the window**

**glutMouseFunc(OnMouseClick); //for mouse callback**

**glutDisplayFunc(display); //for display callback**

**glutMainLoop();**

**}**