EXPERIMENT NO. 11

AIM: - To tudy and implement basic Open GL functions to draw basic primitives.

S/W REQUIRED: - VC++, Open GL.

THEORY:-

The first line of the main() routine initializes a *window* on the screen:
The InitializeAWindowPlease() routine is meant as a placeholder for window system-specific routines, which are generally not OpenGL calls. The next two lines are OpenGL commands that clear the window to black: glClearColor() establishes what color the window will be cleared to, and glClear() actually clears the window. Once the clearing color is set, the window is cleared to that color whenever glClear() is called. This clearing color can be changed with another call to glClearColor(). Similarly, theglColor3f() command establishes what color to use for drawing objects - in this case, the color is white. All objects drawn after this point use this color, until it's changed with another call to set the color.

The next OpenGL command used in the program, glOrtho(), specifies the coordinate system OpenGL assumes as it draws the final image and how the image gets mapped to the screen. The next calls, which are bracketed by glBegin() and glEnd(), define the object to be drawn - in this example, a polygon with four vertices. The polygon's "corners" are defined by the glVertex3f()commands. As you might be able to guess from the arguments, which are (*x, y, z*) coordinates, the polygon is a rectangle on the z=0 plane.

Finally, glFlush() ensures that the drawing commands are actually executed rather than stored in a *buffer* awaiting additional OpenGL commands.
The UpdateTheWindowAndCheckForEvents()placeholder routine manages the contents of the window and begins event processing.

OPENGL COMMAND SYNTAX:-

As you might have observed from the simple program in the previous section, OpenGL commands use the prefix **gl** and initial capital letters for each word making up the command name (recall**glClearColor()**, for example). Similarly, OpenGL defined constants begin with GL_, use all capital letters, and use underscores to separate words (like GL_COLOR_BUFFER_BIT)..

Include Files

For all OpenGL applications, you want to include the gl.h header file in every file. Almost all OpenGL applications use GLU, the aforementioned OpenGL Utility Library, which requires inclusion of the glu.h header file. So almost every OpenGL source file begins with

#include <GL/gl.h>

#include <GL/glu.h>

GLUT for managing window manager tasks, you should include

#include <GL/glut.h>

WINDOW MANAGEMENT:-Five routines perform tasks necessary to initialize a window.

- **glutInit**(int **argc*, char ***argv*) initializes GLUT and processes any command line arguments (for X, this would be options like -display and -geometry). **glutInit()** should be called before any other GLUT routine.
- **glutInitDisplayMode**(unsigned int *mode*) specifies whether to use an *RGBA* or color-index color model. You can also specify whether you want a single- or double-buffered window. (If you're working in color-index mode, you'll want to load certain colors into the color map; use **glutSetColor()** to do this.) Finally, you can use this routine to indicate that you want the window to have an associated depth, stencil, and/or accumulation buffer. For example, if you want a window with double buffering, the RGBA color model, and a depth buffer, you might call**glutInitDisplayMode**(*GLUT_DOUBLE* |

GLUT_RGB | *GLUT_DEPTH*).

- **glutInitWindowPosition**(int *x*, int *y*) specifies the screen location for the upper-left corner of your window.
- **glutInitWindowSize**(int *width*, int *size*) specifies the size, in pixels, of your window.
- int **glutCreateWindow**(char **string*) creates a window with an OpenGL context. It returns a unique identifier for the new window. Be warned: Until **glutMainLoop()** is called (see next section), the window is not yet displayed.

HANDLING INPUT EVENTS:-You can use these routines to register callback commands that are invoked when specified events occur.

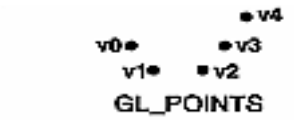
- **glutReshapeFunc**(void (**func*)(int *w*, int *h*)) indicates what action should be taken when the window is resized.
- **glutKeyboardFunc**(void (**func*)(unsigned char *key*, int *x*, int *y*)) and **glutMouseFunc**(void (**func*)(int *button*, int *state*, int *x*, int *y*)) allow you to link a keyboard key or a mouse button with a routine that's invoked when the key or mouse button is pressed or released.
- **glutMotionFunc**(void (**func*)(int *x*, int *y*)) registers a routine to call back when the mouse is moved while a mouse button is also pressed.

OPENGL GEOMETRIC DRAWING PRIMITIVES:-

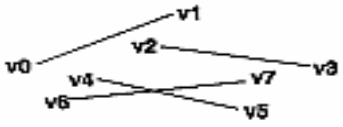
OpenGL to create a set of points, a line, or a polygon from those vertices. To do this, bracket each set of vertices between a call to **glBegin()** and a call to **glEnd()**. The argument passed to **glBegin()** determines what sort of geometric primitive is constructed from the vertices.

Value	Meaning
GL_POINTS	individual points
GL_LINES	individual line segments
GL_LINE_STRIP	series of connected line segments
GL_LINE_LOOP	Lines with a segment added between last and first vertices
GL_TRIANGLES	triples of vertices interpreted as triangles

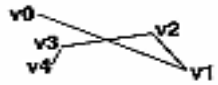
GL_TRIANGLE_STRIP	linked strip of triangles
GL_TRIANGLE_FAN	linked fan of triangles
GL_QUADS	vertices interpreted as four-sided polygons
GL_QUAD_STRIP	linked strip of quadrilaterals
GL_POLYGON	boundary of a simple, convex polygon



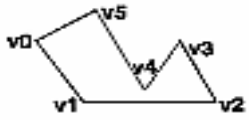
GL_POINTS



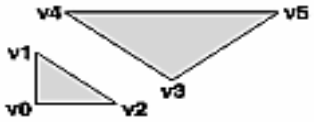
GL_LINES



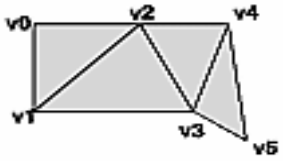
GL_LINE_STRIP



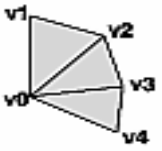
GL_LINE_LOOP



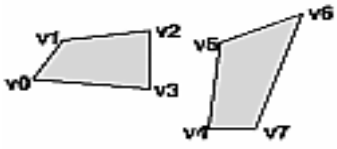
GL_TRIANGLES



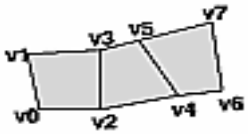
GL_TRIANGLE_STRIP



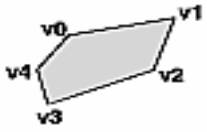
GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP



GL_POLYGON

CONCLUSION:- Successfully studied and implemented basic Open GL functions to draw basic primitives.

SIGN **GRADE**

DATE

Program:

```
/*
 * GL01Hello.cpp: Test OpenGL/GLUT C/C++ Setup
 * Tested under Eclipse CDT with MinGW/Cygwin and CodeBlocks with MinGW
 * To compile with -lfreeglut -lglu32 -lopengl32
 */
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/* Handler for window-repaint event. Call back when the window first appears and
whenever the window needs to be re-painted. */
void display() {
glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque
glClear(GL_COLOR_BUFFER_BIT);         // Clear the color buffer (background)

  // Draw a Red 1x1 Square centered at origin
glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
glColor3f(1.0f, 0.0f, 0.0f); // Red
glVertex2f(-0.5f, -0.5f);    // x, y
glVertex2f( 0.5f, -0.5f);
glVertex2f( 0.5f,  0.5f);
glVertex2f(-0.5f,  0.5f);
glEnd();

glFlush();  // Render now
}
```

```
/* Main function: GLUT runs as a console application starting at main()  */
int main(intargc, char** argv) {
glutInit(&argc, argv);                 // Initialize GLUT
glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title
glutInitWindowSize(320, 320);   // Set the window's initial width & height
glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
glutDisplayFunc(display); // Register display callback handler for window re-paint
glutMainLoop();           // Enter the event-processing loop
return 0;
}
```

Output:-