OpenGL:
- A high-performance, window-system independent, software interface to graphics hardware.

**Overview of an OpenGL program**
- **Main**
  - Open window and configure frame buffer (using GLUT for example)
  - Initialize GL states and display (Double buffer, color mode, etc.)
- **Loop**
  - Check for events
    - if window event (resize, unhide, maximize etc.)
      - modify the viewport
      - and Redraw
    - else if input event (keyboard and mouse etc.)
      - handle the event (such as move the camera or change the state)
      - and usually draw the scene
• Redraw
  o Clear the screen (and buffers e.g., z-buffer)
  o Change states (if desired)
  o Render
  o Swap buffers (if double buffer)

**OpenGL order of operations**
• Construct shapes (geometric descriptions of objects – vertices, edges, polygons etc.)
• Use OpenGL to
  o Arrange shape in 3D (using transformations)
  o Select your vantage point (and perhaps lights)
  o Calculate color and texture properties of each object
  o Convert shapes into pixels on screen

**OpenGL Syntax**
• All functions have the form: gl*
  o glVertex3f() – 3 means that this function take three arguments, and f means that the type of those arguments is float
  o glVertex2i() – 2 means that this function take two arguments, and i means that the type of those arguments is integer
• All variable types have the form: GL*
  o In OpenGL program it is better to use OpenGL variable types (portability)
    ▪ GLfloat instead of float
    ▪ Glint instead of int

**OpenGL primitives**
• Drawing two lines

```c
glBegin(GL_LINES);
  glVertex3f(_,_,_); // start point of line 1
  glVertex3f(_,_,_); // end point of line 1
  glVertex3f(_,_,_); // start point of line 2
  glVertex3f(_,_,_); // end point of line 2
glEnd();
```

We can replace GL_LINES with GL_POINTS, GL_LINELOOP, GL_POLYGON etc. (See OpenGL API for a complete lis).

**OpenGL states**
• On/off (e.g., depth buffer test)
  o glEnable( GLenum )
  o glDisable( GLenum )
  o Examples:
    ▪ glEnable(GL_DEPTH_TEST);
    ▪ glDisable(GL_LIGHTING);
• Mode States
  o Once the mode is set the effect stays until reset
Examples:
  - `glShadeModel(GL_FLAT)` or `glShadeModel(GL_SMOOTH)`
  - `glLightModel(...)` etc.

**Drawing in 3D**
- Depth buffer (or z-buffer) allows scene to remove hidden surfaces. Use `glEnable(GL_DEPTH_TEST)` to enable it.
- `glPolygonMode( Face, Mode )`
  - Face: `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK`
  - Mode: `GL_LINE`, `GL_POINT`, `GL_FILL`
- `glCullFace( Mode )`
  - Mode: `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK`
- `glFrontFace( Vertex_Ordering )`
  - Vertex Ordering: `GL_CW` or `GL_CCW`

**Viewing transformation**
- `glMatrixMode ( Mode )`
  - Mode: `GL_MODELVIEW`, `GL_PROJECTION`, or `GL_TEXTURE`
- `glLoadIdentity()`
- `glTranslate3f(x,y,z)`
- `glRotate3f(angle,x,y,z)`
- `glScale3f(x,y,z)`

**3D Projection (i.e., virtual camera)**
- Perspective
  - `glFrustum( left, right, bottom, top, znear, zfar );`
  - Also look at: `gluPerspective()`
- Orthographic
  - `glOrtho( left, right, bottom, top, zNear, zFar );`
Lighting

- Direction light source
- Position light source
- `glLightfv( Light#, Attribute, ... )`
  - `GLfloat position[] = {10, 10, 10, W}`
    - `glLightfv(GL_LIGHT0, GL_POSITION, position)`
      - If `(W)` is zero the position is treated as a direction (a 1x3 vector); otherwise, it is treated as a position (a 1x4 vector)
- `glEnable(GL_LIGTHING)`
- `glEnable(GL_LIGHT0)`

A program (Objects, Lights, Camera and ....)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <sys/types.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

int main(int argc, char **argv)
{
    /* initialize GLUT, OpenGL */
    glutInit(&argc, argv);
    
    /* set the window pos---let the windowing system determine */
    glutInitWindowPosition(-1,-1);
    
    /* set the window size */
    glutInitWindowSize(250,250);
    
    /* set the window display modes (hopefully supported)
       GLUT_DOUBLE: double-buffered
       GLUT_RGBA:   rgba colors (no colormap)
       GLUT_DEPTH:  z-buffering
    */
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
    
    /* create the window */
    glutCreateWindow("test");
    
    glClearColor(0.0,0.0,0.0,0.0);
    glShadeModel(GL_FLAT);
    glEnable(GL_DEPTH_TEST);
    glutDisplayFunc(do_redraw);
}
```
glutReshapeFunc(do_resize);
glutKeyboardFunc(keyboard);

/* let it go! */
glutMainLoop();
}

/* Glut callback function */
void do_resize(int w, int h)
{
    glViewport(0,0,(GLsizei)w,(GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0,(GLfloat)w/(GLfloat)h,1.0,30.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0,0.0,-3.6);
}

/* Actual scene drawing */
void do_redraw(void)
{
    /* clear back buffer */
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable( ... );
    glBegin(GL_POLYGON);
    glVertex3f(0,0,0);
    glVertex3f(10,0,0);
    glVertex3f(10,10,0);
    glEnd();
    glFlush();
    glDisable( ... );
    /* swap buffers */
    glutSwapBuffers();
}

/* handling input */
void keyboard(unsigned char key, int x, int y)
{
    switch(key) {
    case 'q':
    case 27:
        exit(0);
        break;
    default:
        break;
    }
}