AIM- Implementation of open GL

THEORY-

**OpenGL** ( ***Open G****raphics****L****ibrary* ) is a [standard set of functions](https://fr.wikipedia.org/wiki/Interface_de_programmation) for [calculation of images](https://fr.wikipedia.org/wiki/Rendu)[2D](https://fr.wikipedia.org/wiki/2D) or [3D](https://fr.wikipedia.org/wiki/3D) launched by [Silicon Graphics](https://fr.wikipedia.org/wiki/Silicon_Graphics) in 1992 [1](https://fr.wikipedia.org/wiki/OpenGL#cite_note-astle-1) . This [API](https://fr.wikipedia.org/wiki/Interface_de_programmation) is available on many platforms where it is used for applications ranging from video game to CAD via modeling [1](https://fr.wikipedia.org/wiki/OpenGL#cite_note-astle-1) .

OpenGL allows a program to report the geometry of objects as points, vectors, polygons, bitmaps and textures. OpenGL then performs calculations [projection](https://fr.wikipedia.org/wiki/G%C3%A9om%C3%A9trie_projective) to determine the image on the screen, taking into account the distance, orientation, shadows, transparency and framing [2](https://fr.wikipedia.org/wiki/OpenGL#cite_note-2),[3](https://fr.wikipedia.org/wiki/OpenGL#cite_note-carbera-3) .

The interface has about 250 functions different that can be used to display complex three-dimensional scenes from simple primitives. Because of its openness, its flexibility and its availability on all platforms, it is used by the majority of scientific, industrial or artistic 3D and some applications [Vector 2D](https://fr.wikipedia.org/wiki/Image_vectorielle) . This [library](https://fr.wikipedia.org/wiki/Biblioth%C3%A8que_logicielle) is also used in the industry [video game](https://fr.wikipedia.org/wiki/Jeu_vid%C3%A9o) where it is often in rivalry with the library [Microsoft](https://fr.wikipedia.org/wiki/Microsoft)  : [Direct3D](https://fr.wikipedia.org/wiki/Direct3D) . A version called [OpenGL ES](https://fr.wikipedia.org/wiki/OpenGL_ES) is designed specifically for embedded applications (mobile phones, pocket diary, game consoles ...).

Drawing 3D cube using specified vertices for its coordinate. In this example we define the six surfaces of the 3d cube. We use glBegin(GL\_QUADS) and glEnd(). Animation (Transformation 3D) and different color at each surface. - See more at: http://www.codemiles.com/c-opengl-examples/draw-3d-cube-using-opengl-t9018.html#sthash.8pmSd7Ct.dpuf

CUBE GLL:

#include <GL/glut.h>

GLfloat light\_diffuse[] = {1.0, 0.0, 0.0, 1.0}; /\* Red diffuse light. \*/

GLfloat light\_position[] = {1.0, 1.0, 1.0, 0.0}; /\* Infinite light location. \*/

GLfloat n[6][3] = { /\* Normals for the 6 faces of a cube. \*/

{-1.0, 0.0, 0.0}, {0.0, 1.0, 0.0}, {1.0, 0.0, 0.0},

{0.0, -1.0, 0.0}, {0.0, 0.0, 1.0}, {0.0, 0.0, -1.0} };

GLint faces[6][4] = { /\* Vertex indices for the 6 faces of a cube. \*/

{0, 1, 2, 3}, {3, 2, 6, 7}, {7, 6, 5, 4},

{4, 5, 1, 0}, {5, 6, 2, 1}, {7, 4, 0, 3} };

GLfloat v[8][3]; /\* Will be filled in with X,Y,Z vertexes. \*/

void

drawBox(void)

{

int i;

for (i = 0; i < 6; i++) {

glBegin(GL\_QUADS);

glNormal3fv(&n[i][0]);

glVertex3fv(&v[faces[i][0]][0]);

glVertex3fv(&v[faces[i][1]][0]);

glVertex3fv(&v[faces[i][2]][0]);

glVertex3fv(&v[faces[i][3]][0]);

glEnd();

}

}

void

display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

drawBox();

glutSwapBuffers();

}

void

init(void)

{

/\* Setup cube vertex data. \*/

v[0][0] = v[1][0] = v[2][0] = v[3][0] = -1;

v[4][0] = v[5][0] = v[6][0] = v[7][0] = 1;

v[0][1] = v[1][1] = v[4][1] = v[5][1] = -1;

v[2][1] = v[3][1] = v[6][1] = v[7][1] = 1;

v[0][2] = v[3][2] = v[4][2] = v[7][2] = 1;

v[1][2] = v[2][2] = v[5][2] = v[6][2] = -1;

/\* Enable a single OpenGL light. \*/

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

/\* Use depth buffering for hidden surface elimination. \*/

glEnable(GL\_DEPTH\_TEST);

/\* Setup the view of the cube. \*/

glMatrixMode(GL\_PROJECTION);

gluPerspective( /\* field of view in degree \*/ 40.0,

/\* aspect ratio \*/ 1.0,

/\* Z near \*/ 1.0, /\* Z far \*/ 10.0);

glMatrixMode(GL\_MODELVIEW);

gluLookAt(0.0, 0.0, 5.0, /\* eye is at (0,0,5) \*/

0.0, 0.0, 0.0, /\* center is at (0,0,0) \*/

0.0, 1.0, 0.); /\* up is in positive Y direction \*/

/\* Adjust cube position to be asthetic angle. \*/

glTranslatef(0.0, 0.0, -1.0);

glRotatef(60, 1.0, 0.0, 0.0);

glRotatef(-20, 0.0, 0.0, 1.0);

}

int

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

{

glutInit(&argc, argv);

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

glutCreateWindow("red 3D lighted cube");

glutDisplayFunc(display);

init();

glutMainLoop();

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

}

CONCLUSION-Thus we have implemented open GL to draw cube.



