**Lab Sheet No : 02**
**Index No : 19APP3936**
**Name : W.H.S.N.Rathnaweera**
**Date : 01.02.2025**

**01.**

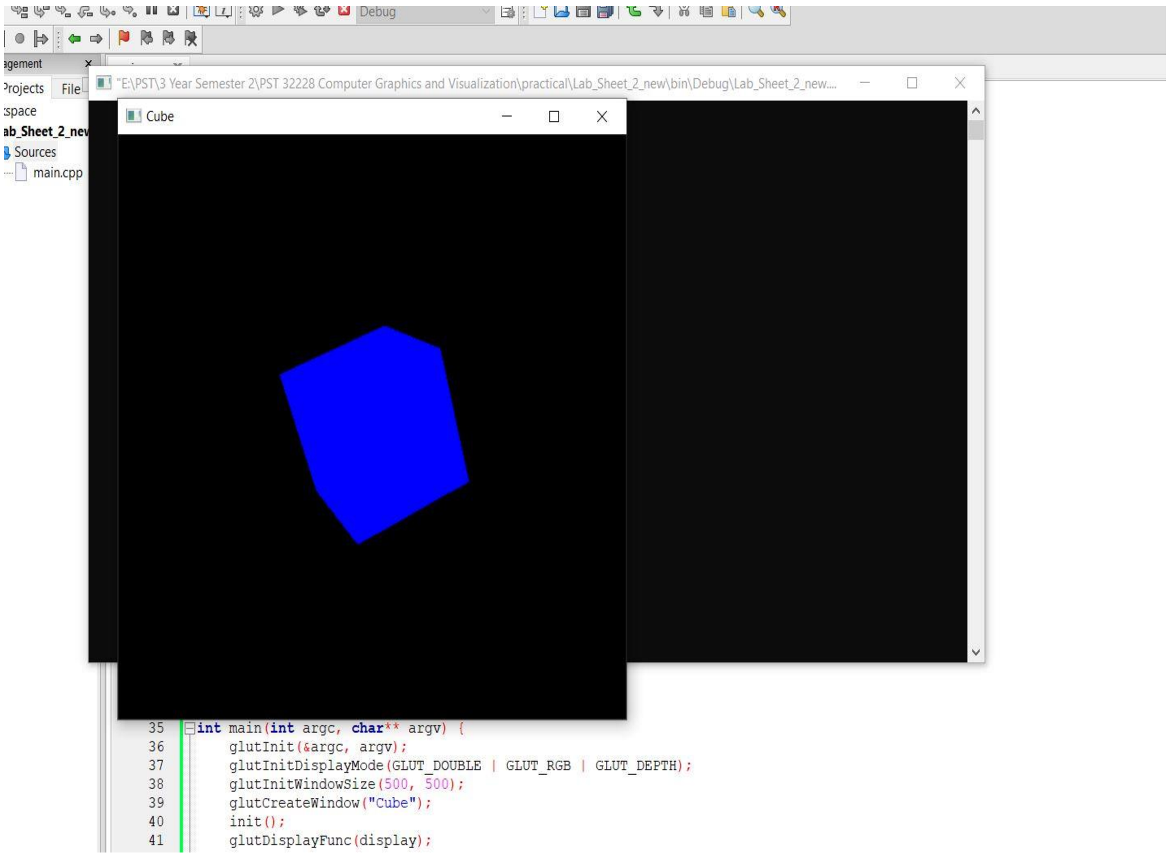
The Transformation Pipeline in computer graphics is a series of steps that convert 3D objects into 2D images for display. It starts with modeling transformations that position, rotate, and scale the objects in their local space. Next, view transformation adjusts the scene based on the camera's perspective. Projection transformation then maps the 3D scene onto a 2D plane using either perspective or orthographic projection. Finally, viewport transformation maps the 2D coordinates to the screen's pixel grid. In OpenGL, these transformations are managed using matrices and shaders, applying each step in sequence to convert 3D vertices into 2D screen space for rendering.

**02.**

**Cube:**

```
#ifdef __APPLE_CC__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
float angle = 0.0f;
void display() {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(45.0f, 1.0f, 1.0f, 200.0f);
   glMatrixMode(GL_MODELVIEW);
   glTranslatef(0.0f, 0.0f, -5.0f);
   glRotatef(angle, 1.0f, 1.0f, 1.0f);

   glColor3f(0.0f, 0.0f, 1.0f);

   glutSolidCube(1.0f);

   glutSwapBuffers();
}

void update(int value) {
   angle += 1.0f;
   if (angle > 360) angle -= 360;
```

```
    glutPostRedisplay();
    glutTimerFunc(16, update, 0);
}

void init() {
    glEnable(GL_DEPTH_TEST);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Cube");
    init();
    glutDisplayFunc(display);
    glutTimerFunc(0, update, 0);
    glutMainLoop();
    return 0;
}
```

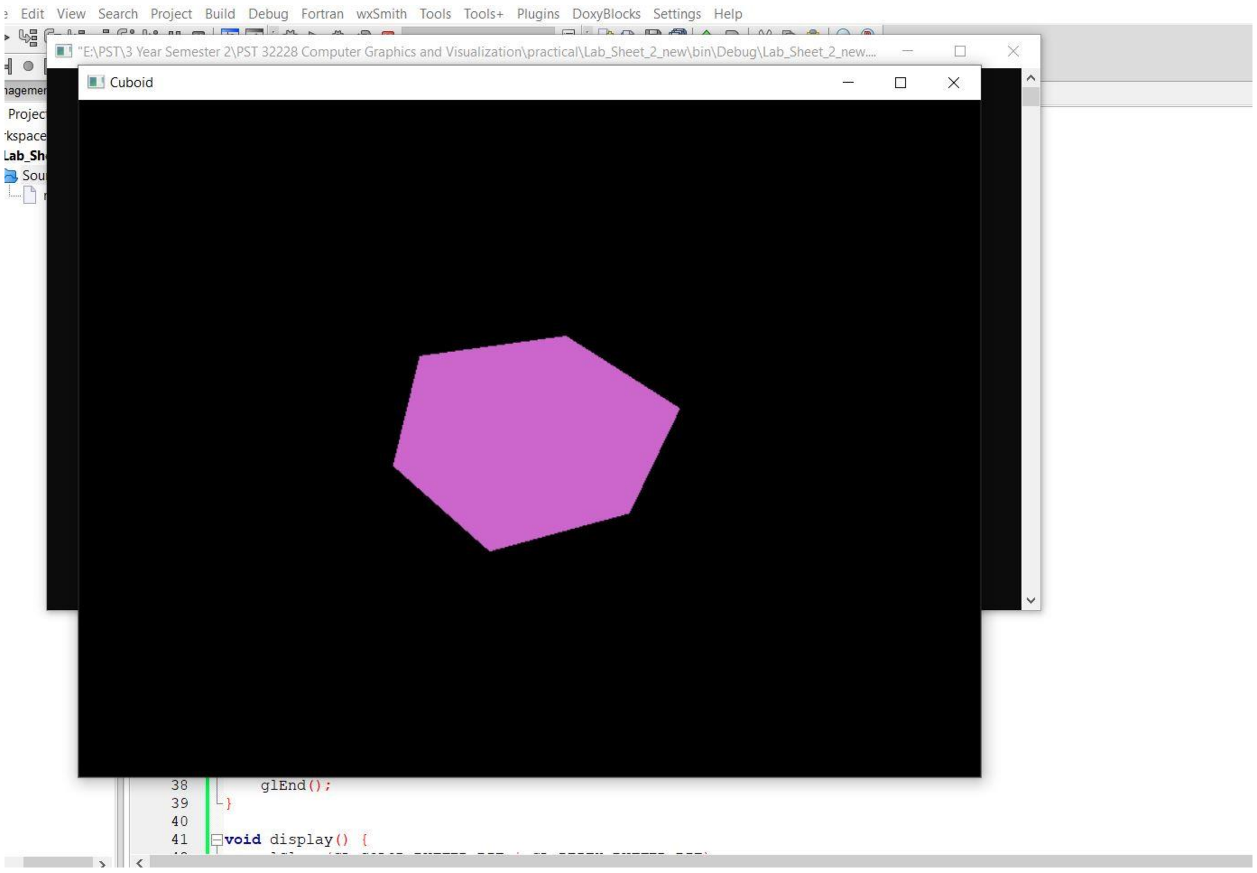


"E:\PST\3 Year Semester 2\PST 32228 Computer Graphics and Visualization\practical\Lab_Sheet_2_new\bin\Debug\Lab_Sheet_2_new....

```
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Cube");
    init();
    glutDisplayFunc(display);
```

**Cuboid:**

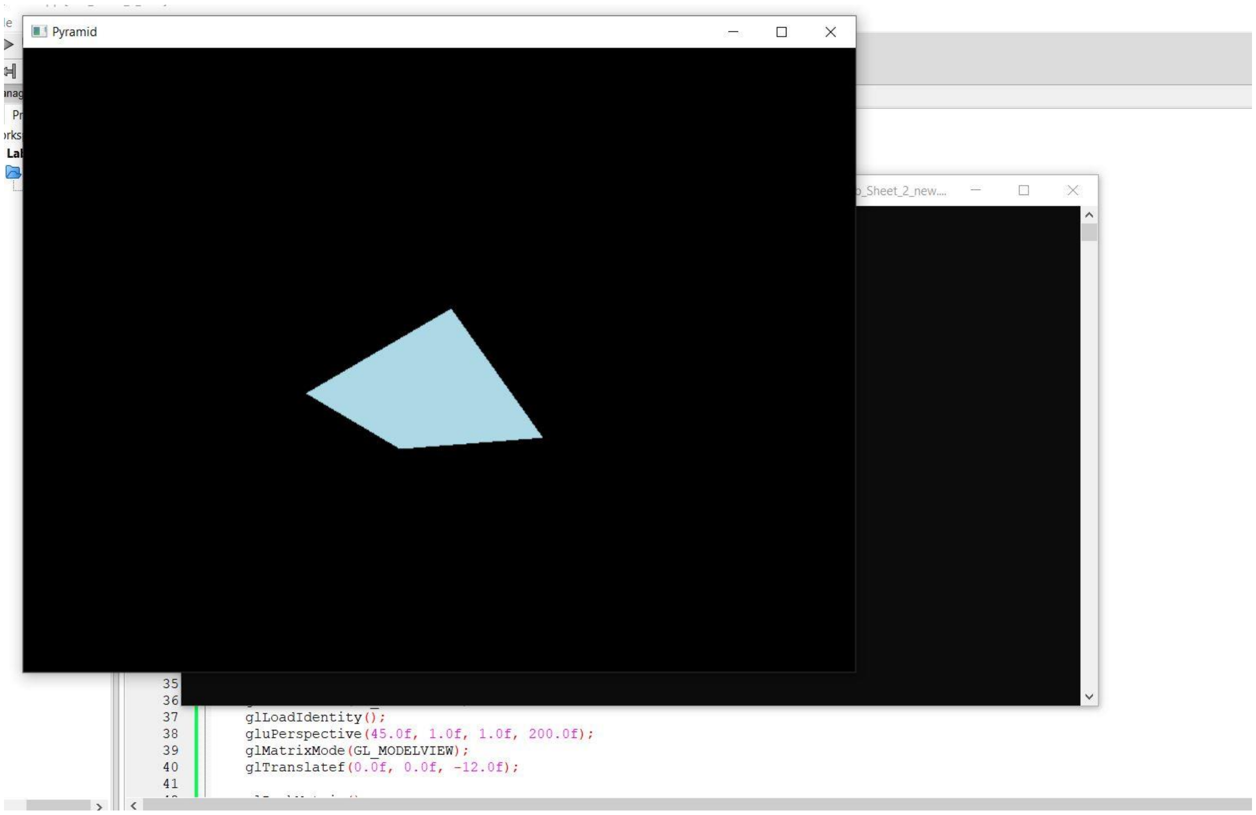
```
#ifdef __APPLE_CC__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

float angle = 0.0f;

void drawCuboid() {

    glColor3f(0.8f, 0.4f, 0.8f);

    glBegin(GL_QUADS);
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f, 1.0f, 1.0f);
    glVertex3f(-1.0f, 1.0f, 1.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);
    glVertex3f(-1.0f, 1.0f, -1.0f);
    glVertex3f( 1.0f, 1.0f, -1.0f);
    glVertex3f( 1.0f, -1.0f, -1.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glVertex3f(-1.0f, 1.0f, 1.0f);
    glVertex3f(-1.0f, 1.0f, -1.0f);
    glVertex3f( 1.0f, -1.0f, -1.0f);
    glVertex3f( 1.0f, 1.0f, -1.0f);
    glVertex3f( 1.0f, 1.0f, 1.0f);
    glVertex3f( 1.0f, -1.0f, 1.0f);
    glVertex3f(-1.0f, 1.0f, -1.0f);
    glVertex3f(-1.0f, 1.0f, 1.0f);
    glVertex3f( 1.0f, 1.0f, 1.0f);
    glVertex3f( 1.0f, 1.0f, -1.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);
    glVertex3f( 1.0f, -1.0f, -1.0f);
    glVertex3f( 1.0f, -1.0f, 1.0f);
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glEnd();
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0f, 1.0f, 1.0f, 200.0f);
    glMatrixMode(GL_MODELVIEW);
    glTranslatef(0.0f, 0.0f, -12.0f);

    glPushMatrix();
    glRotatef(angle, 1.0f, 1.0f, 0.0f);
    drawCuboid();
    glPopMatrix();

    glutSwapBuffers();
}

void update(int value) {
    angle += 1.0f;
    if (angle > 360) angle -= 360;
    glutPostRedisplay();
    glutTimerFunc(16, update, 0);
}

void init() {
    glEnable(GL_DEPTH_TEST);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Cuboid");
    init();
    glutDisplayFunc(display);
    glutTimerFunc(0, update, 0);
    glutMainLoop();
    return 0;
}
```


Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"E:\PST\3 Year Semester 2\PST 32228 Computer Graphics and Visualization\practical\Lab_Sheet_2_new\bin\Debug\Lab_Sheet_2_new....
Cuboid

```
    glEnd();
}

void display() {
```

**Pyramid:**

```
#ifdef __APPLE_CC__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

float angle = 0.0f;

void drawPyramid() {

    glColor3f(0.6784f, 0.8471f, 0.9020f);

    glBegin(GL_TRIANGLES);

    glVertex3f( 0.0f, 1.0f, 0.0f);
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f, -1.0f, 1.0f);

    glVertex3f( 0.0f, 1.0f, 0.0f);
    glVertex3f( 1.0f, -1.0f, 1.0f);
    glVertex3f( 1.0f, -1.0f, -1.0f);

    glVertex3f( 0.0f, 1.0f, 0.0f);
    glVertex3f( 1.0f, -1.0f, -1.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);

    glVertex3f( 0.0f, 1.0f, 0.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glEnd();
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0f, 1.0f, 1.0f, 200.0f);
    glMatrixMode(GL_MODELVIEW);
    glTranslatef(0.0f, 0.0f, -12.0f);

    glPushMatrix();
    glRotatef(angle, 1.0f, 1.0f, 0.0f);
    drawPyramid();
    glPopMatrix();

    glutSwapBuffers();
}

void update(int value) {
    angle += 1.0f;
    if (angle > 360) angle -= 360;
    glutPostRedisplay();
    glutTimerFunc(16, update, 0);
}

void init() {
    glEnable(GL_DEPTH_TEST);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Pyramid");
    init();
    glutDisplayFunc(display);
    glutTimerFunc(0, update, 0);
    glutMainLoop();
    return 0;
}
```

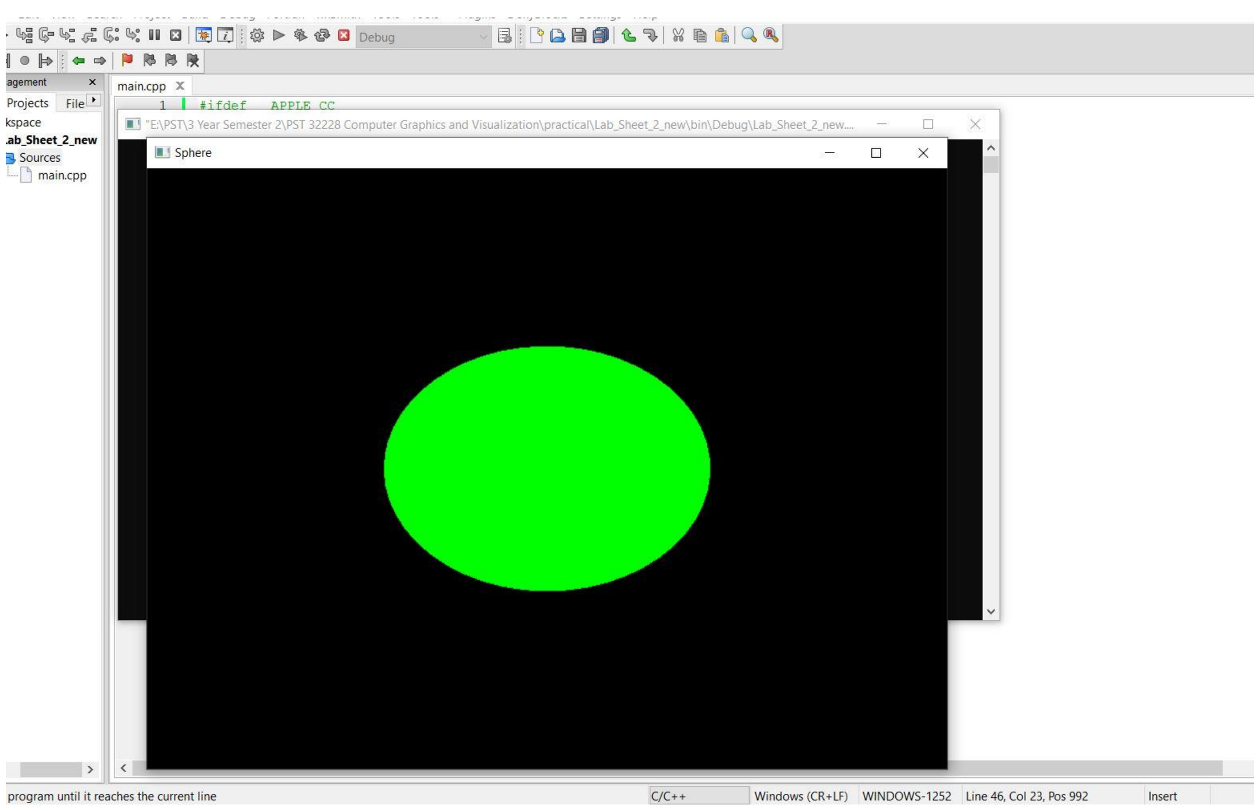


```
    glLoadIdentity();
    gluPerspective(45.0f, 1.0f, 1.0f, 200.0f);
    glMatrixMode(GL_MODELVIEW);
    glTranslatef(0.0f, 0.0f, -12.0f);
```

**Sphere:**

```
#ifdef __APPLE_CC__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

float angle = 0.0f;

void drawSphere() {
    glColor3f(0.0f, 1.0f, 0.0f);
    glutSolidSphere(2.0, 50, 50);
}
```

```
void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0f, 1.0f, 1.0f, 200.0f);
    glMatrixMode(GL_MODELVIEW);
    glTranslatef(0.0f, 0.0f, -12.0f);

    glPushMatrix();
    glRotatef(angle, 1.0f, 1.0f, 0.0f);
    drawSphere();
    glPopMatrix();

    glutSwapBuffers();
}

void update(int value) {
    angle += 1.0f;
    if (angle > 360) angle -= 360;
    glutPostRedisplay();
    glutTimerFunc(16, update, 0);
}

void init() {
    glEnable(GL_DEPTH_TEST);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Sphere");
    init();
    glutDisplayFunc(display);
    glutTimerFunc(0, update, 0);
    glutMainLoop();
    return 0;
}
```



```
#ifdef  APPLE CC
```

"E:\PST\3 Year Semester 2\PST 32228 Computer Graphics and Visualization\practical\Lab_Sheet_2_new\bin\Debug\Lab_Sheet_2_new...

program until it reaches the current line | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 46, Col 23, Pos 992 | Insert