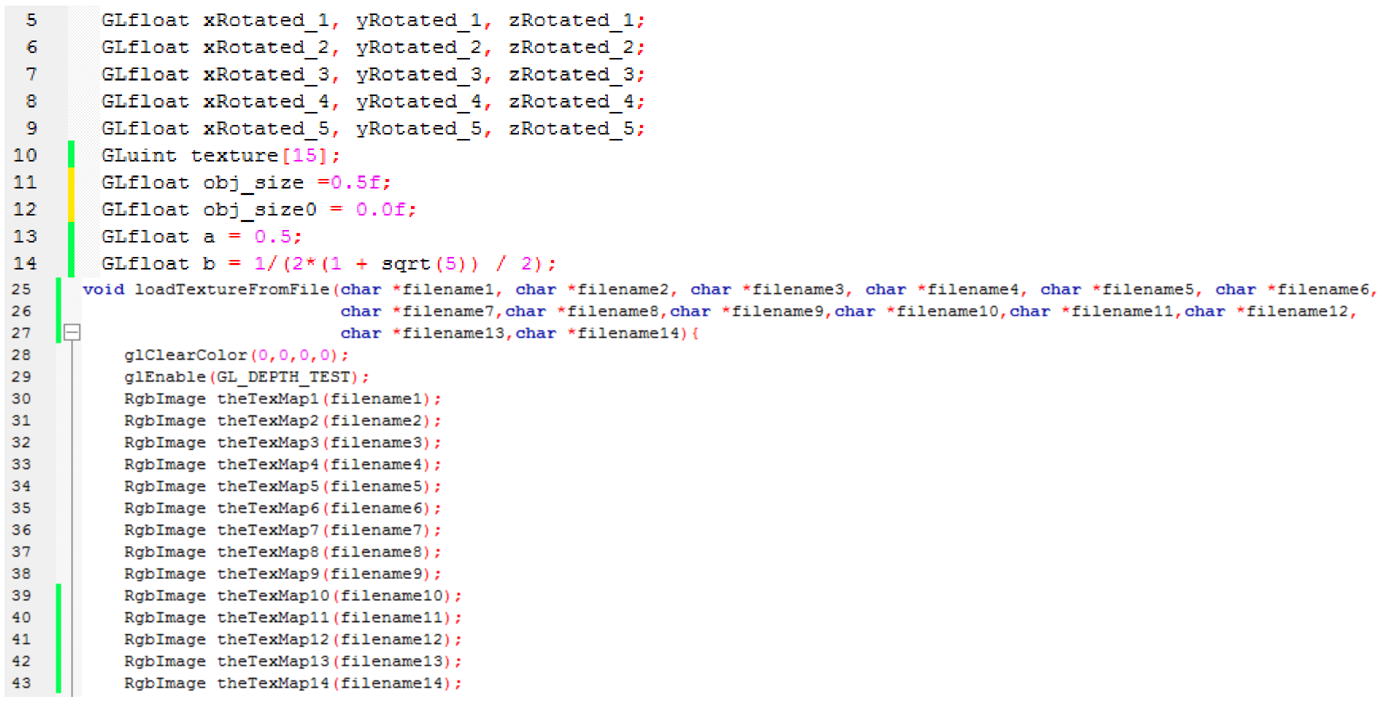
Computer Graphics HW04

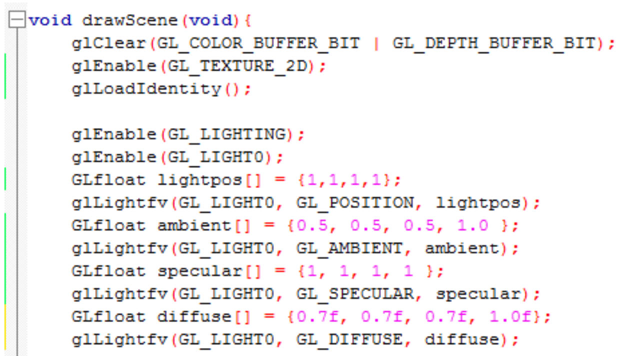
소프트웨어학부
소프트웨어학과
2017012197
여채린

문영식 교수님

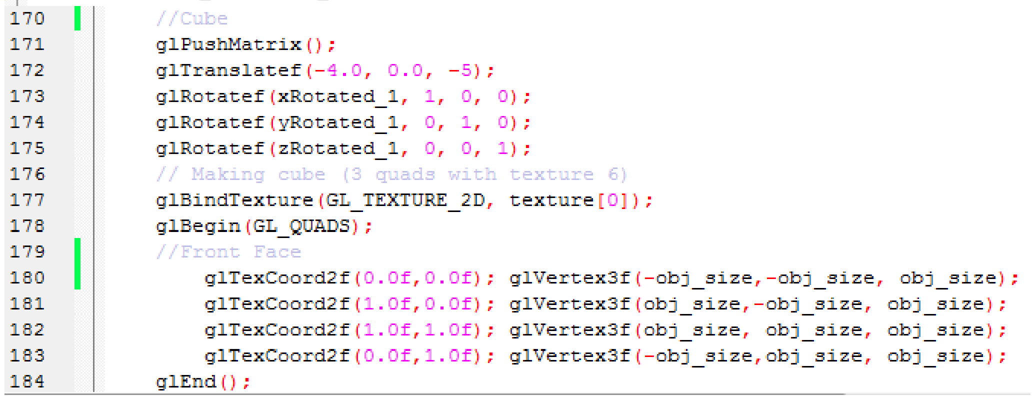
1. Source code for the main part of the program



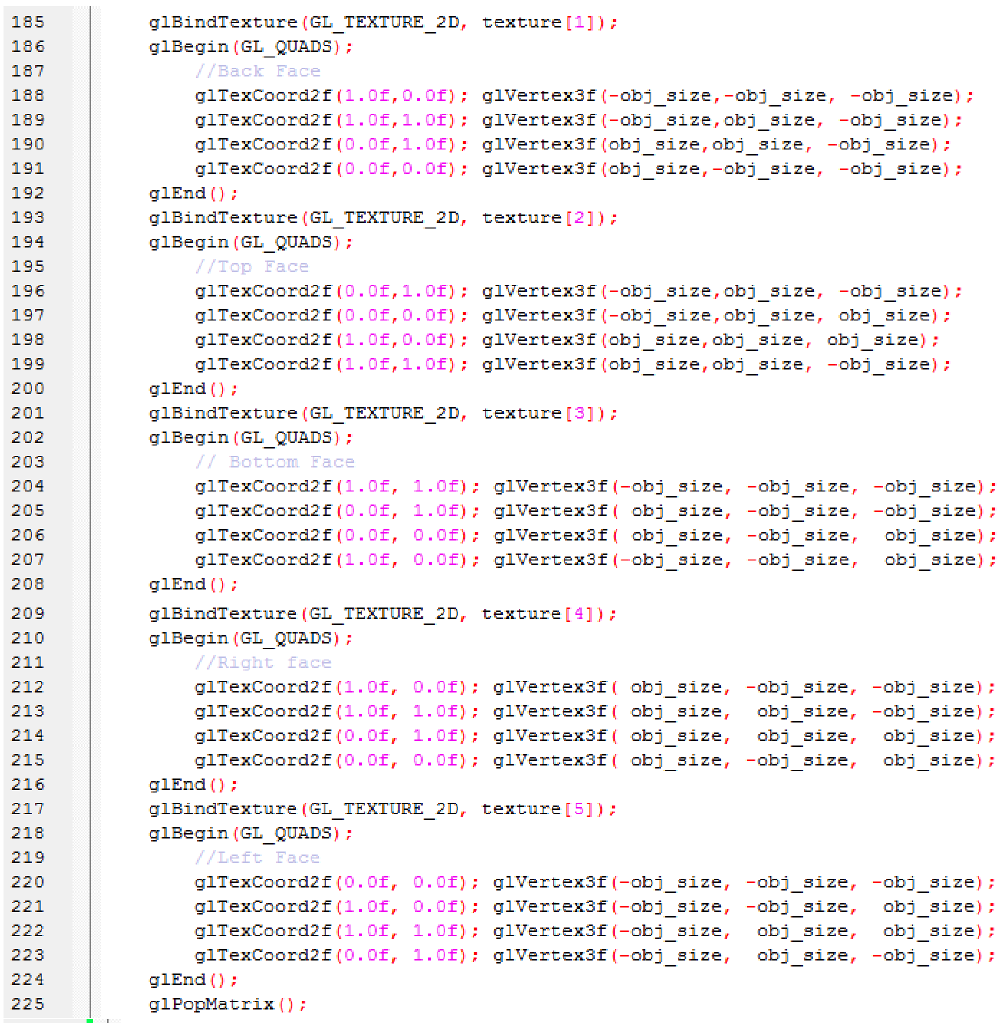
```
GLfloat xRotated_1, yRotated_1, zRotated_1;
GLfloat xRotated_2, yRotated_2, zRotated_2;
GLfloat xRotated_3, yRotated_3, zRotated_3;
GLfloat xRotated_4, yRotated_4, zRotated_4;
GLfloat xRotated_5, yRotated_5, zRotated_5;
GLuint texture[15];
GLfloat obj_size =0.5f;
GLfloat obj_size0 = 0.0f;
GLfloat a = 0.5;
GLfloat b = 1/(2*(1 + sqrt(5)) / 2);
void loadTextureFromFile(char *filename1, char *filename2, char *filename3, char *filename4, char *filename5, char *filename6,
                         char *filename7,char *filename8,char *filename9,char *filename10,char *filename11,char *filename12,
                         char *filename13,char *filename14){
    glClearColor(0,0,0,0);
    glEnable(GL_DEPTH_TEST);
    RgbImage theTexMap1(filename1);
    RgbImage theTexMap2(filename2);
    RgbImage theTexMap3(filename3);
    RgbImage theTexMap4(filename4);
    RgbImage theTexMap5(filename5);
    RgbImage theTexMap6(filename6);
    RgbImage theTexMap7(filename7);
    RgbImage theTexMap8(filename8);
    RgbImage theTexMap9(filename9);
    RgbImage theTexMap10(filename10);
    RgbImage theTexMap11(filename11);
    RgbImage theTexMap12(filename12);
    RgbImage theTexMap13(filename13);
    RgbImage theTexMap14(filename14);
```



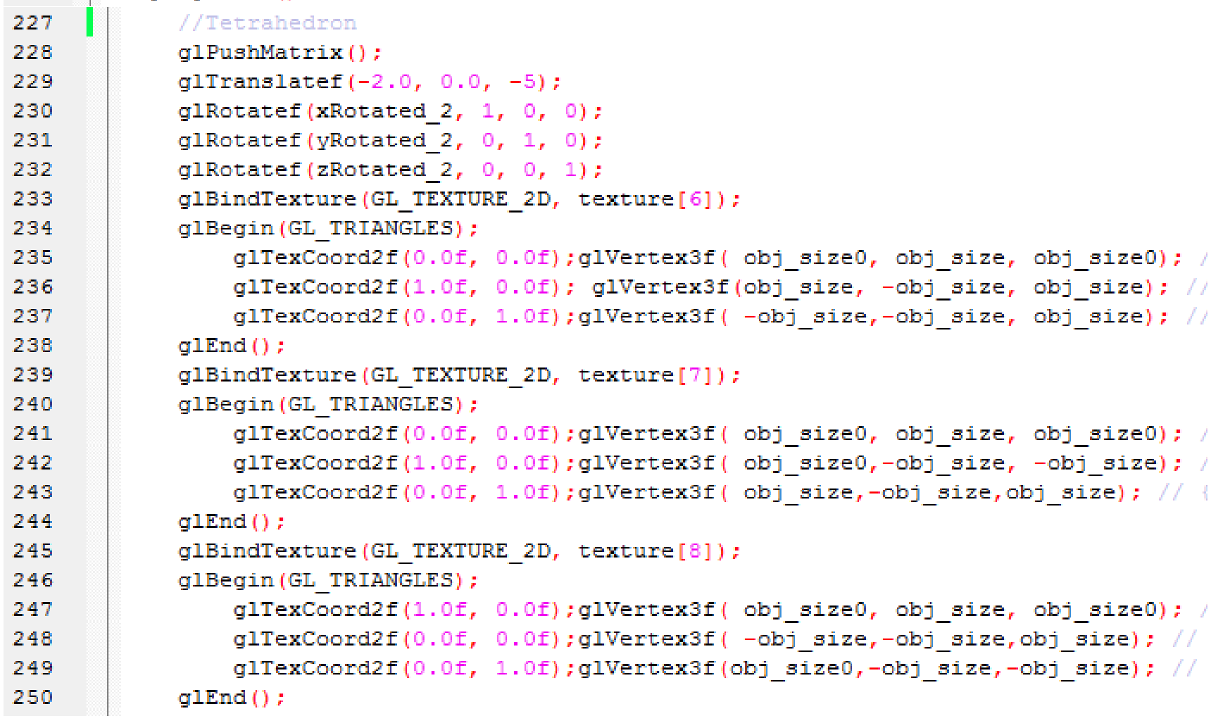
```
void drawScene(void){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_TEXTURE_2D);
    glLoadIdentity();

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    GLfloat lightpos[] = {1,1,1,1};
    glLightfv(GL_LIGHT0, GL_POSITION, lightpos);
    GLfloat ambient[] = {0.5, 0.5, 0.5, 1.0 };
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    GLfloat specular[] = {1, 1, 1, 1 };
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
    GLfloat diffuse[] = {0.7f, 0.7f, 0.7f, 1.0f};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
```



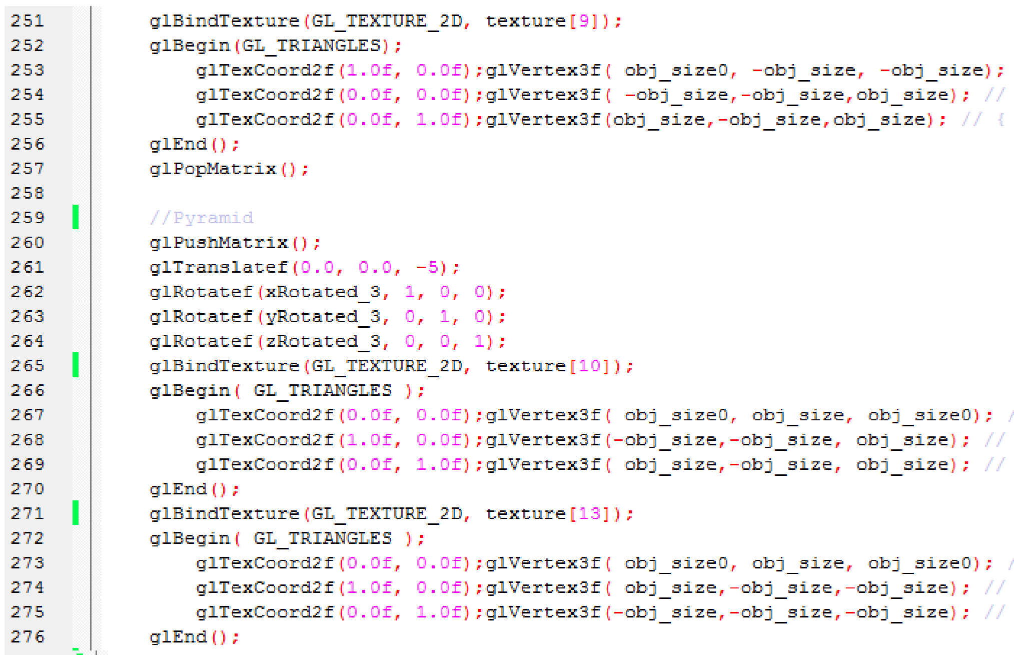
```
    //Cube
    glPushMatrix();
    glTranslatef(-4.0, 0.0, -5);
    glRotatef(xRotated_1, 1, 0, 0);
    glRotatef(yRotated_1, 0, 1, 0);
    glRotatef(zRotated_1, 0, 0, 1);
    // Making cube (3 quads with texture 6)
    glBindTexture(GL_TEXTURE_2D, texture[0]);
    glBegin(GL_QUADS);
    //Front Face
        glTexCoord2f(0.0f,0.0f); glVertex3f(-obj_size,-obj_size, obj_size);
        glTexCoord2f(1.0f,0.0f); glVertex3f(obj_size,-obj_size, obj_size);
        glTexCoord2f(1.0f,1.0f); glVertex3f(obj_size, obj_size, obj_size);
        glTexCoord2f(0.0f,1.0f); glVertex3f(-obj_size,obj_size, obj_size);
    glEnd();
```



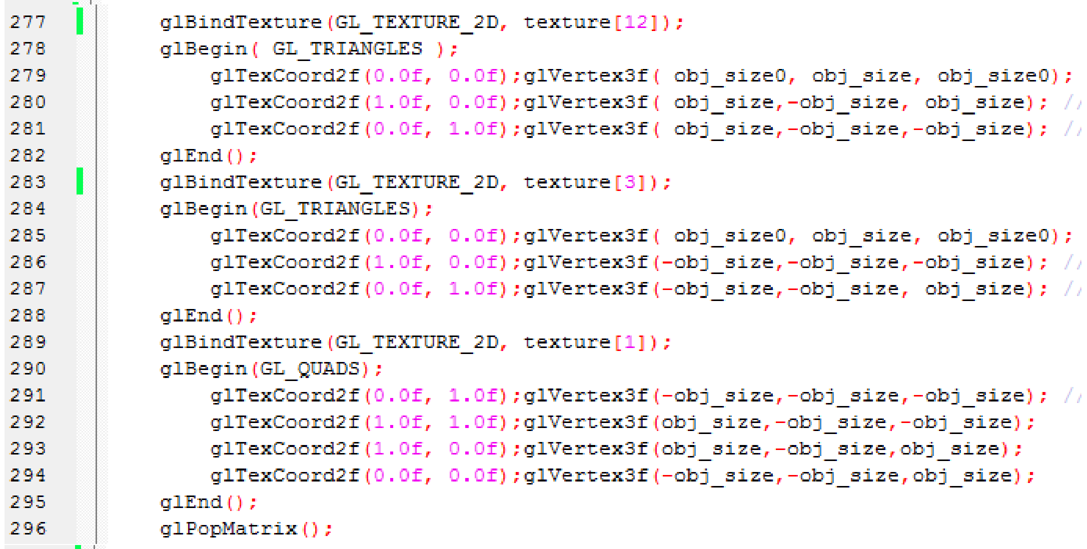
```
    glBindTexture(GL_TEXTURE_2D, texture[1]);
    glBegin(GL_QUADS);
        //Back Face
        glTexCoord2f(1.0f,0.0f); glVertex3f(-obj_size,-obj_size, -obj_size);
        glTexCoord2f(1.0f,1.0f); glVertex3f(-obj_size,obj_size, -obj_size);
        glTexCoord2f(0.0f,1.0f); glVertex3f(obj_size,obj_size, -obj_size);
        glTexCoord2f(0.0f,0.0f); glVertex3f(obj_size,-obj_size, -obj_size);
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[2]);
    glBegin(GL_QUADS);
        //Top Face
        glTexCoord2f(0.0f,1.0f); glVertex3f(-obj_size,obj_size, -obj_size);
        glTexCoord2f(0.0f,0.0f); glVertex3f(-obj_size,obj_size, obj_size);
        glTexCoord2f(1.0f,0.0f); glVertex3f(obj_size,obj_size, obj_size);
        glTexCoord2f(1.0f,1.0f); glVertex3f(obj_size,obj_size, -obj_size);
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[3]);
    glBegin(GL_QUADS);
        // Bottom Face
        glTexCoord2f(1.0f, 1.0f); glVertex3f(-obj_size, -obj_size, -obj_size);
        glTexCoord2f(0.0f, 1.0f); glVertex3f( obj_size, -obj_size, -obj_size);
        glTexCoord2f(0.0f, 0.0f); glVertex3f( obj_size, -obj_size,  obj_size);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(-obj_size, -obj_size,  obj_size);
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[4]);
    glBegin(GL_QUADS);
        //Right face
        glTexCoord2f(1.0f, 0.0f); glVertex3f( obj_size, -obj_size, -obj_size);
        glTexCoord2f(1.0f, 1.0f); glVertex3f( obj_size,  obj_size, -obj_size);
        glTexCoord2f(0.0f, 1.0f); glVertex3f( obj_size,  obj_size,  obj_size);
        glTexCoord2f(0.0f, 0.0f); glVertex3f( obj_size, -obj_size,  obj_size);
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[5]);
    glBegin(GL_QUADS);
        //Left Face
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-obj_size, -obj_size, -obj_size);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(-obj_size, -obj_size,  obj_size);
        glTexCoord2f(1.0f, 1.0f); glVertex3f(-obj_size,  obj_size,  obj_size);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(-obj_size,  obj_size, -obj_size);
    glEnd();
    glPopMatrix();
```



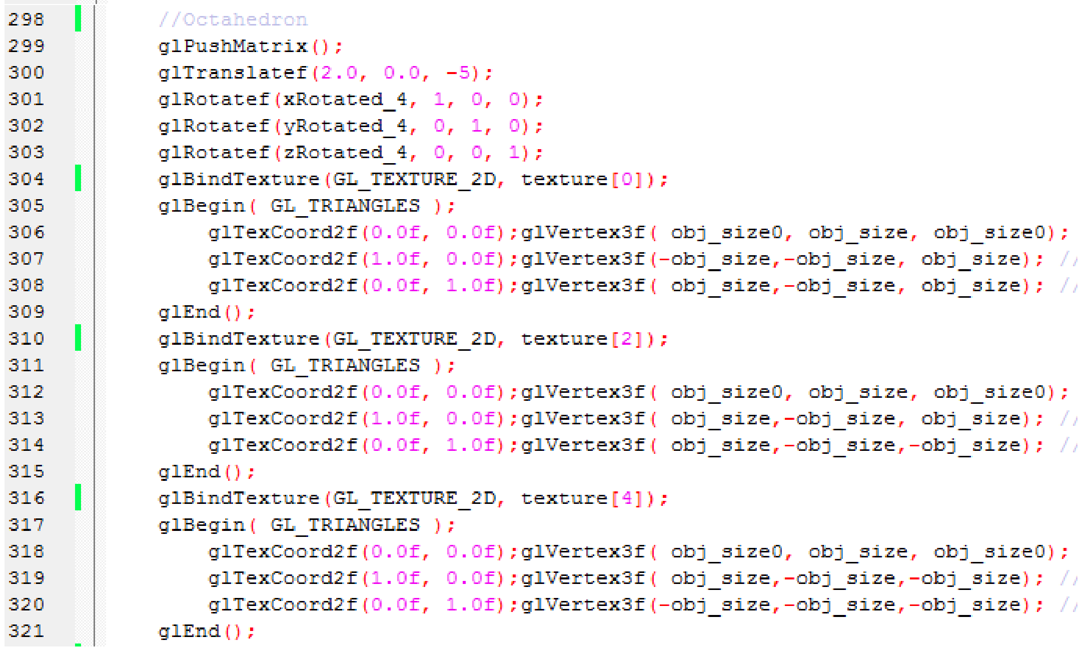
```
    //Tetrahedron
    glPushMatrix();
    glTranslatef(-2.0, 0.0, -5);
    glRotatef(xRotated_2, 1, 0, 0);
    glRotatef(yRotated_2, 0, 1, 0);
    glRotatef(zRotated_2, 0, 0, 1);
    glBindTexture(GL_TEXTURE_2D, texture[6]);
    glBegin(GL_TRIANGLES);
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); /
        glTexCoord2f(1.0f, 0.0f); glVertex3f(obj_size, -obj_size, obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f( -obj_size,-obj_size, obj_size); //
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[7]);
    glBegin(GL_TRIANGLES);
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); /
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size0,-obj_size, -obj_size); /
        glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size,obj_size); // (
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[8]);
    glBegin(GL_TRIANGLES);
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); /
        glTexCoord2f(0.0f, 0.0f);glVertex3f( -obj_size,-obj_size,obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f(obj_size0,-obj_size,-obj_size); //
    glEnd();
```



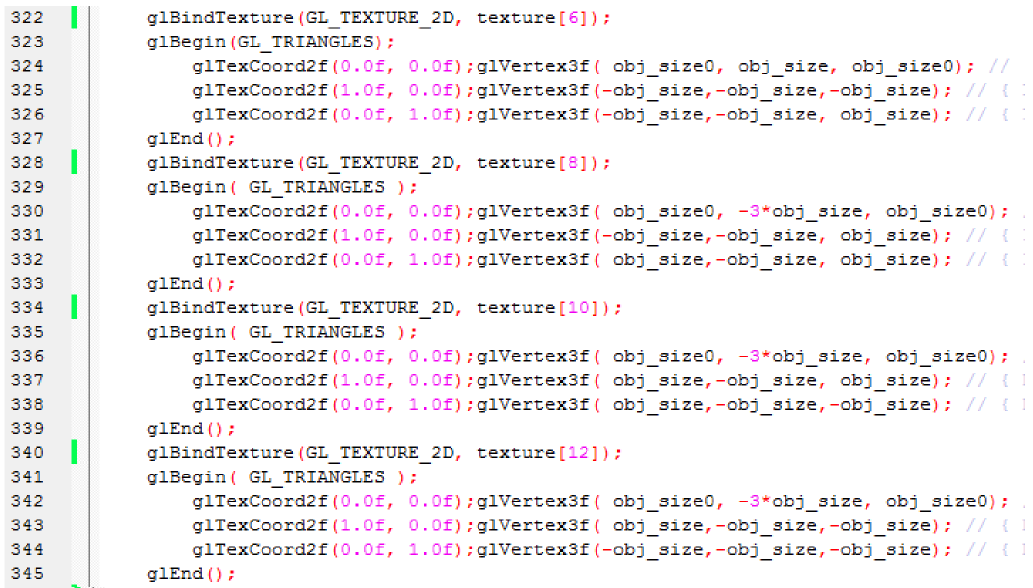
```
    glBindTexture(GL_TEXTURE_2D, texture[9]);
    glBegin(GL_TRIANGLES);
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size0, -obj_size, -obj_size);
        glTexCoord2f(0.0f, 0.0f);glVertex3f( -obj_size,-obj_size,obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f(obj_size,-obj_size,obj_size); // (
    glEnd();
    glPopMatrix();

    //Pyramid
    glPushMatrix();
    glTranslatef(0.0, 0.0, -5);
    glRotatef(xRotated_3, 1, 0, 0);
    glRotatef(yRotated_3, 0, 1, 0);
    glRotatef(zRotated_3, 0, 0, 1);
    glBindTexture(GL_TEXTURE_2D, texture[10]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); /
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size, obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size, obj_size); //
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[13]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); /
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size,-obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size,-obj_size); //
    glEnd();
```



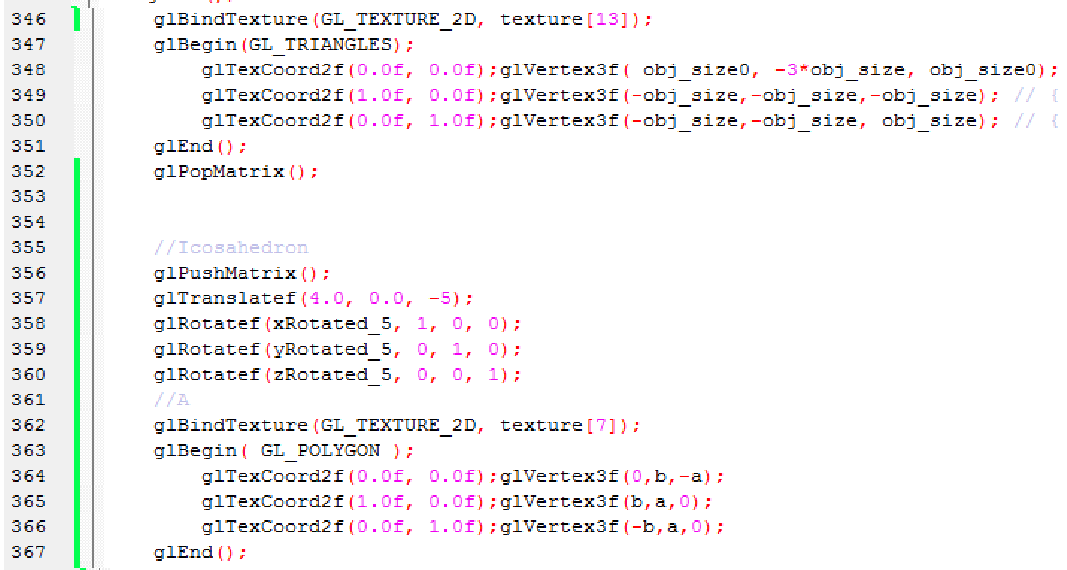
```
    glBindTexture(GL_TEXTURE_2D, texture[12]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size, obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size,-obj_size); //
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[3]);
    glBegin(GL_TRIANGLES);
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size,-obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size, obj_size); //
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[1]);
    glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size,-obj_size); //
        glTexCoord2f(1.0f, 1.0f);glVertex3f(obj_size,-obj_size,-obj_size);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(obj_size,-obj_size,obj_size);
        glTexCoord2f(0.0f, 0.0f);glVertex3f(-obj_size,-obj_size,obj_size);
    glEnd();
    glPopMatrix();
```



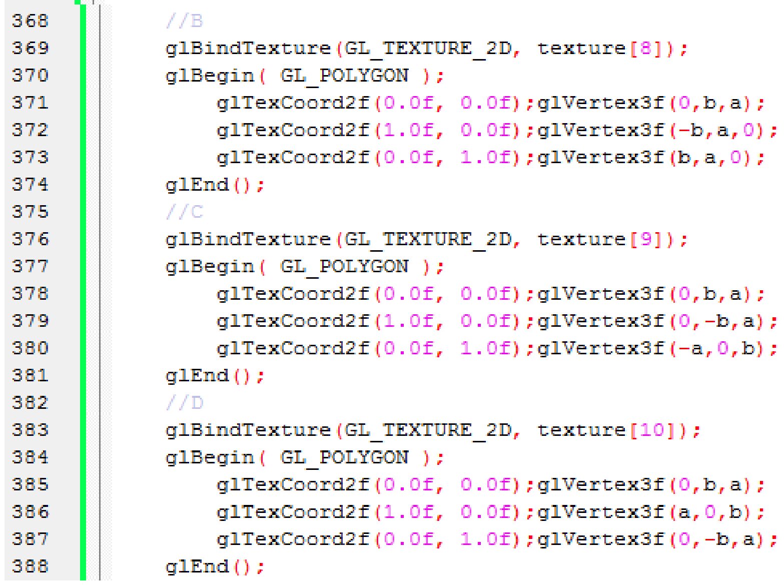
```
    //Octahedron
    glPushMatrix();
    glTranslatef(2.0, 0.0, -5);
    glRotatef(xRotated_4, 1, 0, 0);
    glRotatef(yRotated_4, 0, 1, 0);
    glRotatef(zRotated_4, 0, 0, 1);
    glBindTexture(GL_TEXTURE_2D, texture[0]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size, obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size, obj_size); //
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[2]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size, obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size,-obj_size); //
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[4]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size,-obj_size); //
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size,-obj_size); //
    glEnd();
```



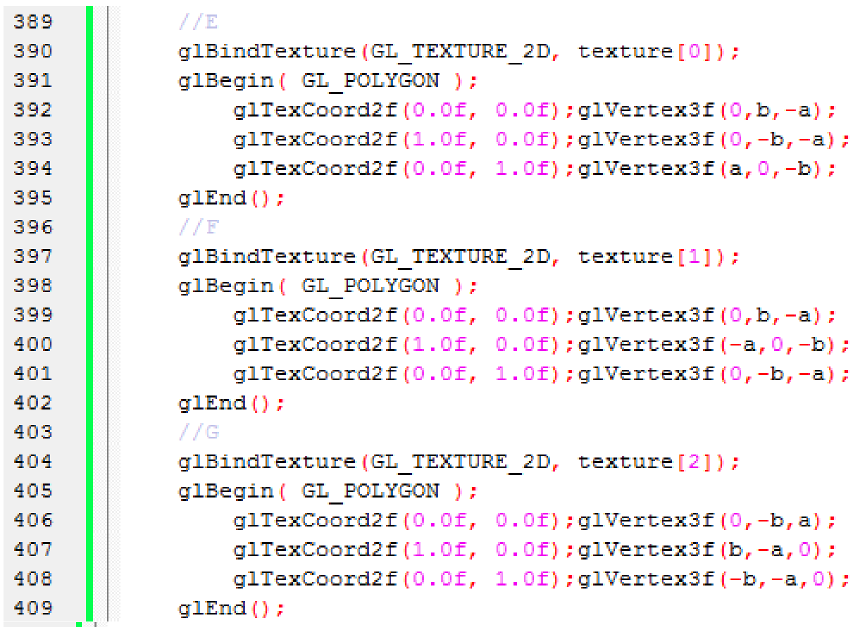
```
    glBindTexture(GL_TEXTURE_2D, texture[6]);
    glBegin(GL_TRIANGLES);
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); //
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size,-obj_size); // {
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size, obj_size); // {
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[8]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, -3*obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size, obj_size); // {
        glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size, obj_size); // {
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[10]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, -3*obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size, obj_size); // {
        glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size,-obj_size); // {
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[12]);
    glBegin( GL_TRIANGLES );
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, -3*obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size,-obj_size); // {
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size,-obj_size); // {
    glEnd();
```



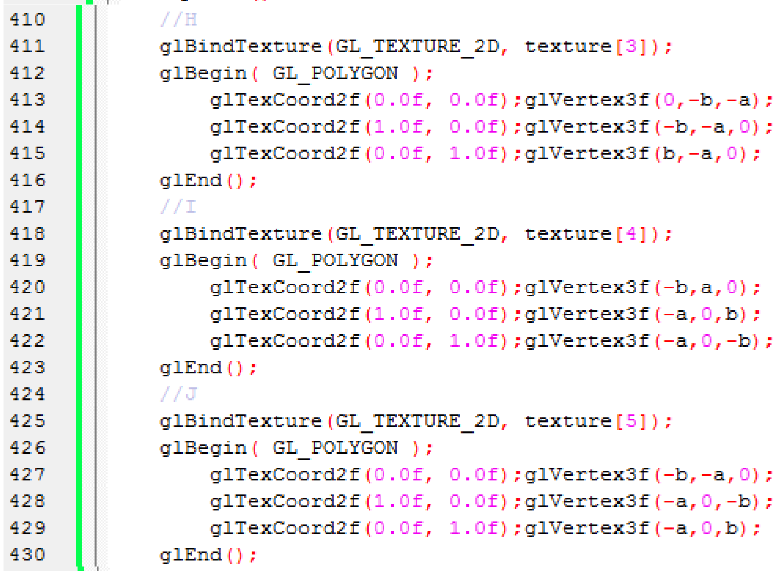
```
    glBindTexture(GL_TEXTURE_2D, texture[13]);
    glBegin(GL_TRIANGLES);
        glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, -3*obj_size, obj_size0);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size,-obj_size); // {
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size, obj_size); // {
    glEnd();
    glPopMatrix();


    //Icosahedron
    glPushMatrix();
    glTranslatef(4.0, 0.0, -5);
    glRotatef(xRotated_5, 1, 0, 0);
    glRotatef(yRotated_5, 0, 1, 0);
    glRotatef(zRotated_5, 0, 0, 1);
    //A
    glBindTexture(GL_TEXTURE_2D, texture[7]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(b,a,0);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-b,a,0);
    glEnd();
```



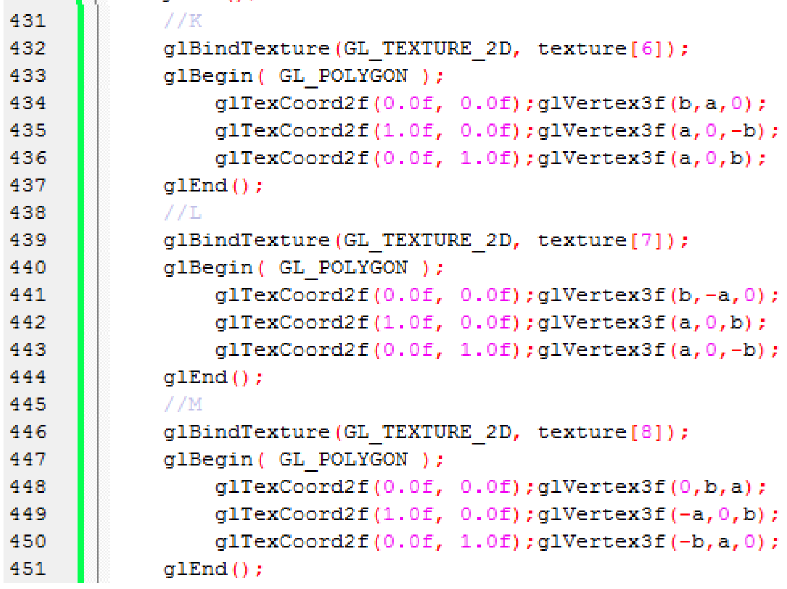
```
    //B
    glBindTexture(GL_TEXTURE_2D, texture[8]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-b,a,0);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(b,a,0);
    glEnd();
    //C
    glBindTexture(GL_TEXTURE_2D, texture[9]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(0,-b,a);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,b);
    glEnd();
    //D
    glBindTexture(GL_TEXTURE_2D, texture[10]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,b);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(0,-b,a);
    glEnd();
```



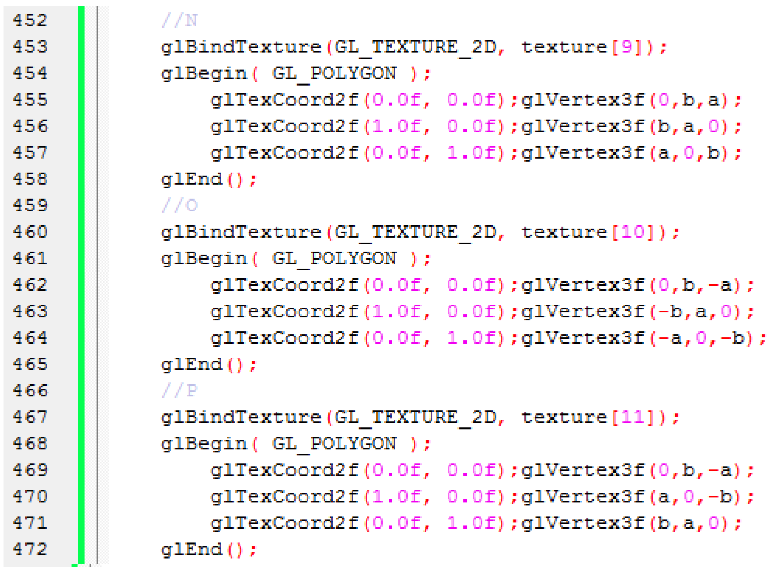
```
//E
glBindTexture(GL_TEXTURE_2D, texture[0]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(0,-b,-a);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,-b);
glEnd();
//F
glBindTexture(GL_TEXTURE_2D, texture[1]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,-b);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(0,-b,-a);
glEnd();
//G
glBindTexture(GL_TEXTURE_2D, texture[2]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,a);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(b,-a,0);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(-b,-a,0);
glEnd();
```



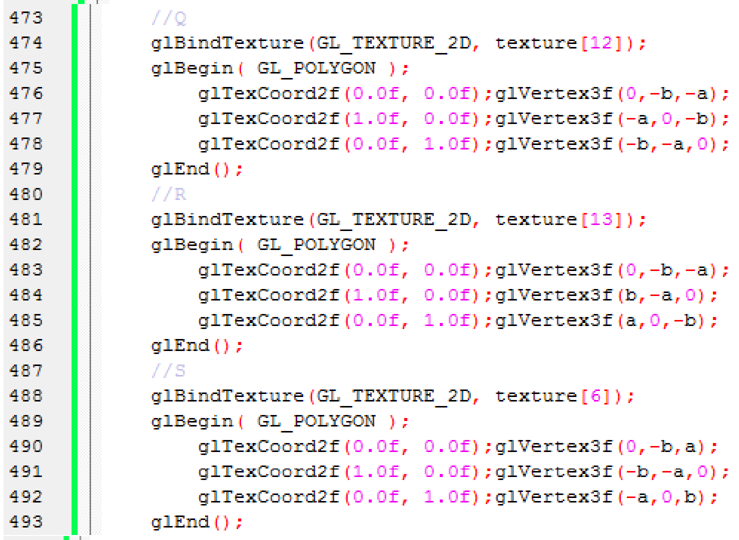
```
//H
glBindTexture(GL_TEXTURE_2D, texture[3]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,-a);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(-b,-a,0);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(b,-a,0);
glEnd();
//I
glBindTexture(GL_TEXTURE_2D, texture[4]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(-b,a,0);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,b);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,-b);
glEnd();
//J
glBindTexture(GL_TEXTURE_2D, texture[5]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(-b,-a,0);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,-b);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,b);
glEnd();
```



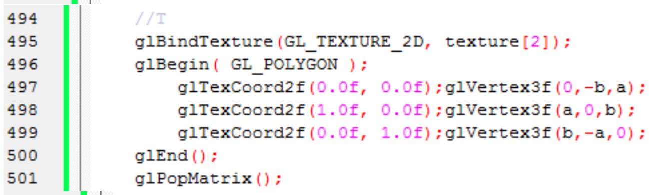
```
//K
glBindTexture(GL_TEXTURE_2D, texture[6]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(b,a,0);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,-b);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,b);
glEnd();
//L
glBindTexture(GL_TEXTURE_2D, texture[7]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(b,-a,0);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,b);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,-b);
glEnd();
//M
glBindTexture(GL_TEXTURE_2D, texture[8]);
glBegin( GL_POLYGON );
    glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
    glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,b);
    glTexCoord2f(0.0f, 1.0f);glVertex3f(-b,a,0);
glEnd();
```



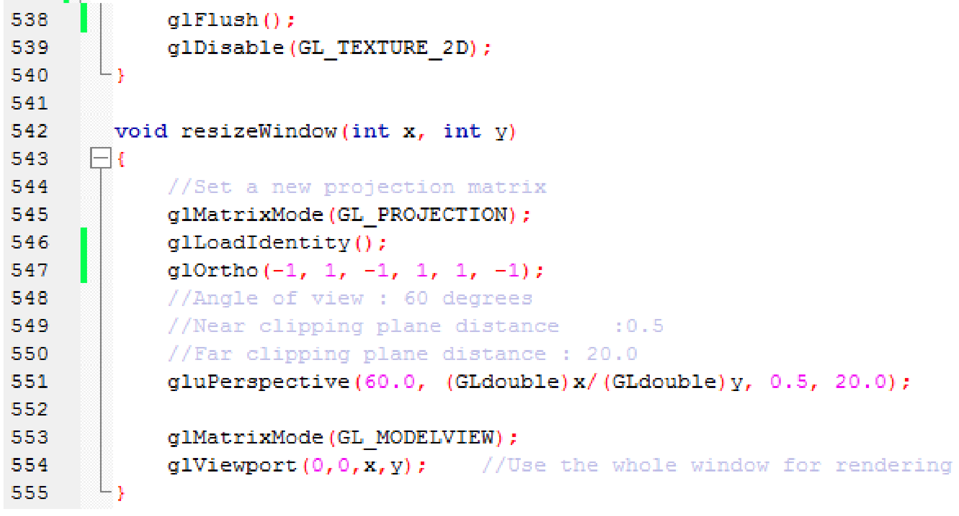
```
    //N
    glBindTexture(GL_TEXTURE_2D, texture[9]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(b,a,0);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,b);
    glEnd();
    //O
    glBindTexture(GL_TEXTURE_2D, texture[10]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-b,a,0);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,-b);
    glEnd();
    //P
    glBindTexture(GL_TEXTURE_2D, texture[11]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,-b);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(b,a,0);
    glEnd();
```



```
    //Q
    glBindTexture(GL_TEXTURE_2D, texture[12]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,-a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,-b);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-b,-a,0);
    glEnd();
    //R
    glBindTexture(GL_TEXTURE_2D, texture[13]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,-a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(b,-a,0);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,-b);
    glEnd();
    //S
    glBindTexture(GL_TEXTURE_2D, texture[6]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(-b,-a,0);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,b);
    glEnd();
```



```
    //T
    glBindTexture(GL_TEXTURE_2D, texture[2]);
    glBegin( GL_POLYGON );
        glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,a);
        glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,b);
        glTexCoord2f(0.0f, 1.0f);glVertex3f(b,-a,0);
    glEnd();
    glPopMatrix();
```

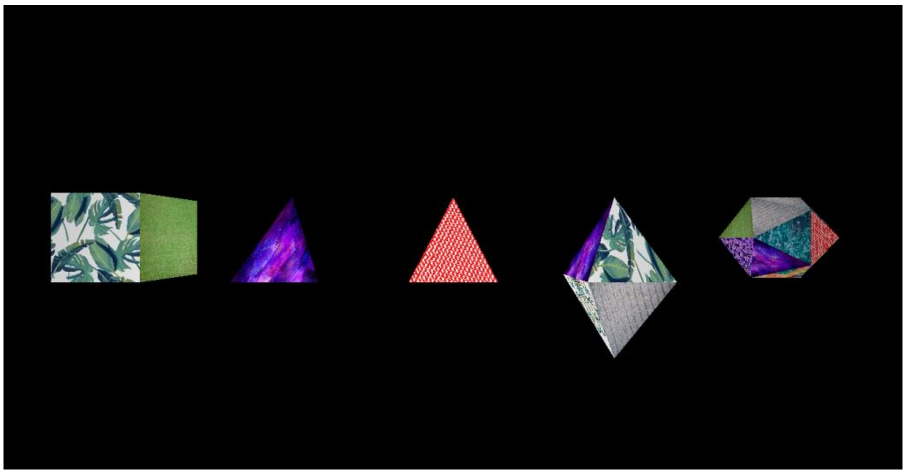


```
    glFlush();
    glDisable(GL_TEXTURE_2D);
}

void resizeWindow(int x, int y)
{
    //Set a new projection matrix
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1, 1, -1, 1, 1, -1);
    //Angle of view : 60 degrees
    //Near clipping plane distance    :0.5
    //Far clipping plane distance : 20.0
    gluPerspective(60.0, (GLdouble)x/(GLdouble)y, 0.5, 20.0);

    glMatrixMode(GL_MODELVIEW);
    glViewport(0,0,x,y);    //Use the whole window for rendering
}
```

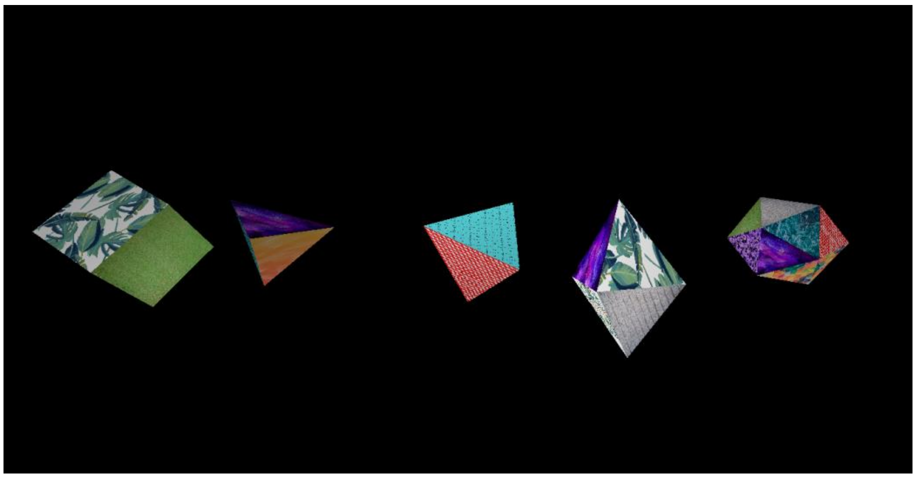
2. Screen capture to see the execution result

In order, hexagon, tetrahedron, pyramid, icosahedron in 3D with different textures.

- When use **GLfloat lightpos[] = {-1,-1,-1,1}, GLfloat ambient[] = {1,1,1,0}**

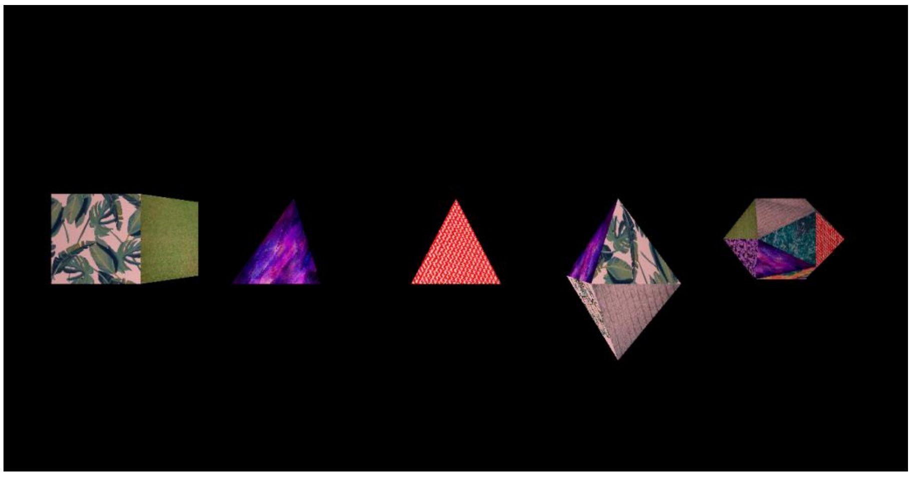


do not rotate

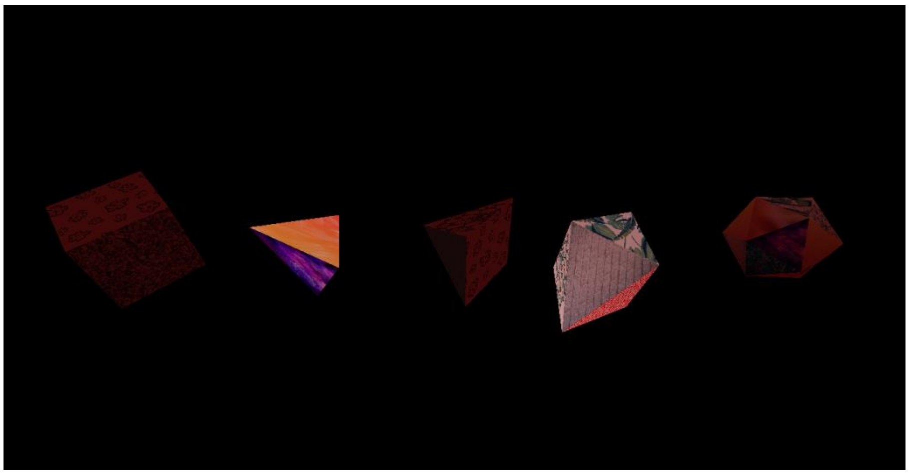


rotate individually

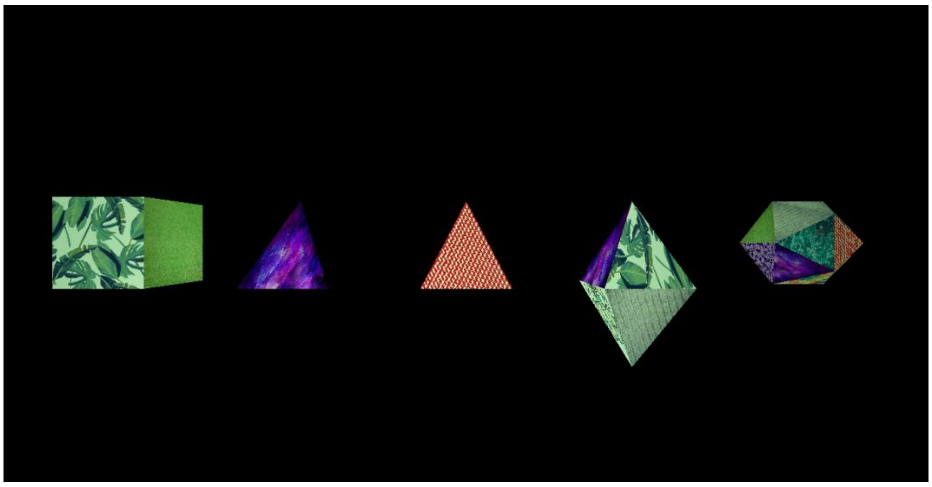
- When use **GLfloat lightpos[] = {-1,-1,-1,1}, GLfloat ambient[] = {1,0,0,0} (Red Light)**

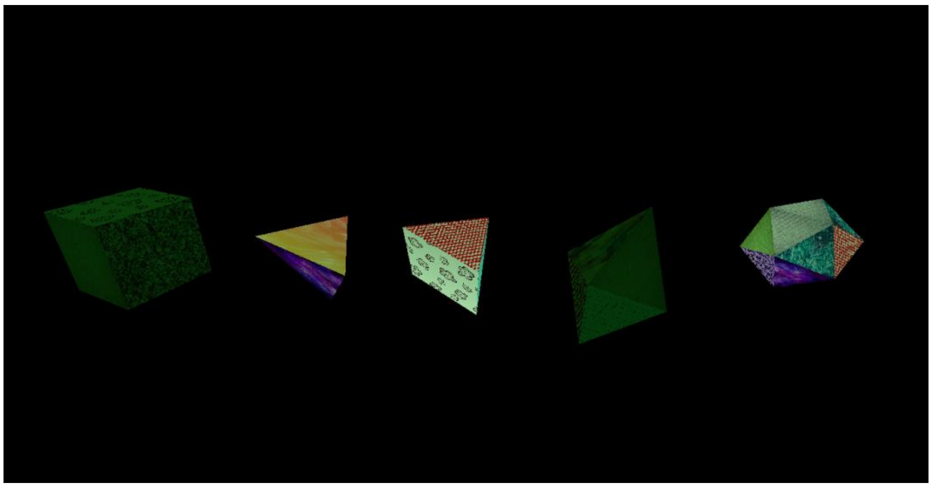


do not rotate

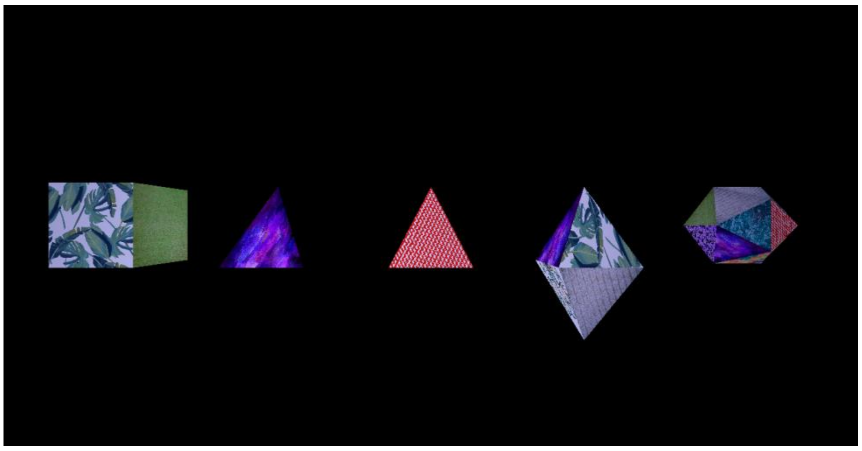

rotate individually

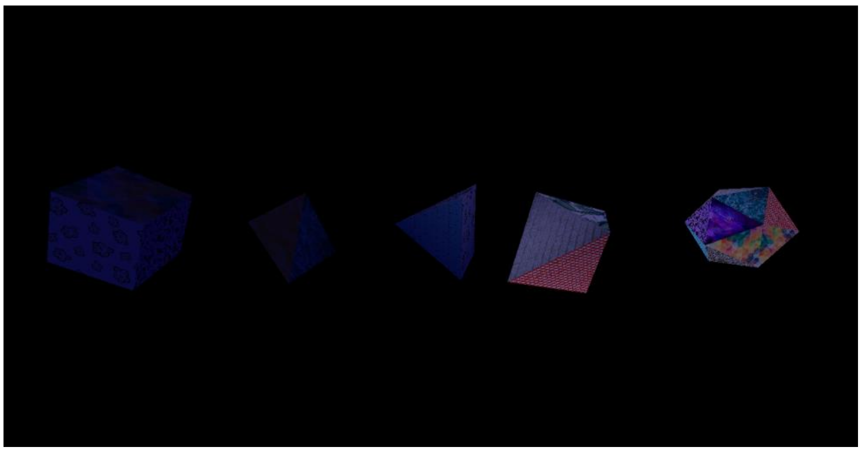
- When **use GLfloat lightpos[] = {-1,-1,-1,1}, GLfloat ambient[] = {0,1,0,0} (Green Light)**


do not rotate

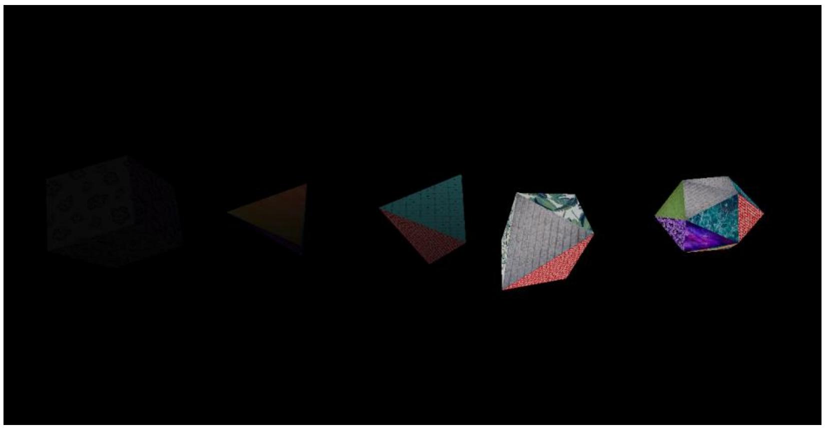

rotate individually

- When use **GLfloat lightpos[] = {-1,-1,-1,1}, GLfloat ambient[] = {0,1,0,0} (Blue Light)**

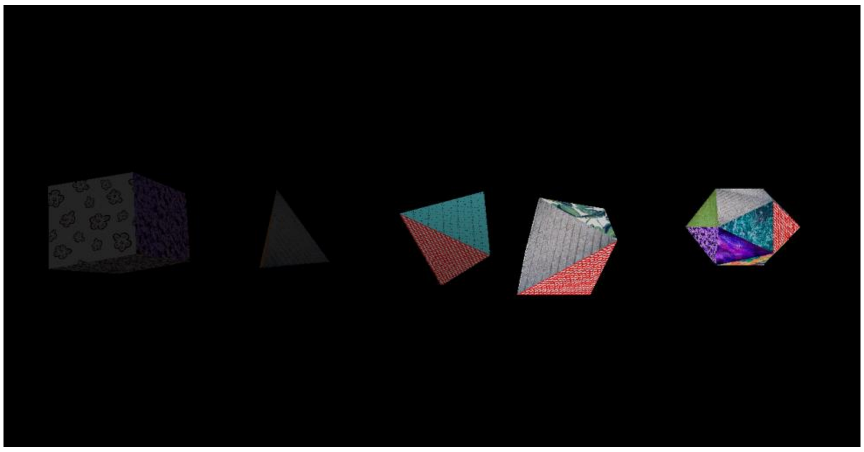

do not rotate


rotate individually

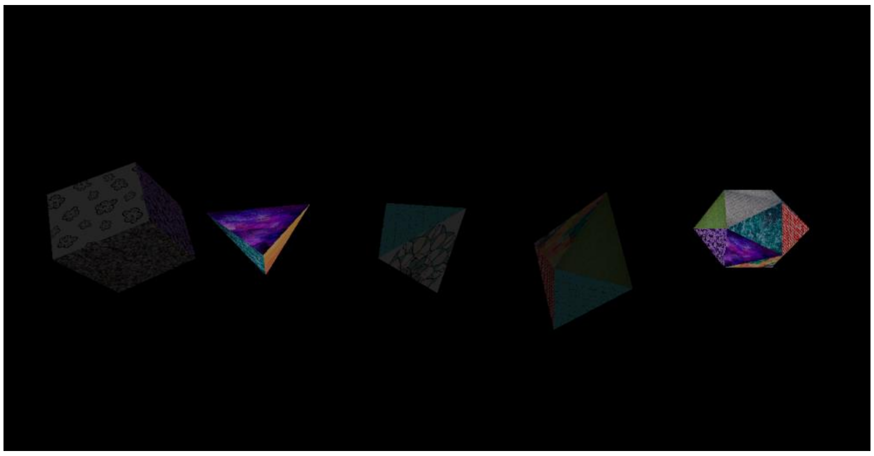
- When use **GLfloat lightpos[] = {1,1,1,1}, GLfloat specular[] = {1, 1, 1, 1 }**


rotate individually

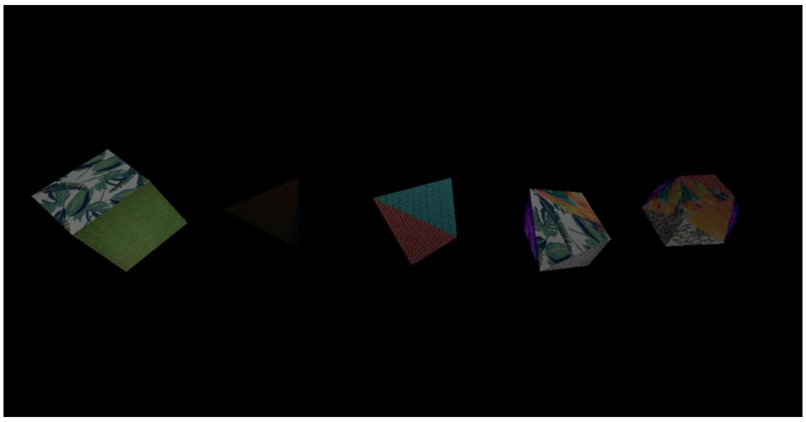
- When use **GLfloat lightpos[] = {1, 1, 1, 1}, GLfloat ambient[] = {0.5, 0.5, 0.5, 1.0 }, GLfloat specular[] = {1, 1, 1, 1 }**


rotate individually

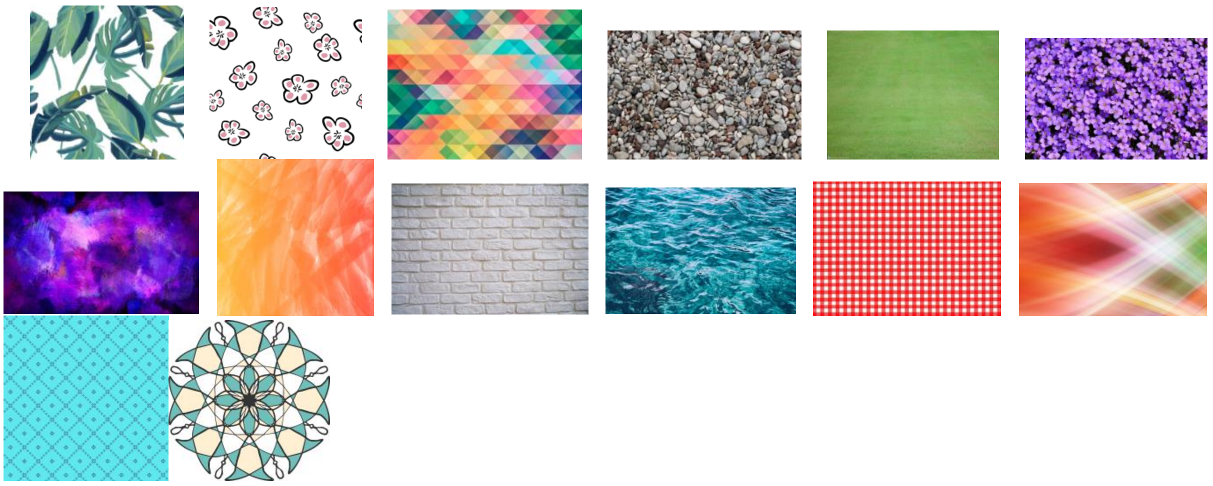
- When use **GLfloat lightpos[] = {0, 0, 0.5, 1}, GLfloat ambient[] = {0.5, 0.5, 0.5, 1.0 }, GLfloat specular[] = {1, 1, 1, 1 }, GLfloat diffuse[] = {0.7, 0.7, 0.7, 1.0}**


rotate individually

- When use **GLfloat lightpos[] = {1,1,1,1}, GLfloat diffuse[] = {0.7f, 0.7f, 0.7f, 1.0f}**

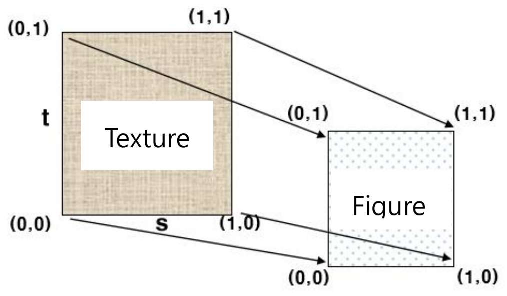

rotate individually

3. Analysis and Discussion of Execution Result



In this assignment, I applied a total of 14 textures to a 3D polygon as follows: All textures were downloaded from a free texture download site. I have made a cube, a tetrahedron, a pyramid, an octahedron and an icosahedron. Since the coordinates of the cube, the tetrahedron, the pyramid, and the octahedron are not very complicated, the size of the shape can be changed by two variables, obj_size and obj_size0. In the case of icosahedron, the coordinate values were calculated by setting a separate variable called a and b. A detailed description of these variables will follow later.

Here's how to draw each shape in openGL: Between glPushMatrix () and glPopMatrix (), set the position of the figure with glTranslatef, and rotate each figure individually using individual glRotatef. After selecting the texture to apply with glBindTexture, you can use glVertex3f to print the vertices of the shape you want to draw between glBegin () and glEnd (). At this time, use glTexCoord2f to paste the texture and the drawn shapes together. This will bring the texture you want to the screen.



Imagefrom:https://m.blog.naver.com/PostView.nhn?blogId=enter_maintanance&logNo=220945341294&proxyReferer=https%3A%2F%2Fwww.google.com%2F

For lightning, GL_AMBIENT, GL_SPECULAR, GL_DIFFUSE are used. I also adjusted the position of the light with GL_POSITION.

GL_AMBIENT is the degree of reflection of ambient light. Specifies the intensity of each color element in the (R, G, B, A) array. R is Red, G is Green, B is Blue, and A is transparency. Ambient light is light that covers the surroundings without a specific direction. It is expressed in constant brightness and color.

GL_SPECULAR is the intensity of reflected light. R, G, B, and A values are used. You can adjust the intensity of the reflected light with a value between. Reflected light is light that flows in a specific direction and is completely reflected in one direction. Strong reflected light creates a highlight on the object.

GL_DIFFUSE is the degree of reflection of diffused light. R, G, B, and A values are used. Diffuse light is light that enters a certain direction and is distributed in various directions on the surface of an object. Although scattered, the surface that receives this light appears brighter than it does. Similar to fluorescent light or sunlight.

In fact, I tried these three lightning effects, and adjusted the colors of the lights by adjusting the R, G, and B values. If there were a curved object, the difference in each lightning effects would be more apparent. Nevertheless, it was confirmed that the contrast of the object becomes more obvious when using specular alone than when using ambient and diffuse.

Now I will explain how I made each polygon. First of all, a cube has the following characteristics: Vertices: 8, Edges: 12, Faces: 6, Edges per face: 4, Edges per vertex: 3, Sin of angle at edge: 1, Surface area: 6 * edgelength ^ 2, Volume: edgelength ^ 3, Circumscribed radius: sqrt (3) / 2 * edgelength, Inscribed radius: 1/2 * edgelength. When calculating Coordinates,

(-1, -1, -1), (1, -1, -1), (1, -1, 1), (-1, -1, 1) / (-1, -1, -1), (-1, -1, 1), (-1, 1, 1), (-1, 1, -1) / (-1, -1, 1), (1, -1, 1), (1, 1, 1), (-1, 1, 1) / (-1, 1, -1), (-1, 1, 1), (1, 1, 1), (1, 1, -1) /(1, -1, -1), (1, 1, -1), (1, 1, 1), (1, -1, 1) / (-1, -1, -1), (-1, 1, -1), (1, 1, -1), (1, -1, -1)

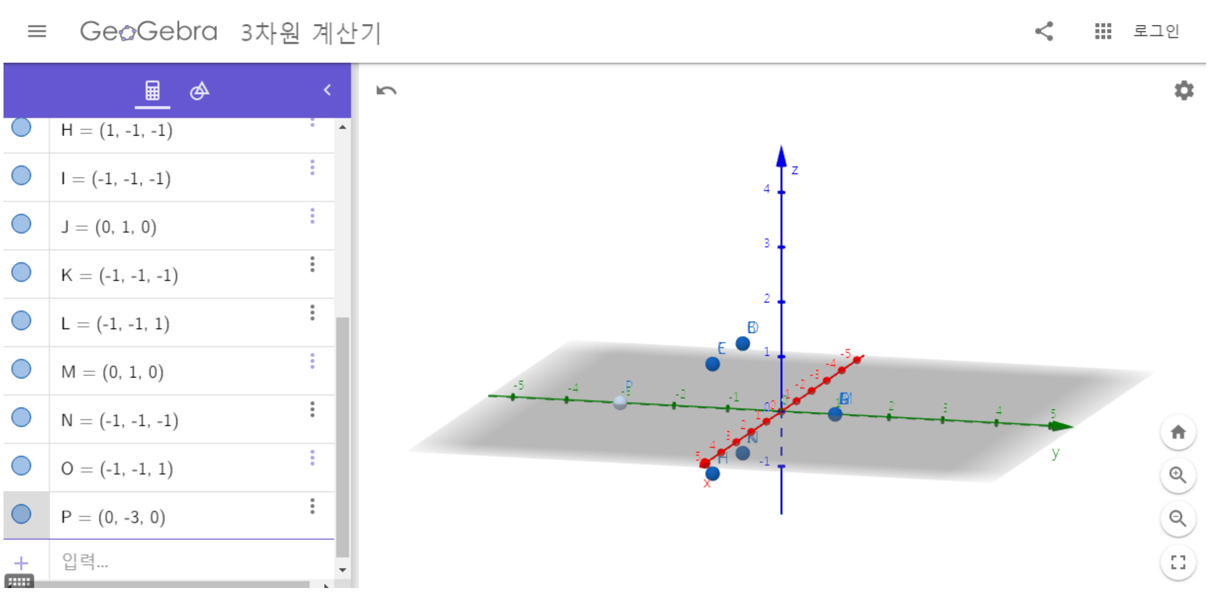
The tetrahedron has the following characteristics : Vertices: 4, Edges: 6, Faces: 4, Edges per face: 3, Edges per vertex: 3, Sin of angle at edge: 2 * sqrt (2) / 3, Surface area: sqrt (3) * edgelength ^ 2, Volume: sqrt (2) / 12 * edgelength ^ 3, Circumscribed radius: sqrt (6) / 4 * edgelength, Inscribed radius: sqrt (6) / 12 * edgelength. When calculating Coordinates,

(1, 1, 1), (-1, 1, -1), (1, -1, -1) / (-1, 1, -1), (-1, -1, 1), (1, -1, -1) / (1, 1, 1), (1, -1, -1), (-1, -1, 1) / (1, 1, 1), (-1, -1, 1), (-1, 1, -1)

Octahedron has the following characteristics: Vertices: 6, Edges: 12, Faces: 8, Edges per face: 3, Edges per vertex: 4, Sin of angle at edge: 2 * sqrt (2) / 3, Surface area: 2 * sqrt (3) * edgelength ^ 2, Volume: sqrt (2) / 3 * edgelength ^ 3, Circumscribed radius: sqrt (2) / 2 * edgelength, Inscribed radius: sqrt (6) / 6 * edgelength. When calculating Coordinates,

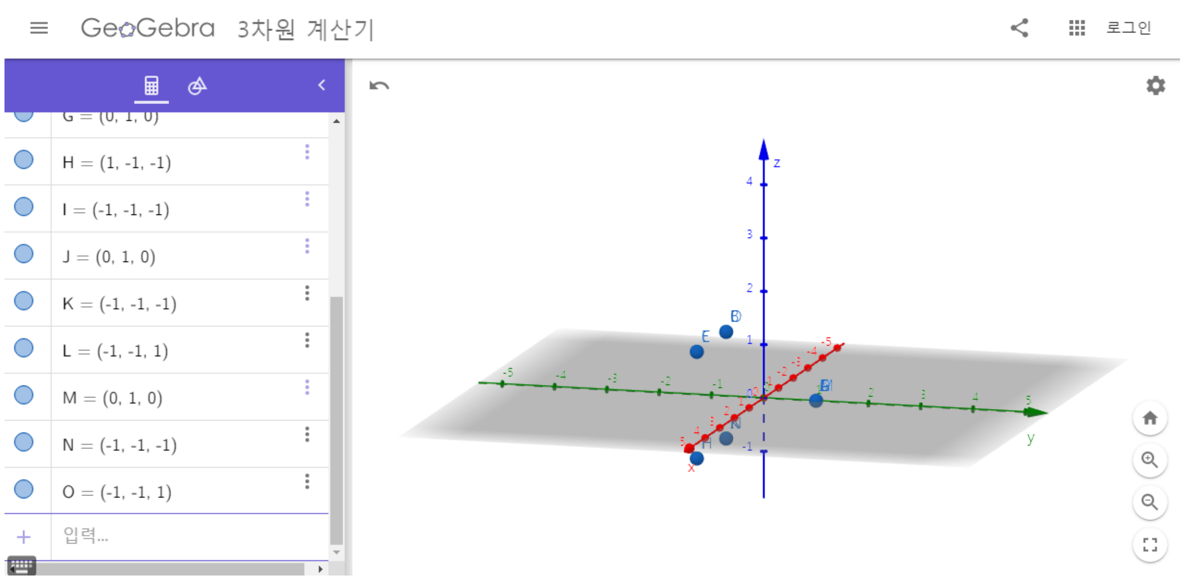
(-a, 0, a), (-a, 0, -a), (0, b, 0) / (-a, 0, -a), (a, 0, -a), (0, b, 0) / (a, 0, -a), (a, 0, a), (0, b, 0) / (a, 0, a), (-a, 0, a), (0, b, 0) / (a, 0, -a), (-a, 0, -a), (0, -b, 0) / (-a, 0, -a), (-a, 0, a), (0, -b, 0) / (a, 0, a), (a, 0, -a), (0, -b, 0) / (-a, 0, a), (a, 0, a), (0, -b, 0)

Or, Coordinates can be created with only 0 and 1. The coordinates of an octahedron created by taking direct coordinates from GeoGebra are as follows.


GeoGebra 3차원 계산기 로그인

H = (1, -1, -1)
I = (-1, -1, -1)
J = (0, 1, 0)
K = (-1, -1, -1)
L = (-1, -1, 1)
M = (0, 1, 0)
N = (-1, -1, -1)
O = (-1, -1, 1)
P = (0, -3, 0)
입력...

The pyramid is an application of octahedron. Removing one of the following coordinates from an octahedron creates a pyramid. Note that removing one of the coordinates below will result in a perforated pyramid. Therefore, you must map the texture by specifying the coordinates of the four points on the bottom. I tried to plot the pyramid using only 0,1, like the octahedron.


GeoGebra 3차원 계산기 로그인

G = (0, 1, 0)
H = (1, -1, -1)
I = (-1, -1, -1)
J = (0, 1, 0)
K = (-1, -1, -1)
L = (-1, -1, 1)
M = (0, 1, 0)
N = (-1, -1, -1)
O = (-1, -1, 1)
입력...

Icosahedron has the following characteristics: Vertices: 12, Edges: 30, Faces: 20, Edges per face: 3, Edges per vertex: 5, Sin of angle at edge: 2/3, Surface area: 5 * sqrt (3) * edgelength ^ 2, Volume: 5 * (3 + sqrt (5)) / 12 * edgelength ^ 3, Circumscribed radius: sqrt (10 + 2 * sqrt (5)) / 4 * edgelength, Inscribed radius: sqrt (42 + 18 * sqrt (5)) / 12 * edgelength, When calculating

Coordinates, a = 1 / (2 * sqrt (2)) and b = 1/2. a = 1/2 and b = 1 / (2 * (1 + sqrt (5)) / 2). . When calculating Coordinates,

(0, b, -a), (b, a, 0), (-b, a, 0) / (0, b, a), (-b, a, 0), (b, a, 0) / (0, b, a), (0, -b, a), (-a, 0, b) / (0, b, a), (a, 0, b), (0,-b, a) / (0, b, -a), (0, -b, -a), (a, 0, -b) / (0, b, -a), (-a, 0, -b), (0, -b, -a) / (0, -b, a), (b, -a, 0), (-b, -a, 0) / (0, -b, -a), (-b, -a, 0), (b, -a, 0) / (-b, a, 0), (-a, 0, b), (-a, 0, -b) / (-b, -a, 0), (-a, 0, -b), (-a, 0, b) / (b, a, 0), (a, 0,-b), (a, 0, b) / (b, -a, 0), (a, 0, b), (a, 0, -b) / (0, b, a), (-a, 0, b), (-b, a, 0) / (0, b, a), (b, a, 0), (a, 0, b) / (0, b, -a), (-b, a, 0), (-a, 0, -b) / (0, b, -a), (a, 0, -b), (b, a, 0) / (0, -b, -a), (-a, 0, -b), (-b, -a, 0) / (0, -b, -a), (b, -a, 0), (a, 0, -b) / (0, -b, a), (-b, -a, 0), (-a, 0, b) / (0, -b, a), (a, 0, b), (b, -a, 0)

The calculated coordinate points are taken to complete the figure, and different textures are applied to each face. You can reduce or increase the size of each figure by adjusting the size of the coordinates.