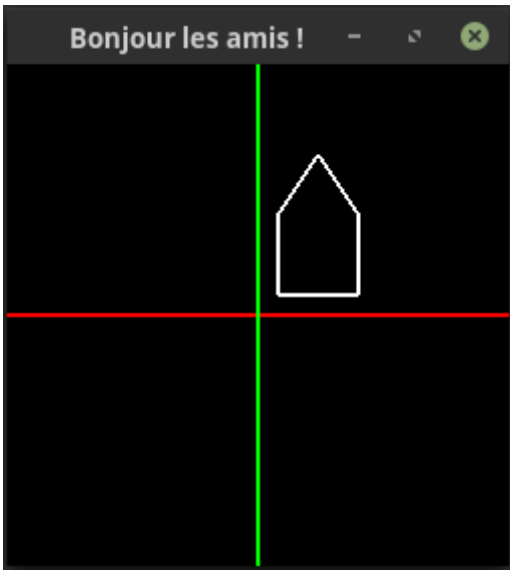
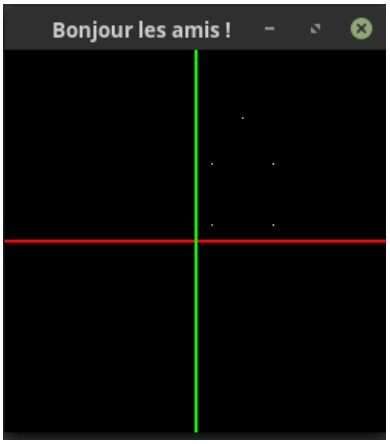
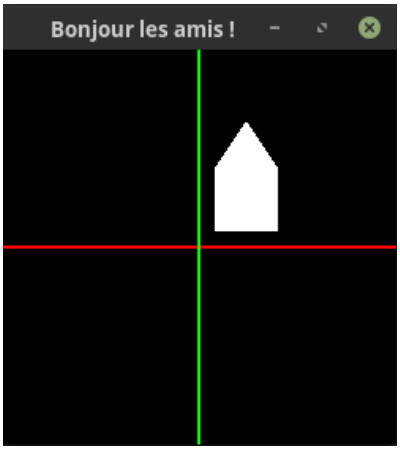


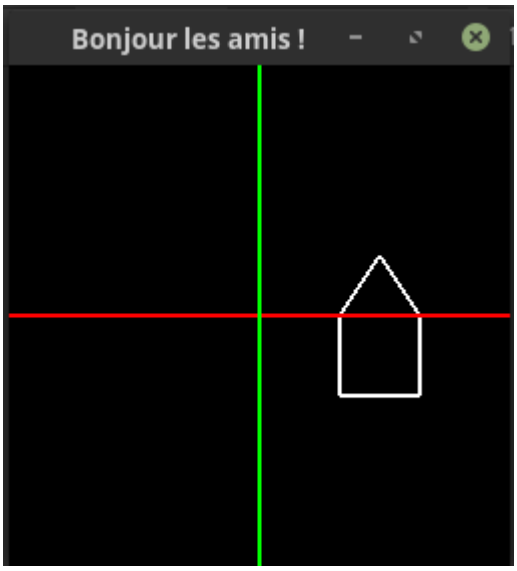

```
glClear(GL_COLOR_BUFFER_BIT);
glLineWidth(2.0);
glColor3f(1,1,1);
glBegin(GL_POLYGON); //Draws polygon
glVertex3f(1,1,0);
glVertex3f(5,1,0);
glVertex3f(5,5,0);
glVertex3f(3,8,0);
glVertex3f(1,5,0);
glEnd();
```

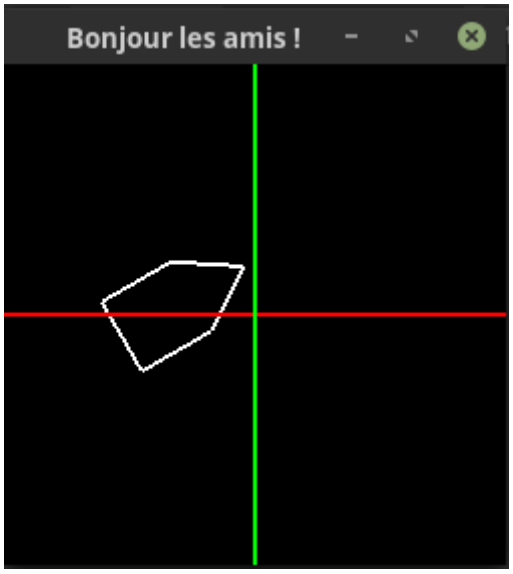


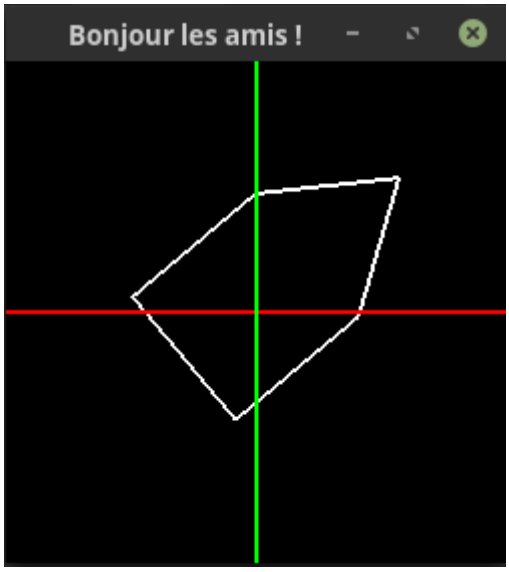



```
case 'p' : {
    glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
    display();
    break;
    }
case 'l' : {
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    display();
    break;
    }
case 'f' : {
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    display();
    break;
    }
```

The results of the transformation explained below.







So here we have some variables which will be modify every time the user press a specific key. For example if he wants to move the object to the left he will press ‘a’. Same with up (‘z’), rotate rigth(‘r’), scale up (‘y’), etc.

```
bool matrixMode = true;
int _width = 10;
int _height = 10;
int transH = 0;
int transV = 0;
int rota = 0;
float scale = 1;
```

```
case 'e' : {
    transH++;
    break;
    }
case 'a' : {
    transH--;
    break;
    }
case 'z' : {
    transV++;
    break;
    }
case 's' : {
    transV--;
    break;
    }
case 'r' : {
    rota = rota+5;
    break;
    }
case 't' : {
    rota = rota-5;
    break;
    }
case 'y' : {
    scale = scale*2;
    break;
    }
case 'u' : {
    scale = scale/2;
    break;
    }
```

This is the code which allows to display the object. When we are in matrixMode we have 3 differents matix : t the translation, s the scale and r the rotation. Thanks to the variable previously quoted the matrix will be multiply to the loaded identity matrix with the modified parameters. When using the normal mode we simply use the OpenGL methods to change the positions of the object.

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glLineWidth(2.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES); //Draws x-axis
    glVertex3f(-_width,0,0);
    glVertex3f(_width,0,0);
    glEnd();

    glColor3f(0.0, 1.0, 0.0);
    glBegin(GL_LINES); //Draws y-axis
    glVertex3f(0,-_height,0);
    glVertex3f(0,_height,0);
    glEnd();

   if(matrixMode){
        float t[16] = {
            1, 0, 0, 0,
            0, 1, 0, 0,
            0, 0, 1, 0,
            transH, transV, 0, 1
        };
        glMultMatrixf(t);

        float s[16] = {
            scale, 0, 0, 0,
            0, scale, 0, 0,
            0, 0, 0, 0,
            0, 0, 0, 1
        };
        glMultMatrixf(s);

        float r[16] = {
            cos(rota*M_PI/180), sin(rota*M_PI/180), 0, 0,
            -sin(rota*M_PI/180), cos(rota*M_PI/180), 0, 0,
            0, 0, 1, 0,
            0, 0, 0, 1
        };
        glMultMatrixf(r);

    }
    else{
        glTranslatef(transH,transV,0);
        glScalef(scale,scale,0);
        glRotatef(rota,0,0,1);
    }

    glColor3f(1,1,1);
    glBegin(GL_POLYGON); //Draws polygon
    glVertex3f(1,1,0);
    glVertex3f(5,1,0);
    glVertex3f(5,5,0);
    glVertex3f(3,8,0);
    glVertex3f(1,5,0);
    glEnd();
    glFlush();
}
```