For step1 and step2 I was done onload method which use buffer reader to load the data file from text that provided for us. These three figures are light source direction at beginning, thus, I store this three in to Vector3D which called lightSource, then read the three vector in one line that represent the coordinates of one triangle. Finally, last three figures in one line is the color of triangle then put each triangle to triangles list when load file. For triangle, I create the class called triangle which contain the three vectors of triangle and calculate normal and shading color.

```
//calculate normal
private void normal() {
    if(vectors[0] != null){
        normal = ((vectors[1].minus(vectors[0])).crossProduct(vectors[2].minus(vectors[1]))).unitVector();
        if(normal.z > 0){
            facing = false;
        }else {
            facing = true;
        }
    }
}


// compute the color shading
public Color shading(Vector3D lightSource, int[] ambitentLight, int[] incidentLight) {
    float cos = normal.dotProduct(lightSource);
    float incidentRedLight = checkIncident(incidentLight[0] * cos);
    float incidentGreenLight = checkIncident(incidentLight[1] * cos);
    float incidentBlueLight = checkIncident(incidentLight[2] * cos);
    r = checkShadingColor((int)((ambitentLight[0] + incidentRedLight)* (color.getRed()/255f)));
    g = checkShadingColor((int)((ambitentLight[0] + incidentGreenLight)* (color.getGreen()/255f)));
    b = checkShadingColor((int)((ambitentLight[0] + incidentBlueLight)* (color.getBlue()/255f)));
    Color colour = new Color(r, g, b);
    return colour;
}
```

According to the right hand rule when normal.z(cross product (v2- v1) x (v3- v2)) is negative that indicate the triangle is visible for viewer, shading need position of light source, ambient light and incident light and Uniform reflectance.

Incident light = incident light intensity * reflectance * cos.

Ambient light = ambient light intensity * reflectance

I directly put vectors to triangle and then create map which called edge list to store x, z which correspond y. Before that I calculate the startY and endY which was used to calculate x, z by x slope and z slope.

```
//put x,z to edge list
    private void putToEdgeList(Vector3D v1, Vector3D v2) {
        if (v1.y <= v2.y) {
            this.start = v1;
            this.end = v2;
        } else {
            this.start = v2;
            this.end = v1;
        }
        int startY = Math.round(start.y);
        int endY = Math.round(end.y);
        for (int y = startY; y < endY; y++) {
            float xSlope = (end.x - start.x) / (end.y - start.y);
            int x = Math.round(xSlope * (y - start.y) + start.x);
            float zSlope = (end.z - start.z) / (end.y - start.y);
            int z = Math.round(zSlope * (y-start.y) + start.z);
            if (!this.edgeList.containsKey(y)) {
                this.edgeList.put(y,new EdgeList());
            }
            this.edgeList.get(y).add(x, z);
        }
    }
```

When edgeList do not contain y then create EdgeList which use to calculate xleft, xright, zleft, zright. Integer[] coordX for store xleft and xright, Float[] coordZ for store zleft and zright. If x <

xleft, then update left side. If x > xright, then update right side.
Implement triangles in method render(), initial pixel zBuffer to infinity at first. For all y in triangle edge list and get x, z which was corresponded calculate zslope = (EL.zright(y) – EL.zleft(y)) / (EL.xright(y) – EL.xleft(y)), z = EL.zleft(y) + slope * (x – EL.xleft(y)). These two stages was done in edge list. thus, if z less then pixels.zbuffer means that z is at front then shade pixels color by using shading method in triangle class.

```
// update pixels
for(Triangle triangle : triangles){
    if(triangle.facing){
        for(Integer y : triangle.edgeList.keySet()){
            for(int x = triangle.edgeList.get(y).xLeft(); x < triangle.edgeList.get(y).xRight(); x++
                if (x > 0 && x < CANVAS_WIDTH && y > 0 && y < CANVAS_HEIGHT) {
                    if(triangle.edgeList.get(y).getZ(x) < pixels[x][y].zBuffer){
                        pixels[x][y].zBuffer = triangle.edgeList.get(y).getZ(x);
                        pixels[x][y].color = triangle.shading(
                        lightSource,getAmbientLight(),getIncidentLight());
                    }
                }
            }
        }
    }
}
```

In first and second steps, I got bug in shading method. At beginning, I only write code like red.getcolor() without divide by 255f. because I want to change it to intensity which between 0 and 1. I fix this bug by using System.out.print to find the color that I return in shading method. And there are some error when I put vector to edgelist find x,z. I still use this way to fix bug.

For steps three. I scale canvas to suit the screen and then translate canvas to center of screen. Each time I want reset the scale and minx, miny, max, maxy. Which means the minimum and maximum in all vectors in triangles. And I scale at first then translate triangle, because I composite this two behavior together. Thus, order of implement is important.

```
// reset the canvas' coordinate and scale
public void resetCoordAndScale() {
    // coordinate
    maxX = Float.MIN_VALUE;
    maxY = Float.MIN_VALUE;
    minX = Float.MAX_VALUE;
    minY = Float.MAX_VALUE;
    for (Triangle t : triangles) {
        if (t.facing) {
            for (int i = 0; i < 3; i++) {
                maxX = Math.max(t.vectors[i].x,maxX);
                minX = Math.min(t.vectors[i].x,minX);
                maxY = Math.max(t.vectors[i].y,maxY);
                minY = Math.min(t.vectors[i].y,minY);
            }
        }
    }
    // scale
    float scaleY = CANVAS_HEIGHT * 1.0f / (maxY - minY)/ 2;
    float scaleX = CANVAS_WIDTH * 1.0f / (maxX - minX)/ 2;
    if(scaleY > scaleX) {
        scale = scaleX;
    }else {
        scale = scaleY;
    }
}
```

Next creating new place, new size ,x rotate and y rotate method in triangle, because that is convenience to loop each triangle. Most methods were given me, that given transformation by input a vector in translation and scale, parameter for rotate is angle which represent by float. I

reset scale and place when run render method. Thus, it will keep the triangles position and scale at rotating. For rotating, I call the newXrotate when press key ‘A or D’. newYrotate when press key ‘ S or W’. A represent move to left with positive angle, D for right with negative angle, S for down with positive, W for down with negative. Because it obey anti-clockwise. At beginning, triangles disappeared when I doing rotate then I output the scale and position of canvas. I find the scale and position back to value that I never changed. Thus, I put reset scale and translate center each time. In addition, each time I reset edge list when I do the transformation. Because edge list used to indicate triangle. If edge list had nothing changed, that will be no helpful to reset scale, translate and rotate.

```
public void newXRotate(float changed){
    for(int i =0; i < vectors.length  && vectors[i]!=null;i++){
    Transform t = Transform.newXRotation(changed);
        vectors[i]  = t.multiply(vectors[i]);
    }
}

// rotate for y
public void newYRotate(float changed){
    for(int i =0; i < vectors.length  && vectors[i]!=null;i++){
    Transform t = Transform.newYRotation(changed);
        vectors[i]  = t.multiply(vectors[i]);
    }
}
// scale vectors
public void newSize(float x,float y, float z){
    Transform t = Transform.newScale(x, y, z);
    for(int i =0; i<vectors.length  && vectors[i]!=null ;i++){
        vectors[i] = t.multiply(vectors[i]);
    }
}


// move vectors to center
public void newPlace(float x, float y, float z){
    Transform t = Transform.newTranslation(x,y,z);
    for(int i = 0; i< vectors.length && vectors[i] != null; i++){
        vectors[i] = t.multiply(vectors[i]);
    }
```

For step four, I was done the add more light source. I create a new shading method which called new Shading, I store red , green, blue value in old shading method, using same way to express new Shading which include light source, ambient light and incident light and different from old shading. These three were decide by my own , then I calculate shading color for rgb and add red, green and blue to them respectively. Finally, I decide the way that light source will be added in press key method. If press ‘J’ light source will be added. Press ’K’ will cancel light source.

```
    if(addOneMore == false) {
        // update pixels
        for(Triangle triangle : triangles){
            if(triangle.facing){
                for(Integer y : triangle.edgeList.keySet()){
                    for(int x = triangle.edgeList.get(y).xLeft(); x < triangle.edgeList.get(y).xRight(); x++
                        if (x > 0 && x < CANVAS_WIDTH && y > 0 && y < CANVAS_HEIGHT) {
                            if(triangle.edgeList.get(y).getZ(x) < pixels[x][y].zBuffer){
                                pixels[x][y].zBuffer = triangle.edgeList.get(y).getZ(x);
                                pixels[x][y].color = triangle.shading(
                                lightSource,getAmbientLight(),getIncidentLight());
                            }
                        }
                    }
                }
            }
        }
    }else {
        // update two light source pixels
        for(Triangle triangle : triangles){
            if(triangle.facing){
                for(Integer y : triangle.edgeList.keySet()){
                    for(int x = triangle.edgeList.get(y).xLeft(); x < triangle.edgeList.get(y).xRight(); x++
                        if (x > 0 && x < CANVAS_WIDTH && y > 0 && y < CANVAS_HEIGHT) {
                            if(triangle.edgeList.get(y).getZ(x) < pixels[x][y].zBuffer){
                                pixels[x][y].zBuffer = triangle.edgeList.get(y).getZ(x);
                                pixels[x][y].color = triangle.newShading();
                            }
                        }
```