ASSIGNMENT 3

HIERARCHICAL MODELLING

SCSV 2213-01: FUNDAMENTAL OF COMPUTER GRAPHICS

LECTURER

DR. NORHAIDA BINTI MOHD SUAIB

PREPARED BY

AIMI BINTI RUSDI (B19EC0001)

AQILAH HANIM BINTI MOHD TAUFIK (B19EC0006)

Bachelor of Computer Science (Computer Graphics and Multimedia Software)

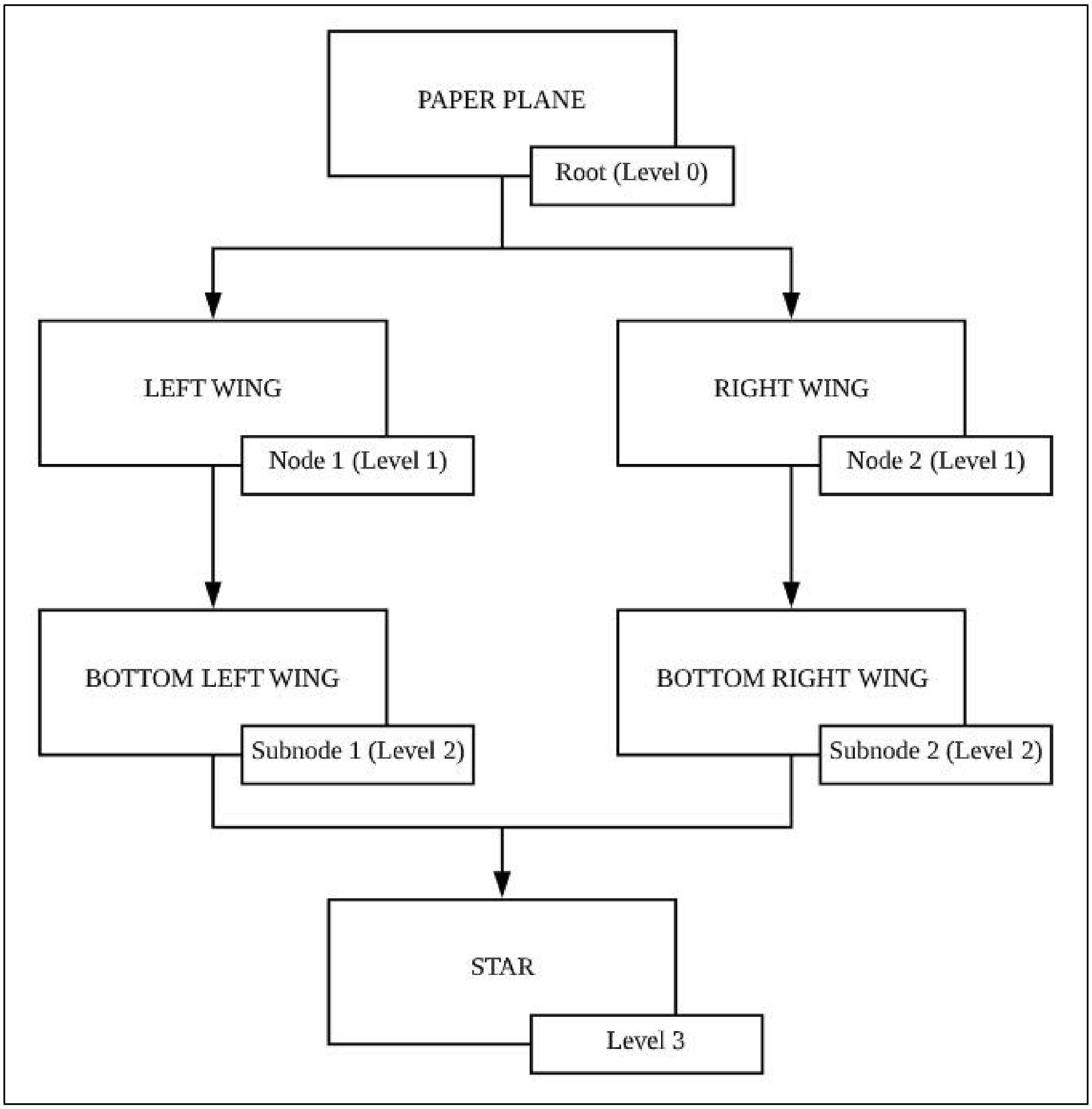
TABLE OF CONTENTS

INTRODUCTION

For our third Fundamentals in Computer Graphics assignment, we were asked to design the structure of a 2D hierarchical model. We were also asked to implement and generate the 2D hierarchical model that is designed using OpenGL functions. For our object this time, we chose to build a 2D paper plane. In OpenGL, an object is made up of geometric primitives such as triangles, quads, line segments and also points.

A primitive is made up of one or more vertices. The paper plane was built by using a set of triangles. The triangles are then positioned by each of its shapes by arranging their vertex. The coordinates of the vertices were counted one-by-one. The triangles could also be positioned by defining their vertices with respective to their own centre. Translation and rotation is then used to position the shapes at the desired locations in the display. Besides that, stars are also added to the display to make it look a little more interesting.

DEVELOPMENT OF HIERARCHICAL MODEL



EXPLAINATION OF HIERARCHICAL MODEL

Primitives can be instanced and composed to create hierarchical models using geometric transformations. Transformations can be thought of as implementing either the geometry, or the coordinate system which the object is drawn in. The idea of hierarchical modelling can be extended to an entire scene, which can include many different objects, lights and also camera positions.

Level 0

The whole paper plane. The root of the object.

Level 1

Consists of two different nodes. Node 1 is the left wing of the paper plane and Node 2 is the right wing of the paper plane.

Level 2

Consists of two different subnodes. Subnode 1 is the bottom left wing of the paper plane and Subnode 2 is the bottom right wing of the paper plane.

Level 3

The stars located on the background of the display.

USER GUIDE

Keyboard	Function
W	Transformation of the root of the diagram to move upwards

	`case 'w':`
	`case 'W':`
	`    rooty++;`
	`    glutPostRedisplay();`
	`    break;`

S	Transformation of the root of the diagram to move downwards

	`case 's':`
	`case 'S':`
	`    rooty--;`
	`    glutPostRedisplay();`
	`    break;`

A	Transformation of the root of the diagram to move anti-clockwise

	`case 'a':`
	`case 'A':`
	`    node = (node + 5) % 360;`
	`    glutPostRedisplay();`
	`    break;`

D	Transformation of the root of the diagram to move clockwise at one point

	`case 'd':`
	`case 'D':`
	`    node = (node - 5) % 360;`
	`    glutPostRedisplay();`
	`    break;`

Q	Transformation of the nodes of the diagram to move anti-clockwise

	`case 'q':`
	`case 'Q':`
	`    root = (root + 5) % 360;`
	`    glutPostRedisplay();`
	`    break;`

E	Transformation of the nodes of the diagram to move clockwise

	`case 'e':`
	`case 'E':`
	`    root = (root - 5) % 360;`
	`    glutPostRedisplay();`
	`    break;`

Z	Transformation of the subnodes of the diagram to move anti-clockwise

	`case 'z':`
	`case 'Z':`
	`    subnode = (subnode + 5) % 360;`
	`    glutPostRedisplay();`
	`    break;`

X	Transformation of the subnodes of the diagram to move clockwise

	`case 'x':`
	`case 'X':`
	`    subnode = (subnode - 5) % 360;`
	`    glutPostRedisplay();`
	`    break;`

C	Transformation of the root of the diagram to move clockwise

	`case 'c':`
	`case 'C':`
	`    subnode2 = (subnode2 + 5) % 360;`
	`    glutPostRedisplay();`
	`    break;`

EXPLANATION OF NODE

Node 1

► Left Wing

```
//node 1 coding = LEFT WING
	glPushMatrix();
	//position node 1
	glTranslatef(0.0, 0.0, 0.0);
	//node 1 transformation
	glTranslatef(0.0, 0.0, 0.0);
	glRotatef((GLfloat)node, 0.0, 0.0, 1.0);
	glTranslatef(-0.2, 0.0, 0.0);
	glColor3f(0, 0, 0);  //black
	leftwing();
	//on top layer
	glScalef(0.95, 0.95, 0.00);
	glColor3f(1, 1, 1);  //white
	leftwing();
	//LEFT WING
	//glPushMatrix();
	leftwing();
	glPopMatrix(); //end of node 1
```

Node 1 is the left wing of the plane. There are 2 separate layers of the node. The top layer is the white layer, `glColor3f(1, 1, 1)`, and the bottom layer is the black layer, `glColor3f(0, 0, 0)`. Both layers are programmed to transform at the same time when a specific keyboard button is pressed. The transformations involved are `glTranslatef(-0.2, 0.0, 0.0)` which is translation, and `glRotatef((GLfloat)node, 0.0, 0.0, 1.0)`, rotation.

Node 2

► Right Wing

```
//node 2 coding = RIGHT WING
       //glPushMatrix(); //effects the whole diagram
       //position node 2
       glTranslatef(0.0, 0.0, 0.0);
       //node 2 transformation
       glTranslatef(0.0, 0.0, 0.0);
       glRotatef((GLfloat)node, 0.0, 0.0, 1.0);
       glTranslatef(-0.2, 0.0, 0.0);
       glColor3f(0, 0, 0);  //black
       rightwing();
       //on top layer
       glScalef(0.95, 0.95, 0.00);
       glColor3f(1, 1, 1);  //white
       rightwing();
       //RIGHT WING
       glPushMatrix();
       rightwing();
       glPopMatrix(); //end of node 2
```

Node 2 is the right wing of the plane. There are also 2 separate layers of the node. The top layer is the white layer, `glColor3f(1, 1, 1)`, and the bottom layer is the black layer, `glColor3f(0, 0, 0)`. Both layers are programmed to transform at the same time when a specific keyboard button is pressed. The transformations involved are `glTranslatef(-0.2, 0.0, 0.0)` which is translation, and `glRotatef((GLfloat)node, 0.0, 0.0, 1.0)`, rotation.

EXPLANATION OF SUBNODE

Subnode 1

► Bottom Left Wing

```
//subnode 1 coding = BOTTOM LEFT WING
       //glPushMatrix();
       //position subnode 1
       glTranslatef(0.2, 0.0, 0.0);
       //subnode 1 transformation
       glTranslatef(0.0, 0.0, 0.0);
       glRotatef((GLfloat)subnode, 0.0, 0.0, 1.0);
       glTranslatef(-0.2, 0.0, 0.0);
       glColor3f(0, 0, 0);  //black
       bottomleft();
       //on top layer
       glScalef(0.95, 0.95, 0.00);
       glColor3f(1, 1, 1);  //white
       bottomleft();
       //BOTTOM LEFT
       glPushMatrix();
       bottomleft();
       glPopMatrix(); //end of subnode 1
```

Subnode 1 is the bottom left wing of the plane. Similarly like Node 1 and Node 2, there are 2 separate layers of the node. The top layer is the white layer, `glColor3f(1, 1, 1)`, and the bottom layer is the black layer, `glColor3f(0, 0, 0)`. Both layers are programmed to transform at the same time when a specific keyboard button is pressed. The transformations involved are `glTranslatef(-0.2, 0.0, 0.0)` which is translation, and `glRotatef((GLfloat)subnode, 0.0, 0.0, 1.0)`, rotation.

Subnode 2

► Bottom Right Wing

```
//subnode 2 coding = BOTTOM RIGHT WING
       //glPushMatrix();
       //position subnode 2
       glTranslatef(0.2, 0.0, 0.0);
       //subnode 2 transformation
       glTranslatef(0.0, 0.0, 0.0);
       glRotatef((GLfloat)subnode, 0.0, 0.0, 1.0);
       glTranslatef(-0.2, 0.0, 0.0);
       glColor3f(0, 0, 0);  //black
       bottomright();
       //on top layer
       glScalef(0.95, 0.95, 0.00);
       glColor3f(1, 1, 1);  //white
       bottomright();
       //BOTTOM RIGHT
       glPushMatrix();
       bottomright();
       glPopMatrix(); //end of subnode 2
```

Subnode 2 is the bottom right wing of the plane. Similarly like Node 1, Node 2 and Subnode 1, there are 2 separate layers of the node. The top layer is the white layer, `glColor3f(1, 1, 1)`, and the bottom layer is the black layer, `glColor3f(0, 0, 0)`. Both layers are programmed to transform at the same time when a specific keyboard button is pressed. The transformations involved are `glTranslatef(-0.2, 0.0, 0.0)` which is translation, and `glRotatef((GLfloat)subnode, 0.0, 0.0, 1.0)`, rotation.

Star Subnode

- ► Subnode2 = 0;

```
void star()

{

	glBegin(GL_POLYGON);//Star1Square

	glColor3f(0.90f, 0.91f, 0.98f);

	glVertex2f(-0.8, 0.75);//a

	glVertex2f(-0.85, 0.75);//b

	glVertex2f(-0.85, 0.7);//c

	glVertex2f(-0.8, 0.7);//d

	glEnd();


	glBegin(GL_POLYGON);//Star1RightTriangle

	glVertex2f(-0.7, 0.725);//e

	glVertex2f(-0.8, 0.75);//a

	glVertex2f(-0.8, 0.7);//d

	glEnd();


	glBegin(GL_POLYGON);//Star1TopTriangle

	glVertex2f(-0.825, 0.85);//f

	glVertex2f(-0.85, 0.75);//b

	glVertex2f(-0.8, 0.75);//a

	glEnd();


	glBegin(GL_POLYGON);//Star1LeftTriangle
```

```
        glVertex2f(-0.95, 0.725);//g

        glVertex2f(-0.85, 0.75);//b

        glVertex2f(-0.85, 0.7);//c

        glEnd();


        glBegin(GL_POLYGON);//Star1BottomTriangle


        glVertex2f(-0.825, 0.6);//h

        glVertex2f(-0.8, 0.7);//d

        glVertex2f(-0.85, 0.7);//c

        glEnd();

}
```

The star subnote is located at Level 3 of the hierarchical model. It represents the stars in the background of the display. The colour of the star is `glColor3f(0.90f, 0.91f, 0.98f),` a colour almost similar to white. The star consists of 4 triangles and a square.

HIERARCHICAL MODELLING FULL CODING

```
#include <GL/glut.h>

#include <stdlib.h>


static int root = 0, node = 0, rootx = 0, rooty = 0, nodex = 0, nodey = 0, subnode = 0,
subnode2 = 0;


void init(void)

{

	glClearColor(0.18, 0.18, 0.31, 0.0); //midnight blue

	glShadeModel(GL_FLAT);

}


//---------------------------------OBJECTS---------------------------------------------


//lukis paksi utama

void lukisPaksi()

{

	glColor3f(1, 1, 1);

	glBegin(GL_LINES);

	glVertex3i(-100, 0, 0);

	glVertex3i(100, 0, 0);

	glVertex3i(0, -100, 0);

	glVertex3i(0, 100, 0);

	glVertex3i(0, 0, -100);

	glVertex3i(0, 0, 100);

	glEnd();

}

void star()

{

	glBegin(GL_POLYGON);//Star1Square

	glColor3f(0.90f, 0.91f, 0.98f);
	glVertex2f(-0.8, 0.75);//a
	glVertex2f(-0.85, 0.75);//b
	glVertex2f(-0.85, 0.7);//c
	glVertex2f(-0.8, 0.7);//d
	glEnd();

	glBegin(GL_POLYGON);//Star1RightTriangle
	glVertex2f(-0.7, 0.725);//e
	glVertex2f(-0.8, 0.75);//a
	glVertex2f(-0.8, 0.7);//d
	glEnd();

	glBegin(GL_POLYGON);//Star1TopTriangle
	glVertex2f(-0.825, 0.85);//f
	glVertex2f(-0.85, 0.75);//b
	glVertex2f(-0.8, 0.75);//a
	glEnd();

	glBegin(GL_POLYGON);//Star1LeftTriangle
	glVertex2f(-0.95, 0.725);//g
	glVertex2f(-0.85, 0.75);//b
	glVertex2f(-0.85, 0.7);//c
	glEnd();

	glBegin(GL_POLYGON);//Star1BottomTriangle
	glVertex2f(-0.825, 0.6);//h
	glVertex2f(-0.8, 0.7);//d
	glVertex2f(-0.85, 0.7);//c
	glEnd();
}
void leftwing()
{
	glBegin(GL_TRIANGLE_STRIP);
	glVertex2f(-3.7f, -0.92f);
	glVertex2f(1.0f, 2.0f);
	glVertex2f(-2.0f, -0.62f);
	glEnd();

	glBegin(GL_LINES);
	glVertex2f(-3.7f, -0.92f);
	glVertex2f(1.0f, 2.0f);
	glVertex2f(-2.0f, -0.62f);
	glEnd();
}
void rightwing()
{
	glBegin(GL_TRIANGLE_STRIP);
	glVertex2f(-0.34f, -0.58f);
	glVertex2f(1.0f, 2.0f);
	glVertex2f(0.66f, -0.94f);
	glEnd();

	glBegin(GL_LINES);
	glVertex2f(-0.34f, -0.58f);
	glVertex2f(1.0f, 2.0f);
	glVertex2f(0.66f, -0.94f);
	glEnd();
}
void bottomleft()
{
	glBegin(GL_TRIANGLE_STRIP);
	glVertex2f(-2.0f, -0.62f);
	glVertex2f(1.0f, 2.0f);
	glVertex2f(-1.60f, -1.22f);
	glEnd();

	glBegin(GL_LINES);
	glVertex2f(-2.0f, -0.62f);
	glVertex2f(1.0f, 2.0f);
	glVertex2f(-1.60f, -1.22f);
	glEnd();
}
void bottomright()
{
	glBegin(GL_TRIANGLE_STRIP);
	glVertex2f(-0.36f, -0.56f);
	glVertex2f(1.0f, 2.0f);
	glVertex2f(-1.60f, -1.22f);
	glEnd();

	glBegin(GL_LINES);
	glVertex2f(-0.34f, -0.58f);
	glVertex2f(1.0f, 2.0f);
	glVertex2f(-1.60f, -1.22f);
	glEnd();
}
//plane coding
void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT);

	//glRotatef((GLfloat)subnode, 0.0, 0.0, 1.0);
	glPushMatrix();
	glTranslatef(1.0, 0.0, 0.0);
	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
	glTranslatef(0.0, 0.0, 0.0);
	star();

	glPopMatrix();


	glPushMatrix();

	glTranslatef(2.0, 2.0, 0.0);

	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

	glTranslatef(0.0, 0.0, 0.0);

	star();

	glPopMatrix();


	glPushMatrix();

	glTranslatef(4.0, 2.0, 0.0);

	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

	glTranslatef(0.0, 0.0, 0.0);

	star();

	glPopMatrix();


	glPushMatrix();

	glTranslatef(6.0, 2.0, 0.0);

	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

	glTranslatef(0.0, 0.0, 0.0);

	star();

	glPopMatrix();


	glPushMatrix();

	glTranslatef(0.0, 2.0, 0.0);

	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

	glTranslatef(0.0, 0.0, 0.0);

	star();

	glPopMatrix();


	glPushMatrix();
```

```
        glTranslatef(-2.0, 2.0, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-4.0, 2.0, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-6.0, -2.0, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(3.0, 0.0, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(5.0, 0.0, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
```

```
	glPopMatrix();


	glPushMatrix();
	glTranslatef(-1.0, 0.0, 0.0);
	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
	glTranslatef(0.0, 0.0, 0.0);
	star();
	glPopMatrix();


	glPushMatrix();
	glTranslatef(-3.0, 0.0, 0.0);
	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
	glTranslatef(0.0, 0.0, 0.0);
	star();
	glPopMatrix();


	glPushMatrix();
	glTranslatef(-5.0, 0.0, 0.0);
	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
	glTranslatef(0.0, 0.0, 0.0);
	star();
	glPopMatrix();


	glPushMatrix();
	glTranslatef(2.0, -2.0, 0.0);
	glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
	glTranslatef(0.0, 0.0, 0.0);
	star();
	glPopMatrix();


	glPushMatrix();
	glTranslatef(4.0, -2.0, 0.0);
```

```
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

        glTranslatef(0.0, 0.0, 0.0);

        star();

        glPopMatrix();


        glPushMatrix();

        glTranslatef(6.0, -2.0, 0.0);

        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

        glTranslatef(0.0, 0.0, 0.0);

        star();

        glPopMatrix();


        glPushMatrix();

        glTranslatef(0.0, -2.0, 0.0);

        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

        glTranslatef(0.0, 0.0, 0.0);

        star();

        glPopMatrix();


        glPushMatrix();

        glTranslatef(-2.0, -2.0, 0.0);

        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

        glTranslatef(0.0, 0.0, 0.0);

        star();

        glPopMatrix();


        glPushMatrix();

        glTranslatef(-4.0, -2.0, 0.0);

        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);

        glTranslatef(0.0, 0.0, 0.0);

        star();

        glPopMatrix();
        glPushMatrix();
        glTranslatef(-6.0, -2.0, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(3.0, -3.5, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(5.0, -3.5, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-1.0, -3.5, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
        glTranslatef(0.0, 0.0, 0.0);
        star();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-3.0, -3.5, 0.0);
        glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
```

```
glTranslatef(0.0, 0.0, 0.0);
star();
glPopMatrix();

glPushMatrix();
glTranslatef(-5.0, -3.5, 0.0);
glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
glTranslatef(0.0, 0.0, 0.0);
star();
glPopMatrix();

glPushMatrix();
glTranslatef(1.0, -3.5, 0.0);
glRotatef((GLfloat)subnode2, 0.0, 0.0, 1.0);
glTranslatef(0.0, 0.0, 0.0);
star();
 glPopMatrix();

 //lukisPaksi();
glPushMatrix(); //start root push
//root position of the diagram
glTranslatef(0.0, 0.0, 0.0);

//transformation root
glRotatef((GLfloat)root, 0.0, 0.0, 1.0);
glTranslatef((GLfloat)rootx, (GLfloat)rooty, 0.0);

//line
glColor3f(1, 1, 1);
glVertex2f(-1.58f, -1.24f);
glVertex2f(1.0f, 2.0f);
glVertex2f(-0.34f, -0.58f);
```

```
//node 1 coding = LEFT WING
glPushMatrix();
//position node 1
glTranslatef(0.0, 0.0, 0.0);
//node 1 transformation
glTranslatef(0.0, 0.0, 0.0);
glRotatef((GLfloat)node, 0.0, 0.0, 1.0);
glTranslatef(-0.2, 0.0, 0.0);
glColor3f(0, 0, 0);  //black
leftwing();
//on top layer
glScalef(0.95, 0.95, 0.00);
glColor3f(1, 1, 1);  //white
leftwing();
//LEFT WING
//glPushMatrix();
leftwing();
glPopMatrix(); //end of node 1

//node 2 coding = RIGHT WING
//glPushMatrix(); //effects the whole diagram
//position node 2
glTranslatef(0.0, 0.0, 0.0);
//node 2 transformation
glTranslatef(0.0, 0.0, 0.0);
glRotatef((GLfloat)node, 0.0, 0.0, 1.0);
glTranslatef(-0.2, 0.0, 0.0);
glColor3f(0, 0, 0);  //black
rightwing();
//on top layer
glScalef(0.95, 0.95, 0.00);
```

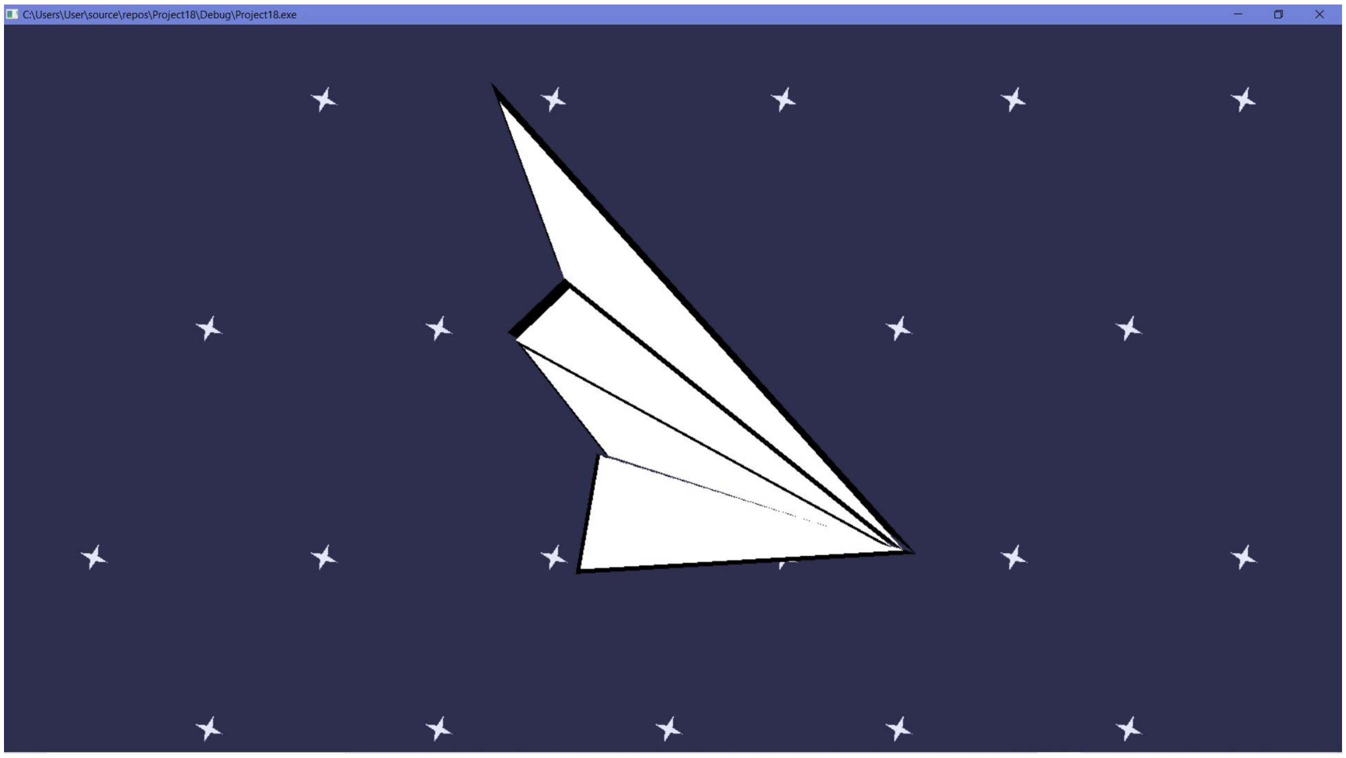
```
	glColor3f(1, 1, 1);  //white
	rightwing();
	//RIGHT WING
	glPushMatrix();
	rightwing();
	glPopMatrix(); //end of node 2

	//subnode 1 coding = BOTTOM LEFT WING
	//glPushMatrix();
	//position subnode 1
	glTranslatef(0.2, 0.0, 0.0);
	//subnode 1 transformation
	glTranslatef(0.0, 0.0, 0.0);
	glRotatef((GLfloat)subnode, 0.0, 0.0, 1.0);
	glTranslatef(-0.2, 0.0, 0.0);
	glColor3f(0, 0, 0);  //black
	bottomleft();
	//on top layer
	glScalef(0.95, 0.95, 0.00);
	glColor3f(1, 1, 1);  //white
	bottomleft();
	//BOTTOM LEFT
	glPushMatrix();
	bottomleft();
	glPopMatrix(); //end of subnode 1

	//subnode 2 coding = BOTTOM RIGHT WING
	//glPushMatrix();
	//position subnode 2
	glTranslatef(0.2, 0.0, 0.0);
	//subnode 2 transformation
	glTranslatef(0.0, 0.0, 0.0);
	glRotatef((GLfloat)subnode, 0.0, 0.0, 1.0);

	glTranslatef(-0.2, 0.0, 0.0);

	glColor3f(0, 0, 0);  //black

	bottomright();

	//on top layer

	glScalef(0.95, 0.95, 0.00);

	glColor3f(1, 1, 1);  //white

	bottomright();

	//BOTTOM RIGHT

	glPushMatrix();

	bottomright();

	glPopMatrix(); //end of subnode 2


	glPopMatrix(); //end root


	glutSwapBuffers();

}

void reshape(int w, int h)

{

	glViewport(0, 0, (GLsizei)w, (GLsizei)h);

	glMatrixMode(GL_PROJECTION);

	glLoadIdentity();

	gluPerspective(65.0, (GLfloat)w / (GLfloat)h, 1.0, 20.0);

	glMatrixMode(GL_MODELVIEW);

	glLoadIdentity();

	glTranslatef(0.0, 0.0, -5.0);

}


void keyboard(unsigned char key, int x, int y)


{

	switch (key) {
```

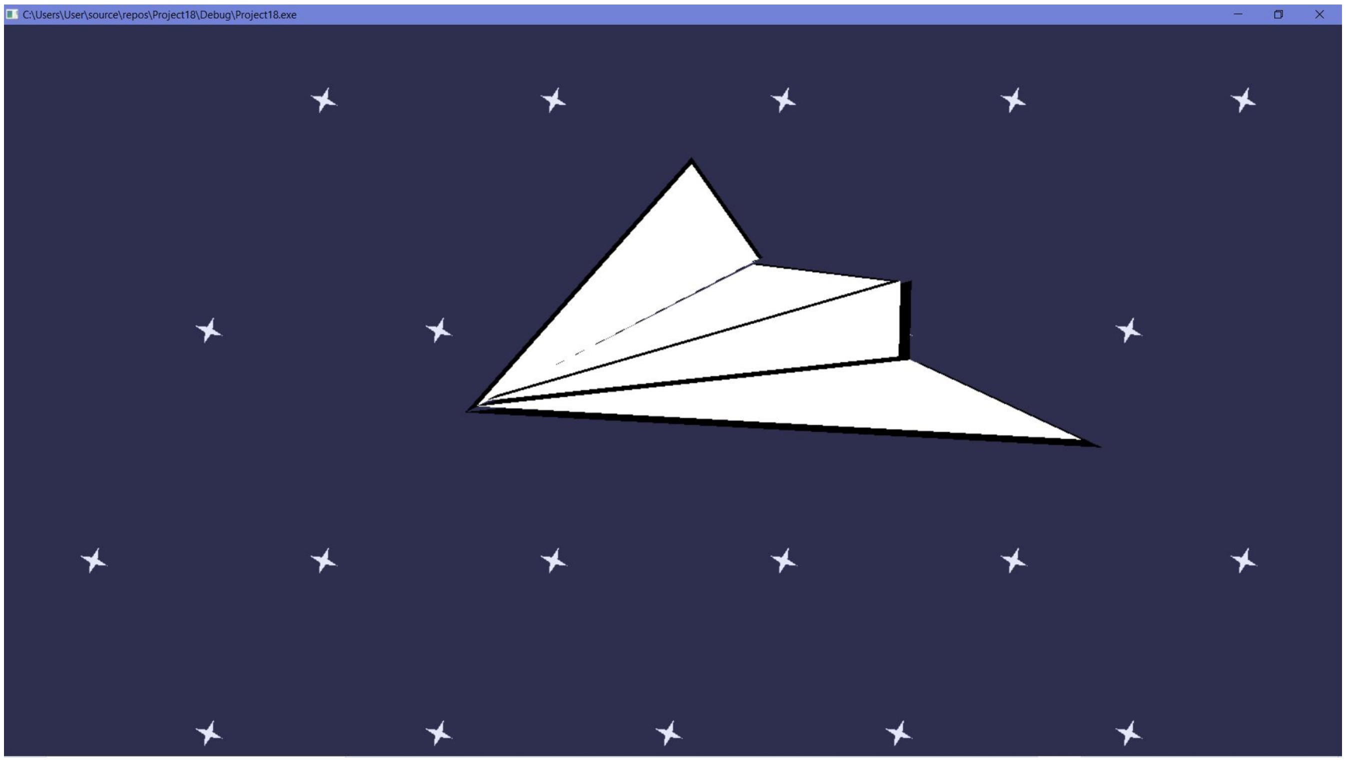
```
	case 'q':
	case 'Q':
		root = (root + 5) % 360;
		glutPostRedisplay();
		break;

	case 'e':
	case 'E':
		root = (root - 5) % 360;
		glutPostRedisplay();
		break;

	case 'a':
	case 'A':
		node = (node + 5) % 360;
		glutPostRedisplay();
		break;

	case 'd':
	case 'D':
		node = (node - 5) % 360;
		glutPostRedisplay();
		break;

	case 'z':
	case 'Z':
		subnode = (subnode + 5) % 360;
		glutPostRedisplay();
		break;

	case 'x':
	case 'X':
```

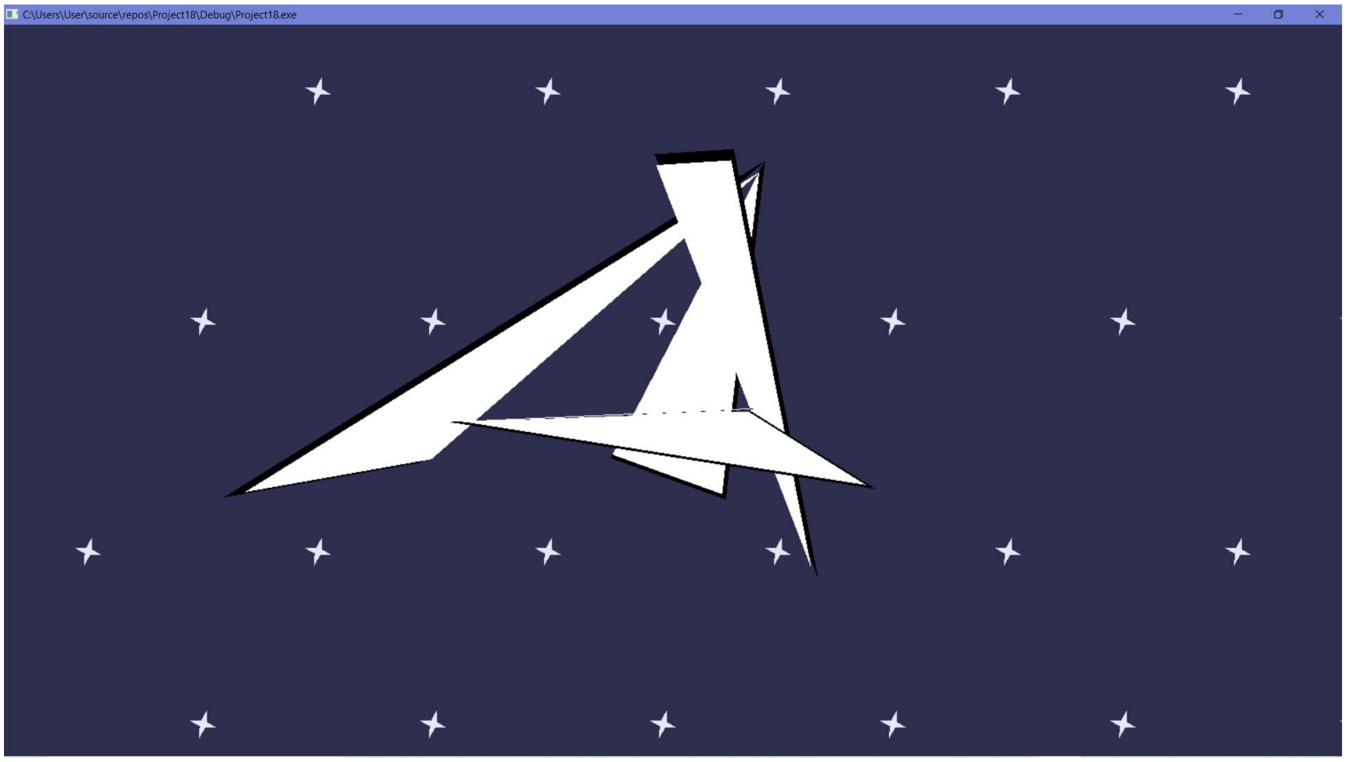
```
                subnode = (subnode - 5) % 360;

                glutPostRedisplay();

                break;


        case 'c':

        case 'C':

                subnode2 = (subnode2 + 5) % 360;

                glutPostRedisplay();

                break;


        case 'w':

        case 'W':

                rooty++;

                glutPostRedisplay();

                break;


        case 's':

        case 'S':

                rooty--;

                glutPostRedisplay();

                break;


        case 27:

                exit(0);

                break;


        default:

                break;

        }

        }

void specialKeys(int key, int x, int y) {

        switch (key) {
```

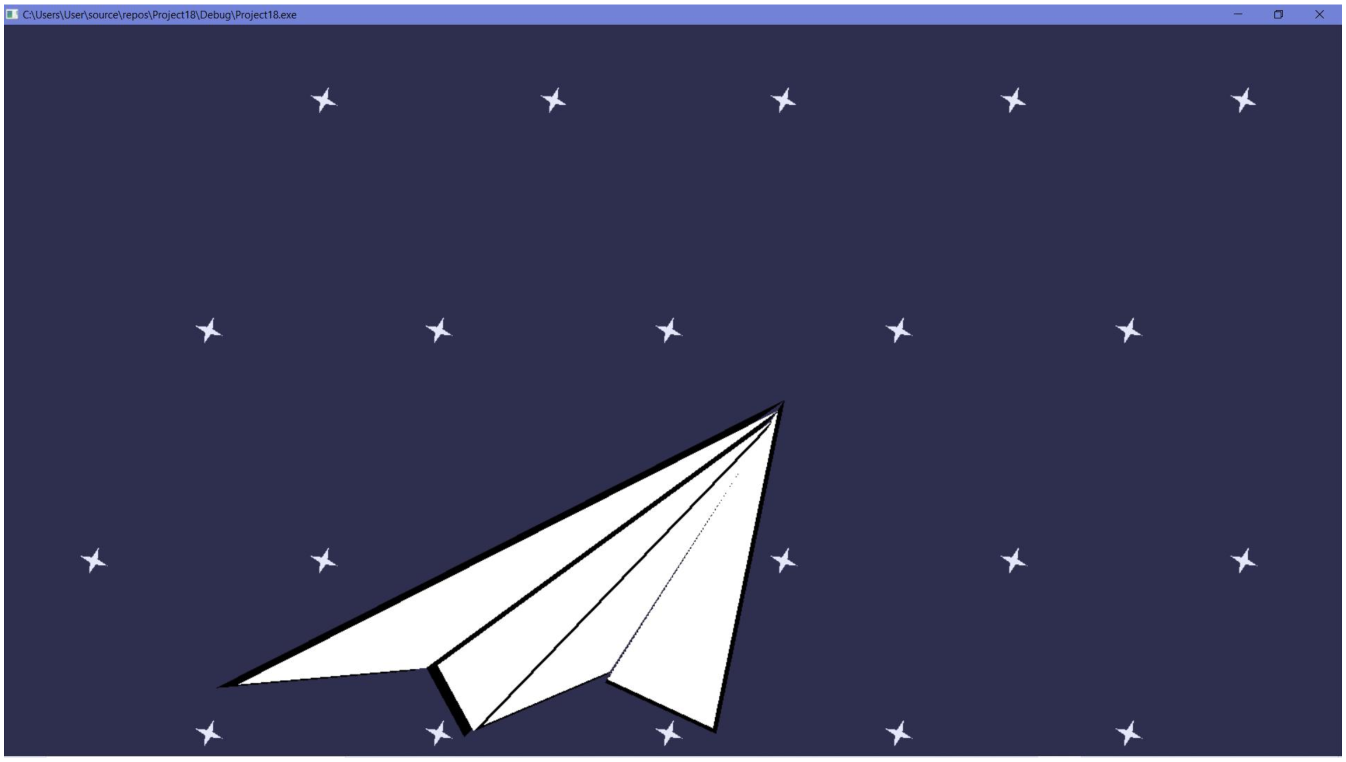
```
	case GLUT_KEY_UP: rooty += 0.1;

		glutPostRedisplay();

		break;

	case GLUT_KEY_DOWN: rooty -= 0.1;

		glutPostRedisplay();

		break;

	case GLUT_KEY_LEFT: rootx -= 0.1;

		glutPostRedisplay();

		break;

	case GLUT_KEY_RIGHT: rootx += 0.1;

		glutPostRedisplay();

		break;

	}

}

int main(int argc, char** argv)

{

	glutInit(&argc, argv);

	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

	glutInitWindowSize(1000, 500);

	glutInitWindowPosition(100, 100);

	glutCreateWindow(argv[0]);

	init();

	glutDisplayFunc(display);

	glutReshapeFunc(reshape);

	glutKeyboardFunc(keyboard);

	glutSpecialFunc(specialKeys);

	glutMainLoop();

	return 0;

	}
```

HIERARCHICAL MODELLING OUTPUT






C:\Users\User\source\repos\Project18\Debug\Project18.exe


C:\Users\User\source\repos\Project18\Debug\Project18.exe

REFERENCES

CS2401 Computer Graphics. (n.d.). Unit 1 - 2D Primitives. Retrieved from: http://fmcet.in/ CSE /CS2401_uw.pdf

Dinesh. (2016). *How to Draw Geometric Primitives in OpenGL using Visual Studio 2015* [Youtube]. Retrieved from: https://www.youtube.com/watch?v=xD60f91nKFw

Hearn, Baker & Carithers. (2014). *Computer Graphics with OpenGL.* United States of America. Pearson Education Limited.