UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FINAL PROJECT

3D 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)

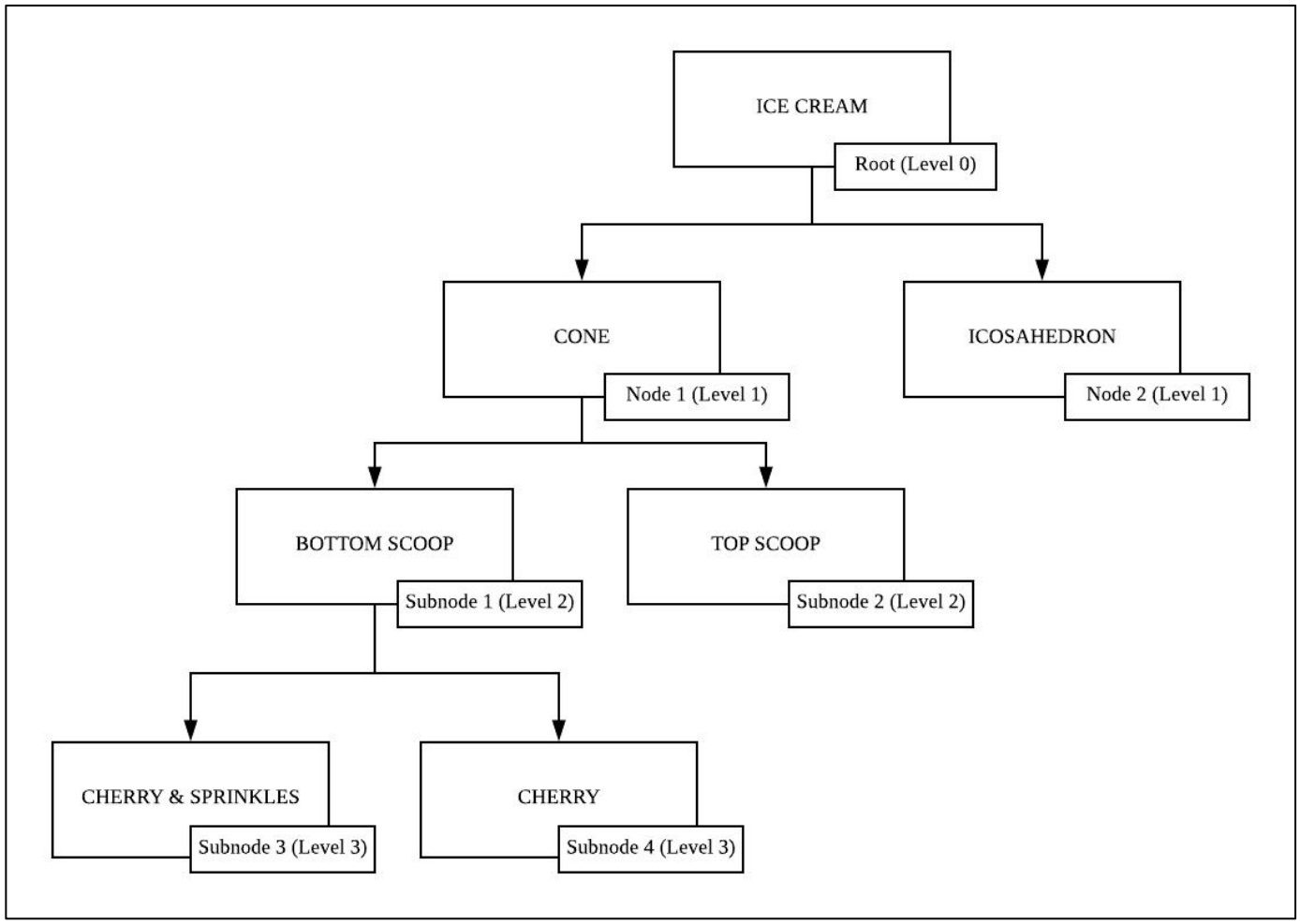
TABLE OF CONTENTS

INTRODUCTION

For our Fundamentals in Computer Graphics Final Project, we were asked to design the structure of a 3D hierarchical model. We were also asked to implement VSD and generate the 3D hierarchical model that is designed using OpenGL functions. For our object this time, we chose to build a 3D Ice cream. In this report, explanations on 3D Hierarchical Model, 3D transformations, camera, viewing and projection and lighting will be explained.

DEVELOPMENT OF 3D HIERARCHICAL MODEL

Hierarchical Table



PROPER 3D TRANSFORMATION

Explanation of each transformations according to the respective nodes will be listed down below.

1. **Root**

   Transformation:

   - Continuous movement of translation at the x axis by moving left and right.
   - Continuous movement of translation at the y axis by moving up and down.

2. **Node – Cone and Icosahedron**

   Transformation:

   - Continuous clockwise rotation about the y axis.
   - Continous anti-clockwise rotation about the y axis.

3. **Subnode – Bottom Scoop, Top Scoop, Cherry and Sprinkles, Cherry.**

   - **Subnode 1 – Bottom Scoop**

     Transformation**:**

     - Continuous clockwise rotation about the y axis.
     - Continuous anti clockwise rotation about the y axis.
     - Continuous translation at the x axis by moving right and left.

   - **Subnode 2 – Top Scoop**

     Transformation:

     - Continuous clockwise rotation about the y axis.
     - Continuous anti clockwise rotation about the y axis.
     - Continuous translation at the x axis by moving right and left.

   - **Subnode 3 – Cherry and Sprinkles**

     Transformation:

     - Continuous translation at the x axis by moving right and left.

   - **Subnode 4 – Cherry**

     Transformation:

     - Continuous translation at the x axis by moving right and left.

LIGHTING AND SHADING

Lighting plays a big role in a 3D Model as it shows the suitable shine that occurs on an object to show the quality of a three dimensional. The lighting for this particular project which is the Ice Cream was done to shine on all the objects which are the cone, top scoop, bottom scoop, cherry and sprinkles. It was activated by enable one light source known as `(GL_LIGHT0);` that can be found in the init() function as below.

```
void init(void)
{
	glEnable(GL_COLOR_MATERIAL);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
	glEnable(GL_DEPTH_TEST);
}
```

By enabling the following light source, a certain shine will be produced reflecting on the ice cream. If the following light source is disabled, the colour of the following object will not have a light source and can be seen as a 2D model such as below.

All of the functions are enabled so that the object will require the right amount of light and shade as needed by including `glShadeModel(GL_SMOOTH);` to show the smoothness of Gouraud shading and not a simple flat shading.

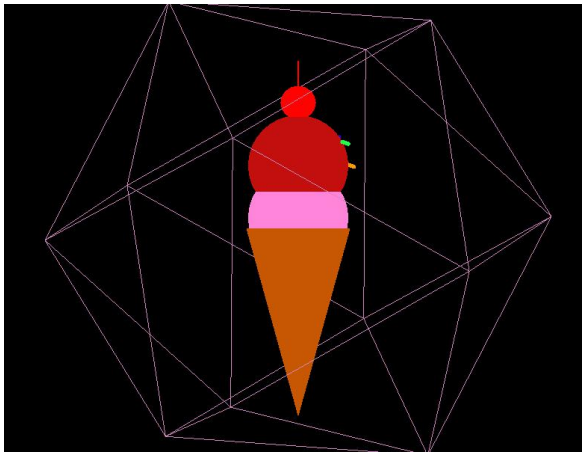


Figure 1. Disable the Light0 function

(GL_LIGHT0);

LIGHT0 controls all properties of lightings known which are specular, diffuse, ambient and shininess. This is known as Phong lighting as it has the combination of the three main lightings which are ambience, diffuse and specular.

```
void init(void)
{
	GLfloat mat_specular[] = { 0.65, 0.05, 0.05, 0.0 };
	//marroon red colour
	GLfloat mat_diffuse[] = { 1.0, 0.05, 0.5 };
	//colour of lighting - this makes the lighting follow specular
	GLfloat mat_ambient[] = { 0.65, 0.05, 0.05, 0.0 };
	GLfloat mat_shininess[] = { 10.0 };
	GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
	//x,y,z and w position of light
	glClearColor(0.0, 0.0, 0.0, 0.0);
	/* clear the background colour to black */
	glShadeModel(GL_SMOOTH);

	glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
	glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
	glColorMaterial(GL_FRONT, GL_EMISSION);
	glLightfv(GL_LIGHT0, GL_POSITION, light_position);
	glLightModelfv(GL_LIGHT_MODEL_AMBIENT, mat_ambient);
}
```

It is shown here that the specular is set to a maroon red colour which will affect the overall lighting of the object but with 0 intensity. This will give a slight red shine light to the object rather than a normal white light. The diffuse lighting is set to a red greenish lighting with an intensity of 0.5 as we do not want to lose the colour of the object due to the lighting. The ambient like then is set similar as the specular so it will shine within a colour and not by many shades. The light position is set at the x and y axis which will reflect on the whole body of the material.

All is set to reflect on the object by the function called `glLightfv` which enables the LIGHT0 to shine at the light position that was set. The material mostly has the lighting set by the ambient lighting as declared in the function which gives out the shiny colour of red at `glLightModelfv`.

The rest are activated at the front side of the image which shows only one direction of light is set by the function of `GL_FRONT();`.

CAMERA, VIEW & PROJECTION

View & Projection

```
void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	if (w <= h)
		glOrtho(-1.5, 1.5, -1.5*(GLfloat)h / (GLfloat)w, 1.5*(GLfloat)h /
		(GLfloat)w, -10.0, 10.0);
	else
		glOrtho(-1.5*(GLfloat)w / (GLfloat)h, 1.5*(GLfloat)w / (GLfloat)h, -1.5,
		1.5, -10.0, 10.0);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}
```

First, in the function of reshape (void), the view of the whole program is set by using `glOrtho()` as we chose to control the views manually. The settings are made as projection and the volume as -1.5 as the distance of the axis is set. The ratio of width and height of the front and back is set to a -10.0 and 10.0 calculation of width divide by height at a certain angle if the volume of width is smaller than height.

Projection is also used by declaring `glMatrixMode(GL_PROJECTION);` to switch the projection matrix so the object or material can be tweaked and viewed in a certain way. To keep on changing the materials after executing, `glMatrixMode(GL_MODELVIEW);` works to switch the projection view back to its model view. In order to reverse or reset the matrix to identification or identity matrix, we use `glLoadIdentity();` to avoid any sudden deletions.

Camera Position

For camera, we use `gluLookAt(xkey, ykey, z, 0.0, ycentre, xcentre, 0.0, 1.0, 0.0);` and change the settings at xkey, ykey, z, ycentre and xcentre to manipulate the angle of the camera. Xkey, ykey and z shows the location of camera while ycentre and xcentre plays with the orientation of the camera. The values can be changed in the display function ().

EXPLANATION OF NODE

Node 1

► Cone

```
//cone - node 1
glPushMatrix();
glColor3d(0.65, 0.33, 0.0);
glTranslatef(0.0, 0.0, 0.0);
glRotatef(90, 0, 1, 0);
glRotatef(90, 1, 0, 0);
glRotatef((GLfloat)node, 0, 1, 0);
glTranslatef(0.0, 0.0, 0.0);
Cone();
glPopMatrix();
//cone
```

Node 1 is the cone. The object is scaled and translated to fit the screen. It is programmed to transform at the same time when a specific keyboard button is pressed which is for root and node. The transformations involved are `glTranslatef(0.0, 0.0, 0.0)` which is translation, and `glRotatef((GLfloat)node, 0.0, 1, 0.0),` as rotation.

Node 2

► Icosahedron

```
//cube - node 2
glPushMatrix();
glColor3d(0.70, 0.52, 0.75);
glScalef(2.5, 2.5, 2.5);
glRotatef(angle, 0.0f, 1.0f, 1.0f);
glRotatef((GLfloat)node2, 0, 1, 0);
glutWireIcosahedron();
glPopMatrix();
glPopMatrix(); //end of node 2
```

The transformations involved are `glScalef(2.5, 2.5, 2.5);` which is to scale, and `glRotatef((GLfloat)node2, 0.0, 1, 0.0),` as rotation and auto rotate angle 360.

EXPLANATION OF SUBNODE

Subnode 1

► Bottom Scoop

```
//bottom scoop - subnode 1
	glPushMatrix();
	glColor3d(0.87, 0.51, 0.85);
	glTranslatef(0.0, 0.0, 0.0);
	glRotatef(-90, 0, 1, 0);
	glTranslatef(0, 0.1, 0.0);
	glRotatef((GLfloat)subnode, 0, 1, 0);
	glTranslatef(subnode, 0, 0);
	sphere();
	glPopMatrix();
	//below ice cream
```

Subnode 1 is the bottom scoop. It is programmed to transform at the same time when a specific keyboard button is pressed which is R and T. The transformations involved are `glTranslatef(subnode, 0.0, 0.0)` which is translation for the movement and `glRotatef((GLfloat)subnode, 0.0, 1.0, 0.0),` rotation at the y axis. Same goes to subnode 2 which is top scoop.

Subnode 2

► Top Scoop

```
//top scoop - subnode 2
glPushMatrix();
glColor3d(0.64, 0.05, 0.05);
glTranslatef(0.0, 0.0, 0.0);
glRotatef(-90, 0, 1, 0);
glTranslatef(0, 0.6, 0.0);
glRotatef((GLfloat)subnode2, 0, 1, 0);
glTranslatef(subnode2, 0, 0);
sphere();
glPopMatrix();
//top ice cream
```

Subnode 1 is the bottom scoop. It is programmed to transform at the same time when a specific keyboard button is pressed which is C and F. The transformations involved are `glTranslatef(subnode2, 0, 0)` which is translation for the movement and `glRotatef((GLfloat)subnode2, 0, 1, 0),` rotation at the y axis.

Subnode 3

- Cherry and Sprinkles

```
//cherry n sprinkles
glPushMatrix();
glTranslatef(subnode3, 0, 0);
```

Subnode 3 is the cherry and sprinkles. It is programmed to transform at the same time when a specific keyboard button is pressed which is B and V. The transformations involved are `glTranslatef(subnode2, 0, 0)` which is translation for the movement and `glRotatef((GLfloat)subnode2, 0, 1, 0),` rotation at the y axis.

Subnode 4

- Cherry

```
glPushMatrix();
//cherry - subnode 4
glPushMatrix();
glColor3d(1.0, 0.0, 0.0);
glTranslatef(subnode4, 0, 0);
glTranslatef(0.0, 0.0, 0.0);
glRotatef(90, 0, 1, 0);
glTranslatef(0, 1.2, 0.0);
cherry();
glPopMatrix();
//cherry

//cherrytop - subnode 4
glPushMatrix();
glColor3d(1.0, 0.0, 0.0);
glTranslatef(0.0, 1.6, 0.0);
glRotatef(90, 1, 0, 0);
glTranslatef(subnode4, 0, 0);
cherrytop();
glPopMatrix();
//cherrytop
```

Subnode 4 is the cherry. It is programmed to transform at the same time when a specific keyboard button is pressed which is U and Y. The transformations involved are `glTranslatef(subnode2, 0, 0)` which is translation for the movement and `glRotatef((GLfloat)subnode2, 0, 1, 0),` rotation at the y axis.

USER GUIDE

Key Transformations

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;`

D	Transformation of the root of the diagram to move right.

	`case 'd':`
	`case 'D':`
	`rootx++;`
	`glutPostRedisplay();`
	`break;`

A	Transformation of the root of the diagram to move left.

	`case 'a':`
	`case 'A':`
	`rootx--;`
	`glutPostRedisplay();`
	`break;`

E	Transformation of the node to move clockwise.

	`case 'e':`
	`case 'E':`
	`node = (node + 5) % 360;`
	`glutPostRedisplay();`
	`break;  //cone`

Q	Transformation of the nodes to move anti-clockwise.

<table>
<tr><td></td><td>

```
case 'q':
case 'Q':
        node = (node - 5) % 360;
        glutPostRedisplay();
        break;        //cone
```

</td></tr>
<tr><td rowspan="2">C</td><td>Transformation of the node2 to move clockwise.</td></tr>
<tr><td>

```
case 'c':
case 'C':
        node2 = (node2 + 5) % 360;
        glutPostRedisplay();
        break;        //cube
```

</td></tr>
<tr><td rowspan="2">Z</td><td>Transformation of the node2 to move anti-clockwise.</td></tr>
<tr><td>

```
case 'z':
case 'Z':
        node2 = (node2 - 5) % 360;
        glutPostRedisplay();
        break;        //cube
```

</td></tr>
<tr><td rowspan="2">T</td><td>Transformation of subnode rotating to clockwise and translate at x-axis.</td></tr>
<tr><td>

```
case 't':
case 'T':
        subnode = (subnode + 5) % 360;
        glutPostRedisplay();
        break;        //bottom scoop
```

</td></tr>
<tr><td rowspan="2">R</td><td>Transformation of subnode rotating to anti-clockwise and translate at x-axis.</td></tr>
<tr><td>

```
case 'r':
case 'R':
        subnode = (subnode - 5) % 360;
        glutPostRedisplay();
        break;        //bottom scoop
```

</td></tr>
<tr><td rowspan="2">G</td><td>Transformation of subnode2 rotating to clockwise and translate at x-axis.</td></tr>
<tr><td>

```
case 'g':
case 'G':
        subnode2 = (subnode2 + 5) % 360;
        glutPostRedisplay();
        break;        //top scoop
```

</td></tr>
<tr><td rowspan="2">F</td><td>Transformation of subnode2 rotating to anti-clockwise and translate at x-axis.</td></tr>
<tr><td>

```
case 'f':
case 'F':
        subnode2 = (subnode2 - 5) % 360;
        glutPostRedisplay();
        break;        //top scoop
```

</td></tr>
</table>

B	Transformation of subnode3 to the right.

	`case 'b':`
	`case 'B':`
	`subnode3++;`
	`glutPostRedisplay();`
	`break; //cherry & sprinkle`

V	Transformation of subnode3 to the left.

	`case 'v':`
	`case 'V':`
	`subnode3--;`
	`glutPostRedisplay();`
	`break; //cherry & sprinkle`

U	Transformation of subnode4 to the right.

	`case 'u':`
	`case 'U':`
	`subnode4++;`
	`//x rotate`
	`glutPostRedisplay();`
	`break; //cherry`

Y	Transformation of subnode4 to the left.

	`case 'y':`
	`case 'Y':`
	`subnode4--;`
	`//x rotate`
	`glutPostRedisplay();`
	`break; //cherry`

Camera Projections

1	Move camera direction at x axis and x center to the right.
	`case '1': xkey += 0.1; xcentre += 0.1; glutPostRedisplay(); break;`
2	Move camera direction at x axis and x center to the left.
	`case '2': xkey -= 0.1; xcentre -= 0.1; glutPostRedisplay(); break;`
3	Move material projection at y axis upwards.
	`case '3': ykey += 0.1; glutPostRedisplay(); break;`
4	Move material projection at y axis downwards.
	`case '4': ykey -= 0.1; glutPostRedisplay(); break;`
5	Move material projection at y center upwards.
	`case '5': ycentre += 0.1; glutPostRedisplay(); break;`
6	Move material projection at y center downwards.
	`case '6': ycentre -= 0.1; glutPostRedisplay(); break;`

<table>
<tr><td rowspan="2">7</td><td>Move camera direction at z axis to the right.</td></tr>
<tr><td>

```
case '7':
        z += 0.1;
        glutPostRedisplay();
        break;
```

</td></tr>
<tr><td rowspan="2">8</td><td>Move camera direction at z axis to the left.</td></tr>
<tr><td>

```
case '8':
        z -= 0.1;
        glutPostRedisplay();
        break;
```

</td></tr>
<tr><td rowspan="2">9</td><td>Move camera direction in reverse on all x, y, z axis and x, y center.</td></tr>
<tr><td>

```
case '9':
        xkey -= 0.1;
        ykey -= 0.1;
        z -= 0.1;
        ycentre -= 0.1;
        xcentre -= 0.1;
        glutPostRedisplay();
        break;
```

</td></tr>
</table>

3D HIERARCHICAL MODELLING FULL CODING

```
//AIMI BINTI RUSDI (B19EC0001)
//AQILAH HANIM BINTI MOHD TAUFIK (B19EC0006)
//Lab 7 & 8 source code

#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>

GLfloat angle = 0.0, xkey = 0.0, ykey = 0.0, z = 0.5, ycentre = 0.0, xcentre = 0.0;
static int root = 0, rootx = 0, rooty = 0, node = 0, node2 = 0, nodex = 0, nodey = 0,
subnode = 0, subnode2 = 0;
static int zsprinkle = -1.5, subnode3 = 0, subnode4 = 0;
bool pushButton = false;

void init(void)
{

	GLfloat mat_specular[] = { 0.65, 0.05, 0.05, 0.0 };
	//marroon red colour
	GLfloat mat_diffuse[] = { 1.0, 0.05, 0.5 };
		//colour of lighting - this makes the lighting follow specular
	GLfloat mat_ambient[] = { 0.65, 0.05, 0.05, 0.0 };
	GLfloat mat_shininess[] = { 10.0 };
	GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
	//x,y,z and w position of light
	glClearColor(0.0, 0.0, 0.0, 0.0);
		/* clear the background colour to black */
	glShadeModel(GL_SMOOTH);

	glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
	glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
	glColorMaterial(GL_FRONT, GL_EMISSION);
	glLightfv(GL_LIGHT0, GL_POSITION, light_position);
	glLightModelfv(GL_LIGHT_MODEL_AMBIENT, mat_ambient);

	glEnable(GL_COLOR_MATERIAL);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
	glEnable(GL_DEPTH_TEST);
}

void Cone(void)
{
	/* Position and display GLUT wire-frame cone. */
	glutSolidCone(0.5, 1.8, 15.0, 7);
		//(base, height, longitude, latitude)
}


void sphere(void)
{
	glutSolidSphere(0.48, 20, 16);
			//(radius, nlogitude, nlatitude)
}
```

```
void cherry(void)
{
	glutSolidSphere(0.17, 10, 8);
			//(radius, nlogitude, nlatitude)
}

void cherrytop(void)
{
	GLUquadricObj* cylinder;
			// Set name for GLU quadric object.
	cylinder = gluNewQuadric();
	gluCylinder(cylinder, 0.01, 0.01, 0.3, 8, 90);
	//(base, top,length, sides,)
}

void sprinkles(void)
{
	GLUquadricObj* cylinder;
	cylinder = gluNewQuadric();
	gluCylinder(cylinder, 0.02, 0.02, 0.15, 8, 1);
}

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glColor3f(0.0, 0.0, 0.0);
			//white
	glLoadIdentity();
				/* clear the matrix */

	/* viewing transformation  */
	gluLookAt(xkey, ykey, z, 0.0, ycentre, xcentre, 0.0, 1.0, 0.0);
	//(x, y, z, xcentre, ycentre, zcentre, xup, yup, zup)

	//----------------------------------- Level 0 ----------------------------------
	glPushMatrix();
				//for all the objects
	glScalef(0.7, 0.7, 0.7);
			//to fit the object in screen
	glRotatef((GLfloat)root, 0.0, 0.0, 0.1);
	glTranslatef((GLfloat)rootx, (GLfloat)rooty, 0.0);

	//----------------------------------- Level 1 ----------------------------------
	//cone - node 1
	glPushMatrix();
	glColor3d(0.65, 0.33, 0.0);
	glTranslatef(0.0, 0.0, 0.0);
	glRotatef(90, 0, 1, 0);
	glRotatef(90, 1, 0, 0);
	glRotatef((GLfloat)node, 0, 1, 0);
	glTranslatef(0.0, 0.0, 0.0);
	Cone();
	glPopMatrix();
	//cone
```

```
        //cube - node 2
        glPushMatrix();
        glColor3d(0.70, 0.52, 0.75);
        glScalef(2.5, 2.5, 2.5);
        glRotatef(angle, 0.0f, 1.0f, 1.0f);
        glRotatef((GLfloat)node2, 0, 1, 0);
        glutWireIcosahedron();
        glPopMatrix();

        //---------------------------------- Level 2 ----------------------------------
        //bottom scoop - subnode 1
        glPushMatrix();
        glColor3d(0.87, 0.51, 0.85);
        glTranslatef(0.0, 0.0, 0.0);
        glRotatef(-90, 0, 1, 0);
        glTranslatef(0, 0.1, 0.0);
        glRotatef((GLfloat)subnode, 0, 1, 0);
        glTranslatef(subnode, 0, 0);
        sphere();
        glPopMatrix();
        //below ice cream

        //top scoop - subnode 2
        glPushMatrix();
        glColor3d(0.64, 0.05, 0.05);
        glTranslatef(0.0, 0.0, 0.0);
        glRotatef(-90, 0, 1, 0);
        glTranslatef(0, 0.6, 0.0);
        glRotatef((GLfloat)subnode2, 0, 1, 0);
        glTranslatef(subnode2, 0, 0);
        sphere();
        glPopMatrix();
        //top ice cream

        //---------------------------------- Level 3 ----------------------------------
        //cherry n sprinkles
        glPushMatrix();
        glTranslatef(subnode3, 0, 0);

        glPushMatrix();
        //cherry - subnode 4
        glPushMatrix();
        glColor3d(1.0, 0.0, 0.0);
        glTranslatef(subnode4, 0, 0);
        glTranslatef(0.0, 0.0, 0.0);
        glRotatef(90, 0, 1, 0);
        glTranslatef(0, 1.2, 0.0);
        cherry();
        glPopMatrix();
        //cherry

        //cherrytop - subnode 4
        glPushMatrix();
        glColor3d(1.0, 0.0, 0.0);
        glTranslatef(0.0, 1.6, 0.0);
        glRotatef(90, 1, 0, 0);
        glTranslatef(subnode4, 0, 0);
        cherrytop();
```

```
        glPopMatrix();
        //cherrytop

        glPopMatrix();
        //subnode 4

        //changes sprinkles
        glPushMatrix();

        //sprinkles 1
        glPushMatrix();
        glColor3d(0.0, 1.3, 0.3);
                        //green
        glRotatef(-25, 1, 3, 0);
        glTranslatef(0.5, 0.8, 0);
        sprinkles();
        glPopMatrix();
        //sprinkles 1

        //sprinkles 2
        glPushMatrix();
        glColor3d(0.9, 0.6, 0.0);
        glRotatef(25, 1, 3, 0);
        glTranslatef(0.5, 0.6, 0);
                        //x to make it stick
        sprinkles();
        glPopMatrix();
        //sprinkles 2

        //sprinkles 3
        glPushMatrix();
        glColor3d(0.0, 0.0, 1.7);
        glRotatef(25, -10, 3, 0);
        glTranslatef(0.4, 0.9, 0);
                        //x to make it stick
        sprinkles();
        glPopMatrix();
        //sprinkles 3

        //sprinkles 4
        glPushMatrix();
        glColor3d(1.0, 0.75, 0.0);
                        //yellow
        glRotatef(30, -10, 3, 0);
        glTranslatef(0.45, 0.7, 0);
                        //x to make it stick
        sprinkles();
        glPopMatrix();
        //sprinkles 4

        glPopMatrix();
        //changes sprinkles

        glPopMatrix();
        //cherry and sprinkles

        glPopMatrix();
        //overall objects
```

```
	angle += 0.3;

	glutSwapBuffers();
	glFlush();
}

void keyboard(unsigned char key, int x, int y)
{
	switch (key) {
	case '1':
		xkey += 0.1;
		xcentre += 0.1;
		glutPostRedisplay();
		break;

	case '2':
		xkey -= 0.1;
		xcentre -= 0.1;
		glutPostRedisplay();
		break;

	case '3':
		ykey += 0.1;
		glutPostRedisplay();
		break;

	case '4':
		ykey -= 0.1;
		glutPostRedisplay();
		break;

	case '5':
		ycentre += 0.1;
		glutPostRedisplay();
		break;

	case '6':
		ycentre -= 0.1;
		glutPostRedisplay();
		break;

	case '7':
		z += 0.1;
		glutPostRedisplay();
		break;

	case '8':
		z -= 0.1;
		glutPostRedisplay();
		break;

	case '9':
		xkey -= 0.1;
		ykey -= 0.1;
		z -= 0.1;
		ycentre -= 0.1;
		xcentre -= 0.1;
```

```
            glutPostRedisplay();
            break;

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

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

        case 'd':
        case 'D':
            rootx++;
            glutPostRedisplay();
            break;

        case 'a':
        case 'A':
            rootx--;
            glutPostRedisplay();
            break;

        case 'e':
        case 'E':
            node = (node + 5) % 360;
            glutPostRedisplay();
            break;        //cone

        case 'q':
        case 'Q':
            node = (node - 5) % 360;
            glutPostRedisplay();
            break;        //cone

        case 'c':
        case 'C':
            node2 = (node2 + 5) % 360;
            glutPostRedisplay();
            break;        //cube

        case 'z':
        case 'Z':
            node2 = (node2 - 5) % 360;
            glutPostRedisplay();
            break;        //cube

        case 't':
        case 'T':
            subnode = (subnode + 5) % 360;
            glutPostRedisplay();
            break;        //bottom scoop
```

```
	case 'r':
	case 'R':
		subnode = (subnode - 5) % 360;
		glutPostRedisplay();
		break;		//bottom scoop

	case 'g':
	case 'G':
		subnode2 = (subnode2 + 5) % 360;
		glutPostRedisplay();
		break;		//top scoop

	case 'f':
	case 'F':
		subnode2 = (subnode2 - 5) % 360;
		glutPostRedisplay();
		break;		//top scoop

	case 'b':
	case 'B':
		subnode3++;
		glutPostRedisplay();
		break;		//cherry & sprinkle

	case 'v':
	case 'V':
		subnode3--;
		glutPostRedisplay();
		break;		//cherry & sprinkle

	case 'y':
	case 'Y':
		subnode4--;
			//x rotate
		glutPostRedisplay();
		break;		//cherry

	case 'u':
	case 'U':
		subnode4++;
			//x rotate
		glutPostRedisplay();
		break;		//cherry

	case 27:
		exit(0);
		break;

	default:
		exit(0);
		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;
	}
}

void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	if (w <= h)
		glOrtho(-1.5, 1.5, -1.5*(GLfloat)h / (GLfloat)w, 1.5*(GLfloat)h /
(GLfloat)w, -10.0, 10.0);
	else
		glOrtho(-1.5*(GLfloat)w / (GLfloat)h, 1.5*(GLfloat)w / (GLfloat)h, -1.5,
1.5, -10.0, 10.0);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
		//single buffer
	glutInitWindowSize(1000, 600);
	glutInitWindowPosition(100, 100);
	glutCreateWindow("3D Ice Cream Project | AIMI & AQILAH HANIM");
	init();
	glutDisplayFunc(display);
	glutIdleFunc(display);
	glutReshapeFunc(reshape);
	glutKeyboardFunc(keyboard);
	glutSpecialFunc(specialKeys);
	glutMainLoop();
	return 0;
}
```

3D HIERARCHICAL MODELLING OUTPUT

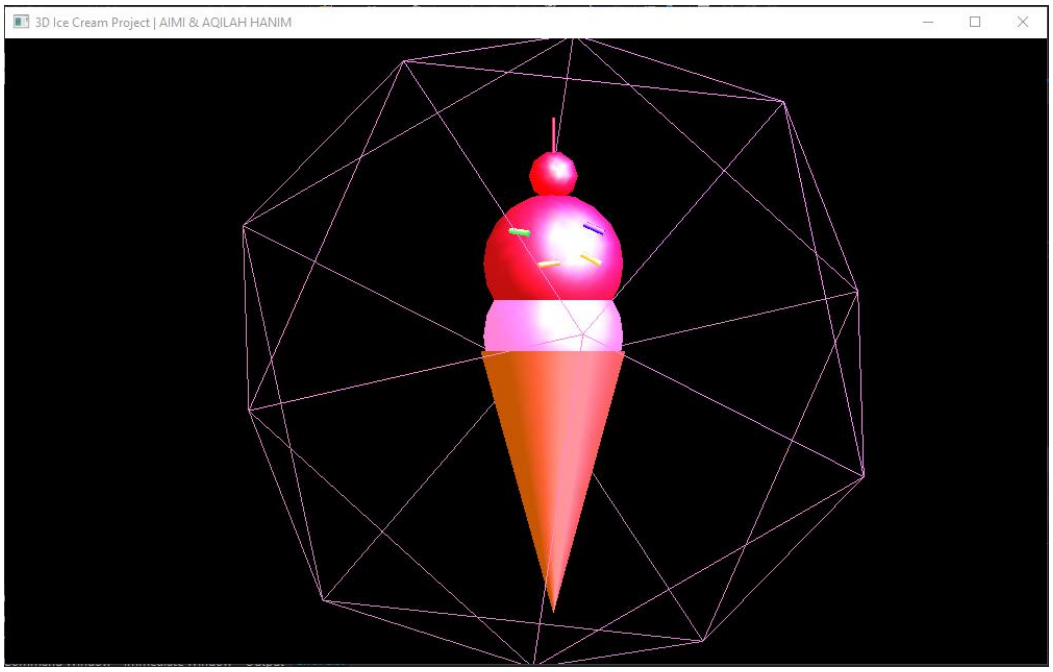


Figure 2. Output of rotation of y axis.

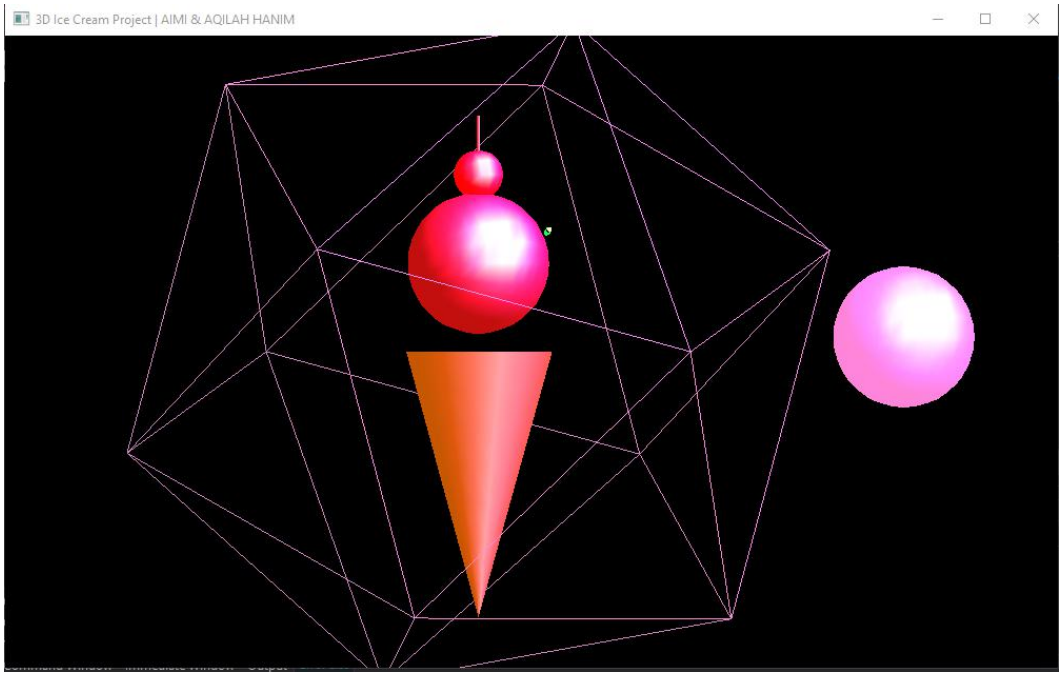


Figure 3. Output of rotation and translation of subnode.

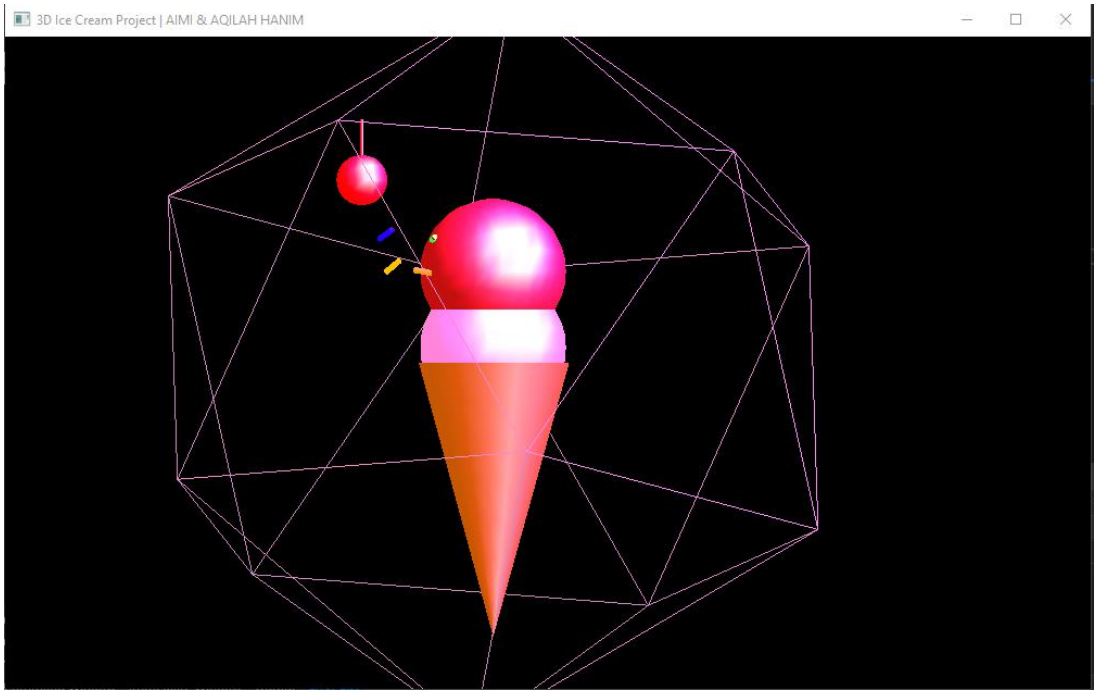


Figure 4. Output of translation of subnode.

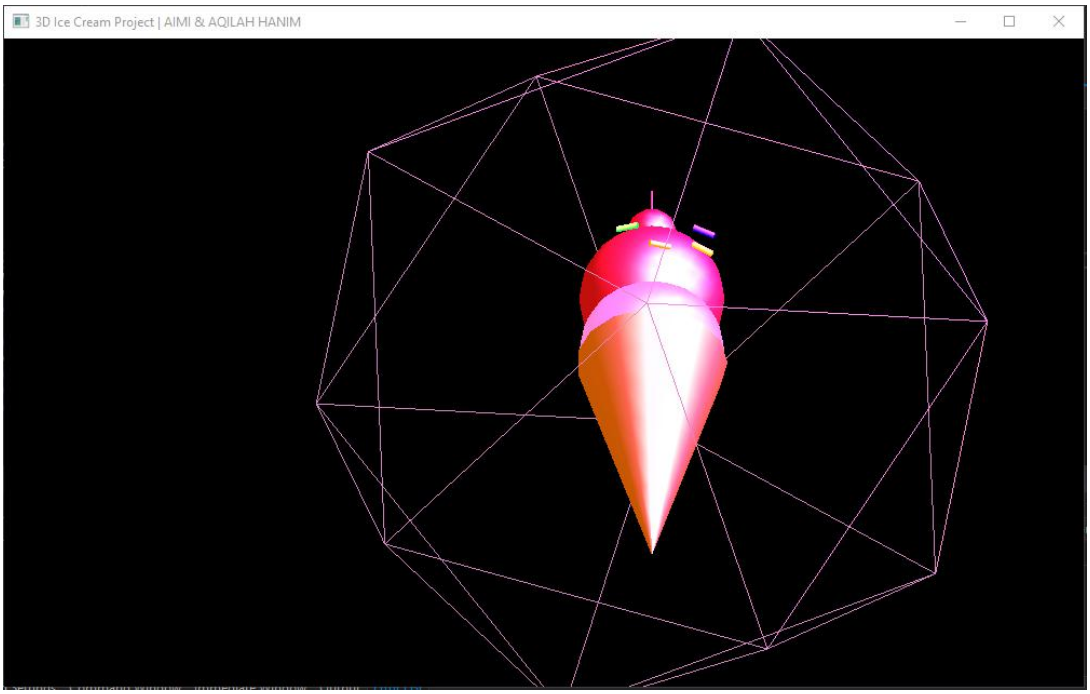


Figure 5. Output of camera view of yaxis and ycenter.

REFERENCES

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