Lab08 for CPT205 Computer Graphics (Hierarchical Modelling)

PART 1. Sample Programs

There are two sample programs for this lab. Both programs are about hierarchical modelling, incorporating other techniques such as interactions and viewing. You are provided with these sample programs to help further understand relevant techniques and OpenGL functions in particular. Practise with the two programs through the following steps:

1) Read the code,
2) Run it,
3) Try to understand important techniques and relevant OpenGL functions,
4) Add your features, e.g. add the moon (which rotates around the Earth) to the Solar System (and later on add some lighting/materials effect and textures when the topics are covered).

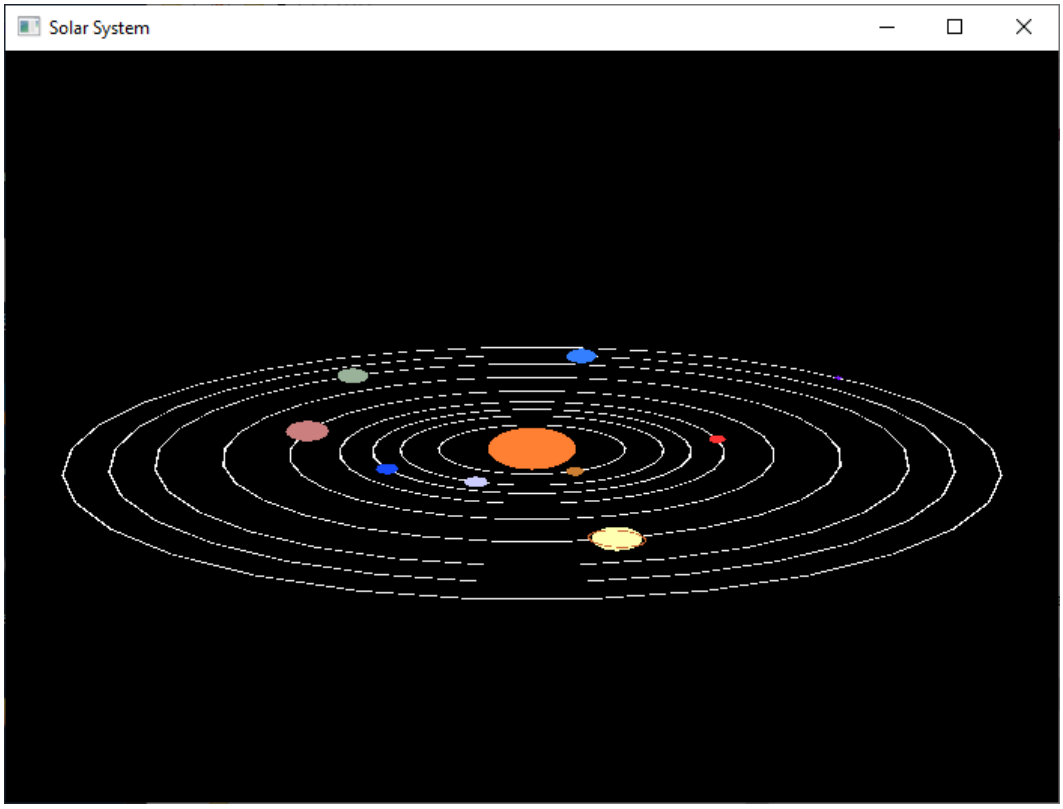
1) The Solar System

```
// Solar System
#define FREEGLUT_STATIC
#include <GL/freeglut.h>
#include <stdlib.h>

void set_initial_state(void);
void myDisplay(void);
void myIdleFunc(void);
void myKeyboardFunc(void);

float mercury = 360;
float venus = 360;
float earth = 360;
float mars = 360;
float jupiter = 360;
float saturn = 360;
float uranus = 360;
float neptune = 360;
float pluto = 360;

GLint winWidth = 700, winHeight = 500;          // initial display window size

GLfloat x0 = 0.0, y0 = 1000.0, z0 = 1500.0;     // viewing co-ordinate origin
GLfloat xref = 0.0, yref = 50.0, zref = 0.0;    // look-at point
GLfloat Vx = 0.0, Vy = 1.0, Vz = 0.0;           // view-up vector

// co-ordinate limits for clipping window
GLfloat xwMin = -40.0, ywMin = -60.0, xwMax = 40.0, ywMax = 60.0;

// positions for near and far clipping planes
GLfloat dnear = 150.0, dfar = 2500.0;

void myReshapeFunc(GLint newWidth, GLint newHeight)
{
    glViewport(0, 0, newWidth, newHeight);
}

void myKeyboardFunc(unsigned char key, int x, int y)
{
    //Original viewing co-ordinate origin
    if (key == 'o' || key == 'O')
        x0 = 0.0, y0 = 1000.0, z0 = 1500.0;

    //Sun origin viewing point
    if (key == 's' || key == 'S')
        x0 = 0.0, y0 = 1000.0, z0 = 1.0;
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(winWidth, winHeight);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Solar System");
```

```
    glutDisplayFunc(myDisplay);
    glutIdleFunc(myIdleFunc);
    glutReshapeFunc(myReshapeFunc);
    glutKeyboardFunc(myKeyboardFunc);
    set_initial_state();
    glutMainLoop();
}

void set_initial_state(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glColor3f(0.0, 0.0, 0.0);
    glLoadIdentity();

    glMatrixMode(GL_PROJECTION);
    glFrustum(xwMin, xwMax, ywMin, ywMax, dnear, dfar);
}

void myIdleFunc(void)
{
    mercury += 1.4;  // incremental angle per frame while 60 frames/second (so very fast)
    venus += 1.2;
    earth += 1.0;
    mars += 0.9;
    jupiter += 0.8;
    saturn += 0.5;
    uranus += 0.6;
    neptune += 0.4;
    pluto += 0.2;

    glutPostRedisplay();
}

void myDisplay(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);

    //Solar System
    glPushMatrix();
    glColor3f(1.0, 1.0, 1.0);
    glRotatef(90, 1, 0, 0);         // from x-y plane to x-z plane
    glutSolidTorus(1, 420, 5, 30);  // minor radius, major radius,
    glutSolidTorus(1, 380, 5, 30);  // number of sides of section circle and number of sections
    glutSolidTorus(1, 340, 5, 30);
    glutSolidTorus(1, 280, 5, 30);
    glutSolidTorus(1, 220, 5, 30);
    glutSolidTorus(1, 175, 5, 30);
    glutSolidTorus(1, 145, 5, 30);
    glutSolidTorus(1, 120, 5, 30);
    glutSolidTorus(1, 85, 5, 30);
    glTranslatef(0, 0, 0);
    glPopMatrix();

    //Sun
    glPushMatrix();
    glColor3f(1.0, 0.5, 0.2);
    glTranslatef(0, 0, 0);

    glPushMatrix();
    glutSolidSphere(40, 40, 40);
    glPopMatrix();
    glPopMatrix();

    //Mercury
    glPushMatrix();
    glColor3f(0.8, 0.5, 0.2);
    glTranslatef(0, 0, 0);
    glRotatef(mercury, 0, 1, 0);

    glPushMatrix();
    glTranslatef(85, 0, 0);  // send Mercury to its obit

    glutSolidSphere(8, 8, 8);
```

```
glPopMatrix();
glPopMatrix();

//Venus
glPushMatrix();
glColor3f(0.8, 0.8, 1.0);
glTranslatef(0, 0, 0);
glRotatef(venus, 0, 1, 0);

glPushMatrix();
glTranslatef(120, 0, 0);
glutSolidSphere(10, 10, 10);
glPopMatrix();
glPopMatrix();

//Earth
glPushMatrix();
glColor3f(0.1, 0.3, 1.0);
glTranslatef(0, 0, 0);
glRotatef(earth, 0, 1, 0);

glPushMatrix();
glTranslatef(145, 0, 0);
glutSolidSphere(10, 10, 10);
glPopMatrix();
glPopMatrix();

//Mars
glPushMatrix();
glColor3f(1.0, 0.2, 0.2);
glTranslatef(0, 0, 0);
glRotatef(mars, 0, 1, 0);

glPushMatrix();
glTranslatef(175, 0, 0);
glutSolidSphere(8, 8, 8);
glPopMatrix();
glPopMatrix();

//Jupiter
glPushMatrix();
glColor3f(0.8, 0.5, 0.5);
glTranslatef(0, 0, 0);
glRotatef(jupiter, 0, 1, 0);

glPushMatrix();
glTranslatef(220, 0, 0);
glutSolidSphere(20, 20, 20);
glPopMatrix();
glPopMatrix();

//Saturn
glPushMatrix();
glColor3f(1.0, 1.0, 0.7);
glRotatef(saturn, 0, 1, 0);
glTranslatef(280, 0, 0);
glutSolidSphere(20, 20, 20);

//Rings Of Saturn
glPushMatrix();
glColor3f(0.8, 0.4, 0.2);
glRotatef(83, 1, 0, 0);
glutSolidTorus(1, 23, 5, 30);
glTranslatef(200, 0, 0);
glPopMatrix();
glPopMatrix();

//Uranus
glPushMatrix();
glColor3f(0.6, 0.7, 0.6);
glTranslatef(0, 0, 0);
glRotatef(uranus, 0, 1, 0);

glPushMatrix();
glTranslatef(340, 0, 0);
glutSolidSphere(16, 16, 16);
glPopMatrix();
```

```
    glPopMatrix();

    //Neptune
    glPushMatrix();
    glColor3f(0.2, 0.5, 1.0);
    glTranslatef(0, 0, 0);
    glRotatef(neptune, 0, 1, 0);

    glPushMatrix();
    glTranslatef(380, 0, 0);
    glutSolidSphere(16, 16, 16);
    glPopMatrix();
    glPopMatrix();

    //Pluto
    glPushMatrix();
    glColor3f(0.5, 0.1, 1.0);
    glTranslatef(0, 0, 0);
    glRotatef(pluto, 0, 1, 0);

    glPushMatrix();
    glTranslatef(420, 0, 0);
    glutSolidSphere(4, 4, 4);
    glPopMatrix();
    glPopMatrix();

    glutSwapBuffers();
}
```



2) A Train System

```
// Train System
#define FREEGLUT_STATIC
#include <GL/freeglut.h>
#include <stdlib.h>
#include<math.h>

int intWinWidth = 600;              //Default window size
int intWinHeight = 400;

float fltAnimationOffset = 0.0;     //Animation offsets for various objects
float fltTreeOffset = 0.0;
float fltBuildingOffset = -2500.0;
float fltPoleOffset = 750.0;
float fltAcceleration = 6.0;        //Speed of animation

float fltColourVal1 = 0.5;          //First colour value
float fltColourVal2 = 0.7;          //Second colour value
float fltColourVal3 = 0.4;          //Third colour value

float fltFOV = 70;                  //Field Of View
float fltZoom = 1.0;                //Zoom amount
float fltX0 = 0.0;                  //Camera position
float fltY0 = 150.0;
float fltZ0 = -500.0;
float fltXRef = 0.0;                //Look At reference point
float fltYRef = 0.0;
float fltZRef = 0.0;
float fltXUp = 0.0;                 //Up vector
float fltYUp = 1.0;
float fltZUp = 0.0;
float fltViewingAngle = 0;          //Used for rotating camera

void animateTracks()
{
    //Increments/decrements animation variables
    fltAnimationOffset -= fltAcceleration;
    if (fltAnimationOffset <= -20.0)
        fltAnimationOffset = 0.0;
    else if (fltAnimationOffset >= 0.0)
        fltAnimationOffset = -20.0;

    fltPoleOffset -= fltAcceleration;
    if (fltPoleOffset <= -400.0)
        fltPoleOffset = 0.0;
    else if (fltPoleOffset >= 0.0)
        fltPoleOffset = -400.0;

    fltTreeOffset -= fltAcceleration;
    if (fltTreeOffset <= -2500.0)
        fltTreeOffset = 2500.0;
    else if (fltTreeOffset >= 2500.0)
        fltTreeOffset = -2500.0;

    fltBuildingOffset -= fltAcceleration;
    if (fltBuildingOffset <= -2500.0)
        fltBuildingOffset = 2500.0;
    else if (fltBuildingOffset >= 2500.0)
        fltBuildingOffset = -2500.0;

    glutPostRedisplay();
}

void groundAndTracks()
{
    //========TELEGRAPH POLES========
    for (int fltPoleNumber = -25; fltPoleNumber < 25; fltPoleNumber += 4.0)
    {
        glPushMatrix();
        glTranslatef(-200.0, 0.0, ((fltPoleNumber * 100) + fltPoleOffset));
        glPushMatrix();
        glTranslatef(0.0, 200.0, 0.0);
        glScalef(80.0, 10.0, 10.0);
        glColor3f(0.7, 0.3, 0.0);
        glutSolidCube(1);
        glColor3f(0.0, 0.0, 0.0);
```

```
    glutWireCube(1);
    glPopMatrix();
    glPushMatrix();
    glTranslatef(0.0, 100.0, 0.0);
    glScalef(7.0, 200.0, 7.0);
    glColor3f(0.7, 0.3, 0.0);
    glutSolidCube(1);
    glColor3f(0.0, 0.0, 0.0);
    glutWireCube(1);
    glPopMatrix();
    glPopMatrix();
}

//========BUILDINGS========
glPushMatrix();
glTranslatef(-400.0, 0.0, fltBuildingOffset);
glPushMatrix();
glTranslatef(0.0, 75, 0.0);
glScalef(200.0, 150.0, 1000.0);
glColor3f(fltColourVal1, fltColourVal2, fltColourVal3);
glutSolidCube(1);
glColor3f(0.0, 0.0, 0.0);
glutWireCube(1);
glPopMatrix();
glPopMatrix();
glPushMatrix();
glTranslatef(400.0, 0.0, fltBuildingOffset);
glPushMatrix();
glTranslatef(0.0, 75, 0.0);
glScalef(400.0, 200.0, 600.0);
glColor3f(fltColourVal3, fltColourVal1, fltColourVal2);
glutSolidCube(1);
glColor3f(0.0, 0.0, 0.0);
glutWireCube(1);
glPopMatrix();
glPopMatrix();

//========TREE========
glPushMatrix();
glTranslatef(150.0, 0.0, fltTreeOffset);
glPushMatrix();
glTranslatef(0.0, 100.0, 0.0);
glScalef(60.0, 65.0, 60.0);
glColor3f(0.0, 0.7, 0.1);
glutSolidIcosahedron();
glColor3f(0.0, 0.0, 0.0);
glutWireIcosahedron();
glPopMatrix();
glPushMatrix();
glTranslatef(0.0, 25.0, 0.0);
glScalef(15.0, 50.0, 15.0);
glColor3f(0.7, 0.3, 0.0);
glutSolidCube(1);
glColor3f(0.0, 0.0, 0.0);
glutWireCube(1);
glPopMatrix();
glPopMatrix();

//========TRACKS========
glPushMatrix();
glTranslatef(-75.0, 5.0, 0.0);
glScalef(5.0, 5.0, 5000.0);
glPushMatrix();
glColor3f(0.3, 0.3, 0.3);
glutSolidCube(1);
glColor3f(0.0, 0.0, 0.0);
glutWireCube(1);
glPopMatrix();
glPopMatrix();
glPushMatrix();
glTranslatef(75.0, 5.0, 0.0);
glScalef(5.0, 5.0, 5000.0);
glPushMatrix();
glColor3f(0.3, 0.3, 0.3);
glutSolidCube(1);
glColor3f(0.0, 0.0, 0.0);
glutWireCube(1);
```

```
    glPopMatrix();
    glPopMatrix();

    //========TRACK SLEEPERS========
    for (int intSleeper = -249;intSleeper < 250;intSleeper += 2)
    {
        float fltZOffset = intSleeper * 10;
        glPushMatrix();
        glTranslatef(0.0, 0.0, fltZOffset + fltAnimationOffset);
        glScalef(200.0, 5.0, 10.0);
        glPushMatrix();
        glColor3f(0.5, 0.25, 0.0);
        glutSolidCube(1);
        glColor3f(0.0, 0.0, 0.0);
        glutWireCube(1);
        glPopMatrix();
        glPopMatrix();
    }

    //========GROUND========
    glPushMatrix();
    glTranslatef(0.0, -10.0, 0.0);
    glScalef(5000.0, 10.0, 5000.0);
    glPushMatrix();
    glColor3f(0.3, 1.0, 0.2);
    glutSolidCube(1);
    glColor3f(0.0, 0.0, 0.0);
    glutWireCube(1);
    glPopMatrix();
    glPopMatrix();
}

void displayObject()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(fltFOV, 1, 0.1, 5000);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(fltX0 * fltZoom, fltY0 * fltZoom, fltZ0 * fltZoom, fltXRef, fltYRef, fltZRef, fltXUp,
fltYUp, fltZUp);

    glEnable(GL_DEPTH_TEST);

    glClearColor(0.2, 0.2, 0.8, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    groundAndTracks();

    glutSwapBuffers();
}

void reshapeWindow(GLint intNewWidth, GLint intNewHeight)
{
    glViewport(0, 0, intNewWidth, intNewHeight);
}

void adjustDisplay(unsigned char key, int x, int y)
{
    if (key == '-' || key == '_')
        fltAcceleration -= 0.5;          //Speed down
    if (key == '=' || key == '+')
        fltAcceleration += 0.5;          //Speed up
    if (key == 'd' || key == 'D')
        if (fltY0 > 35)
            fltY0 -= 5;                  //Camera down
    if (key == 'u' || key == 'U')
        if (fltY0 < 500)
            fltY0 += 5;                  //Camera up
    if (key == 'i' || key == 'I')
        if (fltZoom > 0.2)
            fltZoom -= 0.1;              //Zoom in
    if (key == 'o' || key == 'O')
        if (fltZoom < 5.0)
            fltZoom += 0.1;              //Zoom out
```

```
    glutPostRedisplay();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition(50, 50);
    glutInitWindowSize(intWinWidth, intWinHeight);
    glutCreateWindow("Train System");
    glutDisplayFunc(displayObject);
    glutReshapeFunc(reshapeWindow);
    glutKeyboardFunc(adjustDisplay);
    glutIdleFunc(animateTracks);  // try to understand static scene by turning this off
    glutMainLoop();
}
```

PART 2. Further Work

1) Read and understand the sample code in the lecture slides, and try to write a complete program for the robot arm.

2) Design and implement a dragonfly with graphic primitives and animate it.