**HELICOPTER PROJECT CODE**

#include <GL/freeglut.h>

#include <stdlib.h>

#include <math.h>

GLint sub\_menu[5];//list of submenus

class GLfloatVector

{

public:

GLfloat x, y, z;

};

class GLfloatPoint

{

public:

GLfloat x, y, z;

};

// Global variables

GLfloatVector eye = {2.0, 8.0, -15.0};

GLfloatPoint pos = {0.0, 0.0, 0.0};

float angle = 0; // To rotate the rotor

float spin = 0; // To rotate the main body

float fovy = 40;

bool spinning = false;

bool light = false;

int w;

int h;

GLUquadric \*quad;

// Idle - what to do when there is nothing else is in the event queue

void idle()

{

if (spinning)

angle+=20;

glutPostRedisplay();

}

// Initialisation

void init(void)

{

GLfloat light\_position[] = { 3.0, 0.0, 3.0, 0.0 }; //1.0 is point light, 0.0 is directional light

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

GLfloat ambient[] = { 0.5, 0.5, 0.5, 1.0 };

GLfloat diffuse[] = { 0.4f, 0.4f, 0.4f, 1.0 };

float specular\_reflec[] = { 0.5f, 0.5f, 0.2f, 1.5 };

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular\_reflec);

glMateriali(GL\_FRONT, GL\_SHININESS, 40);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

// Enable color tracking

glEnable(GL\_COLOR\_MATERIAL);

// Set material properties which will be assigned by glColor

glColorMaterial(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE);

glClearColor(0.0,0.0,0.0,0.0);

glShadeModel(GL\_SMOOTH);

}

void display\_stars()

{

int i;

glDisable(GL\_LIGHTING);

glTranslatef(0,0,-100);

glPointSize(2.0);

glColor3f(1,1,1);

glBegin(GL\_POINTS);

for(i=0;i<1000;i++)

{

glVertex3f(rand()%600,rand()%600,rand()%600);

glVertex3f(-rand()%600,rand()%600,rand()%600);

glVertex3f(-rand()%600,-rand()%600,rand()%600);

glVertex3f(rand()%600,-rand()%600,rand()%600);

glVertex3f(rand()%600,rand()%600,-rand()%600);

glVertex3f(-rand()%600,rand()%600,-rand()%600);

glVertex3f(-rand()%600,-rand()%600,-rand()%600);

glVertex3f(rand()%600,-rand()%600,-rand()%600);

}

glEnd();

glEnable(GL\_LIGHTING);

}

//Draw helicopter - Main body

void mainBody()

{

// Main body torso

glPushMatrix();

glColor3f(1.0, 0.0, 0.0);

glutSolidCube(1);

glPopMatrix();

// Seat bottom

glPushMatrix();

glTranslated(0.75, 0, -0.25);

glScaled(0.85, 0.75, 0.25);

glColor3f(1,0,0);

glutSolidCube(0.5);

glPopMatrix();

// Seat back

glPushMatrix();

glTranslated(0.6, 0, 0);

glScaled(0.25, 0.75, 0.8);

glColor3f(1,1,0);

glutSolidCube(0.5);

glPopMatrix();

//The tail section

glPushMatrix();

glColor3f(1.0, 0.0, 0.0);

// Positive x side

glBegin(GL\_POLYGON);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(-1.0, 0.15, -0.1);

glVertex3f(-2.0f, 0.05, -0.1);

glVertex3f(-2.0, 0.05, -0.3);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-2.0f, 0.05f, -0.1f);

glVertex3f(-2.25f, 0.05f, 0.2f);

glVertex3f(-2.5f, 0.05f, 0.2f);

glVertex3f(-2.25f, 0.05f, -0.3f);

glVertex3f(-2.0f, 0.05f, -0.3f);

glEnd();

glColor3f(1,0,0);

// Negative x side

glBegin(GL\_POLYGON);

glVertex3f(-0.5, -0.5, -0.5);

glVertex3f(-0.5, -0.5, 0.5);

glVertex3f(-1.0f, -0.15f, -0.1f);

glVertex3f(-2.0f, -0.05f, -0.1f);

glVertex3f(-2.0f, -0.05f, -0.3f);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-2.0f, -0.05f, -0.1f);

glVertex3f(-2.25f, -0.05f, 0.2f);

glVertex3f(-2.5f, -0.05f, 0.2f);

glVertex3f(-2.25f, -0.05f, -0.3f);

glVertex3f(-2.0f, -0.05f, -0.3f);

glEnd();

// Top

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(-0.5, -0.5, 0.5);

glVertex3f(-1.0f, -0.15f, -0.1f);

glVertex3f(-1.0f, 0.15f, -0.1f);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-1.0f, -0.15f, -0.1f);

glVertex3f(-2.0f, -0.05f, -0.1f);

glVertex3f(-2.0f, 0.05f, -0.1f);

glVertex3f(-1.0f, 0.15f, -0.1f);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-2.0f, -0.05f, -0.1f);

glVertex3f(-2.25f, -0.05f, 0.2f);

glVertex3f(-2.25f, 0.05f, 0.2f);

glVertex3f(-2.0f, 0.05f, -0.1f);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-2.25f, -0.05f, 0.2f);

glVertex3f(-2.5f, -0.05f, 0.2f);

glVertex3f(-2.5f, 0.05f, 0.2f);

glVertex3f(-2.25f, 0.05f, 0.2f);

glEnd();

glColor3f(1,0,0);

// Bottom

glBegin(GL\_POLYGON);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, -0.5, -0.5);

glVertex3f(-2.0f, -0.05f, -0.3f);

glVertex3f(-2.0f, 0.05f, -0.3f);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-2.0f, -0.05f, -0.3f);

glVertex3f(-2.25f, -0.05f, -0.3f);

glVertex3f(-2.25f, 0.05f, -0.3f);

glVertex3f(-2.0f, 0.05f, -0.3f);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(-2.25f, -0.05f, -0.3f);

glVertex3d(-2.5, -0.05, 0.2);

glVertex3f(-2.5f, 0.05f, 0.2f);

glVertex3f(-2.25f, 0.05f, -0.3f);

glEnd();

glPopMatrix();

}

// Draw helicopter - Rotor

void rotor()

{

//overhear rotor

glPushMatrix();

glTranslated(0.0,0.0,0.65);

glRotated(angle, 0.0, 0.0, 1.0);

glScaled(4.0, 0.25, 0.01);

glColor3f(0.5, 0.5, 0.5);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(0.0,0.0,0.65);

glRotated(angle, 0.0, 0.0, 1.0);

glScaled(0.25, 4.0, 0.01);

glColor3f(0.5, 0.5, 0.5);

glutSolidCube(1);

glPopMatrix();

//Tail rotor

glPushMatrix();

glTranslated(-2.3, 0.06, 0.08);

glRotated(angle, 0.0, 1.0, 0.0);

glScaled(0.65, 0.01, 0.08);

glColor3f(0.5, 0.5, 0.5);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();

glTranslated(-2.3, 0.06, 0.08);

glRotated(angle, 0.0, 1.0, 0.0);

glScaled(0.08, 0.01, 0.65);

glColor3f(0.5, 0.5, 0.5);

glutSolidCube(1);

glPopMatrix();

// Draw - Rotor axle

glPushMatrix();

glTranslated(0.0,0.0,0.5);

glColor3f(0.5,0.5,0.5);

quad = gluNewQuadric();

gluCylinder(quad,0.15,0.15,0.15,10,10);

glPopMatrix();

}

//Draw front cabin

void cabin (void)

{

glPushMatrix();

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3d(0.5, -0.5, 0.5);

glVertex3d(0.5, 0.5, 0.5);

glVertex3f(0.8f, 0.5f, 0.4f);

glVertex3f(0.8f, -0.5f, 0.4f);

glEnd();

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3d(1.2, 0.5, 0.0);

glVertex3d(1.2, -0.5, 0.0);

glVertex3f(1.4f, -0.5f, -0.1f);

glVertex3f(1.4f, 0.5f, -0.1f);

glEnd();

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3f(1.4f, -0.5f, -0.1f);

glVertex3f(1.4f, 0.5f, -0.1f);

glVertex3d(1.4, 0.5, -0.4);

glVertex3d(1.4, -0.5, -0.4);

glEnd();

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3f(1.4f, -0.5f, -0.4f);

glVertex3f(1.4f, 0.5f, -0.4f);

glVertex3d(1.2, 0.5, -0.5);

glVertex3d(1.2, -0.5, -0.5);

glEnd();

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3d(1.2, 0.5, -0.5);

glVertex3d(1.2, -0.5, -0.5);

glVertex3f(0.5f, -0.5f, -0.5f);

glVertex3f(0.5f, 0.5f, -0.5f);

glEnd();

// Positive y side

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3d(1.0, 0.5, 0.2);

glVertex3f(1.2f, 0.5f, 0.0f);

glVertex3f(1.4f, 0.5f, -0.1f);

glVertex3d(1.4, 0.5, -0.4);

glVertex3d(1.2, 0.5, -0.5);

glVertex3d(1.0, 0.5, -0.5);

glEnd();

// Negative y side

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3d(1.0, -0.5, 0.2);

glVertex3f(1.2f, -0.5f, 0.0f);

glVertex3f(1.4f, -0.5f, -0.1f);

glVertex3d(1.4, -0.5, -0.4);

glVertex3d(1.2, -0.5, -0.5);

glVertex3d(1.0, -0.5, -0.5);

glEnd();

glPopMatrix();

}

void skids(void)

{

// Skid legs - front

glPushMatrix();

glTranslated(0.8,-0.45,-0.75);

glColor3f(0.5,0.5,0.5);

quad = gluNewQuadric();

//gluCylinder(quad,0.03,0.03,0.25,10,10);

glPopMatrix();

glPushMatrix();

glTranslated(0.8,0.45,-0.75);

glColor3f(0.5,0.5,0.5);

quad = gluNewQuadric();

gluCylinder(quad,0.03,0.03,0.25,10,10);

glPopMatrix();

// Skid legs - back

glPushMatrix();

glTranslated(-0.4,-0.45,-0.75);

glColor3f(0.5,0.5,0.5);

quad = gluNewQuadric();

gluCylinder(quad,0.03,0.03,0.25,10,10);

glPopMatrix();

glPushMatrix();

glTranslated(-0.4,0.45,-0.75);

glColor3f(0.5,0.5,0.5);

quad = gluNewQuadric();

gluCylinder(quad,0.03,0.03,0.25,10,10);

glPopMatrix();

// Bottom of skids

glPushMatrix();

glRotated(90, 0, 1, 0);

glTranslated(0.75,-0.45,-0.75);

glColor3f(0.5,0.5,0.5);

quad = gluNewQuadric();

gluCylinder(quad,0.03,0.03,2.0,10,10);

glPopMatrix();

glPushMatrix();

glRotated(90, 0, 1, 0);

glTranslated(0.75,0.45,-0.75);

glColor3f(0.5,0.5,0.5);

quad = gluNewQuadric();

gluCylinder(quad,0.03,0.03,2.0,10,10);

glPopMatrix();

}

// Drawing all transparent objects

void transparentObj(void)

{

glPushMatrix();

glDepthMask(GL\_FALSE);

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

// Doors

// Positive y side

glColor4d(.8 ,0.8 ,0.8, 0.9);

glBegin(GL\_POLYGON);

glVertex3d(0.5, 0.5, 0.5);

glVertex3d(0.8, 0.5, 0.4);

glVertex3d(1.0, 0.5, 0.2);

glVertex3d(1.0, 0.5, -0.5);

glVertex3d(0.5, 0.5, -0.5);

glEnd();

// Negative y side

glColor4d(0.8 ,0.8 ,0.8, 0.9);

glBegin(GL\_POLYGON);

glVertex3d(0.5, -0.5, 0.5);

glVertex3d(0.8, -0.5, 0.4);

glVertex3d(1.0, -0.5, 0.2);

glVertex3d(1.0, -0.5, -0.5);

glVertex3d(0.5, -0.5, -0.5);

glEnd();

// Windscreen

glColor4d(0.0, 1.0, 1.0, 0.5);

glBegin(GL\_POLYGON);

glVertex3f(0.8f, -0.5f, 0.4f);

glVertex3f(0.8f, 0.5f, 0.4f);

glVertex3d(1.2, 0.5, 0.0);

glVertex3d(1.2, -0.5, 0.0);

glEnd();

glDisable(GL\_BLEND);

glDepthMask(GL\_TRUE);

glPopMatrix();

}

// Main drawing function

void display(void)

{

//GLUquadric \*quad;

glLoadIdentity();

gluLookAt(eye.x,eye.y,eye.z, 0, 0, 0, 0.0, 1.0, 0.0);

//gluLookAt(30.0, 30.0, 30.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 0.0);

//ground(50.0);

// Helicopter's line of movement

glPushMatrix();

glTranslatef(pos.x, pos.y, pos.z); // x, y, z axis

glRotatef(spin, 0.0, 1.0, 0.0); // Rotate around the y axis

// Adding 5 helicoptor componants to build objcet

glTranslated(0, 3, 0);

glRotated(-90, 1, 0, 0);

mainBody();

rotor();

cabin();

skids();

transparentObj();

glPopMatrix();

display\_stars();

glutSwapBuffers();

}

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

{

//explicit viewer movements

if(key=='x') {eye.x+=1.0;}

if(key=='X') {eye.x-=1.0;}

if(key=='y') {eye.y+=1.0;}

if(key=='Y') {eye.y-=1.0;}

if(key=='z') {eye.z+=1.0;}

if(key=='Z') {eye.z-=1.0;}

if(key=='r') {spin++;}

if(key=='R') {spin--;}

if(key=='s') {spinning=true; }

if(key=='S') {spinning=false;}

}

void special1(int key,int x,int y)

{

if(spinning)

{

if(key==GLUT\_KEY\_UP) {pos.x+=.05;}

if(key==GLUT\_KEY\_LEFT) {pos.z+=0.05;}

if(key==GLUT\_KEY\_RIGHT) {pos.z-=0.05;}

}

}

void special2(int key,int x,int y)

{

if(spinning)

{

if(key==GLUT\_KEY\_UP) {pos.x+=.1;}

if(key==GLUT\_KEY\_LEFT) {pos.z+=0.1;}

if(key==GLUT\_KEY\_RIGHT) {pos.z-=0.1;}

}

}

void special4(int key,int x,int y)

{

if(spinning)

{

if(key==GLUT\_KEY\_UP) {pos.x+=.2;}

if(key==GLUT\_KEY\_LEFT) {pos.z+=0.2;}

if(key==GLUT\_KEY\_RIGHT) {pos.z-=0.2;}

}

}

void my\_menu(int id)

{

switch(id)

{

case 1: spinning=true;

break;

case 3:spinning=false;

break;

case 4:exit(0);

}

glutPostRedisplay();

}

void speed\_menu(int id)

{

switch(id)

{

case 1: glutSpecialFunc(special1);

break;

case 2: glutSpecialFunc(special2);

break;

case 3: glutSpecialFunc(special4);

}

glutPostRedisplay();

}

void rotate\_menu(int id)

{

switch(id)

{

case 1:spin=90.0;

break;

case 2:spin=-90.0;

break;

case 3:spin=0.0;

break;

}

glutPostRedisplay();

}

void reshape(int w, int h)

{

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(fovy, (GLfloat) w/(GLfloat) h, 1.5, 300);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB |GLUT\_DEPTH);

glutInitWindowSize(800, 800);

glutInitWindowPosition(100,100);

glutCreateWindow("Helicopter");

init();

glutIdleFunc(idle);

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glutSpecialFunc(special1);

glutReshapeFunc (reshape);

sub\_menu[0] =glutCreateMenu(speed\_menu);

glutAddMenuEntry("Speed 1x",1);

glutAddMenuEntry("Speed 2x",2);

glutAddMenuEntry("Speed 4x",3);

sub\_menu[1] =glutCreateMenu(rotate\_menu);

glutAddMenuEntry("90 degree",1);

glutAddMenuEntry("-90 degree",2);

glutAddMenuEntry("0 degree",3);

glutCreateMenu(my\_menu);

glutAddMenuEntry("START",1);

glutAddSubMenu("Speed",sub\_menu[0]);

glutAddSubMenu("Rotate",sub\_menu[1]);

glutAddMenuEntry("STOP",3);

glutAddMenuEntry("Exit",4);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutMainLoop();

return 0;

}