#include<string.h>

#include<stdio.h>

#include <GL/glut.h>

// Rotation amounts

static GLfloat xRot = 90.0f;

static GLfloat yRot = 180.0f;

static GLfloat pxTra = 0.0f;

static GLfloat nxTra = 0.0f;

static int flag = 0;

// Called to draw scene

void RenderScene(void)

{

// Clear the window with current clearing color

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

// Save matrix state and do the rotation

glPushMatrix();

glRotatef(xRot, 1.0f, 0.0f, 0.0f);

glRotatef(yRot, 0.0f, 1.0f, 0.0f);

glTranslatef(nxTra, 0.0, 0.0);

glTranslatef(pxTra, 0.0, 0.0);

// Nose Cone /////////////////////////////

glBegin(GL\_TRIANGLES);

//Red Color

glColor3ub(205, 0, 0);

glVertex3f(0.0f, 0.0f, 60.0f);

glVertex3f(-15.0f, 0.0f, 30.0f);

glVertex3f(15.0f, 0.0f, 30.0f);

//Red Color

glColor3ub(245, 0, 0);

glVertex3f(15.0f, 0.0f, 30.0f);

glVertex3f(0.0f, 15.0f, 30.0f);

glVertex3f(0.0f, 0.0f, 60.0f);

//Red Color

glColor3ub(255, 0, 0);

glVertex3f(0.0f, 0.0f, 60.0f);

glVertex3f(0.0f, 15.0f, 30.0f);

glVertex3f(-15.0f, 0.0f, 30.0f);

// Body of the Plane ////////////////////////

glColor3ub(128, 128, 128);

glVertex3f(-15.0f, 0.0f, 30.0f);

glVertex3f(0.0f, 15.0f, 30.0f);

glVertex3f(0.0f, 0.0f, -56.0f);

glColor3ub(140, 140, 140);

glVertex3f(0.0f, 0.0f, -56.0f);

glVertex3f(0.0f, 15.0f, 30.0f);

glVertex3f(15.0f, 0.0f, 30.0f);

glColor3ub(130, 130, 130);

glVertex3f(15.0f, 0.0f, 30.0f);

glVertex3f(-15.0f, 0.0f, 30.0f);

glVertex3f(0.0f, 0.0f, -56.0f);

///////////////////////////////////////////////

// Left wing

// Large triangle for bottom of wing

glColor3ub(128, 128, 128);

glVertex3f(0.0f, 2.0f, 27.0f);

glVertex3f(-60.0f, 2.0f, -8.0f);

glVertex3f(60.0f, 2.0f, -8.0f);

glColor3ub(64, 64, 64);

glVertex3f(60.0f, 2.0f, -8.0f);

glVertex3f(0.0f, 7.0f, -8.0f);

glVertex3f(0.0f, 2.0f, 27.0f);

glColor3ub(192, 192, 192);

glVertex3f(60.0f, 2.0f, -8.0f);

glVertex3f(-60.0f, 2.0f, -8.0f);

glVertex3f(0.0f, 7.0f, -8.0f);

// Other wing top section

glColor3ub(64, 64, 64);

glVertex3f(0.0f, 2.0f, 27.0f);

glVertex3f(0.0f, 7.0f, -8.0f);

glVertex3f(-60.0f, 2.0f, -8.0f);

// Tail section///////////////////////////////

// Bottom of back fin

glColor3ub(128, 128, 128);

glVertex3f(-30.0f, -0.50f, -57.0f);

glVertex3f(30.0f, -0.50f, -57.0f);

glVertex3f(0.0f, -0.50f, -40.0f);

// top of left side

glColor3ub(128, 128, 0);

glVertex3f(0.0f, -0.5f, -40.0f);

glVertex3f(30.0f, -0.5f, -57.0f);

glVertex3f(0.0f, 4.0f, -57.0f);

// top of right side

glColor3ub(128, 128, 0);

glVertex3f(0.0f, 4.0f, -57.0f);

glVertex3f(-30.0f, -0.5f, -57.0f);

glVertex3f(0.0f, -0.5f, -40.0f);

// back of bottom of tail

glColor3ub(128, 0, 0);

glVertex3f(30.0f, -0.5f, -57.0f);

glVertex3f(-30.0f, -0.5f, -57.0f);

glVertex3f(0.0f, 4.0f, -57.0f);

// Top of Tail section left

glColor3ub(128, 0, 0);

glVertex3f(0.0f, 0.5f, -40.0f);

glVertex3f(3.0f, 0.5f, -57.0f);

glVertex3f(0.0f, 25.0f, -65.0f);

glColor3ub(117, 0, 0);

glVertex3f(0.0f, 25.0f, -65.0f);

glVertex3f(-3.0f, 0.5f, -57.0f);

glVertex3f(0.0f, 0.5f, -40.0f);

// Back of horizontal section

glColor3ub(128, 128, 128);

glVertex3f(3.0f, 0.5f, -57.0f);

glVertex3f(-3.0f, 0.5f, -57.0f);

glVertex3f(0.0f, 25.0f, -65.0f);

glEnd(); // End Of Jet

glPopMatrix();

//To display controls used for jet movement

if (flag)

{

char string[] = "z--->Move Left x--->Move Right n--->Original Position";

char string1[] = "Arrow Keys:";

char string2[] = "UP --->Clockwise UP Rotation";

char string3[] = "DOWN--->Anticlockwise DOWN Rotation";

char string4[] = "LEFT--->Clockwise LEFT Rotation";

char string5[] = "RIGHT-->Anticlockwise RIGHT Rotation";

int i = 0;

void \*font = GLUT\_BITMAP\_TIMES\_ROMAN\_24;

glColor3f(0.0, 0.0, 0.0);

glRasterPos3f(-100, 70, 0);

for (i = 0; i<strlen(string); i++)

glutBitmapCharacter(font, string[i]);

glRasterPos3f(-100, 65, 0);

for (i = 0; i<strlen(string1); i++)

glutBitmapCharacter(font, string1[i]);

glRasterPos3f(-100, 60, 0);

for (i = 0; i<strlen(string2); i++)

glutBitmapCharacter(font, string2[i]);

glRasterPos3f(-100, 55, 0);

for (i = 0; i<strlen(string3); i++)

glutBitmapCharacter(font, string3[i]);

glRasterPos3f(-100, 50, 0);

for (i = 0; i<strlen(string4); i++)

glutBitmapCharacter(font, string4[i]);

glRasterPos3f(-100, 45, 0);

for (i = 0; i<strlen(string5); i++)

glutBitmapCharacter(font, string5[i]);

}

// Display the results

glutSwapBuffers();

}

// This function does any needed initialization on the rendering

// context.

void SetupRC()

{

// Light values and coordinates

GLfloat ambientLight[] = { 0.3f, 0.3f, 0.3f, 1.0f };

GLfloat diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };

GLfloat specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

GLfloat specref[] = { 1.0f, 1.0f, 1.0f, 1.0f };

glEnable(GL\_DEPTH\_TEST); // Hidden surface removal

glFrontFace(GL\_CCW); // Counter clock-wise polygons face out

glEnable(GL\_CULL\_FACE); // Do not calculate inside of jet

// Enable lighting

glEnable(GL\_LIGHTING);

// Setup and enable light 0

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambientLight);

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

glLightfv(GL\_LIGHT0, GL\_SPECULAR, specular);

glEnable(GL\_LIGHT0);

// Enable color tracking

glEnable(GL\_COLOR\_MATERIAL);

// Set Material properties to follow glColor values

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

// All materials hereafter have full specular reflectivity

// with a high shine

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specref);

glMateriali(GL\_FRONT, GL\_SHININESS, 128);

// Light blue background

glClearColor(0.0f, 0.0f, 1.0f, 1.0f);

}

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

{

if (key == GLUT\_KEY\_UP)

{

xRot -= 2.0f; flag = 0;

}

if (key == GLUT\_KEY\_DOWN)

{

xRot += 2.0f; flag = 0;

}

if (key == GLUT\_KEY\_LEFT)

{

yRot -= 2.0f; flag = 0;

}

if (key == GLUT\_KEY\_RIGHT)

{

yRot += 2.0f; flag = 0;

}

if (key > 358.0f)

xRot = 0.0f;

if (key < -1.0f)

xRot = 358.0f;

if (key > 358.0f)

yRot = 0.0f;

if (key < -1.0f)

yRot = 358.0f;

// Refresh the Window

glutPostRedisplay();

}

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

{

if (key == 'x')

{

pxTra -= 1.0; flag = 0;

}

if (key == 'z')

{

nxTra += 1.0; flag = 0;

}

if (key == 'n')

{

pxTra = 0; nxTra = 0; flag = 0;

}

RenderScene();

}

void ChangeSize(int w, int h)

{

GLfloat nRange = 80.0f;

// Prevent a divide by zero

if (h == 0)

h = 1;

// Set Viewport to window dimensions

glViewport(0, 0, w, h);

// Reset coordinate system

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

// Establish clipping volume (left, right, bottom, top, near, far)

if (w <= h)

glOrtho(-nRange, nRange, -nRange\*h / w, nRange\*h / w, -nRange, nRange);

else

glOrtho(-nRange\*w / h, nRange\*w / h, -nRange, nRange, -nRange, nRange);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void control\_menu(int id1)

{

switch (id1)

{

case 1: flag = 1;

break;

case 2: flag = 0;

break;

}

}

void menu(int id2)

{

switch (id2)

{

case 8:exit(0);

break;

}

}

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

{

int submenu1, submenu2;

glutInit(&argc, argv);

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

glutCreateWindow("Jet");

glutReshapeFunc(ChangeSize);

glutSpecialFunc(SpecialKeys);

glutKeyboardFunc(Mykeys);

submenu1 = glutCreateMenu(control\_menu);

glutAddMenuEntry("Display", 1);

// glutAddMenuEntry("Cancal",2);

glutCreateMenu(menu);

glutAddSubMenu("Controls", submenu1);

glutAddMenuEntry("Quit", 8);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutDisplayFunc(RenderScene);

SetupRC();

glutMainLoop();

return 0;

}