/\*robot arm three parts animated \*/

/\* rotate base : press b (to stop, press b again) \*/

/\* rotate lower arm : press l (to stop, press l again) \*/

/\* rotate upper arm : press u (to stop, press u again) \*/

#include <stdlib.h>

#include <GL/glut.h>

#define TORSO\_RADIUS 2.0

#define TORSO\_LENGTH 2.5

#define LEG\_RADIUS 0.7

#define LEG\_LENGTH 2.5

#define HEAD\_RADIUS 2

GLUquadricObj \*p; // pointer to quadric object

GLfloat theta[4] = {0.0, 0.0, 0.0,0.0};

GLboolean torsoj = GL\_FALSE; //judge

GLboolean leg1j = GL\_FALSE;

GLboolean leg2j = GL\_FALSE;

GLboolean headj = GL\_FALSE;

GLint t\_direction = 1;

GLint l\_direction = 1;

GLint h\_direction = 1;

void torso()

{

glPushMatrix();

glScalef(TORSO\_RADIUS,TORSO\_RADIUS, TORSO\_LENGTH);

glutSolidSphere(1,10,10);

glPopMatrix();

}

void leg()

{

glPushMatrix();

glTranslatef(0.0,-LEG\_LENGTH,0.0);

glScalef(LEG\_RADIUS, LEG\_LENGTH, LEG\_RADIUS);

glRotatef (-90.0, 1.0, 0.0, 0.0);

gluCylinder (p, 1, 1, 1, 20, 20);

glPopMatrix();

glPushMatrix();

glTranslatef(0.0,-LEG\_LENGTH,0.0);

glScalef(LEG\_RADIUS, LEG\_RADIUS, LEG\_RADIUS);

glutSolidSphere(1,10,10);

glPopMatrix();

}

void head()

{

glPushMatrix();

glTranslatef(0.0,0.0,TORSO\_RADIUS);

glRotatef(-10, 1.0, 0.0, 0.0);

glScalef(HEAD\_RADIUS, HEAD\_RADIUS+0.3, HEAD\_RADIUS-0.3);

glutSolidSphere(1,20,20);

glPopMatrix();

}

void display()

{

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glRotatef(theta[0], 0.0, 1.0, 0.0);//大家一起转

glColor3f (0.0, 0.0, 1.0);

torso();

glPushMatrix();

glTranslatef(0.5\*TORSO\_RADIUS, 0.0, 0.5\*TORSO\_LENGTH);//leg1的位置

glRotatef(10, 0.0, 0.0, 1.0);//倾斜10度

glRotatef(theta[1], 1.0, 0.0, 0.0);

glColor3f (0.0, 1.0, 0.0);

leg();

glPopMatrix();

glPushMatrix();

glTranslatef(-0.5\*TORSO\_RADIUS, 0.0, 0.5\*TORSO\_LENGTH);//leg2的位置

glRotatef(-10, 0.0, 0.0, 1.0);//倾斜10度

glRotatef(theta[2], 1.0, 0.0, 0.0);

glColor3f (0.0, 1.0, 0.0);

leg();

glPopMatrix();

glPushMatrix();

glTranslatef(-0.5\*TORSO\_RADIUS, 0.0, -0.5\*TORSO\_LENGTH);//leg3的位置

glRotatef(-10, 0.0, 0.0, 1.0);//倾斜10度

glRotatef(theta[1], 1.0, 0.0, 0.0);

glColor3f (0.0, 1.0, 0.0);

leg();

glPopMatrix();

glPushMatrix();

glTranslatef(0.5\*TORSO\_RADIUS, 0.0, -0.5\*TORSO\_LENGTH);//leg4的位置

glRotatef(10, 0.0, 0.0, 1.0);//倾斜10度

glRotatef(theta[2], 1.0, 0.0, 0.0);

glColor3f (0.0, 1.0, 0.0);

leg();

glPopMatrix();

glPushMatrix();

glTranslatef(0.0, TORSO\_RADIUS, 0.0);//坐标归位

glTranslatef(0.0, 0.0, 0.5\*TORSO\_LENGTH);//将坐标前(z)移动0.5 TORSO\_LENGTH

glRotatef(theta[3], 0.0, 0.0, 1.0);

glColor3f (1.0, 0.0, 0.0);

head();

glPopMatrix();

glutSwapBuffers();

}

void init()

{

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

p = gluNewQuadric();

}

void reshape(int w, int h)

{

glViewport (0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-7.0, 7.0, -7.0, 7.0, -10.0, 10.0);

}

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

{

if(key == 't' | key == 'T')

torsoj = !torsoj;

if(key == 'l' | key == 'L')

leg1j = !leg1j;

if(key == 'l' | key == 'L')

leg2j = !leg2j;

if(key == 'h' | key == 'H')

headj = !headj;

}

void idle()

{

if (torsoj) {

theta[0] += 2;

if (theta[0] > 360) theta[0] -= 360;

}

//torso

if (leg1j) {

if (l\_direction > 0) {

theta[1] += 2;

if (theta[1] > 15) {

l\_direction = -1;

theta[1] -= 2;

}

}

else {

theta[1] -= 2;

if (theta[1] < -15) {

l\_direction = 1;

theta[1] += 2;

}

}

}

//leg1

if (leg2j) {

if (l\_direction > 0) {

theta[2] -= 2;

if (theta[2] > 15) {

l\_direction = -1;

theta[2] -= 2;

}

}

else {

theta[2] += 2;

if (theta[2] < -15) {

l\_direction = 1;

theta[2] -= 2;

}

}

}

//leg2

if (headj) {

if (h\_direction > 0) {

theta[3] += 2;

if (theta[3] > 70) {

h\_direction = -1;

theta[3] -= 2;

}

}

else {

theta[3] -= 2;

if (theta[3] < -70) {

h\_direction = 1;

theta[3] += 2;

}

}

}

//head

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

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

glutInitWindowSize(500, 500);

glutCreateWindow("cow");

glutKeyboardFunc(mykey);

glutIdleFunc(idle);

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glEnable(GL\_DEPTH\_TEST); /\* Enable hidden-surface removal \*/

init();

glutMainLoop();

}