/\*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 1.7

#define TORSO\_LENGTH 2.5

#define LEG\_RADIUS 0.7

#define LEG\_LENGTH 2.5

#define HEAD\_RADIUS 1.7

#define RING\_OUTSIDE\_RADIUS 0.7

#define RING\_INSIDE\_RADIUS 0.07

#define HORN\_LENGTH 1.3

#define HORN\_RADIUS 0.25

#define TAIL\_RADIUS 0.10

#define TAIL\_LENGTH 1.25

#define EYE\_RADIUS 0.10

#define EYE\_GAP 0.40

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;

GLboolean movej = GL\_FALSE;

GLint t\_direction = 1;

GLint l\_direction = 1;

GLint h\_direction = 1;

GLint m\_direction = 1;

GLdouble move[] = {0.0, 0.0, 0.0};

GLdouble m=0.1;

GLint menu\_id = 0;

GLint submenu\_id = 0;

void tail()

{

glRotatef (45.0, 1.0, 0.0, 0.0);

glTranslatef(0.0,0.3,-TAIL\_LENGTH-0.1);

glPushMatrix();

glColor3ub (111, 111,111);

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

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

glPopMatrix();

//ball in the tail

glPushMatrix();

glColor3ub (200, 200, 200);

glScalef(2.5\*TAIL\_RADIUS,2.5\*TAIL\_RADIUS,2.5\*TAIL\_RADIUS);

glutSolidSphere(1,10,10);

glPopMatrix();

}

void torso()

{

glPushMatrix();

glColor3ub (200, 200, 200);

glScalef(TORSO\_RADIUS+0.2,TORSO\_RADIUS, TORSO\_LENGTH);

glutSolidSphere(1,10,10);

glPopMatrix();

}

void leg()

{

//leg

glPushMatrix();

glColor3ub (111, 111, 111);

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();

//feet

glPushMatrix();

glColor3ub (200, 200, 200);

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

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

glutSolidSphere(1,10,10);

glPopMatrix();

}

void head()

{

glPushMatrix();

glColor3ub (111, 111, 111);

glTranslatef(0.0,-0.3,1.3\*HEAD\_RADIUS);

glScalef(HEAD\_RADIUS-0.1, HEAD\_RADIUS, HEAD\_RADIUS-0.2);

glutSolidSphere(1,20,20);

glPopMatrix();

//right eye

glPushMatrix();

glColor3ub (0.0, 0.0, 0.0);

glTranslatef(-EYE\_GAP,0.2\*HEAD\_RADIUS,2.1\*HEAD\_RADIUS);

glScalef(EYE\_RADIUS, EYE\_RADIUS, EYE\_RADIUS);

glutSolidSphere(1,10,10);

glPopMatrix();

//left eye

glPushMatrix();

glColor3ub (0.0, 0.0, 0.0);

glTranslatef(EYE\_GAP,0.2\*HEAD\_RADIUS,2.1\*HEAD\_RADIUS);

glScalef(EYE\_RADIUS, EYE\_RADIUS, EYE\_RADIUS);

glutSolidSphere(1,10,10);

glPopMatrix();

//ring in the nose

glPushMatrix();

glColor3ub (255, 255, 0.0);

glTranslatef(0.0,-0.8\*HEAD\_RADIUS,2.2\*HEAD\_RADIUS);

glRotatef (-20.0, 1.0, 0.0, 0.0);

glutSolidTorus(RING\_INSIDE\_RADIUS,RING\_OUTSIDE\_RADIUS,40, 40);

glPopMatrix();

//horn

glPushMatrix();

glColor3f (1.0, 1.0, 0.0);

glTranslatef(-0.5\*HORN\_LENGTH,0.7\*HEAD\_RADIUS,1.7\*HEAD\_RADIUS);

glRotatef (90.0, 0.0, 1.0, 0.0);

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

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

glPopMatrix();

glPushMatrix();

glColor3f (1.0, 1.0, 0.0);

glTranslatef(0.5\*HORN\_LENGTH,0.7\*HEAD\_RADIUS,1.7\*HEAD\_RADIUS);

glRotatef (50.0, 0.0, 1.0, 0.0);

glutSolidCone(HORN\_RADIUS, 1\*HORN\_LENGTH, 20,20);

glPopMatrix();

glPushMatrix();

glColor3f (1.0, 1.0, 0.0);

glTranslatef(-0.5\*HORN\_LENGTH,0.7\*HEAD\_RADIUS,1.7\*HEAD\_RADIUS);

glRotatef (-50.0, 0.0, 1.0, 0.0);

glutSolidCone(HORN\_RADIUS, 1\*HORN\_LENGTH, 20,20);

glPopMatrix();

}

void object()

{

glTranslatef(move[0], move[1], move[2]);//object move

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

glColor3f (0.0, 0.0, 1.0);

torso();

glPushMatrix();

glTranslatef(0.0, 0.0, -TORSO\_LENGTH);//tail的位置

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

tail();

glPopMatrix();

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);

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);

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();

glRotatef (-20.0, 1.0, 0.0, 0.0);//整个抬头

glTranslatef(0.0, -0.7\*HEAD\_RADIUS, 0.5\*TORSO\_LENGTH);//将坐标前(z)移动0.5 TORSO\_LENGTH

glRotatef(theta[3], 0.0, 0.0, 1.0);//顺序是从上到下，先前移再转

head();

glPopMatrix();

}

void mymenu(int id)

{

if(id == 1) {

movej=!movej;

}

if(id == 2) {

movej=!movej;

}

if(id == 3) {

torsoj=!torsoj;

}

}

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);

object();

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 == 'r' | key == 'R') //Rotating

torsoj = !torsoj;

if(key == 'l' | key == 'L') //leg1 move

leg1j = !leg1j;

if(key == 'l' | key == 'L') //leg2 move

leg2j = !leg2j;

if(key == 'h' | key == 'H') //head&tail rotate

headj = !headj;

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

movej = !movej;

}

void mysubmenu(int id)

{

if(id == 1) leg1j=!leg1j;

if(id == 1) leg2j=!leg2j;

if(id == 2) headj=!headj;

}

void menu\_system(){

submenu\_id = glutCreateMenu(mysubmenu);

glutAddMenuEntry("Leg move(l)", 1);

glutAddMenuEntry("head and tail move(h)", 2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

menu\_id = glutCreateMenu(mymenu);

glutAddMenuEntry("Translating(t)", 1);

glutAddMenuEntry("Rotating(r)", 3);

glutAddSubMenu ("Bending", submenu\_id);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void torso\_rotate(){

//torso rotate judgement

if (torsoj) {

theta[0] += 2;

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

}

}

void leg\_rotate(){

//leg1 rotate judgement

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;

}

}

}

//leg2 rotate judgement

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;

}

}

}

}

void head\_rotate(){

//head rotate judgement

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;

}

}

}

}

void object\_move(){

//object move judgement

if (movej) {

if (m\_direction > 0) {//initial m\_direction = 1;

move[2] += m;

if (move[2] > 5) {

m\_direction = -1;

move[2] -= m;

}

}

else {

move[2] -= m;

if (move[2] < -5) {

m\_direction = 1;

move[2] += m;

}

}

}

}

void idle()

{

torso\_rotate();

leg\_rotate();

head\_rotate();

object\_move();

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();

menu\_system();

glutMainLoop();

}