#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 legj = GL\_FALSE;

GLboolean headj = GL\_FALSE;

GLboolean movej = GL\_FALSE;

GLboolean relativelight = GL\_FALSE;

GLint t\_direction = 1;

GLint l\_direction = 1;

GLint h\_direction = 1;

GLint m\_direction = 1;

GLint light=3;// light=3 means current light is white light

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

GLdouble m=0.1;

// shading ...

typedef struct materialStruct {

GLfloat ambient[4];

GLfloat diffuse[4];

GLfloat specular[4];

GLfloat emission[4];

GLfloat shininess;

} materialStruct;

materialStruct MetalShinyMaterials = {

{0.33, 0.22, 0.03, 1.0},

{0.78, 0.57, 0.11, 1.0},

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

100.0

};

materialStruct BlackShinyMaterials = {

{0.1, 0.1, 0.1, 1.0},

{0.1, 0.3, 0.6, 1.0},

{0.0, 0.0, 0.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

2.0

};

materialStruct BlackPlasticMaterials = {

{0.1, 0.1, 0.1, 1.0},

{0.1, 0.5, 0.8, 1.0},

{0.5, 0.5, 0.5, 1.0},

{0.0, 0.0, 0.0, 1.0},

100.0

};

materialStruct BurningMaterials = {

{1.0, 0.6, 0.0, 1.0},

{1.0, 0.6, 0.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

100.0

};

typedef struct lightingStruct {

GLfloat ambient[4];

GLfloat diffuse[4];

GLfloat specular[4];

} lightingStruct;

lightingStruct whiteLighting = {

{0.0, 0.0, 0.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

{1.0, 1.0, 1.0, 1.0},

};

lightingStruct colouredLighting = {

{1.0, 1.0, 0.0, 1.0},

{0.7, 0.7, 0.7, 1.0},

{1.0, 1.0, 1.0, 1.0},

};

lightingStruct ambientLighting = {

{1.0, 1.0, 1.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

{0.0, 0.0, 0.0, 1.0},

};

lightingStruct distantLighting = {

{0.0, 0.0, 0.0, 0.0},

{1.0, 1.0, 1.0, 0.0},

{1.0, 1.0, 1.0, 0.0},

};

GLfloat light0\_pos[4] = {0.0, 0.0, 3.0, 0.0};

materialStruct \*currentMaterials;

lightingStruct \*currentLighting;

void currentLight(){

switch(light) {

case 1:

currentLighting = &ambientLighting;

break;

case 2:

currentLighting = &distantLighting;

break;

case 3:

currentLighting = &whiteLighting;

break;

case 4:

currentLighting = &colouredLighting;

break;

}

}

void lightsource(){

currentLight();

glLightfv(GL\_LIGHT0, GL\_AMBIENT, currentLighting -> ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, currentLighting -> diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, currentLighting -> specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light0\_pos);

glLightModeli(GL\_LIGHT\_MODEL\_TWO\_SIDE, 1);

}

void MetalShiny(){

currentMaterials = &MetalShinyMaterials;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, currentMaterials -> ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, currentMaterials -> diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, currentMaterials -> specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &currentMaterials -> shininess);

}

void BlackShiny(){

currentMaterials = &BlackShinyMaterials;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, currentMaterials -> ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, currentMaterials -> diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, currentMaterials -> specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &currentMaterials -> shininess);

}

void BlackPlastic(){

currentMaterials = &BlackPlasticMaterials;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, currentMaterials -> ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, currentMaterials -> diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, currentMaterials -> specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &currentMaterials -> shininess);

}

void Burning(){

currentMaterials = &BurningMaterials;

glMaterialfv(GL\_FRONT, GL\_AMBIENT, currentMaterials -> ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, currentMaterials -> diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, currentMaterials -> specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, &currentMaterials -> shininess);

}

void tail()

{

glRotatef (45.0, 1.0, 0.0, 0.0);

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

glPushMatrix();

BlackShiny();

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

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

glPopMatrix();

//ball in the tail

glPushMatrix();

MetalShiny();

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

glutSolidSphere(1,10,10);

glPopMatrix();

}

void torso()

{

glPushMatrix();

BlackPlastic();

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

glutSolidSphere(1,10,10);

glPopMatrix();

}

void leg()

{

//leg

glPushMatrix();

BlackShiny();

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();

Burning();

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

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

glutSolidSphere(1,10,10);

glPopMatrix();

}

void head()

{

glPushMatrix();

BlackPlastic();

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();

Burning();

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();

Burning();

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();

MetalShiny();

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();

MetalShiny();

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();

MetalShiny();

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

glRotatef (50.0, 0.0, 1.0, 0.0);

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

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

glPopMatrix();

glPushMatrix();

MetalShiny();

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

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

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

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

glPopMatrix();

}

void object()

{

glPushMatrix();

if (relativelight){

glRotatef(theta[0], 0.0, 1.0, 0.0);//object rotate along the Y axis

}

lightsource();

glPopMatrix();

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

glRotatef(theta[0], 0.0, 1.0, 0.0);//object rotate along the Y axis

torso();

glPushMatrix();

glTranslatef(0.0, 0.0, -TORSO\_LENGTH);//the porsition of the tail

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

tail();

glPopMatrix();

glPushMatrix();

glTranslatef(0.5\*TORSO\_RADIUS, 0.0, 0.5\*TORSO\_LENGTH);//the porsition of leg1

glRotatef(10, 0.0, 0.0, 1.0);//slope 10 degree

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

leg();

glPopMatrix();

glPushMatrix();

glTranslatef(-0.5\*TORSO\_RADIUS, 0.0, 0.5\*TORSO\_LENGTH);//the porsition of leg2

glRotatef(-10, 0.0, 0.0, 1.0);//slope 10 degree

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

leg();

glPopMatrix();

glPushMatrix();

glTranslatef(-0.5\*TORSO\_RADIUS, 0.0, -0.5\*TORSO\_LENGTH);//the porsition of leg3

glRotatef(-10, 0.0, 0.0, 1.0);//slope 10 degree

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);//the porsition of leg4

glRotatef(10, 0.0, 0.0, 1.0);//slope 10 degree

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);//rise the head

glTranslatef(0.0, -0.7\*HEAD\_RADIUS, 0.5\*TORSO\_LENGTH);//the porsition of the head

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

head();

glPopMatrix();

}

void display()

{

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

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

object();

glutSwapBuffers();

}

void init()

{

glEnable(GL\_NORMALIZE);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

glClearColor(0.0f, 0.0f, 0.0f, 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)

{

switch(key) {//fast keyboard

// case 'r':

// torsoj = !torsoj;

// break;

case 'l':

legj = !legj;

break;

case 'h':

headj = !headj;

break;

case 't':

movej = !movej;

break;

case 'f':

relativelight=!relativelight;

break;

case 'r':

relativelight=!relativelight;

break;

//Lighting key

case 'a':

light=1;

break;

case 'd':

light=2;

break;

case 'w':

light=3;

break;

case 'c':

light=4;

break;

}

}

void mymouse(int btn, int state, int x, int y)

{

if(btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN){

torsoj = !torsoj;

headj = !headj;

legj = !legj;

}

}

void stop(){

//all movement stop

}

void torso\_rotate(){

theta[0] += 2;

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

}

void leg\_rotate(){

//leg1 rotate

if (l\_direction > 0) {

theta[1] += 1;

if (theta[1] > 15) {

l\_direction = -1;

theta[1] -= 1;

}

}

else {

theta[1] -= 1;

if (theta[1] < -15) {

l\_direction = 1;

theta[1] += 1;

}

}

//leg2 rotate

if (l\_direction > 0) {

theta[2] -= 1;

if (theta[2] > 15) {

l\_direction = -1;

theta[2] -= 1;

}

}

else {

theta[2] += 1;

if (theta[2] < -15) {

l\_direction = 1;

theta[2] -= 1;

}

}

}

void head\_rotate(){

if (h\_direction > 0) {

theta[3] += 1;

if (theta[3] > 70) {

h\_direction = -1;

theta[3] -= 1;

}

}

else {

theta[3] -= 1;

if (theta[3] < -70) {

h\_direction = 1;

theta[3] += 1;

}

}

}

void object\_move(){

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;

}

}

leg\_rotate();

head\_rotate();

}

void idle()

{

if (torsoj){

torso\_rotate();

}

else {

stop();

}

if (legj){

leg\_rotate();

}

else {

stop();

}

if (headj){

head\_rotate();

}

else {

stop();

}

if (movej){

object\_move();

}

else {

stop();

}

glutPostRedisplay();

}

void mymenu(int id){

}

void mysubmenu1(int id){

if(id == 1) {

relativelight=!relativelight;

}

if(id == 2) {

relativelight=!relativelight;

}

}

void mysubmenu2(int id){

if(id == 1) {

light=1;//ambient light

}

if(id == 2) {

light=2;//distant light

}

}

void mysubmenu3(int id){

if(id == 1) {

light=3;//white light

}

if(id == 2) {

light=4;//coloured light

}

}

void menu\_system(){

GLint menu\_id = 0;

GLint submenu\_1 = 0;

GLint submenu\_2 = 0;

GLint submenu\_3 = 0;

submenu\_1 = glutCreateMenu(mysubmenu1);

glutAddMenuEntry("fixed(f)", 1);

glutAddMenuEntry("relative(r)", 2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

submenu\_2 = glutCreateMenu(mysubmenu2);

glutAddMenuEntry("ambient(a)", 1);

glutAddMenuEntry("distant(d)", 2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

submenu\_3 = glutCreateMenu(mysubmenu3);

glutAddMenuEntry("white(w)", 1);

glutAddMenuEntry("coloured(r)", 2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

menu\_id = glutCreateMenu(mymenu);

glutAddSubMenu("light position", submenu\_1);

glutAddSubMenu("light type", submenu\_2);

glutAddSubMenu("light properties", submenu\_3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

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

{

glutInit(&argc, argv);

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

glutInitWindowSize(500, 500);

glutCreateWindow("cow");

glutKeyboardFunc(mykey);

glutMouseFunc(mymouse);

glutIdleFunc(idle);

glutDisplayFunc(display);

glutReshapeFunc(reshape);

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

menu\_system();

init();

glutMainLoop();

}