#include<stdlib.h>

#include<math.h>

#include<time.h>

#include<string.h>

#include<GL/glut.h>

GLint countx, county;

void movmt();

void drawroad();

int flag = 0, flagw = 1;

GLint count = 3;

float level = 0, l;

GLfloat cv[][3] = { { -1.0,-0.75,0 },{ -1.25,-0.65,0.0 },{ -1.25,0.65,0 },{ -1.0,0.75,0.0 },{ 0.8,0.75,0.0 },{ 1.25,0.5,0.0 },{ 1.25,-0.5,0.0 },{ 0.8,-0.75,0.0 } };

GLfloat cs[][3] = { { -1.35,-0.5,0 },{ -1.35,0.5,0 },{ -1.25,0.5,0 },{ -1.25,-0.5,0 } };

GLfloat mrr[][3] = { { 0.15,-0.6,0 },{ 0.15,0.6,0 },{ 0.7,0.6,0 },{ 0.7,-0.6,0 } };

GLfloat light[][3] = { { 0,0,0 },{ 0,0.12,0 },{ 0.12,0.12,0 },{ 0.12,0,0 } };

GLfloat w[][3] = { { 0,0,0 },{ 0,0.12,0 },{ 0.33,0.12,0 },{ 0.33,0,0 } };

GLfloat rv[4][3] = { { -15.0,0.0,0.0 },{ -15.0,2.0,0.0 },{ 15.0,2.0,0.0 },{ 15.0,0.0,0.0 } };

GLfloat rm[4][3] = { { 0.0,0.0,0.0 },{ 0.0,0.05,0.0 },{ 0.6,0.05,0.0 },{ 0.6,0.0,0.0 } };

GLfloat rd[4][3] = { { 0.0,0.0,0.0 },{ 0.0,0.250,0.0 },{ 2.0,0.250,0.0 },{ 2.0,0.0,0.0 } };

GLfloat colors[][3] = { { 1.0,0.0,0.0 },{ 0.0,1.0,0.0 },{ 1.0,0.0,1.0 },{ 1.0,0.78,0.530 },{ 0.0,0.750,0.5 },{ 0.50,0.0,1.0 },{ 0.3,0.0,0.0 },{ 0.2,0.9,1.9 },{ 1.0,0.444,1.0 } };

GLfloat bw[][3] = { { 0.0,0.0,0.0 },{ 0.55,0.50,0.0 } };

GLfloat f[4][3] = { { -1.0,-13.0,0.0 },{ -1.0,-11.5,0.0 },{ 1.0,-11.5,0.0 },{ 1.0,-13.0,0.0 } };

GLfloat f2[4][3] = { { -1.0,-13.0,0.0 },{ -1.0,-11.5,0.0 },{ 1.0,-11.5,0.0 },{ 1.0,-13.0,0.0 } };

GLfloat mirclr[][3] = { { 0,0,1.0 },{ 1,1,1 } };

void move\_key(int key, int x, int y)

{

switch (key)

{

case GLUT\_KEY\_LEFT:flag = 1;

break;

case GLUT\_KEY\_RIGHT:flag = 2;

break;

case GLUT\_KEY\_UP:flag = 3;

break;

case GLUT\_KEY\_DOWN:flag = 4;

break;

}

}

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

{

if (key==32)flagw++;

else if (key==113||key==81) flag = 10;

else if (key==76 || key==108) flag = 9;

}

void output1(int x, int y, char \*string)

{

int len, i;

glRasterPos2f(x, y);

len = (int)strlen(string);

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

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, string[i]);

}

}

void output(int x, int y, char \*string)

{

int len, i;

glRasterPos2f(x, y);

len = (int)strlen(string);

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

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, string[i]);

}

}

void sqr(GLfloat \*a, GLfloat \*b, GLfloat \*c, GLfloat \*d)

{

glBegin(GL\_QUADS);

glVertex3fv(a);

glVertex3fv(b);

glVertex3fv(c);

glVertex3fv(d);

glEnd();

}

void carmirror()

{

glColor3fv(mirclr[0]);

//glBegin(GL\_QUADS);

sqr(mrr[0], mrr[1], mrr[2], mrr[3]);

/\*glVertex3fv(mrr[0]);

glVertex3fv(mrr[1]);

glVertex3fv(mrr[2]);

glVertex3fv(mrr[3]);

glEnd();\*/

}

void carlight()

{

glBegin(GL\_QUADS);

glVertex3fv(light[0]);

glVertex3fv(light[1]);

glVertex3fv(light[2]);

glVertex3fv(light[3]);

glEnd();

}

void wheel()

{

glBegin(GL\_QUADS);

glVertex3fv(w[0]);

glVertex3fv(w[1]);

glVertex3fv(w[2]);

glVertex3fv(w[3]);

glEnd();

}

GLfloat theta = 0, delta = 0, a[20][2], kx = 0.0, ky = -9.0;

void circle(GLfloat r, GLfloat kx, GLfloat ky)

{

int i, n = 20;

double MPI = 3.1415926;

delta = (360/n)\*(MPI/180);

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

{

a[i][0] = r\*cos(theta)+kx;

a[i][1] = r\*sin(theta)+ky;

theta += delta;

}

glBegin(GL\_POLYGON);

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

glVertex2fv(a[i]);

glEnd();

}

void sheep()

{

glColor3f(1.0, 1.0, 0.90);

circle(0.6, kx, ky);

circle(0.25, kx, ky+0.7);

glColor3f(0, 0, 0);

circle(0.1, kx, ky+0.9);

glColor3f(1.0, 1.0, 0.90);

circle(0.15, kx, ky-0.58);

}

int i = 0;

void car1(){

glBegin(GL\_POLYGON);

glVertex3fv(cv[0]);

glVertex3fv(cv[1]);

glVertex3fv(cv[2]);

glVertex3fv(cv[3]);

glVertex3fv(cv[4]);

glVertex3fv(cv[5]);

glVertex3fv(cv[6]);

glVertex3fv(cv[7]);

glEnd();

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_QUADS);

glVertex3fv(cs[0]);

glVertex3fv(cs[1]);

glVertex3fv(cs[2]);

glVertex3fv(cs[3]);

glEnd();

glColor3f(1, 1, 0);

glTranslatef(1.065, 0.3, 0);

carlight();

glTranslatef(-1.065, -0.3, 0);

glTranslatef(1.065, -0.45, 0);

carlight();

glTranslatef(-1.065, 0.45, 0);

carmirror();

glTranslatef(-1, -0, 0);

carmirror();

glTranslatef(1, 0, 0);

glColor3f(0.0, 0.0, 0);

glTranslatef(0.35, 0.75, 0);

wheel();

glTranslatef(-0.35, -0.75, 0);

glTranslatef(0.35, -0.82, 0);

wheel();

glTranslatef(-0.35, 0.82, 0);

glTranslatef(-0.8, 0.75, 0);

wheel();

glTranslatef(0.8, -0.75, 0);

glTranslatef(-0.8, -0.82, 0);

wheel();

glTranslatef(0.8, 0.82, 0);

}

void drawroad(GLfloat x, GLfloat y)

{

int c = 0;

int n = 7;

while (n)

{

while (x<10.0)

{

glTranslatef(x, y, 0.0);

glColor3fv(bw[c%2]);

sqr(rd[0], rd[1], rd[2], rd[3]);

glTranslatef(-x, -y, -0.0);

x = x+2.0;

c++;

}

glColor3f(0.5, 0.5, 0.5);

glTranslatef(0.0, (y+0.25), 0.0);

sqr(rv[0], rv[1], rv[2], rv[3]);

glTranslatef(0.0, -(y+0.25), 0.0);

x = -10.0;

while (x<10.0)

{

glTranslatef(x, (y+1.25), 0.0);

glColor3f(1.0, 1.0, 1.0);

sqr(rm[0], rm[1], rm[2], rm[3]);

glTranslatef(-x, -(y+1.25), 0.0);

x = x+1.250;

}

n--;

y = y+2.250;

x = -10.0;

}

while (x<10.0)

{

glTranslatef(x, y, 0.0);

glColor3fv(bw[c%2]);

sqr(rd[0], rd[1], rd[2], rd[3]);

glTranslatef(-x, -y, -0.0);

x = x+2.0;

c++;

}

}

void delay(int n1)

{

output(-2, -2, "GAME OVER");

for (int i = 0; i<9999\*n1; i++)

for (int j = 0; j<999\*n1; j++)

{

;

}

}

void life()

{

//if(count==0)

//glClearColor(0.0,0.0,0.0,0.0);

//output(-2,-2,"GAME OVER");

flag = 100;

delay(7);

//exit(0);

}

static float x = -10.0, y = 0.0;

static float x1 = -10.0, y11 = 4.50;

static float x2 = -10.0, y2 = -4.50;

static float x3 = 10.0, y3 = 2.25;

static float x4 = 10.0, y4 = -6.75;

static float x5 = 10.0, y5 = -2.25;

static float x6 = 10.0, y6 = 6.75;

static float x7 = -16.0, y7 = -4.50;

static float x8 = 16.0, y8 = -6.75;

static float x9 = 22.0, y9 = -6.75;

//static float x8=10.0,y8=2.25;

void vehicle()

{

glColor3fv(colors[1]);

glTranslatef(x, y, 0.0);

car1();

glTranslatef(-x, -y, 0.0);

x = x+0.0210+l;

if (x>=13.0)

{

x = -10.0;

}

glColor3fv(colors[2]);

glTranslatef(x1, y11, 0.0);

car1();

glTranslatef(-x1, -y11, 0.0);

x1 = x1+0.0425+l;

if (x1>=13.0)

{

x1 = -13.0;

}

glColor3fv(colors[3]);

glTranslatef(x2, y2, 0.0);

car1();

glTranslatef(-x2, -y2, 0.0);

x2 = x2+0.0220+l;

if (x2>=16.0)

{

x2 = -10.0;

}

if (level==1)

{

glColor3fv(colors[0]);

glTranslatef(x7, y7, 0.0);

car1();

glTranslatef(-x7, -y7, 0.0);

x7 = x7+0.0220+l;

if (x7>=10.0)

{

x7 = -16.0;

}

}

glColor3fv(colors[4]);

glTranslatef(x3, y3, 0.0);

glRotatef(180, 0, 0, 1);

car1();

glRotatef(180, 0, 0, 1);

glTranslatef(-x3, -y3, 0.0);

x3 = x3-0.0250-l;

if (x3<=-13.0)

{

x3 = 13.0;

}

glColor3fv(colors[5]);

glTranslatef(x4, y4, 0.0);

glRotatef(180, 0, 0, 1);

car1();

glRotatef(180, 0, 0, 1);

glTranslatef(-x4, -y4, 0.0);

x4 = x4-0.01250-l;

if (x4<=-24.0)

{

x4 = 10.0;

}

glColor3fv(colors[6]);

glTranslatef(x8, y8, 0.0);

glRotatef(180, 0, 0, 1);

car1();

glRotatef(180, 0, 0, 1);

glTranslatef(-x8, -y8, 0.0);

x8 = x8-0.01250-l;

if (x8<=-18.0)

{

x8 = 16.0;

}

glColor3fv(colors[7]);

glTranslatef(x9, y9, 0.0);

glRotatef(180, 0, 0, 1);

car1();

glRotatef(180, 0, 0, 1);

glTranslatef(-x9, -y9, 0.0);

x9 = x9-0.01250-l;

if (x9<=-12.0)

{

x9 = 22.0;

}

glColor3fv(colors[8]);

glTranslatef(x5, y5, 0.0);

glRotatef(180, 0, 0, 1);

car1();

glRotatef(180, 0, 0, 1);

glTranslatef(-x5, -y5, 0.0);

x5 = x5-0.01450-l;

if (x5<=-13.0)

{

x5 = 13.0;

}

glColor3fv(colors[0]);

glTranslatef(x6, y6, 0.0);

glRotatef(180, 0, 0, 1);

car1();

glRotatef(180, 0, 0, 1);

glTranslatef(-x6, -y6, 0.0);

x6 = x6-0.0210-l;

if (x6<=-13.0)

{

x6 = 13.0;

}

}

void movmt()

{

if (flag==3)

{

if (ky<9)

{

ky = ky+2.25;

sheep();

flag = 0;

}

}

if (flag==1)

{

if (kx>-9)

{

kx = kx-1.0;

sheep();

flag = 0;

}

}

if (flag==2)

{

if (kx<9)

{

kx = kx+1.0;

sheep();

flag = 0;

}

}

if (flag==4)

{

if (ky>-9)

{

ky = ky-2.25;

sheep();

flag = 0;

}

}

}

void check()

{

if (ky==y) {

if ((x-1<=kx-0.6 && kx-0.6<=x+1) || (x-1<=kx+0.6 && kx+0.6<=x+1)) {// && kx+0.6>=x-1) {

life();

}

}

if (ky==y11) {

if ((x1-1<=kx-0.6 && kx-0.6<=x1+1) || (x1-1<=kx+0.6 && kx+0.6<=x1+1)) {

life();

}

}

if (ky==y2) {

if ((x2-1<=kx-0.6 && kx-0.6<=x2+1) || (x2-1<=kx+0.6 && kx+0.6<=x2+1)) {

life();

}

}

if (ky==y7) {

if ((x7-1<=kx-0.6 && kx-0.6<=x7+1) || (x7-1<=kx+0.6 && kx+0.6<=x7+1)) {

life();

}

}

if (ky==y3) {

if ((x3-1<=kx-0.6 && kx-0.6<=x3+1) || (x3-1<=kx+0.6 && kx+0.6<=x3+1)) {

life();

}

}

if (ky==y4) {

if ((x4-1<=kx-0.6 && kx-0.6<=x4+1) || (x4-1<=kx+0.6 && kx+0.6<=x4+1)) {

life();

}

}

if (ky==y5) {

if ((x5-1<=kx-0.6 && kx-0.6<=x5+1) || (x5-1<=kx+0.6 && kx+0.6<=x5+1)) {

life();

}

}

if (ky==y6) {

if ((x6-1<=kx-0.6 && kx-0.6<=x6+1) || (x6-1<=kx+0.6 && kx+0.6<=x6+1)) {

life();

}

}

if (ky==y8) {

if ((x8-1<=kx-0.6 && kx-0.6<=x8+1) || (x8-1<=kx+0.6 && kx+0.6<=x8+1)) {

life();

}

}

if (ky==y9) {

if ((x9-1<=kx-0.6 && kx-0.6<=x9+1) || (x9-1<=kx+0.6 && kx+0.6<=x9+1)) {

life();

}

}

}

void display()

{

int i, j;

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

glLoadIdentity();

glColor3f(1.0, 1.0, 1.0);

if (flag==100)

{

output(-2, 1, "--------------------");

output1(-2, 0, "GAME OVER");

output(-4, -1, "---------------------");

}

else if (flagw==1)

{

output1(-2, 4, "SHEEPISH");

output1(-1, -5, " A Game by :");

output(2, -6, "NOUREEN");

output(2, -7, "20231CSE0292");

output1(-3, -0, "Use SPACE BAR to Continue.........");

}

else if (flagw==2)

{

output1(-2, 4, "...INSTRUCTION...");

output(-3, -0, "Use ARROW KEYS to move the object");

output(-3, -1, "to reach home");

}

else

{

drawroad(-10.0, -8.0);

glColor3f(0.7, 0.4, 1.0);

check();

vehicle();

sheep();

if (ky>=9.0)

{

glColor3f(1.0, 1.0, 1.0);

output1(-2, 0, "LEVEL COMPLETE");

output(-2, -2, "press 'L' or 'l' to NEXT level");

output(-2, -4, "press 'Q' or 'q' to quit");

if (flag==10) exit(0);

else if (flag==9)

{

level = level+1;

ky = -9.0;

kx = 0;

sheep();

if (level==1) l = 0.00005;

else if (level==2) l = 0.0010;

else if (level==3) l = 0.0015;

else if (level==4) l = 0.0020;

else if (level==5) l = 0.0025;

else if (level==6) l = 0.0030;

else if (level==7) l = 0.0035;

else if (level==8) l = 0.0040;

}

}

if (flag==1 || flag==2 || flag==3 || flag==4)

{

movmt();

}

}

glFlush();

glutPostRedisplay();

glutSwapBuffers();

}

void myReshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

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

//if(w<=h)

//glOrtho(-10.0,10.0,-10.0 \* (GLfloat) h/ (GLfloat) w,10.0 \* (GLfloat) h/ (GLfloat) w,-10.0,10.0);

//else

//glOrtho(-10.0 \* (GLfloat) w/ (GLfloat) h,10.0 \* (GLfloat) w/ (GLfloat) h,-10.0,10.0,-10.0,10.0);

//gluLookAt(0.0,0.0,1.25,0.0,0.0,0.0,5.0,5.0,5.0);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc, argv);

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

glutInitWindowSize(800, 800);

glutInitWindowPosition(100, 0);

glutCreateWindow("SHEEPISH");

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glutSpecialFunc(move\_key);

//glutIdleFunc(hn);

//glClearColor(1.0,1.0,1.0,1.0);

glClearColor(0.0, 0.5, 0.0, 0.0);

glutMainLoop();

return 0;

}