1. Design outline

We would like to implement a three dimensional maze game through OpenGL in this project.

In the skeleton, we have a function which can read the map, when reading the 3 is the door while reading the 1 is the wall. When the wall of the coordinates for the 1, it is necessary to consider the coordinate around him, because a wall has 4 sides. When the coordinate equals 0, there is no wall. The most important is the collision detection, when the distance from the wall to the center is less than radius, it will collision. We can imagine the 3D map as a 2D map instead, and the core part for constructing it is to build an array.

If the coordinates equals 2, which means the player’s initial position. If the coordinates is 1, it is the wall, need to change direction to go. When the coordinates is 3, it is the door, this is victory, and return to the initial state. In the project, we have to calculate the player’s position，through calculating the position of the player，to determine whether has been completed, and detect the collision.

1. Key code fragments or algorithms of your program
2. Generate textures and load textures

glEnable(GL\_TEXTURE\_2D);

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 4);

glGenTextures(1, &FLOOR);

glBindTexture(GL\_TEXTURE\_2D, FLOOR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

floorimage = TextureLoadBitmap("FLOOR.BMP", &floorWidth, &floorHeight);

gluBuild2DMipmaps(GL\_TEXTURE\_2D, 3, floorWidth, floorHeight,

GL\_RGB, GL\_UNSIGNED\_BYTE, floorimage);

glGenTextures(1, &WALL);

glBindTexture(GL\_TEXTURE\_2D, WALL);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

wallimage = TextureLoadBitmap("WALL.BMP", &wallWidth, &wallHeight);

gluBuild2DMipmaps(GL\_TEXTURE\_2D, 3, wallWidth, wallHeight,

GL\_RGB, GL\_UNSIGNED\_BYTE, wallimage);

glGenTextures(1, &DOOR);

glBindTexture(GL\_TEXTURE\_2D, DOOR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

doorimage = TextureLoadBitmap("DOOR.BMP", &doorWidth, &doorHeight);

gluBuild2DMipmaps(GL\_TEXTURE\_2D, 3, doorWidth, doorHeight,

GL\_RGB, GL\_UNSIGNED\_BYTE, doorimage);

1. Draw the floor, wall and door

//====== Drawing functions ===============

void DrawGround()

{

// Draw the ground here

// 拆分成一块一块地板

int i, j;

glPushMatrix();

for (j = 0; j < \_mapz; j++) {

for (i = 0; i < \_mapx; i++) {

glPushMatrix();

glTranslatef((2 \* i + 1) \* \_wallScale / 2.0, 0.0, (2 \* j + 1) \* \_wallScale / 2.0);

glScalef(\_wallScale, 1.0, \_wallScale);

//if (\_map[i][j] == 0)

{

glBindTexture(GL\_TEXTURE\_2D, FLOOR);

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 1.0); glVertex3f(-0.5, 0.0, 0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(0.5, 0.0, 0.5);

glTexCoord2f(1.0, 0.0); glVertex3f(0.5, 0.0, -0.5);

glTexCoord2f(0.0, 0.0); glVertex3f(-0.5, 0.0, -0.5);

glEnd();

}

glPopMatrix();

}

}

glPopMatrix();

}

void DrawWalls()

{

// Draw the maze's walls here

int i, j;

glPushMatrix();

//glColor3f(0.0, 1.0, 1.0);

for (j = 0; j < \_mapz; j++) {

for (i = 0; i < \_mapx; i++) {

glPushMatrix();

glTranslatef((2\*i+1) \* \_wallScale / 2.0, 0.0 , (2\*j+1) \* \_wallScale / 2.0);

glScalef(\_wallScale, 1.0, \_wallScale);

if (\_map[i][j] == 1)

{

glBindTexture(GL\_TEXTURE\_2D, WALL);

glBegin(GL\_QUADS);

glTexCoord2f(1.0, 0.0); glVertex3f(-0.5, 0.0, 0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(-0.5, 1.0, 0.5);

glTexCoord2f(0.0, 1.0); glVertex3f(-0.5, 1.0, -0.5);

glTexCoord2f(0.0, 0.0); glVertex3f(-0.5, 0.0, -0.5);

glEnd();

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(-0.5, 0.0, 0.5);

glTexCoord2f(1.0, 0.0); glVertex3f(0.5, 0.0, 0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(0.5, 1.0, 0.5);

glTexCoord2f(0.0, 1.0); glVertex3f(-0.5, 1.0, 0.5);

glEnd();

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(0.5, 0.0, 0.5);

glTexCoord2f(1.0, 0.0); glVertex3f(0.5, 0.0, -0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(0.5, 1.0, -0.5);

glTexCoord2f(0.0, 1.0); glVertex3f(0.5, 1.0, 0.5);

glEnd();

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(0.5, 0.0, -0.5);

glTexCoord2f(1.0, 0.0); glVertex3f(-0.5, 0.0, -0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(-0.5, 1.0, -0.5);

glTexCoord2f(0.0, 1.0); glVertex3f(0.5, 1.0, -0.5);

glEnd();

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 1.0); glVertex3f(-0.5, 1.0, 0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(0.5, 1.0, 0.5);

glTexCoord2f(1.0, 0.0); glVertex3f(0.5, 1.0, -0.5);

glTexCoord2f(0.0, 0.0); glVertex3f(-0.5, 1.0, -0.5);

glEnd();

}

glPopMatrix();

}

}

glPopMatrix();

}

void DrawDoor()

{

int i, j;

glPushMatrix();

//glColor3f(1.0, 0.0, 0.0);

for (j = 0; j < \_mapz; j++) {

for (i = 0; i < \_mapx; i++) {

glPushMatrix();

glTranslatef((2 \* i + 1) \* \_wallScale / 2.0, 0.0, (2 \* j + 1) \* \_wallScale / 2.0);

glScalef(\_wallScale, 1.0, \_wallScale);

if (\_map[i][j] == 3)

{

glBindTexture(GL\_TEXTURE\_2D, DOOR);

glBegin(GL\_QUADS);

glTexCoord2f(1.0, 0.0); glVertex3f(-0.5, 0.0, 0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(-0.5, 1.0, 0.5);

glTexCoord2f(0.0, 1.0); glVertex3f(-0.5, 1.0, -0.5);

glTexCoord2f(0.0, 0.0); glVertex3f(-0.5, 0.0, -0.5);

glEnd();

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(-0.5, 0.0, 0.5);

glTexCoord2f(1.0, 0.0); glVertex3f(0.5, 0.0, 0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(0.5, 1.0, 0.5);

glTexCoord2f(0.0, 1.0); glVertex3f(-0.5, 1.0, 0.5);

glEnd();

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(0.5, 0.0, 0.5);

glTexCoord2f(1.0, 0.0); glVertex3f(0.5, 0.0, -0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(0.5, 1.0, -0.5);

glTexCoord2f(0.0, 1.0); glVertex3f(0.5, 1.0, 0.5);

glEnd();

glBegin(GL\_QUADS);

glTexCoord2f(0.0, 0.0); glVertex3f(0.5, 0.0, -0.5);

glTexCoord2f(1.0, 0.0); glVertex3f(-0.5, 0.0, -0.5);

glTexCoord2f(1.0, 1.0); glVertex3f(-0.5, 1.0, -0.5);

glTexCoord2f(0.0, 1.0); glVertex3f(0.5, 1.0, -0.5);

glEnd();

}

glPopMatrix();

}

}

glPopMatrix();

}

1. draw the player

void DrawPlayer()

{

// Draw your player here

glPushMatrix();

glTranslatef(\_player.pos[0], \_player.pos[1], \_player.pos[2]);

glColor3f(1.0, 1.0, 1.0);

glutSolidSphere(\_player.mySize, 15, 15);

glPushMatrix();

glTranslatef(-0.15, 0.15, 0.0);

glutSolidSphere(0.05, 15, 15);

glPopMatrix();

glPushMatrix();

glTranslatef(0.15, 0.15, 0.0);

glutSolidSphere(0.05, 15, 15);

glPopMatrix();

glPushMatrix();

glTranslatef(0.0, -0.1, 0.0);

glRotatef(180, 1, 0, 0);

glutSolidCone(0.03, 0.5, 15, 15);

glPopMatrix();

glPopMatrix();

}

1. you win

In the collision dectection, we can get the isWin true when the player arrived at the exit.

if (isWin == TRUE)

{//reset

Sleep(2000);

initplayer();

isWin = FALSE;

}

1. Collision detection

void checkcollide()

{

float dx, dz;

int i = (int)(\_player.pos[0] / \_wallScale); //强制转换，获得当前区块x坐标

int j = (int)(\_player.pos[2] / \_wallScale); //获得当前区块z坐标

// Check collision of walls here

// Update the current position

dx = \_player.forward \* sin(\_player.degree \* M\_PI / 180.0);

dz = \_player.forward \* cos(\_player.degree \* M\_PI / 180.0);

float current\_x = \_player.pos[0] + dx;

float current\_z = \_player.pos[2] + dz;

float distance;

// move forward in x direction

if (dx > 0)

{

if (i + 1 >= \_mapx ||

\_map[i + 1][j] == 1)

{

distance = \_player.mySize - ((i + 1) \* \_wallScale - current\_x); //玩家的半径与玩家中心距墙距离的比较

if (distance < 0)

{

\_player.pos[0] += dx;

}

}

else

{

\_player.pos[0] += dx;

}

}

else // move backward in x direction

{

if (i <= 0 ||

\_map[i - 1][j] == 1)

{

distance = \_player.mySize - (current\_x - i \* \_wallScale); //玩家的半径与玩家中心距墙距离的比较

if (distance < 0)

{

\_player.pos[0] += dx;

}

}

else

{

\_player.pos[0] += dx;

}

}

// move forward in z direction

if (dz>0)

{

if (j + 1 >= \_mapz ||

\_map[i][j + 1] == 1)

{

distance = \_player.mySize - ((j + 1) \* \_wallScale - current\_z); //玩家的半径与玩家中心距墙距离的比较

if (distance < 0)

{

\_player.pos[2] += dz;

}

}

else

{

\_player.pos[2] += dz;

}

}

// move backward in z direction

else

{

if (j <= 0 ||

\_map[i][j - 1] == 1)

{

distance = \_player.mySize - (current\_z - j \* \_wallScale); //玩家的半径与玩家中心距墙距离的比较

if (distance < 0)

{

\_player.pos[2] += dz;

}

}

else

{

\_player.pos[2] += dz;

}

}

if (\_map[i][j] == 3)

{

isWin = true;

}

//\_player.pos[0] += dx;

//\_player.pos[2] += dz;

}

1. lighting and material

// material

GLfloat mat\_ambient[] = { 0.7, 0.7, 0.7, 1.0 };

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat mat\_shininess[] = { 50.0 };

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

// lighting

GLfloat positionLight[] = { 0.0f, 10.0f, 1.0f, 1.0f };

GLfloat ambientLight[] = { 0.50f, 0.50f, 0.50f, 1.0f };

GLfloat diffuseLight[] = { 0.45f, 0.15f, 0.75f, 1.0f };

GLfloat specularLight[] = { 0.40f, 0.40f, 0.40f, 1.0f };

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

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

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

glLightfv(GL\_LIGHT0, GL\_POSITION, positionLight);

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

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

1. How to use your program

Left click the mouse, the left rotation. right click the mouse, turn to the right. Left and right click the mouse together, move forward.

We have 3 basic functions here:

First, we can just click the mouse for control the player to turn and move. We can look at the function here:

// Please read this function carefully, you can learn how to

// make use the mouse buttons to control the Test Object/Player

void mouse(int button, int state, int x, int y)

{

static int buttonhold = 0;

if ((button == GLUT\_LEFT\_BUTTON) && (state == GLUT\_UP)) {

if (buttonhold >= 2) {

// Stop forward and turn right

\_player.forward = 0.0;

\_player.spin = -\_player.spinStepSize; // Turn right

}

else

\_player.spin = 0.0; // Stop turn left

buttonhold--;

}

if ((button == GLUT\_RIGHT\_BUTTON) && (state == GLUT\_UP)) {

if (buttonhold >= 2) {

// Stop forward and turn left

\_player.forward = 0.0;

\_player.spin = \_player.spinStepSize; // Turn left

}

else

\_player.spin = 0.0; // Stop turn right

buttonhold--;

}

if ((button == GLUT\_MIDDLE\_BUTTON) && (state == GLUT\_UP)) {

\_player.forward = 0.0;

}

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

if (buttonhold > 0) {

\_player.forward = \_player.forwardStepSize;

\_player.spin = 0.0;

}

else

\_player.spin = \_player.spinStepSize; // Turn left

buttonhold++;

}

if ((button == GLUT\_RIGHT\_BUTTON) && (state == GLUT\_DOWN)) {

if (buttonhold > 0) {

\_player.forward = \_player.forwardStepSize;

\_player.spin = 0.0;

}

else

\_player.spin = -\_player.spinStepSize; // Turn right

buttonhold++;

}

if ((button == GLUT\_MIDDLE\_BUTTON) && (state == GLUT\_DOWN)) {

\_player.forward = \_player.forwardStepSize;

}

}

It means when we click on button separately, we will get the direction changed. But if we click two buttons together, we will move the player forward.

Second, we can click ‘a’ or ‘A’ to make the camera watching all the maze, we can switch the regular mode to this cheating mode.

if (camera == TRUE)

{

gluLookAt(\_player.pos[0] - 2.0 \* sin(\_player.degree \* M\_PI / 180.0), // eye

\_player.pos[1] + 0.25,

\_player.pos[2] - 2.0 \* cos(\_player.degree\* M\_PI / 180.0),

\_player.pos[0], // at

\_player.pos[1],

\_player.pos[2],

0.0, 1.0, 0.0); // up

}

//俯拍视角

else if (camera == FALSE)

{

gluLookAt(8.0, // eye

20.0,

8.0,

8.0, // at

0.0,

8.0,

0.0, 0.0, -1.0); //up

}

Third, we can click ‘f’ or ‘F’ to open the fog effect. The original status is disabled.

// Fog

GLfloat fog[4] = { 0.3f, 0.3f, 0.6f, 1.0f };

glDisable(GL\_FOG);

glFogi(GL\_FOG\_MODE, GL\_EXP);

glFogfv(GL\_FOG\_COLOR, fog);

glFogf(GL\_FOG\_DENSITY, 0.15);

glHint(GL\_FOG\_HINT, GL\_DONT\_CARE);

glFogf(GL\_FOG\_START, 1.0);

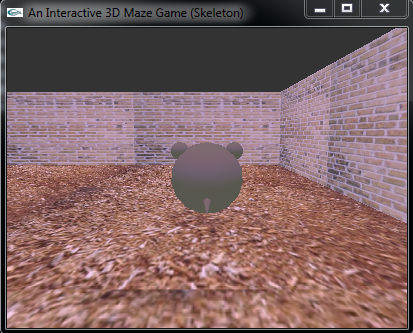
glFogf(GL\_FOG\_END, 4.0);

glEnable(GL\_POLYGON\_SMOOTH);

glHint(GL\_POLYGON\_SMOOTH\_HINT, GL\_NICEST);

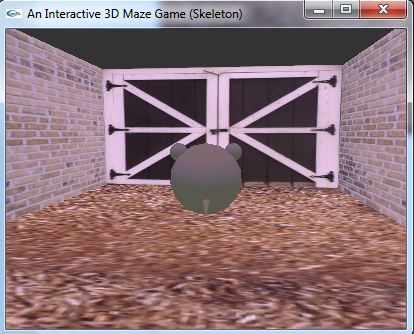
4. Experimental results

In the game：

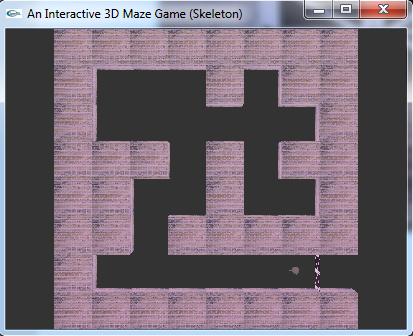




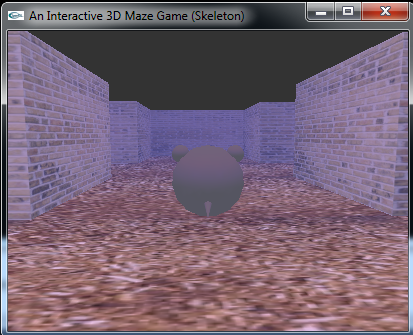
To win the game:



Mode A:



Mode F:



1. Your feelings or opinions about this project.

Through this study to complete the project, I know that how to use OpenGL to complete the basic operations or tasks including translate, rotating, enabling texture, material and lighting. Meanwhile, I need to understand some algorithms about constructing the map and checking the collision.