Report

1409853G-I011-0074
李煜麟

Design outline:

First, I design my ground and wall. Because it's easy. Then I design the door. And map the texture on them. And finally set the role of detection collision. And a maze contains 4 kinds of wall, so we need define 4 kinds of wall. And add some fog effect in the init().And when I design the player, I try to add a light on the player.

Key code fragments or algorithms of my program:

```
typedef struct _playerInfo {
    GLfloat degree;  // Object orientation
    GLfloat rotate;  // Object orientation
    GLfloat forward, spin;
    GLfloat pos[3]; // User position
    GLfloat mySize; // User radial size
    GLfloat forwardStepSize;    // Step size
    GLfloat spinStepSize;   // Rotate step size
} playerInfo;

GLuint groundTextureID = 0;
GLuint wallTextureID = 0;
GLuint doorTextureID = 0;
GLuint tireTextureID = 0;


playerInfo _player;
```

This part is important to be used to create object for player.

```
void DrawGround()
{
    // Draw the ground here
    glPushMatrix();
    glTranslatef(_wallScale * _mapCols / 2.0, 0.0, _wallScale * _mapRows / 2.0);
    glScalef(_wallScale * _mapCols, 1.0, _wallScale * _mapRows);

    glColor3f(1.0, 1.0, 1.0);
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, groundTextureID);
    glBegin(GL_QUADS);
    glTexCoord2f(0, 0);
    glVertex3f(-0.5, 0.0, 0.5);

    glTexCoord2f(_mapCols, 0);
    glVertex3f(0.5, 0.0, 0.5);

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

    glTexCoord2f(0, _mapRows);
    glVertex3f(-0.5, 0.0, -0.5);
    glEnd();
    glDisable(GL_TEXTURE_2D);
    glPopMatrix();
}
```

This part is used to creating the ground.

```
void DrawWall(float cx, float cy, wallDir dir, GLuint textureID)
{
    glBindTexture(GL_TEXTURE_2D, textureID);
    glPushMatrix();
    glScalef(1.0, 0.5, 1.0);
    glTranslatef(cx, 0.5*_wallScale, cy);
    glRotatef(dir*90.0, 0, 1, 0);

    glBegin(GL_QUADS);
    glTexCoord2f(0, 0);
    glVertex3f(-_wallScale*0.5, -_wallScale*0.5, -_wallScale*0.5);

    glTexCoord2f(0, 1);
    glVertex3f(-_wallScale*0.5, _wallScale*0.5, -_wallScale*0.5);

    glTexCoord2f(1, 1);
    glVertex3f(_wallScale*0.5, _wallScale*0.5, -_wallScale*0.5);

    glTexCoord2f(1, 0);
    glVertex3f(_wallScale*0.5, -_wallScale*0.5, -_wallScale*0.5);

    glEnd();

    glPopMatrix();
}
```

This part is used to draw the Wall. And cx, cy means the center point of the wall. Dir means the direction of the wall. textureID is the material of the wall.

```
enum wallDir
{
    NORTH,
    WEST,
    SOUTH,
    EAST
};
```

This part is used to define the direction of the wall.

```
void DrawWalls()
{
    // Draw the maze's walls here
    glEnable(GL_TEXTURE_2D);

    for (int i = 0; i < _mapRows; ++i)
    {
        for (int j = 0; j < _mapCols; ++j)
        {
            if (_map[i][j] % 2 != 0)    //wall or door
            {
                GLuint texture = _map[i][j] == 1 ? wallTextureID : doorTextureID;
                if (i - 1 >= 0 && !(_map[i - 1][j] % 2))        //north
                {
                    DrawWall((j + 0.5)*_wallScale, (i + 0.5)*_wallScale, NORTH, texture);
                }
                if (j + 1 < _mapCols && !(_map[i][j + 1] % 2))      //east
                {
                    DrawWall((j + 0.5)*_wallScale, (i + 0.5)*_wallScale, EAST, texture);
                }
                if (i + 1 < _mapRows&& !(_map[i + 1][j] % 2))       //south
                {
                    DrawWall((j + 0.5)*_wallScale, (i + 0.5)*_wallScale, SOUTH, texture);
                }
                if (j - 1 >= 0 && !(_map[i][j - 1] % 2))        //west
                {
                    DrawWall((j + 0.5)*_wallScale, (i + 0.5)*_wallScale, WEST, texture);
                }
            }
        }
    }
```

This part is used to creating 4 kinds of wall.

```
void DrawPlayer()
{
    // Draw your player here
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, tireTextureID);
    glPushMatrix();

    glTranslatef(_player.pos[0], _player.pos[1], _player.pos[2]);
    glScalef(_player.pos[1], _player.pos[1], _player.pos[1]);
    glRotatef(_player.degree, 0, 1, 0);
    glRotatef(_player.rotate, 1, 0, 0);

    glBegin(GL_TRIANGLE_STRIP);

    for (int i = 0; i < 61; ++i)
    {
        glTexCoord2f(0, i*0.05);
        glVertex3f(-0.5, 0.9*sin(i * 6 * M_PI / 180.0), 0.9*cos(i * 6 * M_PI / 180.0));
        glTexCoord2f(0.1, i*0.05);
        glVertex3f(-0.4, sin(i * 6 * M_PI / 180.0), cos(i * 6 * M_PI / 180.0));
    }

    for (int i = 0; i < 61; ++i)
    {
        glTexCoord2f(0.1, i*0.05);
        glVertex3f(-0.4, sin(i * 6 * M_PI / 180.0), cos(i * 6 * M_PI / 180.0));
        glTexCoord2f(0.9, i*0.05);
        glVertex3f(0.4, sin(i * 6 * M_PI / 180.0), cos(i * 6 * M_PI / 180.0));
    }

    for (int i = 0; i < 61; ++i)
    {
        glTexCoord2f(0.9, i*0.05);
        glVertex3f(0.4, sin(i * 6 * M_PI / 180.0), cos(i * 6 * M_PI / 180.0));
        glTexCoord2f(1.0, i*0.05);
        glVertex3f(0.5, 0.9*sin(i * 6 * M_PI / 180.0), 0.9*cos(i * 6 * M_PI / 180.0));
    }

    glEnd();

    glDisable(GL_TEXTURE_2D);
    glPopMatrix();
}
```

This part is used to draw the player, it’s a tire.

```
void checkcollide()
{
    static auto timeStamp = clock() / 1000.0f;
    if (_player.forward == 0.0)
    {
        timeStamp = clock() / 1000.0f;
        return;
    }
    float dx, dz;
    // Check collision of walls here

    // Update the current position
    auto nowTime = clock() / 1000.0f;

    dx = _player.forward *(nowTime - timeStamp)* sin(_player.degree * M_PI / 180.0);
    dz = _player.forward *(nowTime - timeStamp)* cos(_player.degree * M_PI / 180.0);

    _player.rotate += (360/(2*M_PI*_player.pos[1]/_player.forward)) * (nowTime - timeStamp);

    _player.pos[0] += dx;
    _player.pos[2] += dz;

    int i = _player.pos[2] / _wallScale;
    int j = _player.pos[0] / _wallScale;

    if (_map[i][j] == 3)
    {
        MessageBox(NULL, "YOU WIN!!!", "SUCCESS", MB_OK);
        initplayer();
    };

    //check north
    if (_player.pos[2] - _player.pos[1]<i*_wallScale&&_map[i-1][j]==1)
    {
        _player.pos[2] = i*_wallScale + _player.pos[1];
    }
    //check east
    if (_player.pos[0] + _player.pos[1]>(j + 1)*_wallScale&&_map[i][j+1] == 1)
    {
        _player.pos[0] =(j + 1)*_wallScale- _player.pos[1];
    }
    //check south
    if (_player.pos[2] + _player.pos[1]>(i + 1)*_wallScale&&_map[i+1][j] == 1)
    {
        _player.pos[2] = (i + 1)*_wallScale - _player.pos[1];
    }
    //check west
    if (_player.pos[0] - _player.pos[1]<j*_wallScale&&_map[i][j - 1] == 1)
    {
        _player.pos[0]= j*_wallScale + _player.pos[1] ;
    }

    timeStamp = nowTime;
}
```

This part is used to check the collision detection.

```
void init()
{
    initplayer();
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClearDepth(1.0);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);

    float fogColor[] = { 0.0, 0.0, 0.0, 1.0 };
    glFogi(GL_FOG_MODE, GL_LINEAR);
    glFogfv(GL_FOG_COLOR, fogColor);
    glFogf(GL_FOG_DENSITY, 0.35f);
    glHint(GL_FOG_HINT, GL_DONT_CARE);
    glFogf(GL_FOG_START, 1.0f);
    glFogf(GL_FOG_END, 10.0f);
    glEnable(GL_FOG);

    glGenTextures(1, &groundTextureID);
    glGenTextures(1, &wallTextureID);
    glGenTextures(1, &doorTextureID);
    glGenTextures(1, &tireTextureID);

    int height, width;
    GLubyte *data = NULL;
    glBindTexture(GL_TEXTURE_2D, groundTextureID);
    data = TextureLoadBitmap("texture//floor.bmp", &width, &height);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

    glBindTexture(GL_TEXTURE_2D, wallTextureID);
    data = TextureLoadBitmap("texture//wall.bmp", &width, &height);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

    glBindTexture(GL_TEXTURE_2D, doorTextureID);
    data = TextureLoadBitmap("texture//door.bmp", &width, &height);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

    glBindTexture(GL_TEXTURE_2D, tireTextureID);
    data = TextureLoadBitmap("texture//tire.bmp", &width, &height);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
}
```

The init() is used to stick the bmp picture on the wall, player, floor and the door.

```
float fogColor[] = { 0.0, 0.0, 0.0, 1.0 };
glFogi(GL_FOG_MODE, GL_LINEAR);
glFogfv(GL_FOG_COLOR, fogColor);
glFogf(GL_FOG_DENSITY, 0.35f);
glHint(GL_FOG_HINT, GL_DONT_CARE);
glFogf(GL_FOG_START, 1.0f);
glFogf(GL_FOG_END, 10.0f);
glEnable(GL_FOG);
```

This part use the fog effect.

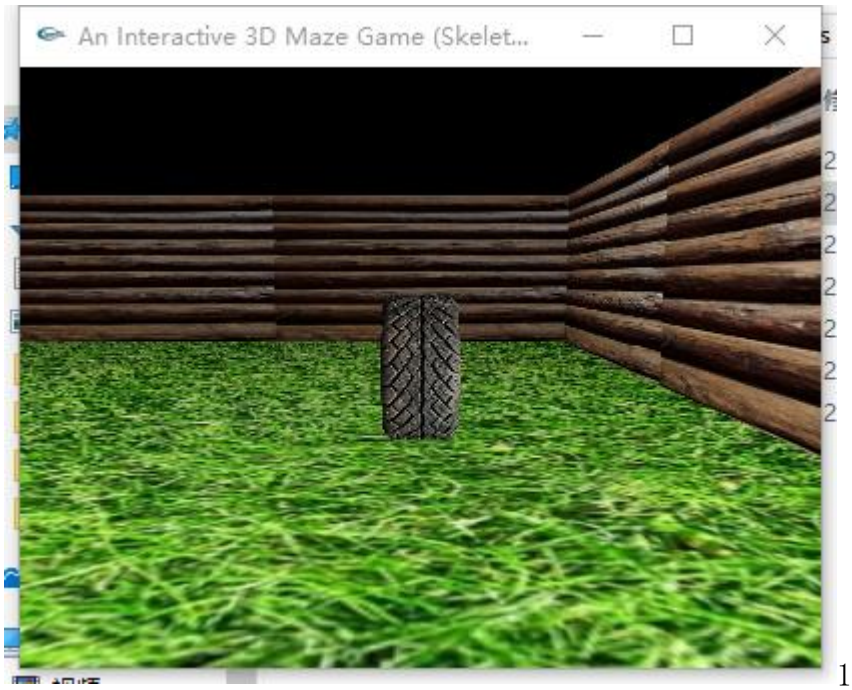
```
void DrawPlayer()
{
    // Draw your player here

    float lightPos[] = { 100, 100, 100 };
    float lightAmbient[] = { 0.2, 0.2, 0.2 };
    float lightDiffuse[] = { 0.8, 0.8, 0.8 };
    float matAmbient[] = { 1.0, 1.0, 1.0 };
    float matDiffuse[] = { 1.0, 1.0, 1.0 };

    glMaterialfv(GL_FRONT, GL_AMBIENT, matAmbient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, matDiffuse);

    glEnable(GL_LIGHT0);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);
    glEnable(GL_LIGHTING);
```

This part used the light and material to set the light on player.

How to use my program:

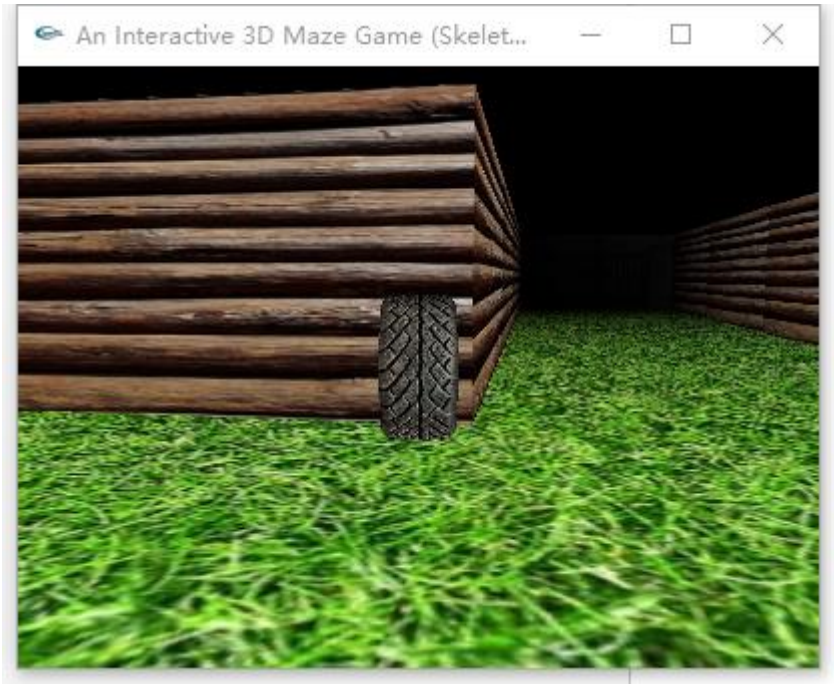
You only need to go to the terminal and run the project6.exe. Then input your map.dat in it, like this.

Experimental Results:



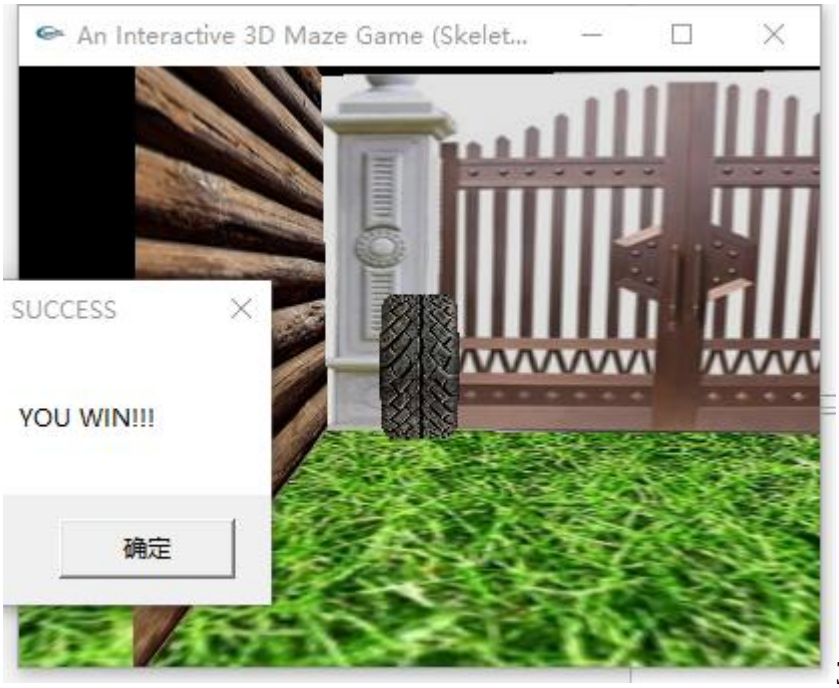
1

The result of the project. The picture1 of the beginning.



2

The picture2 of playing the maze.



3

The picture3 of winning.

4

Picture4 of final.

My feelings or opinions about this project:

It’s an interesting project and this is the first time for me to design the maze game. And when I finish it, I’m proud of myself. And the process of finding the picture and designing maze is also an unforgettable experience. And when I add some fog effect and I think the lighting is not as obvious as the fog effect.