Project for CG 100433 course

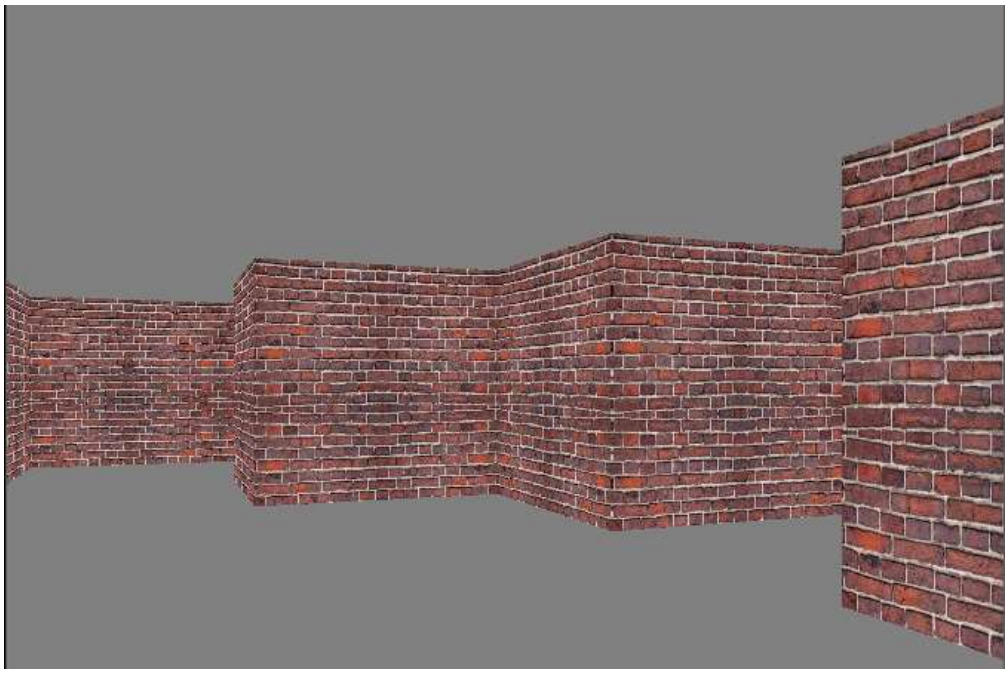
- **Team member**

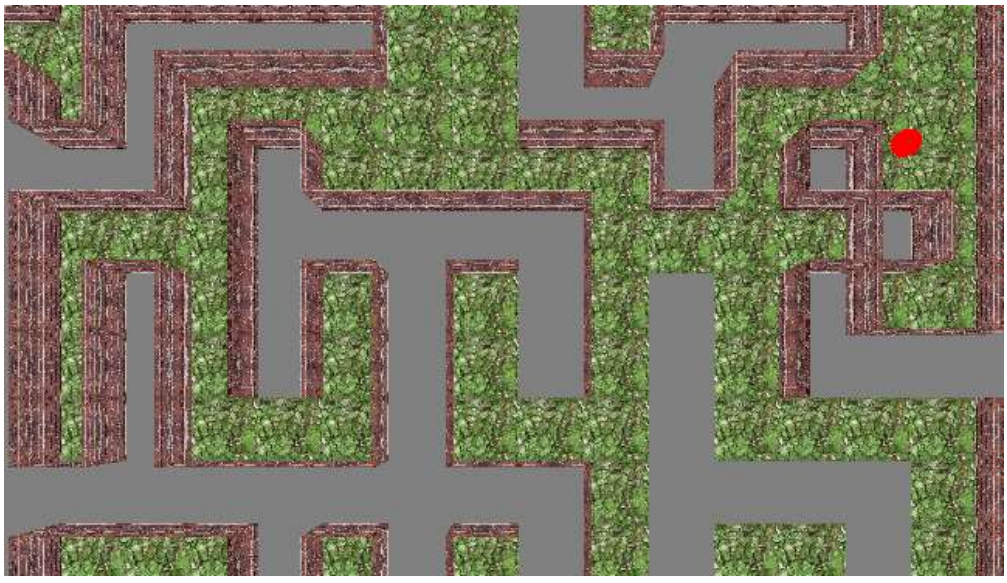
  J 组：代文海，李先龙，尹俏，蔡涵萱，周羲楠，张江华

- **Project Title**

  基于 openGL 实现的 3D 迷宫游戏

- **printscreen**





- **Code of the project**

```
#include "StdAfx.h"
#include <stdio.h>
#include <stdlib.h>
#include <glut.h>
#include <math.h>
#include <iostream>

#include <Windows.h>
#include <GLAUX.H>
#pragma comment(lib,"GLAUX.LIB")
GLuint texture[2];

using namespace std;
void drawwalls(void);
void drawtop(void);
void drawball(void);
void drawground(void);



#define IDM_APPLICATION_EXIT       (101)
#define IDM_APPLICATION_TEXTURE (102)
#define IDM_APPLICATION_BANK       (103)
#define MAZE_HEIGHT (16)
#define MAZE_WIDTH    (16)
#define STARTING_POINT_X (13.5f);
#define STARTING_POINT_Y (1.5f);
#define STARTING_HEADING (90.0f);

#define room_h 5.0f

float player_x = STARTING_POINT_X;
float player_y = STARTING_POINT_Y;
float player_h = STARTING_HEADING;   // player's heading
float player_s = 0.0f;                    // forward speed of the player
float player_m = 1.0f;   // speed multiplier of the player
float player_t = 0.0f;   // player's turning (change in heading)
float player_b = 0.0f;   // viewpoint bank (roll)
static float texcoordX = 0.0f;

int     walllist = 0;
int     mazelist = 0;
int     balllist = 0;
int   status = 1;
bool searchroute = false; //是否已经找到出路
```

```
bool keystate[4] = { false,false,false,false };//记录按键的状态信息（是否处于按下状态）

char mazedata[MAZE_HEIGHT][MAZE_WIDTH] = {
 { 'H','H','H','H','H','H','H','H','H','H','H','H','H','H','H','H' },
 { 'H',' ',' ',' ',' ',' ',' ',' ',' ','H',' ',' ',' ',' ',' ',' ','H' },
 { 'H',' ','H',' ','H','H','H',' ','H',' ','H',' ',' ',' ',' ','H' },
 { 'H',' ','H','H',' ',' ','H',' ','H','H','H',' ','H',' ','H',' ','H' },
 { 'H',' ',' ',' ',' ',' ',' ','H',' ',' ',' ',' ',' ',' ',' ','H',' ','H' },
 { 'H',' ','H','H','H','H','H','H','H','H',' ','H','H','H',' ','H' },
 { 'H',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','H',' ',' ','H' },
 { 'H',' ','H','H','H','H','H',' ','H','H','H',' ',' ',' ',' ','H' },
 { 'H',' ','H',' ',' ',' ',' ','H',' ',' ',' ',' ','H',' ',' ',' ',' ','H' },
 { 'H',' ',' ',' ','H','H','H','H','H','H','H',' ',' ',' ',' ','H' },
 { 'H',' ','H',' ',' ',' ',' ','H',' ',' ',' ',' ','H',' ',' ',' ','H',' ','H' },
 { 'H',' ','H','H','H','H','H',' ','H','H','H','H',' ','H',' ','H' },
 { 'H',' ',' ',' ',' ',' ',' ','H',' ',' ',' ',' ',' ',' ',' ','H',' ','H' },
 { 'H',' ',' ','H','H',' ',' ','H','H','H','H',' ',' ','H','H','H',' ','H' },
 { 'H',' ',' ',' ','H',' ',' ','H',' ',' ',' ',' ',' ',' ','H',' ',' ',' ','H' },
 { 'H','H','H','H','H','H','H','H','H','H','H','H','H','H',' ','H','H' },
};

//纹理生成
AUX_RGBImageRec *LoadBMP(char *Filename)
{
 FILE *File = NULL;
 if (!Filename)
        return NULL;
 fopen_s(&File,Filename, "r");
 //filename 转换为 LPCWSTR
 wchar_t wszClassName[256];
 memset(wszClassName, 0, sizeof(wszClassName));
 MultiByteToWideChar(CP_ACP, 0, Filename, strlen(Filename) + 1, wszClassName,
        sizeof(wszClassName) / sizeof(wszClassName[0]));
 if (File) {
        fclose(File);
        return auxDIBImageLoad(wszClassName);
 }
 return NULL;
}

int LoadGLTextures()
{
 int Status = FALSE;
 AUX_RGBImageRec *TextureImage[3];
```

```
    memset(TextureImage, 0, sizeof(void *) * 3);
    if (TextureImage[0] = LoadBMP("F:/豚豚/计算机图形学/maze/wall1.bmp")) {
        Status = TRUE;
        glGenTextures(1, &texture[0]);
        glBindTexture(GL_TEXTURE_2D, texture[0]);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, TextureImage[0]->sizeX,
TextureImage[0]->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, TextureImage[0]->data);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    }
    if (TextureImage[1] = LoadBMP("F:/豚豚/计算机图形学/maze/ground.bmp")) {
        Status = TRUE;
        glGenTextures(1, &texture[1]);
        glBindTexture(GL_TEXTURE_2D, texture[1]);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, TextureImage[1]->sizeX,
TextureImage[1]->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, TextureImage[1]->data);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    }
    if (TextureImage[2] = LoadBMP("F:/豚豚/计算机图形学/maze/floor.bmp")) {
        Status = TRUE;
        glGenTextures(1, &texture[2]);
        glBindTexture(GL_TEXTURE_2D, texture[2]);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, TextureImage[2]->sizeX,
TextureImage[2]->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, TextureImage[2]->data);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    }
    if (TextureImage[0]) {
        if (TextureImage[0]->data)
            free(TextureImage[0]->data);
        free(TextureImage[0]);
    }
    if (TextureImage[1]){
        if (TextureImage[1]->data)
            free(TextureImage[1]->data);
        free(TextureImage[1]);
    }
    if (TextureImage[2]) {
        if (TextureImage[2]->data)
            free(TextureImage[2]->data);
        free(TextureImage[2]);
    }
    return Status;
```

```
}

void myinit()
{
 //试试加上纹理
 if (!LoadGLTextures())
        return ;

 glClearColor(0.5f, 0.5f, 0.5f, 0.0f);
 glColor3f(1.0, 1.0, 1.0);
 glEnable(GL_DEPTH_TEST);
 glEnable(GL_TEXTURE_2D);

 walllist = glGenLists(2);
 mazelist = walllist + 1;
 balllist = walllist + 2;
 glNewList(walllist, GL_COMPILE);
 drawwalls();
 glEndList();
 glNewList(mazelist, GL_COMPILE);
 drawground();
 glEndList();
 glNewList(mazelist, GL_COMPILE);
 drawtop();
 glEndList();
 glNewList(balllist, GL_COMPILE);
 drawball();
 glEndList();
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluPerspective(60.0, 1.0, 0.1, 60.0);
 glMatrixMode(GL_MODELVIEW);
 glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);//使纹理不变形

}

void drawground(void)
{
 glEnable(GL_TEXTURE_2D);
 glPushMatrix();
 glBindTexture(GL_TEXTURE_2D, texture[2]);

 glBegin(GL_QUAD_STRIP);
 float i, j;
```

```
	for (i = MAZE_WIDTH / 2; i <= MAZE_WIDTH; i = i * 2)
	{
		for (j = -MAZE_HEIGHT / 2; j>-MAZE_HEIGHT * 2; j = j * 2)
		{
			glTexCoord2f(0.0, 0.0); glVertex3f(i - MAZE_WIDTH / 2, 0.0, j +
MAZE_HEIGHT / 2);
			glTexCoord2f(1.0, 0.0); glVertex3f(i, 0.0, j + MAZE_HEIGHT / 2);
			glTexCoord2f(1.0, 1.0); glVertex3f(i, 0.0, j);
			glTexCoord2f(0.0, 1.0); glVertex3f(i - MAZE_WIDTH / 2, 0.0, j);
		}
	}
	glEnd();
	glPopMatrix();
	}

	bool wall(int x, int y) {
	 return (x >= 0 && y >= 0 && x<MAZE_WIDTH && y<MAZE_HEIGHT &&
mazedata[y][x] != ' ');
	}
	bool onopen(int x, int y) {
	 if (wall(x, y)) {
		return(mazedata[y][x] == 'H');
	 }
	}
	void closeit(int x, int y) {
	 if (onopen(x, y))
	 {
		mazedata[y][x] = 'X';
	 }
	}
	bool neighbor(int x, int y, int w, int *nx, int *ny) {
	 switch (w) {
	 case 0:
		*nx = x - 1; *ny = y;	break;
	 case 1:
		*nx = x;	*ny = y + 1; break;
	 case 2:
		*nx = x + 1; *ny = y;	break;
	 case 3:
		*nx = x;	*ny = y - 1; break;
	 default:
		break;
	 }
	 return wall(*nx, *ny);
```

```
}
bool diagnal(int x, int y, int w, int *nx, int *ny) {
 switch (w) {
 case 0:
        *nx = x - 1; *ny = y - 1; break;
 case 1:
        *nx = x - 1; *ny = y + 1; break;
 case 2:
        *nx = x + 1; *ny = y + 1; break;
 case 3:
        *nx = x + 1; *ny = y - 1; break;
 default:
        break;
 }
 return wall(*nx, *ny);
}
void dw(int x, int y, int p) {
 int w = p;
 closeit(x, y);
 do {
        int x2 = 0;
        int y2 = 0;
        if (neighbor(x, y, w, &x2, &y2)) {
                if (onopen(x2, y2)) {
                        dw(x2, y2, (w + 3) % 4);
                }
                else {
                        if ((w + 1) % 4 == p)
                        {
                                return;
                        }
                }
        }
        else {
                float fx;
                float fy;
                if (diagnal(x, y, w, &x2, &y2) && onopen(x2, y2)) {
                        dw(x2, y2, (w + 2) % 4);
                }
                texcoordX = (texcoordX<0.5) ? 1.0f : 0.0f;
                fx = (float)x + ((w == 1 || w == 2) ? 1.0f : 0.0f);
                fy = (float)y + ((w == 0 || w == 1) ? 1.0f : 0.0f);
                glTexCoord2f(texcoordX, 0.0f);
                glVertex3f(fx, fy, 0.0f);
```

```
                glTexCoord2f(texcoordX, 1.0f);
                glVertex3f(fx, fy, 1.0f);
        }
        w++; w %= 4;
 } while (w != p);
 return;
}
void drawwalls() {
 glEnable(GL_TEXTURE_2D);

 glBindTexture(GL_TEXTURE_2D, texture[0]);

 glBegin(GL_QUAD_STRIP);
 glColor3f(1.0, 1.0, 1.0);
 glVertex3f(0.0f, 0.0f, 0.0f);
 glVertex3f(0.0f, 0.0f, 1.0f);
 dw(0, 0, 0);
 glEnd();
}
void drawtop() {
 int x, y;

 glPushMatrix();
 glBindTexture(GL_TEXTURE_2D, texture[1]);

 glBegin(GL_QUADS);
 for (y = 0; y<MAZE_HEIGHT; y++) {
        for (x = 0; x<MAZE_WIDTH; x++) {
                if (wall(x, y)) {
                        mazedata[y][x] = 'X';
                }else{
                        glTexCoord2f(1.0f, 1.0f);
                        glVertex3f(x + 0.0f, y + 0.0f, 1.0f);
                        glTexCoord2f(0.0f, 1.0f);
                        glVertex3f(x + 1.0f, y + 0.0f, 1.0f);
                        glTexCoord2f(0.0f, 0.0f);
                        glVertex3f(x + 1.0f, y + 1.0f, 1.0f);
                        glTexCoord2f(1.0f, 0.0f);
                        glVertex3f(x + 0.0f, y + 1.0f, 1.0f);
                }
        }
 }
 glEnd();
 glPopMatrix();
```

```
}
void forward(float px, float py, float bf) {//碰撞检测
 int x = ((int)player_x);
 int y = ((int)player_y);
 if ((px> x + 1.0f - bf) && wall(x + 1, y)) {
        px = (float)(x)+1.0f - bf;
 }
 if (py> y + 1.0f - bf && wall(x, y + 1)) {
        py = (float)(y)+1.0f - bf;
 }
 if (px< x + bf && wall(x - 1, y)) {
        px = (float)(x)+bf;
 }
 if (py< y + bf && wall(x, y - 1)) {
        py = (float)(y)+bf;
 }
 player_x = px;
 player_y = py;
}
void drawball()//画小地图中的小红点
{
 glDisable(GL_TEXTURE_2D);
 glColor3f(1.0, 0.0, 0.0);
 glutSolidSphere(0.2f, 15, 15);
}
void navmaze1()
{
 forward(player_x + player_s*(float)sin(player_h*3.14 / 180),
        player_y + player_s*(float)cos(player_h*3.14 / 180), 0.2f);
 //cout<<player_x<<player_y<<endl;//X、Y 坐标
 player_h += player_t;
 player_b = 3 * player_b / 4 + player_t / 4;
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glLoadIdentity();
 glPushMatrix();
 glRotatef(-90.0f, 1.0f, 0.0f, 0.0f);
 glRotatef(player_h, 0.0f, 0.0f, 1.0f);
 glTranslatef(-player_x, -player_y, -0.5f);
 glCallList(walllist);
 glPopMatrix();
}
void navmaze2()
{
 forward(player_x + player_m*player_s*(float)sin(player_h*3.14 / 180),
```

```
        player_y + player_m*player_s*(float)cos(player_h*3.14 / 180), 0.2f);
 //cout<<player_x<<player_x<<endl;
 player_h += player_t;
 player_b = 3 * player_b / 4 + player_t / 4;
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glLoadIdentity();
 glOrtho(-16.0, 16.0, -16.0, 16.0, -2.0, 20.0);
 glPushMatrix();
 glRotatef(90.0f, 0.0f, 0.0f, 1.0f);
 glTranslatef(-MAZE_WIDTH / 2, -MAZE_HEIGHT / 2, -0.5f);
 glCallList(walllist);
 glCallList(mazelist);
 glPushMatrix();
 glTranslatef(player_x, player_y, 0.5f);
 glCallList(balllist);
 glPopMatrix();
 glPopMatrix();
}
void myDisplay()
{
 if (status == 1)//绘制第一视角图
 {
        if (searchroute == true)
        {
        }
        else navmaze1();
 }
 if (status == 3)//绘制小地图
 {
        if (searchroute == true)
        {
        }
        else navmaze2();
 }
 glFlush();                //刷新到输出
 glutSwapBuffers(); //交换缓冲区
}
void myReshape(int w, int h)
{
 glViewport(0, 0, w, h);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
```

```
 glutPostRedisplay();
}
void specialKeys(int key, int x, int y)
{
 switch (key)
 {
 case GLUT_KEY_LEFT:
        keystate[2] = true;
        player_t = -2.0f;
        break;
 case GLUT_KEY_RIGHT:
        keystate[3] = true;
        player_t = 2.0f;
        break;
 case GLUT_KEY_UP:
        keystate[0] = true;
        player_s = 0.01f;
        break;
 case GLUT_KEY_DOWN:
        keystate[1] = true;
        player_s = -0.01f;
        break;
 default:break;
 }
}
void keyboard(unsigned char key, int x, int y)//按键切换状态信息
{
 switch (key)
 {
 case '1':
        status = 1;
        break;
 case '3':
        status = 3;
        break;
 default:break;
 }
 glutPostRedisplay();     //重新绘制
}
void upSpecialKeyboard(int key, int x, int y)
{
 switch (key)
 {
 case GLUT_KEY_LEFT:
```

```
            keystate[2] = false;
            player_t = 0.0f;
            break;
     case GLUT_KEY_RIGHT:
            keystate[3] = false;
            player_t = 0.0f;
            break;
     case GLUT_KEY_UP:
            keystate[0] = false;
            player_s = 0.0f;
            break;
     case GLUT_KEY_DOWN:
            keystate[1] = false;
            player_s = 0.0f;
            break;
     default:break;
     }
     //     glutPostRedisplay();
    }
    void idle()
    {
     if (keystate[0] || keystate[1] || keystate[2] || keystate[3]) glutPostRedisplay();
     else {}
    }//释放按键后就进入空闲状态，如果空闲状态不一直重复绘图，就会画面停滞。次方法缺点是开销很大



    void main(int argc, char** argv)
    {
     glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
     glutInitWindowSize(800, 800);
     glutCreateWindow("maze");
     glutDisplayFunc(myDisplay);
     myinit();
     glutSpecialFunc(specialKeys);
     glutKeyboardFunc(keyboard);
     glutSpecialUpFunc(upSpecialKeyboard);
     glutIdleFunc(idle);
     glutMainLoop();
    }
```