浙江大学

# 《第一人称圣庙逃亡》

# 设计说明报告

小组成员：王睿

刘振东

卫宇鹏

指导老师：吴鸿智 朱瑞昇

所在学院：计算机学院

2021 年 1 月 15 日

目录


一、 引言 3
1.1 背景 3
二、 综合描述 3
2.1 运行环境 3
2.2 游戏玩法 3
2.2.1 游戏主界面。 3
2.2.2 游戏界面 4
2.3 游戏流程 5
三、 软件设计说明 5
3.1 游戏组成要素 5
3.1.1 火焰效果 5
3.1.2 杆 5
3.1.3 陷井 6
3.1.4 ufo 装饰 6
3.1.5 小地图以及游戏状态 6
3.2 子模块功能实现及其原理 7
3.2.1 碰撞检测模块 7
3.2.2 摄像机模块 7
3.2.3 物理运动模块 8
3.2.4 粒子系统 8
3.2.5 地图类 8
3.2.6 obj loader 9
3.2.7 天空盒 9
3.2.8 光照 10
四、 改进的方向 10
五、参考文档 10

# 一、引言

## 1.1 背景

本游戏灵感来源于《神庙逃亡》以及《吃豆人》，是一款跑酷冒险类游戏，结合 OpenGL，我们将原有的轨道地图换为自由地图。保留“怪物”追赶机制，地图上会设有相关的障碍，经过碰撞检测，若发生碰撞，则相应的缩短与怪物的距离。游戏最终目的，吃完所有金币。

游戏开始玩家在世界中的不断前进，利用鼠标改变方向，在世界里面进行 3D 漫游。

# 二、综合描述

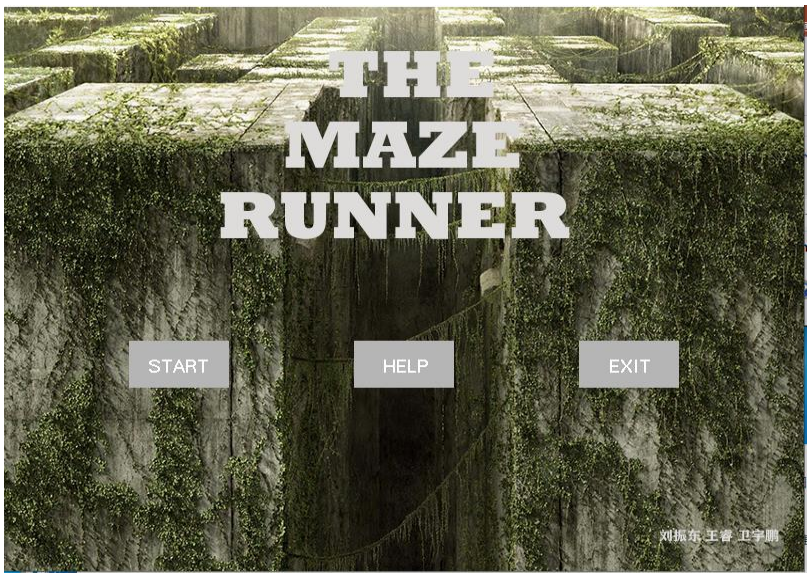
## 2.1 运行环境

本游戏利用 glut 为主要 openGL 图形工具，结合其他相关的辅助库（SOIL、glm 等）实现。

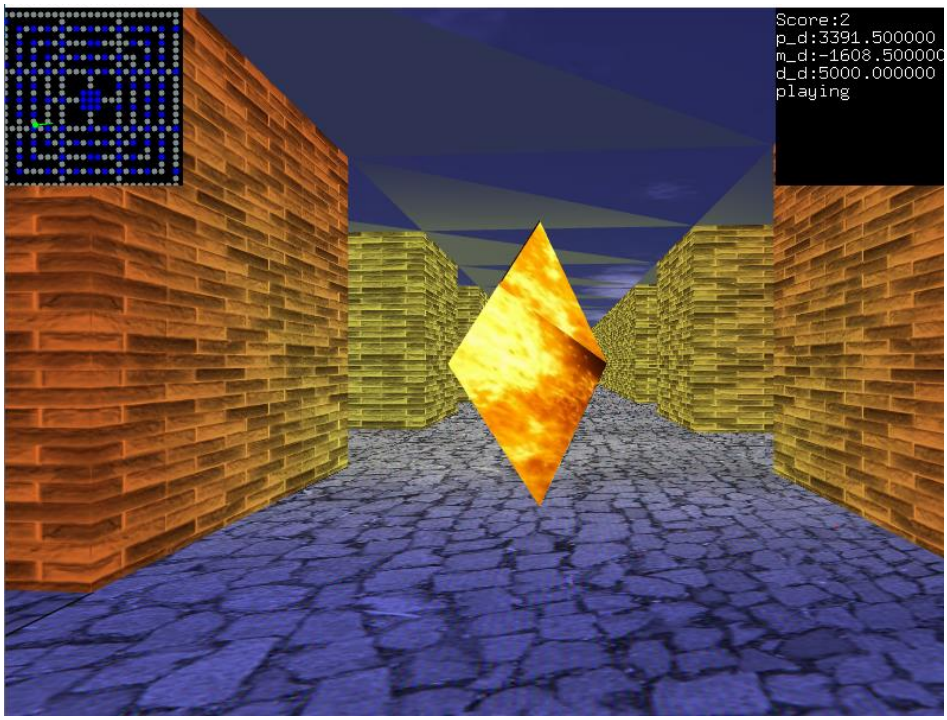
## 2.2 游戏玩法

2.2.1 游戏主界面。

主要采用鼠标交互，‘START’进入游戏界面；‘HELP’：进入说明界面；‘EXIT’：退出游戏。



2.2.2 游戏界面




游戏界面需要鼠标和按键进行交互，其中鼠标点击拖动实现方向的改变；

按键功能如下：

空格：暂停

‘R’：重新开始游戏

‘J’：跳跃

‘K’：下蹲

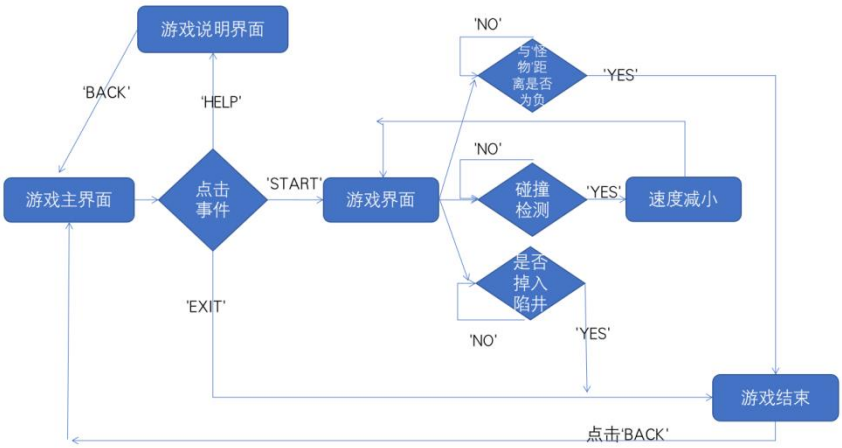
‘u’：实现三种光源（0 随摄像机移动的聚光灯，1 定点聚光灯，2 平行光）

‘i’：改变光照强度(0~1 10 等份)

‘o’：改变光的 color

‘p’：在定点聚光灯下改变光源位置

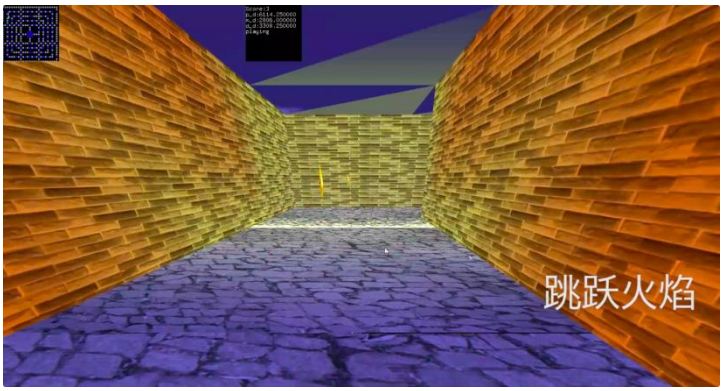
## 2.3 游戏流程



# 三、软件设计说明

## 3.1 游戏组成要素

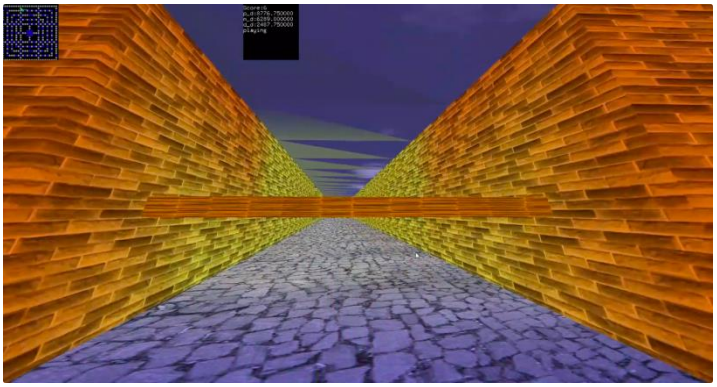
### 3.1.1 火焰效果



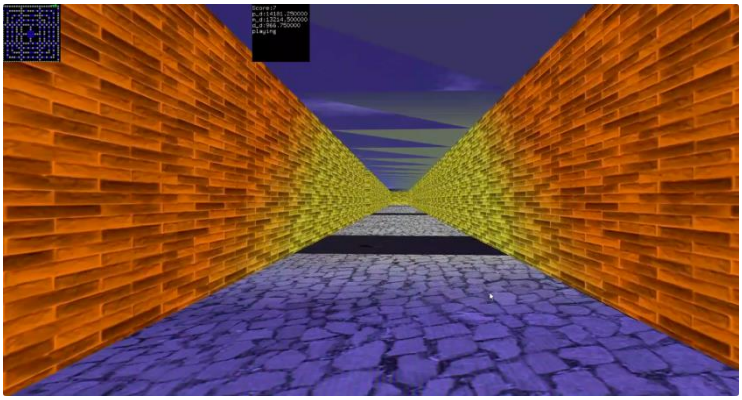

(1) 设计原理

搭建粒子系统，为每一个粒子，建立初始位置，移动方向，以及生命周期。

### 3.1.2 杆

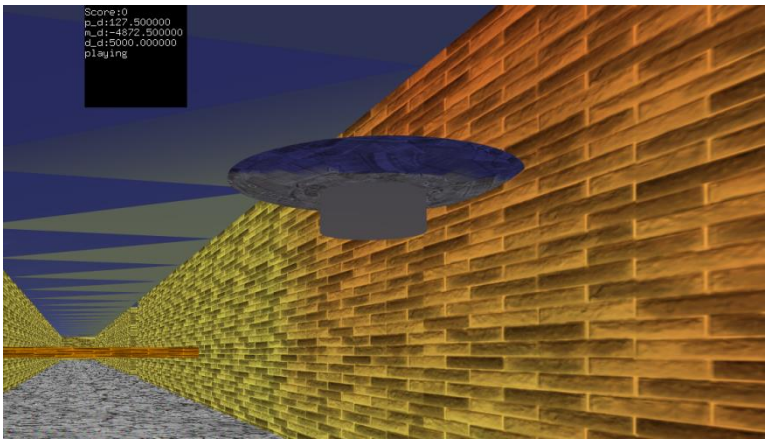


### 3.1.3 陷井



3.1.4 ufo 装饰

主要通过.obj 文件的导入实现。




3.1.5 小地图以及游戏状态


## 3.2 子模块功能实现及其原理

3.2.1 碰撞检测模块

碰撞检测以球体以及 AABB 盒作为包围盒。检测触碰金币通过球体检测，检测触碰火焰或者杆通过 AABB 盒检测。

球体检测的原理是计算摄像机中心和金币的中心距离，若小于某个阈值，则属于触碰。

对于 AABB 盒，以摄像机位置做一个与坐标系平行的包围盒，对火焰/杆作包围盒，将包围盒在三个平面下投影，在二维的平面检测，若三个平面下都发生碰撞，则三维检测为碰撞。下面是二维检测的原理。

二维平面下，两个坐标系下的坐标各有一个 min 和 max。二维是否发生碰撞代码如下：

```
class AABB_2D {
public:
GLfloat min_x, max_x;
GLfloat min_y, max_y;
void set(GLfloat x1, GLfloat x2, GLfloat y1, GLfloat y2) {
min_x = x1;
max_x = x2;
min_y = y1;
max_y = y2;
```

```
}
};

bool check_2D_collide(AABB_2D A,AABB_2D B) {
if (A.max_x < B.min_x) return 0;
if (A.max_y < B.min_y) return 0;
if (B.max_x < A.min_x) return 0;
if (B.max_y < A.min_y) return 0;
return 1;
}
```

若三个平面下都碰撞，则 3 维情况下检测为碰撞，flag 为返回类型，是 fire 还是 rod

```
if (flag) {

B_xy.set(x - grid_size / 4, x + grid_size / 4, y -grid_size, y);
B_xz.set(x - grid_size / 4, x + grid_size / 4, z - grid_size / 4, z + grid_size / 4);
B_yz.set(y-grid_size, y , z - grid_size / 4, z + grid_size / 4);
checkxy =check_2D_collide(A_xy,B_xy) ;
checkxz = check_2D_collide(A_xz, B_xz);
checkyz = check_2D_collide(A_yz, B_yz);
}
if (checkxy&&checkxz&&checkyz) {
return flag;
}
```

### 3.2.2 摄像机模块

```
GLfloat Camera_x, Camera_y, Camera_z;//
GLfloat Gaze_x, Gaze_y, Gaze_z;
GLfloat Camera_radius=grid_size/2;
static GLfloat move_angle = -90.0;
```

通过检测，若发现摄像机移动方向可通，则进行下面计算：

```
//if(ctrl_grid[px_now][pz]!=-1)
Camera_z += speed1 * sin(move_angle * 2 * PI / 360);
//if(ctrl_grid[px][pz_now]!=-1)
Camera_x += speed1 * cos(move_angle * 2 * PI / 360);
```

每一次键盘事件'w'，可以触发移动效果，系统规则是按时间自动设置键盘事件'w'，所以不用人工按下‘w’键。

### 3.2.3 物理运动模块

(1)摄像机水平方向：

$$V = V0 + a * dt;\ (V < V_{max})$$
$$S = V0 * dt + (1/2) * a * (dt)^2;$$

(2)垂直方向：

$$V_y = V_{y0} + g * dt;$$
$$S_y = V_{y0} * dt + (1/2) * g * (dt)^2;$$
$$\text{跳跃}(jumpstate):\ g < 0, V_y0 > 0;$$
$$\text{下蹲}(downstate):\ g > 0, V_y0 < 0;$$
$$\text{其他状态下，}\ S_y\text{衡}0,\ V_y\text{衡}0$$

### 3.2.4 粒子系统

```
#define MAX_PARTICLES 1000 //最大的粒子数
class particle {
public:
bool active;
GLfloat fire_grid_size;
float life;//lift time
float fade;//the decreasing of life
float r, g, b;//RGB
float x, y, z;//position
float xi, yi, zi;//mov direction
float xg, yg, zg;//Acceleration
};
```

每一个粒子有其独立的生存周期，运动方向，和当前 size，以及活跃状态；若为活跃状态则在当下粒子运动到的地方绘制；若检测到 life<0，则重新初始化粒子。

### 3.2.5 地图类

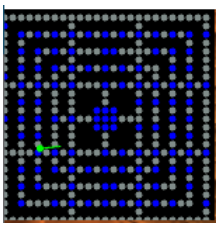
整个 3D 世界地图通过 mygrid 类控制

```
class grid {
public:
int walltype[4];// 0 前  1 后  2 左  3 右边  若值为 0  表示没有墙
int type;
int is_rod;
GLfloat position[3];//zï0
int livestate;//the state of live

void init(GLfloat x, GLfloat y, GLfloat z);
void draw_grid() ;
void draw_wall(GLint text) {
if(walltype[0]!=0){//前
}
if (walltype[1] != 0) {//后
}
if (walltype[2] != 0) {//左
}
if (walltype[3] != 0) {//右边
}}
```

有了这样的 grid 类，整体的地图只需要其中 ctrl_grid[grid_num][grid_num]决定当前位置显示的类型，金币/杆/火焰。

```
#define NOTHING 1
#define COIN 2
#define ISROD_ 3 //朝向
#define ISROD_1 4
#define fire_ 5
#define fire_1 6
```

ctrl_grid 可用于自窗口的小地图绘制：



3.2.6 obj loader

我们自己实现了一个 obj parser，支持导入两种格式的 obj 文件：f x y z 格式与 f x/x/x y/y/y z/z/z 格式。我们将 obj loader 封装在一个 objLoader 中，类的声明如下：

```
class objLoader
{
public:
 objLoader(const char* filename, GLuint type, GLfloat x, GLfloat y, GLfloat z, GLfloat scale);
 void draw();
private:
 std::vector<glm::vec3> out_vertices;
 std::vector<glm::vec2> out_uvs;
 std::vector<glm::vec3> out_normals;
 GLuint type;
};
```

该类包含 4 个成员变量，分别记录空间点、纹理坐标、法向量坐标以及类型（用于区别两种 obj 格式）。解析的逻辑很简单，首先根据每行的第一个字符确定改行所表述的信息是哪种类型（v/vt/vn/f），如果是前三种类型，则分别记录至 temp_vertices，temp_uvs 和 temp_normals 的 vector 内，如果是 f 类型，则记录点坐标、纹理坐标和法向量坐标的 index 至各自的 index 向量内。

```
 if (strcmp(lineHeader, "v") == 0) {
  glm::vec3 vertex;
  fscanf(file, "%f %f %f\n", &vertex.x, &vertex.y, &vertex.z);
  temp_vertices.push_back(vertex);
 }
 else if (strcmp(lineHeader, "vt") == 0) {
  glm::vec2 uv;
  fscanf(file, "%f %f\n", &uv.x, &uv.y);
  //uv.y = -uv.y; // Invert V coordinate since we will only use DDS texture, which are inverted. Remove if yo
u want to use TGA or BMP loaders.
  temp_uvs.push_back(uv);
 }
 else if (strcmp(lineHeader, "vn") == 0) {
  glm::vec3 normal;
  fscanf(file, "%f %f %f\n", &normal.x, &normal.y, &normal.z);
  temp_normals.push_back(normal);
 }
 else if (strcmp(lineHeader, "f") == 0) {
  std::string vertex1, vertex2, vertex3;
  unsigned int vertexIndex[3], uvIndex[3], normalIndex[3];
  int matches = fscanf(file, "%d/%d/%d %d/%d/%d %d/%d/%d\n", &vertexIndex[0], &uvIndex[0], &normall
ndex[0], &vertexIndex[1], &uvIndex[1], &normalIndex[1], &vertexIndex[2], &uvIndex[2], &normalIndex[2]);
  if (matches != 9) {
   printf("NOT TYPE1!! File can't be read by our simple parser :-( Try exporting with other options\n");
```

```
    fclose(file);
    exit(-1);
  }
  vertexIndices.push_back(vertexIndex[0]);
  vertexIndices.push_back(vertexIndex[1]);
  vertexIndices.push_back(vertexIndex[2]);
  uvIndices.push_back(uvIndex[0]);
  uvIndices.push_back(uvIndex[1]);
  uvIndices.push_back(uvIndex[2]);
  normalIndices.push_back(normalIndex[0]);
  normalIndices.push_back(normalIndex[1]);
  normalIndices.push_back(normalIndex[2]);
}
```

全部读入后。我们就可以根据 index 在对应的坐标向量内寻找真正的坐标并存在成员变量中。

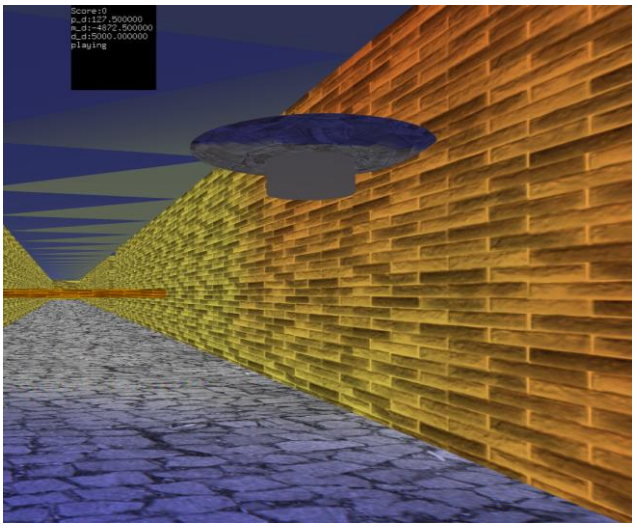
```
// For each vertex of each triangle
for (unsigned int i = 0; i < vertexIndices.size(); i++) {

  // Get the indices of its attributes
  unsigned int vertexIndex = vertexIndices[i];
  unsigned int uvIndex = uvIndices[i];
  unsigned int normalIndex = normalIndices[i];

  // Get the attributes thanks to the index
  glm::vec3 vertex = temp_vertices[vertexIndex - 1];
  glm::vec2 uv = temp_uvs[uvIndex - 1];
  glm::vec3 normal = temp_normals[normalIndex - 1];

  // Put the attributes in buffers
  out_vertices.push_back(vertex);
  out_uvs.push_back(uv);
  out_normals.push_back(normal);
```

绘制 obj 时，因为是面都是由三角形构成，因此我们只需要开启贴图绘制，并设置 glBegin(GL_TRIANGLES)，即可将纹理贴图与面同时绘制出来。

```
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, My_texture.obj_texture[0]);

  glBegin(GL_TRIANGLES);
  for (int i = 0; i < out_uvs.size(); i ++) {
    glTexCoord2f(out_uvs[i].x, out_uvs[i].y);
    glVertex3f(out_vertices[i].x, out_vertices[i].y, out_vertices[i].z);
  }
```

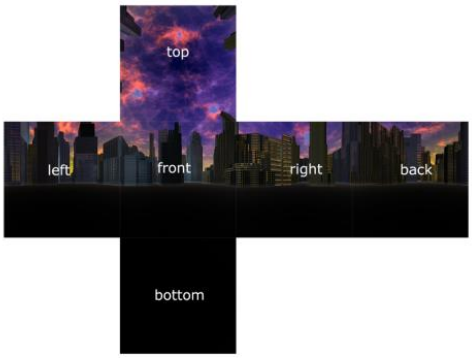
```
glEnd();

glDisable(GL_TEXTURE_2D);
```

我们导入了一个 UFO 的模型，效果如下：



### 3.2.7 天空盒

即立方体贴图




天空盒(Skybox)是一个包裹整个场景的立方体，它由 6 个图像构成一个环绕的环境，给玩家一种他所在的场景比实际的要大得多的幻觉。我们将贴图导入后只需要给它设置对应的纹理坐标即可实现。

```
void Texture::Draw_Skybox(GLfloat x, GLfloat y, GLfloat z, GLfloat width, GLfloat height, GLfloat length)
{
 // Center the Skybox around the given x,y,z position
 x = x - width / 2;
 y = y - height / 2;
 z = z - length / 2;

 // Draw Front side
 glEnable(GL_TEXTURE_2D);
 glBindTexture(GL_TEXTURE_2D, skyboxTexture[SKYFRONT]);
```

```
glBegin(GL_QUADS);
glTexCoord2f(1.0f, 0.0f); glVertex3f(x, y + height, z + length);
glTexCoord2f(1.0f, 1.0f); glVertex3f(x, y, z + length);
glTexCoord2f(0.0f, 1.0f); glVertex3f(x + width, y, z + length);
glTexCoord2f(0.0f, 0.0f); glVertex3f(x + width, y + height, z + length);
glEnd();

// Draw Back side
glBindTexture(GL_TEXTURE_2D, skyboxTexture[SKYBACK]);
glBegin(GL_QUADS);
glTexCoord2f(1.0f, 0.0f); glVertex3f(x + width, y + height, z);
glTexCoord2f(1.0f, 1.0f); glVertex3f(x + width, y, z);
glTexCoord2f(0.0f, 1.0f); glVertex3f(x, y, z);
glTexCoord2f(0.0f, 0.0f); glVertex3f(x, y + height, z);
glEnd();

// Draw Left side
glBindTexture(GL_TEXTURE_2D, skyboxTexture[SKYLEFT]);
glBegin(GL_QUADS);
glTexCoord2f(1.0f, 1.0f); glVertex3f(x + width, y, z + length);
glTexCoord2f(0.0f, 1.0f); glVertex3f(x + width, y, z);
glTexCoord2f(0.0f, 0.0f); glVertex3f(x + width, y + height, z);
glTexCoord2f(1.0f, 0.0f); glVertex3f(x + width, y + height, z + length);
glEnd();

// Draw Right side
glBindTexture(GL_TEXTURE_2D, skyboxTexture[SKYRIGHT]);
glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f); glVertex3f(x, y + height, z + length);
glTexCoord2f(1.0f, 0.0f); glVertex3f(x, y + height, z);
glTexCoord2f(1.0f, 1.0f); glVertex3f(x, y, z);
glTexCoord2f(0.0f, 1.0f); glVertex3f(x, y, z + length);
glEnd();

// Draw Up side
glBindTexture(GL_TEXTURE_2D, skyboxTexture[SKYUP]);
glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f); glVertex3f(x + width, y + height, z);
glTexCoord2f(1.0f, 0.0f); glVertex3f(x, y + height, z);
glTexCoord2f(1.0f, 1.0f); glVertex3f(x, y + height, z + length);
glTexCoord2f(0.0f, 1.0f); glVertex3f(x + width, y + height, z + length);
glEnd();

// Draw Down side
```

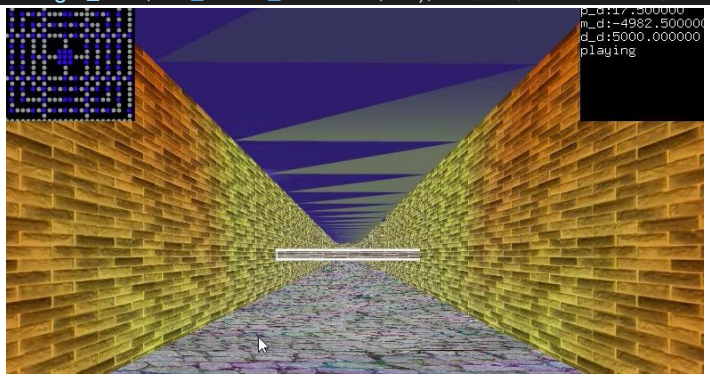
```
 glBindTexture(GL_TEXTURE_2D, skyboxTexture[SKYDOWN]);
 glBegin(GL_QUADS);
 glTexCoord2f(0.0f, 0.0f); glVertex3f(x + width, y, z + length);
 glTexCoord2f(1.0f, 0.0f); glVertex3f(x, y, z + length);
 glTexCoord2f(1.0f, 1.0f); glVertex3f(x, y, z);
 glTexCoord2f(0.0f, 1.0f); glVertex3f(x + width, y, z);
 glEnd();
 glDisable(GL_TEXTURE_2D);
}
```

3.2.8 光照

聚光灯：

位置性光源

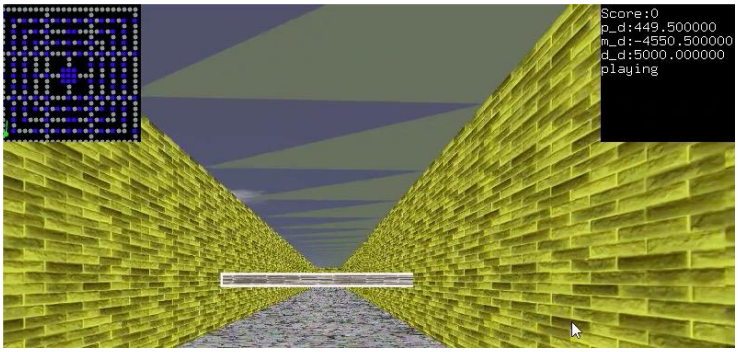
```
spot_position[0] = 0; spot_position[1] = -1; spot_position[2] = 0;
glLightf(GL_LIGHT0 + light_num, GL_SPOT_CUTOFF, 45);//点辐射光源，设置聚光灯的角度,
```




平行光：

```
light_position[3] = 0;//无限远；
```

将 light_position[3]的参数设置为 0.即可表示平行光，此时不是位置性光源，所以角度此时没用。




# 四、改进的方向

这次大作业的方向切入点主要是游戏的控制逻辑，没有很关注场景的真实感。后续改进的方向主要是实现 shader map 以及增加场景特效。

# 五、参考文档


- 碰撞检测 https://www.cnblogs.com/lyggqm/p/5386174.html
- 光照 https://www.cnblogs.com/jizhen521/archive/2013/05/13/3075729.html
- 粒子系统：https://blog.csdn.net/qq_31615919/article/details/78968434
- 子窗口：https://my.oschina.net/u/4312361/blog/4288798
- 天空盒：
  https://learnopengl-cn.github.io/04%20Advanced%20OpenGL/06%20Cubemaps/