Project report for the CG 100433 course

Project Title

Horror Corridor 恐怖走廊

Team member

* 1950509 马家昱
* 1951590 董浩
* 1951858 陶思月
* 1953080 田宇
* 1956211 覃雯

Abstract

本小组意在使用 C++实现一个以 OpenGL 相关图形库为基础的第一人称恐怖游戏。游戏场景为氛围总体昏暗的无尽（自动生成）走廊，具有一定数量的交互要素（门锁、油画等），并且使用鼠标键盘进行游戏控制，并希望设计并搭配合适的剧情与音效。

Motivation

本小组在初期讨论该做什么项目的时候，分别提出了几个比较感兴趣的元素：无尽地图、随机生成、第一人称、可移动探索等元素，并且能够区别于往届学长学姐们的作品，做一个让人眼前一亮、有意思的作品。 在一阵头脑风暴后，融合了上述的几个元素，决定制作一款第一人称迷宫探索恐怖游戏，其中包含一条随机生成的无尽的走廊地图、恐怖场景氛围以及可控制的人物来进行移动并且探索等元素。

The Goal of the project

- 实现场景渲染，包括走廊和小物件模型
- 实现游戏的物理效果，玩家不可移动到走廊外
- 实现玩家的简单交互
- 实现对光源的模拟，主要为人造光源，点光源、聚光灯
- 实现走廊的自动生成
- 法线贴图、镜面光贴图等效果
- 音效及触发机制烘托游戏氛围

The Scope of the project

- 本游戏为第一人称视角，所以玩家的形象不会出现在画面中，摄像机视角即为玩家视角
- 并不使用物理引擎，通过坐标判断保证玩家不会移动出走廊的范围

Involved CG techniques

- 模型导入和移动
- 图形渲染流水线，灯光模拟

Project contents

一个恐怖氛围游戏，主要特点如下

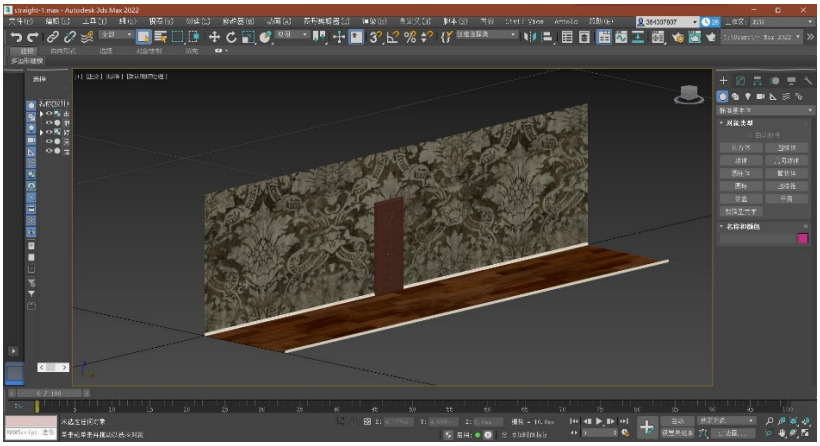
1. 游戏场景基于无限长走廊，通过随机算法无限生成。
2. 玩家通过键鼠控制角色，WASD 控制水平移动，可在走廊里移动。
3. 环境音效的配合，玩家移动时会有脚步声；游戏中有鬼怪形象，如果玩家被追赶上，背景音乐会有变化。
4. 场景和模型的渲染，通过不同光源的使用，点光源和聚光灯营造游戏氛围

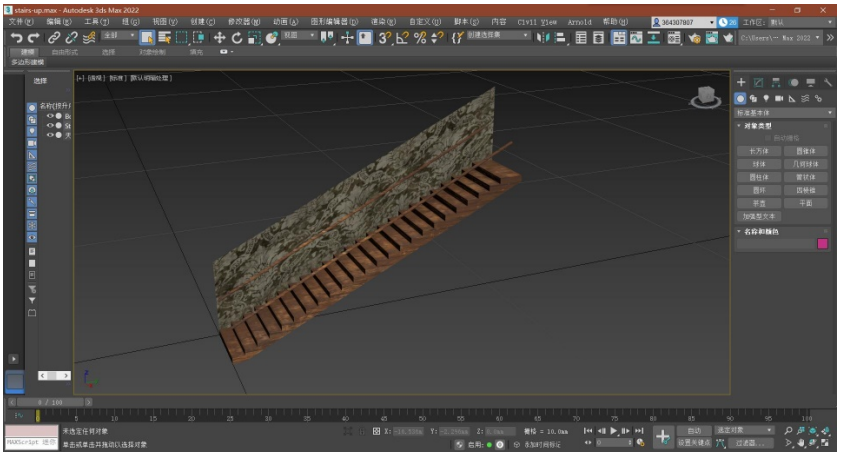
Implementation

一、模型

通过 3DsMax 进行制作并贴图，也在网络上下载了一些模型，修改完善，最后得到.obj 和.mtl 文件导入，使用的第三方库：assimp，将读出的数据转化为对应的 mesh 结构的数据。

模型在 3DsMax 示例如下：





二、光源和贴图

为了营造游戏恐怖的氛围，游戏光照整体较为昏暗，采用了点光源和聚光灯两种光源。其中点光源实现走廊中广泛存在的昏暗照明，聚光灯以手电筒的形式呈现，玩家可以选择是否开启手电筒。

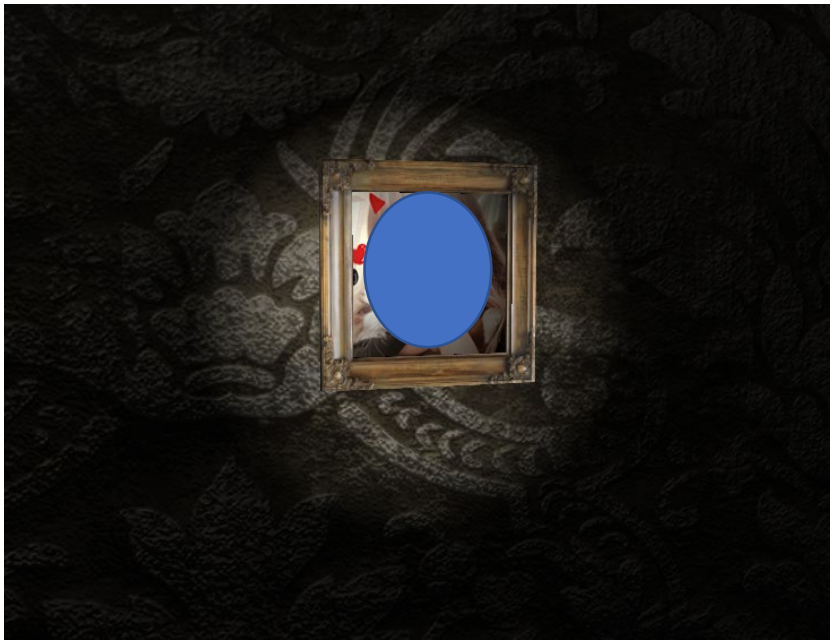
光照采用了 Blinn-Phong 光照模型，在着色器传入顶点位置、光照位置、法线、纹理坐标即可。对于不同的模型和材质，可以传入不同的参数，以实现更为逼真的光照效果。我们根据不同材质的特点，实现了环境光、漫反射、镜面反射，将几者累加作为光照效果。

为了呈现更画框真实的镜面光效果，我们也使用一个专门用于镜面高光纹理贴图，通过贴图每个像素的亮度来获取。

除此之外，为了使环境更为逼真，我们对走廊的墙壁实现了法线贴图的效果，提升了墙面的细节呈现。法线贴图是每个 fragment 使用各自的法线，而不是一个面上所有 fragment 使用同一个法线的方法。

在切线空间中进行光照，法线贴图向量总是指向这个坐标空间的正 z 方向，所有的光照向量都相对与这个正 z 方向进行变换。通过计算表面的 TBN 矩阵，可以把切线空间的 z 方向和表面的法线方向对齐。非常便利的是，我们采用 Assimp 导入模型时，调用其中的 ReadFile 函数时，会为顶点自动计算出切线和副切线，不需要手动计算。

法线贴图和镜面光贴图如下：(第二张为油画内容打码，不影响画框镜面光效果的查看)



在恐怖走廊这一游戏设定中，我们的光源，尤其是固定的灯部件“发出的“点光源，有不定时闪烁的效果，该效果通过随机来触发效果，并伴有音效。

通过键盘的”Q“键，可以控制手电筒的开关。

三、音效

我们使用为 irrKlang 的音频管理库来实现特定场景下的声音效果。

通过 irrKlang 库，可以实现简单音乐的播放，我们在此基础上对其功能进行改进。实现每次触发同一音效时，不是从头播放，而是继续上次播放停止的位置播放。

播放一段音乐特定时间通过记录播放音乐的播放时间来实现。

```
1.  //最简单地播放一段音乐 可选择是否循环 背景音乐
2.  void mySoundEngline::play2D(const char musicPath[], int isLoop)
3.  {
4.      SoundEngine->play2D(musicPath, isLoop);
5.  }
6.
7.  //播放某段音频特定地秒数 期间再次触发本函数不再从头开始播放 脚步声等 可以用此函数
8.  void mySoundEngline::playFewSecond(const char musicPath[], double time)
9.  {
10.     string nowPath = musicPath;
11.     int isexist = 0;
12.     int isovertime = 0;
13.     for (vector<Music>::iterator it = PlayingMusic.begin(); it != PlayingMus
    ic.end(); ++it)
14.     {
15.         if ((*it).path == nowPath)
16.         {
17.             isexist = 1;
18.             if (double(clock()) / 1000.0 - (*it).start_time > (*it).play_tim
    e)
19.             {
20.                 PlayingMusic.erase(it);
21.                 isexist = 0;
22.             }
23.             break;
24.         }
25.     }
26.     if (!isexist)
27.     {
28.         Music newM = { double(clock()) / 1000.0 ,time,musicPath };
29.         PlayingMusic.push_back(newM);
30.         SoundEngine->play2D(musicPath, 0);
31.     }
32. }
```

四、场景

这一部分是我们游戏呈现的主题环境。包括自动生成的，带方向变化的无限走廊，以及物理判断，保证玩家不会移动的走廊之外的范围。

走廊无限自动生成的思路是，一次记录三个走廊及其类型、位置信息，包括方向、左下

角坐标位置、直走廊/楼梯/转弯类型、变换矩阵等。分别记录为 mid、front、back。

然后根据 mid 走廊类型来随机生成一种，其中直走廊只能接转弯或上下楼，转弯走廊不能再接转弯走廊，两个楼梯走廊不能相邻。通过与类型相匹配的模型，即可在场景中将走廊绘制渲染出来。

```
corridorType Scene::genNewCorridor(corridorType mid) //随机返回一种走廊类型
{
    if (mid == straightCorridor) //直走廊只能接转弯或上下楼
    {
        corridorType list[4] = { cornerCorridorL, cornerCorridorR, stairsCorridorU, stairsCorridorD };
        int rand_num = rand() % 4;
        return list[rand_num];
    }
    else if (mid == cornerCorridorL || mid == cornerCorridorR) //转弯走廊不能再接转弯走廊
    {
        corridorType list[3] = { stairsCorridorU, stairsCorridorD, straightCorridor };
        int rand_num = rand() % 3;
        return list[rand_num];
    }
    else
    {
        corridorType list[3] = { cornerCorridorL, cornerCorridorR, straightCorridor }; //两个楼梯不能相邻
        int rand_num = rand() % 3;
        return list[rand_num];
    }
}
```

玩家不会移动到走廊范围外的实现思路是：

根据不同的走廊类型和走廊的方向，结合模型的具体坐标位置，当玩家通过键盘控制移动摄像机位置，从而坐标超过走廊的边界范围时，就重新设置坐标，使其不超出走廊的范围。

如左右转弯走廊，使用判断左右的方法来做，根据 xP、zP 两种走廊方向，计算四个协助判断的距离量，然后根据具体的情况，将 x、z 坐标限制在走廊内，即可完成这一物理实现。

```
// 2 种转弯统一，使用判断左右的方法来做
    else if (this->type == cornerCorridorL)
```

```
3.      {
4.          switch (dir)
5.          {
6.          case xP:
7.              dis1 = loc.z - (pos.z - _width / 2);
8.              dis2 = loc.x + 8 - (pos.x - _width / 2);
9.              dis3 = (pos.z + _width / 2) - (loc.z + 2);
10.             dis4 = (pos.x + _width / 2) - (loc.x + 10);
11.             if (dis1 > 0 && dis2 > 0)
12.             {
13.                 if (dis1 < dis2)
14.                 {
15.                     pos.z = loc.z + _width / 2;
16.                 }
17.                 else
18.                 {
19.                     pos.x = loc.x + 8 + _width / 2;
20.                 }
21.             }
22.             if (dis3 > 0)
23.             {
24.                 pos.z = loc.z + 2 - _width / 2;
25.             }
26.             if (dis4 > 0)
27.             {
28.                 pos.x = loc.x + 10 - _width / 2;
29.             }
30.
31.             break;
32.         case zP:
33.             dis1 = pos.x + _width / 2 - loc.x;
34.             dis2 = loc.z + 8 - (pos.z - _width / 2);
35.             dis3 = loc.x - 2 - (pos.x - _width / 2);
36.             dis4 = (pos.z + _width / 2) - (loc.z + 10);
37.             if (dis1 > 0 && dis2 > 0)
38.             {
39.                 if (dis1 < dis2)
40.                 {
41.                     pos.x = loc.x - _width / 2;
42.                 }
43.                 else
44.                 {
45.                     pos.z = loc.z + 8 + _width / 2;
46.                 }
```

```
            }
            if (dis3 > 0)
            {
                pos.x = loc.x - 2 + _width / 2;
            }
            if (dis4 > 0)
            {
                pos.z = loc.z + 10 - _width / 2;
            }

            break;
        case xN:
            dis1 = pos.z + _width / 2 - loc.z;
            dis2 = pos.x + _width / 2 - (loc.x - 8);
            dis3 = loc.z - 2 - (pos.z - _width / 2);
            dis4 = loc.x - 10 - (pos.x - _width / 2);
            if (dis1 > 0 && dis2 > 0)
            {
                if (dis1 < dis2)
                {
                    pos.z = loc.z - _width / 2;
                }
                else
                {
                    pos.x = loc.x - 8 - _width / 2;
                }
            }
            if (dis3 > 0)
            {
                pos.z = loc.z - 2 + _width / 2;
            }
            if (dis4 > 0)
            {
                pos.x = loc.x - 10 + _width / 2;
            }

            break;
        case zN:
            dis1 = loc.x - (pos.x - _width / 2);
            dis2 = pos.z + _width / 2 - (loc.z - 8);
            dis3 = pos.x + _width / 2 - (loc.x + 2);
            dis4 = loc.z - 10 - (pos.z - _width / 2);
            if (dis1 > 0 && dis2 > 0)
            {
```

```
                if (dis1 < dis2)
                {
                    pos.x = loc.x + _width / 2;
                }
                else
                {
                    pos.z = loc.z - 8 - _width / 2;
                }
            }

             if (dis3 > 0)
             {
                 pos.x = loc.x + 2 - _width / 2;
             }
             if (dis4 > 0)
             {
                 pos.z = loc.z - 10 + _width / 2;
             }

             break;
         }
     }
```

五、镜子

在游戏设计的计划中，为了增加游戏场景的丰富性和交互性，我们设计了镜子这一功能的实现。由于我们的场景较为单一，物品较为固定，我们没有采用光线追踪的方法实现，而是根据物理原理的分析，在摄像机对应镜子的位置设置一个假摄像机，通过这一假摄像机的视野，即可绘制出镜子应该反射的画面。

我们根据这一思路实现了镜子的 demo，但是在合并加入游戏的时候由于一些未知的 bug，镜子的效果没有呈现出来，花费很多时间也没有找到问题所在，很遗憾没有在游戏中见面。

Results

部分效果贴图如下，具体效果，尤其是音效配合等详见视频。

Roles in group

人员	内容
1950509 马家昱	走廊的自动生成与模型导入
1951590 董浩	光照，灯光闪烁，音效的配合
1951858 陶思月	光照，法线贴图，报告和视频
1953080 田宇	物理检测，上下楼梯的抖动，镜子
1956211 覃雯	模型的制作、修改

References

LeanrOpenGL CN