计图期末项目18组展示

design by lyh

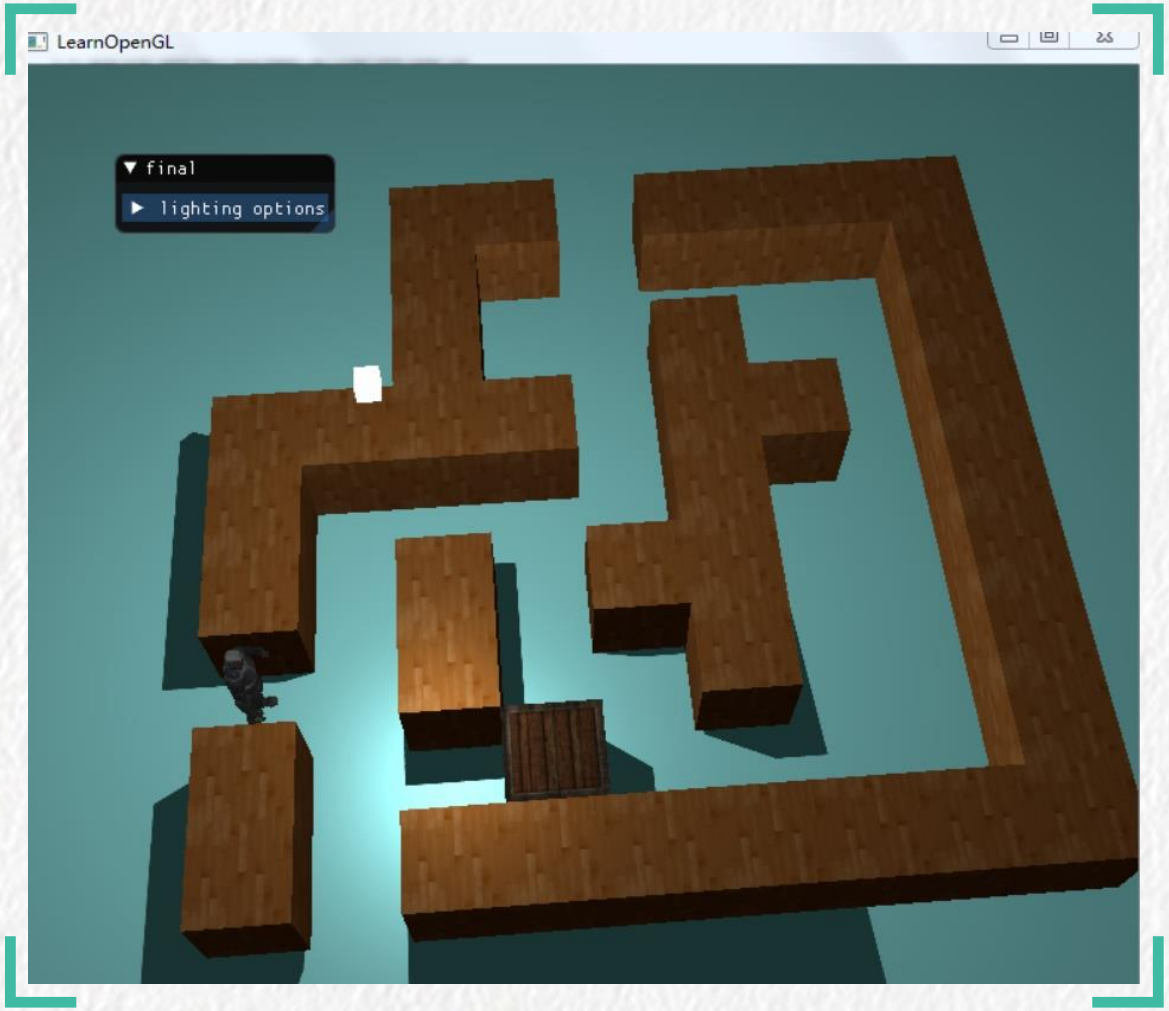
组长：王亮岛 16340219
组员：刘沅昊 15331220
组员：廖蕾　16340135
组员：王思诚 16340223

目 录

Contents

项目介绍及实现结果

项目介绍及实现结果



项目介绍

本项目实现的是一个模仿吃豆人的走迷宫的小游戏，现阶段只实现了迷宫的构建和可操作对象的移动和视角的移动。

操作方式

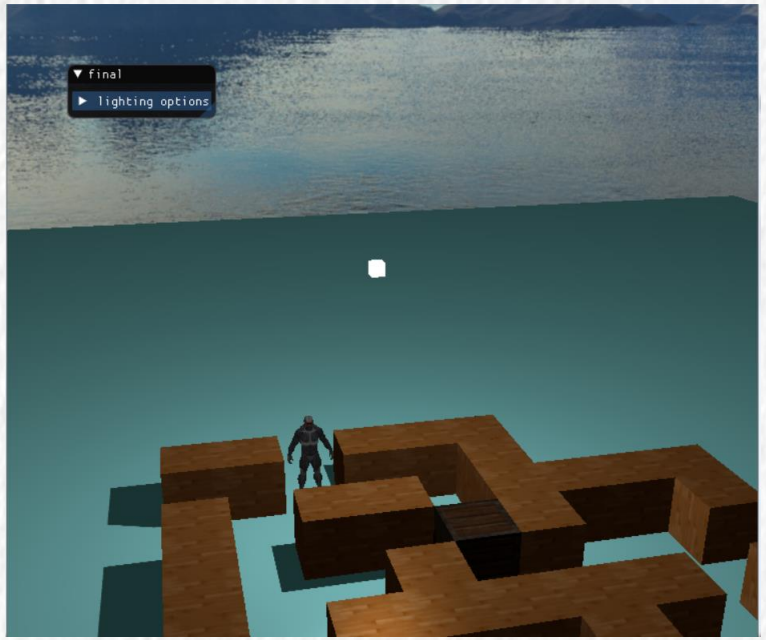
C键：用于控制是否移动视角，按下之后可以进行摄像机的移动。

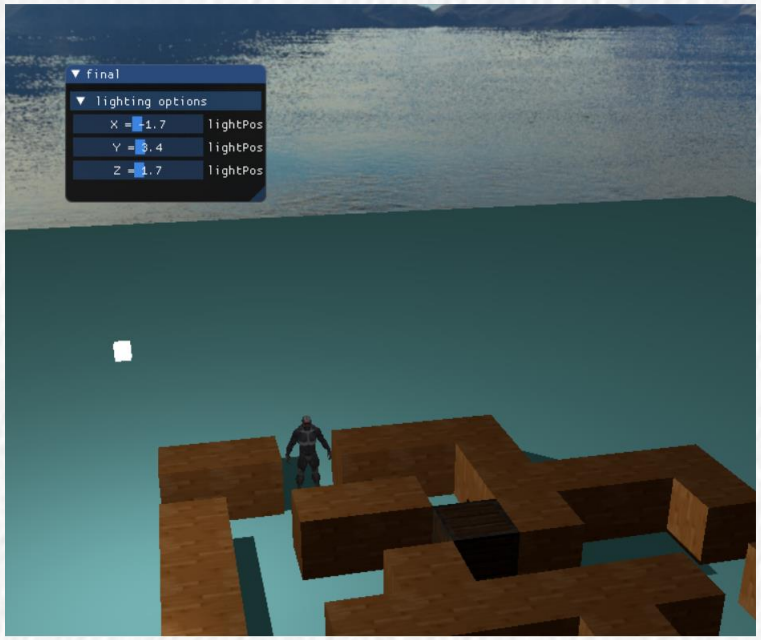
V键：用于锁定视角，按下之后就无法进行摄像机移动

WASD和鼠标：控制摄像机移动

JKLI：控制物体的移动

项目介绍及实现结果


final
lighting options

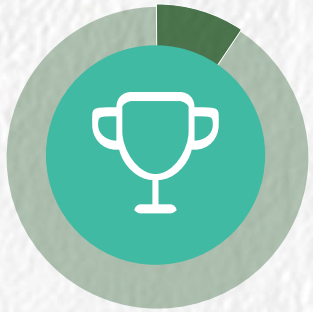

final
lighting options
X -1.7 lightPos
Y 3.4 lightPos
Z -1.7 lightPos


final
lighting options
X -1.7 lightPos
Y 3.4 lightPos
Z -1.7 lightPos

人员分配和工作

王亮岛

实现光照，阴影，场景的构建，物体的贴图，吃豆人的移动逻辑，碰撞检测，模型的加载



王思诚

实现天空盒

廖蕾

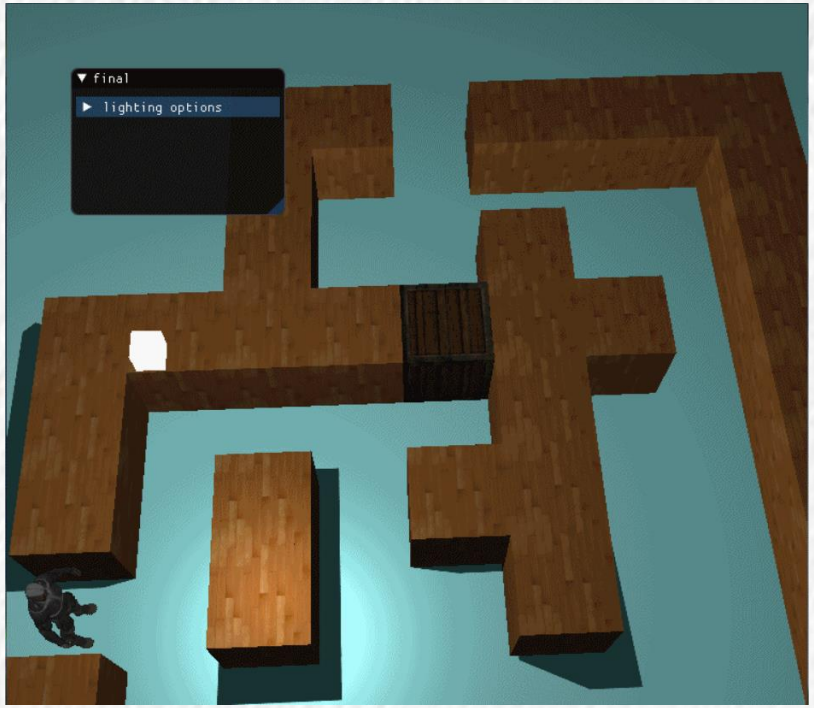
文本渲染的实现

刘沅昊

主要工作是另一个终止了的项目的基本功能实现

实现功能列表

实现功能列表



基本功能

1.相机移动
2.光照
3.阴影实现
4.物体贴图
5.模型载入

额外功能

1.物体碰撞检测
2.天空盒
3.文本渲染

基本功能

相机移动

1.camera中包含了摄像机的各种成员变量，可以通过成员函数改变相应的成员变量，当调用getViewMatrix时就返回一个最新的摄像机的view矩阵

2.通过回调函数监听键盘输入事件，每次监听到就改变摄像机的位置

3.通过一个全局变量deltaTime平衡上下帧之间的时间，进而平衡摄像机的移动速度

4.处理鼠标输入的时候，有以下步骤：

- 计算鼠标距上一帧的偏移量。
- 把偏移量添加到摄像机的俯仰角和偏航角中。
- 对偏航角和俯仰角进行最大和最小值的限制。
- 计算方向向量。

5.处理视野缩放函数

```
void Camera::processMouseScroll(float yoffset) {
    if (Zoom >= 1.0f && Zoom <= 45.0f)
        Zoom -= yoffset;
    if (Zoom <= 1.0f)
        Zoom = 1.0f;
    if (Zoom >= 45.0f)
        Zoom = 45.0f;
}
```

基本功能

光照

采用的是冯氏光照模型，由于地面是没有纹理的，而物体和迷宫的墙壁是有纹理的，因此我设置了一个参数可以选择纹理或者颜色。这样就可以实现光照贴图是时候既能展示没有纹理的地板，也可以展示有纹理的物体。

阴影

使用之前作业实现阴影的方法，直接就拿过来用了，简单来说，实现阴影有两个步骤：

1.首先，需要渲染深度贴图

2.使用生成的深度贴图计算片元是否在阴影中，以摄像机方向。

效果见左图

基本功能

纹理贴图

1.首先调用stb_image.h库中的函数加载纹理图片。

2.然后是生成纹理，这里我们使用一个函数将整个纹理生成的过程封装起来

3.应用纹理的时候需要在顶点数组中添加纹理坐标，然后在顶点着色器中调整顶点着色器使其能够接受顶点坐标为一个顶点属性，并把坐标传给片段着色器。

4.然后因为我们的纹理贴图结合了光照，因此在片段着色器中，通过纹理坐标，从纹理中采样片段的漫反射，镜面反射的颜色值，将得到的光贴图FragColor渲染到物体上。

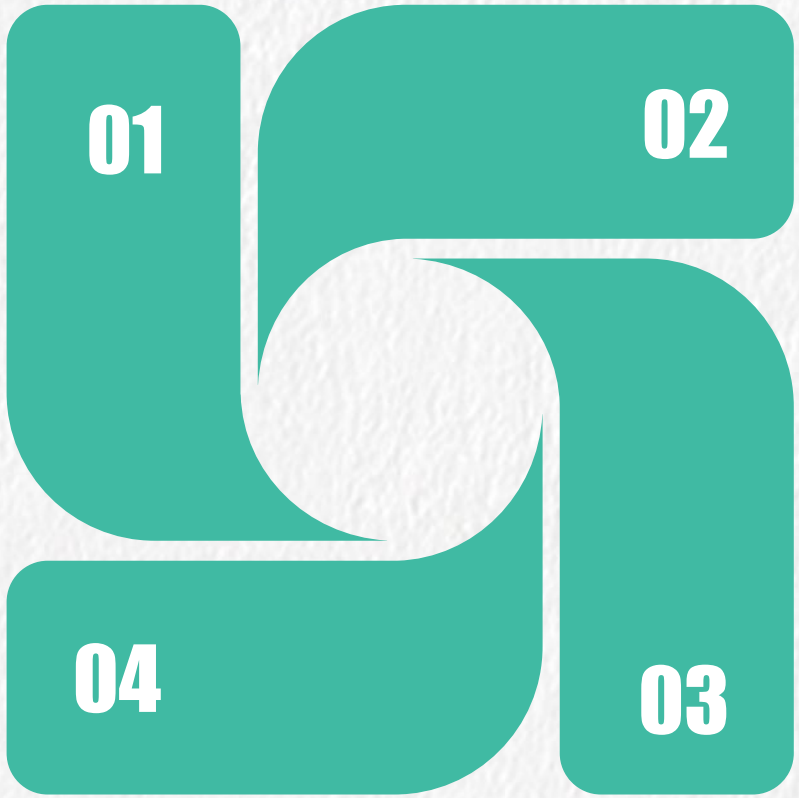
5.在调用glDrawElements之前绑定纹理了，它会自动把纹理赋值给片段着色器的采样器：

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, containerMap);
```

基本功能

模型载入

使用的是Assimp库，当使用Assimp导入一个模型的时候，它通常会将整个模型加载进一个场景(Scene)对象，它会包含导入的模型/场景中的所有数据。Assimp会将场景载入为一系列的节点(Node)，每个节点包含了场景对象中所储存数据的索引，每个节点都可以有任意数量的子节点。



因此，使用Assimp导入模型需要做的第一件事是将一个物体加载到Scene对象中，遍历节点，获取对应的Mesh对象（我们需要递归搜索每个节点的子节点），并处理每个Mesh对象来获取顶点数据、索引以及它的材质属性。最终的结果是一系列的网格数据，我们会将它们包含在一个Model对象中。

其中一个Mesh类需要的变量包括：一系列顶点，用于索引绘制的索引以及纹理形式的材质数据。然后是初始化，包括配置正确的缓冲，并通过顶点属性指针定义顶点着色器的布局。接着是定义Mesh函数进行渲染。

对于Model类，包括初始化的时候调用loadModel函数，这个函数会使用Assimp来加载模型至Assimp的一个叫做scene的数据结构中，然后是处理场景中所有节点，处理完之后遍历所有网格调用Draw函数，生成模型。

基本功能

模型载入

额外功能

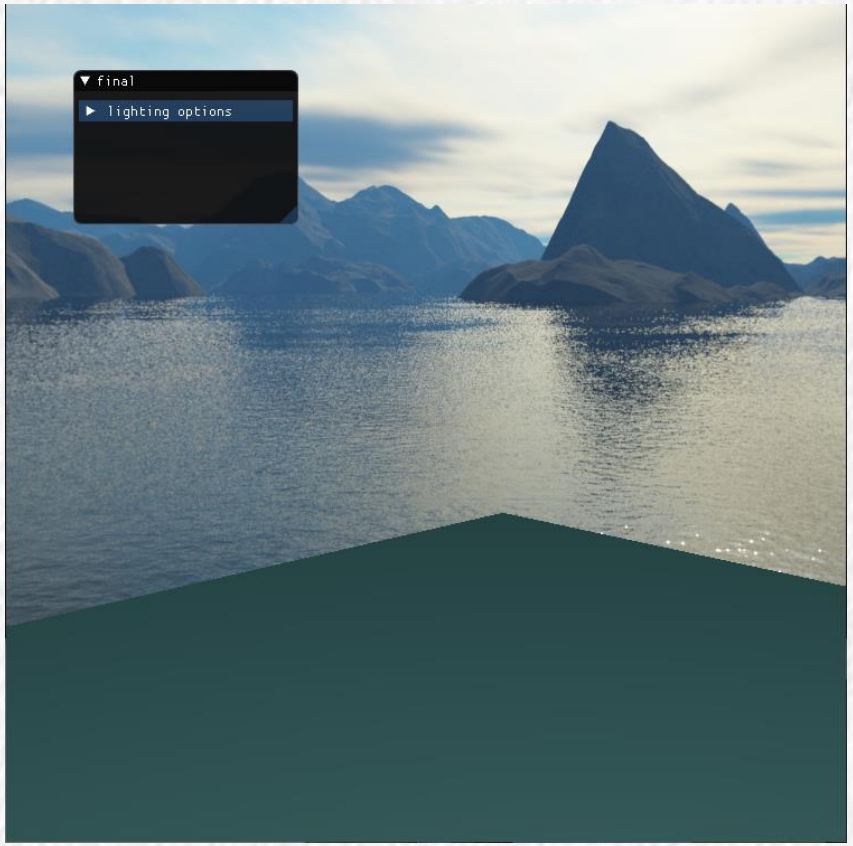
物体碰撞检测

这里使用的是最简单的碰撞检测，通过存储每一个组成墙壁的正方体的坐标，然后物体的大小设计成和正方体的大小一致，物体每次移动一个正方体的长度，也就是说，物体每一次移动一格，然后通过map判断这一格是否存在墙壁正方体，如果存在，将物体的位置回退一格。

```
rightTo -= 15.0f*deltaTime;
glm::vec3 position = glm::vec3(roundf(forwardTo), 0.0f, roundf(rightTo));
if (!canMove(position)) {
    rightTo += 15.0f*deltaTime;
}
//canMove
std::map<int, glm::vec3>::iterator iter;
for (iter = wall.begin(); iter != wall.end(); iter++) {
    if (iter->second == nextPosition) return false;
}
return true;
```

额外功能

天空盒



天空盒是一个包含了整个场景的（大）立方体，它包含周围环境的6个图像，让玩家以为他处在一个比实际大得多的环境当中。这里使用的贴图是OpenGL教程中的贴图。加载天空盒和加载一个立方体的贴图是类似的，不过这个贴图的六个面用不同的纹理贴图，因此我们会定义一个vector并把这六张图的地址传入进去，然后用一个函数来加载。

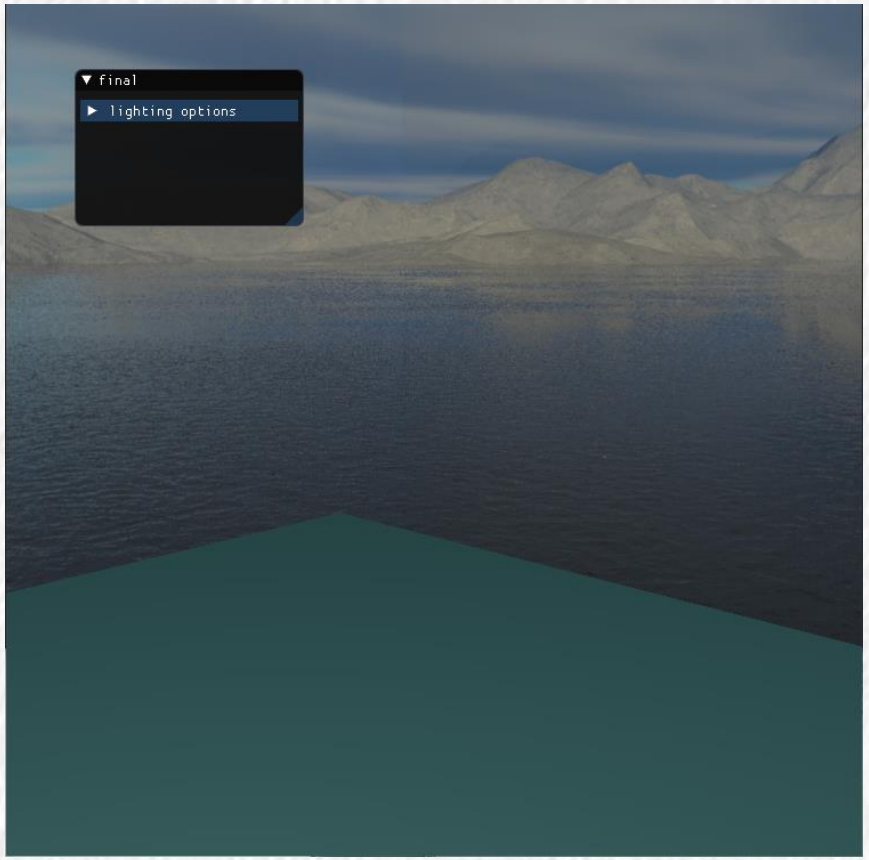
```
unsigned int loadCubemap(vector<std::string> faces)
{
    unsigned int textureID;
    glGenTextures(1, &textureID);
    glBindTexture(GL_TEXTURE_CUBE_MAP, textureID);

    int width, height, nrChannels;
    for (unsigned int i = 0; i < faces.size(); i++)
    {
        unsigned char *data = stbi_load(faces[i].c_str(), &width, &height, &nrChannels, 0);
        if (data)
        {
            glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE,
            stbi_image_free(data);
        }
        else
        {
            std::cout << "Cubemap texture failed to load at path: " << faces[i] << std::endl;
            stbi_image_free(data);
        }
    }
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);

    return textureID;
}
```

额外功能

天空盒



在加载天空盒的时候还需要注意一些问题，比如说天空盒是一个背景，在我们显示的物体中为最外层或者说最底层，因此我们要把它作为作为第一个渲染的物体并且禁用深度写入，这样子就会永远在其他物体背后。还有就是在我们移动物体的时候，需要营造场景非常大的现象，那么在以玩家的视角里，天空的相对位置应该不变，所以我们要保持天空盒贴图不移动，在这里我们需要将天空盒的观察矩阵由4×4取其左上角的3×3矩阵，移除掉变换的部分。也就是将观察矩阵先转为3×3矩阵，再转回4×4矩阵。 同时也做了一个轻微的优化，将天空盒的z分量设置为1.0，也就是最大的深度值，这样子天空盒就只会在没有其他可见物体的地方渲染，我们将其放置最后渲染，就能节省带宽。

vector的位置：

```
vector<std::string> faces
{
    "./resources/skybox/right.jpg",
    "./resources/skybox/left.jpg",
    "./resources/skybox/top.jpg",
    "./resources/skybox/bottom.jpg",
    "./resources/skybox/front.jpg",
    "./resources/skybox/back.jpg"
};
unsigned int cubemapTexture = loadCubemap(faces);
```

额外功能

文本渲染

这里是使用了freetype的库去渲染英文的一个字母的。FreeType所做的事就是加载TrueType字体并为每一个字形生成位图以及计算几个度量值(Metric)。我们可以提取出它生成的位图作为字形的纹理，并使用这些度量值定位字符的字形。

首先需要加载一种字体。由于在一个项目中可能没有权限去访问到Windows/Fonts，所以这里我是将arial.ttf这个字体放到了文件资源库中，代码如下：

```
FT_Library ft;
if(FT_Init_FreeType(&ft))
    std::cout << "ERROR::FREETYPE: Could not init FreeType Library" << std::endl;

// Load font as face
FT_Face face;
if (FT_New_Face(ft, ".\\resources\\arial.ttf", 0, &face))
    std::cout << "ERROR::FREETYPE: Failed to load font" << std::endl;
```

额外功能

文本渲染

加载字体之后，需要定义字体的大小，然后也需要从字体中获取ASCII字符集中的英文字母。所以我们需要先定义一个结构体Character去存储字符以及渲染一个字符需要的全部数据。

字体着色器部分大致和之前的没有区别，于是此处报告就不单独提出着色器代码部分。主要思想是：顶点着色器需要获取当前的投影矩阵坐标，计算文本渲染的位置；片段着色器需要有两个uniform变量，一个是单颜色通道的字形位图纹理，另一个是颜色uniform，调整最终颜色。这两个需要从主代码中传递值进去。

最后需要用textVBO和textVAO去渲染四边形，所以我们需要预先分配内存，在渲染的时候，每个2D四边形需要6个顶点，每个顶点是由4float向量组成。

未来的改进

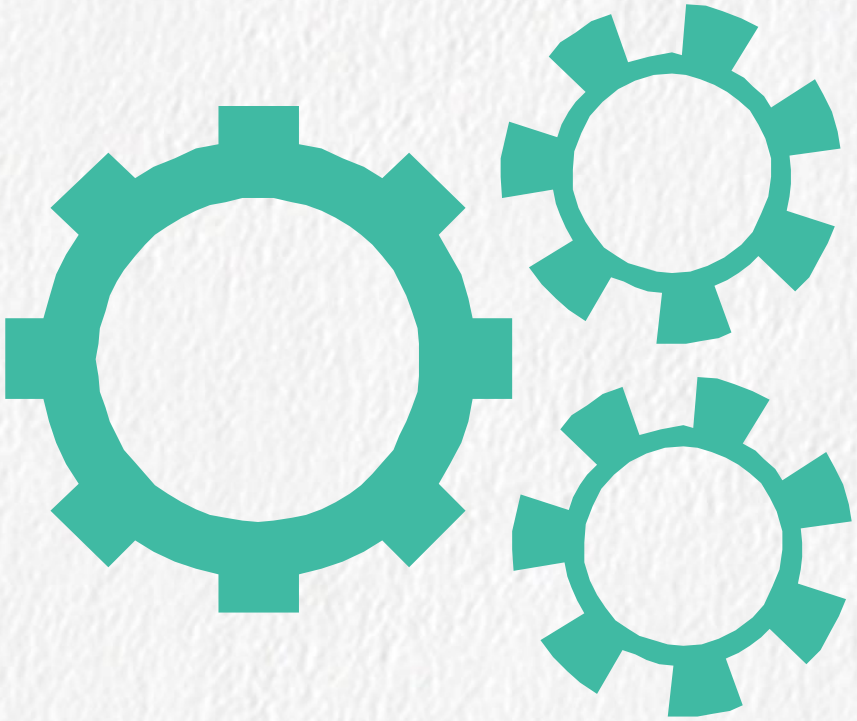
未来的改进

添加得分物体

人物吃到得分物体会有得分，然后能获取分数

模型移动

添加键盘控制模型移动的逻辑



粒子系统

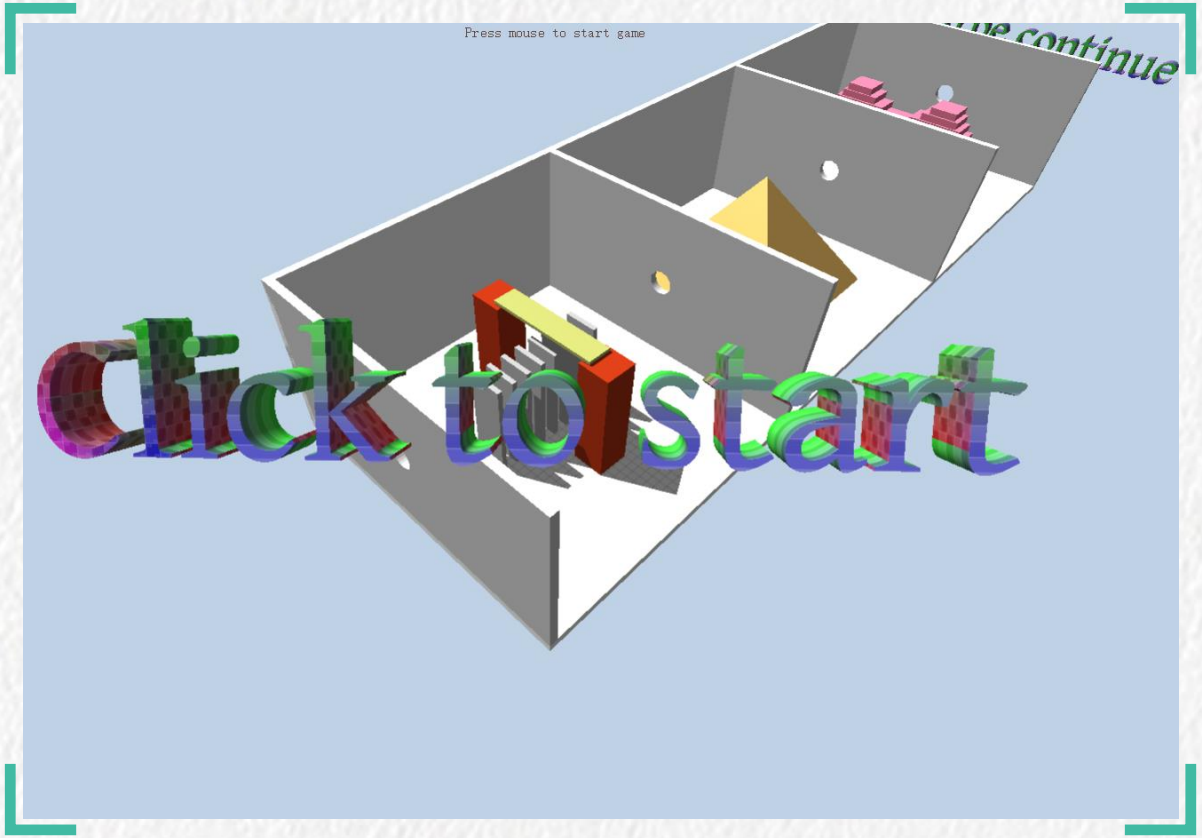
在吃到物体的时候会有爆炸效果

场景模型构造和导入

给整体游戏添加一个场景，将文字和其他部分融合的更加和谐。

另一个中止的项目

项目介绍及实现结果



项目介绍

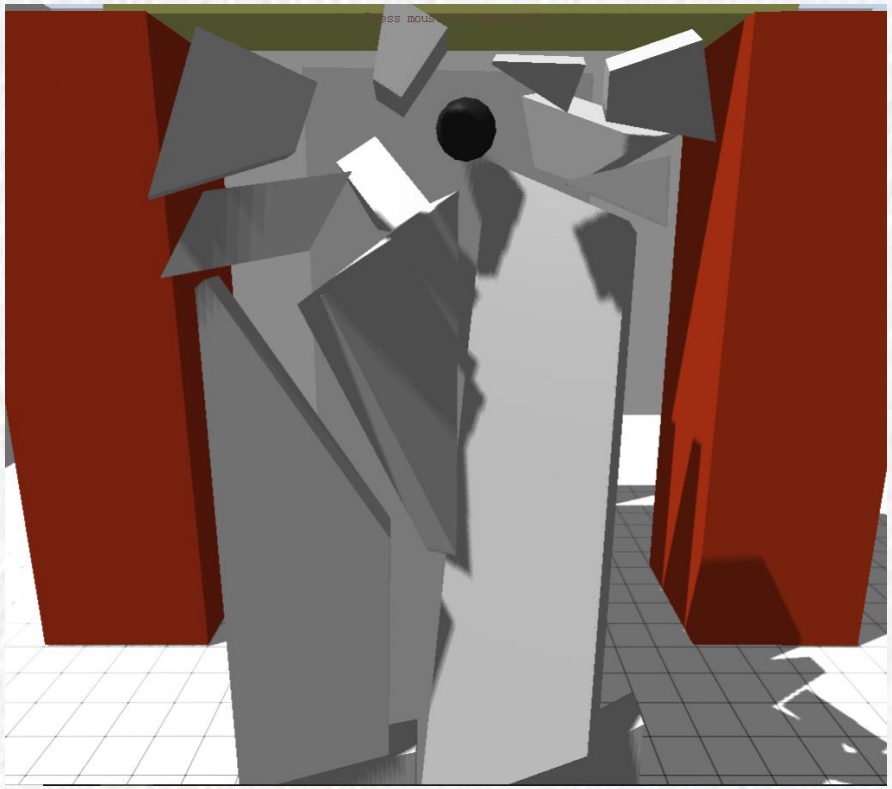
游戏场景是一个个的小房间，墙壁上有小洞，摄像机以一定速度，从一边进入，一边离开。房间中有很多物体，摄像头不能碰到物体，碰到就算失败。点击鼠标能从摄像机发射子弹击碎物体，以顺利通过出入口。

操作方式

鼠标点击或者手机点击屏幕发射小球击碎物体

https://sen.hotsarstore.com/raybo/hole/

实现功能列表



基本功能

1.相机移动
2.光照
3.阴影实现
4.纹理贴图
5.模型载入

额外功能

1.重力系统物体碰撞检测
2.3D字体

基本功能

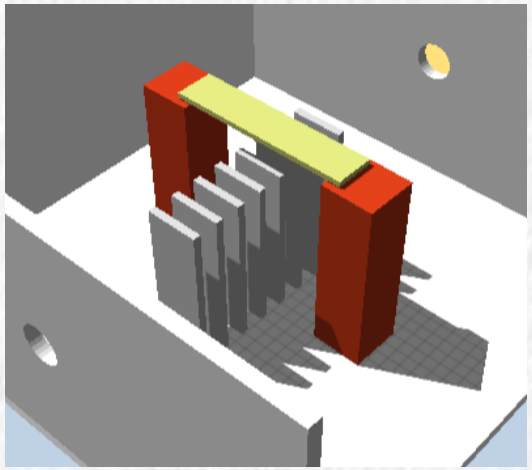
相机移动

- 游戏开始动画，照相机从俯瞰到洞口中心线
- 摄像机从入口移动到出口

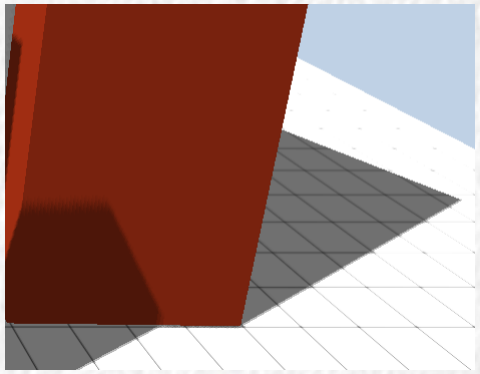
光照(Phong)

场景中有一个上方的光源，物体也能够反射光线。

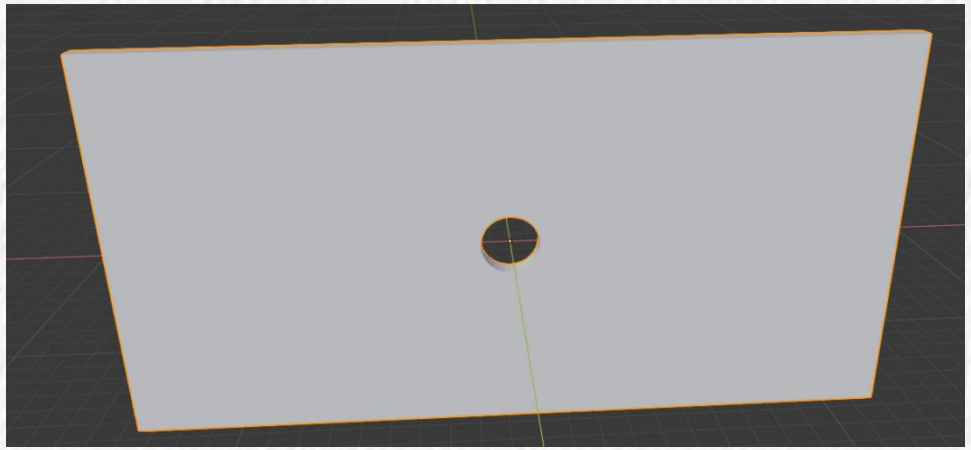
阴影

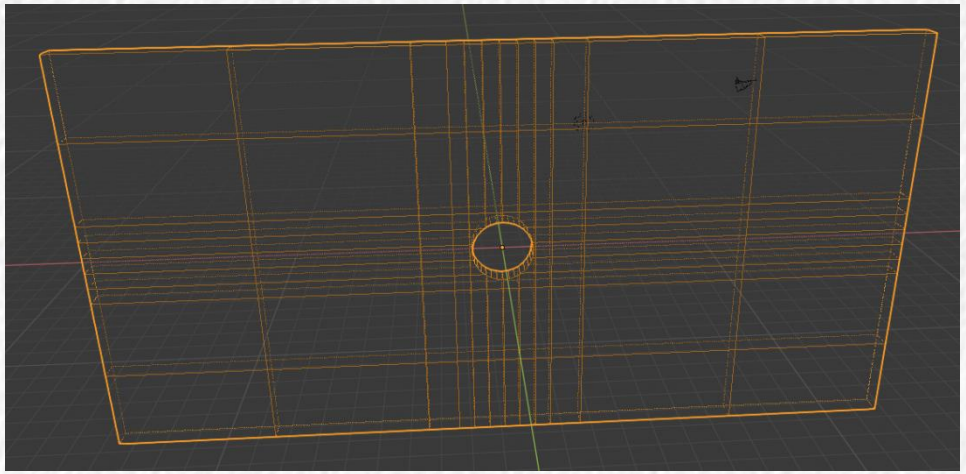


纹理贴图

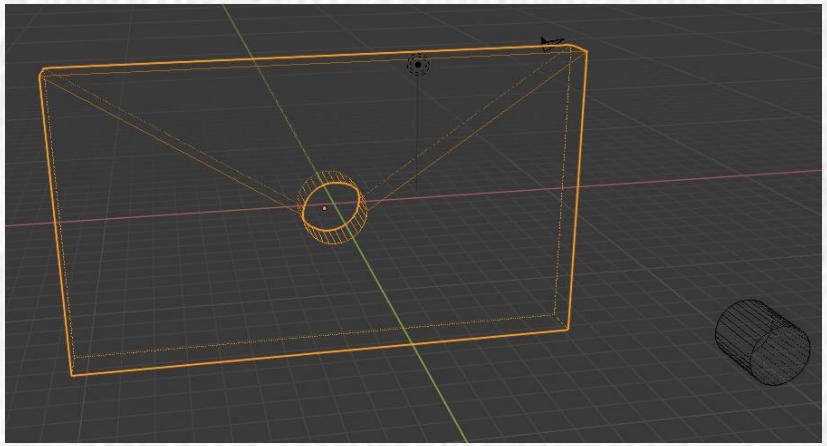


模型载入

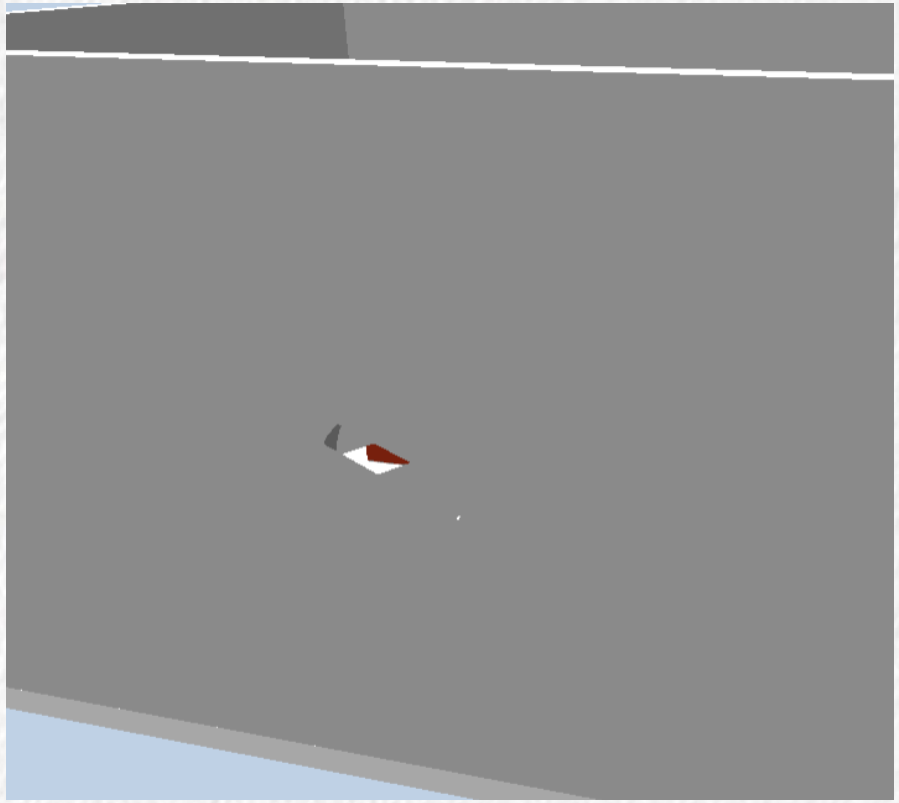


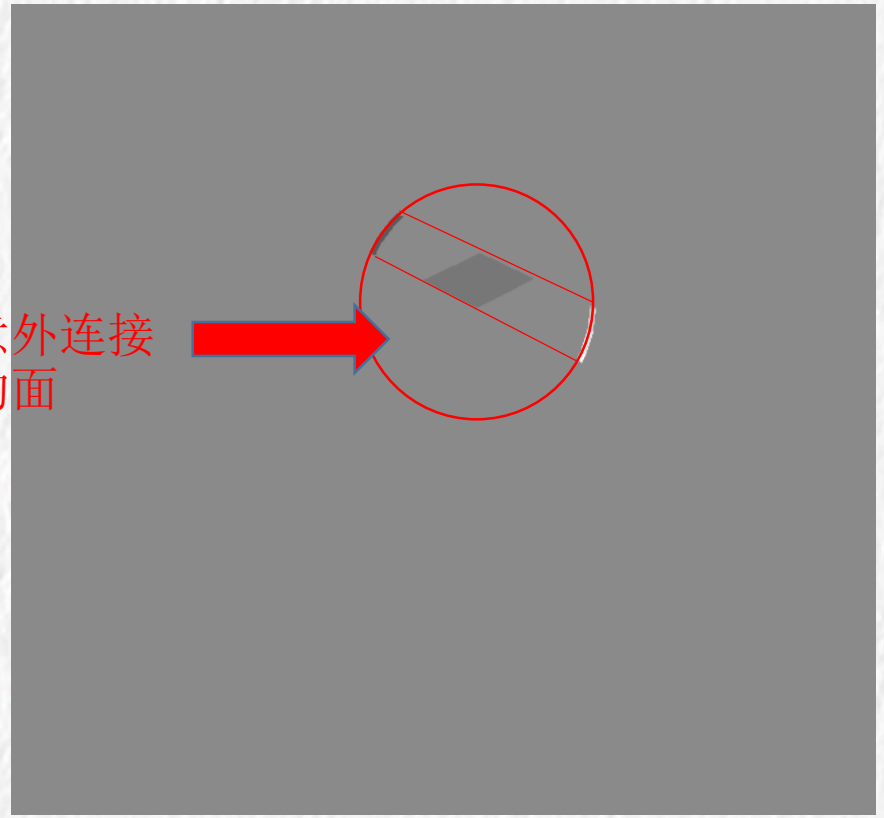


怎么在墙上挖洞这个比较麻烦，一般长方体可以给出长宽高就能创建，而椎体可以通过创建一个长方体，再将其中一个面的四个角push到一个点。我采用的方法就是通过模型的导入，先在blender创建一个长方体，再创建一个圆柱体，通过布尔运算的插值，在长方体上挖出一个洞。可是最开始这个洞也有问题，导入到webgl中洞口平面不完整，有些面连在一起了。后来找到问题是初始的长方体网格太少了，片元太大，后来在挖洞的地方细分了多次就解决了这个问题。总结出来在建模的时候尽量不要使用太大网格，在需要做一些特殊处理，比如这种挖洞，要多划分网格。

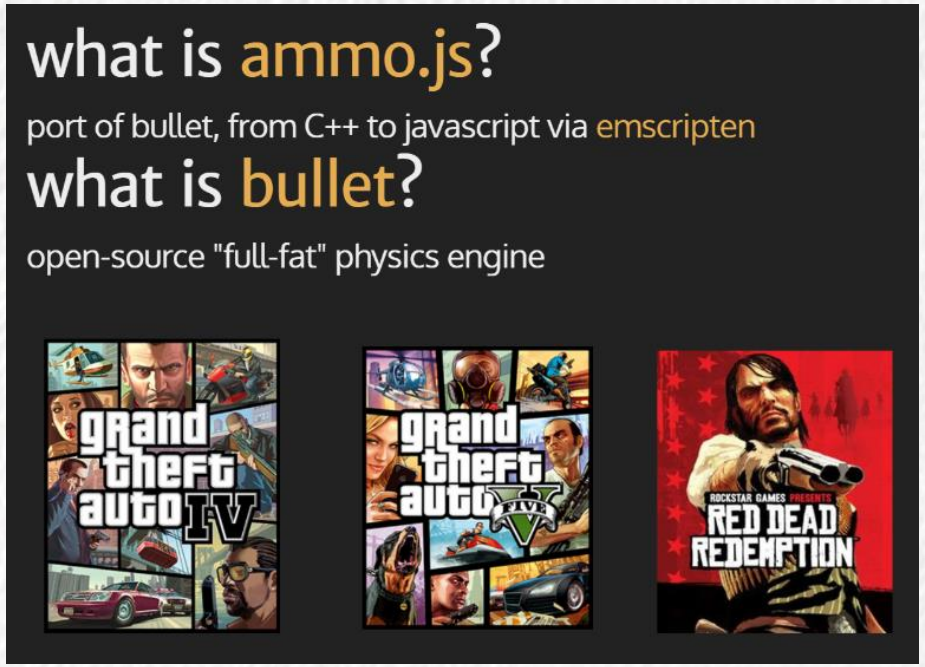


模型载入





重力系统和碰撞检测



Ammo.js

证明 Javascript 在性能方面走了很远的路的是Ammo.js，一个用 C++ 语言编写的物理库，它是 Bullet 项目的Emscripten 分支。

使用流程

1.three.js制作简单场景
2.添加物理场景
3.初始化物理空间函数initPhysics
4.物体场景更新函数
5.刚体生成函数

重力系统和碰撞检测

1.three.js制作简单场景

主函数一共两个，分别是init和animate，作用类似于unity3d中的Start()和Update()。

init中只有一个子函数initGraphics，其创建一个空场景，并添加一个环境光和一个线性光，线性光可以产生阴影。initGraphics也是为了和后文中的initPhysics区分开。

animate函数的主要作用是更新场景，绘制下一帧，更新相机位置。

```
// Graphics variables
var container, stats;
var camera, controls, scene, renderer;
var textureLoader;
var fontModel;

var clock = new THREE.Clock();
```

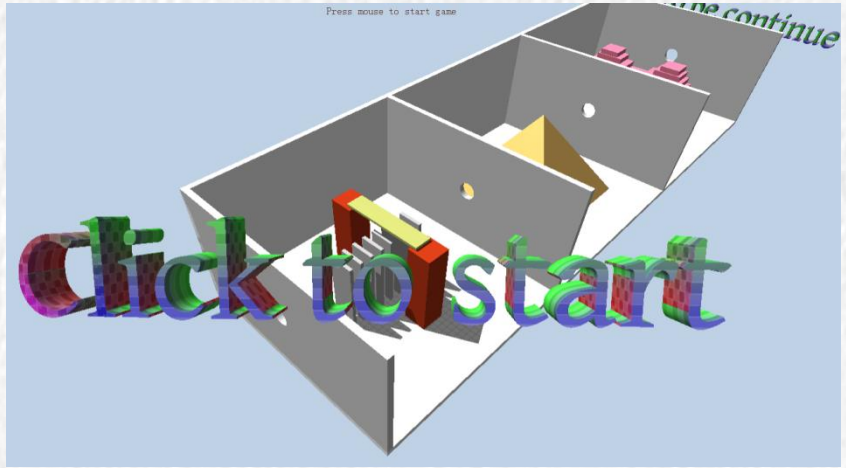
2.添加物理场景

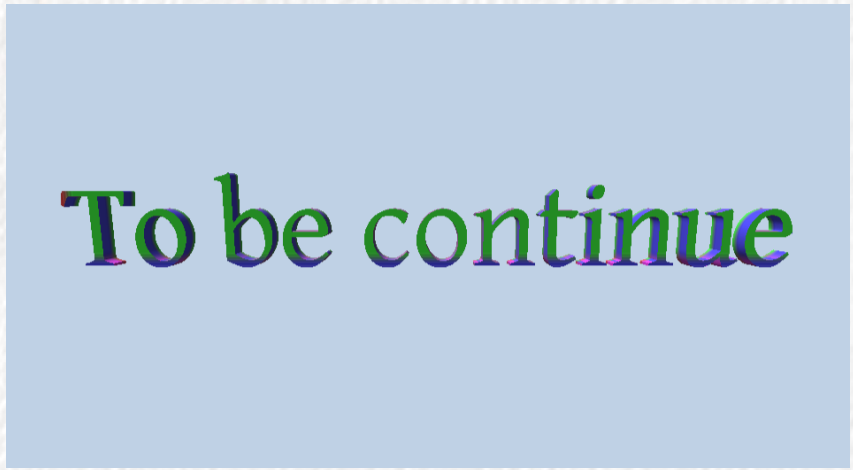
在上一个场景的基础上添加初始化物理场景函数initPhysics、物理场景更新函数updatePhysics、将物理场景中的物体和绘图空间中的场景关联起来并添加到两个场景中的函数createRigidBody(前几个物理场景只支持刚体)，向场景中添加平行六面体的函数createParallellepiped，以及初始化绘图空间和物理空间中物体的函数createObjects。

rigidBodies同时保存刚体在绘图空间和物理空间中的形状。其实就是把物体的物理形状保存在THREEObject的userData属性中。

```
// Physics variables
var gravityConstant = 7.8;
var collisionConfiguration;
var dispatcher;
var broadphase;
var solver;
var physicsWorld;
var margin = 0.05;
var convexBreaker = new THREE.ConvexObjectBreaker();
// Rigid bodies include all movable objects
var rigidBodies = [];
```

3D字体





使用流程

1.读取json格式的字体样式

2.设置字体属性，大小、位置、材质等

3.将字体网格添加到绘图场景中

```
103 function initFont() {
104     var font;
105     var loader = new THREE.FontLoader();
106     loader.load("fonts/gentilis_regular.typeface.json", function (res) {
107         font = new THREE.TextBufferGeometry("To be continue", {
108             font: res,
109             size: 6,
110             height: 2
111         });
112 
113         font.computeBoundingBox(); // 运行以后设置font的boundingBox属性对象，如果不运行无法获得。
114         //font.computeVertexNormals();
115 
116         var map = new THREE.TextureLoader().load("textures/UV_Grid_Sm.jpg");
117         var material = new THREE.MeshLambertMaterial({ map: map, side: THREE.DoubleSide });
118 
119         fontModel = new THREE.Mesh(font, material);
120 
121         //设置位置
122         fontModel.position.x = -25;
123         fontModel.position.y = 5;
124         fontModel.position.z = -150;
125 
126         scene.add(fontModel);
127     });
```

谢谢欣赏！

design by lyh

Time goes by so fast, people go in and out of your life. You must never miss the opportunity to tell these people how much they mean to you.