大作业报告

基本要求说明

1.具有基本体素（立方体、球、圆柱、圆锥、多面棱柱、多面棱台）的建模表达能力

天空球是一个半球形，其相关代码为“SkyDome.h/SkyDome.cpp”，实现了对固定R的半球形生成对应obj模型并输出为文件的功能。具体效果在ppt中有展示。

2.具有基本三维网格导入导出功能

“Material.h/Material.cpp”文件实现了导入mtl文件的功能，“Model.cpp/Model.h”文件实现了导入obj模型的功能，这两部分代码都有参考网络代码，前者参考了博客：https://blog.csdn.net/woaicide/article/details/50752511，后者参考了github代码：https://github.com/penguinsource/3D_Model_Viewer。不过它们都有挺多写错了的地方……最后相似的部分也并不多。

3.具有基本材质、纹理的显示和编辑能力

材质纹理的显示在上面部分完成，修改对应位置文件的贴图能够对材质进行编辑。

4.具有基本几何变换功能（旋转、平移、缩放等）

对炮塔能够进行旋转。

5.基本光照明模型要求，并实现基本的光源编辑（如调整光源的位置，光强等参数）

可以调节环境光和0号点光源，在ppt中有展示。

6.能对建模后场景进行漫游

实现了平移，缩放，旋转视角，在放置单位和操作炮台时也实现了自动旋转至合适视角。

操作说明

控制：“WASD”控制左右上下移动，”ZX“控制缩放，”OP“控制环境光照强度，”IJKL“控制点光源移动，鼠标拖动控制视角旋转。

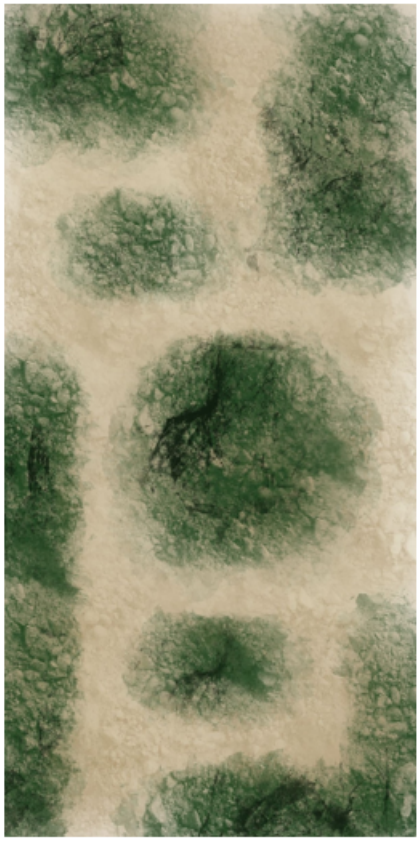
cost随时间增长，消灭敌人一名cost+=10。

花费20cost进行部署和炮台使用，部署方式是单击目标物体再单击地图上目标位置；转动炮台后将自动瞄准范围内最接近终点的敌人消灭。

值得记录的细节

地形生成

绘制灰度图，图上每个点的灰度值代表对应位置的y坐标，导出为obj文件，完成这部分工作的代码是”Landscape.cpp/Landscape.h“。



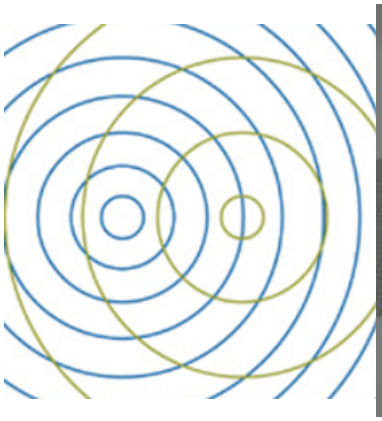
怪物的弹跳和碰撞

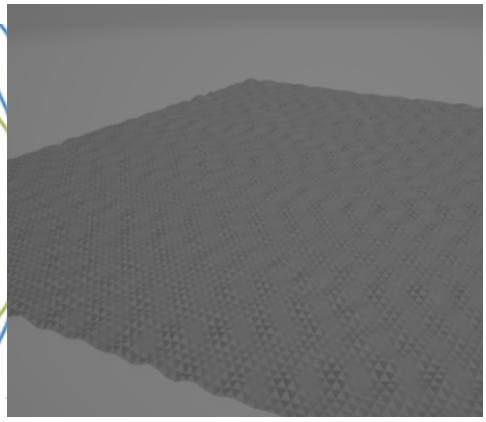
竖直方向做自由落体运动，通过方形区域内最高的地图y坐标和怪物y坐标的对比判断是否有碰撞，如果有碰撞，设置为地面不动的完全弹性碰撞（反转v的方向）。

水面

用正弦波模拟水面，设置了位置在中央地图下面的3x6=18个正弦波。参考了博客https://blog.csdn.net/qianxiao_1/article/details/78711199以及知乎回答https://zhuanlan.zhihu.com/p/23378778。

$$H(x,y,t)=\sum\left(A_i\times\sin\left(\mathbf{D}_i\cdot(x,y)\times w_i+t\times\varphi_i\right)\right),$$





死亡效果

计算每个三角形中心点和整个object中心点连成的方向，作为水平加速度的方向；竖直方向做自由落体运动；三角形顶点向中心点匀速运动，整个三角形随时间缩小。

ppt中demo过程解释

程序启动后，等待费用到达20，选择部署攻击单位，视角自动转到正上方；费用到达20后，选择控制炮台，视角自动转到正上方，转动炮台瞄准第一个敌人，视角自动转回，转回过程中游戏仍然暂停，但是炮弹会发射，造成极简版子弹时间（x）的效果，击中的敌人会立即死亡；费用到达20后，部署箱子改变敌人前进方向；费用到达20后，部署攻击单位结束战斗。

视频最后一段进行了上下左右前后的移动以及光源的调整和移动，证明完成了移动旋转和缩放功能，以及光照调节功能的操作。在光源位置移动放面，地图的反射系数设置得很小，并不明显，比较明显的是看箱子和柱子是否有被光照。

（并不重要的）代码解释

类解释

Assist

攻击单位的类，包含每一帧的update函数和判断是否和地图有穿模的check_y函数。

Cannon

炮台的类，包含每一帧自己的update函数和子弹的update函数，子弹走自由落体运动，包含一定的简单残影。

Enemy

敌人的类，包含每一帧的update函数（其中用DFS0判断最优路径），判断和地图是否穿模的check_y函数，和设置死亡参数的dead函数。

Landscape

加载地图y_map图片和输出obj模型。

myMap

用于储存地图信息，也提供得到地图某一区域的最大y坐标的功能，用于更方便地判断是否穿模。

Material

用于处理mtl文件，基本上就是加载纹理。

Metalbox

箱子的类，只有一个计算世界坐标的函数。

Model

加载obj模型的类。

SkyDome

加载天空球的类。

Tower

加载炮台下面的塔。

Water

水面的类，包括计算y坐标的函数cal_vertex，计算obj模型的函数cal_obj和save_obj，绘制水面的函数draw_water。

main函数解释

通过drawobject(Object* object,double deadtime)函数绘制有obj模型的元素，其他元素用draw_xxx()函数单独完成，如怪物血条（draw_blood），攻击单位的范围效果(draw_circle)等。

通过update_game函数每帧更新所有元素的状态，display函数根据每一个元素的状态决定是否绘制以及用什么方式绘制元素，这里我控制视角时没有真正去改相机位置，而是在绘制除了操作界面之外的部分前根据dx/dy等视角控制参数做坐标系变化来达到控制视角的目的。

通过xxx_recall函数进行鼠标和键盘的回调，改变游戏的状态（是否是在转视角状态，是否在控制状态，在这些状态下游戏时钟会暂停）。

通过init_xxx函数进行元素状态，obj模型以及整个绘制的初始化。

（更不重要的）感想心得

说实话就是只想拿基础分x

我选择solo这个大作业，最开始就是因为~~明日方舟是我玩的第一款游戏也让我改变了对游戏的想法所以~~想要做一个自己想做的东西，对我来说完整更重要。这样的话bonus就不想写，细节也不够好。这些的想法对队友很不负责，所以决定自己独立完成这个作业。

最终和我估计我能完成的水平确实差不多xd，最难受的是我自己的电脑速度真的慢，有些功能都写好了但是因为渲染速度限制，一旦加上去就卡成ppt（对就是说水面x），不得不放弃。中间也遇到了很多问题，绝大部分都是因为偷懒参考的网络代码里面有坑。。。最后一天晚上还在调光照的问题，最后发现是因为参考的纹理加载代码用的是纹理覆盖光照。如果最开始能自己去看使用文档写这部分代码，也许是不用花这么多时间去排查到底是哪里出问题了的。

已经完成了整一个低配3D明日方舟的想法，希望以后能去玩更多有趣的高配的东西:)