计算机图形学考试大作业报告

一、作业要求

①设计实现一个片头动画：要在一本金属材质的书封上出现《盗墓笔记》的 bump mapping 视效和动画（如图 1(a)的纹理），或者是一个带 bump mapping 视效的八卦陀螺飞入场景（如图 2 所示的模型和纹理），整个场景要有合适的背景，有灯光变换，然后这本书（或陀螺）缓慢消失在迷雾中。

②迷雾逐渐散去，出现一个暗室，键盘敲击空格键，会有一个手电朝前照射，看到一个石堆。此时，再按“B”键，一个炸弹向这个石堆飞去，碰撞产生爆炸，手电关闭。这里可复用作业 3 的模型和粒子系统，只是爆炸的碎片变成石块。

③石块散去，剩下一个 Buddha 样子的粗糙模型（Happy Buddha 的 LOD 三角网格模型将会给大家 ply 格式，如图 3），材质起初是粗糙的灰色石质，但当手电重新打开照亮模型，Buddha 模型开始变得越来越光滑精细，材质也逐渐转变为银色最后到金色。算法效率问题要解决。

二、作业具体工作及问题的解决方案

1、在书封上出现《盗墓笔记》的 bump mapping 视效

观察给定的图片发现四个字是凸起的，即字体笔画内部的像素高度应高于两边。因此想法是做一张灰度图，存储书封图片的灰度信息，字体笔画中间的部分灰度值较高，两边较低，灰度值代表了 bump mapping 的高度值。将原给出的书封图片做了一些处理，变成以下灰度图：

将该图读入 opengl，其灰度数据存储在一个二维数组中。

接着要对图片进行 bump mapping。对灰度图上的每个点进行梯度计算：

$$x\_gradient = pixel(x-1, y) - pixel(x+1, y)$$

$$y\_gradient = pixel(x, y-1) - pixel(x, y+1)$$

计算出了梯度就可以计算出当前该点的法线方向：

normal[i][j] = normal[i][j] - (position[i][j] - position[i][j - 1]).Unitization() * (height[i][j] - height[i][j - 1])

其中等式右边的 normal[i][j]是扰动前的法线方向， (position[i][j] - position[i][j - 1]).Unitization() * (height[i][j] - height[i][j - 1])为法向量的扰动量。

计算出了整幅图片的法向量之后，使用画三角的方式将每一个法向量安排在一个三角片上，最终画出整幅凹凸图。

2、场景的雾效及物体的隐没和出现，场景背景的设置

迷雾的效果是使用了 GL_FOG 实现的，先是将雾的颜色设置为了灰色，为了营造出缓慢消失的效果，在 Idle 函数中控制了雾的浓度（density 变量）随时间的变化。使用了全屏雾效，将背景和书一起隐于雾中。

过程中的问题：书的消失和石堆的出现之间有一个突变的过程，如果雾的浓度不足够的话会出现奇怪的跳变，因此将雾出现的时长以及浓度加深，这样即可以将二者的切换在雾最浓时实现。

场景的背景使用了一张盗墓笔记的大门图片，由于需要将背景和书一起隐没于雾中，

因此是将背景图片映射在了书后方的一个大型的矩形上，二者的 z 方向坐标被雾效的 GL_FOG_START 以及 GL_FOG_END 包围住，这样二者就可以同时消失在雾中。

3、石堆、手电和炸弹的实现

由于石堆是由一块块的石头组成的一座小山样的物体，因此我使用了 OpenGL 的正二十面体进行绘制，将 20 个正二十面体堆放成小山的形状，如图所示：

上图为加了手电之后的石堆效果，可以看到照亮石堆的是一个圆形的光圈。

手电是使用了聚光源，即将一个位置光源设置了方向和聚拢角度之后，得到一个类似于手电的光源。本作业中的手电是将聚光源的位置设置在了观察的方向，即 z 轴正方向，光的传播方向是设置为了 z 轴负方向加 y 轴正方向的一个微小的仰角。

另外，为了体现出手电的聚光效果，使用 GL_SPOT_CUTOFF 将光源的聚拢角根据视点和石堆的距离关系进行多次调整，最后获得一个比较好的类似手电的效果。

炸弹的实现由于在 assignment3 中已经建立好了模型，因此可直接拿来使用。这里对于炸弹向石堆飞去的动画是这样做的：

设置好炸弹的初始位置之后，计算出炸弹与石堆之间的距离，然后在 idle 函数中不断地将距离进行缩小；另外炸弹的飞行过程是一个类似于抛物线的过程，因此在炸弹沿 z 轴负方向飞行的过程中还有一个 y 轴先正方向后负方向的一个运动，同样也是在 idle 函数中对 y 方向的位置变量进行先正后负的变化。

炸弹的飞行过程中还要实现引信的不断变短以及火花效果，同样也是使用了 assignment3 的效果进行实现，将炸弹相关的所有模型（包括炸弹本身、引信、火花）

全部放在一个 glPushMatrix()和 glPopMatrix()中进行移动，这样就不需要针对炸弹的每一个部分都进行一次移动计算，而只需要将炸弹、引信、火花移动视作一个整体进行移动即可。

4、爆炸效果的实现

当炸弹飞到石堆之后，石堆爆炸。这里的石堆爆炸的粒子系统的效果也是复用 assignment3 已实现的粒子效果，将原本的纸屑替换成了这里的碎石块。碎石块的实现也是使用了 opengl 自带的正二十面体作为粒子进行发射，为了体现出炸弹爆炸时碎石块的随机性，也是设置了 8 个石块碎片的角度{0，45，90，135，180，225，270，305，360}进行随机发射。

爆炸效果图

5、ply 格式模型的读取和显示

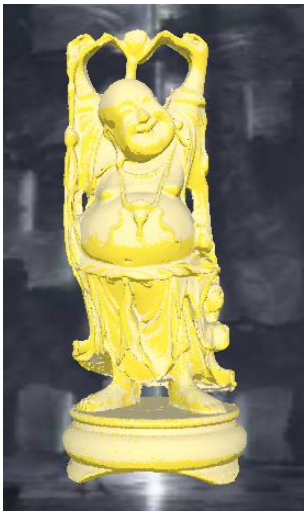
根据给定的 ply 格式的模型，使用 windows10 自带的 3d 查看器可以查看到原本的模型的样子：

使用文本编辑器打开的 ply 文件之后，内部显示的是如图所示的文件格式：

```
ply
format ascii 1.0
comment generated by ply_writer
element vertex 7108
property float x
property float y
property float z
element face 15536
property list uchar int vertex_indices
end_header
-0.00725025 0.0651677 -0.047265
-0.00435303 0.0650287 -0.047238
-0.00138566 0.0651022 -0.0470334
-0.00740545 0.0683118 -0.0467399
-0.00413745 0.0682059 -0.0467928
-0.00729968 0.0618636 -0.0464262
```

上网查阅了一些博客，了解到以下信息：开头的前 3 行是一些对于模型本身并没有用的信息，第四行是表示顶点数量，第五行到第七行是指顶点的坐标及其类型，第八行是指面元数量，第九行是面元的格式，list 表示的是列表，uchar 表示的是第一个数 3（即该面元由三点组成），int 表示面元中点的索引值，vertex_indices 表示的是属性名。End_header 下面的就是点的列表和面的列表，依次读取进来即可。

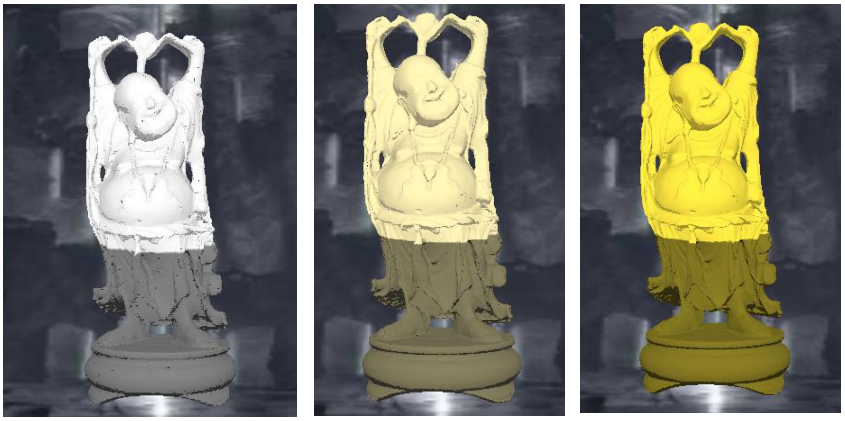
最后渲染在程序中的一部分中间结果(此中间结果是第一种效率较低的方法渲染的结果，关于模型的渲染效率在下面分析）：



由于想要做出一种模型不断由粗糙到光滑的感觉，因此原本是将 4 个模型同时画在了画面的同一位置，随着时间的变化不断地去掉粗糙的模型，但是发现这样做的效率非常非常低，原因后来分析应该是因为由于最后要呈现的是最精细的那个模型，因此最精细的模型按照这种方法的话是存在时间最久的，也即每一帧画面都需要渲染一遍这个最细的模型，效率自然非常低。因此最后选择按照时间的顺序先画最粗的模型，再画中间

的模型，最后画最细的模型，这样以来，前两个模型渲染的时间就少了很多，画面也流畅了不少。

最后的渲染结果按照时间顺序如图：



（图为手电只照射了佛像的上半身的效果，从左至右粗糙->光滑，颜色银->金）

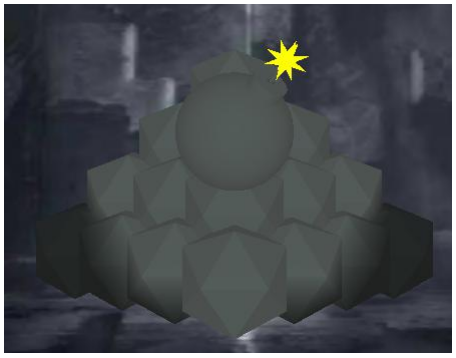
三、程序的操作方法

1、开头动画时：无需任何操作

2、中间出现石堆时：

当雾散后出现石堆时，按下键盘的空格键即可点亮手电筒，打开手电的效果即为上图所示。

此时，按下键盘“B”键后，会有一个点燃引信的炸弹向石堆飞去，如图所示：



3、炸弹爆炸后，会出现一个灰色的粗糙的佛像。此时再次按键盘“B”键，即可看到如前所述的由银色逐渐变为金色、粗糙逐渐变为精细的 buddha 模型。