《平时作业2-多边形细分及粒子系统》作业报告

徐培公522031910399

## 叙述为实现作业内容进行的具体工作，描述解决作业问题所采用的技术方案

### 在作业1的基础上，导入一个桌子模型放置在黑板前面（10分）

下载安装assimp库并编译，下载stb库，将上述两个库添加进项目中。将“上海交通大学软件学院本科计算机图形学课程代码仓库”中的models文件夹添加进项目，并将仓库里的objLoader项目中的christmas\_tree.fs，christmas\_tree.vs，model.h，mesh.h添加进项目。使用model.h中的类Model将models/obj/table/table.obj模型文件导入，并在循环渲染时使用glm::translate，glm::scale和glm::rotate分别进行平移、缩放和旋转，将桌子模型放置在黑板前面。

### 计算生成一个圆形平台初始网格，采用三角网格细分和随机偏移及平滑滤波，生成有起伏的地形模型，将该地形表面进行扫描拉伸成有一定高度的圆台，水平放置在桌面上，导入圣诞树模型，放置在地形上（30分）

首先生成一个x-z平面上投影为正方形的点阵。确定正方形每行点数n，使用std::vector<float>记录每个点的坐标，其中第i行第j列的点的(x, y, z)坐标在vector中的索引为((i \* n + j) \* 3 + 0, (i \* n + j) \* 3 + 1, (i \* n + j) \* 3 + 2)。初始化该点阵时，x和z坐标范围均为从-0.5到0.5以1/n递增，y坐标采用glm::perlin使用柏林噪声生成。

使用std::vector<unsigned int>记录渲染该地形面时的索引顺序。顺序遍历点阵，对于每个点与其附近的三个点形成的小正方形，构造两个三角形的索引，添加进该vector中。

初始化该点阵后，将x-z平面构成的正方形映射成圆形。对于在x-z平面范围为[-1, 1]的正方形，有如下公式映射到圆心为(0, 1)半径为1的1圆形：

对于vector中的每个点的x和z坐标采用上述公式进行映射，便得到一个x-z平面上投影为圆形的点阵。

随后对该点阵进行平滑滤波。采用均值滤波的方法，遍历每个点。对于每个点，将其与其周围总计九个点的y值做平均后作为该点的新y值。

接下来进行三角网络细分。将“上海交通大学软件学院本科计算机图形学课程代码仓库”中的tessellation项目中的test.vert，test.frag，test.tesc，test.tese，test.gs添加进项目中。该五个文件分别为顶点着色器、片段着色器、曲面细分控制着色器、曲面细分评估着色器和几何着色器，实现了三角网络细分效果。

至此地形平面已经生成完。对于圆台侧面的生成，使用一个std::vector<float>记录每个点。顺序遍历上述地形点阵的x=0或z=0的点，即地形边缘点，对于每个这样的点，将该点的(x, y, z)坐标值与(x, 0, z)构造两个点放入圆台侧面点的vector中，每加入一个点后，将该微分面的法向量(x, 0, z)也一并加入该圆台侧面点的vector中，以便渲染光照。该vector生成完毕后，使用std::vector<unsigned int>记录渲染该圆台侧面的索引顺序。顺序遍历点阵，对于每个点与其附近的三个点形成的小正方形，构造两个三角形的索引，添加进该vector中。圆台侧面也可以进行渲染了。

使用Model类将models/obj/christmas\_tree/christmas\_tree.obj模型文件导入，并在循环渲染时使用glm::translate，glm::scale和glm::rotate分别进行平移、缩放和旋转，将圣诞树模型放在地形面上面。

### 实现快捷键E点击后地台上方开始下雪，雪花落到与地面或圣诞树碰撞后消失（20分），雪粒子运动范围可在圆柱体内，或在球体内向下飘落，也可参考圣诞水晶球内的雪花运动模式。

定义一个结构体记录每个雪粒子的位置信息、速度信息和颜色信息，使用std::vector记录全部的雪粒子。初始化的时候，在合理的区间随机生成雪粒子的位置、速度，并随机生成白色至浅蓝色区间的雪粒子颜色。其中位置生成为了使其在圆柱体内，依然可以采用上述正方形到圆形的映射公式使其的x轴和z轴坐标位于圆形内。

编写雪粒子的顶点着色器与片段着色器。其中在顶点着色器中，计算FragPos = vec3(model \* vec4(aPos, 1.0))，viewPos = view \* vec4(FragPos, 1.0)，并将gl\_PointSize设置为-viewPos.z的反比例，以便控制雪粒子的近大远小的显示尺寸。

对于雪花落到地面或圣诞树碰撞后消失，采用碰撞检测。其中圣诞树采用近似的圆锥形包围盒，不断调整圆锥的顶点和底面圆心的位置以及半径的大小，使圆锥与圣诞树尽可能拟合在一起。确定圆锥的位置后，在每一帧更新雪花位置的时候，检测雪花的位置，如果位于圣诞树的包围盒中或者位于地面以下，则按照上述方式重新生成该雪花的位置、速度与颜色。

### 实现快捷键L点击后，圣诞树上有彩灯亮起，并且有光粒子特效出现，特效模式个人设计，不做特定要求（20分）

查找models/obj/christmas\_tree/christmas\_tree.mtl文件，获取彩灯信息。观察到“newmtl gold”和“newmtl red-ball”，即为彩灯。在圣诞树的片段着色器中，加入判断，如果Kd的颜色rgb值与“newmtl gold”或“newmtl red-ball”相同，则说明是彩灯，就把输出的颜色调亮即可。

定义一个结构体记录每个光粒子的位置信息、速度信息、颜色信息和闪烁时间偏移值（用于光粒子闪烁效果），用std::vector记录全部的光粒子。初始化的时候，在合理的区间随机生成光粒子的位置、速度、闪烁时间，并随机生成白色至浅黄色区间的光粒子颜色。其中位置生成为了使其在圆柱体内，依然可以采用上述正方形到圆形的映射公式使其的x轴和z轴坐标位于圆形内。光粒子在运动时，如果位置超出了圆柱体范围，则将其速度方向调位反向，保证光粒子始终位于圆柱体内。

为了使光粒子呈现径向半透明模糊的效果，首先生成一张径向半透明模糊的白色圆形图textures/glow.png，并在主程序中读取该图片作为光粒子的纹理。光粒子的片段着色器中，将纹理的每一像素的透明度alpha值与该纹理对应处相乘，得到白色至淡黄色且呈现径向半透明模糊的光粒子效果。同时主程序在此之前开启glEnable(GL\_BLEND)和glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)以允许实现透明颜色的渲染。

为了使光粒子呈现闪烁的效果，在片段着色器中，输入当前时间加上闪烁时间偏移值并对此去正弦值，得到的结果乘以一个系数作为闪烁颜色的波动值，并加上一个基础值作为初始透明度。这样可以使得透明度随时间变换而周期变化，呈现闪烁效果；闪烁时间偏移值则使不同的光粒子开始闪烁的时间不相同。