CG大作业报告

XXX 51902191XXXX

1. 总体完成任务
   1. 实现了交互界面，通过绘制贝塞尔曲线生成由三角面片构成的模型。可以在画面内渲染多个模型，同一时间只可以选中一个模型并且编辑该模型，可以对于模型的位置进行修改。
   2. 玩家可以选择并修改系统预设的贴图，可以更改环境光照的数值并且可以调整环境平行光的方向。可以通过手动输入模型的文件名称的方式保存或加载模型的数据到已创建模型中，进行实时的渲染。
   3. 玩家可以通过相机动画在天空盒内进行多角度观察，并且可以修改天空盒贴突围系统预设贴图。
   4. 玩家在进入游戏场景后会自动渲染桃花飞落的粒子效果。
2. 操作说明
   1. 用户可以通过WASD和鼠标调整相机的位置及视角，可以在场景自由的运动。
   2. 用户可以通过UI界面修改光照参数和模型参数，如模型大小、坐标等、修改材质贴图为预设的其他贴图，生成并绘制新的模型。
   3. 用户可以通过自己手动输入文件地址来储存或加载模型。
   4. 用户可以实时查看帧率。
   5. 在点击Draw按钮后，用户可以通过enter键查看当前贝塞尔曲线渲染的结果，可以通过E键位
   6. 用户在绘制模型时，应该注意绘制的模型轮廓线就是实际的轮廓线，如果是单层的实际上模型也只有一层。
3. 实现三角面片渲染模型
   1. 绘制贝塞尔曲线：

在点击绘制按钮后，玩家可以通过在GUI里显示的线的左侧对于贝塞尔曲线进行绘制。本游戏采用一条n阶贝塞尔曲线作为轮廓线，在绘制模式后左键为增加点到对应位置，右键为删除上一个点。鼠标操作通过鼠标回调函数获取，然后调用贝塞尔曲线内部函数将其存入。

游戏内部存储贝塞尔曲线对应的点的具体坐标以及连接关系，每条贝塞尔曲线对应一个lineVAO和一个bezierlineVAO，前者用于保存辅助直线的数值，并且采用EBO进行渲染；后者在得到前者最新值（用户每次按下enter键）后运算得出的500个点进行渲染。

* 1. 获得三角面片参数：

通过贝塞尔曲线的x，y坐标求出目标图形的半径r和对应高度h，并且对于数据进行标准化计算，后传入cylinder类里，通过一圈圈三角面片进行渲染。每一圈相同半径取num\_sample个点，通过三角函数计算这些点的x，y，z值存入对应的vector中。

同时，三角面片的连接关系是： 在两行圆圈内，上下相同角度的点构成一个四边形，从右上点连接左下点，构成两个三角形，即为我们需要的三角面片。比如一共取50个点，构成0、1、50和1、51、50两个三角面片。

讲三角面片顶点坐标和连接方式依次存入VBO和EBO传入shader进行模型的渲染，通过DrawElement绘制三角形作为三角面片。

* 1. 三角面片贴图：

贴图的左下角是（0，0），右上角是（1，1），所以只需要求出对应的顶点的对应的texture坐标即可判断。所以texture坐标可以直接计算对应角度的弧度，然后进行标准化进行计算。得出数据后放入VBO，在绑定texture后传入shader即可正确渲染。

三角面片贴图在创建cylinder类的时候已经全部读取绑定好并且将对应的int值存入vector中，在更新的时候只需要更新对应的textureID即可。

* 1. 三角面片光照：

三角面片的光照需要依靠法向量。计算法向量的方法是通过其切线方向和旋转方向的向量作差乘，即可得到法向量。切线方向的向量是上方点指向下方点的向量，旋转方向的向量是通过相似三角形，可以得知旋转方向的向量应该为（z，0，-x）。则差乘切线方向和旋转方向的向量，然后normal向量为1，将其作为法向量放入VBO，传入shader进行渲染。

* 1. Shader的配置修改：

光照采用了冯式光照模型。

Shader参考了LearnOpenGL漫反射贴图的写法，由于贴图的原图较多都只有图案，而少有材质、阴影等附加贴图，所以仅采用了漫反射贴图为唯一的输入的sample2D进行渲染。同时，由于要模拟阳光，所以采用了平行光对于光照进行了计算。

同时，Shader也引用了坐标系统的转换矩阵。在3D模型里使用具有camera的z值的projection，在2d模型使用固定的原生projection，便于调整视距和物体实际大小之间的关系。

至此，我们已经可以渲染出一个模型。

1. 实现相机动画以及天空盒

摄像机直接调用了LearnOpenGL提供的camera类，动画实现是通过绑定processInput来通过键盘操作修改camera的地址，同时通过鼠标的回调函数来修改鼠标移动对于相机的位移，然后修改camera类里的cameraFront来修改摄像机的指向。同时对摄像机进行限制，防止其角度超过90度。

天空盒是复现了LearnOpenGL的天空盒，天空盒渲染了一个6个面的立方体，并且disable了深度检测，所以摄像机无论如何移动z轴距离都是始终不变的。

1. 多模型存储管理

由于代码结构问题，我把GUI类作为主要管理类进行管理。不同于LearnOpenGL的封装了game或者window类别单独放在main函数里进行渲染循环，我把大部分的参数调用都直接放在了main函数里，为了便于传递参数和检查，GUI类相当于整个游戏的管理类，所有游戏内的变量都通过GUI类储存生成，同时GUI类为全局变量，便于main函数以及其它回调函数调用GUI里内容。

GUI首先要储存可在GUI内修改的参数，比如光照方向、材质类型等等，其次GUI保存了贝塞尔曲线类型的vector，通过GUI可以修改贝塞尔曲线vector现在编辑的index，这样可以实现修改当先编辑的model，并且也便于增加/修改其他的模型。

同时，文件的储存采用用户直接输入文件名字，然后对于该模型进行存入/读取。由于macOS没有便于使用的文件资源管理器进行文件管理，故这里直接采用了文件名读取的方法进行实现。

1. UI绘制管理

为了便于UI渲染和管理，本项目里引入ImGUI库进行绘制。但由于ImGui没有内置的文件资源管理器，所以不能实现对于文件的可视化选择地址储存。由于ImGui库的引用，整个系统的UI较为统一，并且可以实时查看帧率便于debug。

1. 实现粒子效果

粒子效果的实现参考了<https://www.cnblogs.com/ForEmail5/p/6848670.html>

粒子系统相当于渲染N个小正方形面片，其中正方形面片的速度、加速度方向是随机的，并且在生成一段时间后，生命周期结束应该主动消亡。正方形面片应该在掉落的时候有在力的作用下所产生的运动轨迹，所以应该在合力的作用下确定其速度。

1. 总结

项目基本完成了所有的要求内容，实现了通过贝塞尔曲线渲染模型，并且场景内部也可以加载其他模型，可以对于场景里渲染的所有的模型的参数进行修改，符合项目的大多数要求。

由于系统环境及编译器原因，在项目进行的后期debug模式失效，没办法对于项目进行双击运行以及在debug模式进行其他修改，所以项目内容更新也因为系统bug暂时停止。

本项目没有成功运行LearnOpenGL的Resource Manager，导致很多Shader、texture的声明都需要显示的放在外面，命名不太有统一性。

由于素材有限，本项目的天空盒并没有找到足够实现效果的其他天空盒，所以呈现效果较差，美观性较差，可以在后期修改天空盒的texture，重新绑定。

1. 其他

可执行文件在cmake-build-debug文件夹内，需要通过命令行执行。

运行实例图如下，同时也可以参考UI界面。

电脑的屏幕截图

描述已自动生成