个人报告

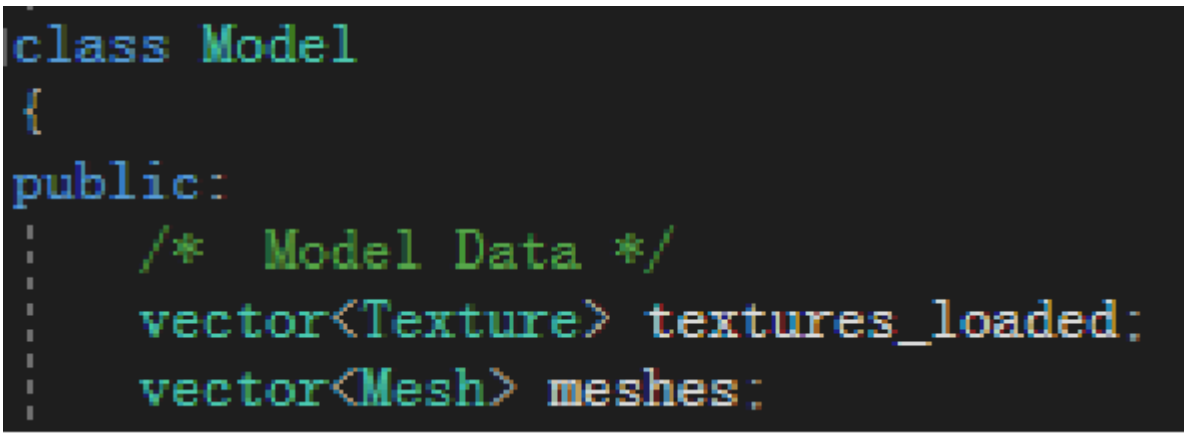
个人工作清单

- Camera Roaming
- Texture mapping
- Model import&Mesh viewing
- Sky Box
- Display text

具体实现内容

模型导入

模型导入需要Assimp库。它导入一个模型的时候，将整个模型加载进一个场景对象中，这个对象包含了许多子节点，里面是各种存储数据的索引。 而我们在将物体加载到场景对象之后，遍历节点来获取对应的网格对象，对其进行处理获得顶点数据，索引和材质属性。这部分工作交给mesh类来完成。 最后得到的一系列网格数据，用一个Model对象包装起来



有了Model类包装网格之后，我们很容易就能导入一个支持格式并且纹理路径正确的模型。 但是导入的模型的大小一般不符合我们的意愿，所以我们需要在搭建场景时，对其进行缩放处理，还有平移旋转等处理，使得其在场景中看起来更合理。

```
// 调整人物的位置
player_model = glm::mat4(1.0f);
player_model = glm::translate(glm::mat4(1.0f), playerPos);
player_model = glm::scale(player_model, glm::vec3(0.05f, 0.05f, 0.05f));
player_model = glm::rotate(player_model, glm::radians(180.0f + rotateAngle),
shader.setMat4("model", player_model);
allModels[0].Draw(shader);

// 调整树的位置
glm::mat4 treemodel1 = glm::mat4(1.0f);
treemodel1 = glm::translate(treemodel1, treePos1);
treemodel1 = glm::scale(treemodel1, glm::vec3(0.05f, 0.05f, 0.05f));
shader.setMat4("model", treemodel1);
allModels[1].Draw(shader);
```

天空盒Sky Box

天空盒实际上就是一个立方体，所以我们要渲染出天空的效果，需要一个立方体的顶点数组（立方体的边长和地面长宽一样），画出这个6方体之后，再进行纹理映射。

然后为了使得天空盒以玩家为中心，即玩家移动，立方体贴图不能移动。因此我移除了观察矩阵中的位移部分，让移动不影响天空盒的位置向量。

```
//天空盒
// 更改深度函数，以便深度测试在值等于深度缓冲区的内容时通过
glDepthFunc(GL_LEQUAL);
skyboxShader.use();
// 从视图矩阵中删除平移
view = glm::mat4(glm::mat3((*camera).GetViewMatrix()));
skyboxShader.setMat4("view", view);
skyboxShader.setMat4("projection", projection);
```

为了更高效地工作，我最后才去渲染天空盒，并且在其顶点着色器中使用透视除法，让其认为天空盒有最大的深度值1.0，这样只有天空盒前面没有其他物体渲染时才去渲染它，节省资源。

```
void main()
{
    TexCoords = aPos;
    vec4 pos = projection * view * vec4(aPos, 1.0);
    gl_Position = pos.xyww;
}
```

效果如图：


Project
WELCOME TO THE THRILLING VILLAGE!


Project
WELCOME TO THE THRILLING VILLAGE!

文字显示Display Text

使用Freetype这个库来实现文本渲染，freetype可以帮我们加载TrueType字体并生成对应位图及度量值，我们则需要提取生成的位图作为纹理即可，为了高效渲染字符，用一个结构体储存字形纹理的ID，及相关度量值，结构体又存进一个map中。

```
// 定义字符
struct Character {
    GLuint TextureID;   // ID handle of the glyph texture
    glm::ivec2 Size;    // Size of glyph
    glm::ivec2 Bearing;  // Offset from baseline to left/top of glyph
    GLuint Advance;    // Horizontal offset to advance to next glyph
};
std::map<GLchar, Character> Characters;
```

这里只生成了ASCII字符集的前128个字符，对每一个字符都生成纹理并将其数据存到前面的结构体里，然后加到前面的map表里。之后在定点着色器渲染字形，在片段着色器渲染文本颜色。

每个字形纹理最后会被绘制在一个四边形上，显示出来。这里用的是教程中给出的RenderText函数对我所需要的字符串进行渲染。

```
// Render glyph texture over quad
glBindTexture(GL_TEXTURE_2D, ch.TextureID);
// Update content of VBO memory
glBindBuffer(GL_ARRAY_BUFFER, textVBO);
glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(vertices), vertices);

glBindBuffer(GL_ARRAY_BUFFER, 0);
// Render quad
glDrawArrays(GL_TRIANGLES, 0, 6);
```

效果如图：

个人总结

这次的期末project，基础部分就是之前所有作业的大杂烩，加在一起重新实现一遍。然后模型导入部分还是耗时有点久的，主要是我也负责素材的搜索和导入，以及场景建模等部分的工作，很多网上的模型导入会失败，原因是他们的格式不够标准，不能被assimp解析。然后bonus部分我还通过学习博客了解到了如何添加天空盒和显示文字，在添加文字前以为比较简单，没想到还是挺复杂的，过程还涉及到一些第三方库的导入，也踩了一些坑，总之通过这次期末project我学到了更多关于cg的东西，能将它们综合起来运用了，不再像写作业时那样，单单实现一个小部分，受益匪浅。