Project Report for CG100433 Course

Team member

学号	姓名
1853659	叶茂尧
1751957	俞少作
1750044	王佳雯
1852414	李昊龙
1752045	赵冰辰

Project Title

基于OpenGL的Minecraft试制版

Abstract

本项目实现了一个Minecraft沙盒游戏的Toy Demo。项目还原了Minecraft中部分种类的方块和场景，卡通效果的光照渲染，运动逻辑以及挖掘和放置方块。由于Minecraft的地图随机生成较为复杂，因此我们参考网上的部分数学模型，实现了较为简单几种场景的随机地图的无限生成。

同时，我们的工作开源在GitHub上https://github.com/SAOHPRWHG/ToyMC

Motivation

Minecraft作为一款在前几年风靡全球的沙盒游戏，以其优秀的游戏性和独特的方块风格深受广大游戏爱好者的喜爱。本组成员均对这款游戏拥有浓厚的兴趣并亲身体验过其趣味性。进而想通过这次小组作业实现一个Minecraft的demo性质的小游戏。本组希望实现能动态随机性生成方块地图，并构建相关场景特征的地图生成器。同时编写对应的物理引擎和框架，对场景内的环境使用shader进行多光源的处理，以实现对Minecraft游戏内容的简单模拟。

本项目具有一定的趣味性和挑战性，我们在完成拟开发基本的动态地图生成器的基础上，加入一定程度的纹理和物理特性的展示，并结合shader，以达到较为惊艳的效果， 此外，本项目搭建拟一个完整的游戏系统，包含跳跃，

破坏和创造方块等交互内容，完成物理引擎和人机交互的部分。

The Goal of the project

1. 实现Minecraft的物理效果；
2. 实现人机交互的规则；
3. 实现常见方块和场景的建模；
4. 实现对自然光源和人造光源的模拟。
5. 实现地图的自动生成

The Scope of the project

- 场景部分只完成较少种类的方块模拟；
- 关于物品捡拾，道具栏，合成等方面暂不涉及；
- 光照主要以太阳和火把两个具体光源为例实现；
- 地图生成算法视项目进展情况而定；

Related CG techniques

1. 图形渲染流水线
   - 光栅化
   - 全局光照
   - 局部光照
   - 纹理
2. 物理逻辑
   - 行走，重力，碰撞，飞行
   - 方块破坏与放置

Project contents

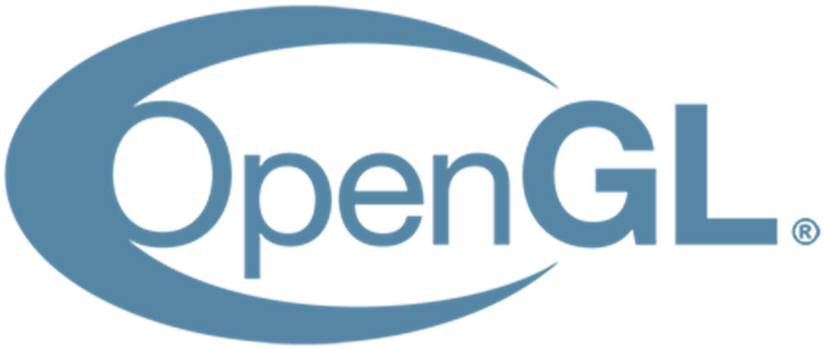
项目是一个Minecraft模拟demo，将实现以下功能：

- 实现各种特性的方块组成的场景
- 人物的行走，跳跃，碰撞，破坏方块与放置
- 全局光照与局部光照
- 地图的随机生成

Implementation

1. 开发环境

本项目使用C++和OpenGL开发，OpenGL环境库采用了GLAD和GLFW，Modern OpenGL版本为3.3



由于本项目5位成员的工作环境包含Windows，MAC，Unbuntu三种不同的操作系统，故跨平台项目协同一度成为一个困难的问题。最终由叶茂尧同学提供了Cmake跨平台编译脚本，完成了跨平台的编译要求。项目编译推荐采用较高版本的Mingw；MSVC可以完成编译，但可能存在一些细微的版本bug。

2. 场景建模

被建模的物体包括：
1. 各种不同类型的方块
2. 花草
3. 树

2.1 方块的建模

由于方块属于非常简单的模型，所以我们未使用导入obj模型的方式进行建模，而是直接在代码中以hard code的形式指定好方块的所有顶点位置、normals、uv坐标、索引等各种信息。相关的部分代码如下图所示。

```
static const float positions[6][4][3] = {
    {{-1, -1, -1}, {-1, -1, +1}, {-1, +1, -1}, {-1, +1, +1}},
    {{+1, -1, -1}, {+1, -1, +1}, {+1, +1, -1}, {+1, +1, +1}},
    {{-1, +1, -1}, {-1, +1, +1}, {+1, +1, -1}, {+1, +1, +1}},
    {{-1, -1, -1}, {-1, -1, +1}, {+1, -1, -1}, {+1, -1, +1}},
    {{-1, -1, -1}, {-1, +1, -1}, {+1, -1, -1}, {+1, +1, -1}},
    {{-1, -1, +1}, {-1, +1, +1}, {+1, -1, +1}, {+1, +1, +1}}
};
static const float normals[6][3] = {
    {-1, 0, 0},
    {+1, 0, 0},
    {0, +1, 0},
    {0, -1, 0},
    {0, 0, -1},
    {0, 0, +1}
};
static const float uvs[6][4][2] = {
    {{0, 0}, {1, 0}, {0, 1}, {1, 1}},
    {{1, 0}, {0, 0}, {1, 1}, {0, 1}},
    {{0, 1}, {0, 0}, {1, 1}, {1, 0}},
    {{0, 0}, {0, 1}, {1, 0}, {1, 1}},
    {{0, 0}, {0, 1}, {1, 0}, {1, 1}},
    {{1, 0}, {1, 1}, {0, 0}, {0, 1}}
};
```

对于不同类型的方块，其实本质只是贴图不同。只需要为我们的方块模型指定不同的纹理即可。

方块的纹理贴图其实只需要一张贴图。由于minecraft是像素风格，因此每个方块每一面的贴图大小只需要16×16。因此我们将不同类型方块的不同面的贴图组合到一起，形成一张256×256的大贴图。我们通过一个数组存储每个不同类型方块不同面的贴图在大贴图中的位置，并且在实际准备数据时，根据这个位置坐标与上图中的uvs数组进行计算，以计算出在大贴图中的实际uv坐标。

2.2 花草的建模

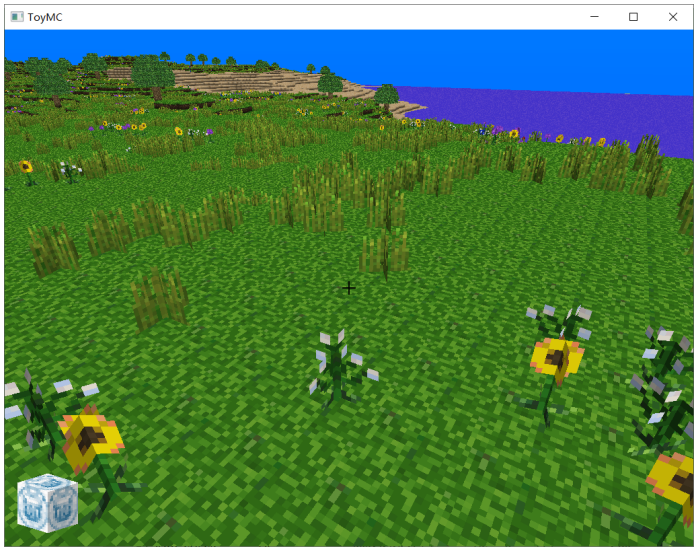
花草在minecraft中是以交叉的两个平面的形式呈现的。同样，我们也对花草以hard code的形式进行建模。相关的部分代码如下图所示。

```
static const float positions[4][4][3] = {
    {{ 0, -1, -1}, { 0, -1, +1}, { 0, +1, -1}, { 0, +1, +1}},
    {{ 0, -1, -1}, { 0, -1, +1}, { 0, +1, -1}, { 0, +1, +1}},
    {{-1, -1,  0}, {-1, +1,  0}, {+1, -1,  0}, {+1, +1,  0}},
    {{-1, -1,  0}, {-1, +1,  0}, {+1, -1,  0}, {+1, +1,  0}}
};
static const float normals[4][3] = {
    {-1, 0, 0},
    {+1, 0, 0},
    {0, 0, -1},
    {0, 0, +1}
};
static const float uvs[4][4][2] = {
    {{0, 0}, {1, 0}, {0, 1}, {1, 1}},
    {{1, 0}, {0, 0}, {1, 1}, {0, 1}},
    {{0, 0}, {0, 1}, {1, 0}, {1, 1}},
    {{1, 0}, {1, 1}, {0, 0}, {0, 1}}
};
static const float indices[4][6] = {
    {0, 3, 2, 0, 1, 3},
    {0, 3, 1, 0, 2, 3},
    {0, 3, 2, 0, 1, 3},
    {0, 3, 1, 0, 2, 3}
};
```

其中，由于花和草的高度不同，因此在实际准备数据的时候，我们将使用一个系数n，将n乘以positions数组中的位置，以实现花与草大小不同的效果。

```
*(d++) = n * positions[i][j][0];
*(d++) = n * positions[i][j][1];
*(d++) = n * positions[i][j][2];
```

花与草的不同纹理的实现与方块相同。



2.3 树的建模

树由多个方块组成。并且，树在生成之后，并不需要我们存储树的数据结构，玩家与树的交互仍以方块为单位，所以我们并不需要对树整体进行建模，而是在地图生成器决定某位置生成树之后，按照某种生成规则在该位置附近生成一系列的方块。生成规则如下：

在树的生成点朝y轴方向（即上方）连续生成6个树干方块；

对于树的生成点上方第3层到第7层，在x轴与z轴方向上3×3的范围内，根据块到树干第3层中点的距离决定是否生成树叶方块。

3. 渲染优化

Minecraft中世界的组成单位为立方体，每个立方体由12个三角形构成，如果每个立方体都进行渲染，仅长宽为10*10的场景，由于生成高度超过128，实际填充块一般大于30，因此实时渲染三角形数量超过36000个，画面会出现明显的卡顿。

因此需要改变立方体的渲染方式。我们最终的解决方案采用以面为单位进行渲染，对每个立方体进行判断，其六个面是否可见（也可以理解为是否与空气接触），对不可见的面则直接跳过不渲染。

采用该策略的10*10场景的方块，渲染的三角形数量约为200个左右，渲染量大约缩小了180倍。

4. 光照处理

4.1 全局光照

全局光照其实是基于对每一个方块的单独实现；

光照模型为去除镜面反射的冯氏光照模型（Phong Shading），

```
vec3 light = ambient + light_color * df;
```

即光照由环境光照+漫反射光照组成；

但光照值是昼夜时间的函数，白天光照值较高，夜晚光照值较低。

4.2 远景虚化/海底效果

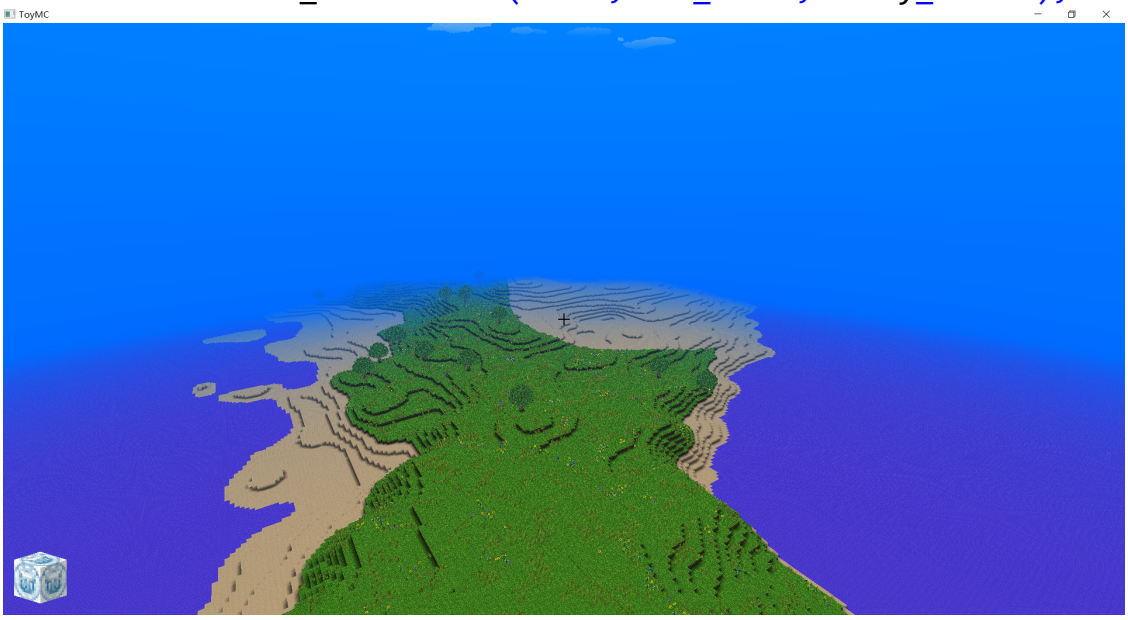
远景虚化通过和天空颜色的线性插值来实现：

```
color = mix(color, sky_color, fuzzy_factor);
```

虚化系数由所在高度和注视物体的。

同理，对水下视野的模糊效果也采用相同的实现方式，线性插值采用海蓝色即可。（该插值方法本来是为水下视野所设计，但由于本项目目前没能实现水的渲染逻辑，所以仅用于远景虚化）

```
Underwater_color = mix(color, sea_color, fuzzy_factor);
```



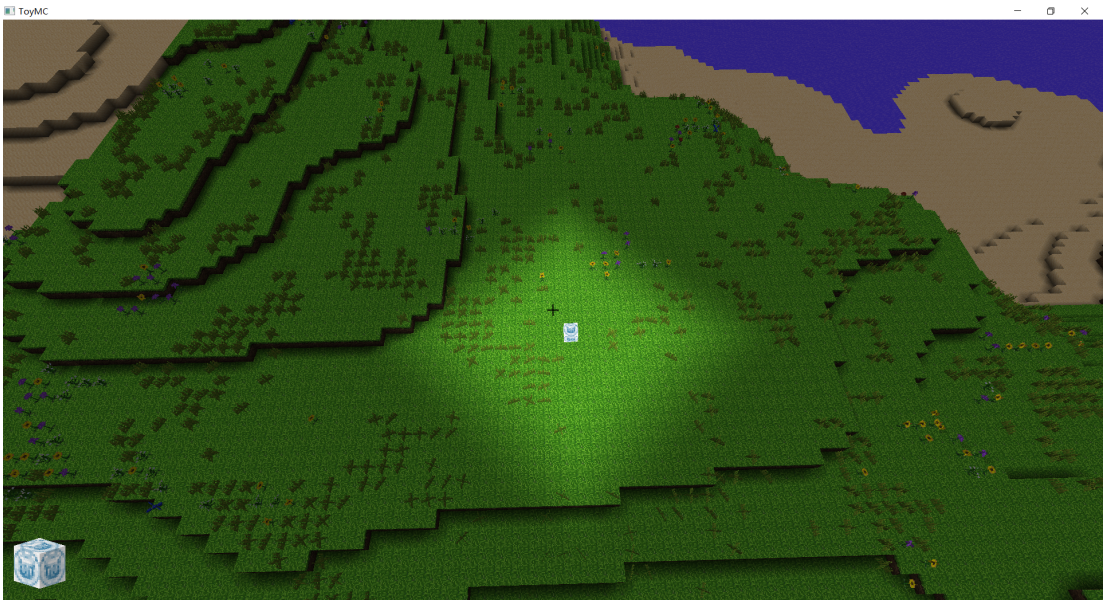
4.3 局部光照

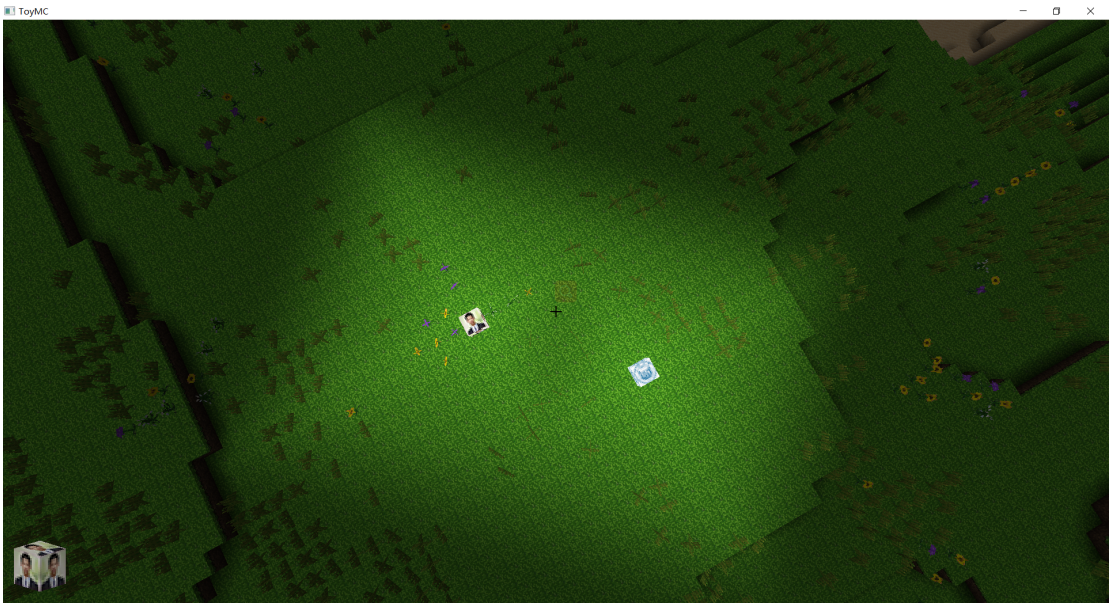
由于火把建模与常规物体有一定区别，最终采用两种发光方块作为光源。局部光照效果的实现，参考Minecraft游戏本身的卡通特性，是自安方法较为简单：

1. 首先，为了避免设置不定数量光源的问题，采用光照地图，在地图中存储每个方块的光照强度信息，在shader中根据光照强度渲染；

2. 从光源开始按照最大光照强度从中心向四周线性递减；

2. 周围同时存在多个光源时，重叠区域光照强度叠加计算，但不会超过最大光照强度；





6. 地图生成逻辑

3.1 地图存储

我们的地图以chunk为单位进行存储。所谓的chunk是包含一个x轴和z轴上各32个方块、y轴上所有方块的一个方块组。地图的生成也是以chunk为单位进行生成的。程序会计算玩家视野范围内能够看到的chunk，若在某个玩家能看到的位置还没有chunk，则会在该位置一次性生成一个chunk。

每个chunk会存储该chunk在所有chunk中的相对坐标，chunk的容量，chunk中已有方块的数量，以及chunk中所有方块的位置和其方块类型。chunk中的所有方块都存储在chunk的一个数组中，而具体哪个方块存储在数组中的哪个位置，则由方块在chunk中的相对坐标(x,y,z)经过哈希之后决定。

3.2 地图生成

在地图生成的过程中，我们会生成两个中间结果地图：

1. heightmap(x,z)：在chunk内(x,z)坐标下，该点的高度；
2. biomemap(x,z)：在chunk内(x,z)坐标下，该点的生态类型。

其中，高度地图又是由生态地图所决定的。

首先，我们使用二次数学模型对chunk内的每一个(x,z)点生成一个生态类型。然后，我们将整个chunk分为四份，在每一份中对上下左右四个角根据生态类型决定的参数以相同的数学模型生成高度，再对除了四个角以外的所有点做插值，最终形成了一份整个chunk内针对(x,z)坐标的高度地图。

有了生态地图和高度地图之后我们就可以进行具体的地图生成了。

在chunk内进行循环，枚举每个(x,y,z)：

a. 如果该点的高度在heightmap(x,z)之上，即高过了地图表面，却在海平面以下（海平面是一个全局参数），则在该点放置类型为水的方块；

b. 如果该点高度等于heightmap(x,z)，即刚好处于地图表面，且在海平面之上4个方块距离内，则放置沙漠类型的方块；

c. 如果该点高度等于heightmap(x,z)，即刚好处于地图表面，且高于海平面4个方块单位，则根据该点的生态类型biomemap(x,z)，放置相应类型的表面方块（例如如果是草地生态类型则放置草的方块，如果是雪地生态类型则放置雪块）。

并且在能够长花草树木的地方（例如草地）随机在该处生成花草或者树；

d. 如果该点高度等于heightmap(x,z)，即刚好处于地图表面，且在海平面以下，则根据生态类型放置相应的水下方块（一般为沙地或泥土）；

e. 如果该点高度小于heightmap(x,z)，即在地下，且与地表距离不超过3，则放置泥土类型方块；

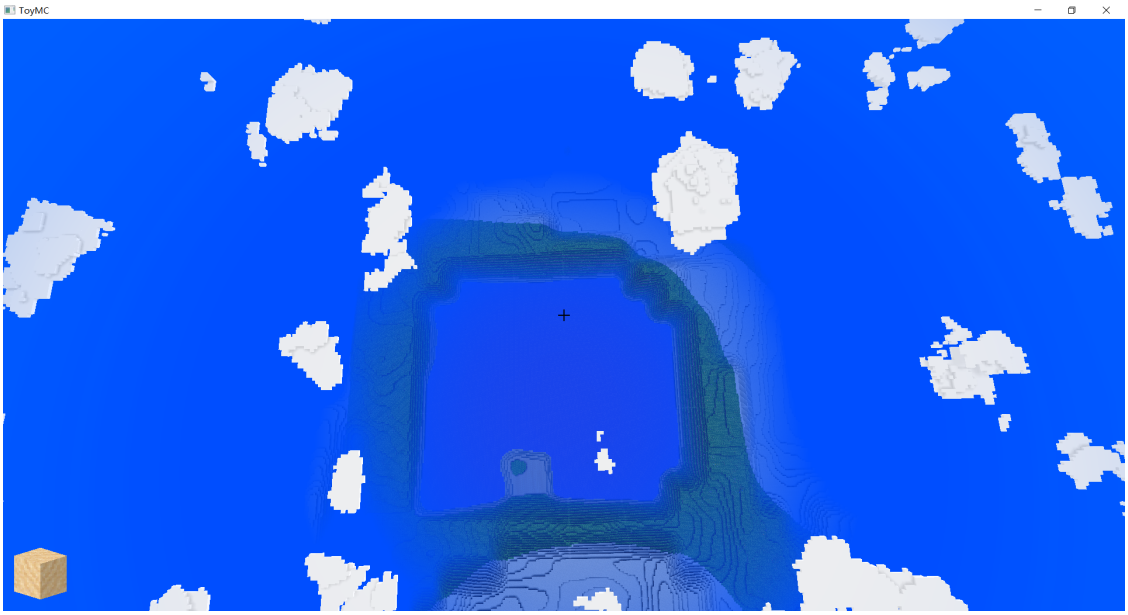
f. 如果该点高度小于heightmap(x,z)，即在地下，且低于地表3个方块距离，则放置石头类型方块。

本地图生成的数学模型参考了[3]中的实现方法。项目部分细节与优化方法参考了[6]中的实现逻辑。

7. 基本操作

1. 双击ToyMC.exe进入游戏；
2. 通过WASD控制方向，空格键为跳跃；
3. 按下F键开启飞行模式，飞行模式下的运动速度为行走速度的2.5倍；再次按下F键切换至普通模式；飞行模式下的空格键为垂直起飞。
4. 左键点击敲除方块，右键点击放置方块；
5. 数字键0-9选择不同种类的方块；
6. 鼠标滚轮也可以选择不同种类的方块。

Results



Roles in group

人员	内容
1853659 叶茂尧	代码总体架构的制定和编写，整合
1751957 俞少作	摄像机与物理逻辑
1750044 王佳雯	场景建模与地图生成
1852414 李昊龙	地图生成
1752045 赵冰辰	方块和光照渲染

References

- [1] Learn OpenGL-CN
- [2] https://github.com/Polytonic/Glitter
- [3] https://github.com/Hopson97/MineCraft-One-Week-Challenge
- [4] https://www.minetest.net/
- [5] https://dev.minetest.net/Main_Page
- [6] https://www.michaelfogleman.com/projects/craft/