Project report for the CG 100433 course

Project Title

瀑布场景渲染

Team member

1652216 张佳蕾

1652217 王思懿

1652223 张文豪

1652224 张子岳

1652412 蒋维康

Abstract

本项目中我们建造了瀑布场景的3D模型，瀑布的原型参考了尼亚加拉瀑布。借助OpenGL，实现了瀑布场景中各元素的渲染，包括环绕瀑布的山、瀑布飞流而下的水、瀑布底端波澜起伏的水面、近距离观察瀑布的船和远处的天空。使用的相关技术包括着色器、粒子系统、法线贴图，通过矩阵变换将各元素缩放到统一尺度并组合在一个场景内，最终展现出环状大瀑布场景的效果。

项目源代码已上传至Github便于小组合作和展示。参见链接：TiOrg/Waterfall-Rendering

Motivation

自然美景精妙绝伦，即便是最顶尖的画家也难以完全还原其鬼斧神工。不过临摹是一切创造的基础，通过临摹来掌握表现实物的技术，这是计算机图形学入门的最佳途径之一。

尼亚加拉瀑布原意雷神之水，其流速、流量在全世界独一无二。同时半弧形的山口营造了很强的空间感，山口的中央也是乘船观赏的最佳位置。取景于此，我们可以较好的实现粒子飞落的冲击感。以及将视角固定在山口的内部位置，也就是船的附近，可以省略远处杂景的渲染工作，使视角范围内的景物较为饱满丰富。

Goal of the project

1. 瀑布渲染和粒子系统的优化，致力于尽可能降低粒子系统的计算量，并且使用效率较高的方式渲染

2. 各部件的合理组合，由于元素种类各不相同，需要对各元素进行调参以合理地融合在同一个场景内
3. 着色器和顶点缓冲编程的实践，运用现代OpenGL技术渲染，而不使用glut相关函数

Scope of the project

1. 只渲染了可见区域内的场景，超过一定视角范围将看见空无一物的天空盒
2. 未处理穿模问题，即未计算各部件的包围盒和可能的碰撞
3. 未能找到合适的粒子颜色模拟真实的瀑布，故选择了蓝色作为替代
4. 使用粒子系统，导致内存和CPU均需要高负荷运转，在一定情况下可能出现些微掉帧卡顿
5. 由于计算量较大，未实现相应的水雾效果

Involved CG techniques

1. 粒子系统-particle system：用于实现瀑布效果。粒子系统产生大量粒子，对每个粒子保存其位置、速度、颜色（RGBA）、生命等状态，并在每一次渲染中更新其状态。若粒子生命超过一定时间，将死亡并被回收。
2. 混合-blending：用于实现瀑布效果。透明化是指一个物体非纯色，它的颜色是物体本身的颜色和背后其它物体的颜色按不同强度结合的结果。使用 OpenGL 混合参数 GL_ONE_MINUS_SRC_COLOR 对粒子进行混合，粒子的最终颜色为

$$\vec{C}_{result} = \vec{C}_{source} * alpha + \vec{C}_{destination} * (1 - alpha)$$

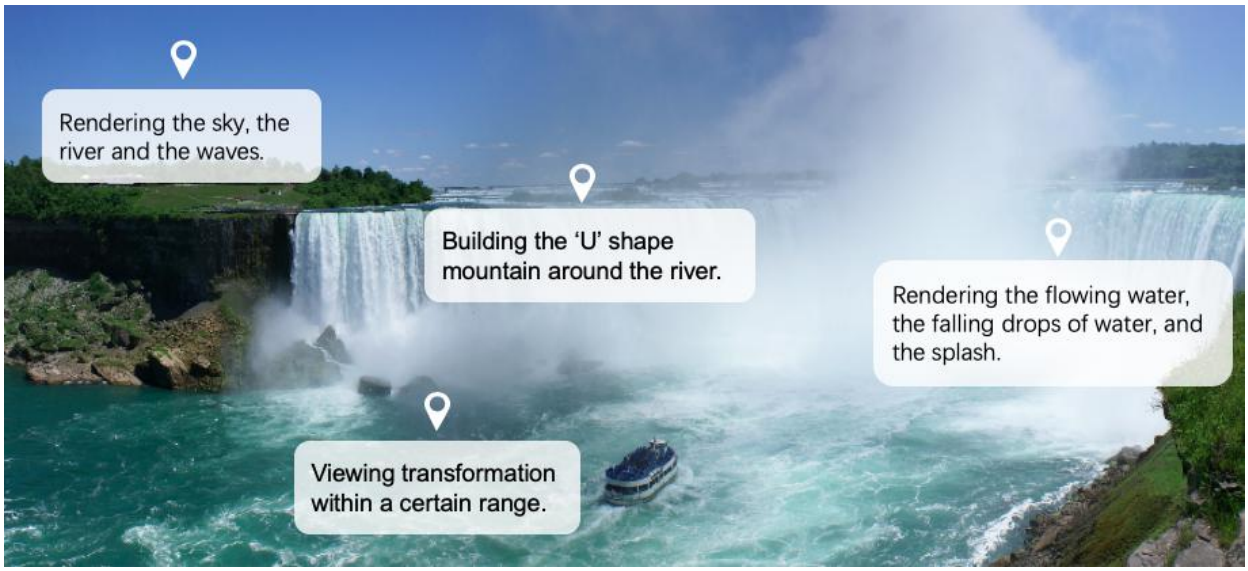
3. Gerstner 波：用于实现水面效果。理想情况下的水波是正弦形状的，但实际中总会受到扰动。使用 Gerstner 波对正弦波进行参数修正，可模拟不平静的水面。

$$\mathbf{P}(x, y, t) = \begin{bmatrix} x + \sum \left(Q_i A_i \times \mathbf{D}_i.x \times \cos\left(w_i \mathbf{D}_i \cdot (x, y) + \varphi_i t \right) \right), \\ y + \sum \left(Q_i A_i \times \mathbf{D}_i.y \times \cos\left(w_i \mathbf{D}_i \cdot (x, y) + \varphi_i t \right) \right), \\ \sum \left(A_i \sin\left(w_i \mathbf{D}_i \cdot (x, y) + \varphi_i t \right) \right) \end{bmatrix}.$$

4. 立方体贴图：用于实现天空盒。简单来说，立方体贴图就是一个包含了 6 个 2D 纹理的纹理，每个 2D 纹理都组成了立方体的一个面，让玩家产生其位于广袤天空中的错觉
5. 3ds MAX：用于实现山体和游轮。全功能的三维计算机图形软件，可以创建复杂的 3d 对象及其纹理。

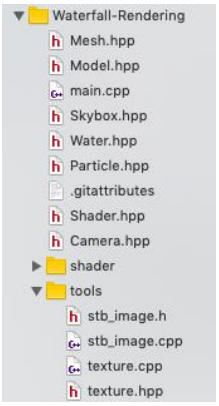
6. 着色器：着色器是运行在 GPU 中负责渲染算法的一类总称，是使用一种叫 GLSL 的类 C 语言写成的。GLSL 是为图形计算量身定制的，它包含一些针对向量和矩阵操作的有用特性。

Project contents

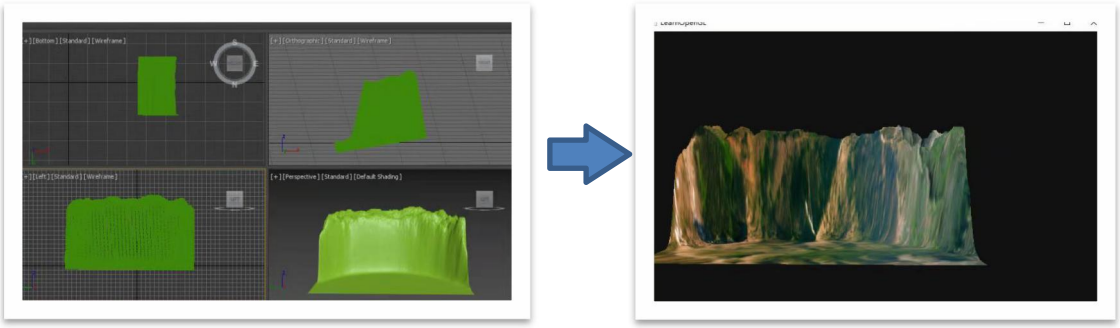


Implementation

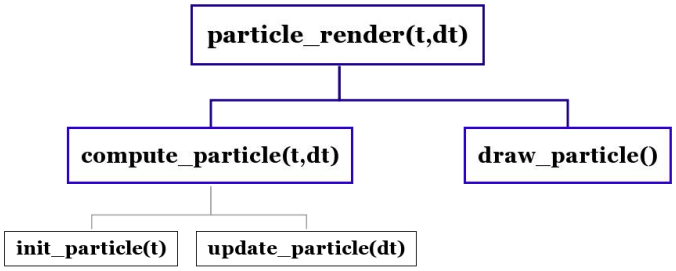
1. 综述：各元素均使用各自的类实现，在 main 函数中构造，为各类初始化着色器和顶点缓冲。在 main 的主循环中读取用户输入，以此更新 camera 并获得 Model、View、Projection 三个矩阵，将三矩阵传给给类并调用 draw 函数进行各元素的绘制。

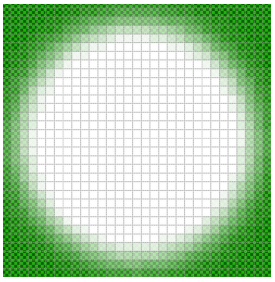


2. 摄像机：记录其位置方向向量，通过读取用户的按键输入更新位置和方向，并设置接口返回 View 矩阵。
3. 纹理加载：包括读取 jpg、png、dds 等纹理图片文件。通过 stb_image 库函数读入数据，并使用着色器创建纹理单元的映射。
4. 模型导入：主要使用 assimp 库。读取 obj 文件及其对应的纹理文件，每个模型使用一个 Model 类处理，每个模型的各个部件（模型支持拼接而成）使用一个 Mesh 类处理。为了提高效率，将 obj 中的顶点数据（即 v）传入顶点缓冲（VBO），将片段数据（即 f）传入元素缓冲（EBO），避免了顶点的重复绘制。



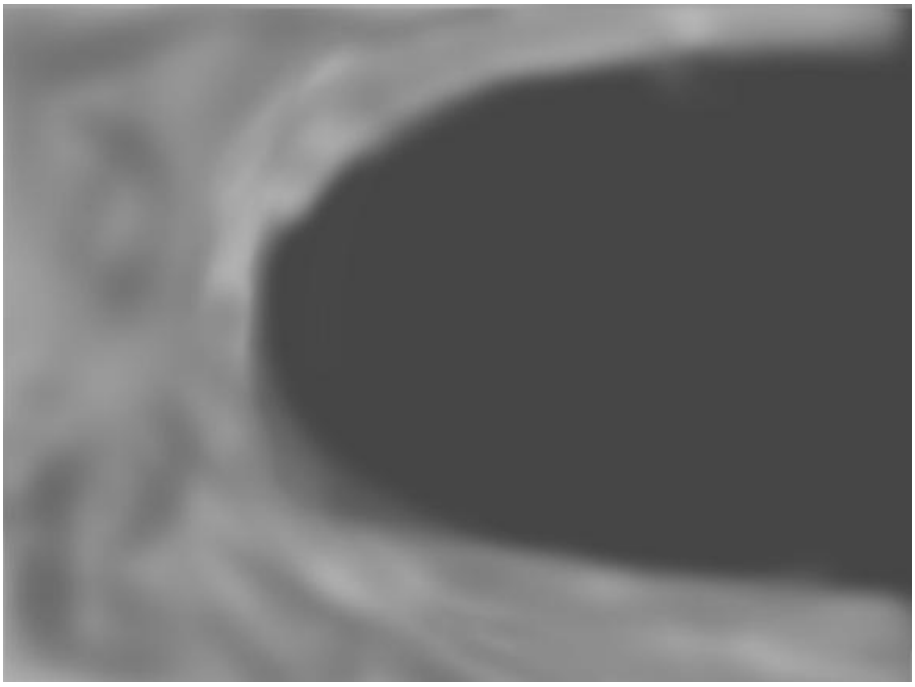
5. 瀑布：使用 150000 个粒子模拟。每一个粒子做初速度随机的平抛运动，并且到达水面的粒子将死亡，从发射点重新落下。以当前时刻 t 为参数来产生新的粒子（SpawnParticles），对其各状态赋初值。以每一帧经过时间 dt 为参数更新每一个粒子的状态（UpdateParticles）。其算法流程图如下：



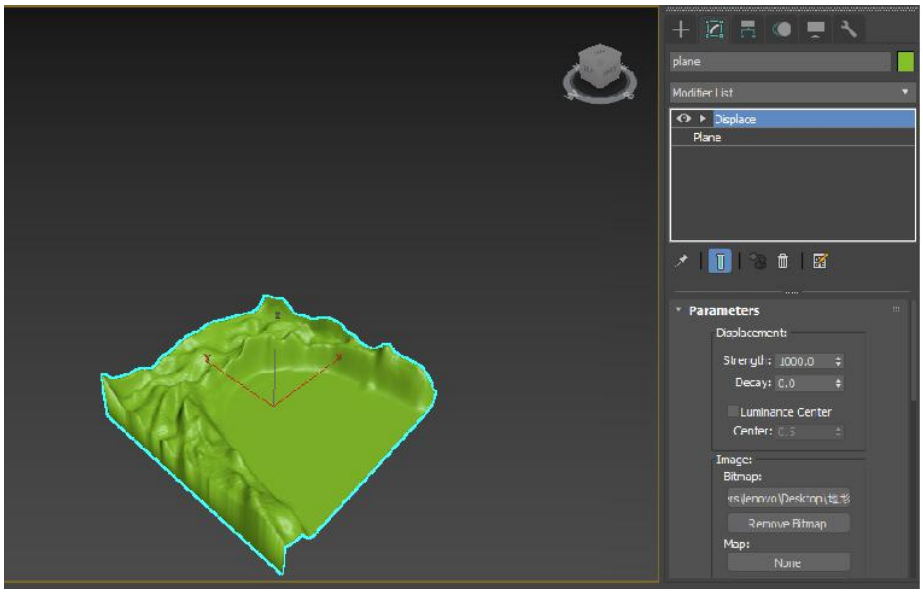


具体实现时粒子又分为两个部分。其一是容器，用于实现物理状态的模拟，也就是前文所述的产生和更新；其二是纹理，每一个粒子其实并不是一个球，而是一个圆形片状纹理，此纹理会根据camera的位置进行旋转，保持垂直于相机的方向（也就是法线与相机方向向量平行），以此实现所有角度看粒子都是一个圆形的效果。DDS纹理原型如右图所示。粒子的透明效果通过混合（blending）实现，为了保证混合的正确性，需要通过粒子与camera的距离对粒子进行排序，由远及近绘制，同时所有粒子都是在场景中最后绘制，这样使得粒子的颜色混合了本身以及后方物体的颜色，实现了水珠的半透明效果。

6. 水面：使用 80*80 的网格以及 6 个不同参数的 Gerstner 波，遍历每个网格顶点计算各波的叠加后的高度，并对每个网格计算法线方向。通过法线贴图，刻画水面的细节纹理，并用着色器为其添加光照。
7. 山脉：使用 3ds max 中的位移修改器进行地形建模。其主要原理为在 3ds max 中新建一个平面，并根据下面的灰度位图进行位移：



3ds max 中的位移修改器(Display)通过一张灰度位图生成物体表面凹凸不平的平面。结果如下图所示：

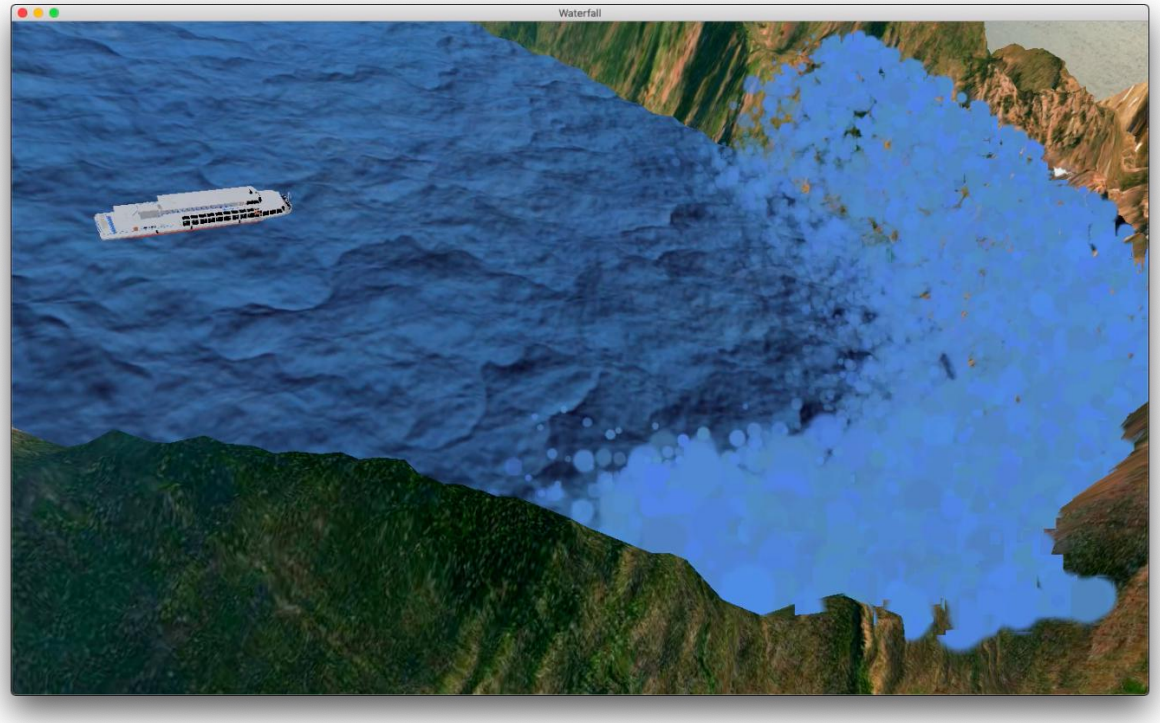


最后将贴图和 obj 文件一起传入模型加载类进行绘制。

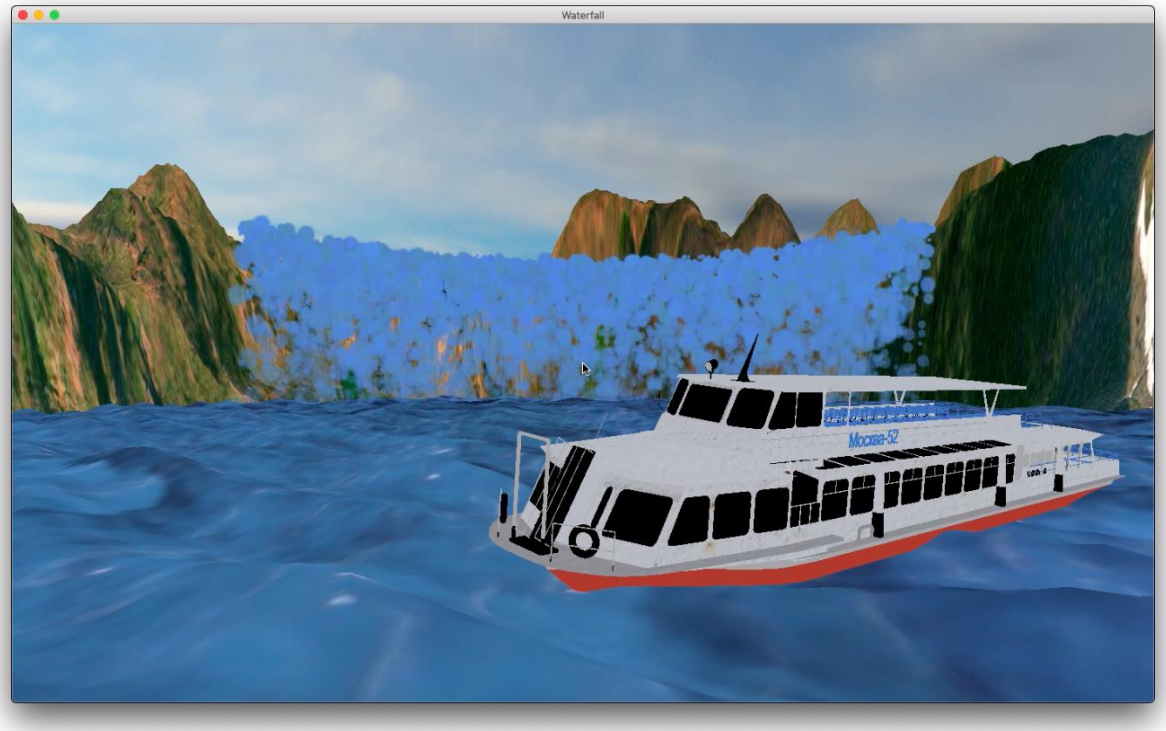
8. 游轮：和山同理，并且船体的各部分分别导入最后拼接而成。
9. 天空盒：使用立方体贴图，共六张纹理图片包括上下左右前后六个面，贴合在一个足够大的立方体内部。素材包括：

Results

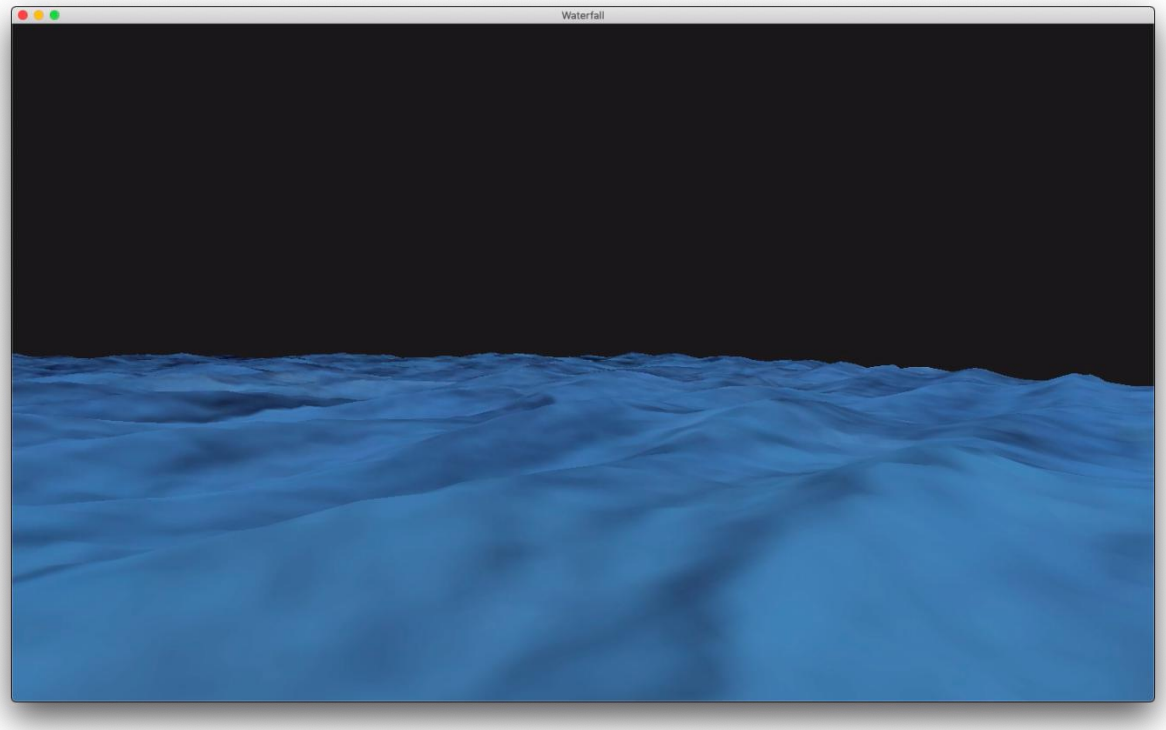
1. 俯视效果



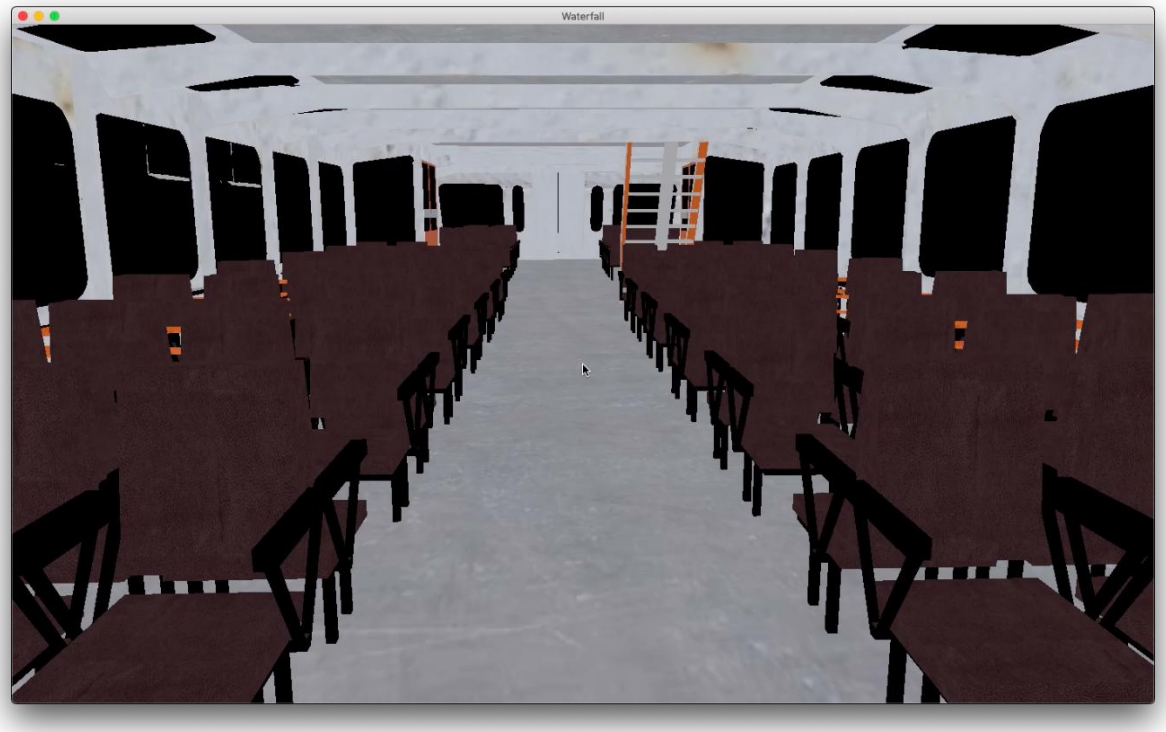
2. 平视效果

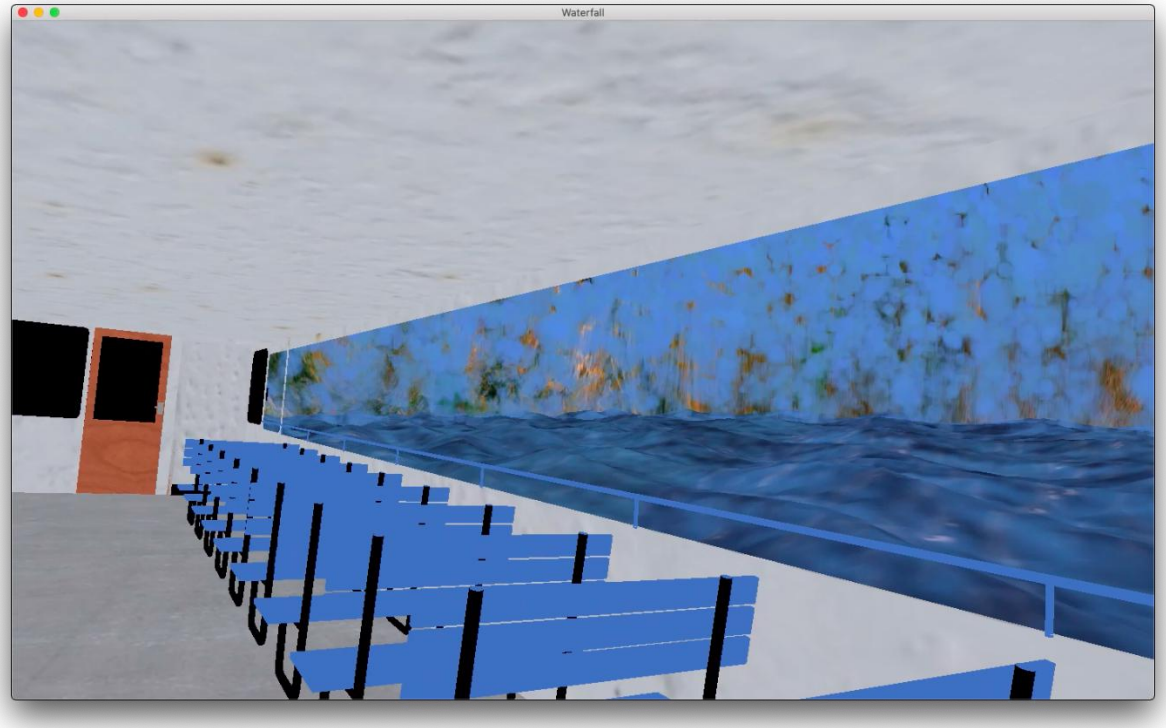


3. 水面效果



4. 船体内部效果





Roles in group

张佳蕾：3ds MAX山模型的制作，天空盒的制作

王思懿：水面的制作，各个元素的调参和素材的选取

张文豪：粒子系统的制作，Camera、Shader类和各工具函数的整合

张子岳：整体框架搭建，粒子系统的制作

蒋维康：模型设计和导入，各个元素的调参

References

- Nvidia. GPU Gems[M]. NVIDIA Corporation,2007:1-144
- Joey de Vries. LearnOpenGL CN[EB/OL]. learnopengl-cn.github.io, 2018-11.