二维/三维情况下的光滑粒子流体模拟和流体的简单非真实渲染

蒋锦豪 2400011785

2026 年 1 月 9 日

限于篇幅, 有关光滑粒子流体模拟的物理原理和代码实现的具体细节不再给出. 该报告主要展示项目实现的效果和一些主要思路.

1 二维光滑粒子流体模拟

1.1 基本功能

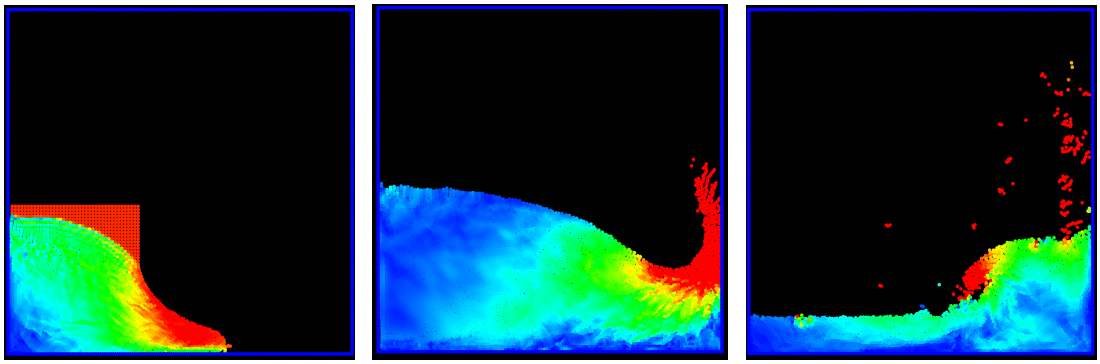
在 `System2d.h` 中定义了结构体 `SPHParams` 和类 `SPHSystem2d`, 分别用于存储光滑粒子流体模拟的参数和系统状态. 类 `SPHParams` 主要包括:

```
class SPHSystem2d {
public:
    std::vector<glm::vec3> positions;   //保存粒子的位置
    std::vector<glm::vec3> velocities;  //保存粒子的速度
    std::vector<float>     densities;   //保存粒子按照插值计算的密度
    std::vector<float>     pressures;   //保存粒子按照状态方程计算的压力
    std::vector<glm::vec3> forces;      //保存粒子受到的力
    int numFluidParticles = 0;      //流体粒子数量
    int numTotalParticles = 0;      //总粒子数量(包括边界粒子)
    SPHParams params;               //光滑粒子流体模拟的参数
    SPHSystem2d();
    SPHSystem2d(const SPHParams & p);
    void Advance2dSPHSystem(const SPHParams & p);   //推进系统状态
private:
    //构建空间哈希网格用于邻域搜索(此处略去)
    //核函数的系数以及核函数的定义(此处略去)
    void BuildGrid();   //根据粒子的位置构建空间哈希网格
    void ComputeDensities();    //计算粒子的密度
    void ComputeForces();   //计算粒子受到的力
    void IntegrateAndCollide();     //使用显式欧拉积分更新粒子的位置和速度
};
```

在 `SPHSystem2d` 中, 主要包含了粒子的位置、速度、质量等信息, 以及用于计算密度、压力和力的函数. 在每一帧更新中, 首先计算每个粒子的密度, 然后根据密度计算压力, 最后计算每个粒子受到的力并更新其位置和速度. 为了使模拟更流畅, 使用了 OpenMP 对所有循环进行加速.

在 `CaseSPHSystm2d.cpp` 中实现了二维光滑粒子流体模拟的可视化界面, 使用着色器 `SPH2d.frag` 和 `SPH2d.vert` 将流体粒子渲染为圆形, 其颜色根据速度的大小插值决定; 同时实现了初始化, 将边界设置为固定不动的粒子.

此外, 在左边的菜单栏中, 可以调整光滑粒子流体模拟的参数, 如粒子数量, 每一帧的迭代次数, 粘性系数等. 通过调整这些参数, 可以观察到不同的流体行为和效果. 下面是一些二维光滑粒子流体模拟的效果图:

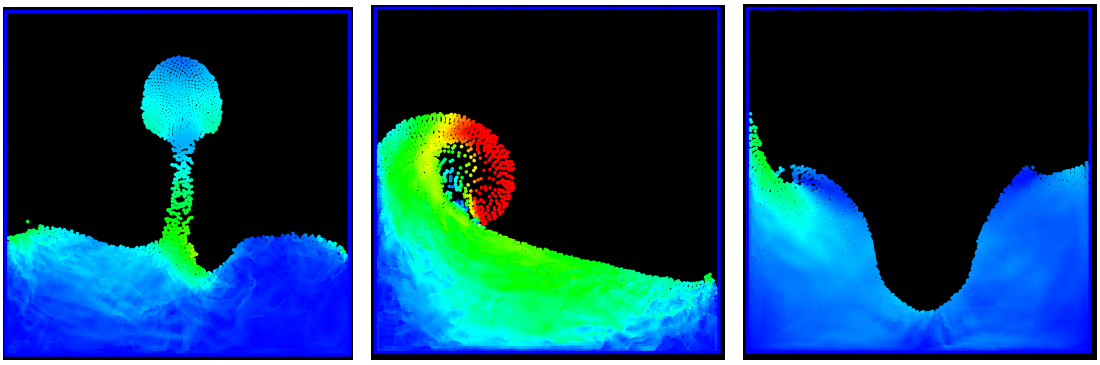


(a) 自由流动 I (b) 自由流动 II (c) 自由流动 III

图 1: 自由流动效果图 (初始状态为排列整齐的矩形)

1.2 交互功能

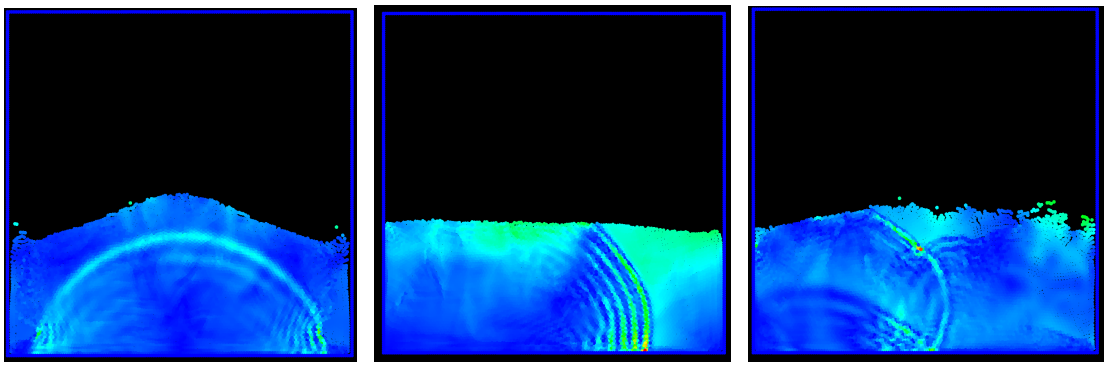
此外, 实现了对流体的交互功能. 按下鼠标左键将对流体施加指向鼠标位置的引力, 按下鼠标右键将对流体施加指向鼠标位置的斥力. 通过这种方式, 可以观察到流体在受到外力作用下的动态变化. 下面是一些交互效果图:



(a) 引力作用 I (b) 引力作用 II (c) 斥力作用

图 2: 流体交互效果图

同时, 还可以用交互功能观察流体中波的传播. 下面是一些波传播的效果图:

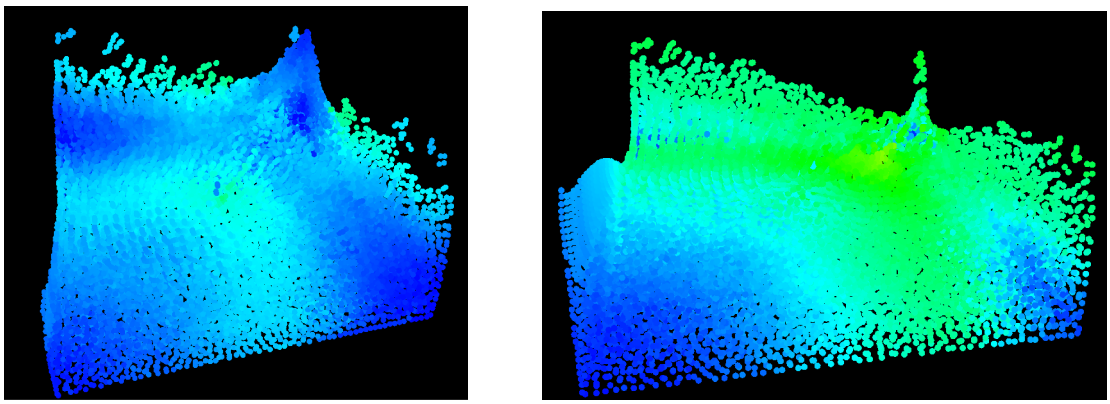


(a) 波的传播 I (b) 波的传播 II (c) 波的传播 III

图 3: 流体中波的传播效果图

2 三维光滑粒子流体模拟

在二维光滑粒子流体模拟的基础上, 实现了三维光滑粒子流体模拟. 在 `System3d.h` 中定义了类 `SPHSystem3d`, 其结构和功能与二维系统类似, 主要区别是各个参数需要适应三维的情形, 因此需要重新计算. 其它的功能和实现方法与二维系统类似. 最终的效果如下 (这里隐去了边界粒子):



(a) 三维流体模拟 I (b) 三维流体模拟 II

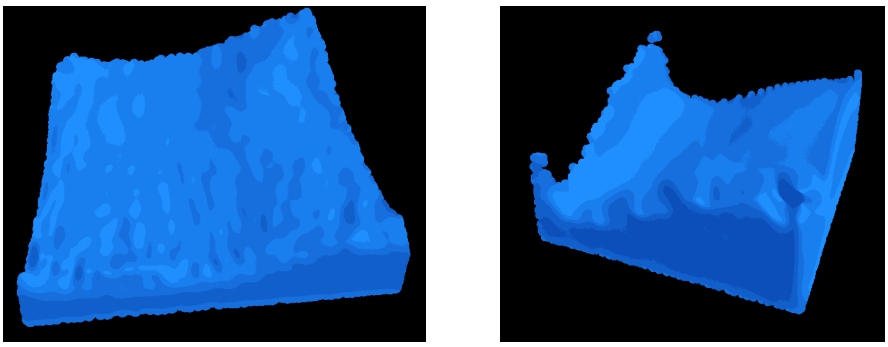
图 4: 三维光滑粒子流体模拟效果图

3 流体的简单非真实渲染

在实现了二维和三维光滑粒子流体模拟后, 对三维流体进行了简单的非真实渲染. 渲染的主要想法是将粒子视为球体, 重建深度纹理, 然后经过平滑后计算法线, 根据法线与视线的夹角计算颜色. 具体实现步骤如下:

1. `_programDepth` 负责用着色器 `depth.frag` 和 `depth.vert` 渲染深度纹理. 在片段着色器中, 计算每个粒子投影到屏幕上的圆形区域内的深度值, 并更新深度缓冲区, 然后将深度作为纹理传入下一步.
2. `_ProgramSmooth` 负责用着色器 `full-screen.vert` 和 `smooth.frag` 对深度纹理进行平滑处理. 在片段着色器中, 使用双边滤波算法对深度纹理进行平滑, 可以做到保留深度边界的同时去除将粒子视作球形时产生的表面凹凸不平的现象.
3. `_ProgramShade` 负责用着色器 `full-screen.vert` 和 `shade.frag` 进行最终渲染. 在片段着色器中, 根据平滑后的深度纹理计算每个像素的法线, 然后根据法线与视线的夹角插值计算颜色, 从而得到流体的非真实渲染效果.

最终的渲染效果如下:



(a) 非真实渲染效果 I (b) 非真实渲染效果 II

图 5: 流体的简单非真实渲染效果图