LAB REPORT: LAB 6

TNM079, MODELING AND ANIMATION

Lucas Hägg
lucha567@student.liu.se

Thursday 13th June, 2024

Abstract

In this lab, an exploration and implementation of fluid simulation techniques were done using the Navier-Stokes equations. The primary objective was to gain experience with a basic fluid simulation system, focusing on the application of external forces and the enforcement of Dirichlet boundary conditions. The projection step was crucial for maintaining the incompressibility of the fluid. Initial results demonstrated the simulation's ability to represent fluid dynamics, even though a challenge for volume conservation were noted.

1 Introduction

The simulation of fluid dynamics is a critical component in the field of computer graphics. This lab was performed to provide experience with a simple fluid simulation system based on the Navier-Stokes equations. The specific purpose of this lab is to understand and implement key steps in fluid simulation, including the application of external forces and the enforcement of boundary conditions to ensure incompressibility. This lab is important because it offers practical insights into the computational techniques used to simulate complex fluid behaviors, which are essential for creating visually accurate animations of fluids such as water, or even smoke and fire. Understanding these techniques not only enhances a programmers knowledge of fluid mechanics but also improves the ability to develop simulations in computer graphics.

2 Background

The focus of this lab was on implementing key aspects of fluid simulation using the Navier-Stokes equations. The tasks implemented include the application of external forces, enforcement of Dirichlet boundary conditions, and the projection step to ensure volume conservation.

2.1 External Forces

The first task involved applying external forces to the fluid. External forces, such as gravity or wind, influence the motion of the fluid. By sampling the external force field at each voxel and using explicit Euler integration, the velocity field of the fluid was updated by:

$$\mathbf{V} = \mathbf{V} + \Delta t \cdot \mathbf{F} \tag{1}$$

where $\mathbf{V}$ is the velocity field. Δt is the time step and $\mathbf{F}$ is the external force field. This method ensures that the forces correctly influence the fluid's motion over time.

2.2 Dirichlet Boundary Conditions

Dirichlet boundary conditions prevent fluid from flowing into solid boundaries. By examining each voxel in the grid and checking its neighbors, velocities that pointed into solid boundaries were set to zero, effectively projecting the velocity onto the boundary plane. This step is essential to maintain realistic interactions between the fluid and solid objects. The mathematical representation for the Dirichlet boundary condition is:

$$\mathbf{V} * \mathbf{n} = 0 \tag{2}$$

where **n** is the normal vector at the boundary. This condition ensures no flow into or out of the boundary surface.

2.3 Projection Step

The most complex task was the projection step. This step ensures that the fluid remains by making the velocity field divergence-free. The projection step involves solving a Poisson equation, which is done using a sparse matrix to represent the discrete Laplacian operator. The conjugate gradient method was used to solve this linear system. This process adjusts the velocity field so that the fluid volume is preserved over time. The mathematical theory behind the projection step is:

1. **Compute the divergence of the velocity field:**

$$\nabla \cdot \mathbf{V} \tag{3}$$

2. **Solve the Poisson equation for pressure p:**

$$\nabla^2 p = \nabla \cdot \mathbf{V}_2 \tag{4}$$

3. **Update the velocity field to make it divergence-free:**

$$\mathbf{V}_{\Delta t} = \mathbf{V}_2 - \nabla p \tag{5}$$

3 Tasks

This chapter goes through how the tasks were implemented during the lab.

3.1 External Forces

The code iterates through each voxel in the 3D grid, checking if it is part of the fluid using the IsFluid function. For each fluid voxel, the coordinates are transformed to world coordinates using TransformGridToWorld. The external force at each voxel is then sampled from the mExternalForces field, and the velocity is updated using equation 1.

3.2 Dirichlet Boundary Conditions

This function ensures that fluid does not flow into solid boundaries by setting the velocity at the boundary to zero. This was done by iterating through each voxel and checking its neighbors. For each fluid voxel, the code checks if any neighboring voxel is solid using the IsSolid() function. If a neighboring voxel is solid and the velocity component points towards the solid, that component is set to zero.

3.3 Projection Step

First, the divergence of the velocity field was calculated and stored in vector b. This was done by iterating through each voxel that is fluid and using central difference:

$$\begin{aligned}\nabla \cdot \mathbf{V}_{i,j,k} = & \frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\Delta x} + \\ & \frac{v_{i,j+1,k} - v_{i,j-1,k}}{2\Delta y} + \\ & \frac{w_{i,j,k+1} - w_{i,j,k-1}}{2\Delta z}\end{aligned} \tag{6}$$

u, v, w are the components of the velocity vector in the x,y and z directions respectively. The components were accessed with the code: VelocityField.GetValue(i, j, k)
Since the grid spacing is uniform, each term was divided by the variable Dx.

Next, the sparse matrix A was created to represent the discrete Laplacian operator. This matrix is used in the Poisson equation 4. To construct A, each element of the matrix corresponded to a voxel in the 3D grid. The central element represented the voxel itself, while the off-diagonal elements represented its neighboring voxels. The values in the matrix were determined by whether the neighboring voxels were fluid, solid, or empty.

- Neighbor checking: Check whether each neighboring voxel were solid using the IsSolid() function. If a neighbor were fluid, it contributes to the corresponding off-diagonal element in A.
- Matrix elements: Each matrix element for neighbor voxels were set to $\frac{1}{\Delta x^2}$ if the neighbor was a fluid voxel. The central element was set to $-\sum \frac{1}{\Delta x^2}$ for each neighbor fluid voxel.

Finally, the conjugate gradient method was used to solve the linear system Ax=b, and the resulting gradient of x was subtracted from the velocity field. The gradient was calculated using central differencing as in equation 6. This step ensured that the velocity field was divergence-free, maintaining the incompressibility of the fluid and preserving the fluid volume over time.

4 Results

The result of the fluid simulation is presented using a simple mesh within a boundary. The mesh is representing the fluid, and the result is presented with pictures during different time stamps/ iterations of the simulation.

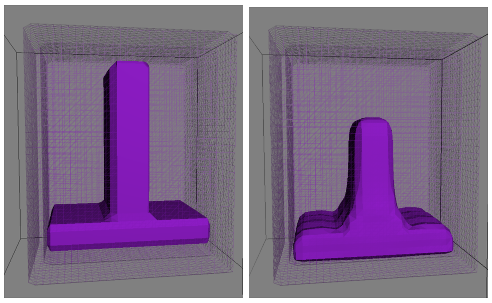
(a) (b)

Figure 1: (a) Start, 0 iterations, (b) 40 iterations

The volume was calculated during each iteration and are presented in the table below.

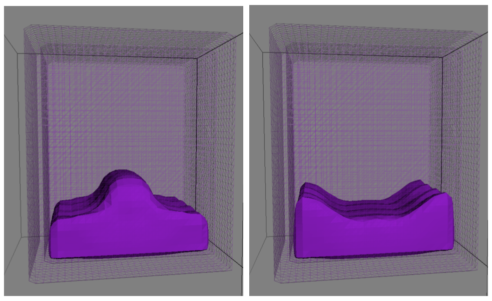
(a) (b)

Figure 2: (a) 60 iterations, (b) 80 iterations

Iterations	Volume
0	0.40140
40	0.38938
60	0.37238
80	0.33101

Table 1: Volume change of the fluid over time

5 Conclusion

This lab provided valuable experience with the implementation of fluid simulation techniques using the Navier-Stokes equations. The result show that simulating fluid was a success. However, it was loosing some volume despite the implementation of the projection step. This was mainly caused by numerical diffusion of sharp features on the level set.

6 Lab partner and grade

Lab partner was Kenton Larsson and the grade I am for is 3.

References

[1] Mark Eric Dieckmann. *MESH DATA STRUCTURES*. Linkoping university, 2023.