LAB REPORT: LAB 6

TNM079, MODELING AND ANIMATION

Julian Voith
julvo429@student.liu.se

Sunday 24th September, 2023

Abstract

This lab report describes the tasks and results from lab 6 of the course TNM079, Modeling and Animation at Linköping University. The lab describes how fluids can be simulated using a stable fluid method and how the overall water-like behavior can be enhanced by the self-advection term. The fluid interface is rendered by using a level-set with the usage of boundary conditions. The method overall produces a useful simulation which however requires a small adaption to be fully realistic.

1 Background

Simulating fluids in computer graphics is not really straight forward. For different types of fluids, different strategies are needed to simulate them properly. In the lab the focus was set to free surface and stable fluids like water by solving the *Navier-Stokes* equations. However, wind and smoke can be simulated with the same techniques discussed in the lab.

The *Navier-Stokes* equations describe how the fluid flow changes over time. The flow is described by the vector field V. The vector field in those equations describe the velocity of the incompressible fluid at every point in space. Hence, it is also called *velocity field*. The velocity field according to Navier-Stokes equations is described by

$$\frac{\partial V}{\partial t} = F + v\nabla^2 V - (V \cdot \nabla)V - \frac{\nabla p}{\rho} \tag{1}$$

$$\nabla \cdot V = 0 \tag{2}$$

where F is the *external force* term, p the pressure field, and ρ the constant density.

In order to solve all those equations, a technique called *operator splitting* is used. It allows to solve the equations in parts one after another in the following order

$$V_0 \xrightarrow{(V\times\nabla)V} V_1 \xrightarrow{F} V_2 \xrightarrow{v\nabla^2 V} V_3 \xrightarrow{\frac{\nabla p}{\rho}, \nabla\times V} V_{\Delta t} \tag{3}$$

where V_0 is the initial velocity field, and $V_{\Delta t}$ is the velocity field at time step Δt.

The lab focused on simulating water, therefore the complexity is reduced a bit further by ignoring the diffusion term, $v\nabla^2 V$. It represents the viscosity of the fluid, and water has a low viscosity. With this simplification the solving order reduces to

$$V_0 \xrightarrow{(V\times\nabla)V} V_1 \xrightarrow{F} V_2 \xrightarrow{\frac{\nabla p}{\rho}, \nabla\times V} V_{\Delta t}. \tag{4}$$

The vector/velocity field is solved on a uniform grid with spacing Δx. All different terms of the simplified version of equation 1 are being laid out in the following paragraphs.

The first term is called *the self-advection term*. It allows the non-linear phenomena like vortices. It can be interpreted as *the motion of the velocity field along itself* and is described by the differential equation

$$\frac{\partial V_1}{\partial t} = -(V_0 \times \nabla)V_0 \tag{5}$$

which can be solved by the *method of characteristics*. This method traces zero-mass particles backwards in time on the velocity field V_0 in order to create V_1. In simpler words, it calculates the position of a particle at Δt time-units ago and feeds it into the new velocity field. The new velocity field is then described by

$$V_1(x) = V_0(T(x, -\Delta t)) \tag{6}$$

where T is the particle trace operator. However, this method relies heavily on interpolation and has a big impact on the quality because the returned position is usually not located on the grid, it has to be interpolated. In this lab the values of V_0 were interpolated by a trilinear interpolation.

The most straight-forward term is the external force term, F. The equation to solve for this term is

$$\frac{\partial V_2}{\partial t} = F \tag{7}$$

which can be solved by the first order Euler time integration:

$$V_2 = V_1 + \Delta t \times F \tag{8}$$

The final step of the Navier-Stokes term is the incompressibility enforcement, $\nabla * V = 0$. This constraint is achieved by applying the projection operator to the vector field V_2. This is done according to the *Helmholtz-Hodge decomposition*, which splits the vector field into a *divergence free* part V_{df} and *curl free* part V_{cf}:

$$V_2 = V_{df} + V_{cf} \tag{9}$$

The divergence-free vector field $V_{\Delta t}$ can be seen as the divergence-free component V_{df} and since gradients are always curl-free, the decomposition 9 can be rewritten as

$$V_2 = V_{\Delta t} + \nabla q \Leftrightarrow V_{\Delta t} = V_2 - \nabla q \tag{10}$$

where q is a scalar field. To assign q, in order to calculate $V_{\Delta t}$, the divergence operator is applied to 10 which leads to the *Poisson equation*

$$\nabla \times V_2 = \nabla^2 q. \tag{11}$$

The divergence operator measures a point in the vector field if it is expanding or contracting and also the magnitude of it. This operator is based on central differencing for a point (i,j,k) which is calculated as

$$\nabla \times V_{i,j,k} = \frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\Delta x} + \frac{v_{i,j+1,k} - v_{i,j-1,k}}{2\Delta y} + \frac{w_{i,j,k+1} - w_{i,j,k-1}}{2\Delta z} \tag{12}$$

where u, $v$and$w$ are the x, y, and z components of the vectors in the vector field V.

The discrete Laplace operator, $\nabla^2 q_{i,j,k}$ can be written in following vector notation

$$\nabla^2 q_{i,j,k} = \frac{1}{\Delta x^2} \begin{bmatrix} 1 & 1 & 1 & -6 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} q_{i+1,j,k} \\ q_{i-1,j,k} \\ q_{i,j+1,k} \\ q_{i,j,k} \\ q_{i,j-1,k} \\ q_{i,j,k+1} \\ q_{i,j,k-1} \end{bmatrix} \tag{13}$$

which describes the material between the voxel (i,j,k) and its neighbors, i.e. is there a solid object or not.

Using these discretization, an equation system

$$Ax = b \tag{14}$$

can be formed from equation 11, where $A = \nabla^2$, $x = q$, and b = $\nabla \times V_2$.

The A matrix is usually a huge matrix with a lot of unknowns. Luckily, it is sparse and contains many zeros. In order to solve the equation system 14, a solver called *conjugate gradient (CG)* until a predefined tolerance ϵ is being used. It exploits the sparse property of the matrix.

Further, for the projection step, boundaries need to be taken into account. In the lab following conditions were used:

$$\text{Dirichlet: } V \times n = 0 \tag{15}$$

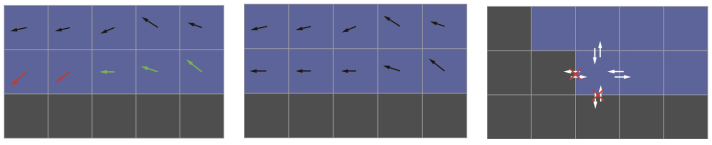
$$\text{Neumann: } \frac{\partial V}{\partial n} = 0 \tag{16}$$

which both specify that there should be no flow in or out of the boundary and no change of flow along the boundary normal n. Information about which voxels are solid, or contain air or fluid is fetched from the scene to allow them to be treated differently.

The Dirichlet boundary is enforced by finding velocity vectors that have a solid objects as a neighbor. Those are being checked whether they point towards the solid object or not. In case the do, the vector are projected onto the tangent plane of the surface.

The Neumann condition is enforced for each voxel, by forcing the difference between solid neighboring voxels, and the current voxel to zero. This is done by modifying the Laplace matrix, A, setting the coefficient corresponding to the solid neighbors to 0 instead of $\frac{1}{\Delta x^2}$ and then adding the number of solid neighbors divided by Δx^2 to the coefficient for the current voxel.

A visualization of these boundaries can be found in figure 1.



(a) Before Dirichlet (b) After Dirichlet (c) Neumann boundary

Figure 1: Visualization of the application of the Dirichlet and Neumann boundaries onto the velocity field. Blue cells = water and grey cells = solid.

An issue of the lab application was, that each time step (mostly when colliding with a solid surface) the fluid looses a significant amount of volume. This is mainly caused by numerical diffusion of sharp features on the level set. A way to mitigate that, is simply adding the lost volume back to the current volume (`mCurrentVolume`. Due to the fact the projection step creates a mass conserving field, an alteration can be implemented which adds the mass back onto the volume. For that, the right hand side of the Poisson equation, $\nabla \times V_2$ can be altered, because it corresponds to sources and sinks which allows us to add an artificial source term which encounters the volume loss. In code terms we compare `mInitialVolume` with `mCurrentVolume` to detect a volume loss. Once this happens following calculation is triggered:

$$b(i,j,k) = (-1) * (\frac{\text{mInitialVolume - mCurrentvolume}}{\frac{\Delta t}{s}}) \tag{17}$$

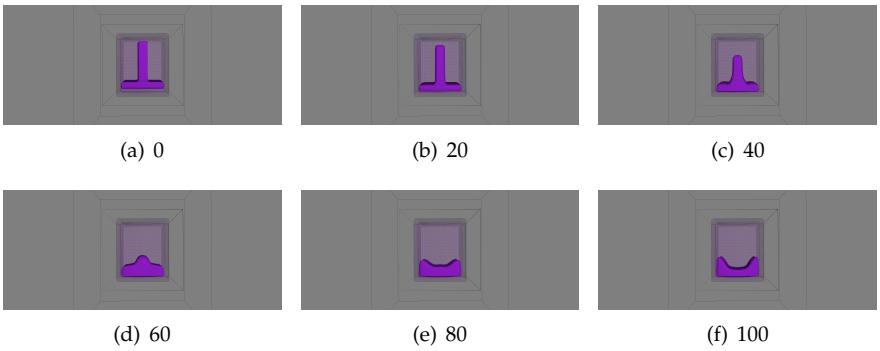
where Δt is the time step and s a scaling parameter. b is indexed by the linear index of the voxels.

2 Results

The results of the lab implementations are presented in following sub chapters.

2.1 Fluid simulation without the self-advection term

In figure 2 the results of the fluid simulation without self-advection can be found.



(a) 0 (b) 20 (c) 40

(d) 60 (e) 80 (f) 100

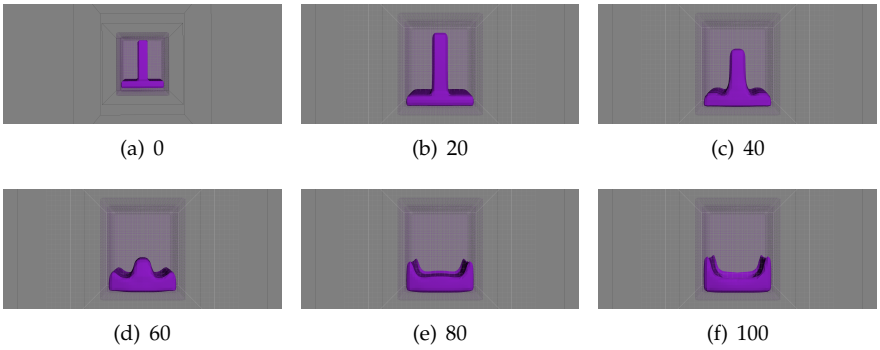
Figure 2: Demonstration of the fluid simulation without the self-advection term after different numbers of iterations

2.2 Fluid simulation with the self-advection term

In figure 3 the results of the fluid simulation with self-advection can be found.

2.3 Improved volume conservation

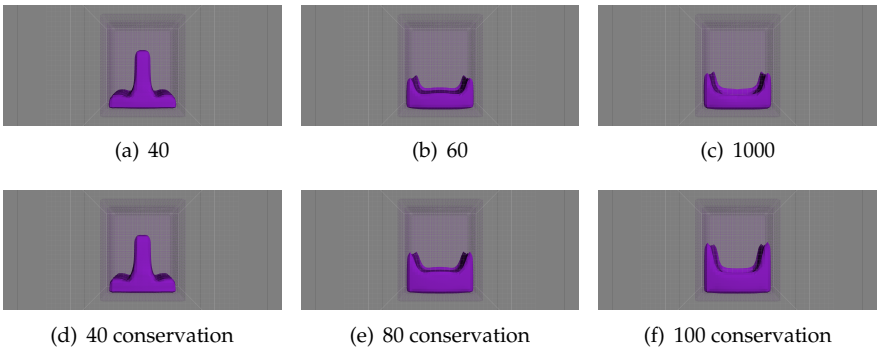
In figure 4 a comparison between the fluid simulation with self-advection without and with improved volume conservation can be found.



(a) 0 (b) 20 (c) 40

(d) 60 (e) 80 (f) 100

Figure 3: Demonstration of the fluid simulation with the self-advection term after different numbers of iterations



(a) 40 (b) 60 (c) 1000

(d) 40 conservation (e) 80 conservation (f) 100 conservation

Figure 4: Comparison between the fluid simulation with the self-advection term with and without improved volume conservation after different numbers of iterations with $s = 10$.

3 Conclusion

The shown results outline the importance of the self-advection term for the behavior of the water-like fluid. It can be seen how the fluid moves within itself and therefor portraits a better/more realistic water-like behaviour.

The simple approach to add back lost volume achieves the intended outcome. The comparison clearly shows that the volume of the fluid was maintained throughout the iterations.

Overall the fluid simulation simulates a water-like fluid with realistic behavior and without the loss of volume

4 Lab partner and grade

The lab and all tasks were solved together with William Toft(wilto938). This report aims for grade 5.