Fluid simulation

Valentin Miu

The project involves fluid simulation, initially based on the methods outlined in Robert Bridson and Matthias Muller-Fischer's 2007 Siggraph course. Liquid fluids, and to a lesser extend gases, can be considered incompressible in most simulations. The behaviour of incompressible fluids are given by the Navier-Stokes equations (1, 2):

$$\frac{\delta \vec{u}}{\delta t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u} \tag{1}$$

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

There are certain fluids, for example water foam or spray, that cannot be adequately described by Navier-Stokes, due to their being compressible.
In the numerical simulation used here, these are simplified by breaking the equation down into three parts: the advection, body, and projection equations, and applying their respective processes `advect()`, `body()` and `project()` in sequence for each time step.
[the three
The data is stored as a discrete 3D grid of cubical cells. Due to the divergence dependency, the simulation uses a staggered MAC grid, which stores the pressure values at the cell centres, and the xyz-components of the velocity $\vec{u}$ (u, v, and w) at the centres of the grid walls (as in Figure 1).

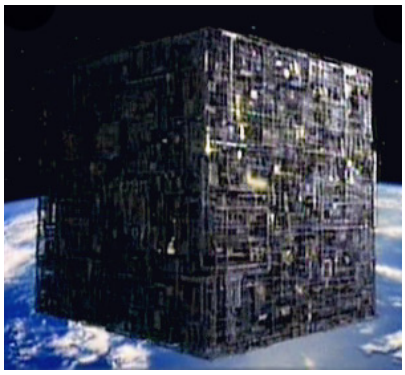

Figure 1: Data storage in one cell of a staggered MAC grid. (PLACEHOLDER)

When applied to the entire fluid, the advection equation enforces the conservation of energy (the fluid mass undergoes no net acceleration without the application of a net force). The `advect()` process applies the velocity field to the quantity q for a time dt. This quantity may be velocity itself, pressure, or another quantity relevant to the particular fluid (such as temperature for smoke simulation).
Since in truth the fluid is acted on by the net force of gravity, the body process simply adds $\vec{g}dt$ to the velocity field (all the members of the grid).

The incompressibility of the fluid is enforced by the `project()` process, by subtracting the correct pressure gradient from the output velocity grid of `body()` (3).

$$\vec{u}^{n+1} = \vec{u}_B^n - \Delta t \frac{1}{\rho} \nabla p \tag{3}$$

In numerical form, this results in a series of n equations, where n is the number of cells in the grid. If $\vec{p}$ is a column vector containing all the pressure unknowns, and $\vec{d}$ is a vector of the corresponding divergence values, the series of equations can be written as

$$Ap = d, \tag{4}$$

where A is a symmetric sparse 3D matrix known as the seven-point Lagrangian matrix. Here, this is solved using the Modified Incomplete Cholesky Conjugate Gradient, Level Zero algorithm (MICCG(0)). This involves using the Preconditioned Conjugate Gradient algorithm (shown in Figure 2),

```
// "·" is the dot product of two vectors; "*" is scalar multiplication
p = 0
r = d
z = applyPreconditioner(r)
s = z
σ = z · r
iterations = 0
while (p not within tolerance) and (iterations < maximumIterations)
    z = A · s
    α = ρ / (z · s)
    p = p + α * s
    r = r - α * z
    // if the element of r with the greatest absolute value is within
    // tolerance, take current p as the solution
    if (max(|r|) ≤ tol) return p
    σ_new = z · r
    β = σ_new / ρ
```

Figure 2: kaejbvksad

with the `applyPreconditioner()` function

Figure 3: kaejbvksad

Seven Point Lagrangian Matrix
MIC(0)

preconditioner

Different types of fluids require specific considerations. For smoke, the body force is replaced with a temperature-dependent buoyancy force. Using the Boussinesq approximation of constant pressure, and assuming a linear dependence on only the temperature and smoke concentration (**??**