**Water Simulation**

**CIS563 Final Project**

**Authored by Leftheris Kaleas**

**Problem Description**

I implemented water simulation as an expansion of the basis for fluid simulation in the second project. In addition I plan to use a third party plug-in to export the output of the fluid simulation into Maya in order to get a nice looking rendering from it and make it look like realistic water.

The overall goal of my project will be to create a realistic simulation of the behavior of water in an enclosed environment, and then to get a decent looking rendering of only the water surface along with standard effects of lighting such as refraction and caustics.

Thus I will be using Level Sets in order to extract the water surface from the water, then rendering using Dongsoo’s Maya Plugin. In addition I plan to get a nice Maya/Mental Ray rendering of the water (caustics, global illumination, etc.).

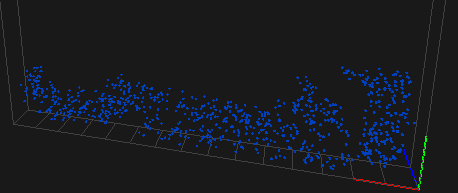
**Approach**

**Stripping Components**

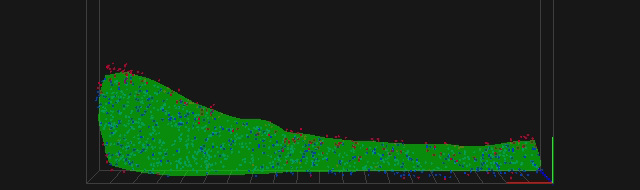
My approach was to start by stripping unnecessary components, which was including buoyancy, density, and temperature, and then to start implementing the other more necessary components of the project. For instance, Marker Particles for the visualization and tracking of the fluid region, as well as RK2 integration for advection and particle evolution. In addition I had to make some large adjustments to the MAC Grid structure in order to account for fluid, air, and solid cell types.

**Marker Particles**

The marker particles were implemented by simple creating a position and a sign (whether the particle was inside the fluid region or outside the fluid region), and then advected using a Runge-Kutta order 2 forward integration scheme:

 Each marker particle was initialized inside of a preset “fluid” region, and then jittered pseudo-randomly around the cell in which it was assigned. I used 32 marker particles per cell in order to get a pretty good distribution of the fluid region. These are rendered out as spheres in OpenGL for previewing.

**Level Sets**

In addition, I implemented level sets in order to keep track of where the water surface is currently. This is implemented as a grid layered on top of the original MACGrid, which the level set distances set discretely at the center of each cell in the grid. We first initialize the level set by setting each grid cell’s value to the nearest marker particle distance as long as it is an “air” cell, and an arbitrary value for fluid cell regions (because we are only concerned with the boundary this tends to be a good estimate for reasonable interior level set φ values). The level set is updated in several steps.

In the first step, the level set is advected using a Semi-Lagrangian approach. However we modify this approach with upwinding differencing in order to get a more accurate estimate of where the level set should be going. Upwinding differencing is a first order integration scheme which uses the current velocity to determine from which direction the gradient of the level set should be computed from.

After this step, because level set advection still has a lot of dissipation (for coarse grids it nearly disappears within several time steps and a lot of movement) we must perform an error correction step using escaped marker particles in order to make sure the level set boundary is accurate. Basically, if a marker particle has escaped the boundary (computed by the locally interpolated level set value), we update the level set by using the closest distance function to the marker particle as we did during initialization. We only need to update local level set cells, because it is most likely that these local cells need their values updated to reflect the correct closest surface distance.

Finally the last step is reinitialization, which is a step to propagate interface information out to surrounding cells, because once advected the level set will not preserve the signed distance function which we have defined on it. We use the Fast Marching method for this, which first finds boundary cells, and propagates this distance information out to surrounding cells. After reinitialization, the surface of the water may have changed a bit, so we include another error correction step as discussed previously in order to make sure the surface boundary roughly corresponds to our marker particle set.

**Velocity Extrapolation**

Once we have the level set, we can use it to extrapolate fluid velocities in the air region surrounding fluid. We will do this by finding the closest point on the surface for each air cell velocity point, and extracting the corresponding velocity from this closest point (I use the level set distance with the level set gradient to get the closest point, then use the interpolated velocity at that point to update the velocity at each cell). This is important for maintaining velocities of particles which are in the process of escaping the level set surface.

**Rendering**

In order to output the level set, I use a Marching Cubes algorithm in order to overlay a another grid on top of the original Level Set grid, and define a triangulated mesh where the interpolated “roots” of the level set are. Thus, I end up with a discretized triangle mesh of the water surface for each time step, which I then output as an OBJ file format. I can load this into Dongsoo’s maya plugin in order to use Maya as a ray tracer in order to get effects like color, environments, lighting, caustics, global illumination, etc.

**Implementation Details**

I used a couple external libraries in order to implement both the fluid simulation. The project is built on the existing project 2 framework, however I use Dongsoo’s Maya Renderer for the final results of the project. In addition I used two specific classes in order to help with level set implementation and level set surface extraction.

They are:

* Fast Marching Algorithm
* Marching Cubes Algorithm

Both are modified and used from the Particle Level Set Library authored by Emud Mokhberi and Petros Faloutsos.

I implemented a marker particle system to track where the fluid region of the discretized grid is at any given point, and in addition at this intermediary step I use these marker particles to visualize the fluid region by drawing each of the particles as a sphere. I also authored velocity extrapolation, and all parts of the level set creation which I have detailed above.

In addition to marker particles, fluid simulation requires other forces such as vortices confinement and external forces in order for the water to act in a realistic way, as well as special methods in order to extrapolate the velocity of the fluid into surrounding air cells so that particles moving into these other cells change their velocities properly as they move around in the grid. I implemented all of these things as well.

In addition I referenced the following papers:

* + Robert Bridson and Matthias Muller-Fischer. 2007. Fluid simulation: SIGGRAPH 2007 course notes
  + Fluid Flow for the Rest of Us: Tutorial of the Marker and Cell Method in Computer Graphics
  + Realistic Animation of Liquids: Nick Foster & Dimitri Metaxas
  + A Particle Level Set Library: Emud Mokberi & Petros Faloutsos

**Implementation Progress**

*First Stage: COMPLETED*

Water using Marker Particles. Rendering in OpenGL.

*Second Stage: COMPLETED*

Water using Level Sets, rendering in Maya using Marching Cubes plug-in.

*Third Stage: INCOMPLETE*

Extra particle system to account for extra cool effects like internal solids and splashing. Marker

Marker particles + Level Sets.

**Running the Project**

**Maya Rendering**

