# Fluid Simulation
# A Post Mortem
# by

**William Ennefors**
wilenn-5@student.ltu.se

## Project Goals

The project that I proposed to do was:

- 3D fluid simulation Using a staggered mac grid
- Pic/Flip method to simulate the liquid
- Run at 60 FPS with at least 1000 particles and 100 cells.

## Project Outcome

The end result is:

- A dynamic staggered mac-grid
- Basic Navier Stokes algorithm to simulate the fluid
- It can currently handle 22500 particles and 750 cells at ~40 fps.

I believe it's fairly correct but it could of course be improved.
There are a few things that bothers me with my outcome., these are:

- The fps is stable but the liquid moves really slow
- Sometimes particles can get stuck on walls causing the container cells to gain near infinite velocities
- The algorithm is not 100% stable, the liquid settles but it takes a while.

I think that actually implementing Pic/Flip and sitting about 1-2 more weeks could yield great results and I'm tempted to actually do so.

## What went well?

**Early Time Plan and Schedule**
The time plan that I set up in the start of the project helped a lot, having small goals at the end of the week did help tremendously. I was able to move forward with a good pace each week.

**Problem solving and Algorithms**
Because the paper described some of the algorithms well I managed to figure out and implement most of the algorithms rather easily, especially the mac-grid generation algorithm.

**Test Driven Development**
By utilizing test driven development I could easily make changes and see what seemed to work and what didn't. I made changes quickly and made progress each day.

## What did not go well?

### The Maths

Due to my limited understanding of vector calculus and differential equations I had to figure out a lot myself, most of the time i had went to trying to understand the equations and it led to a lot of guessing and trial and error, hence the test driven approach.

### Unclear Direction

When it came to the later stages of my development process I lost my direction and basically wandered in circles, I was stuck on a few parts and did not figure out a way to get past them so I just tried everything until it work and it took way longer than expected.

### Too much guessing

What I should've done when I got stuck was to turn to all the different papers and research the equations to get a deeper understanding of how to use them and how exactly they worked, I relied a little too much on a single source which narrowed my views.

## What have I learned?

### Spend more time on pre-development phases

I need to make sure I really understand something before implementing it, The majority of my time was wasted because I implemented calculations incorrectly multiple times and then thought that the problem were somewhere else in the code. Thanks to this I had to rethink and rewrite entire parts of my calculations several times before getting it to work. By making sure to do proper research and test phases this could've been avoided entirely.

### Structure the code better

The code I wrote in the beginning of the project quickly became messy and even sometimes redundant so I had to clean up the code several times. I had to rename variables several times and I should've divided the code into more sub functions to make it even easier to read and get back into.