Chocolate Milk: A Fluid Simulation Framework and Implementation

Daniel Knowlton
dtknowlton@gmail.com
University of Pennsylvania

Norman I. Badler
badler@seas.upenn.edu
University of Pennsylvania

Aline Normoyle
alinen@seas.upenn.edu
University of Pennsylvania

Abstract

In recent years physically based fluid simulations in the computer graphics field have become increasingly realistic both in their visual appearance as well as in their physical characteristics. These methods for realistic fluid simulation are being used throughout both the Animation and Visual Effects industries to create stunning and realistic images of liquids and gases. New methods for modeling fluids such as hybrid particle-grid, FLIP, and particle level set methods have pushed the limits of fluid simulation allowing for even more realistic simulations.

The goal of this project is to build a fluid simulation framework using many of these new techniques in physically based animation. Based on recent papers published at SIGGRAPH, the framework will include some of the state of the art methods in fluid simulation and serve as a foundation that is easily extensible for new methods that may become available in the coming years.

Project Blog: http://chocolatemilkfluids.wordpress.com

1 Introduction

The Animation and Visual Effects industries today rely heavily on the use of realistic physically based effects in order to create truly dynamic and interesting worlds for the cinema. A large problem within the realm of physically based effects is the topic of fluid simulation that includes smoke, water, etc. SIGGRAPH is constantly accepting papers that deal with new methods of fluid simulation to both increase the efficiency of the simulations and to create more and more realistic models of physical phenomena.

With this in mind, the end goal and the main problem that this project will address is making physically based simulations as realistic and efficient as possible. In computer graphics, the challenge is to develop methods to model the physical world, which is continuous, in a digital form. This has led to a variety of SIGGRAPH papers dealing with many techniques for modeling the complex behavior of fluids. A main issue with the increasing number of methods and systems for fluid simulation is that to some degree the methods seem to diverge between grid based methods and particle based methods. In order to get the best of both worlds, a hybrid method is needed that makes use of ideas in both the grid-based and particle-based simulation techniques.

This problem of realistic fluid simulation is growing more important in the animation and visual effects industries as films continue to push the limits of realism and physical effects. With movies such as *Rango* and *Battleship* , Industrial Light & Magic faced huge challenges with the amount of fluid simulation that was required to bring these environments to life on the big screen. In the future, the need for fluid effects will only increase. This is important because it has led to a lot of research into fluid simulation for computer graphics and the next large step for this field is how to combine the many diverse methods to create the most realistic simulation possible.

This project aims to build a fluid simulation framework that combines many of the cutting edge papers. The framework utilizes a semi-lagrangian MAC-Grid fluid solver, but also uses marker particles to define the fluid and help define smaller details of the fluid simulation. It is the goal of the project to implement a few recent SIGGRAPH papers into this framework and create a system that can produce realistic fluid effects.

Upon the completion of the project, the main contributions of the project will include 1) a complete and well documented fluid simulation framework that can be easily extended and added to at later dates, 2) a hybrid particle-grid method for fluid simulation that uses a semi-lagrangian approach as well as fluid marker particles, and 3) implementations of a variety (at least 2) of recent SIGGRAPH papers dealing with fluid simulation. The possible SIGGRAPH implementations include such topics as mass conservation, rigid body interaction with fluids, the FLIP method, and new fluid grid structures such as octrees and the OpenVDB library.

1.1 Design Goals

The target audience for the fluid simulation will at first be the computer graphics community, as the focus early on will be simply implementing the physical effects accurately without focusing on how to integrate these effects into an artistic package such as Autodesk Maya. The main goal of the project is to create a fluid simulation that is as physically accurate as possible by reviewing and implementing a variety of recent SIGGRAPH papers on the topic. The secondary goal is to create a fluid simulation framework that can easily by extended and maintained. The framework will allow for the simple addition of new features and hopefully be able to be a working demonstration of many state of the art techniques in fluid simulation.

Although another goal of the project is to create the framework in such a way that it can easily be rendered out and integrated with a production environment, the emphasis will be places on implementing the physical effects. As such, the framework will support simple OBJ importing and exporting in order to render through systems such as Autodesk Maya. However, if time allows an additional goal of the project would be to create an efficient pipeline between Maya and the fluid simulation to allow artists a streamlined way to create physically based effects in their scenes.

1.2 Project Proposed Features and Functionality

The primary functionality that will be implemented within the fluid framework includes:

- Basic MAC-Grid based simulation with PCGSolver
- Viscosity Implementation
- Methods for mass conservation to avoid volume loss
- Fluid interactions with rigid bodies
- Multiphase fluid grid to model liquid/air interactions
- Better support for interactions between multiple fluids and Particle Level Sets

The following features are secondary goals which will be implemented if there is time:

- Efficient pipeline for integration with the Maya environment
- Octree grid data structure for adaptive grids

2 Related Work

In terms of related work, recent years have seen a dramatic increase in the quantity and quality of fluid simulations and SIGGRAPH papers pertaining to fluid simulations. Production quality physically based simulations such as Naiad and PhysBam as well as software suites such as Realflow and Houdini are constantly being updated with these newest simulation methods. The basis for my project is to implement a few existing SIGGRAPH papers and integrate these effects into a fluid simulation framework based on the foundation described by Robert Bridson [2008]. The following sections detail some of the new techniques that may be implemented as part of the simulation framework.

2.1 Multiple Interacting Liquids

The starting point for the fluid framework will be the "Chocolate Syrup" multiple fluids simulation developed by Dan Knowlton and Yining Karl Li. This simulation was built based on the paper "Multiple Interacting Liquids" by Losasso et al. The main contribution of the paper was to develop a method for tracking multiple fluids on the same simulation grid. The fluids could have different properties (viscosity, density, etc) and thus would behave differently in the grid. The method utilized multiple level sets to keep track of the surface of each of the fluids and then would have an extra step in which all overlap between the different level sets would be solved for. [Losasso et al. 2006] also deals with methods for dealing with attributes such as viscosity and surface tension within the multiple fluid system.

One area that will need to be revisited in the new fluid framework of this project is the idea of Particle Level Sets to help maintain the boundaries between different types of fluids. While the "Chocolate Syrup" base used particles to track the location of the fluid as a whole, particle level sets were not used to track the surface of each level set and ensure the boundaries between fluids remained intact and the fluids remained distinct. In addition, [Enright et al. 2002] proposes a hybrid particle level set method that can be used in this attempt to track the boundary between different types of fluids.

2.2 Mass and Momentum Conservation

Another challenge to be addressed in the fluid simulation framework is the problem of mass conservation as well as momentum conservation. The problem arises due to the fact that through the semi-lagrangian method details of the fluid surface are sometimes lost leading to volume loss and a decrease in mass.

[Lentine et al. 2011a] describes that semi-lagrangian advection as well as the vorticity confinement steps which are very common to fluid simulations do not conserve mass or momentum in their native form. It is important to note that the non-conservative nature of fluid simulations is a result of the need to take discrete time steps and work on a grid rather than a continuous space. Lentine proposes new steps in the advection and vorticity confinement steps of the fluid simulation that ensure that momentum and mass are conserved at each timestep of the simulation. [Lentine et al. 2011b] also proposes a second step to "forward advect" any additional mass that is traditionally missed and unaccounted for in semi-lagrangian simulations. Adding this mass and momentum conservation will be essential to making the fluid framework appear realistic and behave in a physically accurate way.

2.3 Fluid Interactions with Solids

One other area that is important to fluid simulation that this project will explore is the interaction between fluid and rigid bodies within the fluid. This interaction is rather complicated and can be very computationally expensive with traditional pressure projection methods. [Batty et al. 2007] presents a method to deal with the interaction of rigid bodies and fluids greatly reduces the cost of the simulation and allows these complex interactions to be simulated efficiently. In their simulation, by thinking of the pressure projection as a minimization of kinetic energy, they are able to draw a connection between rigid body interactions and fluid interactions. This makes it feasible to simulate such complex situations efficiently.

2.4 Fluid Grid Data Structures

Another area of interest in modern fluid simulations is the actual data structure and representation of the simulation grid. In a standard simulation, the fluid grid remains a constant, uniform size for the simulation. This results in the loss of details in the fluid that are smaller than the grid dimensions. However, some methods exist to help deal with these problems. [Losasso et al. 2004] presented an octree data structure for fluid simulation which is capable of capturing smaller surface details than traditional grids could. They also presented a method to ensure that the standard preconditioned conjugate gradient method could still be used even with the octree structure.

In addition to the octree technique, other data structures and libraries have become available specifically for the storage of volumetric data. One such method, OpenVDB, which was released

last year, is a library that may be worth investigating as an add-on to the fluid simulation framework.

3 Project Proposal

The overall goal of the fluid simulation framework is to produce an extensible and well-documented code base capable of simulating a variety of fluids and the interactions between them. At a high level, the simulation will use a semi-lagrangian MAC-Grid approach and contain many modifications and additions based on recent SIGGRAPH papers. The framework will run as an OpenGL app as well as allow the export of the simulation data for more sophisticated rendering.

3.1 Anticipated Approach

At its core, the simulation framework will be based off of the "Chocolate Syrup" fluid simulator that Dan Knowlton and Yining Karl Li developed for the final project of CIS563 at the University of Pennsylvania. This base framework consists of a semi-lagrangian MAC-Grid structure with the addition of marker particles to mark the location of the fluid on the grid implemented in C++ and OpenGL. The simulation also makes use of the a Preconditioned Conjugate Gradient solver based on the solver proposed by Bridson [2008]. The "Chocolate Syrup" project is in turn based on a viscosity framework and simulation developed by Christopher Batty. This code base will serve as a reference and starting point for the new simulation framework, but the plan is to rewrite the entire source from the ground up in order to create the most successful and extensible framework as possible.

The first step in the creation of the fluid framework will be to revisit the base code of the "Chocolate Syrup" project and rewrite this basic functionality. The functionality to be built and rewritten for the base framework includes a semi-lagrangian MAC-Grid structure with marker particles to represent the fluid. The base framework will include basic viscosity solvers but these will need to be updated. The framework will define a scene file format as well as utilities for importing and exporting scene data for use with rendering and scene setup. The base framework also includes an OpenGL environment to view the simulation as it is running as well as tools to screen capture the simulation as it runs. Finally, the basic framework will allow for multiple types of fluids to be simulated on the same grid although there is much additional work required to finish the implementation of "Multiple Interacting Liquids" [Losasso et al. 2006]. The initial framework will also require the implementation of more complete boundary conditions for fluids' interactions with objects and walls.

A part of the implementation of the framework will be a OpenGL viewer and interaction window that has a working camera and tools for creating fluids or different viscosities/properties, importing OBJ meshes as fluids and rigid bodies, as well a tools to visualize the various fluid simulation components including the velocity field, level sets, pressure, temperature, particles, etc. An important aspect of the viewer that will be included with the framework is that the scene should be easy to manipulate and alter from directly within the OpenGL view to allow for making quick changes to a simulation.

The next step of the project will involve beginning to implement some of the SIGGRAPH papers and integrating them with the framework. The first two challenges that I propose to solve are better mass conservation as well as fluid/rigid body interactions. These papers will most likely have to be adapted to function within the context of the fluid simulation framework that was laid out previously. This is the part in which most of the creativity and new ideas will be formed in that there will most likely not be a one to one mapping between the SIGGRAPH implementation and an implementation that will fit within the framework.

A secondary goal of the project will be to integrate the framework with a new grid data structure to both speed up the fluid simulation as well as give more detail to the simulation. Possible new grid methods will include implementing a fluid octree grid structure or possibly integrating the simulation with the OpenVDB grid library.

3.2 Target Platforms

The target platform will initially be for Unix based x64 machines using the GLUT OpenGL bindings. All integration with renderers will be through Autodesk Maya 2012. A secondary goal is to build the framework so it is completely cross platform and able to be compiled and run through Visual Studio 2010 on Windows as well as Unix machines.

3.3 Evaluation Criteria

The evaluation of the project will involve reviewing the SIGGRAPH papers implemented and visualizing the final simulation results of the fluid simulation. Final results should be as realistic as possible while using the physically based SIGGRAPH papers as guides for implementation and design choices. In addition to the basic fluid framework, the final simulation should include as many new features and physically based algorithms as possible with the minimum standard for implementation being 2 of the SIGGRAPH papers discussed previously. Furthermore the framework for the simulation should be well documented and easily extensible for future projects.

4 Research Timeline

The research timeline is broken down into three main components: the preliminary Alpha Version Report, the Final Project Deliverables, and the Future Tasks of the project. (Please consult the GANT Chart on the final page of the proposal.)

4.1 Project Milestone Report (Alpha Version - Feb. 15)

- Background Reading ([Bridson 2008] and [Losasso et al. 2006])
- Basic fluid framework built and documented without new SIGGRAPH paper implementations
- Start reading SIGGRAPH papers and decide on 2 that will be tackled first

4.2 Project Final Deliverables

- Fluid Framework with at least 2 SIGGRAPH paper implementations
- Demo of fluid functionality in OpenGL environment
- Final renders of fluid simulations
- Final presentation of work accomplished
- Poster for the final project poster session

4.3 Project Future Tasks

- Octree Data Structure/OpenVDB integration
- Better integration with Maya environment

5 Method

The main methodology that I followed was broken down into two main components. The first phase of the project was to create a simulation base on which the rest of the methods and the framework would rely upon. Some of the elements of this phase included handling the basic grid data structures as well as core pieces of the functionality such as the matrix solver and the code for advection and other shared functions.

As a first step in tackling a major fluid simulation project, I thought it would be a good idea to do some review of the basic math and science (and breaks from this science) before I get started doing any major modifications and implementation. My methodology for reviewing Navier-Stokes involved loading up an article, then reading until I needed to look something else up, then opening a new tab for that new topic, and so on. It turns out the rabbit hole is very deep, and I wound up with upwards of 40 tabs of fluid related documents and webpages. I tried to compile some basic Navier Stokes notes not necessarily just for computer simulations, but on the Navier Stokes equations in general.

At this stage, I also began working on completely refactoring and re-architecting the smoke simulation from Physically Based Animation as well as the fluid simulator developed by me and Karl. Some specific items I have focused on in the refactoring process is isolating the OpenGL aspects of the code and separating all GL calls and functions from the core of the fluid simulation. This was a problem with the previous version as the fluid simulation was tightly linked with the GL framework and somewhat hard to extend and organize. Another aspect I was working on in this stage was to isolate the pressure solver so I can experiment with different solvers and be able to plug in any solver that I want without a large change in the codebase. For example, Robert Bridsons example PCG solver available online is one solver that I was looking to test and play with as part of the simulation.

Finally, another aspect of the refactor is just to make the code more consistent and readable. Both the original smoke simulator and the fluid simulator suffered from late nights and lots of competing styles and formatting (probably due to integrating with existing basecode frameworks). I aimed to rewrite the new fluid simulation in a consistent style and a more readable and extensible structure.

While building the core framework, I first developed the smoke simulation, as this simulation relies on the core functonality but at the same time it is rather straight forward to implement and verify its accuracy. While refactoring and rewriting the fluid solver, I discovered a problem with my original smoke implementation that in certain scenarios has pretty drastic effects. It comes down to how interpolation is handled at fluid boundaries. In the previous version, when the point being interpolated was outside of the grid, the value returned for that point (density, temperature, etc) was given a default value. This value in my original framework was 0. This is a source of a *huge* amount of volume loss and value loss. Every time an advection step needs to sample a point that ends up being outside of the grid, this interpolation would end up averaging in a 0 value rather than a value representative of the values in the grid. The solution to this is fairly simple and involves averaging the value from the nearest grid cell rather than a zero when you need to interpolate a density/velocity/temperature value that exists outside of the grid.

In addition to the basic framework, other things that I added pretty early on included a CFL condition to calculate the substep to take in the simulation. The substep is chosen so that it is less than or equal to five times the cell size divided by the max velocity. In essence this ensures that the farthest away an advection uses in a calculation is 5 grid cells away. I also implemented 2nd order Runge Kutta integration for the advection steps. Finally, I reworked the OpenGL viewer for the fluid so that it can be completely isolated from the solver itself. I added a better camera to the GL viewer and have included the previous functionality of screen capture, visualization of the velocity field, viewing of density/temperature, etc. I also implemented a debug view for the GL view that allows the user to inspect certain areas of the grid and check values in that grid cell such as the divergence or the change in density (and later the value of the signed distance field for the level set).

In the next stage I focused on implementing the particle level set method and various methods on top in order to render the fluid simulation with a ray tracer. More precisely, I added the ability for the MACGrid to hold signed distance quantities so that it will be able to store the level set function. I also added a system to initialize the signed distance field that revolves around passing a signed distance function as input and constructing the whole signed distance field for the grid. An example of this is that for a sphere, the function takes in a position and returns the value of: length(position sphere-¿center) sphere-¿radius. Thus, a position inside of the sphere will have a negative value and a position outside of the sphere will have a positive value.

The method for creating the signed distance field can be used to initialize the MACGrids different properties and will eventually be used to initialize the fluid region as well as set the boundary objects. The method for calculating the signed distance on the grid is able to combine multiple functions and create pretty complex signed distance fields by combining smaller signed distance functions. For example in the following two images, the grid is initialized with two different sets of signed distance functions:

The next steps involved implementing a series of papers as support for the particle level set method. One of the key papers that I had to review and implement was the fast sweeping method for eikonal equations. This paper by Hongkau Zhao outlines the method in which the signed distance values can be extrapolated outward from the interface of the fluid and expanded to the rest of the grid. This is an important stage for the particle level set method because in order for the interpolation used in the advection step to remain accurate the signed distance field must remain valid. Since the validity of the signed distance field is not guaranteed after the advection step, a fast sweeping pass must be carried out.

The main idea behind the fast sweep is that you sweep the grid multiple times from different directions and calculate values of the grid based on the lesser magnitude values surrounding it. In effect, this boils down to propagating the signed distance values from the values closest to the surface of the fluid to areas farther from the fluid. This propagation is also very important to the rendering of the signed distance field.

In the simulator that Karl and I produced previously, in order to render the results, we ran marching cubes on the grid to output a mesh representing the fluid. While this was mostly successful there are a couple major issues with this approach. First of all, since the marching cubes algorithm is run at each frame to output a mesh, there is no consistency between the meshes from frame to frame. Each mesh could have entirely different triangle counts and topology. This leads to artifacts such as flickering in the final rendered result. Another issue is the amount of disk space required to store all of the OBJ data files for the liquid. Each frame must produce its own OBJ mesh and for large simulations the amount of mesh data is very very large.

These problems led me to implement raytracing of signed distance fields in my own renderer instead of outputting a mesh for every frame. There are some nice qualities of the signed distance field that help in the raytracing step. First of all, for any point in the 3D grid, you can determine how far away you are from the surface of the liquid. (Each value in the signed distance field represents exactly this distance; negative inside the fluid, and positive outside). This helps with the ray marching through the grid, because instead of taking steps of a constant size, the step size can vary based on the values of the signed distance field. This works because although this value may not be the closest surface in the rays direction, it is guaranteed that the the surface is no closer than this value so you will not overshoot. After sampling the signed distance field, a step of that size is taken and the signed distance field is resampled at this new point. This process is continued until the ray is within one grid cell of the surface in which case simple interpolation is used to calculate the intersection point with the surface.

The second quality of the signed distance field that helps with the raytracing method is that at any point in the grid, the gradient of the signed distance values at that point provide you with the direction of the steepest increase in signed distance values. Thus, when you calculate an intersection point with the liquid through ray marching, you can determine the normal of the surface by taking the gradient of the signed distance field at this point.

As a final stage of the project, and as a jumping off point for future work, I started to implement specific fluid simulation methods such as the particle levelset method and the fluid implicit particle method. In regards to the particle level set method, I developed the functionality to seed particles along the interface to serve as marker particles for the fluid. There are two sets of particles: one representing the inside of the interface and one representing the outside of the interface. The particle level set method works by advecting both the particles and the levelset through the velocity field. At each time step, there is a check to ensure that the levelset matches up to the positions of the particles, i.e. the inner particles are on the interior of the level set and the outer particles remain on the exterior of the levelset. If this is not the case after advection, the levelset is moved to match the particles.

The other simulation method I tackled was the fluid implicit particle method. This method relies on much of the same functionality as the particle in cell method with the major difference of instead of storing the velocities on the grid, the FLIP method stores the fluid velocities in the particles themselved. In order to run the projection step in the simulation, the particle velocities are then projected onto the grid and used for the pressure solve. Once the pressure solve is complete, the updated velocities are translated back onto the particles and used to advect the particles through the grid. The main gain from using this method is that particles retain more of their motion and are not damped by interpolation effects like they are in the particle in cell method.

For my final simulation videos, I actually used a combination of the particle in cell methods and the fluid implicit particle method. The reason for this weighted average between the two methods was that it allows for less interpolation artifacts in the FLIP method but it also limits the amount of noise caused by using FLIP by itself.

6 Results

Below are some of the results images and diagrams from the fluid simulation framework project.

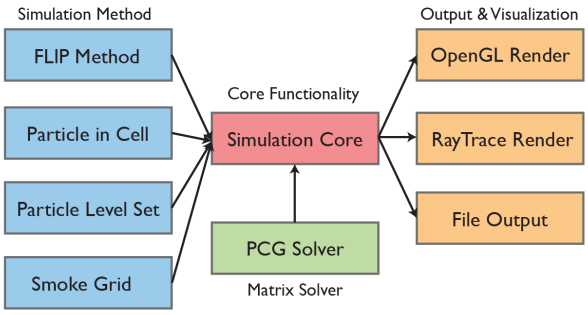

Figure 1: *Diagram of the framework structure with emphasis on independent modules that can be connected together independently of the other modules. The end goal was to make the frameowork as extensible as possible and allow for new modules to be added in the future.*

See more figures near the end of the document.

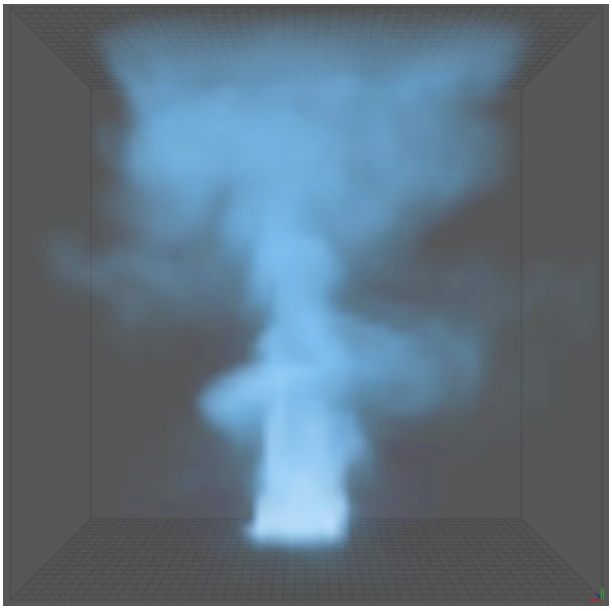

Figure 2: *Basic demonstration of the smoke solver rendered out using the OpenGL renderer.*

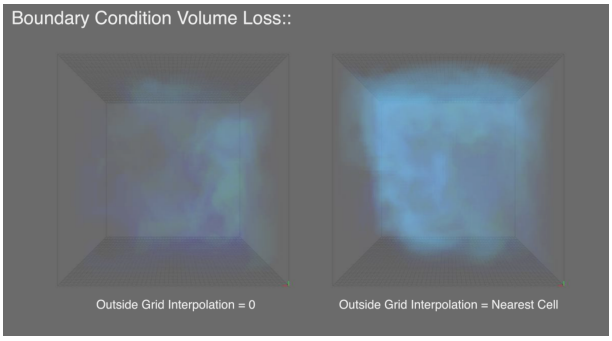

Figure 3: *Volume loss caused by the boundary conditions and the updated boundary condition to preserve smoke volume.*

7 Conclusions

At the end of the senior project period, I believe that I have made good progress towards building a fluid simulation framework. I have been able to build the base implementation and system for connecting different simulation methods as well as a variety of methods on top of the simulation framework. Some of these methods include: the Particle In Cell method, a basic Particle Level Set method, and the Fluid Implicit Particle method.

Although I was unable to complete all of the functionality that I originally set out to complete in my proposal, I was able to tackle a large set of problems in the area of fluid simulation and build a strong base on which to build onto in the future. I think the main reason for this lesser extent of papers implemented was due to the fact that there are so many underlying methods and papers that one must implement before reaching the core of higher level papers. As I started working to develop the system I began to realize the extent of the "rabbit hole". However, I am very happy with the current

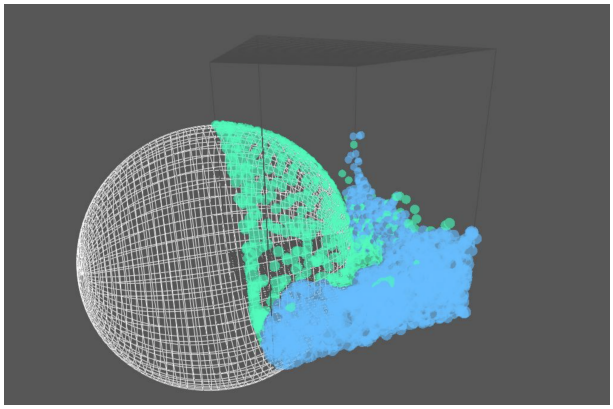

Figure 4: *Basic OpenGL render of the particle in cell method and some complex fluid boundaries.*

status of the simulation, and I again feel like it is a great jumping off point for future work.

8 Future Work

There is much future work that I will consider as I move foward with constructing the simulator. Some of these ideas for future work include different matrix solvers for the pressure solve. The other solvers might include a different form of solver besides the basic PCG solver with Modified Incomplete Cholesky solver or a GPU implementation of the Preconditioned Conjugate Gradient method.

I also want to continue to explore different simulations and develop the existing simulations I have further. At the moment the Particle Level Set implementation is not at the same level of physical realness as the FLIP and Particle in Cell methods. I want to change this so that I can implement other SIGGRAPH papers which are based on the Particle Level Set method. I am also interested in exploring the recent Position Based Fluids approach to fluid simulation.

A long term problem that I also wish to explore in the future is how can I integrate the simulation framework more easily and efficiently with a production pipeline. This will include finding a way to output the simulation data to Maya, how to better raytrace against the fluid grid, and how to render the fluid with other renderers such as Krakatoa.

References

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. In *ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, SIGGRAPH '07.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. A. K. Peters.

ENRIGHT, D., FEDKIW, R., FERZIGER, J., AND MITCHELL, I. 2002. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys. 183*, 1 (Nov.), 83–116.

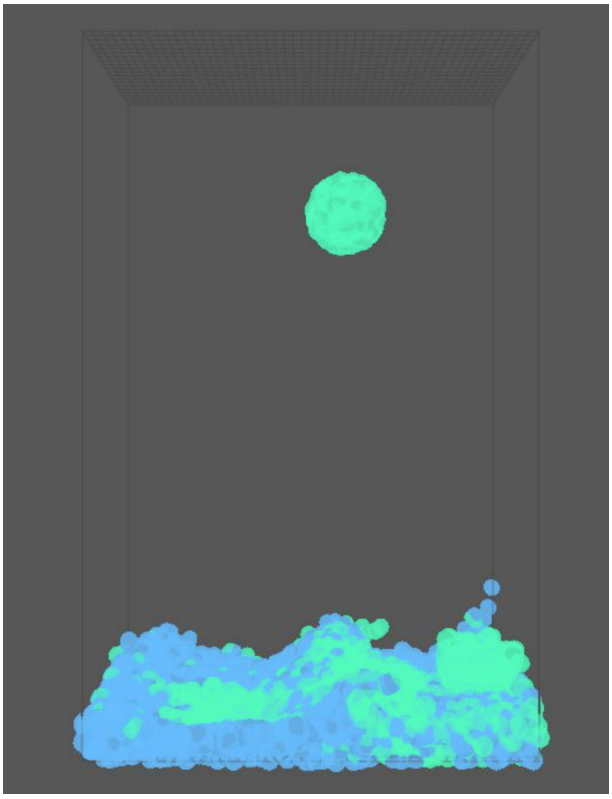

Figure 5: *Dropping raindrops into the fluid using the particle in cell method.*

JAMRISKA, O. 2010. Interactive ray tracing of distance fields.

LENTINE, M., AANJANEYA, M., AND FEDKIW, R. 2011. Mass and momentum conservation for fluid simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '11, 91–100.

LENTINE, M., GRÉTARSSON, J. T., AND FEDKIW, R. 2011. An unconditionally stable fully conservative semi-lagrangian method. *J. Comput. Phys. 230*, 8 (Apr.), 2857–2879.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, SIGGRAPH '04, 457–462.

LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. In *ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, SIGGRAPH '06, 812–819.

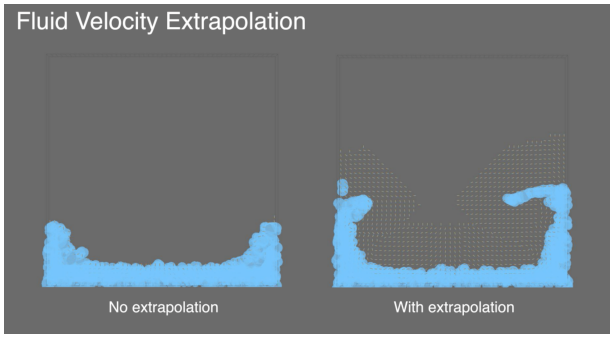

Figure 6: *Difference between simulation with extrapolation and simulation without extrapolation.*

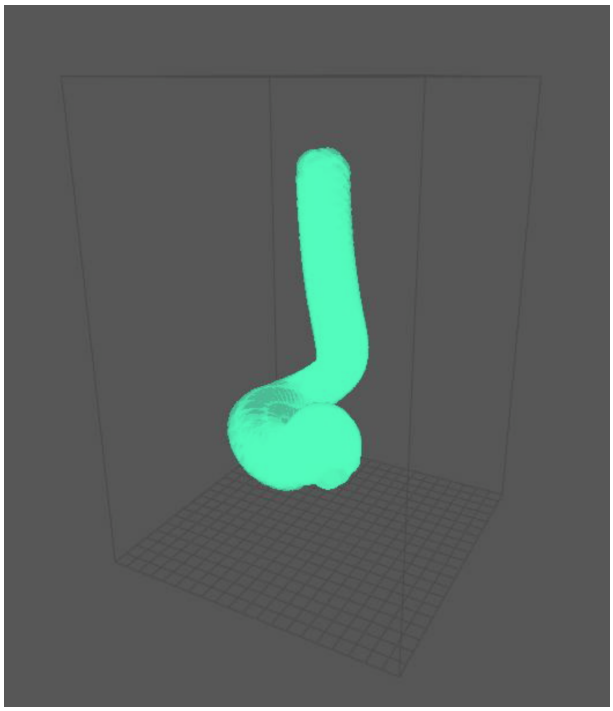

Figure 7: *Viscosity running with the particle level set method simulating a highly viscous fluid.*

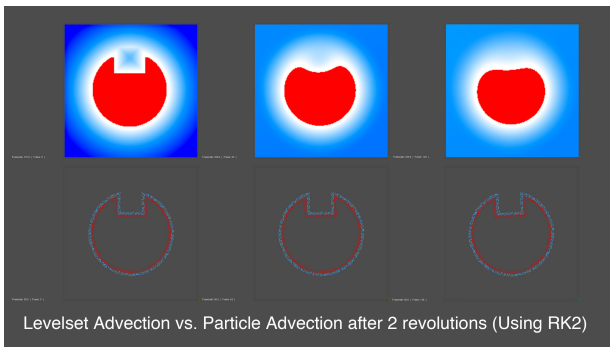

Figure 8: *Particle Level Set base comparing levelset advection to particle advection for preserving surface detail.*

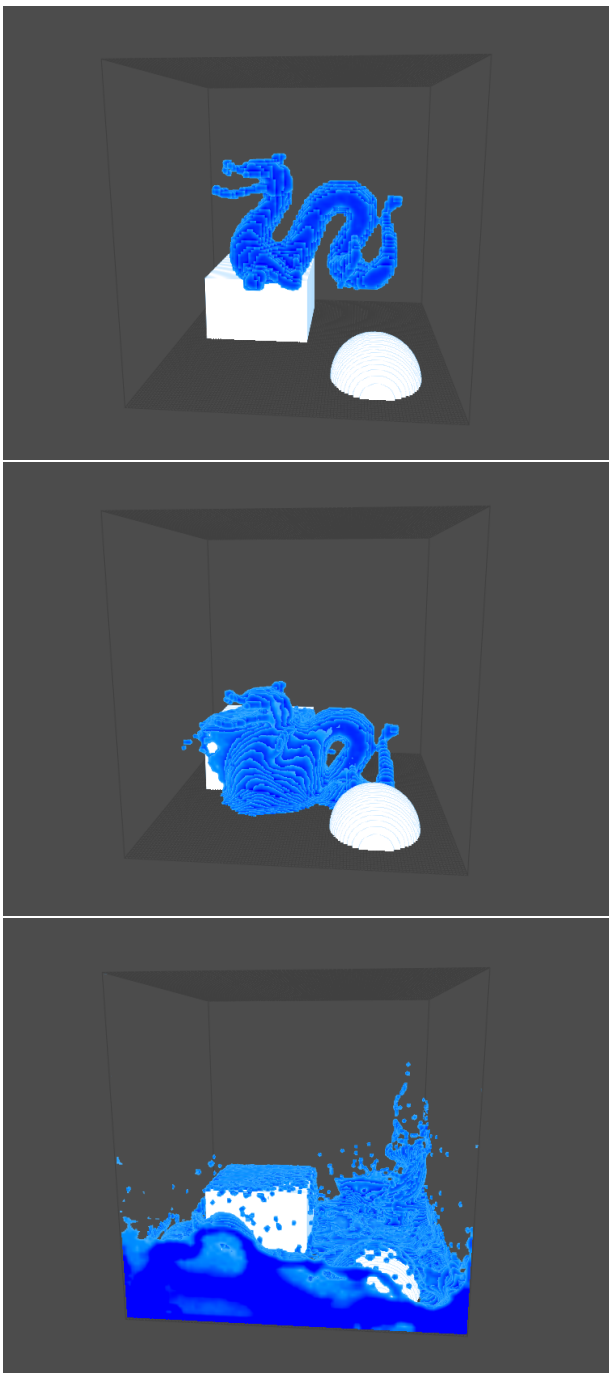

Figure 9: *Combination of FLIP and Particle in cell methods to simulate dropping a fluid dragon.*

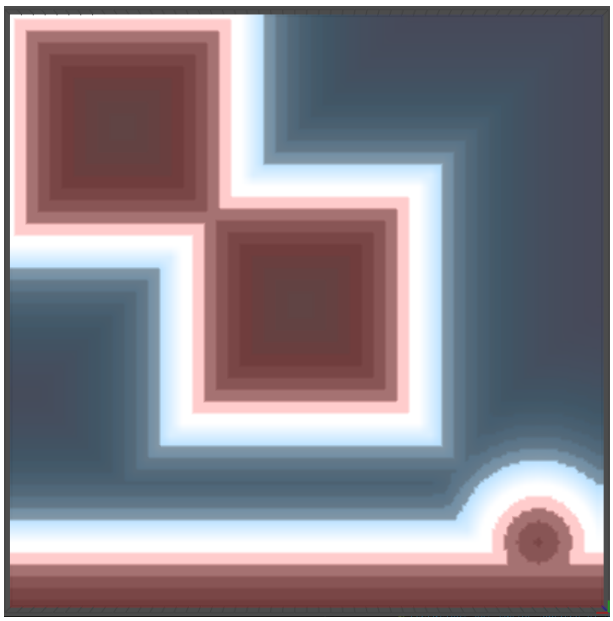

Figure 10: *View of the levelset function defined on a 2D grid.*

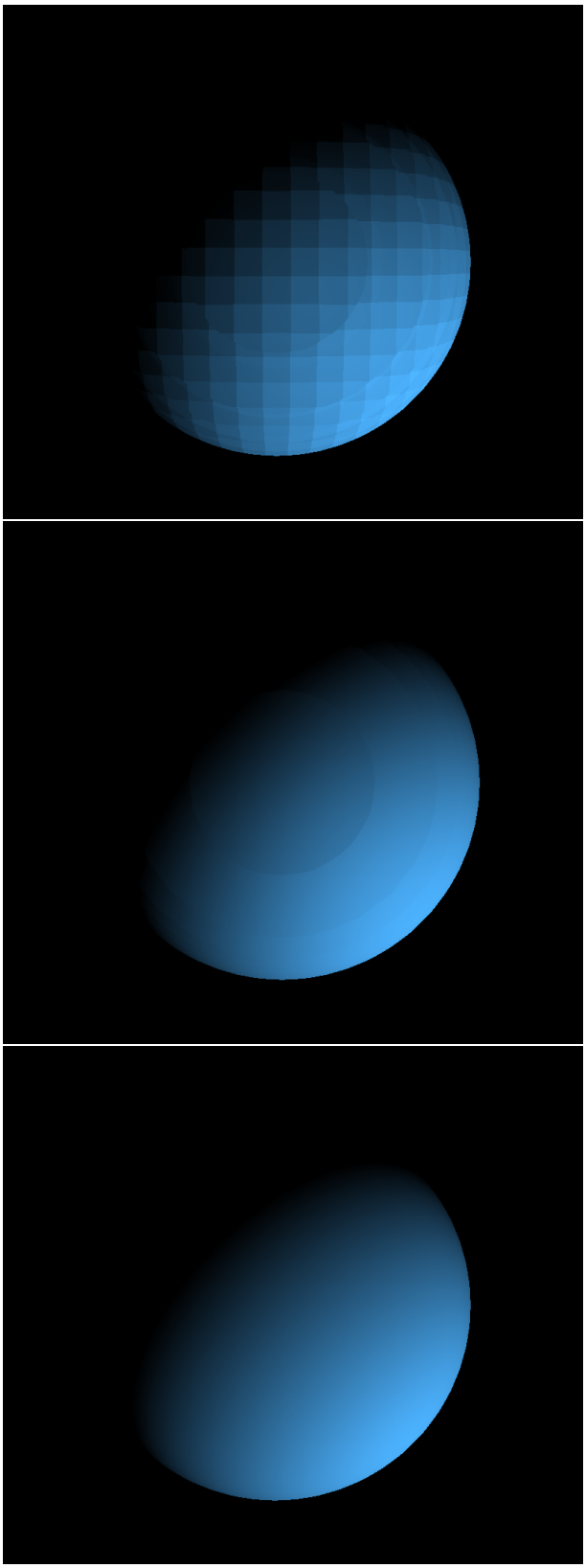

Figure 11: *Comparison of various raytracing of signed distance field methods with the last method using interpolation of multiple gradients.*

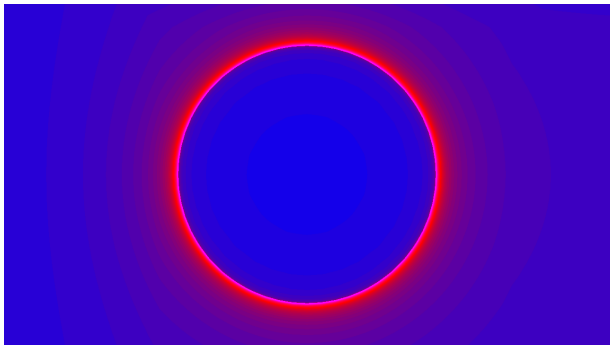

Figure 12: *Number of steps required to raytrace the signed distance field. Red is more steps required.*

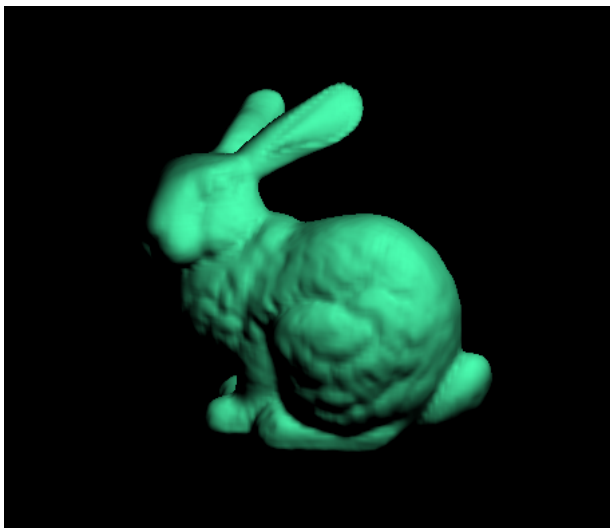

Figure 13: *Signed distance field render of the Stanford bunny.*

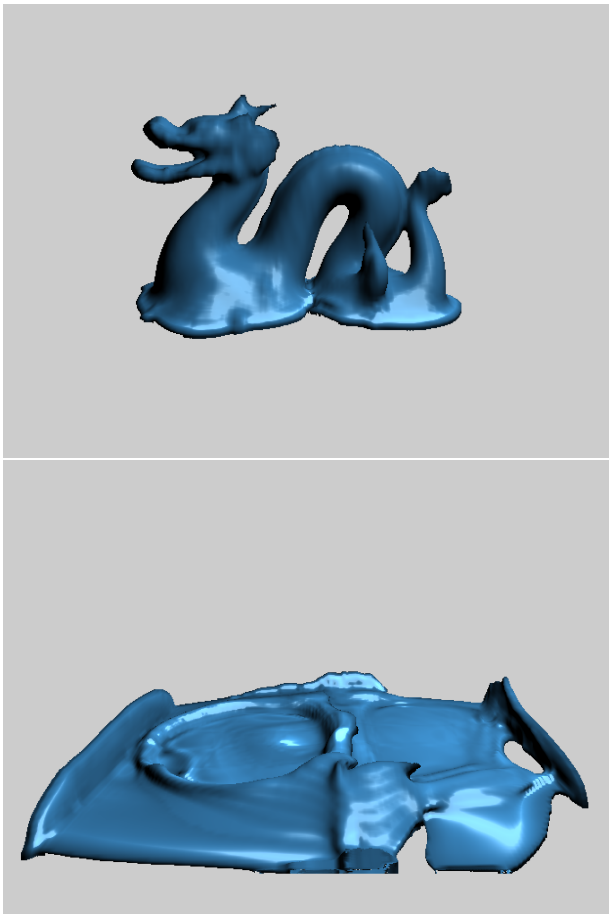

Figure 14: *Signed distance render of a fluid dragon.*

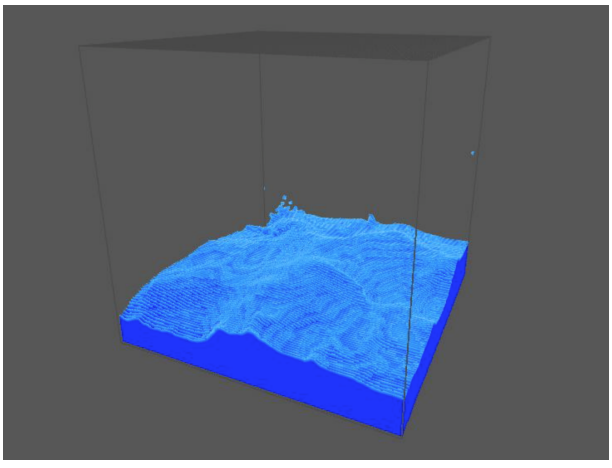

Figure 15: *FLIP and Particle in Cell with OpenGL render.*

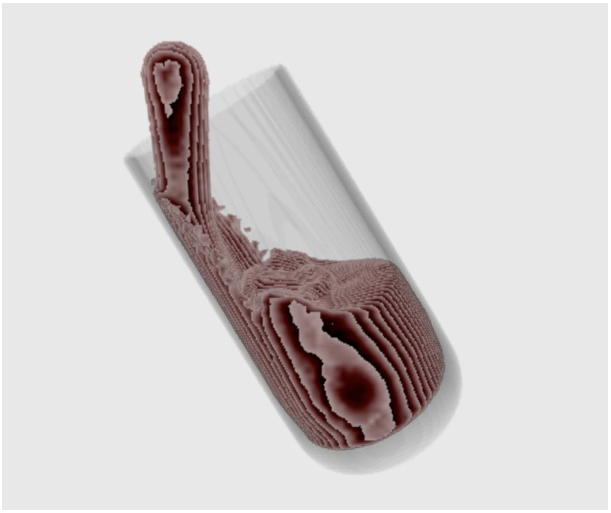

Figure 16: *Chocolate Milk being poured into a glass with FLIP and Particle in Cell*

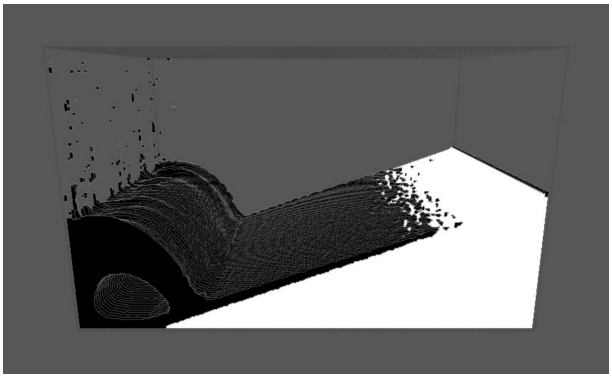

Figure 17: *Wave simulation with FLIP and the Particle in Cell method.*

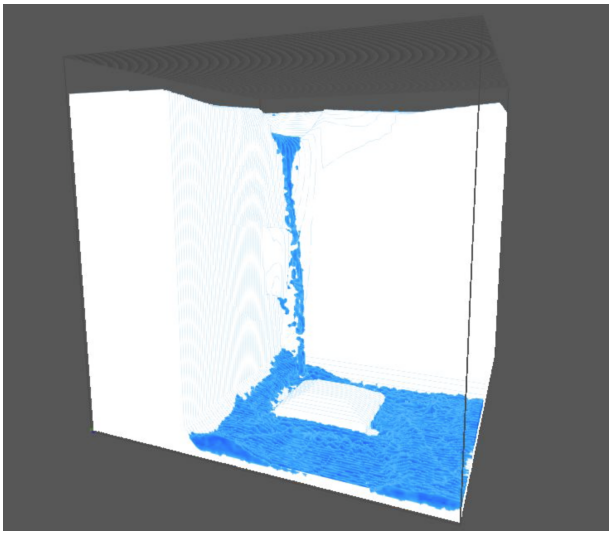

Figure 18: *Waterfall simulation against an OBJ imported from Maya with FLIP and Particle in Cell.*

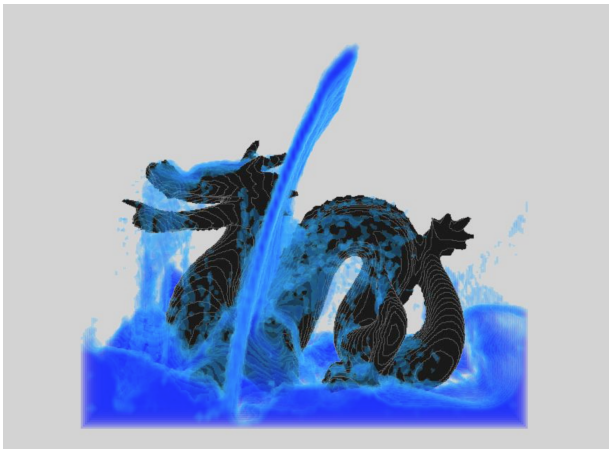

Figure 19: *Giving the Stanford dragon a shower with the FLIP and Particle in Cell method.*

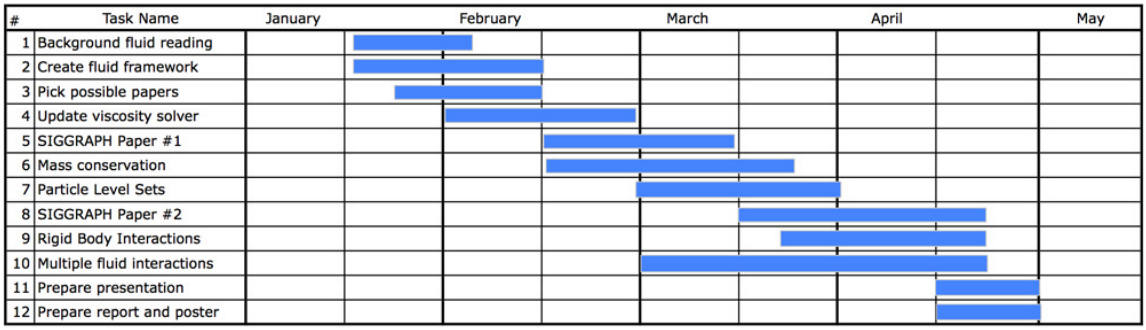

Figure 20: *Gant Chart of projected progress throughout semester*