Smooth Particle Hydrodynamics Solver for Lagrangian Fluid Simulation

Shalin Kiran Banjara

02-December-2011

Master Thesis of MSc Computer Animation and Visual Effects

2010 - 2011

Bournemouth University

NCCA

Abstract

This thesis is a part of Master Dissertation on implementation of a simple smooth particle hydrodynamics solver for Lagrangian fluids simulations. Implementation is shown with a full functioning SPH solver written in C++ and OpenGL using OpenMP framework for parallel processing the simulation iterations and spatial hashing technique for neighbour search. The application also has the facility of exporting data out in geo file format.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Computational Fluid Dynamics (CFD) is defined as “the set of methodologies that enable the computer to provide us with a numerical simulation of fluid flows”(Hirsch , 2007, page 1). With the progress in computer developments, there is a steady advancement in the study of fluid simulation. Application of fluid simulation as incompressible fluid flow is seen in areas such as cosmological study, medical science and applications, colloidal science and even in widely recognized motion pictures. Prime examples of such movies are 2012, Avatar, Posiedon, etc. It is noted that effective use of such fluid simulation results in success of these films and consequently lay the foundations to study them further for appropriate applications.

This master thesis is inspired by the science of incompressible fluid simulation. It is known as incompressible because unlike compressible fluid flow which involves the change in volume of the fluid, incompressible fluid simulation involves no volume dynamics. Instead it would be appropriate to believe that because fluid change volume as per movements to certain extent, it is regarded negligible(Bridson , 2008)There are a numerous models used to understand and evaluate the behavior of fluids. Out of these, highly famous is the Navier-Stokes equation. As per Kelager (2006), for simulating fluids, there are three prime ways namely Lagrangian, Eularian and a hybrid of the two. The aim of this thesis is to show a simple implementation of a particle-based Lagrangian Fluid solver to generate fluid simulation by solving Navier-Stokes equation with the help of Smooth Particle Hydrodynamics method. Fluid simulation consist of two main parts, simulation part and visualisation part. The scope of this thesis is limited mainly to simulation rather than its visualisation part. Thus, the simulation data her is visualised as sphere primitives in OpenGL. Also, an additional functionality of exporting simulation data out has been implemented.

The thesis comprises of a series of chapters which includes an abstract of previous related work on fluid simulation. Proceeding this, emphasis is drawn on the design considerations for the solution along with its implementation. The

later part comprises of simulation results and analysis. The report finally concludes with evaluation of the project with respect to technical and implementation aspects along-with improvements and future works.

Chapter 2

Previous Work

The history of fluid simulation can be traced as early as nineteenth century when Claude Navier and George Stokes in 1822 and 1845 respectfully derived Navier-Stokes Equations to model the fluid flow (Muller et al , 2003). LaterLucy (1977) build Smoothed-Particle Hydrodynamics (SPH) and Gingold and Monaghan (1982)developed it for the simulation of astronomical problems(Hoetzlein and Hollerer , 2009). Moreover in 1983, particle systems for simulating fuzzy objects were pioneered by Reeves . Also, in-order to simulate viscous liquids and melting, Miller and Pearce (1989) invented particle–based methods (Becker and Teschner , 2007). Since SPH adapts well to the simulation of complex and free surfaces, (Monaghan , 1994) was the first to apply free surface flows. Similarly, application of SPH in the simulation of gas and fire phenomena was pioneered by Stam and Fiume (1995). SPH was first applied to animate highly deformed bodies by Desburn and Cani (1996). Later, in 1999, Stam proposed a grid based stable semi-Lagrangian advection method that works very well for real-time simulation of fluids. Application of particle methods for explosive flames was first done by Takeshita et al. (2003). Likewise, Muller et al (2003) and Muller et al (2005) by using Dynamic Air Particles, showcased a real-time implementation of SPH in-order to capture the splashing effects of water. By using Interface Tension Forces, they also presented simulation of multiple fluids. This was done by implementing Teschner et al. (2003) method of Spatial Hashing for optimized neighbour search. Clavet et al. (2005) then demonstrated viscoelastic properties to SPH.

Till now the approaches used to calculate pressure from density was done using ideal gas equation; the latter induces undesired bounciness when applied to liquid as it was primarily designed to work with gas which is highly compressible(Becker and Teschner , 2007). The Moving Particles Semi-implicit method was presented by Premoze et al. (2003) as an effort to solve the high compressibility problem. Techniques to simulate non-Newtonian fluids for melting and flowing were initiated by Carlson et al. (2002) Also, among other work to simulate, was fluid behaviour with functions by Steele et al. (2004). It was Paive et al. (2009) who worked on viscoplastic fluids that along with the change in force, changes viscosity.

Notable references used for this work are Kelager (2006), Bridson (2008), Muller et al. (2003) and Muller et al (2005). Also a number of works on the website of Fedkiw website have been refered while developing this project.

Chapter 3

Technical Background

In this master thesis, Lagrangian particle-based approach has been considered and hence implemented along with Smooth Particle Hydro Dynamics for generating fluid simulation. The proceeding section highlights this approach and how it differs from Eularian approach to solve Navier-Stokes equation of fluid.

3.1 Theory of Fluids for Computer Graphics

Fluids comprises of an integral part of our planet. This ranges from spectacular waterfalls to the air we breathe and from the raging fire of forests to swirling smoke. This, over the years, has become a pivotal part in the study of computer graphics. This study has enabled a broader research area to study the underlined theory of motion required to understand the flow of fluids. A vast amount of the study of fluid flow in animation is calculated by the famous Navier-Stokes equation. (Bridson , 2008)

3.1.1 Navier-Stokes equations

According to Bridson (2008), Navier-Stokes equations are defined as, “a set of partial different equations that are supposed to hold throughout the fluid.”

$$\partial u \partial t + u \cdot \nabla u + 1\rho\nabla p = g + \nu\nabla \cdot \nabla u \tag{3.1}$$

$$\nabla \cdot u = 0 \tag{3.2}$$

where,

u = velocity of the fluid, ρ= density of the fluid, p = pressure, g = gravity, ν = viscosity, $\nabla.\nabla$= differential operator that measures how far a quantity is from the average around it is the Laplacian.

3.1.2 Eulerian and Lagrangian Viewpoints

The Eulerian and Lagrangian viewpoints are the two prime approaches to track fluid motion.(Bridson , 2008)

The Eulerian approach which is named after the famous Swiss mathematician Euler divides fluid space into fixed cells.(Bridson , 2008) It sees how varying attributes of fluid like density, velocity, viscosity, etc. of those fixed cells change with time. (Lamb, 1994) These attributes of fluids are visualised as fields in . Advection is a process through which these field quantities are transported in accordance to time. (Bridson and Muller-Fischer , 2007)

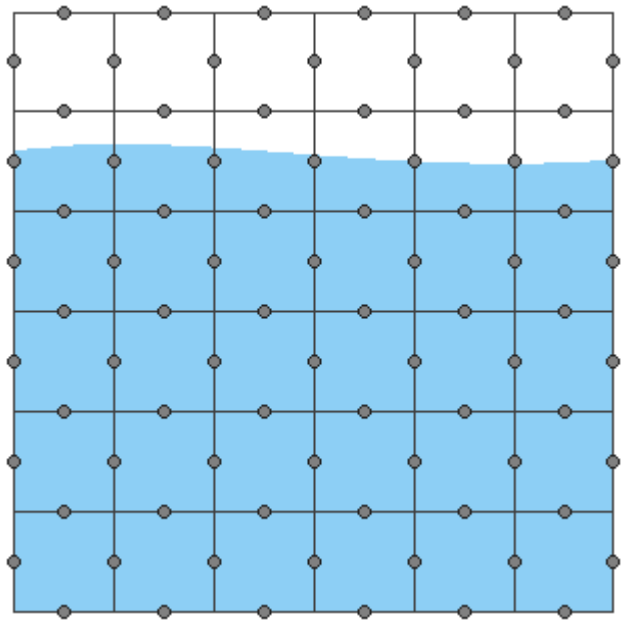

Figure 3.1: *Euler grid-based fluid structure in 2D. The discrete velocity field is represented at the dots.* (Kelager , 2006)

On the other hand, Lagrangian approach is named after a French mathematician Lagrange. In this approach the fluid as a whole is considered a system of

particles wherein each particle can be treated as a molecule of fluid. (Bridson) Apart from containing positions and moving throughout the simulation space, they also contain all the field values necessary for the simulation. (Kelager , 2006)

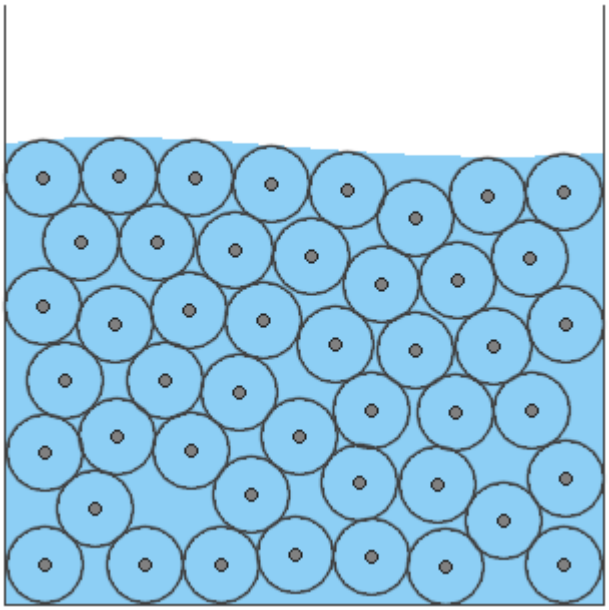

Figure 3.2: *Lagrange particle-based fluid structure in 2D. The particles are represented by the dots. The circles represent the volume of each particle.* (Kelager , 2006)

"In contrast to Eulerian grid-based approaches, the particle-based approach makes mass conservation equations and convection terms dispensable which reduces the complexity of the simulation."(Muller et al. , 2003)

Three fields denote isothermal fluids, namely velocity field v, density field ρ and a pressure field p. Conservation of mass and conservation of momentum are the two equations that signify the evolution of these quantities over the period of time (Muller et al , 2003).

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho v) = 0, \tag{3.3}$$

The main contributing factor for which the above equation can be omitted is that the particle count in the Lagrangian method is constant and that the mass of each of these particles is constant. (Muller et al , 2003) Thus, the Navier-Strokes equation of incompressible fluids is as follows:

$$\rho\left(\frac{\partial \nu}{\partial t} + \nu.\nabla\nu\right) = -\nabla p + \mu\nabla^2\nu + \rho g \tag{3.4}$$

where g is the external gravity force and $\mu$is the viscosity constant.

The substantial derivative $\frac{\partial \nu}{\partial t}$ replaces the convective derivative $\nu.\nabla\nu$ in the above equation. This is because particles in the Langragian method carry the attributes of fluid along with them while moving with the fluid in space.(Muller et al. , 2003) Hence, the Lagrangian Navier-Strokes equation is as follows:

$$\rho\frac{\partial \nu}{\partial t} = -\nabla p + \mu\nabla^2\nu + \rho g \tag{3.5}$$

Newton's second law of motion for fluids is applied to Navier-Strokes equation as three terms $(-\nabla p, \mu\nabla^2\nu, \rho g)$ compute the internal and external forces on the fluid. Thus acceleration and velocities of the particles can be derived by differentiating them. As per Auer (2008), density denotes "mass in a volume", and therefore density ρ replaces mass m.

$$a = \frac{F}{m}$$

$$a = \frac{-\nabla p + \mu\nabla^2\nu + \rho g}{\rho}$$

The acceleration and movement of the fluid particles is a resultant of internal and external force on the fluid computed by these tree terms.

In the above equation, $-\nabla p$ is the pressure force, $\mu\nabla^2\nu$ is the viscosity force and ρg is the external forces. The pressure force and viscosity force are the internal forces produced by the fluid itself while the last term represents all the external forces like gravity force.(Auer , 2008)

With the help of pressure force, the fluid molecules are held together. In absence of this force, the fluid would collapse on hitting the ground. As stated by Auer (2008) pressure force is created due to pressure difference and makes the fluid flow to low from high pressure areas due to negative pressure gradient. Sustaining equal density of material throughout the fluid is the prime aim of pressure force. (Pelfrey , 2010)

Viscosity force is the force of internal resistance of a fluid to flow. It can also be referred as a measure of fluid friction. This force determines the viscosity of the fluid as in the case of water against magma. The term$\nabla^2\nu$ is Laplacian of the velocity field of the fluid and aims to smoothen the velocity difference over time. This term provides the deviation of the latter from its average value. μ is the viscosity coefficient defining the intensity of the force(Auer , 2008).

The last term defines all the external forces. These forces can be user generated interactions or surface tension to natural forces like gravity or wind. These external forces like surface tension force will be further discussed in-depth later.

3.2 Smoothed-Particle Hydrodynamics

One of the first person to use smooth particle hydrodynamics for solving Navier Stokes equation was Monaghan (1992).

Smooth Particle Hydrodynamics(SPH) is defined as "an interpolation method to approximate values and derivatives of continuous field quantities by using discrete sample points. The sample points are identified as smoothed particles that carry concrete entities." (Kelager , 2006, page 8)

Representation of the continuous field of fluid is not possible because of the finite nature of Lagrangian particles. The state of field at any position within the fluid space can be evaluated with SPH interpolation method by using these particles as discrete samples. Herein, it is presumed that a fraction of the problem space is occupied by every single particle. (Kelager , 2006)

Radial symmetrical smoothing kernels are used to do the interpolation in such a manner that the weighted sum of contribution from neighbour particles will return the approximation of a quantity A at some particle position r (Muller et al , 2003). The equation is as follows:

$$A(r) = \sum_j m_j \frac{A_j}{\rho_j} W(r - r_j, h) \tag{3.6}$$

The number of particles here are determined by the core radius h. These particles then become a factor for the approximation value at -*9r. Particles at a greater distance then h are not considered for the calculation.

As the smoothing kernel function is exclusively affected by the SPH fluid equation, there arises a need for calculating quantities' derivatives.

$$\nabla A(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(r - r_j, h) \tag{3.7}$$

$$\nabla^2 A(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(r - r_j, h) \tag{3.8}$$

However, implementation of SPH in solving the internal fluid forces of Navier-Stokes equation results in the rise of non-symmetric forces resulting in instabilities in the simulation. Numerous methods have been developed to counter-act these instabilities in simulation. This is discussed in the later sections. Visually pleasing simulation of liquids generated using SPH requires to handle parameters with utmost care as it suffers compressibility hitch. Compressibility aspect still an area of active research.

3.3 Smoothing Kernel

Smoothing Kernel play an important role in smooth particle hydrodynamics calculations as the stability and accuracy of these calculations are highly dependent on them. It also effects the speed of the simulation as these calculations are made for each time-step. Three kernel proposed by (Muller et al , 2003) are as follows:

- Poly6 Kernel

As r appears in most of the calculations and is only squared, it makes the calculation faster to compute.(Auer , 2008)

$$W_{poly6}(r,h) = \begin{cases} \frac{315}{64\pi h^9}\left(h^2 - r^2\right)^3 & 0 \leq r \leq h \\ 0 & otherwise \end{cases} \tag{3.9}$$

with $r = ||r||$

$$\nabla W_{poly6}(r,h) = -r\frac{945}{32\pi h^9}\left(h^2 - r^2\right)^2 \tag{3.10}$$

$$\nabla^2 W_{poly6}(r,h) = \frac{945}{32\pi h^9}\left(h^2 - r^2\right)\left(3h^2 - 7r^2\right) \tag{3.11}$$

- Spiky Kernel

For the usage of pressure calculation, this gradient of the kernel is used. This is because the repulsive force between the particles that are too close to each other nullify due to the gradient of the Poly 6 kernel as its value goes to zero near the centre thereby resulting in clumping. (Auer , 2008)

$$\nabla W_{spiky}(r,h) = -\frac{45}{\pi h^6}\frac{r}{||r||}(h-r)^2 \tag{3.12}$$

- Viscosity Kernel

The viscosity kernel is used instead of the Laplacian of the Poly6 kernel, because Poly6 kernel provides negative value resulting in high-energy viscosity forces which in turn causes high speed particles to accelerate instead of slowing down. Unlike Poly6 kernel, Viscosity kernel always gives positive Laplacian

to make the viscosity force act as a damping force in all cases and help to reduce the velocity of the high-energy particles. (Kelager , 2006)

$$\nabla^2 W_{poly6}(r,h) = \frac{45}{\pi h^6}(h-r) \tag{3.13}$$

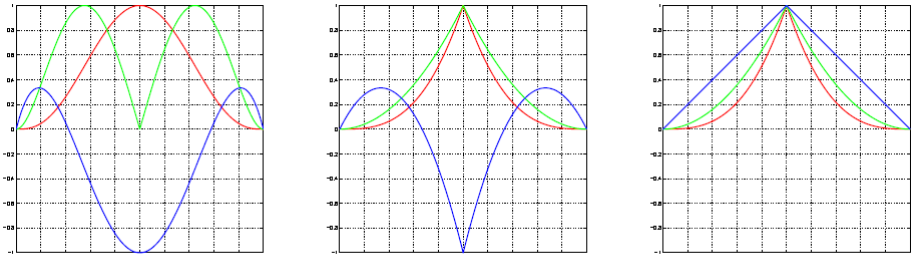

Figure 3.3: *Kernels used in the SPH method. (Red) Kernel value (Green) Gradient of kernel (Blue) Laplacian of kernel. Note that the kernels and the individual curves are scaled differently in these plots.*(Angst , 2007)

Chapter 4

Design

This chapter provides a brief overview of the scope of the project, design consideration and pipeline. These forms the basis of final implementation.

4.1 Objective

The aim of the project is to implement a basic 3D particle-based Lagrangain fluid solver using smooth particle hydrodynamics to simulate the flow of liquids. Simulation of these liquids will be based on the forces of the Navier stokes equation along with some additional forces. Following is the list of objectives to be achieved by the fluidsolver.

- Simulating the flow of liquid like water, milk, oil, honey etc.
- Realtime visualisation of the of the fluid particles.
- Interactions with the boundary container like box and rigid bodies like sphere.
- Simulation of interaction of multiple fluids.
- Exporting simulation data out in a readable format to 3D Packages like Houdini.

4.2 Class Diagram

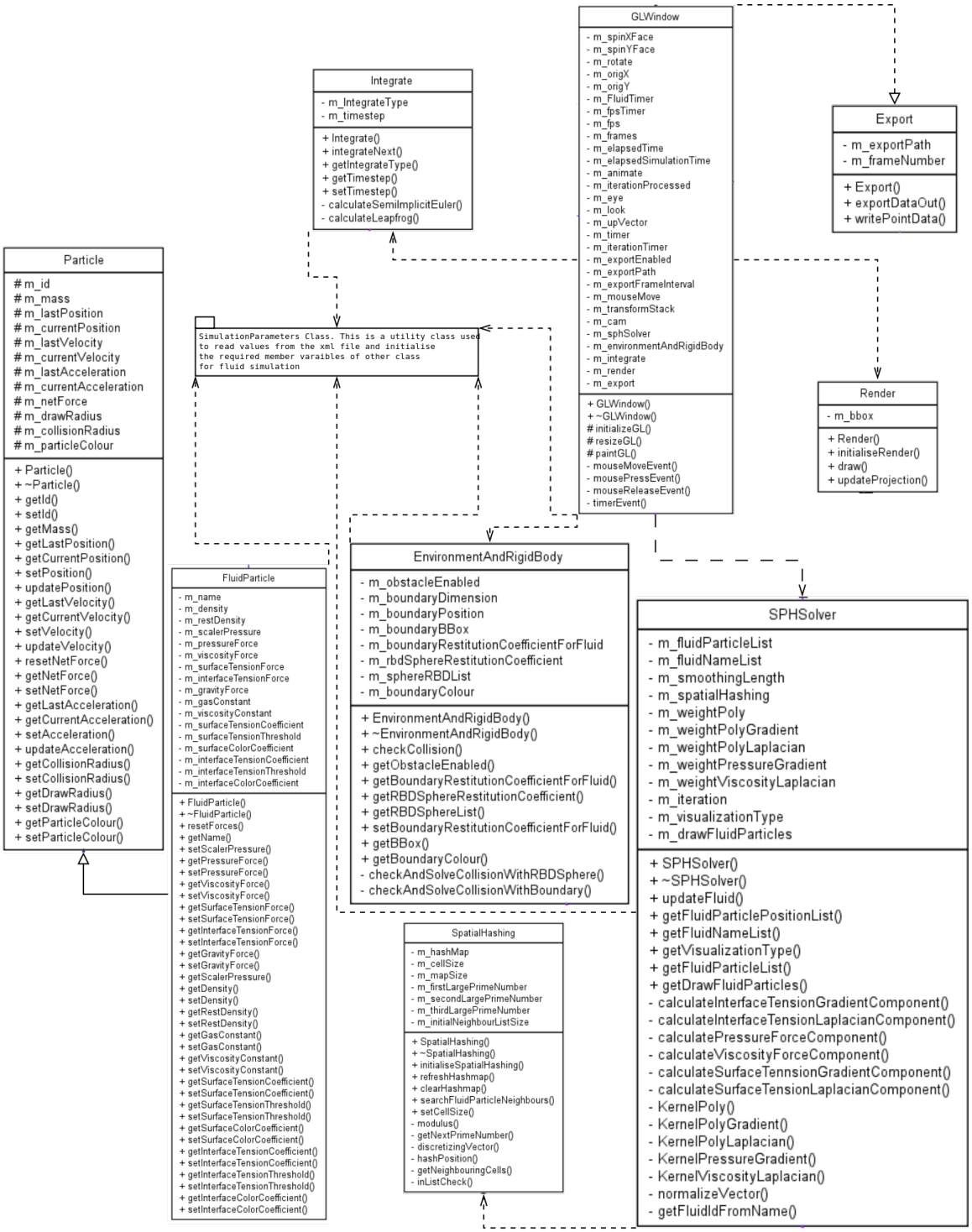

Figure 4.1:

4.3 Pipeline

A fluid consists of large number of particles. Due to this, it is a challenging undertaking to layer and calculate their preliminary positions manually. External models are used to create fluid particles at the initial positions or their vertices.(Priscott , 2010)

Creating fluids of varying shapes and resolutions becomes easy with the help of this method. The simulation data can be visualised primarily as spheres primitives. Option of exporting simulation data as geo is also included in the application.

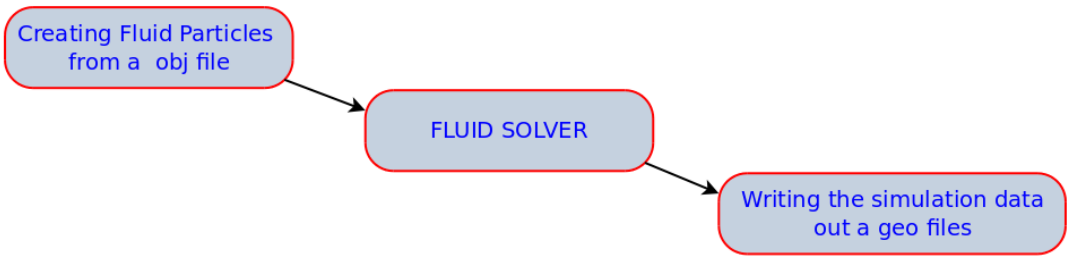

Figure 4.2: Pipeline

Chapter 5

Implementation

The implementation of this project is dependent upon NCCA NGL's Library 4.0 , OpenGL 3.2, Qt. The NCCA NGL's Library is dependent upon Glew and Boost 1.44.(Macey , 2010) The project has a xml configuration file to read in all the input parameters for initialising fluid simulation and scenario developments.

5.1 Algorithm

A much simple and linear algorithm is used here for generating fluid simulation for each iteration. At each iterations, calculations for the density and pressure value for all the fluid particles are done based on which, then various forces such as viscosity, pressure, surface tension, interface tension and gravity are calculated. Once the net force is know, calculation for the acceleration is done and then acceleration is integrated to get the next velocity and position. Finally, before displacing the fluid particle, a collision detection and resolution is done.

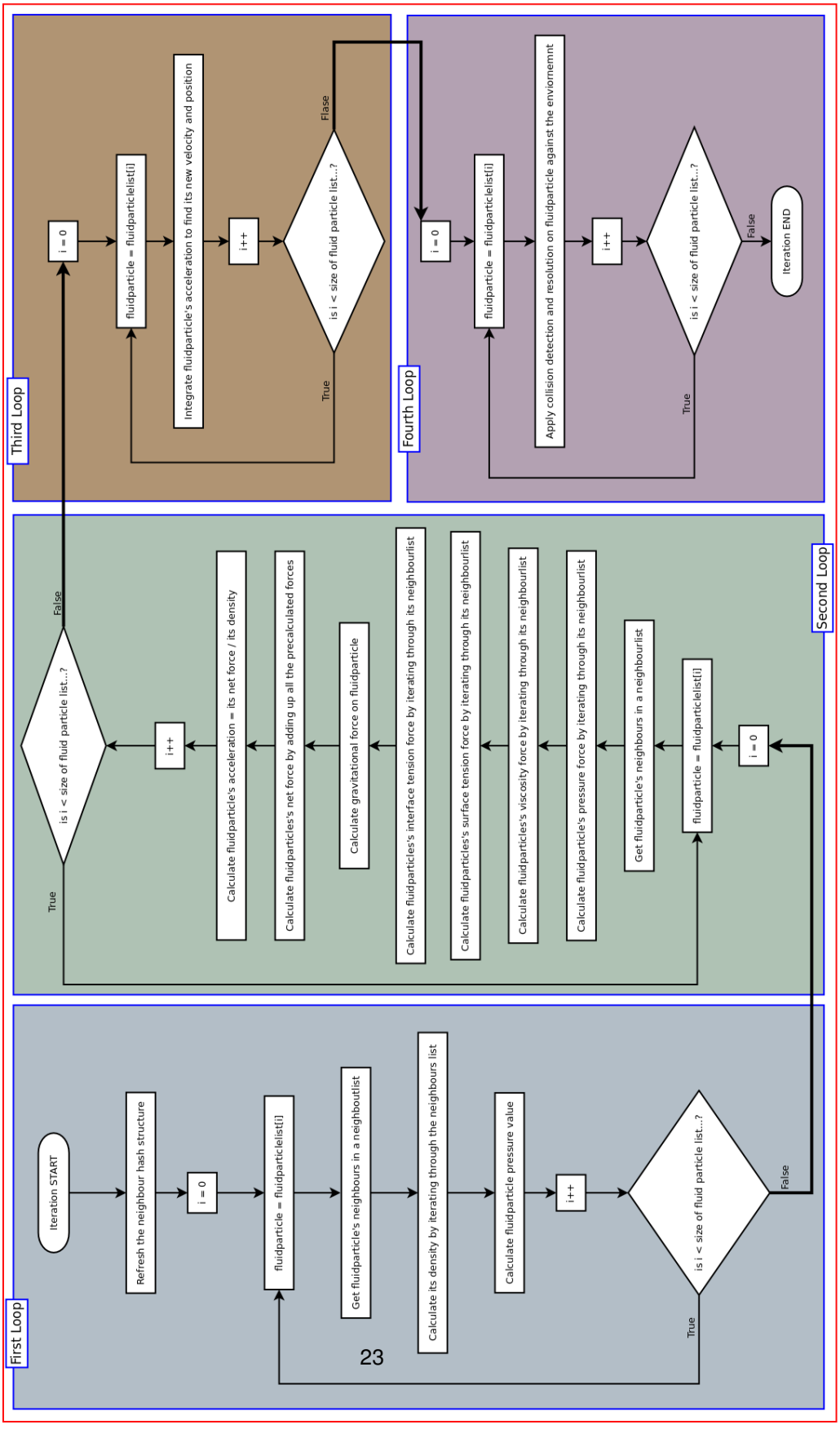

Figure 5.1: Flowchart for a Fluid solver at a single time-step

Once all the calculations are done depending upon the visualization flag set to true from the setting.xml file, the program will render the fluids as spheres in OpenGL window or skip this step it is set to false. Also depending upon the export file flag value in the setting.xml file, if set to true will export simulation data out as geo files else if set to false will skip this step.

5.2 Fluid Generation

The solver application creates the fluids by reading an external obj file using NGL Graphics Library.(Muller et al. , 2003) The external model for fluid creation is created in Houdini using a poly sphere or poly cube. An additional pointfromvolume sop is used to fill the primitives with points. Its then saved as a obj file and the path is passed to the fluid solver application. The following Figure 5.1 shows the generation of model in Houdini.

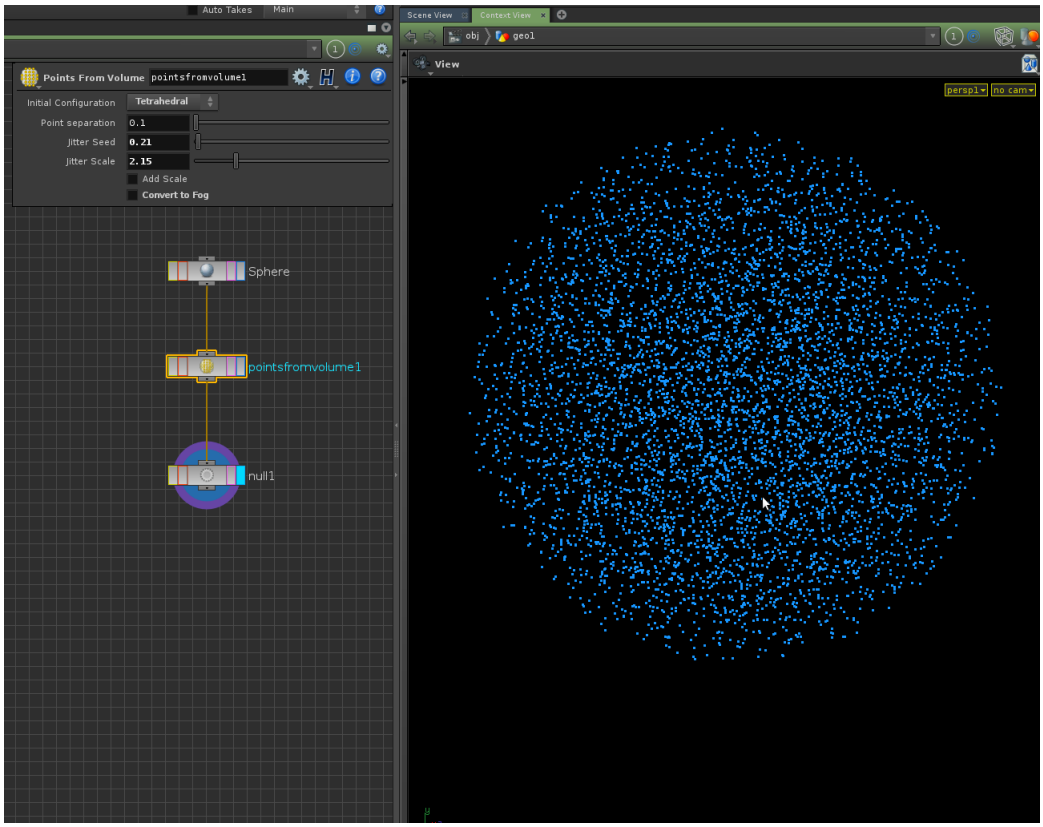

Figure 5.2: Model Creation for Fluid using Houdini

The SPHFluidSolver Application generates a fluid particle for each point by loading and reading the point information of the obj file. Mass of the particle is fixed at the creation and is calculated by the following formula (Kelager , 2006)

$$mass = density * \frac{volume}{particlecount} \tag{5.1}$$

In the program the fluid is generated by reading the obj file. The process is as follows:

- Calculate total number of vertices of the obj file.
- Created fluid particles at each vertices and assigning them a fixed mass calculated the above given 5.1. The difference is here our particle count in the total number of vertices and density is the rest density assigned.

5.3 Forces Calculations

5.3.1 Density Calculation

Density at position r is calculated using the following formula (Kelager , 2006)

$$\rho_i = \sum_j \rho_j \frac{m_j}{\rho_j} W\left(r_i - r_j, h\right)$$

$$= \sum_j m_j W\left(r_i - r_j, h\right) \tag{5.2}$$

5.3.2 Pressure Per Particle Calculation

For calculating the pressure term of term of the Navier stokes equation, the pressure p_i for each fluid particle should be know before calculating the force itself. Thus while calculating the new densities values for each fluid particles, its new pressure is also calculated. The following modified version of the ideal gas state equation (Desbrun and Gascuel , 1996)is being used in the application to calculate the pressure value.

$$p = k\left(\rho - \rho_0\right) \tag{5.3}$$

In the 5.3 k is the gas constant and $\rho_0$is the rest density of the material.

The equation returns positive and negative value which results in generating attractive and repulse force depending on the density and rest density value.

5.3.3 Pressure Force Calculation

Pressure force is calculated using the SPH and Poly6 smoothing kernel. The non-symmetrical force issue is solved by the following equation and is also implemented in this project.(Kelager , 2006)

$$f_i^{pressure} = \rho_i \sum_{j \neq i} \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_i^2} \right) m_j \nabla W \left(r_i - r_j, h \right) \tag{5.4}$$

5.3.4 Viscosity Force Calculation

Viscosity force provides stability to the simulation and also acts as a damping force for the fluid particles. Non-symmetrical force also arises in the system as SPH is used. A solution to this was provided by (Muller et al , 2003). As stated in the paper, velocity difference is used instead of absolute value to solve the issue. The equation is as follows:

$$f_i^{viscosity} = \mu \sum_{j \neq i} \left(v_j - v_i \right) \frac{m_j}{\rho_j} \nabla^2 W \left(r_i - r_j, h \right) \tag{5.5}$$

5.3.5 Surface Tension Force Calculation

Surface Tension force is the first external force that is not from Navier -Stokes equation which has been implemented in this program. It acts on the particles at the surface of the fluid.

Equal intramollecular attraction equal in all direction exist on the molecules inside a fluid due to which they are held in perfect balance. But this is not the same for the particles at the fluid surface. This imbalance arises a surface tension force. The particles at the surface acts in an inward direction along the surface normals with the aim of reducing the fluid surface curvature. This smooths the fluid surface.(Muller et al , 2003)

The following formulaes have been adapted from the paper of (Kelager , 2006). The curvature of the fluid surface is identified using the colour Field c. Each fluid particle is assigned a colour quantity of 1 while the air is assigned a value of 0. Calculations using SPH are as follows:

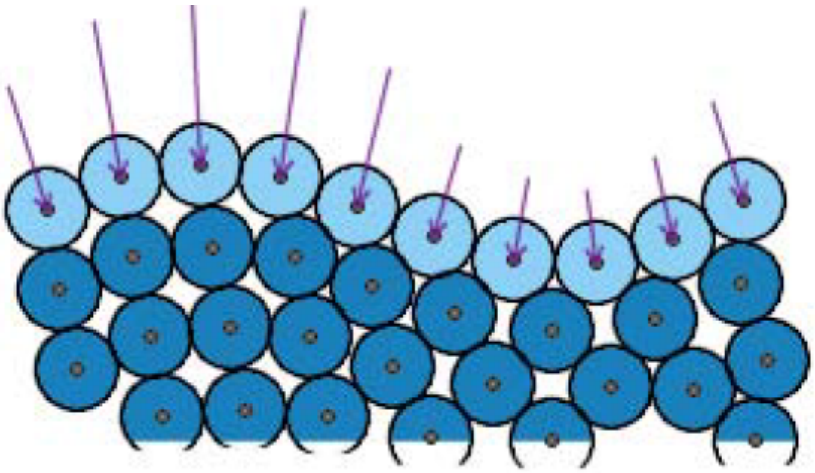

Figure 5.3: Behaviour of the surface tension force acting along the inward surface normals, in the direction towards the fluid (Kelager, 2006).

$$c_i = c\left(r_i\right) \tag{5.6}$$

$$= \sum_j c_j \frac{m_j}{\rho_j} W\left(r_i - r_j, h\right) \tag{5.7}$$

$$= \sum_j \frac{m_j}{\rho_j} W\left(r_i - r_j, h\right) \tag{5.8}$$

Its gradient length determines how close the particle are to the surface while n gives the surface normal. Calculations are as follows:

$$n_i = \nabla c\left(r_i\right) \tag{5.9}$$

$$= \sum_j \frac{m_j}{\rho_j} \nabla W\left(r_i - r_j, h\right) \tag{5.10}$$

The surface curvature is calculated by its Laplacian. The formula is as follows:

$$k = -\frac{\nabla n}{||n||} = -\frac{\nabla^2 c}{||n||} \tag{5.11}$$

Finally the calculation of the surface tension force is as follows:

$$f_i^{surface} = \sigma k_i n_i = -\sigma \nabla^2 c_i \frac{n_i}{||n_i||} \tag{5.12}$$

In the above equation the tension coefficient σ influences how strong a fluid surface holds to itself and is different for the interactions of different fluids. The length of gradient n leads to instability in the calculation of the force as it nears zero. So only when the gradient Lent exceeds the threshold value of β , the force is calculated or else it is taken as zero.(Kelager , 2006)

5.3.6 Interface Tension Force Calculation

Interface Tension is an another external force used in this program apart from surface tension for interaction of multiple fluids . It was first suggested byMuller et al in 2005. The curvature between the two fluids is minimised due to the interface force acting perpendicularly to the interface between them.

It has been adapted from the surface tension force discussed above with a difference. The difference is the colour Field. Here, taking into consideration the interaction of polar and non-polar fluids, it has been observed that a polar fluid will tend to mix with another polar fluid but not with non-polar one. Consequently, polar fluids are assigned with a colour value of -0.5 whereas the latter is given a colour value of +0.5.(Muller et al , 2005)

Gradient and Laplacian are calculated in the same way expect that the new colour values used would be 0.5 and -0.5.

5.3.7 Gravity Force Calculation

Gravitational force is yet another external force implemented in this program. Gravity causes an acceleration in negative Y direction, and is also referred to as free fall and its values is (0,-9.8,0). Similarly other external forces can be implemented in this program by simple vectors. As for example a vector of (0,0,-5) can be added to the accumulated forces and can be considered as wind blowing in negative Z direction.

5.4 Integration

The acceleration value of the each fluid particle is calculated by dividing the netforce by density. The next step is to find the new velocity and consequently the new position of the fluid particle. This is done by integrating the acceleration. There are two different methods of integration implemented in this project namely; Semi-Implicit Euler and Leapfrog.

5.4.1 Semi-Implicit Euler Method

Semi-Implicit Euler method is a slight derivative of the explicit Euler method and is implemented in this project. Explicit Euler method is the simplest integration method and it suffers from serious instabilities when the value of time step is kept higher. Semi-Implicit method also suffers from this numeric instability issue at higher timestep value but though its known to conserve the energy. This makes is a slightly bit more accurate. The calculations are as follows:

$$v_{i+1} = v_i + a.dt \tag{5.13}$$

$$x_{i+1} = x_i + v_{i+1}.dt \tag{5.14}$$

5.4.2 Leapfrog Method

Leapfrog is a second order integration method. So naturally it is more accurate then the first order Euler method. In this integration method both velocity and position is calculated at interleaved times but velocity is calculated at half times while the position is calculated at full integer times. The formulas used to calculate the velocity and position are as follows (Priscott , 2010)

$$v_{i+\frac{1}{2}} = v_{i-\frac{1}{2}} + a.dt \tag{5.15}$$

$$x_{i+1} = x_i + v_{i+\frac{1}{t}}.dt \tag{5.16}$$

Initial velocity offset $v_{-\frac{1}{2}}$ is calculated as follows using the Euler method.

$$v_{-\frac{1}{2}} = v_0 - \frac{1}{2}.dt.a_0 \tag{5.17}$$

While in this project the leap frog method is implemented with a different set of formulaes proposed by (Hut and Makino , 2004). The above formulas have been rearranged and rewrited as follows to find the value at full time step.

$$v_{i+1} = v_i + \frac{(a_i + a_{i+1}).dt}{2} \tag{5.18}$$

$$x_{i+1} = x_i + v_i.dt + \frac{ai.dt^2}{2} \tag{5.19}$$

In the above formula the the velocity is calculated as an average of current and next acceleration. This results in providing more stability to the displacement of fluid particles.

5.5 Optimisations

5.5.1 Neighbour Search using Spatial Hashing technique

The core part of SPH is the search of neighbours and their contributions. This core process affects the accuracy of the SPH calculations and the speed of simulation as it has to be performed at every time-step of the simulation. Moreover, the checking every particle against all other particle creates complexity of O(n2) and consequently limiting the number of particles that can be simulated. This is the most undesirable part of the native process.

There is no point of searching of all other particles beyond core radius h as they have no contribution to the current particle. Due this reason, the resource of CPU is substantially consumed.

Several algorithms exist to find neighbours efficiently and fall under the nearest neighbour search algorithms. Teschner et al. (2003) developed one such very efficient algorithm for SPH fluid simulation which is known as Spatial Hashing. This algorithm decreases the complexity from O(n2) to O(nm) where m is the average number of numbers found. Moreover, with the help of Kelager , 2006 this can even be reduced with a more uniform distribution of particles.

Spatial Hash Function

In the method of Spatial hashing, hash function converts a 3D position in space to a scalar hash key. Particles are stored in cells and also the neighbour particles are retrieved using this has key. This method is known as Spatial Hashing method.

The base of this method is the hash function as its design prevents the particles that are significantly apart to be hashed to the same cell.(Priscott , 2010)

(Kelager , 2006) defines it as follows:

$$hash(\widehat{r}\}) = (\widehat{r}xp_1 xor \widehat{r}yp_2 xor \widehat{r}y_p 3) mod n_H \tag{5.20}$$

To get a rounded integer value a discrediting function is used as it divides the position by cell size l . The equation is as follows:

$$r(\widehat{r}) = (\lfloor r_x/l \rfloor, \lfloor r_y/l \rfloor, \lfloor r_z/l \rfloor)^T \tag{5.21}$$

Smoothing length of the SPH Kernels is the cell size l here.

Size of the hash map n_H can be calculated as follows:

$$n_H = prime\,(2n) \tag{5.22}$$

Here function prime(x) returns the next prime number 0≥x and n is the particle count.

Still hash function requires the last missing piece and that is the values of p_1, p_2 and p_3. (Kelager , 2006) has suggested values for this and are as follows:

$$p_1 = 73856093,\ p_2 = 19349663 \text{ and } p_3 = 83492791$$

Insertion in Hash Map

Before the calculation of the forces, insertion of particles and filling the hash map is done. Hash map is cleared before the insertion of particles.

Now we iterate through the fluid particle list for getting their position, discretizing them, generating a hash key using hash function and finally inserting the hash key and particle in the hash map.

Searching the Neighbours

After the Hash map is refreshed, hash key is generated from the position of the particle to search for its neighbours and then the generated hash key is used as an index for retrieving all the particles from the map. This works if all the neighbour particles are in the same cell but that is not the case here. Kelager (2006)suggested to extend the search area around the particles using by using a bounding box. The limits of this box for a particle at position r are defined as follows:

$$BBmin = \widehat{r}(r - (h,h,h)^T), BBmax = \widehat{r}(r + (h,h,h)^T) \tag{5.23}$$

The two main reasons behind using bounding box are easy to iterate through and simple to calculate its limits. Now with respect to the search position only particles within the distance of the smoothing length(h) are required and they can be calculated as follows :

$$||r - r_j|| \leq h \tag{5.24}$$

The algorithm for this was adapted from (Priscott , 2010) where he describes how to optimise the spatial hash neighbour search with the use of bounding box. Its described as below:

Algorithm 5.1 Algorithm used to find the neighbours for a particle (adapted from (Priscott , 2010) and (Perseedoss , 2011))

Input Required: Particle p and its position x

- Search and Get Particles in the same cell as p
- Use of hash function to generate a hash key with position x
- Adding all the particles to map using the generated hash key
- Define Bounding Box containing all the search particles
- Define Bounding Box min and max limits
- Discretize the Bounding Box min and max limits
- for min to max
  - create a Cartesian position c within the bounding box
  - Using hash function create a new hash key with respect to position c
  - With the new hash key, search map for the particles
  - If they are not duplicate and if they are lying withing the smoothing length
  - Add them to neighbour list.
- End

5.5.2 CPU usage optimisation using OpenMP

"OpenMP is an Application Program Interface (API), jointly defined by a group of major computer hardware and software vendors. OpenMP provides a portable, scalable model for developers of shared memory parallel applications. The API supports C/C++ and Fortran on a wide variety of architectures." (Barney , 2011, https://computing.llnl.gov/tutorials/ openMP/)

OpenMP is based on shared memory model where jobs and shared memory space are created and allocated by multiple threads which are used in synchronising the result of their execution. Private and Shared memory variables concept used here, also clearly distinguishes the two. Preventing race condition is necessary while using shared memory as different threads overwrite the shared variable resulting in its undetermined state. (Barney , 2011)

OpenMP uses a pre-processor directive. Sections of code or procedure or tasks marked with #pragma omp, will be executed in parallel. On the basis of Barney's (2011) fork join model, this permits a combination of parallel and sequential code. In addition, it can be added at any stage of the project without hampering the structure of the code. Parallelism is provided at three different

levels, independent procedures, loops and higher level tasks. Loop parallelism works mainly on dividing loop iterations among multiple threads and where each thread executes their allocated iterations independently as shown in the figure below. (Barney , 2011)

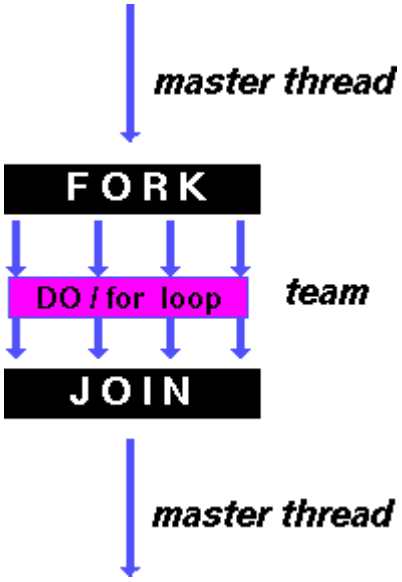

Figure 5.4: Figure Describes the Loop Parallelism of Fork Join Model. (Barney , 2011)

The simulation algorithm used here in the solver to achive new velocity and consequently new postion of the fluid is made up of four basic section. Density calculation, Force Calculation, Integration and Collision check and resolution. Looking at its nature its quite clearly revealed that it had the parallel potential. As all of these task in the program were performed in a similar fashion i.e. looping through the fluid particle list along with all other iterations. It was also observed that each of this task were completely independent of each other, thus simplifying the use of OpenMP.

Algorithm 5.2 Integration of OpenMP directives in the simulation algorithm

```
foreach iterations
refresh neighbours search structure
#pragma omp for
foreach fluid particle in fluid particle list do
Calculate density and pressure
end
#pragma omp for
foreach fluid particle in fluid particle list do
Calculate all the resultant forces
Integrate acceleration to get position and velocity
Collision Check and Resolution
end
end
```

5.6 Environment Interaction

5.6.1 Boundary Condition/Collision

In this project the boundary condition is applied using a simple primitive bounding box. A simple primitive Bounding Box is used as a boundary as it allows for really cheap calculations for detecting and resolving collision.

As the bounding box is made up of 6 faces, each will represent a min and max X, Y and Z limits. We simply check the x,y and z components of the fluid particle against these limits and in case if a collision is detected we simply set them to these limit values.

In case of bounding box the surface normals are unit normals. So the velocity is simply reflected as follows

$$v_i = v_i * -c_R \tag{5.25}$$

5.6.2 Collision with Passive rigid bodies

For all fluid particles, collision detection has been performed at each time-step. The logic used behind collision detection is to solve a implicit function at the current fluid particle position. If the result of the implicit function is negative it implies that collision has occurred.

If the collision is detected then the following three parameters needs to be determined for resolving it.

- the penetration point cp, where the particle has penetrated the surface.
- penetration depth d, whose value can be obtained by the solved implicit function
- surface normal n of the colliding surface.

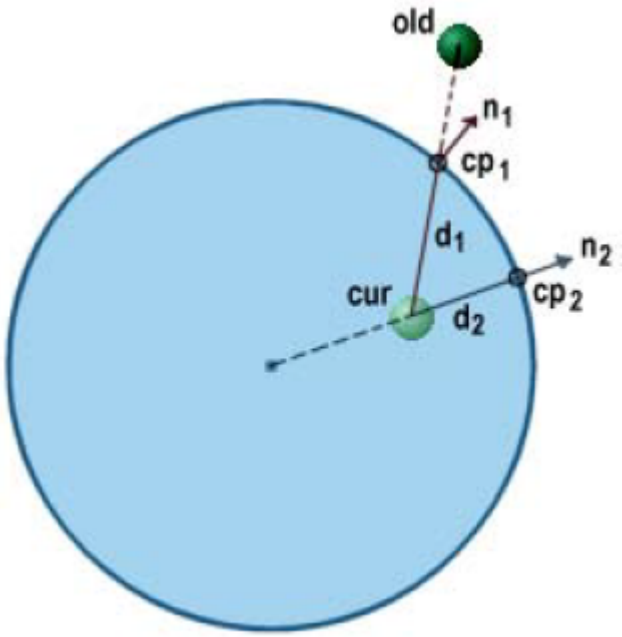

Figure 5.5: Two possible collision determinations from the same particle position update with cp1 being the correct point and cp2 the calculated one. (Kelager , 2006)

Contact point cp is calculated from the centre of rigid body along the normal. As shown in figure 5.2 there is a divergence occurring between the true contact point and the calculate contact point. But under small time-step value its is highly minimised.

This project implements the hybrid impulse projection technique proposed by (Kelager , 2006). According to this technique, the collided particle is simply projected back to the surface along the surface normal.

So the new position $r_i$is the simply the contact point.

$$r_i = cp \tag{5.26}$$

A Restitution Coefficient C_r is used to to determine the amount of velocity reflected back along the surface normal. Value of C_r is kept greater then 0

and less than 1. Thus, it mimic loss of energy on collision. With a value of C_r = 0 , will give rise to a inelastic collision and with the value of C_r = 1 simulating a perfect elastic situation.(Kelager , 2006) The formula to calculate the resultant velocity is as follows:

$$v_i = v_i - (1 + cR)(v_i.n)n \tag{5.27}$$

5.7 OpenGL Visualisation

The fluids particles here are visualised a sphere creates using vbo primitives from NGL Library.(Macey , 2010) Users can assign different colour to the the fluids through the setting xml file.

There can be specialised rendering technique like GPU base marching cube to view the simulation data, but rendering the simulation data is not the primary scope of the project. As mentioned earlier the primary scope of the project is to generate simulation data based on SPH. Also there are packages like Houdini which have tools capable of generating surface from point data.

5.8 Exporting Simulation Data to Houdini geo Format

The application specifically writes the data out in geo file format. Geo file is a Houdini's native geometry ASCII format.

Geo files can be readily loaded in Houdini with the file node. Various operation can then performed on these point data in Houdini , using a copy sop to display the point data as spherical particles or using a particle fluid surface node to generate fluid surface. Exporting the simulation data out is just written in these project to visualise the simulation data in a more enhanced way. There are a lot of possible extensions and techniques that can implemented to write data out more efficiently. But as stated earlier, primary focus of the this project is tracking fluid motion and generating simulation data rather then rendering it.

Chapter 6

Simulation Results and Analysis

This chapter shows different simulations generated using the solver and their results. The later part describes the issues that were faced and their considerations.

6.1 Simulation Scenarios

Multiple diverse scenarios have been set up and simulated, including single fluid, multiple fluids and their interaction with passive rigid bodies. Their results have also been demonstrated in the following section.

6.1.1 Single Fluid

The following four phases of liquid flow were observed after simulating several scenarios.

1. Fluid Particles fall down due to gravity force and splash on hitting the ground.
2. Fluid Particles moves towards the boundary, away from the centre of the initial splash. This is because of the induced pressure generated due to high density.

3. Fluid Particles after colliding with the boundary, slowly starts moving towards the initial splash position. This is observed due to pressure difference generated due to higher density at the boundary and lower density at the initial splash position. This also reflects the working to pressure term in the Navier-Strokes formula.

4. Fluid Particles come together at the initial splash position and then again start moving away due to pressure difference.

The results can be seen in the video file accompaning this project, path is "./ResultsOfTheSolver/" .

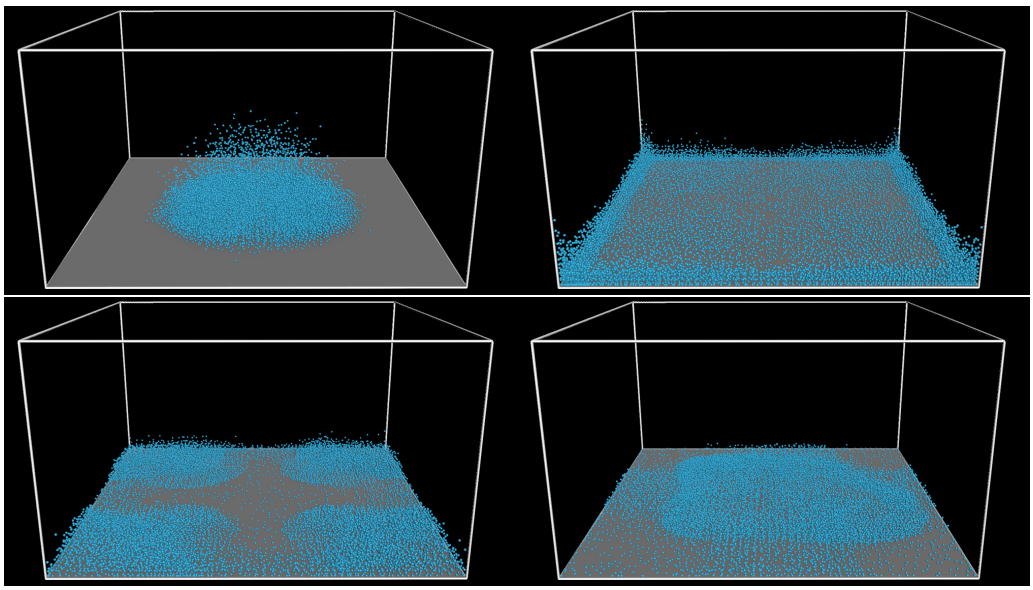

Figure 6.1: Result of the solver for single fluid simulation

6.1.2 Multiple Fluids

As interface forces(Muller et al , 2005) were implemented in this project, several test were made to observe interaction of multiple fluids. The results can be seen in the video file accompaning this project, path is "./ResultsOfTheSolver/" .

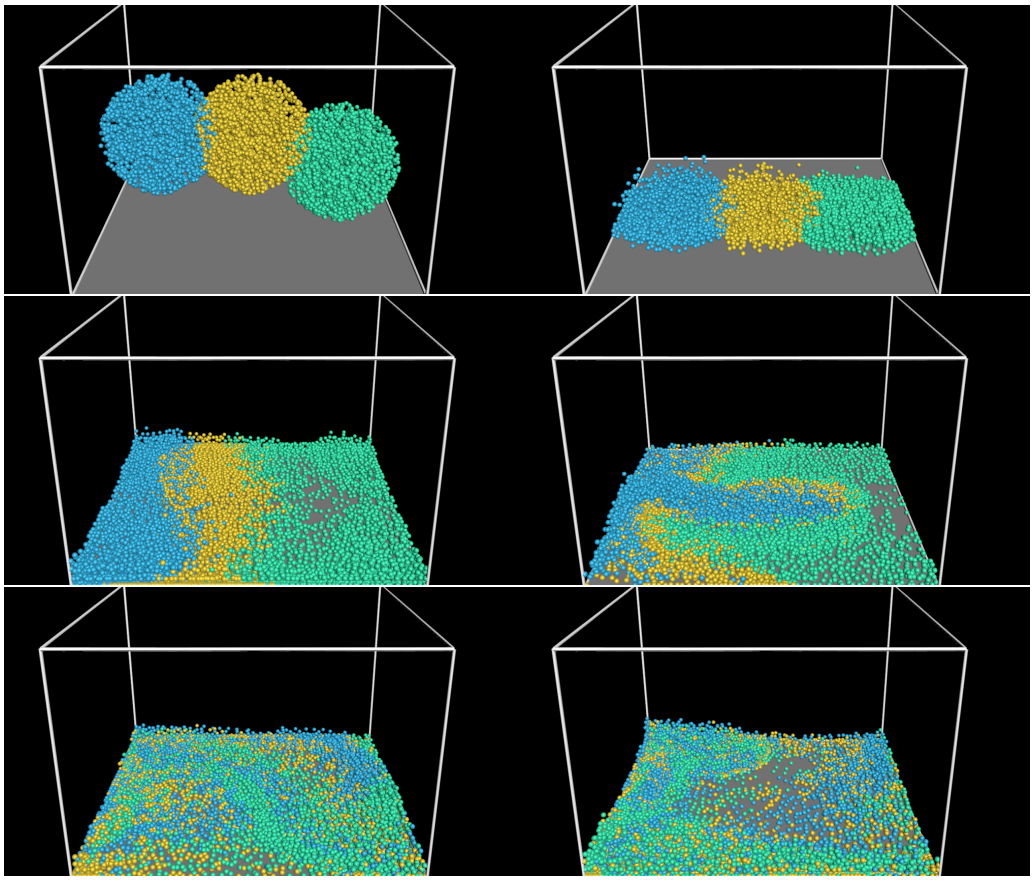

Figure 6.2: Result of the solver for Fluid and Rigid Body Collision

6.1.3 Fluid interaction with rigid body

Multiple and Single Fluids were tested colliding to passive rigid body (sphere). The results can be seen in the video file accompaning this project, path is “./ResultsOfTheSolver/” .

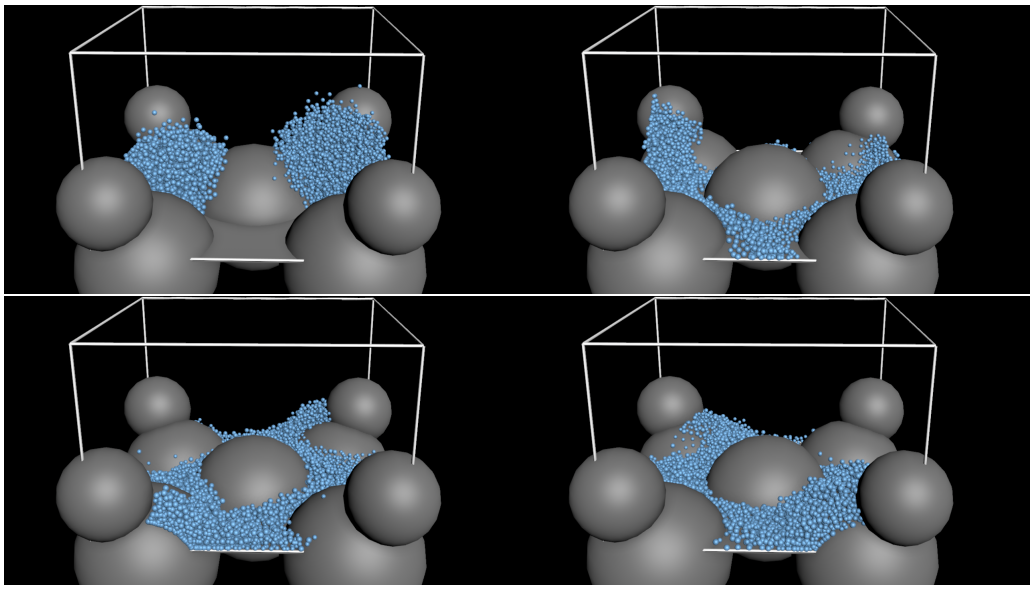

Figure 6.3: Result of the solver for Fluid and Rigid Body Collision

6.2 Issues and Considerations

6.2.1 Smoothing Length

The overall stability of the system is highly dependent on the smoothing length of the smoothing kernels in the SPH calculations. The following figure 6.1 clearly describes the results of various sizes of smoothing length.

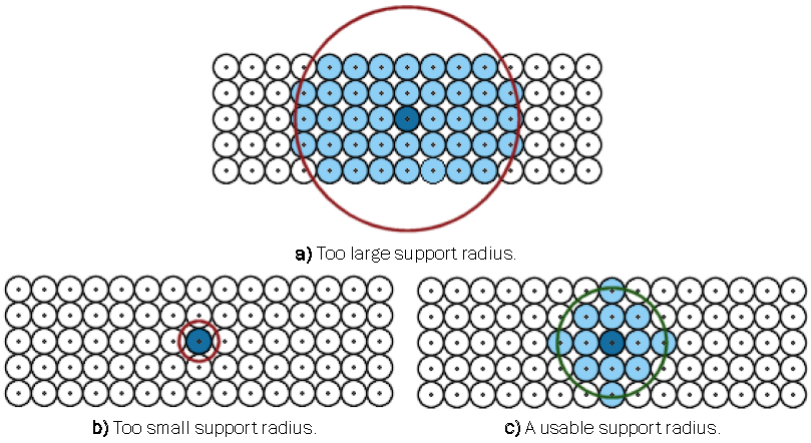

Figure 6.4: “A 2D illustration of the problem of using either a) a too large support radius, or b) a too small support radius. The dark sphere in the centre is the particle in question. The support radius is illustrated as a circle. In this example the support radius in c) will be a good choice.”(Kelager , 2006)

Larger smoothing length can adversely effect the stability of the system as its results in too many neighbours. Larger the neighbour count means more complex is the force calculation resulting in high processing time. Also large number of neighbour count will lower the influence of the close neighbouring fluid particles to the fluid particle being tested. This results in stretching of smooth kernels over a larger distance. Kelager (2006)provides with a temporary solution of specifying an average number of particles s for the kernel as show in the 6.1 .

$$h = \sqrt[3]{\frac{3Vs}{4\pi n}} \tag{6.1}$$

In this equation 6.1, “V” refers to volume and “n” to the fluid particle count. The aim is to keep the value of s small which will lower the complexity of force calculations resulting in a stable output.

On the other end the neighbour counts will result in being too less if the smoothing length value is assigned too small. This in-turn will result in unstable and erratic simulation as there is not enough contribution of neighbours in the force calculation.

Thus, for this project the smoothing length is set to a value of 0.3 as this value approximates the result of the equation 6.1 when s = 10. But still the user has the flexibility to change it as and when required.

6.2.2 Time step

The integration time-step value is chosen very critically as it directly affects the stability of the simulation performed due to integration. Considering high gas constant and viscosity, the time-step should be quite small. With implementation of all the optimisation techniques a time-step value of 0.01 was achieved keeping everything stable.

6.2.3 Viscosity and Dampening

In simulation scenarios, the liquid appear to be over damped then in real world. A simple solution to this is decreasing the value of viscosity constant. While doing this, one needs to make sure of making the time-step value very small to prevent explosion but which in turn will adversely effect the simulation interactivity. Mathematically, it would be proper to say that the simulation would remain as the damping introduced through viscosity would preventing it from exploding under huge pressure shock or pressure difference. Dues to this, it would not be a good alternative to decrease the value viscosity constant. In order to achieve the simulations at an higher time step, it narrows the margin of viscosity constant values for the project. One way of solving this issue is by implementing "implicit integration" but the process is highly computational-intensive.

6.2.4 Compressibility

In the pressure calculation(refer formula) from the density, a gas constant is used to determine how strongly the equalisation of the pressure difference occurs in the fluid. The near-incompressible behaviour of liquid can be achieved by making the value of gas constant as high as possible resulting in liquid coming to rest fast after a pressure change.

When a large change in pressure occurs, a high value of gas constant makes it behave like a hard spring that undergoes a huge numerical instabilities as it follows the behaviour of a mass-spring model.Kelager (2006) Thus, the other option is to decrease the integration time-step to a smaller value for stabilising the system. But this results in reducing the interactivity of the simulation. The best solution in this situation would be making a compromise between time-step and gas constant, in such a way that the maximum value possible is kept without stabilising the system. As seen in the config/setting.xml file the value of gas constant is 10 with time-step of 0:01 seconds is used to achieve a stable liquid simulation.

6.2.5 Performance and Efficiency

The main two approaches of optimisation, used in this project to boost efficiency and performance of the solver are neighbour search optimisation and

integration of OpenMP. Both the optimisation technique gave a considerable performance boost to the solver.

Neighbour search optimisation

Amongst all the process in the simulation loop, it was observed that the process of neighbour search was the most complex and used intensive resources. The main reason for this were the number of mathematical operations performed to search each particles neighbour. To add to this, the same process was repeated for each iterations. The situation even deteriorates as the neighbours needed to be calculated twice in the simulation algorithm, one for density calculations and other for force calculations.

While implementation two main methods of querying neighbours were used to see which suits the best. The first was getting the neighbour as and when need during the simulation algorithm. In this method, it was required to search and get the neighbours for each fluid particles, minimum two times per iterations. The second was to search and store all the neighbours of each particles in a huge neighbour list before performing all the simulation calculations. In this method, it was required search the neighbours and get all the neighbours for all the fluid particle, only once before the simulation calculations and store them in the huge neighbours list.

It was clearly seen that the second method was helpful in getting the performance boost in the execution time. But it increased the memory usage as it needed the storage to keep the neighbours. With high number of particle count, it resulted in degradation of performance of the data-structure used for storage.

Along with integration of OpenMP

Again both the above methods of querying neighbours were tested with OpenMP.

The results were totally opposite to the previous one. The second method of neighbour querying failed to deliver the performance boost in the execution time here. This was mainly due to the huge list of neighbours becoming a shared object between the threads. It became necessary to use the OpenMP ordered construct here to ensure that only one thread updates the huge neighbour list at a time and that too in order. This resulted in hindering the parallel working of the threads which consequently resulted in the increased execution time.

On the other hand, the first method here gave a performance boost in execution time. This was mainly due to the usage of all the cores. Thus, as a result the first neighbour querying method was used in the project along-with OpenMP integration.

Chapter 7

Conclusion

The main aim of the project was to implement a 3D particle-based Lagrangain Fluid Solver that solves Navier Stokes equation using Smooth Particle Hydrodynamics to generate fluid simulations.

7.1 Analysis and Report

The aim and objectives set in the design phase have been accomplished. Different scenarios have been generated and effective simulations of fluid has been shown. Multiple fluids interaction comes in as an inheritance with the Lagrangian particle-based methods. This project shows the implementation of this feature too. Interaction between fluids and passive rigid body has also been achieved and some interesting scenarios have been demonstrated in the above section.

Even successful implementation of the simulation pipeline has been demonstrated as the fluid is created using external obj's and the simulation data has been successfully exported out as geo files.

Furthermore, the implementation of neighbour search optimization techniques have been achieved. Integration of OpenMP has also been carried out resulting in giving a significant performance boost.

A Houdini file along with solver code also shows the fluid surface generation and simulation data representation by spheres or metaballs from the simulation data exported by the solver in form of geo files.

The solver also uses a xml configuration file to initialise the solver parameters and generates scenarios for simulation as defined in the xml file.

Thus, it can be concluded that all the objectives set initially have been successfully achieved.

There are various challenges to be met with, while simulating the flow of liquid as some of the parameters cannot be tweaked over a wide range of value. Due to this reason, a set of certain values have been used to generate fluid simulation presented throughout the paper. Also as discussed above in the simulation results and analysis section, altering the value of viscosity will not assure the generation of a stable simulation. Assumption is made that the simulations are stable as far as the value to time-step is kept low.

7.2 Future Works and Improvements

There is a big scope for future work by looking at the nature of these project. There are a number of issues that can be addressed and worked upon for improvements.

Obtaining a good simulation through SPH poses a lot of challenges that must be addressed as discussed in section 6.2. Some of these issues like calculation of density and pressure can be done with better accuracy by using alternative formulaes like Tait's equation(Monaghan , 2005). Issue of high compressibility can be resolved by the use of conjugate gradient method (Becker and Teschner, 2007). Issue of time-step limit can be resolved by using implicit integration methods (Stam , 1999).

The features which may be incorporated in the application for further enhancement and future study are:

- Developing a user interface for easy tweaking of parameters of the simulating fluids.
- Interaction with active rigid bodies.
- Reaction of fluid particle forces on active rigid bodies.
- Collision detection and resolution on base of a polygon mesh.
- Improving simulation algorithm to deliver high performance.
- Alternative technologies like MPI, OpenCl, etc can be used to boost the performance drastically.
- Improving the current spatial hash algorithm or implementing a new more efficient neighbour search technique that can handle a very large number of particles more efficiently.
- Integrating the solver directly into a 3D Package like Maya or Houdini.

Bibliography

Muller, M., Charypar, D., and Gross, M. 2003. Particle-based fluid simulation for interactive applications. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 154-159. Eurographics Association.

Hoetzlein, R. and Hollerer, T. 2009. Interactive water streams with sphere scan conversion. In Proceedings of the 2009 symposium on Interactive 3D graphics and games, I3D '09, pages 107-114, New York, NY, USA. ACM.

Bridson, R., 2008. Fluid simulation for computer graphics. Natick, MA, USA: A K Peters.

Griebel, M., Dornsheifer, T., and Neunhoeffer, T., 1997. Numerical Simulation in Fluid Dynamics: A Practical Introduction. (Monographs on Mathematical Modeling and Computation). SIAM: Society for Industrial and Applied Mathematics.

Stam, J. (2003). Real-time fluid dynamics for games., Proceedings of the Game Developers Conference.

Kelager, M. 2006. Lagrangian fluid dynamics using smoothed particle hydrodynamics. Available from: http://image.diku.dk/projects/media/ kelager.06.pdf [Accessed 5th December 2011].

Macey, J. (2010). Ngl graphics library. Available from: http://nccastaff.bournemouth.ac.uk/jmacey/GraphicsLib/ [Accessed 19 Aug 2011].

Priscott, C. (2010). 3d langrangian fluid solver using sph approximations. Master's thesis, Bournemouth University, NCCA. Available from: http://nccastaff.bournemouth.ac.uk/jmacey/MastersProjects/MSc2010/08ChrisPriscott/ChrisPriscott_THESIS.pdf [Accessed 19 Aug 2011].

Desbrun, M. and Gascuel, M.-P. (1996). Smoothed particles: A new paradigm for animating highly deformable bodies, In Computer Anima- tion and Simulation 96 (Proceedings of EG Workshop on Animation and Simulation, Springer-Verlag, pp. 61-76).

Hut, P. and Makino, J.(2004).The art of computational science. Available from: http://www.artcompsci.org/kali/vol/two_body_problem_2/ch02.html#rdocsect8 [Accessed 19 Aug 2011].

Muller, M., Charypar, D., and Gross, M. (2003). Particle-based fluid simulation for interactive applications. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 154-159. Eurographics Association.

Muller, M., Solenthaler, B., Keiser, R., and Gross, M. (2005). Particle-based Fluid-Fluid interaction. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 237-244. ACM.

Steele, K., Cline, D., Egbert, P. K., and Dinerstein, J. (2004). Modeling and rendering viscous liquids: Research articles. Comput. Animat. Virtual Worlds, 15:183{192.

Auer, S. (2008). Realtime particle-based fluid simulation. Master's thesis, Technical University Munich. Available from: http://www.google.co.uk/url?sa=t&source=web&cd=2&ved=0CCIQFjAB&url=http%3A%2F%2Fluid-particles.googlecode.com%2Ffiles%2Frealtime%2520particle%2520based%2520fluid%2520simulation%%2520thesis.pdf&rct=j&q=Realtime%20particle-based%20fluid%20simulation%20auer&ei=3AxOTqiLFIeHhQf7qMDeBg&usg=AFQjCNH6qXUW7Pi-jNjsuOAOo3h0LrQYgA&sig2=cZm71gZz7Q0pTAmq-9WD-A [Accessed 19 Aug 2011].

Angst, R. (2007). Fluid eects in cutting simulations. Semester thesis, Swiss Federal Institute Of Technology, Zurich. Available from: http://www.inf.ethz.ch/personal/rangst/publications/ST07.pdf [Accessed 19 Aug 2011].

Horvath, P. and Illes, D. (2007). Sph-based fluid simulation for special eects. In The 11th Central European Seminar on Computer Graphics. Available from: http://www.cg.tuwien.ac.at/hostings/cescg/CESCG-2007/papers/TUBudapest-Horvath-Peter/TUBudapest-Horvath-Peter.pdf [Accessed 19 Aug 2011].

Pelfrey, B. (2010). An informal tutorial on smoothed particle hydrodynamics for interactive simulation. Available from: http://www.cs.clemson.edu/ ~bpelfre/sph_tutorial.pdf [Accessed 19 Aug 2011].

Hirsch, C. 2007. Numerical computation of internal and external flows. 2nd ed. USA: John Wiley & Sons, Ltd, page 1),

Monaghan, J. J. (2005). Smoothed particle hydrodynamics. Reports on Progress in Physics, 68(8):1703-1759.

Monaghan, J. J. (1994). Simulating free surface ows with sph. Journal of Computational Physics, 110(2):399-406.

Hoetzlein, R. and Hollerer, T. (2009). Interactive water streams with sphere scan conversion. In Proceedings of the 2009 symposium on Interactive 3D graphics and games, I3D '09, pages 107{114, New York, NY, USA. ACM.

Lucy, L. (1977). A numerical approach to the testing of the ssion hypothesis. Astronomical Journal, 82:1013-1024.

Gingold, R. and Monaghan, J. (1982). Kernel estimates as a basis for general particle methods in hydrodynamics. Journal of Computational Physics, 46(3):429-453.

Stam, J. and Fiume, E. (1995). Depicting re and other gaseous phenomena using diusion processes. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, SIGGRAPH '95, pages 129-136, New York, NY, USA. ACM.

Becker, M. and Teschner, M. (2007). Weakly compressible sph for free surfaceows. In Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '07, pages 209-217, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Miller, G. and Pearce, A. (1989). Globular dynamics: A connected particle system for animating viscous uids. Computers & Graphics, 13(3):305-309.

Desbrun, M. and Cani, M.-P. (1996). *Smoothed Particles: A new paradigm for animating highly deformable bodies*. In Boulic, R. and Hegron, G., editors, Eurographics Workshop on Computer Animation and Simulation (EGCAS), pages 61-76, Poitiers, France. Springer-Verlag. Published under the name Marie-Paule Gascuel.

Stam, J. (1999). Stable Fluids. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 121{128. ACM Press/Addison-Wesley Publishing Co. Available from: http://www.dgp.toronto.edu/people/stam/reality/Research/pdf/ns.pdf [Accessed 19 Aug 2011].

Reeves, W. T. (1983). Particle systems : a technique for modeling a class of fuzzy objects. ACM Trans. Graph., 2:91-108.

Takeshita, D., Ota, S., Tamura, M., Fujimoto, T., Muraoka, K., and Chiba, N. (2003). Particle-based visual simulation of explosive ames. Computer Graphics and Applications, Pacic Conference on, 0:482.

Teschner, M., Heidelberger, B., Muller, M., Pomeranets, D., and Gross, M. (2003). Optimized spatial hashing for collision detection of deformable objects. In Proceedings of Vision, Modeling, Visualization VMV03, pages 47-54. Available from: http://www.beosil.com/download/ CollisionDetectionHashing_VMV03.pdf [Accessed 08 Dec 2011].

Clavet, S., Beaudoin, P., and Poulin, P. (2005). Particle-based viscoelasticuid simulation. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '05, pages 219-228, New York, NY, USA. ACM.

Premoze, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R. T. (2003). Particle-based simulation of uids. Computer Graphics Forum, 22(3):401-410. Available from: http://www.sci.utah.edu/~tolga/pubs/ ParticleFluidsHiRes.pdf [Accessed 19 Aug 2011].

Carlson, M., Mucha, P. J., Van Horn, III, R. B., and Turk, G. (2002). Melting and owing. In Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '02, pages 167-174, New York, NY, USA. ACM.

Steele, K., Cline, D., Egbert, P. K., and Dinerstein, J. (2004). Modeling and rendering viscous liquids: Research articles. Comput. Animat. Virtual Worlds, 15:183-192.

Paiva, A., Petronetto, F., Lewiner, T., and Tavares, G. (2009). Particle-based viscoplastic uid/solid simulation. Computer-Aided Design, 41(4):306-314.

Bridson, R. and Muller-Fischer, M. (2007). Fluid simulation: Siggraph 2007 course notes. In ACM SIGGRAPH 2007 courses, SIGGRAPH '07, pages 1-81, New York, NY, USA. ACM.

Barney, B. (2011). Openmp. Technical report, Lawrence Livermore National Laboratory. Available from: https://computing.llnl.gov/tutorials/ openMP/ [Accessed 19 Aug 2011].

Perseedoss, R., 2011. Lagrangian Liquid Simulation using SPH. Available from http://nccastaff.bournemouth.ac.uk/jmacey/MastersProjects/MSc11/Rajiv/MasterThesis.pdf [Accessed 30 November 2011]

Fedkiw, R., No year. Available from http://physbam.stanford.edu/~fedkiw/ [Accessed 15 August 2011]