8-2010

SIMULATION OF THE MIXING OF INK AND WATER IN HOUDINI

Chen Sun
Clemson University, chens@clemson.edu

SIMULATION OF THE MIXING OF INK AND WATER IN HOUDINI

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Fine Arts
Digital Production Arts

---

by
Chen Sun
August 2010

Accepted by:
Donald House, Committee Chair
Timothy Davis
Tony Penna

ABSTRACT

Have you observed the process of one ink drop falls into water? When the ink drop spreads out in the water, it looks like the ink molecules are having a beautiful dance show. The texture is smooth and delicate as a piece of silk. This thesis uses computer graphic tools to simulate the mixing of ink and water. Anyone interested in recreating ink mixing with water effect or using this as an art form can reference this thesis paper in their own production.

ACKNOWLEDGMENTS

Much thanks to Dr.House who has been a great guide and invaluable resource. I would also like to thank my committee members, Tony Penna and Dr.Davis, for the help their knowledge and direction has been.

Thanks to my DPA friends who have helped me get going.

Finally, thanks to my parents for their love and support.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Visual effects combine live action footage and Computer Generated Imagery (CGI) together to create realistic scenes that would otherwise be costly, dangerous or simply impossible to capture on film. Because of this, visual effects have been widely used in film, television, gaming and other parts of the entertainment industry. Fluid simulation is one of the most popular forms of visual effects used to generate realistic animations of water, smoke, explosions, and related phenomena. Within this broad area, this thesis focuses on the simulation of the motion of two fluids mixing together.

The collision of two kinds of fluid molecules makes the fluid surface look delicate and beautiful. It could be used in commercials, films and other media for creative purposes, such as close-up shots of rigid bodies turning into fluid. Most fluid simulation is made for larger water bodies, such as springs, lakes or even oceans, since they are more widely used in productions. Hence most production software packages are made to do that too. You can easily find software that can creates a river or a lake, however it is much harder for effects artists to use those software packages to simulate small amount of fluid like blood drops or ink drops. When simulating larger bodies of water, the motion of the fluid surface is the focus. Small internal movements can be ignored since it would be hard to see them anyway. On the other hand, when it comes to simulating a smaller amount of fluid, such as a single drop of ink, every tiny movement will be put under heavy scrutiny. It requires the effects artist to simulate the motion of the ink drop

spreading out and mixing together with the water. Realistic integration occurs not only with the fluid surface but also with the internal motion of the fluid itself.

There are two major challenges: motion and shading. The motion of liquid molecules spreading out in another liquid is hard to achieve because it is found somewhere between unpredictable and completely predictable. This motion follows certain constant physical rules, however it also has a certain amount of randomness as well, as is shown in Fig1. Randomness can only be simulated through a lot of experiments. Another challenge is the shader. Because of the physical attributes of the liquid surface, light can go through the liquid surface. It adds more complications onto the shader of liquid.

This thesis focuses on simulating the motion of ink drops into the water. The entire process begins with the ink dropping into the water, then spreading out and finally dissipating in the water. The interested reader can watch the video shown in [1]. Houdini is the main production software package in this thesis.

Fig1. Reference images, ink mixing with water

CHAPTER TWO

BACKGROUND

2.1 Particle Systems

According to William Reeves' Siggraph paper [2], particle systems are widely used in modeling fuzzy objects like fire, smoke, liquid etc. They consist of a group of many points in 3D space. Each particle has a combination of the following parameters: position, velocity, acceleration, color, alpha, size, life span etc.

For each time frame, the following steps are performed: new particles are created according to the setting; if the particle's life exceeds its life span, the particle is destroyed; compute the new acceleration, new velocity and new position according to the equation below:

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

$$v_2 = v_1 + t \cdot a$$

$$p_2 = p_1 + t \cdot v_1$$

Where a is acceleration, f is force, m is mass, t is time step, v_2 is the new velocity, v_1 is the old velocity, p_2 is the new position, p_1 is the old position.

Update the new parameters for each particle after each time step.

2.2 Particle Rendering

As Karl Sims [3] explains it, the data parallel method is used to render anti-aliasing and blurred image of large amount of particles which have variable colors, alpha

channels and radius. The system sorts the fragments by pixel and z-depth according to the effective width in the scene, and then sends color information to each pixel.

Generally speaking, there are three ways for rendering particles:

- Render particles as points: render each particle as a point in 3D space.

- Render particles as sprites: render the particle as one square with a texture attached to it, the sprites will always face the camera.

- Render particles as other geometry: in Houdini, particles can be rendered as lines, spheres, disks, lines, tubes etc.

Geometry can be rendered as points in Houdini without having to create particles.

2.3 Particle Advection

Transporting mechanical properties from fluid like smoke and liquid to particles is called particle advection.

In Houdini, the Advect by Volumes" node can advect particles by a velocity field defined volumetrically.

2.4 Fluid simulation

Most fluid simulation software nowadays is based on simplified incompressible Navier-Stokes equations [4, 5].

$$\overbrace{\rho\left(\underbrace{\frac{\partial v}{\partial t}}_{\substack{\text{Unsteady}\\ \text{acceleration}}} + \underbrace{v\cdot\nabla v}_{\substack{\text{Convective}\\ \text{acceleration}}}\right)}^{\text{Inertia(per volume)}} = \overbrace{\underbrace{-\nabla p}_{\substack{\text{Pressure}\\ \text{gradient}}} + \underbrace{\mu\nabla^2 v}_{\text{Viscosity}}}^{\text{Divergence of stress}} + \underbrace{f}_{\substack{\text{Other}\\ \text{body}\\ \text{forces}}}$$

Where ρ is the fluid density, v is the fluid velocity, p is the pressure, μ is the dynamic viscosity, f is the other body forces (per unit volume).

Under the assumption that fluid is incompressible, density is a constant:

$$\nabla \cdot v = 0$$

Wejchert and Haumann [6] developed the method of simulating laminar fluid flow. The velocity field of the laminar fluid flow satisfies Navier-Stokes equation [4, 5] and boundary limitation. After that, Foster and Metaxas [7] presented a comprehensive methodology of simulating the realistic animation of liquids. It is the first paper which talks about dynamic fluids using Navier-Stokes equations. On top of that, Stam's method [8] makes it possible to do fluid simulation in real time. In 2003, Muller, Charypar and Gross [9] proposed a method based on Smoothed Particles Hydrodynamics to simulate arbitrary fluid motion.

In Houdini, there are parameters that can be used to tweak the motion of the fluid. The parameters are:

Viscosity: the velocity in one grid affects the velocity of neighboring grids. This parameter can affect the dissipating motion of the fluid.

Cooling Rate: defines how soon the temperature field goes to zero.

Temperature Diffusion: defines how fast the temperature field mixes together. It also controls how smooth the fluid spreads out.

Buoyancy Direction: the direction of buoyancy force.

Buoyancy Lift: proportional to the difference between ambient temperature and this voxel's temperature. If the voxel's temperature is above ambient temperature, the

buoyancy force goes up while the buoyancy force goes down if the voxel's temperature is below ambient temperature.

In Houdini, seed vorticles can be added to create paddlewheel-like forces. The size of the bounding box, delete number of vorticles, magnitude and radius of each vorticle force, and up direction can be changed. However, Houdini automatically randomly generates the position of vorticles. The position can be tweaked by changing the random seed, but Houdini does not allow for more than that. Hence, the exact position needed cannot be obtained. Later in this paper, the procedure for aligning the vorticles into the exact needed position will be discussed.

Vorticle Strength: control the scale for the Vorticle forces

Vortex Confinement: According to Steinhoff J and Hu G [10-12], vortex Confinement is used to counteract the numerical diffusion by artificially generating tilting/stretching and convection flux. This parameter can be used to generate motions like air flow around a helicopter.

CHAPTER 3

METHODOLOGY

In the following part, I will discuss the process of generating ink drops into a water simulation step by step.

3.1 General Pipeline

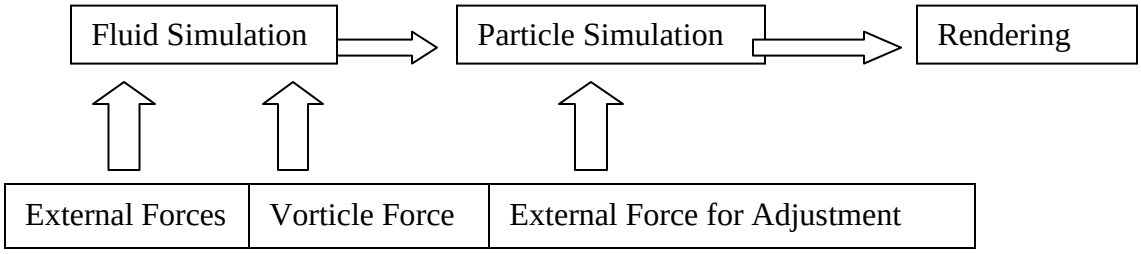

Fig2. General Pipeline

Fig2 shows the general pipeline of this thesis project. In order to get more details for the motion of ink spreading out in the water, fluid simulation and particle simulation are both used in the pipeline. Fluid simulation is used to generate the overall motion. It simulates the ink drop through settings like temperature, viscosity, temperature diffuse, buoyancy lift, etc. Other external forces like fan force, gravity, and drag force can be applied onto fluid simulation to help simulate the real world force. Internal force like vorticle force could be added to make the simulation more interesting. Vorticle force applies onto velocity field of the simulation to create paddlewheel-like forces around each vorticle. Compared to particle simulation, fluid simulation is more physically accurate in simulating the motion of two different fluids. However, having only fluid simulation is not good enough, because fluid simulation does not give the effects artist

enough control. Normally, the job of the effects artist is to feed multiple levels of complex data into a fluid simulation and then wait for the result. Normally, the job of the effects artist is to feed multiple levels of complex data into a fluid simulation and then wait for the result. If the result is not what they want, they can only change the setting or change the parameters. Then they are required to restart the simulation from the beginning. Another reason why solely using fluid simulation would not achieve a successful, realistic simulation is because the internal fluid is equally as important as the surface. The surface will spread out and become more transparent, making the internal fluid effects more obvious. Particle simulation is necessary to achieve the proper effect and it also provides more control for the effects artist. A particle's parameters such as alpha, life span, velocity, position, acceleration can help show the ink drop's minor movement. Another important point is backlight; it is needed in this simulation and will enable the particle's normal to be changed freely when necessary. The particle's normal can be changed freely if necessary.

The last procedure in the pipeline is rendering. Rendering in this simulation is a major problem. The volumetric shader doesn't show enough detail. It is hard to improve or avoid this problem because it is a characteristic of volumetric shading. Particle rendering is a better choice. However two major problems need to be solved. First, because every point is rendered, it is likely to get a grainy look if there are not enough particles. Solving this problem requires a large number of particles. Second, the light goes through the ink drop surface so the back side of the back surface is lit by light also.

However the feature in Houdini “Render as Points” does not pass the normal of the points to the shader.

3.2 Fluid Simulation

In Houdini, creating a volumetric fluid simulation is easy and simple. Houdini has some pre-settings that can be chosen, such as smoke simulation, pyro simulation and fire simulation. In this thesis project, smoke simulation is chosen to start with. The following procedure is how to build a default fluid simulation in Houdini.

The fluid simulation is volumetric fluids, so the emitter and volume container are the two major parts that need to be created. First, create a sphere in the scene to be the emitter of the ink drop. Then create a container according to which kind of fluid is going to be generated. Adjust the length, width and height of the container. Move the container to make sure the emitter is in the container. In this particular situation, the ink drop falls from the top of the scene. In order to make the container as effective as possible, the emitter is put at the top of the container but within its range. After that, build a dynamic operation (DOP) network based on the emitter and container. The smoke is generated. Fig3 shows the three main nodes of the fluid simulation in the network editor: emitter, smoke object and the DOP network. “Sphere_object1” is the sphere object just created and is the emitter as well. “Smokeobject1” is the smoke object created from emitter. “AutoDopNetwork” is the network which processes all the dynamic simulation.

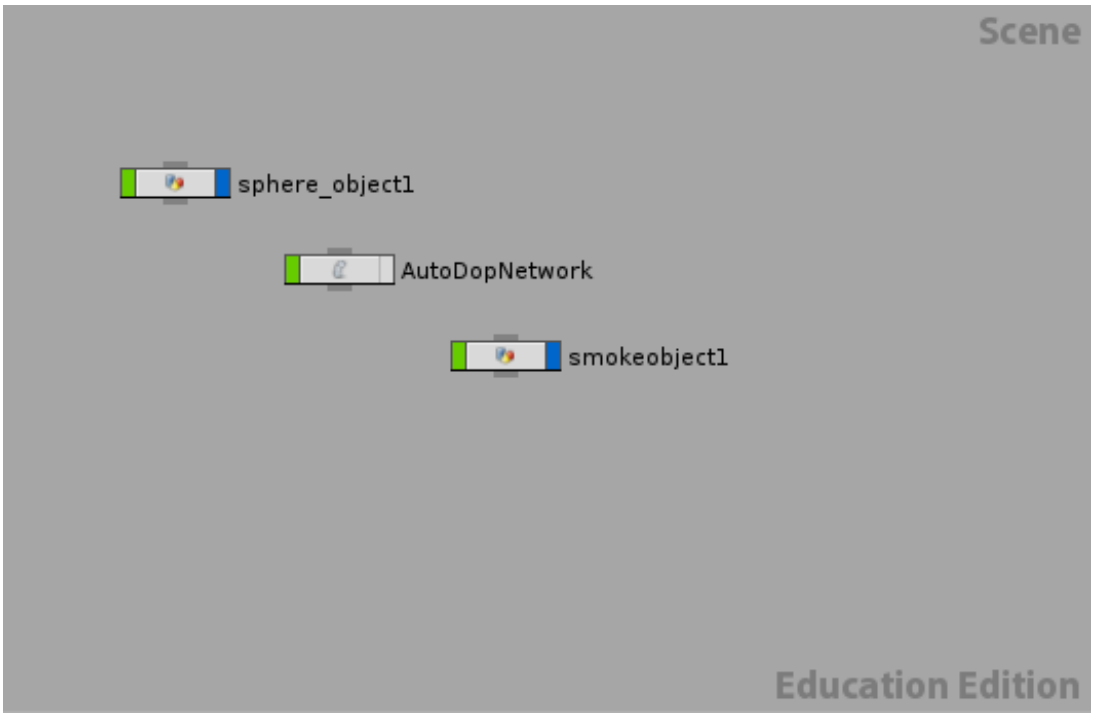

Fig3. Network for the default smoke volumetric fluid simulation

Next step is to change the basic settings based on the default smoke simulation to get closer to the final goal. Table1 shows the attribute value for this particular case. Due to the gravity, the ink drop is supposed to sink down into the water. Other than adding the gravity, controlling the temperature of the emitter is another way to make the smoke sink down. The basic concept is when the higher temperature goes up, the lower temperature sinks down. Here the temperature is set to a ramp from 0 to -0.5 so the smoke gradually goes down.

According to the character of the volumetric fluid simulation, the smoke only exists inside the smoke container. The system ignores all the other spaces outside the smoke container. So the size of the smoke container should be adjusted according to the smoke simulation so that the smoke is not cut off by the container.

Under the default setting, the smoke simulation is fast to compute. However, it is hard to see the detail movement of the smoke simulation. And the velocity field doesn't show up at all, which makes it hard to tell the direction of the velocity. In Houdini, the volumetric container is divided into a specific number of divisions. The more divisions the container is divided into, the more detail the fluid simulation receives, and the more computation time needed. The default uniform division normally does not satisfy the need of the simulation. A proper division number should be a good balance between having enough details and being relatively fast to compute. Velocity field can be visualized in Houdini also. It is very helpful for effects artists.

In order to simulate the real world forces, external forces like gravity and drag force are necessary. Everything should be affected by gravity while the drag force simulates the fraction force between the ink drop and water. Drag force opposes the relative motion of the object, acting on the direction opposite of the velocity. It depends on the object's velocity directly.

By observing the reference video, with the ink drop falling down, ink molecules spread out and interact with the water, which makes the ink drop expand and have certain amount of swirling motion occurring. In order to simulate this swirling motion, the parameter "vortex confinement" needs to be adjusted.

Now the fluid simulation has an overall swirling motion after adjusting the "vortex confinement" parameter. However, only having a swirling motion makes the simulation too uniform after the ink molecules have been interacting with the water for a while, there is another, smaller variant motion going on under the main tendency.

In order to make the fluid simulation more varied, vorticle force is added. Another problem can occur in that the exact vorticles' positions are needed so the effects artist can control the vorticle force based on the video reference. However in Houdini, the way in which vorticle force works is that the system creates a box which has the same size as the volumetric container. Then transform the box to an Isosurface. Sample a certain amount of points randomly from the Isosurface. Those points are the positions of vorticles. Fig4 shows the network editor for the default vorticle force. The only way to change the vorticles' positions directly is changing the seed of the randomness. But this is not effective enough for this thesis project since the specific direction and position of the vorticle force is needed.

Other than completely rewriting a plugin for vorticle force, there is an easier way to solve this problem. The basic idea is separating the vorticles into several groups. Then transfer each group into the position needed. Each group has its own bounding box. Merge all the groups together in the end. How many groups the vorticles are separated into and how many vorticles each group contains can be varied based on the specific situation. Fig5 shows the modified network of the vorticle SOP net. SOP means surface operation in Houdini.

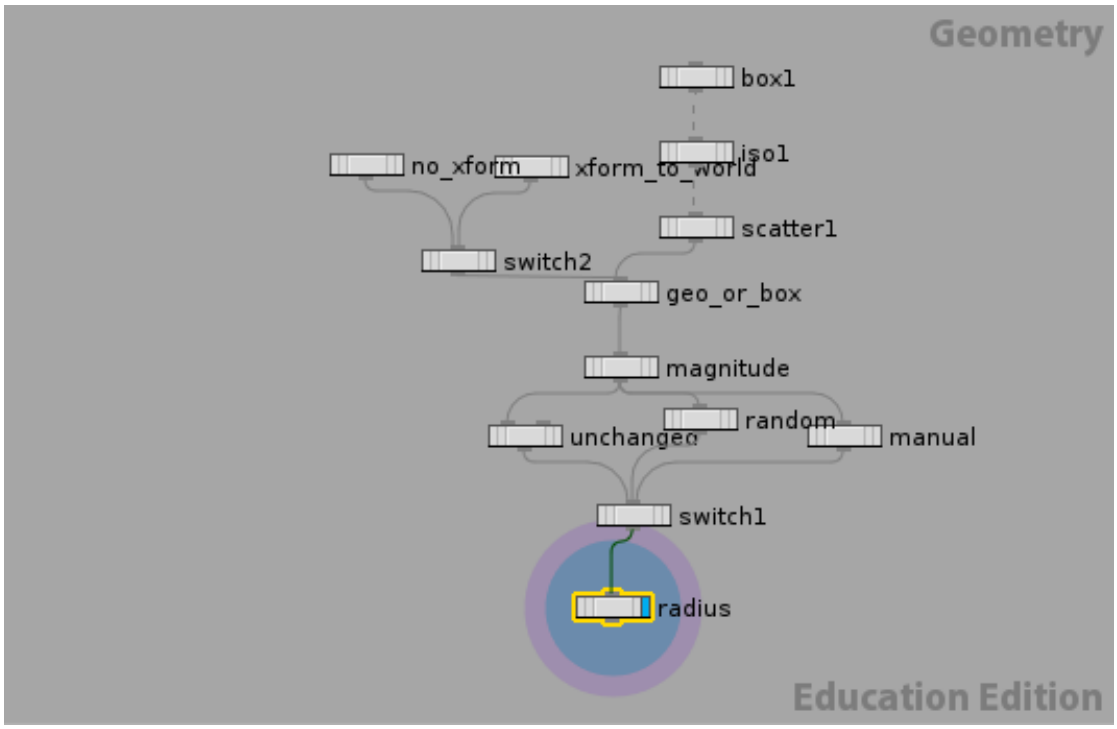

Fig4. Default network editor for vorticle force

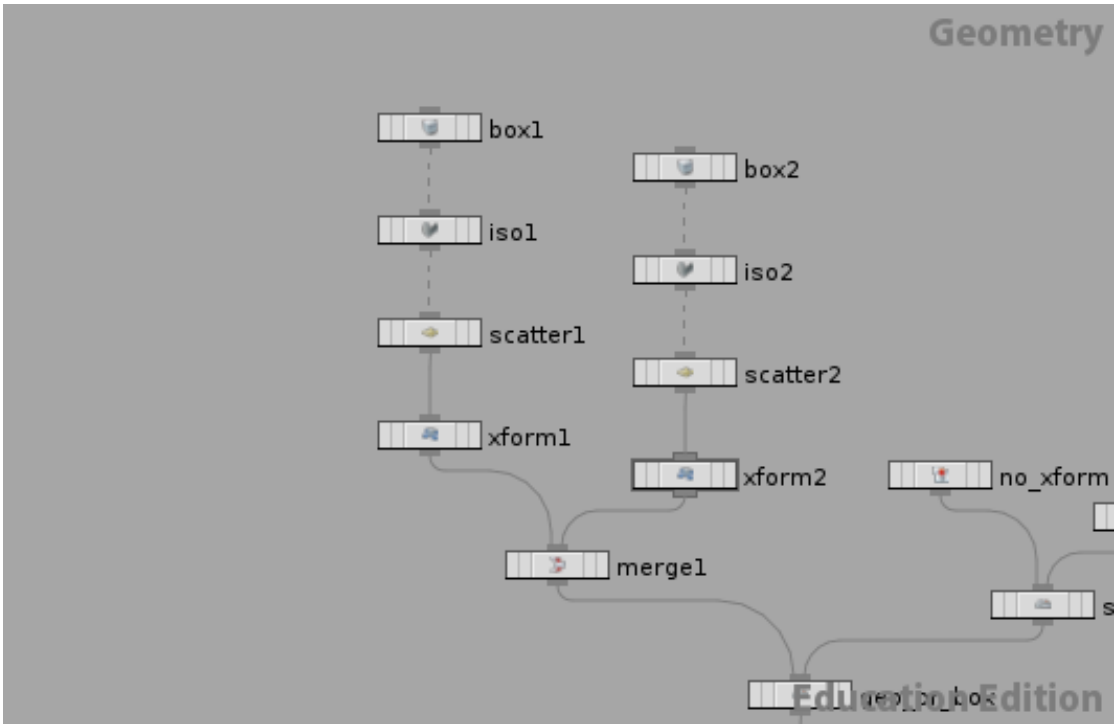

Fig5. Modified network of the position of vorticles.

Attribute Name	Value
Scalar Ramp of Sphere Emitter	-0.5
Uniform Divisions of Smoke Object	30
Size of Smoke Object	(10,15,10)
Center of Smoke Object	(0,2.6,0)
Viscosity of Smoke Solver	1
Cooling Rate	0.9
Temperature Diffuse	0.3
Buoyancy Direction	(0,1,0)
Buoyancy Lift	25
Vorticle Strength	0.3
Vortex Confinement	3

Table1. Attribute value for fluid simulation

3.3 Particle Simulation

Now the fluid simulation is finished. After increasing volume division, add external forces and vorticle forces. Does it mean everything has been done to improve the quality of the simulation? What else can be done to help add even more details onto the simulation? The answer is particle systems. Each particle represents one point in 3 dimensional spaces. It is a lot smaller than one grid in volume.

The next challenge is creating a fluid system and a particles system that will work together effectively. If the particles can inherit fluid simulation's motion, it will be more

efficient and effective for the effect artists to modify the particles' motion based on the original motion. In Houdini, the method which advects particles by volume can be used to solve the problem above. By advecting the particles by volume, the motion of the volume is transferred to particles. The particles could either be advected by velocity field or force field. Through experiment, advecting by velocity field is closer to what is needed.

In order to simulate the ink drop falling down into the water, another sphere is created on top of the container to be the emitter of the particles. How many particles does the emitter emit per frame and where are those particles emitted from is another problem that needs to be tested. As discussed at the beginning, the motion of the ink drop is not only the motion of the fluid surface; the fluid under the surface is important too. According to this project, particles cannot only be emitted from the surface of the sphere emitter; they need to be emitted inside the sphere as well. Fig6 shows the network of where particles are emitted. Scatter the sphere SOP to get the emitted position on the sphere surface. Adding the Isosurface node can get the volume inside the sphere. Scatter the volume to get the emitted position inside the sphere.

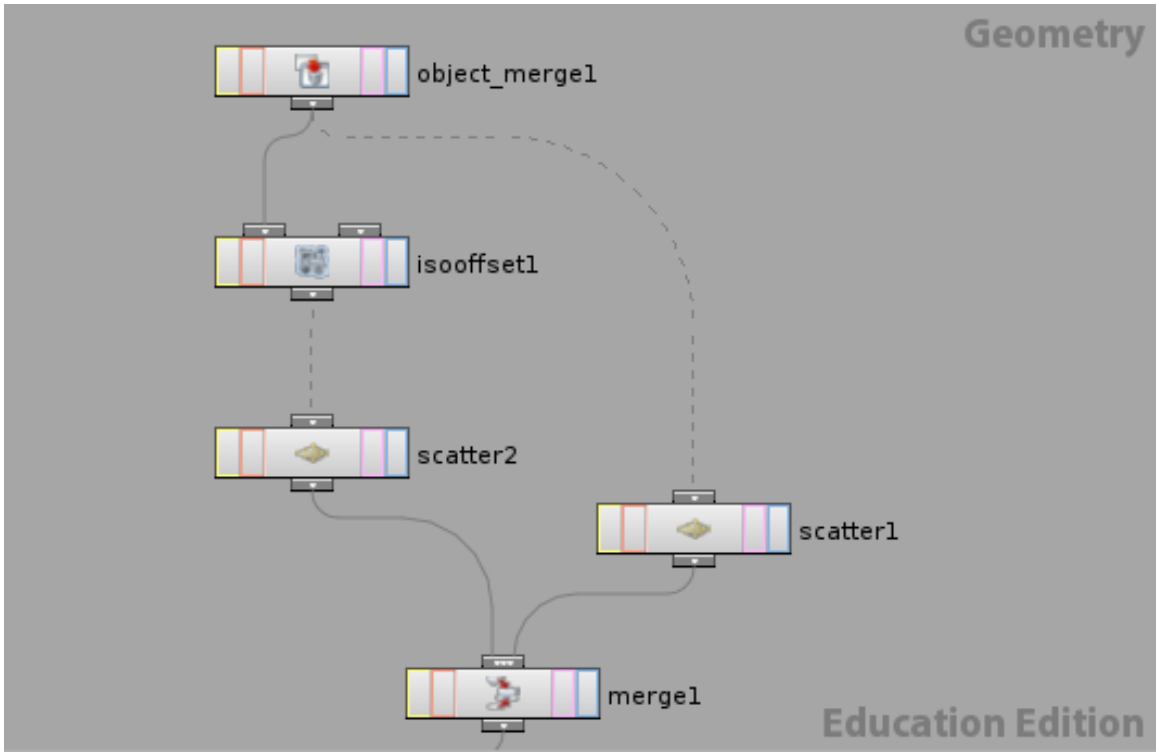

Fig6. Network of where particles are emitted

In this thesis project, the simulation is focused on one drop so the entire environment has fixed amount of inks molecules. In this case, the simulation can show the movement of the ink molecules clearly. In order to achieve this, the emitter is set to emit particles for the first 10 frames. Particles will not die.

Now that the velocity field has been transferred from fluid simulation to particles, other forces can be added to apply onto the particle systems to adjust the final motion of particles. Which force should be added and what the magnitude should be is totally depended on the effect artists' aesthetic decision. However, the change of the movement should be minor compare to the original fluid simulation. In this particular case, because the sphere emitter is located on top of fluid container, the gravity applied to fluid simulation does not work for the particles. So gravity force needs to be added again in the

particle network. Make the gravity force applied to particle systems active only before a certain frame so it creates the momentum for the ink drop falling into the water and does not overlap with the gravity force applied to fluid simulation.

Refer to Table 2 for attributes values of particle network.

Attribute Name	Value
Impulse Activation	$F<10
Impulse Birth Rate	$NPT
Life Expectancy	100
Advection Type	Update Velocity
Velocity Blend	0.3
Scale for Gravity Force	10

Table2. Attribute for particle network

3.4 Rendering

The motion of particles is all set. The remaining steps is the challenge of rendering the particles into the ink. Mantra renderer is used in this thesis project.

Compare to other rendering method such as volumetric fluid rendering and particle sprites rendering, point rendering for particles are better choice. Volumetric fluid rendering is suitable for larger body of fluid. If using this method, a lot of details will be lost in the final image. It is the same reason for particle sprites rendering. Point rendering can show every particle in the scene. There is no detail loss in point rendering.

However there are two major problems for point rendering. The first problem is it is very easy to get a grainy look in point rendering which we do not want. The only way to overcome this problem is using more particles. Generating more particles will add a lot more computational time, but the final image

is a lot more satisfying.

In order to create more particles but not kill the machine, multiple seeds can be used in generating particles. Simply change the random parameter in scatter node, then cache out the particles geometry on hard drive. Read the particles back in when rendering the final image. Refer to Table3 for the amount of scatter points in this thesis project.

Attribute Name	Value
Number of Scatter Points	500 000
Random Seed	$F
Scale for the points	0.009

Table3. Attribute for geometry network

The second problem is lighting. Because the water refracts light, the light will go through the ink drop when the ink drop spreads out in the water. In order to see the inside of the ink drop, back light is necessary. This creates a new challenge in creating backlight in the scene. How to create back light in the scene? The answer is easy --- change the normal of the particles.

However in this case, the feature in Houdini “render geometry as points” is used. It speeds up the rendering for large amounts of particles. The reason why “render geometry as points” is fast is because each point only calls the shader once, and the

points do not pass the normal to the shader. [13] In order to make the shader work with the normal, new vector normal is created.

Create a new attribute in the network. Set the value of the attribute as reverse of particles' original normal. Then import the parameter into shader and make it as the normal used in shader. The network is like Fig7. Fig8-Fig9 shows some additional network of the shader.

Attribute Name	Type	Value
ptN	Vector	-$NX, -$NY, $NZ

Table4. New created normal attribute for points

Table4 shows the value of the normal attribute. The reason it is set as (-$NX, -$NY, $NZ) is because a negative scaling on the z-axis immediately prior to instancing the light source shader.[14] Since the front face is not used on the points, the surface normal is reversed. The value is changed from ($NX, $NY, -$NZ) to (-$NX, -$NY, $NZ) as shown in the Table1.

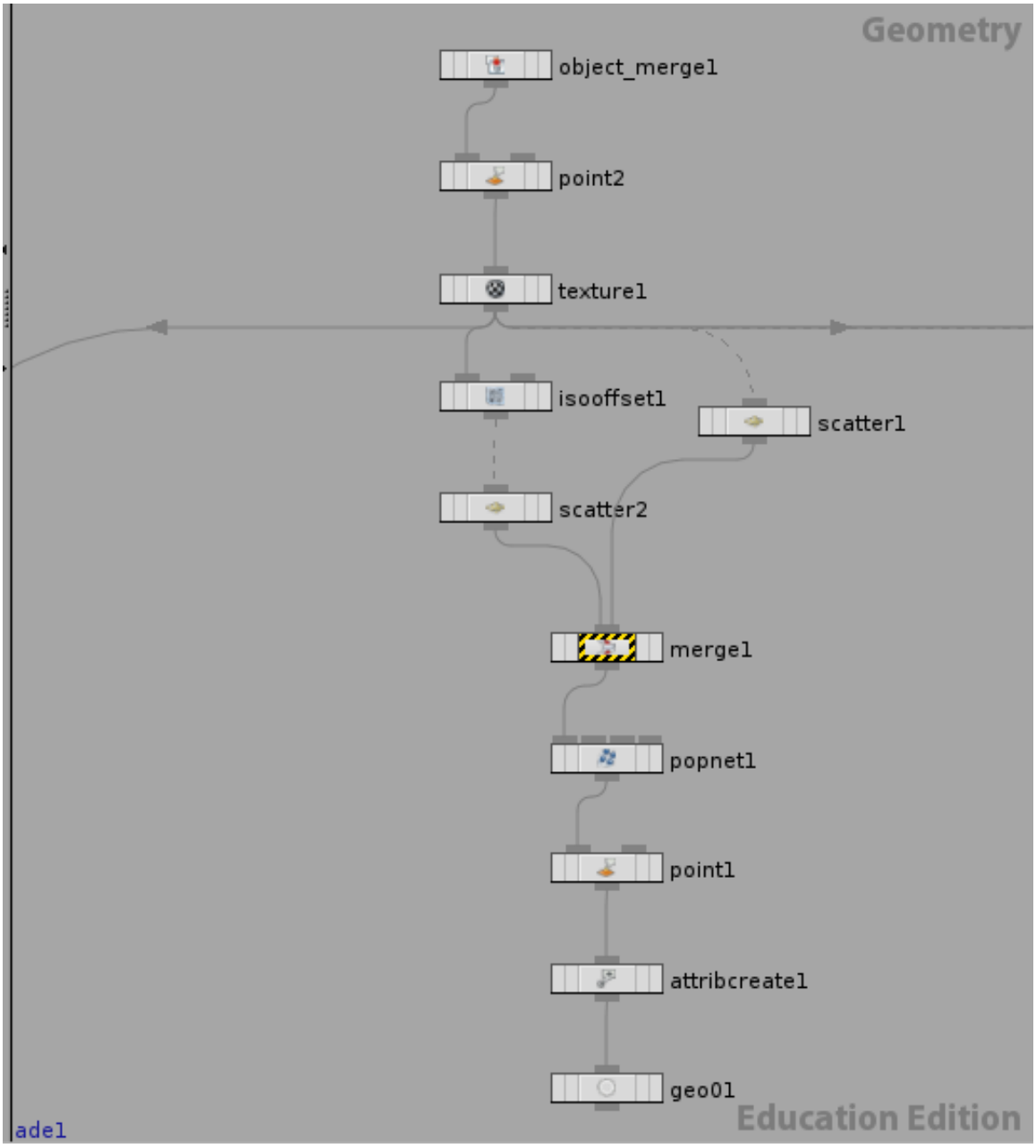

Fig7. The network of the newly created attribute.

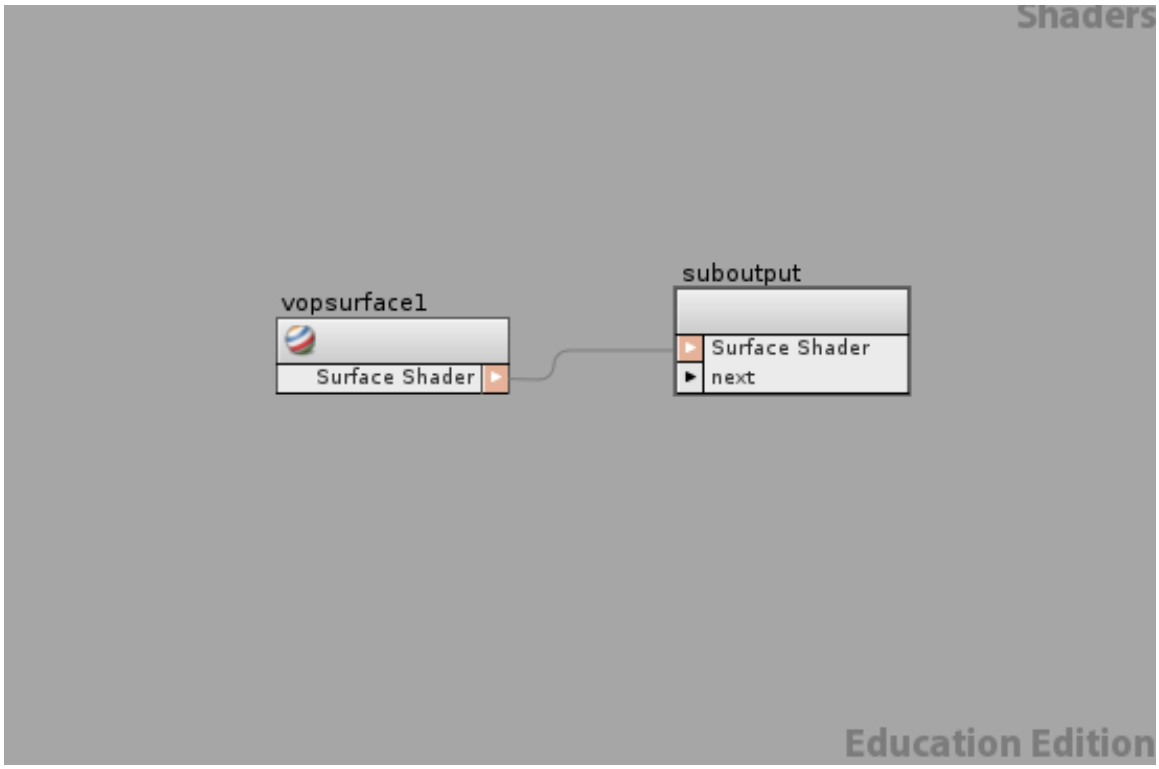

Fig8. Shader network

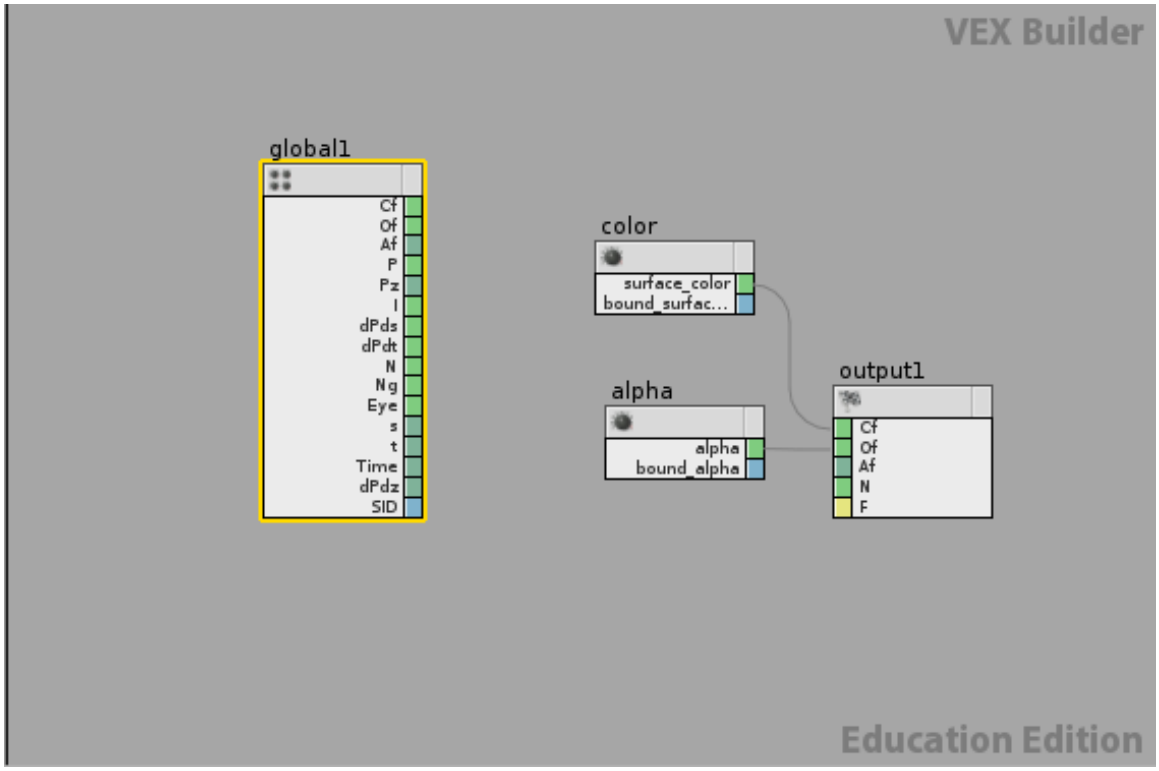

Fig9. VOP network

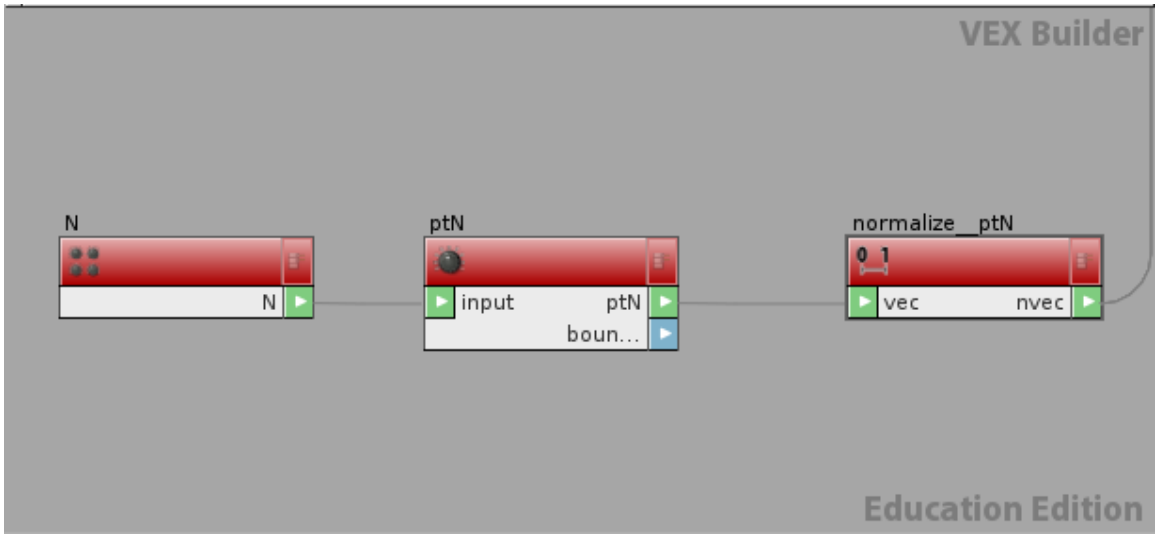

Fig10. Import the attribute of reversed normal into shader

After that, mark the “Render As Points(Mantra)” to speed up the point rendering.

CHAPTER FOUR

RESULT

Fig11 shows the still rendering images on different frames for the same simulation. The rendering shows the details of the movement as needed. Generally speaking the rendering is smooth for most of the parts. However, there are still some noticeable parts that are grainy, especially on the edge of the ink.

As it is discussed in the methodology part, the sphere emitter's radius is 1 unit. The temperature attribute of the emitter is a ramp from 0 to -0.5. The smoke container's size is 10X15X10, and its uniform division is 30. The substep of smoke solver is 1. The uniform sampling division parameter of iso-offset for the sphere emitter is 100. The number of scattered points inside the sphere emitter is 500,000. The number of scattered points on the sphere surface is 500,000. So the total number of the particles emitted in one frame is 1,000,000. The particles are emitted in the first 10 frames and they do not die. So the number of particles for one seed is 1,000,000 multiply by 10, which is 10,000,000. There are two seeds of particles in this simulation. The final number of particles in this simulation is 20,000,000.

On the MAC pro which has 2x2.8 GHz Quad-Core Intel Xeon processor and 4GB 800 Mhz DDR2 FB-DIMM memory, it takes about 38 minutes to cach out 10 million particles for one frame. It costs 1 hour to render each frame for final image using Mantra renderer. The resolution of the rendered image is 1920x1080. Image type is PNG. There is one key light in the scene. Ray-traced shadow is turned on.

The animation shows the dissipating motion and swirling motion. But the viscosity feeling needs to be enhanced.

The ink fluids are translucent; the motion of the back part of the fluid can be seen clearly. But the lighting variant is not obvious on the images.

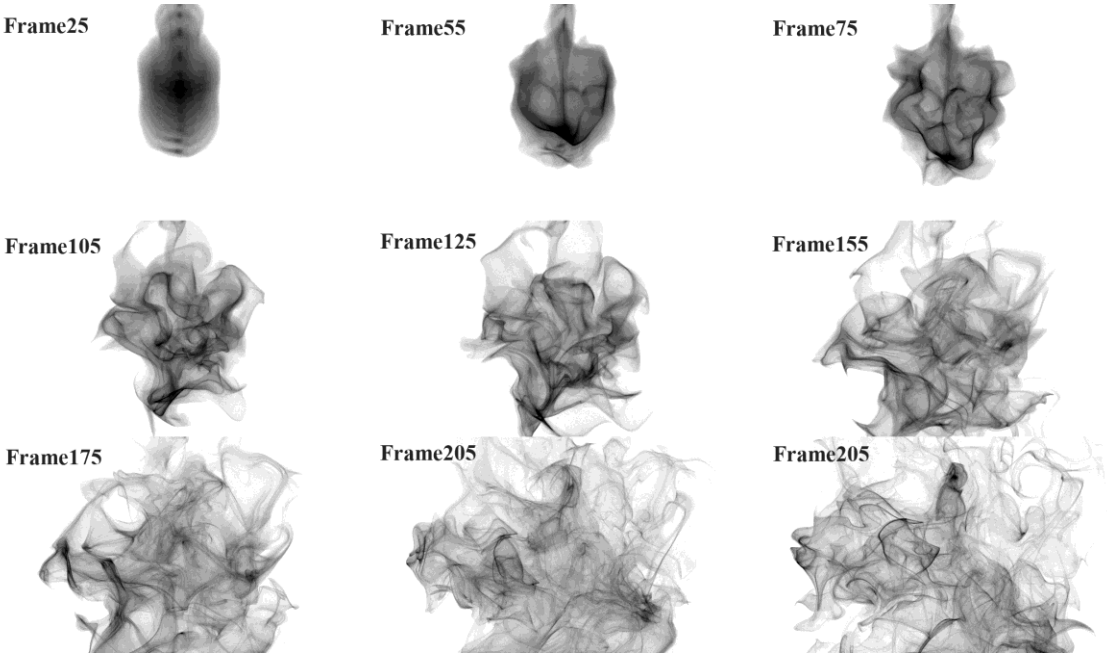

Fig 11 Rendering Result

CHAPTER FOUR

CONCLUSION

The work reported in this thesis resulted in a short animation of a simulated ink drop as it falls into the water and dissipates slowly. The animation is in slow motion, expanding the whole process and clearly showing the audience the dissipating motion of the ink drop. The image of the animation is simple but visually appealing, showing a lot of details of the ink drop.

Houdini is used in this thesis as the main production software. Houdini's fluid simulation system, particle system, and mantra rendering system form the major parts of the pipeline. Houdini's node based system makes it a lot easier to modify features according to what is needed in this project.

This kind of simulation can be applied in films, commercials, and games that require simulating small amount of fluids with high quality and a lot of detail. <Golden Fish> [15] short animation film used the same technique in this thesis. Fig.12 is the screen capture of the ink simulation in the film. The amount of particles can be adjusted according to the available facilities, time, and budget. Theoretically any production software can produce the same result using the same methodology. For further refinements in the rendering process, it would be worth investigating Renderman to render the particles. Other techniques such as adding motion blur and assigning volumes to particles can also help enhance the results.

Fig.12 The screen capture of ink simulation in <Golden Fish> short animation film.

REFERENCE

[1] ThallProducions <http://www.youtube.com/watch?v=6FECGuQku c>

[2] William T. Reeves, “Particle Systems – A Technique for Modeling a Class of Fuzzy Objects”, Computer Graphics 17:3 pp.359-376, 1983 (SIGGRAPH83).

[3] Karl Sims, “Particle Animation and Rendering Using Data Parallel Computation”, Computer Graphics 24:4 August 1990 (SIGGRAPH90).

[4] Feynman R, Leighton R, Sands M (1965) “The Feynman Lectures on Physics”, Addison Wesley.

[5] Patterson A (1989) “A First Course in Fluid Dynamics” Cambridge University Press.

[6] Jakub Wejchert, David Haumann (1991) “Animation Aerodynamics” Computer Graphics, 25:4, 1991(SIGGRAPH 91)

[7] Nick Foster, Dimitri Metaxas (1996) “Realistic Animation of Liquids” Computer Graphics Proceedings, Annual conference Series, 1997, pages 181-188, August 1997.

[8] Jos Stam “Stable fluids” Computer Graphics, pages 121-128, ACM Press/Addison-Wesley Publishing Co. 1999.

[9] Matthias Muller, David Charypar and Markus Gross, “Particle-Based Fluid Simulation for Interactive Applications” Eurographics/SIGGRAPH Symposium on Computer Animation (2003)

[10] Steinhoff J. and Underhill, D., “Modification of the Euler Equations for “Vorticity Confinement”: Application to the computation of interacting vortex rings,” Physics of Fluids, Vol.6, 1994, pp.2738-2744.

[11] Steinhoff, J., Yonghu, W., and Lesong, W., “Efficient Computation of Separating High Reynolds Number Incompressible Flows Using Vorticity Confinement,” AIAA Paper 99-3316, 1999.

[12] Hu, G., Grossman, B., and Steinhoff, J., "A Numerical Method for Vortex Confinement in Compressible Flow," AIAA Paper 2000-0281, 2000.

Houdini Documents (give a web address like number1)

[13] http://forums.odforce.net/index.php?/topic/11100-particles-like-krakatoa/

[14] http://www.fundza.com/rman_shaders/lights/directional/index.html#maya_houdini