# Controlling Fluid Simulations with Custom Fields in Houdini
Master Thesis

Peter Claes

N.C.C.A Bournemouth University
August 21, 2009

## Contents

## 1 Abstract

There are various different algorithms and techniques for fluid simulation, both physical simulation and more artistic simulation. Some of these techniques are available to the general public through existing 3d software packages. The purpose of this thesis is about developing a workflow that provides more control for artistic fluid simulation in Houdini. This workflow consists of two main elements, the development of a tool that helps with the creation of custom volumetric fields and the implementation of these fields in Houdini's dynamic context.

## 2 Introduction

Fluids and gaseous materials are hard to art direct. Physical simulation of fire and smoke can result in realistic results, but can also make it harder to control them as they are bound by certain rules. For certain shots in films a physically impossible behaviour might be required and that is when artists need to be able to truly direct where the smoke or fire needs to go and how it needs to get there. In two dimensions it is straightforward to paint strokes on a canvas to define where certain densities, colours or other properties should be. In a 3D field it becomes a lot harder to define these artistic strokes.

Sometimes it is not the movement that needs to be different, but the look. There might be a demand to mix different types of smoke together, for instance a red gas and a blue gas forming purple gas when they mix. If this colour property is linked to the behaviour perhaps only the red gas is inflammable, or perhaps they become inflammable when they start mixing, similar to chemical reactions. Because of this complexity in both look and behaviour, software that can handle this could be a useful tool. This project focuses on the development of a tool and workflow that can provide this kind control in Houdini.

## 3 Previous work

### 3.1 Introduction

There are several approaches to creating volumetric effects. The two main approaches are either using a large amount of very transparent particles as developed by Frantic Films [1] and rendering them to an accumulation buffer or using a voxelgrid, ray-marching through the volume [8]. Both approaches could benefit from additional user control through custom fields. However, most of the current solutions are either part of visual effects studios proprietary software, or programmed in isolated simulators.

Some of the fluid solvers that are incorporated in mainstream 3d applications provide not enough control to be able to create more advanced visual effects. A plug-in such as FumeFx for 3dsmax is basically a black box with little extra control beyond the provided parameters. Similarly the fluid solver in Maya is

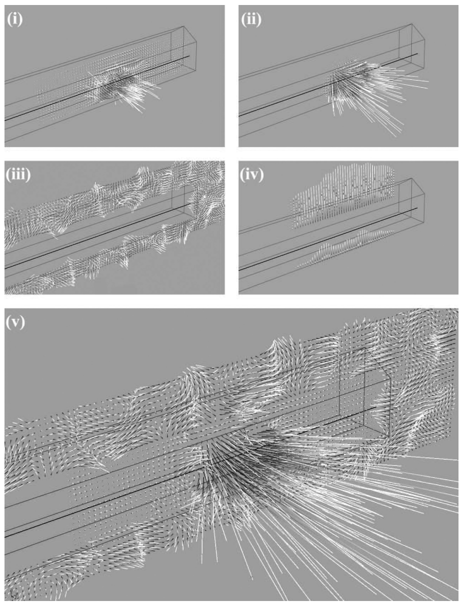


Figure 1: Separate user defined fields combined into a final velocity field, created by Rhythm and Hues using FELT. Image taken from [6]

also limited in terms of control. These solvers will provide good results quickly, but are not able to be extended or internally modified by users for more specific requirements.

## 3.2 Proprietary Systems

### 3.2.1 Rhythm and Hues' Field Expression Language Toolkit

An example of a proprietary system that helps define 3d scalar and vector fields is a tool developed by Rhythm and Hues. They create and combine user defined vector fields using "FELT", their Field Expression Language Toolkit as can be seen in Figure 1. The resulting velocity field is used to drive particles for the movie The Incredible Hulk [6].

### 3.2.2 ILM's directable, high resolution fire

ILM has created high detailed fire for the movie Harry Potter and The Half Blood Prince, using particles that were first advected by a low resolution fluid simulation and then were used to fill up field information such as temperature, velocity for high resolution 2d fields [10]. Those fields are then processed by a fluid solver solving velocity, density and temperature. The particles are projected on 2d slices perpendicular to the camera, similar to the way sprites face the camera as can be seen in Figure 2. Because the slices are 2d, the resolu-

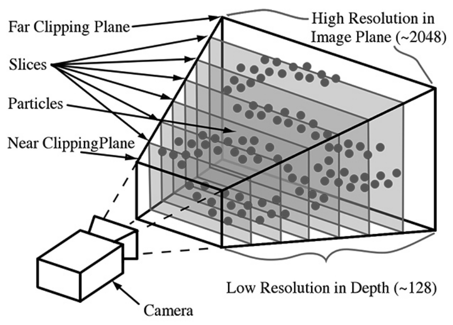

Far Clipping Plane
Slices
Particles
Near ClippingPlane
High Resolution in Image Plane (~2048)
Low Resolution in Depth (~128)
Camera

Figure 2: High resolution 2d fluid containers controlled by information from projected particles created by ILM. Image taken from [10]

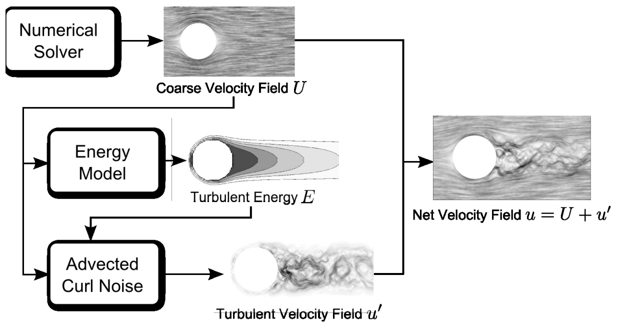

Numerical Solver
Coarse Velocity Field $U$
Energy Model
Turbulent Energy $E$
Advected Curl Noise
Turbulent Velocity Field $u'$
Net Velocity Field $u = U + u'$

Figure 3: Advecting the simulation with curl noise where turbulent energy is detected. Image taken from [14]

tion can be raised quite high. The interesting part is that the relatively small amount of particles are used as a way to define the high resolution fields. The particles represent a custom field, this hints towards the idea that the custom field does not need to have a very high resolution as some of the simulations only used 20.000 to 100.000 particles.

## 3.3 Research

### 3.3.1 Introducing noise in the velocity field

Another example of the use of a custom field to give more interesting results in fluid simulation is the introduction of curl-noise for procedural fluid flow [3]. They generate turbulent velocity fields based on Perlin noise. This type of noise is procedural but could just as well be added to fluid simulations, which is what is being done in [14] with the additional condition that the curl noise is not advecting the velocity everywhere, but only where their physically motivated energy model dictates it as can be seen in Figure 3.

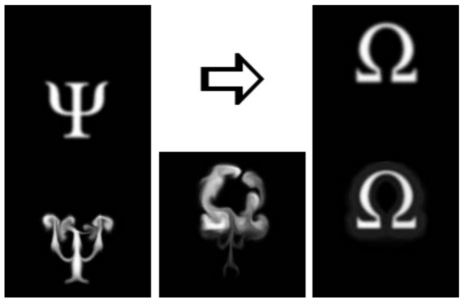


Figure 4: The different states of target driven smoke using custom fields. Image taken from [7]

#### 3.3.2 Target driven smoke animation

Custom fields can also be used to create target driven smoke animations, as in [7]. They introduce a driving force term that causes the fluid to carry the smoke towards a particular target and they also add a smoke gathering term which prevents the smoke from diffusing too much due to numerical dissipation. The core concept behind this technique is using the gradient of a blurred density field of a target shape as direction vectors to advect the velocity field of the smoke. The resulting vector field looks like all the vectors are pointing towards the target shape. Because the start and end shape might have different densities or the smoke might lose density along the way, density is added as it interpolates between the two shapes as can be seen in Figure 4.

### 3.4 A broader public

The above approaches use custom fields and geometry to gain more control over the simulation, but they are separate or proprietary software solutions. Because they are custom built they have the advantage that they will fit within the pipeline of a particular company and are able to explore new hardware platforms such as the GPU. The drawback is that these tools are not very portable and useable by a broader public.

#### 3.4.1 Node based dynamic simulations

One company that is trying to change the way fluid simulations are set up is Exotic Matter with their software called Naiad [5]. The creators are Marcus Nordenstam and Robert Bridson who have both contributed greatly to the development of fluid solvers over the past ten years. Robert Bridson has published several insightful papers and also the book Fluid Dynamics for Computer Graphics [2]. At its core Naiad is a dynamics solver and a simulation framework. Naiad allows for the creation of a description of a simulation scene. This description can then be simulated. The description can be made through a node based interface which resembles that of node based compositing software as can

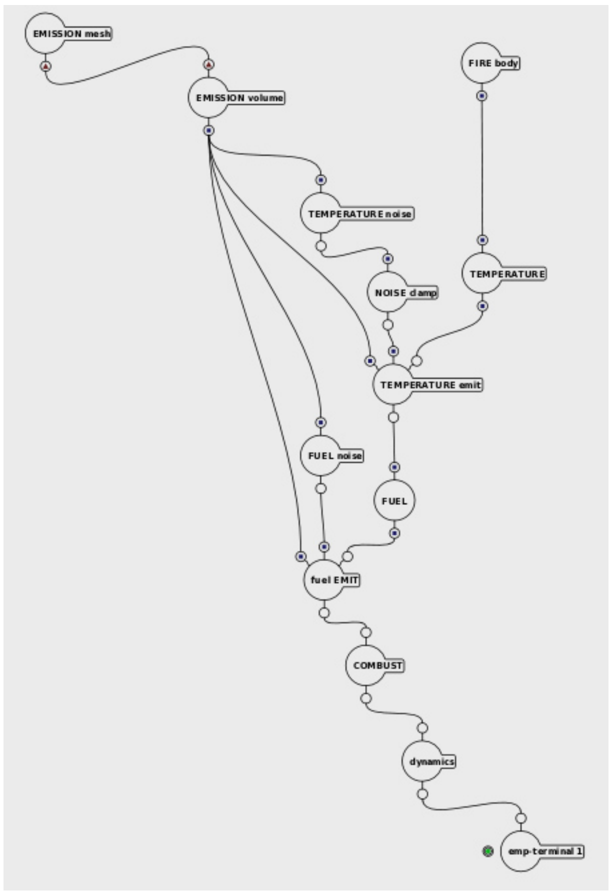


Figure 5: The node based operator graph in Naiad, by Exotic Matter. Image taken from [5]

be seen in Figure 5. They are trying to establish a "Nip" (Naiad Interface Protocol) format which is similar to a RenderMan RIB format, but then for dynamic simulation. This has not been done before and will hopefully help establish a standard for dynamic simulations. This is also one of the few systems which uses a node based approach for fluid dynamics, the other major node based dynamics system is part of Houdini.

### 3.4.2 A brief history of Houdini's dynamic context

Since Houdini 8 node based dynamic operators (DOPS) were introduced and since Houdini 9 fluids and gas microsolvers were added. Houdini 10 brought a further development of a preset pyrosolver which can produce high quality pyrotechnic results combined with an extensive pyroshader. They also introduced

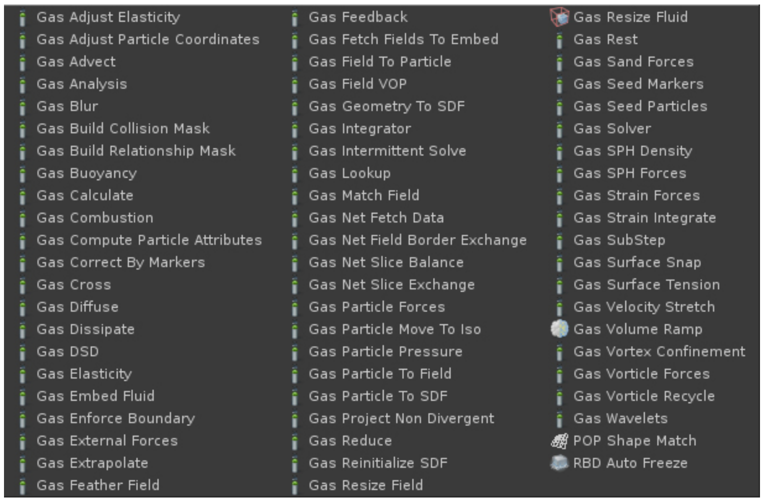


Figure 6: The various microsolvers available in Houdini's dynamics context.

an upres technique which greatly increases the detail in the final high resolution result. To handle these high resolution simulations they allow distributed simulation over a network of computers. These developments and techniques will be explained in greater detail in the next section as they are important to understand the proposed workflow and tool.

## 4 Technical Background

Houdini's node based architecture is open and not a black box. For example the preset pyrosolver is a digital asset, which means it can be unlocked, modified to support extra functionality and saved as a new custom solver. Therefore Houdini is very well suited to get more control over fluid simulations. However in order to determine where to insert extra functionality and make modifications an explanation of some of the current fluid techniques and their implementation in Houdini is required. Some of the equations of fluids will be explained for completeness and some of the equivalent microsolver networks will be shown alongside.

### 4.1 What are microsolvers?

Microsolvers are nodes inside of DOPS that perform specific tasks related to the fluid solving process. Instead of solving various aspects of a fluid simulation in one big solver node as is done with the preset smoke- or pyrosolver, a microsolver performs only a specific mathematical task. By wiring these microsolvers together more complex operations can be performed, this is how the bigger smoke and pyro solvers were created. Some microsolvers are digital assets, which means they are built from a network of smaller microsolvers inside. There are currently around 60 microsolvers in Houdini 10 as can be seen in Figure (6). It is beyond the scope of this thesis to cover all of them so instead a few of the more important ones of which the equivalent often shows up in fluid equations will be covered. Microsolvers are wired together by using a merge dop, the order of the inputs is very important as it will define which microsolver contributes to the final result first. The order of operations works from left to right, top to bottom. This is quite different from the pure top to bottom node flow inside of the surface operators context (SOPS).

## 4.2 How do volume fields work in Houdini?

### 4.2.1 Defining scalar and vector fields

Volume fields in Houdini come in two types: scalar fields and vector fields. They are defined as primitives in SOPS (surface operators) but can also be created in DOPS (dynamic operators) and represent an attribute of the volume, the most commonly known is the density scalar field. Other custom fields can be defined using a volume sop which will define an empty volume primitive or an iso-offset sop which will define a volume based on the closed mesh of an input geometry. One of the more powerful ways to define custom volumes is by using the volumevop. This runs CVEX (Houdini's vertex expression language) over a set of volume primitives, the operations can be defined through code or by building a CVEX VOP (vertex operators) network. Not only do you have a lot of low level mathematical functionality, using VOPS will compile the resulting network in vex code which will make it a lot faster to run in comparison to normal expressions which would have to be interpreted. This volumevop is used as part of the tool that was developed to define custom fields in SOPS.

In DOPS fields can be created using a (SOP) Scalar Field node or a (SOP) Vector Field node. The (SOP) part allows you to reference fields from SOPS, but sometimes temporary fields are needed to store results of calculations and the field is created without the need to reference it from SOPS as it will be filled up with data that is calculated from other fields in DOPS. To quickly create an empty field based upon the properties such as the dimensions and resolution of another field, a Gas Match Field microsolver can be used.

A scalar volume field such as density will contain a single primitive, whereas a vector field such as velocity will contain 3 primitives, one for each component. By convention the velocity field is named "vel", the components are vel.x, vel.y and vel.z.

### 4.2.2 Naming volume primitives

It is important to give the correct names to scalar fields before merging them into a single vector field. Some fields are recognised by the preset shaders, such as the color field (Cd), but are not yet supported by the preset fluid solvers. By modifying the preset shaders you can have custom fields influence the look and create extra output variables that can be rendered to separate image planes for greater control in a compositing application. Another useful field to have as part of the simulation is the rest field, which basically carries a set of uv coordinates that can be used to map textures to the fluid.

Fields can be given any name, but only certain specific fields will be picked up by the Mantra render engine, for example if the "vel" field is detected and motion blur for rendering is turned on, the information of the velocity field will be used to apply motion blur on a fast moving fluid simulation.

### 4.2.3 Rest field interpolation

In the preset smoke solver a single rest field is advected along with the other fields. This rest field is then used inside a volume shader to map various kind of noise to the fluids and increase the detail of the smoke as is also described in [17]. When this rest field is advected far away from the initial position, the textures will become distorted which can give undesirable results. In the new pyrosolver in Houdini 10 a second rest field (rest2) is introduced to solve this issue. The initialization of both fields (rest and rest2) is staggered in time and the pyro shader continually interpolates between them. For example if the reset value on the pyrosolver is set to 50, the "rest" field will be re-initialized every 50 frames, starting at the first frame. The "rest2" field will also be re-initialized every 50 frames, but will start at frame 25 (half of the reset value). The pyro shader has a corresponding Reset Rate parameter that should be set to the same value. This will cause the shader to interpolate between the two rest fields without showing any popping when the rest fields are reinitialized.

```
rest  : +----------------+-------------------+------------------  (etc)
rest2 : -------+-------------------+-------------------+--------  (etc)
```

At each + in that graphic, the field resets. The pyro shader then constantly cross-fades between the two fields, such that a given field has full contribution only at the reset frame [12].

### 4.2.4 Volume field viewport visualization

Within DOPS there are different way to visualize the information stored in a field. This visualization can greatly help to visually debug the values of a field. To be able to enable this visualization for a specific field, a Scalar/Vector Field Visualization node needs to be attached to the data input of a field. This visualization can consist of:

- showing the field as smoke (useful for density),
- as straight lines depicting both length and direction of the values of a vector field (generally useful for a static or dynamic velocity field),
- as streamers, which are curves depicting the flow of a vector field, (useful for viewing the interpolation of the velocity between voxels)

- as an axis aligned plane that shows a coloured slice in the volume, mapping the values of the field to the colours on the plane. (useful for temperature)

Within SOPS a volume will generally be visualized as smoke when the values of the field are positive. This means as well that when you are trying to visualize a field that has negative values, such as velocity or temperature, nothing will be shown in the viewport in SOPS, even though the data is actually there. A quick check can be performed by taking the absolute value of the data inside a volumevop just for visualization purposes to make sure the data exists. At this point visualizing your custom fields in dops will give you more options as described above. The reason why this section on visualizing volumes in SOPS is included is because it is not in DOPS, which means no solving needs to happen to visualize the field which makes it faster to shape volume fields independent of the previous frames. Also it is important to understand negative values are not displayed in the viewport as this clarification will hopefully avoid confusion later on when the tool is used and there "seems to be no output".

#### 4.2.5 Houdini specific training

Sidefx, the creators of Houdini, provide a great set of masterclass tutorials on fluids and the pyrotools on their website which will help when learning how to use microsolvers and understand volume fields in Houdini [15].

## 4.3 The equations of fluids

A significant amount of research has already been done in the area of computational fluid dynamics (CFD) for simulating smoke, fire or various types of liquids. A few of the equations will be covered to help explain the components that make up a simple smoke solver in Houdini.

### 4.3.1 The incompressible Navier Stokes equations

There is a consensus among scientists that the Navier Stokes equations are a very good model for fluid flow [17]. That is why they tend to be used as a foundation for a lot of solvers, that will then in some way modify or extend these equations:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla \rho = \vec{g} + v \nabla \cdot \nabla \vec{u} \qquad (1)$$

$$\nabla \cdot \overrightarrow{u} = 0 \qquad (2)$$

Where:

| | |
|---|---|
| $\vec{u}$ | velocity |
| $t$ | time |
| $\rho$ | pressure |
| $g$ | body forces |
| $v$ | viscosity |

And (1) is called the momentum equation and tells us how the fluid accelerates due to the forces acting on it and can be derived from Newton's equation $\overrightarrow{F} = m\overrightarrow{a}$ as explained in [4]. And (2) is the incompressibility condition and states that the divergence of the velocity must always be equal to zero. This constraint ensures that the sum of all velocities entering and leaving at a point in space is zero. When this is conserved, the total mass is conserved over the entire simulation [18].

### 4.3.2 The steps to solve the Navier Stokes equations

Three main steps will be required to solve the Navier Stokes equations, these steps are very well described in [18] but will be included here again for completeness and some of the explanations of the terms summarized.

1. Advection
2. Diffusion and External forces (Body forces)
3. pressure/incompressibility

**Advection**
$(\overrightarrow{u} \cdot \nabla)\overrightarrow{u}$

Advection represents the fact that the motion of the fluid causes motion of the entities within it. This can be thought of as the velocity of the fluid moving itself along, and is sometimes referred to as self-advection because of that property. The advection part of the equations can also be used to model motion of other entities inside the fluid [18].

Within Houdini the Gas Advect microsolver can be used for advecting one or several fields at the same time. For more advanced interactions between fields, other actions can be performed at this step that will update the fields. This can be seen in the node tree for the fields_updates section inside the preset smoke solver in Figure (7). For instance:

- A Gas Blur is used to diffuse the temperature field,
- A Gas Calculate is used to cool the temperature field or to copy information from a source field into a heat field.
- Fields are not the only thing that can be advected, geometry can as well, such as vorticle geometry which is explained in the external forces step.

**Diffusion and External forces**

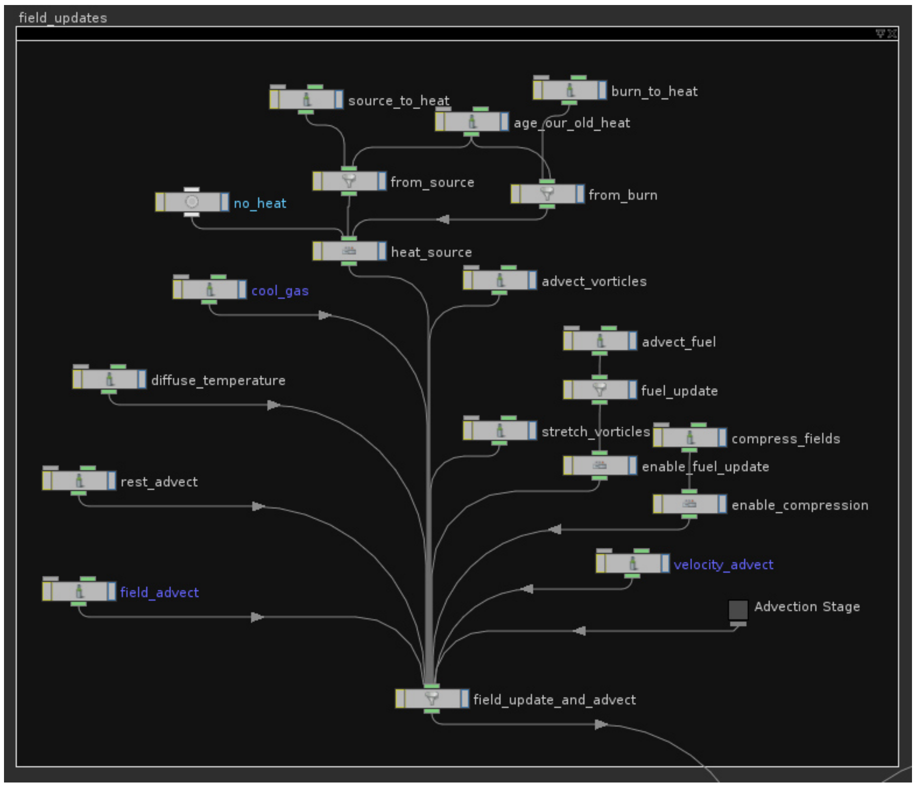


Figure 7: The node tree for the update fields section in the preset smoke solver in Houdini as described in (4.3.2)

**Diffusion**
$v\nabla \cdot \nabla \vec{u}$

The diffusion allows the velocity to propagate outwards from its current location, with the viscosity parameter controlling how fast this happens. High viscosity yields thick and slow fluids while, while a low viscosity yields lively fluids [18].

Within Houdini the viscosity can be implemented using a Gas Diffuse microsolver. This outwards going velocity is a modification to the velocity field.

**External forces**
$\vec{g}$

This is considered as the sum of all external forces. Some typical forces are wind, drag or gravity, but also include more advanced forces that require separate simulations, such as the forces generated by temperature differences. These external forces can be extended and provide a lot more control by adding user defined volumetric fields. This is one of the areas that will be extended with custom fields defining them in SOPS with the custom tool that is developed.

Within Houdini these forces can be represented by a variety of Gas Microsolvers, that create the total resulting force when merged together. Typical microsolvers that are part of the preset smoke solver for the forces component are the following and can be seen in the node tree of the forces section of the preset smoke solver in Figure (8):

- Gas Vortex Confinement: to re-introduce some of the lost velocity due to numerical dissipation, a high vortex confinement force will introduces a lot of swirling motion in the simulation,
- Gas External Forces: both for external forces relative to density or absolute. When relative it will scale the velocity by another field, masked by the density, when absolute it will simply scale the velocity by another field without the mask,
- Gas Buoyancy: to calculate an approximate buoyancy force dependent on a temperature field,
- Gas Diffuse: for the viscosity force, as explained in the diffusion step above,
- Gas Vorticle Forces: for introducing extra swirling motion around each vorticle, according to the vorticle attributes. By default when adding vorticles, the vorticles are particles that exist throughout the entire volume and that are advected by the velocity,
- Gas Calculate: for setting minimum and maximum speed limits if required.

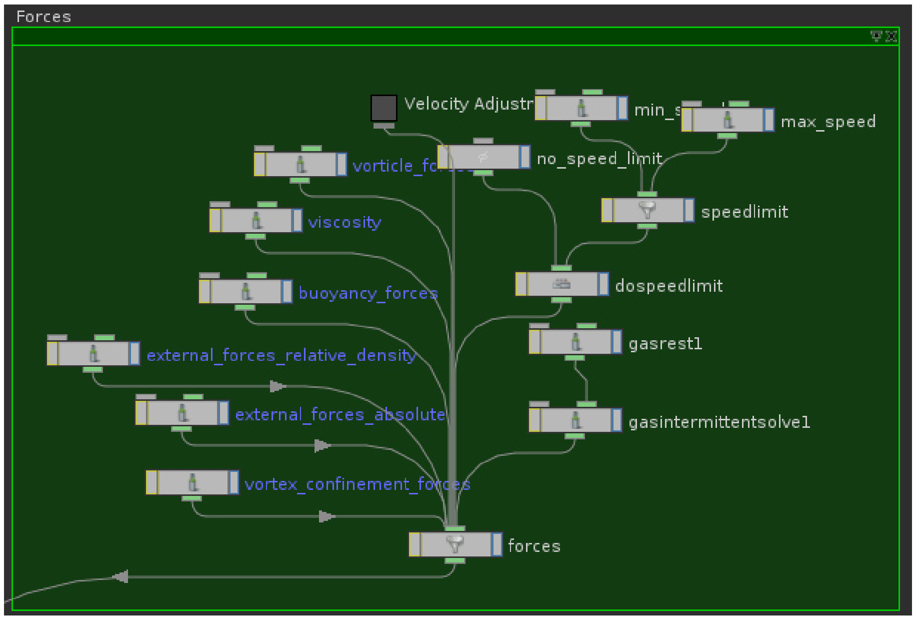


Figure 8: The node tree showing the body forces inside the preset smoke solver in Houdini as discussed in (4.3.2)

#### Pressure and Incompressibility (non-divergence)

**Pressure**

$p$

The pressure can be considered as the fluid from the area with high pressure that will be pushed by the pressure towards the area with lower pressure. This force is represented by the gradient of the pressure field, $p$ [18].

**The incompressibility condition**

$\nabla \cdot \overrightarrow{u} = 0$

Another way to think about pressure is that it is whatever it takes to keep the velocity divergence-free so the incompressibility condition is satisfied [4].

Within Houdini these terms come in the shape of a Gas Project Non Divergent microsolver that removes any divergent portions of a velocity field. These are parts of the velocity field that represent expansion or contraction. This is done by computing a pressure field that counteracts any compression and applying that pressure field instantaneously [16].

### 4.3.3 Vorticity Confinement

When simulations of stable fluids are run, some of the velocity is lost due to numerical dissipation which is the results of a necessary weighted averaging step during the advection term. It has the effect of unintentionally adding viscous behavior to the fluid, which can damp down some of the intricate turbulent behavior seen in natural smoke and fire[13]. In [8] a method is described for detecting the vorticity and adding the rotational forces back in. Each small piece of vorticity can be thought of as a paddle wheel trying to spin the flow field in a particular direction. Artificial numerical dissipation damps out the effect of these paddle wheels and the key idea is to simply add it back.

The vorticity confinement calculation is a good example in Houdini of how some of the microsolvers and fields can be used together. A Vortex Confinement microsolver exists, but it is really a digital asset made out of smaller microsolvers.

These are the steps that are executed within the Vortex Confinement microsolver and the accompanying node tree can be seen in Figure (9):

1. Create the fields for temporary results: curl (vector), curl magnitude (scalar) and vortex direction (vector) using Gas Match Field microsolvers.

2. Calculate the curl of the velocity field with a Gas Analysis microsolver and store the result in the curl field.

3. Calculate the length of the curl field and store it in the curl magnitude field using a Gas Analysis microsolver.

4. Calculate the gradient of the curl magnitude field and store it in the vortex direction field using a Gas Analysis microsolver.

5. Normalize the vortex direction field using a Gas Analysis microsolver.

6. Calculate the cross product between the vortex direction field and the curl field with a Gas Cross microsolver and store the result back in the vortex direction field.

7. Multiply the vortex direction field by a scalar to increase the amount of confinement using a Gas Calculate microsolver.

8. Multiply the vortex direction field by a confinement field to increase the amount of confinement only at specific location within the field using a Gas Calculate.

9. Update the velocity by adding the vortex direction field to it using a Gas Calculate microsolver.

10. Clear the temporary fields by copying 0 into them using a Gas Calculate microsolver.

From the above steps it becomes clear that the Gas Calculate, the Gas Analysis and the Gas Match Field are very useful microsolvers, and will come back all the time. They are some of the mathematical building blocks to build more complex operations. While developing these operations the visualization of the resulting steps can be very helpful as shown in Figure (10).

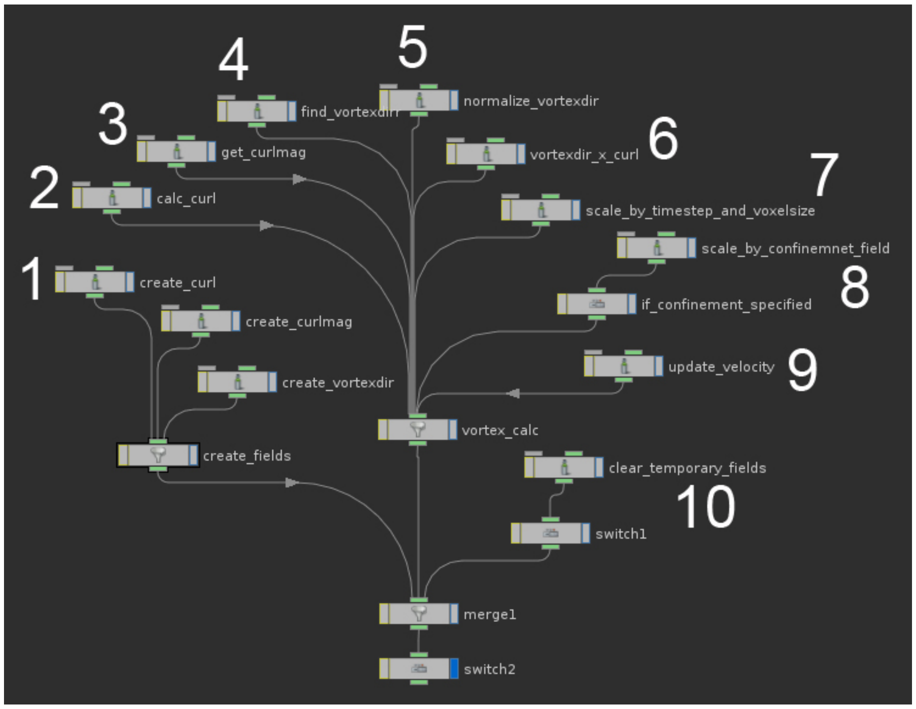


Figure 9: The node tree in Houdini for the Vortex Confinement microsolver showing all the steps as described in (4.3.3)

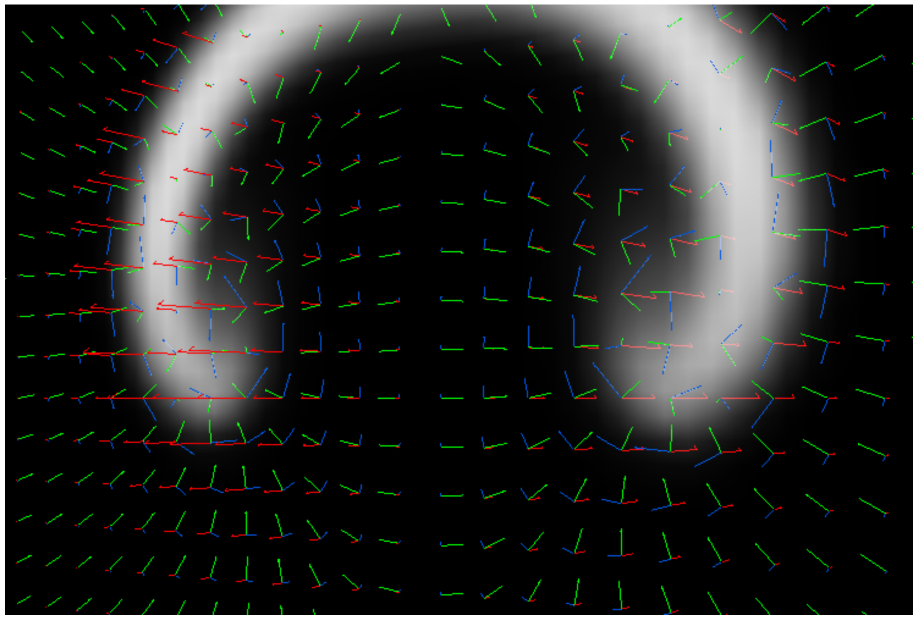


Figure 10: The geometric representation of the steps inside the Vortex Confinement microsolver as described in (4.3.3). Red represents the curl field after step 2, green represents the gradient of the curl magnitude field after step 5, blue represents the vortex direction after step 8.

## 4.4 Up Res technique

In Houdini 10 a new technique was introduced were a low resolution simulation can be used as the basis for a higher resolution simulation. The key concept is that you get the general motion from the lower resolution simulation and introduce high frequency noise into the velocity field, but only where there is a high curvature. The introduced noise has a low amplitude so it does not interfere with the general motion coming from the low resolution velocity field. A cut down version of the pyrosolver is used for the Up Res phase as a lot of the information is coming from the low res fields rather than having to be calculated from scratch. Because it is still a high resolution simulation it can take a long time. The foundations for this Up Res technique were initially built in Houdini 9.5 based on [11]. It then became a fully integrated tool supporting Constant, Wavelets and Curl noise. The Gas Up Res solver is also a digital asset, which means it can be modified to support custom user defined fields coming from the low res simulation or added in to support a certain type of control or new type of noise. Those fields can then be up-ressed and written to disk at the end of each simulation step.

## 4.5 Distributed simulations

Running high resolution simulations can take a lot of time. Therefore the functionality has been built in to be able to split a heavy fluid container into slices. These slices represent partitions of the volume. The amount of overlap between the slices that is required for a good distributed simulation depends on the speed of the fluid and needs to be set by the user. There are several ways the slices can be defined, but by default they will partition the space inside the container similarly to a binary tree. Each slice can then be calculated on a different machine that is connected to a host machine that is running a tracker. The machines share the data between them in the overlapping regions through inter process communication. The speed gains are significant, but not linear as the extra overlap and communication between machines takes up a bit of additional processing power. The simulation generally runs as fast as the slowest machine. If all the machines have the same specifications this is not a problem, but if one of them is significantly slower than the others, the faster machines will not be used as efficiently.

Sidefx has provided Hqueue, a python based renderfarm software to help with this distribution. However HQueue was not tested for this project due to limited administrative permissions. Instead a few shell scripts were written that remotely start a tracker and the simulations and make use of the exact same distribution techniques. Again it is convenient that open access is provided to the distribution techniques without forcing users to use HQueue.

# 5 A workflow for using and creating custom volume fields in Houdini.

## 5.1 Introduction

The aim of this project was to get more and better control over fluid simulations using custom fields. To be able to have this kind of control the problem is approached from two sides. The first side is by defining custom fields inside of SOPS, the second side is by using microsolvers to integrate those custom fields with existing solvers and perform certain mathematical calculations to solve the required needs inside of DOPS.

These custom fields are defined in SOPS and brought into DOPS. In order for them to work correctly they need to be inserted in a fluid solver at the right place, some of this has been covered in section (4). Custom fields can add any kind of data to the simulation. For example: add color, temperature, velocity, fuel, density, etc. Some of these attributes are already supported by the preset solvers and can be easily modified. Others will need extra implementation through the use of microsolvers. There has already been done a lot of work in the field of computational fluid dynamics and some of the preset solvers in Houdini cover a lot of this functionality. The aim of this project is not to create a full fluid solver with a combustion model from the ground up, but rather to be able to modify and augment existing solvers.

By controlling the fluid simulation through custom fields, depending on what exactly is modified, some of the physical accurateness of the simulation will be lost, but that is the whole point as by introducing custom fields (art-)directable simulations are possible. The goal of this project is not to build a physical simulator, but rather to understand a physical simulator and then insert extra fields into it to augment the possibilities of the solver. Ideally the fluid flow and turbulent behaviour is maintained within the modifications, but if certain fields will be overwriting other fields because this provides better visual results then this is considered an improvement.

## 5.2 The implementation of an attribute transfer tool in SOPS to help define custom volume fields.

There are a few microsolvers inside DOPS that will allow to bring data from SOPS into DOPS already:

- The Sop Scalar/Vector field microsolver which can bring in a fog volume that has been defined in SOPS through an iso-offset sop.
- The Gas Particle To Field microsolver allows point attributes from particles to be copied into a field, after the particle geometry has been defined inside a SOP Geometry dop. This microsolver allows for most of the required functionality, but it is in dops which requires the simulation to cook up to the current frame as soon as a change is made.

- The final option is with a Gas Field Vop which provides all the functionality from vops so you can reference in any object from a different context, or from a file on disk.

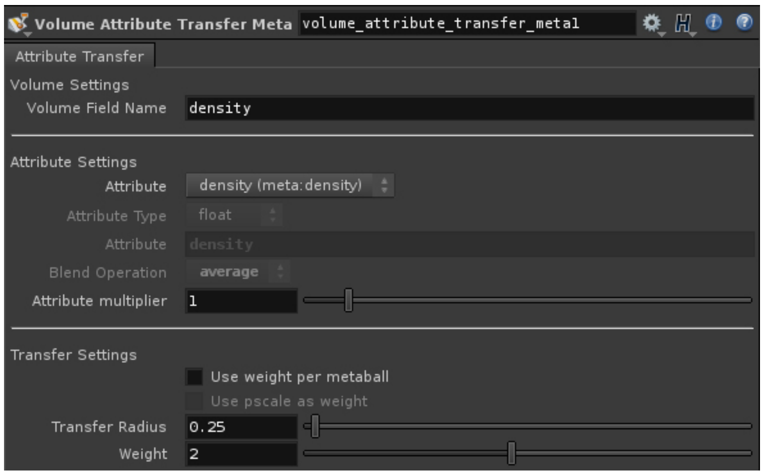
In SOPS however this attribute transfer from points to volumes does not exist and is a welcome addition as it does not have the requirement to simulate all the previous frames to define the field. Also the resulting custom volume fields can easily be used to advect particles inside of a pop network by using an Advect By Volumes pop without even going near DOPS. The reason that this just works is because the tool follows the standards used by Houdini to define volume fields as described in (4.2).

### 5.2.1 A pointcloud based approach

The first approach to transferring point attributes from a number of points makes use of pointcloud functionality inside of a volumevop in sops. The underlying technicalities are hidden away behind an easy to use interface. I deliberately kept this digital asset compact as it needed to perform only a single function. The first input requires an empty volume, the second input requires at its lowest level a number of points. The volume field name can be specified, as well as the attribute to transfer and the attribute type. The attribute type will also define the type of volume that is created (either a scalar or vector field). The distance threshold defines how big the influence radius of each point is. In the pcloud tab a maximum search radius is provided to prevent voxels looking up values from points outside this radius. Also the maximum number of points to use when performing the filtering operation can be set here. Generally the default values work for most scenarios.

Internally the asset creates a box that matches the position and divisions of the voxel grid. Then an attribute transfer sop is used to transfer the attribute to the grid of points. The grid is used instead of using the points directly because the attribute transfer sop uses a metaball kernel inside to define the falloff of the attributes and give softer results around the edges of a shape. These points are then referenced by a volume vop which will read them as if they were a pointcloud from disk. For each voxel it will then filter the attribute value and assign it to the output density. The output density is just a name inside the volume vop, it could just as well be the x component of a velocity field. The real naming of the volume occurs before or after the volume vop assigns the values to the field. In case of a vector field and a vector attribute each individual component is handled at a time and then the resulting scalar fields are merged together to form the resulting vector field.

This gives quite accurate results, but can get slow for high resolution volumes, as the attribute transfer sop can take a while to cook. Bypassing the attribute transfer and using the points directly will definitely speed it up, but does not give as nice results, this bypass is triggered by turning on the Use Raw Points checkbox on the digital asset interface in the pcloud tab. Also this approach does not allow a way to define a transfer radius per point, only a global radius can be used.


Volume Attribute Transfer Meta volume_attribute_transfer_meta1

Attribute Transfer

Volume Settings

Volume Field Name: density

Attribute Settings

Attribute: density (meta:density)

Attribute Type: float

Attribute: density

Blend Operation: average

Attribute multiplier: 1

Transfer Settings

Use weight per metaball

Use pscale as weight

Transfer Radius: 0.25

Weight: 2

Figure 11: The parameters of the metaball-based volume attribute transfer tool.

### 5.2.2 A metaloop based approach

The original way of defining volumes through metaballs involves copying metaballs onto the points, converting them to a fog volume using an iso-offset with the Mode set to Meta Balls. The problem is that this only uses the density of the metaballs and none of the potential other attributes.

The volume attribute transfer tool is based upon this workflow but again makes use of vops functionality to perform the lookup of the attributes manually. Similar controls are provides as with the pointcloud approach, although this tool was further developed and presets were added to make it easy for users to transfer popular attributes that can be used from a drop down menu. The interface of the tool can be seen in Figure (11). This tool also allows the possibility to define a per point transfer radius by using the pscale attribute if it is present on the template geometry.

Inside of the digital asset, expressions are used to help define the preset settings. Metaballs are copied onto the template points and are then referenced by a volumevop. Inside of the volumevop a while loop is created that will loop through all the metaballs, lookup the value of each one at a given position and add the result together. This is the default density calculation. When trying to add a different attribute together the results were very blocky, the reason why this was happening was because the attribute would be defined as a solid box of a constant value within the bounding box of a metaball. This is a problem very similar to 2d sprites that have not had there alpha channel premultiplied. So the solution was to multiply the attribute with the density value of the metaball, this problem is represented in Figure (12). The density needs to be multiplied inside the deepest level of the metaloop, just before the attribute is added to the result of the previous metaball, not as a post multiplication operation because in overlapping areas the masking will be incorrect, the post multiplication effect

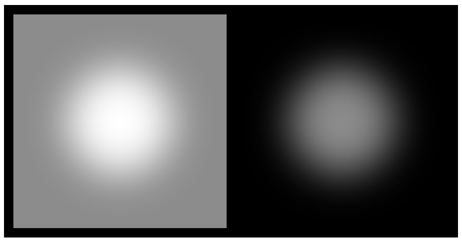


Figure 12: Premultiplication problems: The image on the left shows the attribute in grey and the density of the metaball in white.The image on the right shows the correct premultiplied values.

is shown in Figure (13), the artifacts are not that visible in a still frame but will cause jittering during animation.

At this point the values are simply added together, which works well for density, but not so well for velocity or color. For example in the case of particles being emitted from a single point in space and then spreading out from that point over time, what would happen is the density would be accumulated and look good, but because of the dense number of particles at that single point, the velocity field would grow extremely large and when this entire resulting volume is then rendered by mantra with motion blur turned on, huge streaks are created due to the high velocity accumulation.

So more control was needed to define the kind of mathematical calculation that takes place. Therefore an averaging operation and a switch is added, so the user can choose whether to accumulate the values or to average them. This is not trivial and this choice should be provided to the user. A temperature field for instance might require either accumulation or averaging, both could work and will give different results. The presets try to alleviate this issue for some of the popular attributes based upon what works better for those attributes in most scenarios so the user does not need to worry too much about this. However if the accumulation of velocity attribute were required, the user can set the preset to “other” and define “v” as the attribute, “vector” as the type and “add” as the blend operation as the “other” preset allows any custom attribute to be used.

Because the copy sop inside of the asset transfers the template point attributes, the special attribute “pscale” can be used. When a pscale attribute is defined on the template points, the metaballs will be scaled based on the pscale value. This tends to give good results, however when the pscale goes to zero, jittering in the volume might occur if the points are moving. It would be as if the small metaball would only occasionally be picked up by a voxel when it is in close enough proximity. A better way of dealing with this problem is by using the weight attribute instead of the pscale. This will not modify the radius, but will instead result in more of a fading effect. The best results can be achieved when both are used together.

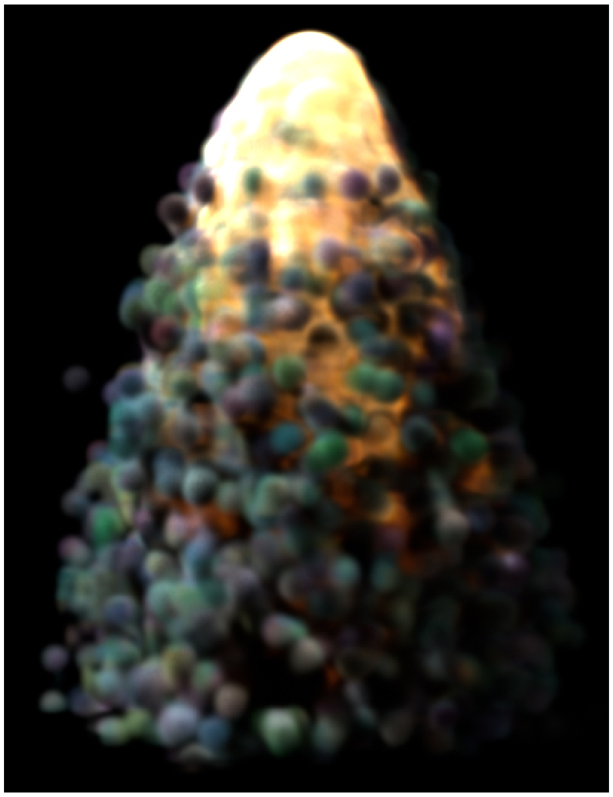


Figure 13: Incorrectly post-multiplying of the attribute (temperature) by the density will cause artifacts, note the banding in the brighter areas, that become more apparent in an animation.

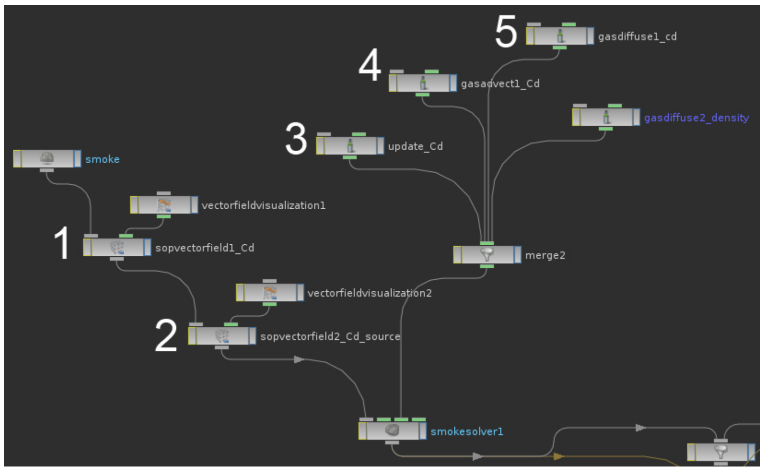


Figure 14: The preset smoke solver with the color field added to it, as described in (5.3.1)

## 5.3 Examples of custom fields

The next sections will talk about how some setups can be created and what is required to implement the custom fields in dops for each example.

### 5.3.1 Colour field addition

Once the colour (Cd) attribute is transferred onto a volume in SOPS using the volume attribute transfer tool, the necessary fields can be added in DOPS, the node tree can be seen in Figure (14) :

1. A Sop Vector Field, Cd, is defined to bring in the newly defined Cd field from SOPS. This field will contain the color information throughout the simulation.

2. Another Sop Vector Field, Cd_source, is defined using the Cd field from SOPS. This field will be used to add color information to the Cd field at the beginning of every frame. If this field is animated, it is required that the “Time” and “SOP Path” are always evaluated. The “Time” parameter should be set to $T.

3. Using a Gas Calculate with a the calculation set to “Maximum”, the destination field set to “Cd” and the source field set to “Cd_source” will copy information from the source into the destination field at the start of every frame using a maximum operation.

4. Advect the color field by the velocity, using a Gas Advect microsolvers

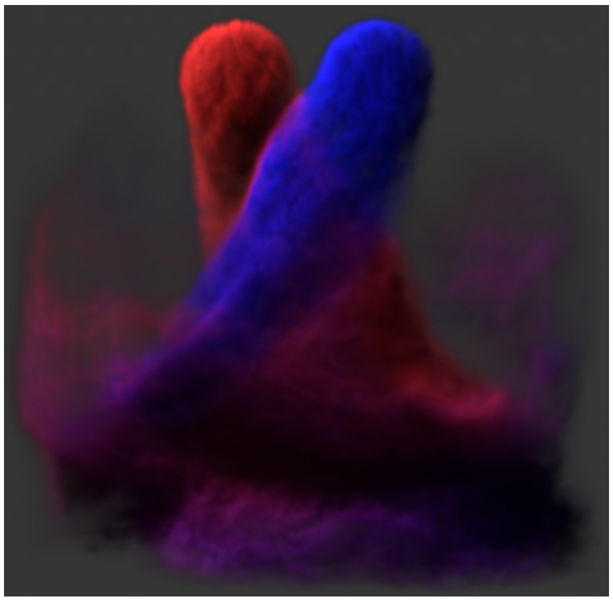


Figure 15: Two spheres emitting density and color with a vortex force applied.

5. Diffuse the color field using a Gas Diffuse microsolver, the diffusion rate should match the density diffusion rate. For fast mixing of the color, this value can be increased.

In Figure (15) two spheres can be seen emitting density and color information into a simulation. This simulation has the color fields defined as copies of the density values for either the red or the blue component of the Cd vector field and does not actually require the transfer tool in SOPS. In Figure (16) you can see a helix curve emitting density, colour and the temperature field is driven by the red values of the colour which causes the red smoke to rise quicker. The hue of the colour is mapped to the parametric u value of the curve.

#### 5.3.2 Advect by curve

In this example the tangent vectors of a curve are used as an extra force that is continuously added to the velocity field. A relatively large amount of force is applied. Also the density field is temporarely blurred, the gradient is taken from this blurred density, then normalized and applied as an additional force. The combined force points towards the density of the curve and along the tangent of the curve. A sphere is placed at the bottom of the helix and is emitting density into the simulation. The density and tangent velocity fields can be seen in Figure (17).

The node tree can be seen in Figure (18) and consists of the main following areas:

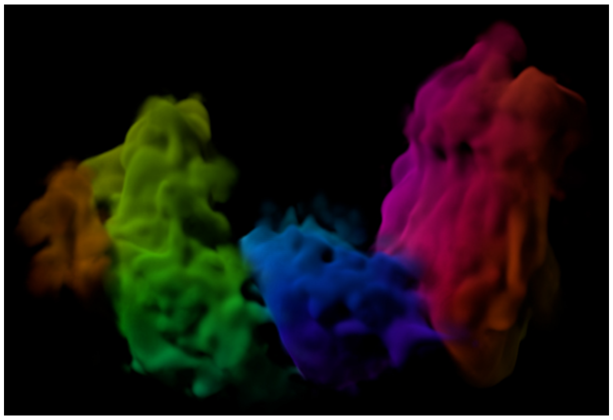


Figure 16: A helix curve emitting density, colour and temperature into the simulation. The red component of the colour is driving the temperature, that is why the red smoke is rising quicker.

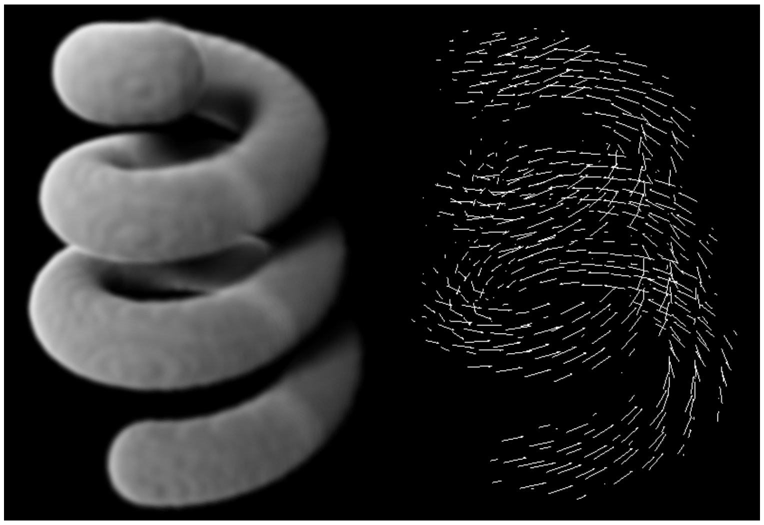
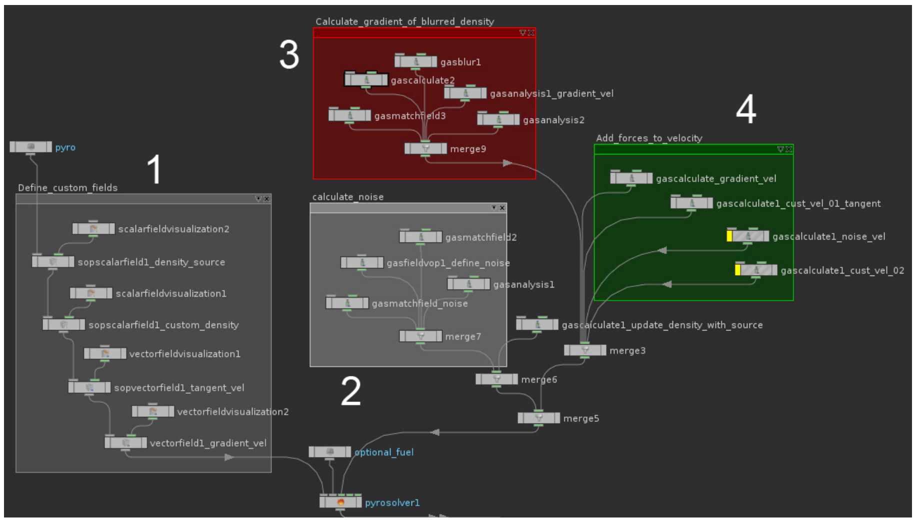


Figure 17: The density field and tangent velocity field defined by a helix curve and used to advect smoke.



3
Calculate_gradient_of_blurred_density
gasblur1
gascalculate2
gasanalysis1_gradient_vel
gasmatchfield3
gasanalysis2
merge9
4
Add_forces_to_velocity
gascalculate_gradient_vel
gascalculate1_cust_vel_01_tangent
gascalculate1_noise_vel
gascalculate1_cust_vel_02
pyro
1
Define_custom_fields
scalarfieldvisualization2
sopscalarfield1_density_source
scalarfieldvisualization1
sopscalarfield1_custom_density
vectorfieldvisualization1
sopvectorfield1_tangent_vel
vectorfieldvisualization2
vectorfield1_gradient_vel
calculate_noise
gasmatchfield2
gasfieldvop1_define_noise
gasanalysis1
gasmatchfield_noise
merge7
gascalculate1_update_density_with_source
merge3
merge6
2
merge5
optional_fuel
pyrosolver1

Figure 18: The node tree for the different steps for the advect by curve simulation as described in (5.3.2).

1. Define a few custom sop vector fields to hold to custom defined volumes from sops, such as the density source (the little sphere), the custom density of the curve, the tangent velocity and the gradient velocity (which will be calculated at a later step).

2. Calculate noise by defining new custom fields with the Gas Match Field microsolver, a scalar "noise_field" and a vector "noise_vel" field and define the noise in the "noise_field" inside of a Gas Field Vop using an Anti-Aliased Flow Noise vop. By using a Gas Analysis microsolver we can calculate the gradient of the "noise_field" and store it in the "noise_vel".

3. Calculate the gradient of the blurred density of the curve. Create a new temporary field to store the blurred_custom_density using a Gas Match Field, matching the custom density field of the curve. Copy the custom_density into the blur_custom_density using a Gas Calculate. Blur the blur_custom_density field using a Gas Blur. Calculate the gradient of the blur_custom_density and store the result in the gradient_velocity. Finally normalize the gradient_vel using a Gas Analysis. The resultant force will appear to be always pointing towards the helix shape.

4. Add the forces to density. By using Gas Calculate nodes, the various custom velocity fields can be premultiplied by higher values for a more powerful effect and added to the velocity field.

The simulation can get unstable when too much force is added, so it can be useful to add a low velocity damp on the pyrosolver, or even turn on speedlimits. A few stages of the simulation can be seen in Figure (19).

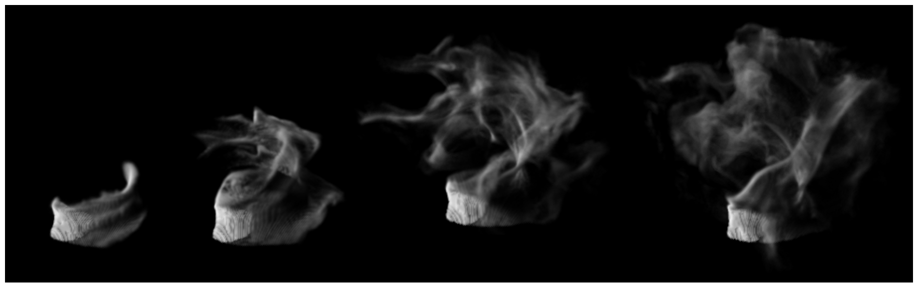


Figure 19: A few stages of the advection by curve. These images were flipbooked from the viewport.

#### 5.3.3 Custom fuel and heat injection

The following examples all make use of the combustion model inside the pyrosolver. By providing fuel and adding temperature, smoke and flames can be generated from the resulting density and heat fields. Some of these simulations will have vorticle geometry in them to add more turbulent motion.

**Fuel from curve** In this example the helix curve was used to define an initial fuel field inside of the pyro object. A fuel field is a default field and does not need a custom field unless you want to have an animated fuel field, for this example however the fuel field is static. Only one custom field is added to add temperature to the bottom part of the fuel field. The buoyancy direction on the pyrosolver is set to a negative y direction to avoid the heat from rising through the helix and igniting other parts before it is supposed to get there even though that is visually quite interesting as well. The only step during the advection process that is added is to add temperature from the custom source temperature field to the temperature field that will then be used by the combustion model and start the ignition. The burn rate on the pyrosolver is set to 2, so it burns quite fast and the temperature output is set to 6, so a lot of heat is generated which will keep the simulation going. The different fields at different stages can be seen in Figure (20).

**Fuel from animated geometry** In this example an animated model of a tree is used as a source geometry to define animated volumes of both fuel and temperature. The animated model was provided by Nick Hampshire, who created the animation through a real-time spring solver that he programmed for his Master thesis [9]. As the geometry represents a moving point cloud at its lowest level it works will with the attribute transfer tool. The fuel field has the appearance of spreading out through the branches which was achieved by copying an animated attribute from a static mesh of the tree to the animated mesh of the tree. The attribute simply spreads in a radial fashion, although it will spread faster at the outer points of the branches and slower at the root of

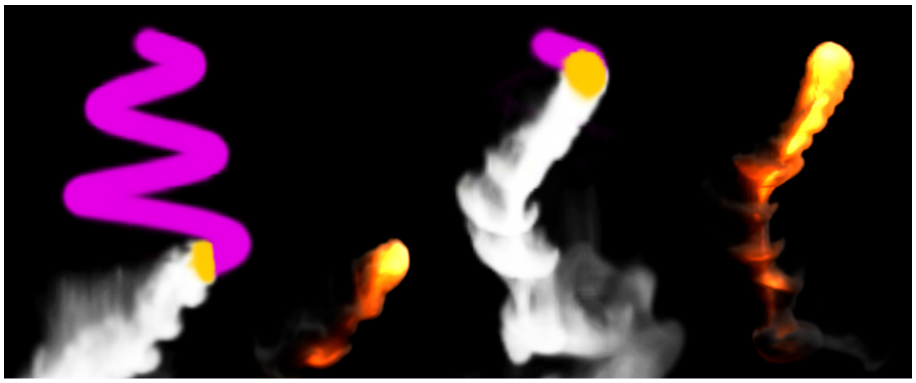


Figure 20: The purple field represents the fuel, the yellow field is the burn and the white field is the density resulting from the combustion in the fuel from curve example.

the tree where the branches are thicker, the animated geometry with the spreading attribute is then baked to disk to speed up the rest of the network. This attribute is then transferred using the volume transfer tool. The temperature field is the same as the fuel field. The tree geometry came with point velocities on it but those could alternatively be calculated with a trail sop as the topology does not change over time. These velocities are also transferred into a custom field.

In DOPS those three custom fields are brought in with Sop Scalar/Vector fields and their SOP path and Time ($T) parameters are set to Always evaluate. During the simulation step the fields are added to their respective target fields using Gas Calculate microsolvers. The fuel field is not added with a maximum operation as this added too much fuel into the simulation which resulted in a massive explosive reaction, so instead it is copied every frame and overwrites the old fuel data from the previous frame. A Gas Dissipate is also added to slowly dissipate the density. The various stages can be seen in Figure (21).

**Fuel from particles** In this example a simple terrain is modeled and particles fall down from above the terrain and collide with it. A particle that is colliding is added to a collision group. Outside of the particle network in sops, the points that are not in the collision group are deleted. The amount of collisions can easily be increased or decreased depending on the birthrate parameter of the source pop inside the pop network. At each frame there are generally between one and three particles in the collision group. By using an attribute transfer, the normal from the collision surface is transferred to the particle. Now the volume attribute transfer tool is used to define the fuel and temperature fields based upon the metaball density. A custom velocity field will also be generated with the tool, using the normal attribute from the particles which has just been transferred from the surface.

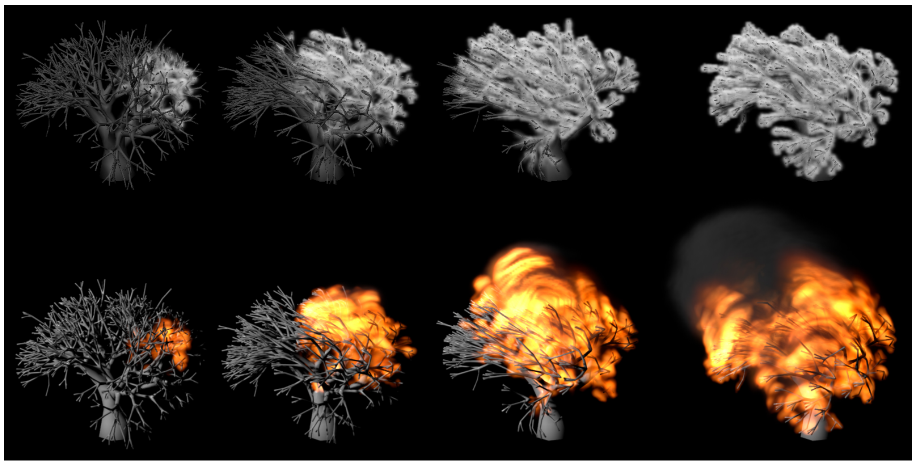


Figure 21: Top: The stages of the animated fuel field, Bottom: The stages of the rendered burning tree.

In DOPS the setup is very similar to the "fuel from animated geometry" setup. Because fuel, temperature and velocity are only added where the particles are currently colliding and a low number of particles is colliding per frame, not too much fuel is added each frame. Therefore in the Gas Calculate the maximum between the custom_fuel_source and the fuel is calculated. The custom_fuel_source field is premultiplied by a factor of 5 to introduce more fuel into the simulation. The transferred custom_velocity_source is heavily premultiplied by a factor of 150 before being added to the velocity field. Since the velocity direction is coming from the surface, the explosions are not directed straight at the sky. On the pyrosolver the inefficiency in the combustion model is set to 0.6 which causes some of the fuel to not be fully burnt. The burn rate is set to 0.8 which will cause the fuel to take longer to be consumed. This can give good results combined with the initial velocity as the burning fuel will be advected by the velocity and will continue to be burn as it is lifted up into the sky. A small amount of velocity damp is also introduced so the velocity wears off after the initial boost. The different fields can be seen in Figure (22).

## 6 Conclusion

The control Houdini offers over manipulating dynamics and performing more advanced mathematical calculations is extensive and there are many more microsolvers to study. Also fluid dynamics for computer graphics is a popular topic of research and new developments are happening all the time. A significant amount of time for this project went into researching existing ideas in fluids for computer graphics and in how volume fields and the microsolvers work together in Houdini. Gaining a complete understanding of how the preset smoke

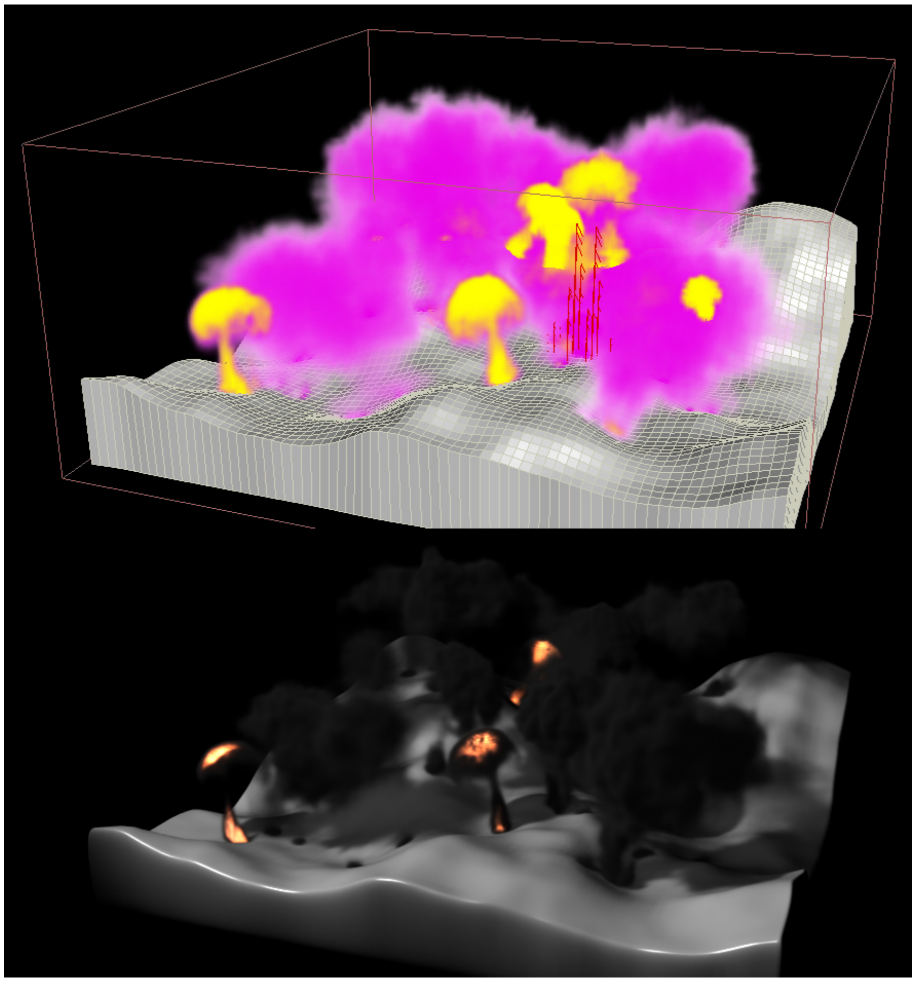


Figure 22: Fuel, temperature and velocity are transferred from colliding particles and set off small explosions. The fuel is purple, the heat is yellow, and the density is white. The initial velocity upon collision is shown as the big red lines.

and pyrosolver work took time. This research only started to become more applicable half way through the project as various experiments were developed. A lot more experiments can be done, especially by modifying the velocity field, but it seemed more valuable to focus on a wider range of different types of custom field implementations to show a broad use of the tool and the possibilities by using custom fields. The aim was to gain control over fluids by using custom fields and this has been achieved through the prototype experiments. These experiments are by no means production ready, as higher resolution simulations could have given much more detailed results. But that would also have taken a lot more time.

The main strength of this project is that the tool that was built is easy to use with a variety of applications, yet enables users to use all the functionality of sops, from creating geometry such as curves or particles to defining specific animated point attributes through expressions or vops to create those artistic volume fields. And because the tool follows conventions for defining volume fields, the fields can be used in other context besides dops, like pops or for rendering with Mantra. Within the metaball tool there is still an optimization that could be made when dealing with vector fields, because the volume vop is used separately for each component and ideally would only be used once for all the components in one go. This would speed up the tool significantly as the metaball loop only needs to be executed once instead of three times. The problem lies with the output of the volume vop in SOPS, which seems to allow only a single scalar field to be exported and what is needed is a vector field export option.

The microsolvers within Houdini are powerful mathematical building blocks that are able to build bigger structures like vortex confinement or entire solvers, however since the author was quite new to the field of fluid dynamics a good understanding and knowledge base needed to be built up first before more interesting experiments could be set up. By following the mathematical equations alongside the microsolver networks and being able to visualize the outcome of a given calculation in the viewport so it would make sense on a geometric level, the author was able to get a good understanding of fluid solvers and how to modify them. The author recommends learning Houdini fluids this way as you get information from both sides. Overall this project has been an interesting experience even though it was biased more towards the research side than the production side than was originally planned. By putting the equations next to the microsolver node networks the author hopes to have been able to create a better understanding for other Houdini users or perhaps opened up the door for researchers who would want to use Houdini for development of fluid solvers. By not focusing on the implementation of one big digital asset performing one specific effect, but instead building a small and comparatively light, very useful building block to help define custom fields, a much broader application of the tool is possible. The tool is like a microsolver in SOPS.

## Thanks

Thanks to Coen Klosters for taking the time to meet several times with me throughout the course of this project to discuss Houdini fluids and a great thanks for giving an extremely interesting masterclass. Thanks to Nick Hampshire for letting me use his L-system animation. Thanks to my classmates for bouncing ideas back and forth and for giving feedback on some of the experiments. Thanks to Jon Macey for giving useful advice during the mid-term crits and for pointing me in the right direction towards resources developed by previous MSc students. Thanks to the houdini community at www.odforce.net and at www.sidefx.com for clarifications and for sharing insightful scene files.

## References

[1] Christopher Batty and Ben Houston. Visual simulation of wispy smoke. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 114, New York, NY, USA, 2005. ACM.

[2] Robert Bridson. *Fluid Simulation for Computer Graphics.* A K Peters, 2008.

[3] Robert Bridson, Jim Houriham, and Marcus Nordenstam. Curl-noise for procedural fluid flow. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 46, New York, NY, USA, 2007. ACM.

[4] Robert Bridson and Matthias Müller-Fischer. Fluid simulation: Siggraph 2007 course notes. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 1–81, New York, NY, USA, 2007. ACM.

[5] Robert Bridson and Marcus Nordenstam. Naiad [computer program], August 2009. http://www.exoticmatter.com/overview/ [Accessed 17 August 2009].

[6] Gordon Chapman, Jerry Tessendorf, and Michael A. Kowalski. Art directing particle flows with custom vector fields. In *SIGGRAPH '08: ACM SIGGRAPH 2008 talks*, pages 1–1, New York, NY, USA, 2008. ACM.

[7] Raanan Fattal and Dani Lischinski. Target-driven smoke animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 441–448, New York, NY, USA, 2004. ACM.

[8] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 2001. ACM.

[9] Nick Hampshire. Dynamic animation and remodeling of l-systems. N.C.C.A., Bournemouth University, August 2009.

[10] Christopher Horvath and Willi Geiger. Directable, high-resolution simulation of fire on the gpu. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8, New York, NY, USA, 2009. ACM.

[11] Theodore Kim, Nils Thürey, Doug James, and Markus Gross. Wavelet turbulence for fluid simulation. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–6, New York, NY, USA, 2008. ACM.

[12] Mario Marengo. Sidefx forum post on "explosion using pyrofx", Jun 2009. http://www.sidefx.com/ [Accessed 17 August 2009].

[13] David Minor. Physical simulation of fire and smoke master thesis. N.C.C.A., Bournemouth University, September 2007.

[14] Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. Fast animation of turbulence using energy transport and procedural synthesis. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–8, New York, NY, USA, 2008. ACM.

[15] Sidefx. Video tutorials, 2008 - 2009. http://www.sidefx.com [Accessed 17 August 2009].

[16] Sidefx. Gas project non divergent dynamics node - help file, 2009. Houdini 10 help file.

[17] Jos Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[18] Mya Yee Win. The implementation of 2d fluid solver plug-in for houdini8.2. A thesis submitted for the degree of M.Sc Computer Animation, NCCA, Bournemouth University, September 2007.