

# TURBULENCE.2D for After Effects v1.2 **USER MANUAL**

© 2008-2009 jawset visual computing. All rights reserved.

# Table of Contents

Quick Start	3
Activating Your License	4
Introduction	5
About Fluid Dynamics	6
Handling The Simulation	6
Using GPU Support	7
Obstacles	8
Velocity Input	9
Combustion models	10
Parameter Overview	10
Source Control	13
Simulation Parameters	16
Rendering Parameters	17

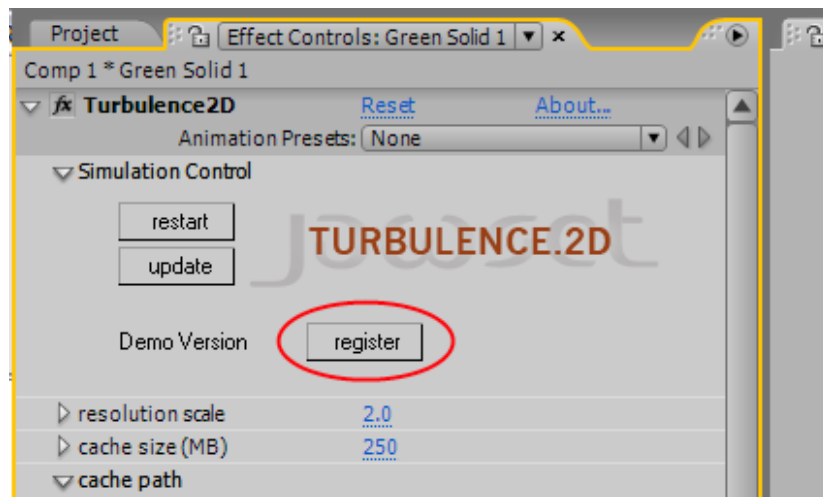
# Quick Start

- Create a new composition
- Create a solid that covers the composition
- Apply the effect from jawset > Turbulence2D to the solid
- Create another layer as input, e.g. some white text on black ground
- Select this layer under Source Control as fuel layer
- Click the „update“-button in the effect's top most control element
- A popup window will appear and the simulation will be computed
- Wait until it finishes or press escape or close the window to abort
- Now you can work in AE as normal, the simulated fluid is available for rendering
- You can change the Rendering Parameters without updating the simulation
- Changes to the values of Source Control or Simulation Parameters or any of the input layers will not affect the output until the simulation is updated
- To simulate at full resolution, set the resolution scale parameter to 1.0

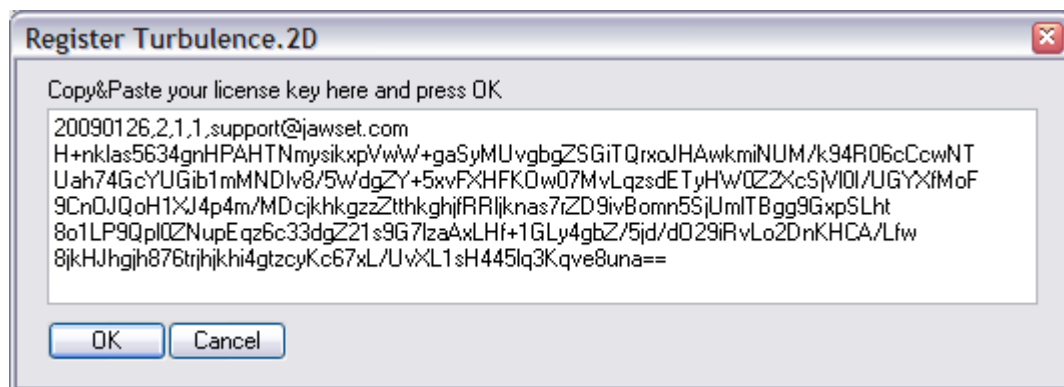
Also check out the example projects from the Examples folder in  
<Your After Effects Directory>/Support-Files/Plug-ins/jawset/Examples

# Activating Your License

When you purchase a license for Turbulence.2D, you will receive a license key. In order to activate your copy of Turbulence.2D, apply the effect in a composition so that you see the effect menu and click the “register” button.



A dialog box will appear that allows you to paste your license key into a text field. A complete license key looks something like in the image below.



After pressing OK, your copy of Turbulence.2D will be licensed and the watermark will disappear.

# Introduction

In computer graphics, images are created with simple geometric primitives that can easily be controlled in every tiny aspect. In order to obtain interesting results, we combine many instances of such simple geometry and apply many modifiers on them. Perfect and shiny images are created like this.

However, using simple geometry and simple modifiers always bears a certain predictability that ultimately is boring. In order to hide the underlying simplicity, we go through a lot of trouble to make scenes more and more complex. The limit to this race is the artist's time and patience. Especially because obtaining much detail usually is very tedious work.



Much tedious work sounds like a job for a computer. Hence, simulation comes into the picture. Physical simulation allows to create complex natural phenomena automatically, investing CPU time instead of artist time.

Fluid simulation in particular produces some of the most complex results in computer graphics with very little manual input. By passing some of the control you have over every pixel to the simulation, you get very organic shapes and motions that add a new depth to your work. Some of the results are so complex, you can watch them over and over again and discover a new detail every time.

Turbulence.2D puts a state-of-the-art fluid simulator into your toolbox that provides many ways of using the it's organic complexity in your work.

# About Fluid Dynamics

*Predictability: Does the Flap of a Butterfly's Wings in Brazil set off a Tornado in Texas?*

*(Edward Lorenz, 1972)*

In a fluid simulation, every pixel potentially affects every other pixel. Every frame affects the next frame and therefore every frame that follows. This means maximum complexity and seemingly chaotic behavior. But we're still in control.

A fluid basically is a set of thousands or millions of small particles that affect each other in a certain "fluid-like" way. This makes fluid simulation different from particle systems, where particles usually don't affect each other. In a fluid, a few particles at one end of the domain can cause a wave that propagates all the way to the other end without those few particles actually moving there. They just push at their neighbors, they push at theirs and so on.

## Handling The Simulation

In order to create a fluid simulation, apply the effect, add input layers like density, fuel, color and temperature (see the Parameter Overview for details) and press the "restart" button the the effect's menu. The preview window will pop up and show the progress of the simulation. You can abort at any time by pressing the Escape key or closing the window. All frames that have been simulated so far are available in After Effects and you can work with your composition as usual. Additionally, you can change all the parameters in the "Rendering parameters" group, without updating the simulation.

A consequence of the frame-by-frame dependence, mentioned in the previous section, is that in order to simulate frame 100, frame 99 will already have to be simulated and so do all frames before that. We cannot just jump to frame 100 to see what it will look like. However, with Turbulence.2D, you don't need to restart the simulation from frame 1 every time to change a setting. You can just interrupt it by closing the preview window, change some settings and continue where the simulation left of, applying the changes immediately.

# Using GPU Support

Fluid simulation is computationally very intensive and typically, fluid software is pretty slow. Turbulence.2D is optimized for speed. Besides using all available processor cores, it also supports the graphics processor (GPU) to be used to accelerate the simulation up to 10 times. This allows for very intuitive experimentation at interactive frame rates.

If GPU Support is available on your machine, the checkbox “GPU Support” will be enabled and checked by default. If that is the case and there is enough free video memory available on your card, Turbulence.2D will use the GPU to speed up the simulation. You can see which mode (“CPU” or “GPU”) is being used in the title bar of the preview window.

Visit [http://jawset.com/gpu\\_support.php](http://jawset.com/gpu_support.php) for a complete list of supported graphics cards and more details on enabling the GPU support on your machine.

# Obstacles

In Turbulence.2D, obstacles are solid objects that do not flow. They affect the fluid but the fluid does not affect them. You can use arbitrary shapes as solids - buckets, paddles, walls, ramps or text. The image below shows an example of text as a solid object, that forces the green vapor to flow around it.

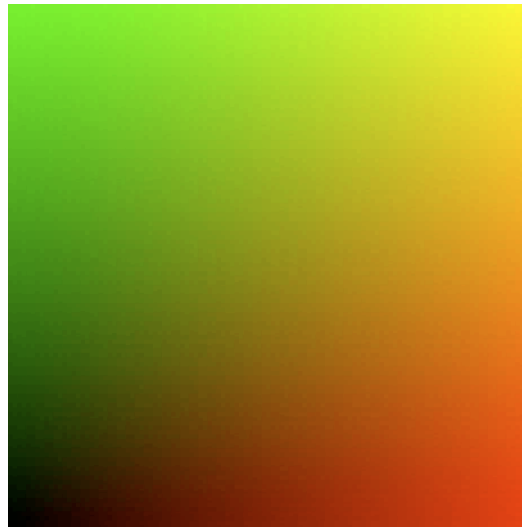


If you select a layer as an obstacle layer in the “Source Control” parameter group, the alpha channel of that layer will define the solid obstacle. See the Parameter Overview for more details.



# Velocity Input

The velocity inputs in Turbulence.2D use Motion Vectors. A 2-dimensional vector is encoded in the red and green channels of a pixel color. The image below shows how colors are mapped to directions. The zero-point in the middle has (127,127,0) in 8-bit color mode, (16383,16383,0) in 16-bit color mode and (0.5,0.5,0) in 32-bit color mode. This color basically represents no velocity. A velocity pointing down and to the left is black, one pointing right and up is yellow and so on.



Using velocity inputs you can for example create wind that blows your density, color or fuel around. To do this, use differential velocity or force.

You can also just reset the velocity field of the fluid to a certain value. For example, if you want the more turbulent behavior that rising smoke has in still air, you can use the non-differential velocity input to reset the velocity to zero.

If you move a solid through a fluid, it will affect the motion of the fluid. For example if you swing a fan through air, it will cause the air to move with the velocity of the fan for a while. In order to simulate the correct movement of the fluid, Turbulence.2D needs to know the velocity of the obstacle. Note that if you set incorrect velocities for the solid, the resulting simulation will look wrong.

The correct velocity is the number of pixels that the solid moves per second. For example, if you have a block that moves 200 pixels to the right in 2 seconds, it's velocity is 100. To tell that to Turbulence.2D, create a layer that shows the block with a single color of (254,0,0) and set the velocity scale to 100.

# Combustion models

Turbulence.2D contains a flexible model for stylized fire. You can add fuel to the flow, using the fuel layer (see the Parameter Overview for details). Fuel always burns. How fast it burns is specified by the “burn rate” parameter. The slower the fuel burns, the larger the flame becomes, because fuel flows around without burning.

When fuel is burnt, it creates soot or density. How much soot is created is specified by the “soot creation” parameter. It specifies how much soot is created per unit fuel. That is, if you set it to 1.0, there will be exactly as much density as fuel has been burnt.

Burning fuel also creates heat. How much can be specified by the “heat creation” parameter. If the fuel creates more heat, the fuel and soot will rise faster.

A flame emits light, that’s what we see. Emitting light makes the flame lose energy, though, effectively cooling it. You can control that with the “cooling” parameter. Higher values create shorter flames here. It behaves differently than just cutting down the fuel supply, though.



Because the air and burning fuel in a flame is very hot, the gas expands. The expansion of a flame is a typical visual phenomenon that can be simulated in Turbulence.2D with the “expansion” parameter.

# Parameter Overview

## *Resolution scale*

Specifies the size of a simulation cell in pixels. A value of 1.0 means that each pixel in the output layer corresponds to one simulation cell.

Set this value to 2.0 or higher for quick previews. The simulation will have low quality, but will be faster. For final rendering, set it close to 1.0. You can also simulate at a higher resolution as the visual output using values below 1.0. Such simulations can take a lot of time but produce extremely detailed flows.

## *RAM Cache size (MB)*

Specifies the amount of memory to reserve for the simulation cache. Unlike the cache built into AE or RAM Preview, the simulation cache allows you to adjust Render Parameters without updating the simulation. If you want to scrub through your timeline a lot, using more memory here will make things smoother.

## *Disk Cache*

The directory where the simulation output is to. If you update or restart a simulation, existing files in the cache directory will be overwritten. You can save the results of a simulation run, for example to compare several simulation settings.

## *Source Control*

Changes to the source control parameters or the contents of the layers selected here will not affect the output until the simulation is updated.

## *Density layer and scale*

Selects the layer that defines the input density. The color values in this layer are converted to gray scale values and used as intensities ranging from 0.0 to the value set for the density scale parameter. That is, a black pixel in the input layer corresponds to 0.0 density, a white pixel corresponds to the amount set for the density scale parameter. Note that the alpha channel masks the input.

The layer is re-scaled in order to fit the output layer. If you want to use a smaller layer, you need to put it into a separate composition and use this composition as an input.

Density is a generic material that is injected into the flow and can be

visualized using a gradient (see Rendering Parameters). You can think of it as smoke- or paint-particles.

Density has a weight. Depending on the amount of density and the value of the gravity (see Simulation Parameters), it will cause the fluid to sink or rise.

## *Color layer*

While density can be rendered using a gradient, you cannot choose arbitrary colors for different areas in the fluid. The color is always defined by the amount of density and the corresponding gradient. Temperature and fuel behave similarly.

In order to have arbitrary color in the fluid, you can add it using the color layer. Like texture coordinates (see below), color input is passive, that is, it does not affect the fluid velocity like density or temperature do. It only flows with the fluid.

You can use the color layer to mix images like in a bucket of paint, while injecting more color along the way.

## *Temperature layer and scale*

Selects the layer that defines the input temperature. The values are computed similar to the density input (see above), only using the temperature scale value.

Like density, temperature also affects the velocity of the fluid. Depending on the temperature and the value set for the bouyancy (see Simulation Parameters), it will cause the fluid to sink or rise. Hot parts of the fluid rise above colder parts.

## *Fuel layer and scale*

Like the density and temperature layers and scales, these two settings specify how much fuel is inserted into the fluid. Unlike density and temperature, fuel does not affect the velocity directly. It has no weight either. However, fuel burns and produces heat and soot, that is temperature and density.

See the chapter on Combustion models for more information.

## *Divergence layer and scale*

Positive divergence means expansion, negative divergence means contraction. The intensity value of this input is scaled by the divergence scale and shifted such that it represents values between  $-X$  and  $X$ , where  $X$  is the scale. That is, a black input pixel will end up being  $-X$  divergence or a contraction of intensity  $X$ ; a white pixel will expansion of intensity  $X$ .

You can think of divergence input as of blowing (positive divergence) and sucking (negative divergence) the fluid.

## *Velocity layer and scale*

While density and temperature affect the velocity of the fluid indirectly, you can also specify it directly with the velocity input. Velocity is a vector, it has a value and a direction that are represented by an  $X$ - and a  $Y$ -coordinate. 2D velocity input is encoded just like Motion Vectors. The red channel of the input layer specifies the  $X$ -coordinate, the green channel specifies the  $Y$ -coordinate.

See the chapter on Velocity Input for more information.

## *Differential velocity*

Differential velocity is actually force. Use it to create wind for example.

Non-differential velocity can be used to clear the velocity field or to immediately set it to a certain value. In order to be able to set only parts of the velocity field, the alpha channel masks the values. Only pixels with an alpha value of 50% or more will be used.

## *Obstacle layer*

Specifies the input layer for solid obstacles. Every pixel with a alpha value of 50% or more is treated as solid. Obstacles force the fluid to flow around them. You can create buckets, ramps, paddles or similar solid objects using this input.

If your obstacles move, you need to specify their velocity as Motion Vectors (see velocity layer above).

See the chapter on Obstacles for more information.

## *Obstacle layer has velocity*

If this box is checked, the red and green channels of the obstacle layer are expected to contain Motion Vectors for the obstacle.

See the chapter on Velocity Input for more information.

## *Obstacle velocity scale*

The velocity scale for the obstacle's Motion Vectors, if obstacle velocity is enabled.

See the chapter on Velocity Input for more information.

## *Use texture*

Enable texture coordinates for the simulation. If you want to use the texture render mode, this box has to be checked during the simulation. Else, it can be left unchecked in order to improve performance.

## *Simulation Parameters*

Changes to the simulation parameters will not affect the output until the simulation is updated.

## *Domain type*

The “open” domain simulates the 2D fluid as if it was a slice of 3D space. Material cannot leave the slice in the 3rd dimension, but pressure can. The “half-open” domain corresponds to fluid on a table or a plate. Pressure can leave to one side, the material remains on the plate. The “closed” domain corresponds to fluid between two glass plates. Pressure cannot leave in the third dimension.

A “closed” domain is what most 2D fluid tools use. “open” and “half-open” look somewhat more realistic, the “closed” domain creates larger vortices.

## *Burn model*

The “oxidized” model assumes that the fuel does not need oxygen in order to combust. Every cell that contains fuel will burn. In the “un-oxidized” model, fuel only burns if enough air is available. Only those cells that contain fuel and are close to the boundary of the fuel-region will combust.

See the chapter on Combustion models for more information.

## *Time scale*

If this value is 1.0, the physical behavior of the fluid is as in real-time. Values larger than 1.0 make it flow faster, values below 1.0 make it slower.

## *Density dissipation*

Values larger than 0.0 make the density disappear over time. For example, you can make smoke dissolve slowly.

## *Burn rate*

The rate at which fuel is burnt. Typical values are lower than the fuel scale of the fuel input layer. The lower this value, the larger the flame will be and the less heat and soot will be created.

See the chapter on Combustion models for more information.

## *Expansion*

The high temperatures of fire make the fluid expand. This value specifies how much. The expansion also depends on the amount of fuel that is burnt. Large values produce thick flames.

See the chapter on Combustion models for more information.

## *Heat creation*

The amount of heat that the flame generates. If the flame is hotter, it will rise faster. This value also depends on the amount of fuel that is burnt.

See the chapter on Combustion models for more information.

## *Soot creation*

The amount of soot (density) that is produced by the flame. This value also depends on the amount of fuel that is burnt.

See the chapter on Combustion models for more information.

## *Cooling*

A flame loses much of its heat through the light it emits. This value specifies how fast this heat is lost. Large values produce shorter flames.

See the chapter on Combustion models for more information.

## *Gravity*

This value specifies how strong the density is affected by gravity. Large values make density sink faster.

## *Bouyancy*

This value specifies how fast hot fluid rises up or falls down.

## *Heat diffusion*

Particles in a fluid mix because of the large-scale motion that gives the fluid it's characteristic look, but also because of so called Brownian motion on a molecular scale. This molecular effect is simulated by diffusion.

The higher this parameter is, the more diffusion will take place, and the blurrier the temperature field will be. Typical values are between 10 and 100.

## *Vorticity*

Sets the amount by which vorticity (curl or swirl) is amplified. Be careful with this value, as it can make your fluid get out of control, degenerating to a noisy chaos.

## *Rendering Parameters*

These parameters can be changed without updating the simulation. You will see the result in the output immediately.

The Render Mode specifies the method that is used to determine colors from the fluid fields. Additionally, the Alpha Mode specifies the method to determine the alpha value of a rendered pixel.

### *Render mode*

As the result of the simulation the density, temperature, fuel, color and velocity fields can be visualized in various ways.

#### *All sources*

Renders the density, temperature and fuel fields by using the corresponding gradients (see below) and adds the colors together.

#### *Fire + Smoke*

Renders fuel and density as fire and smoke using the corresponding gradients (see below).

#### *Stylized Fire*

Renders the density and temperature by multiplying their intensities and using the value to drive the temperature gradient. Then adds the color



obtained from the fuel field and gradient.

The multiplication of density and temperature is similar to how realistic fire is visualized. Fire is basically glowing soot – the more soot there is and the hotter the soot is, the brighter the flame will be.

## Color

Render the color channel of the simulation. Use this mode if you have a color source layer. This can be used to let the flow mix colors. Inputs can for example be photos. This is different to the texture mode, though because it actually mixes the colors of the source, not just distorts the image.

## Velocity

Render the Motion Vectors of the simulated velocity field. This can be used to drive other effects like motion blur.

See the chapter on Velocity input for a detailed description of the format of Motion Vectors.

## Texture

This mode renders the texture layer (see below) distorted by the fluid. It requires that “use texture” has been enabled during the last simulation update.

## Refract

The refraction mode renders the texture layer (see below) refracted by the fluid.

## *Alpha mode*

The alpha mode specifies how the simulation properties are combined in order to determine the alpha value of a rendered pixel. In all modes, the value is clamped to the range between 0.0 and 1.0 (or corresponding 8-bit or 16-bit values) before being used as an alpha value.

## Source

Uses the intensity of the source layers as alpha value.

## Maximum

Uses only the maximum of the intensities of the source layers as alpha value.

## Opaque

Set the alpha value constantly to 1.0 (or corresponding 8-bit or 16-bit values). However, you can specify a falloff point. If the maximum source intensity falls below that point, the alpha will fade out, such that the result can be properly composed above other layers.

## *Texture layer*

Selects the input layer for the “texture” and “refract” rendering modes (see above).

## *Motion Blur*

Sets the amount of motion blur that will be applied to the output. This is very useful for fast moving fluids like fire.

## *Max. velocity*

The maximum length of velocity vector for the “velocity” render mode. For example, the color (254, 0, 0) (or corresponding 16-bit or 32-bit values) of a motion vector will correspond to a vector pointing to the right with the maximum velocity set here.

## *Refraction scale*

Sets the strength of the refractive displacement for the “refract” render mode.

## *Refraction smoothness*

The high detail in the fluid may cause inevitable aliasing artefacts when refracting pixel maps. Smoothing the fluid first, allows to control that effect. This parameter specifies how strong the fluid shall be smoothened.

## *Density, temperature and fuel color*

These color gradients are used to render in the first 3 render modes (see above for details).

Click on empty space below the gradient bar in order to add another sample. Double-click the handle of a sample in order to select the color. Drag samples to the left/right in order to move samples in the gradient. Drag a handle out of the gradient control area in order to remove a sample.

Click one of the preset buttons on the left to use these predefined gradients.