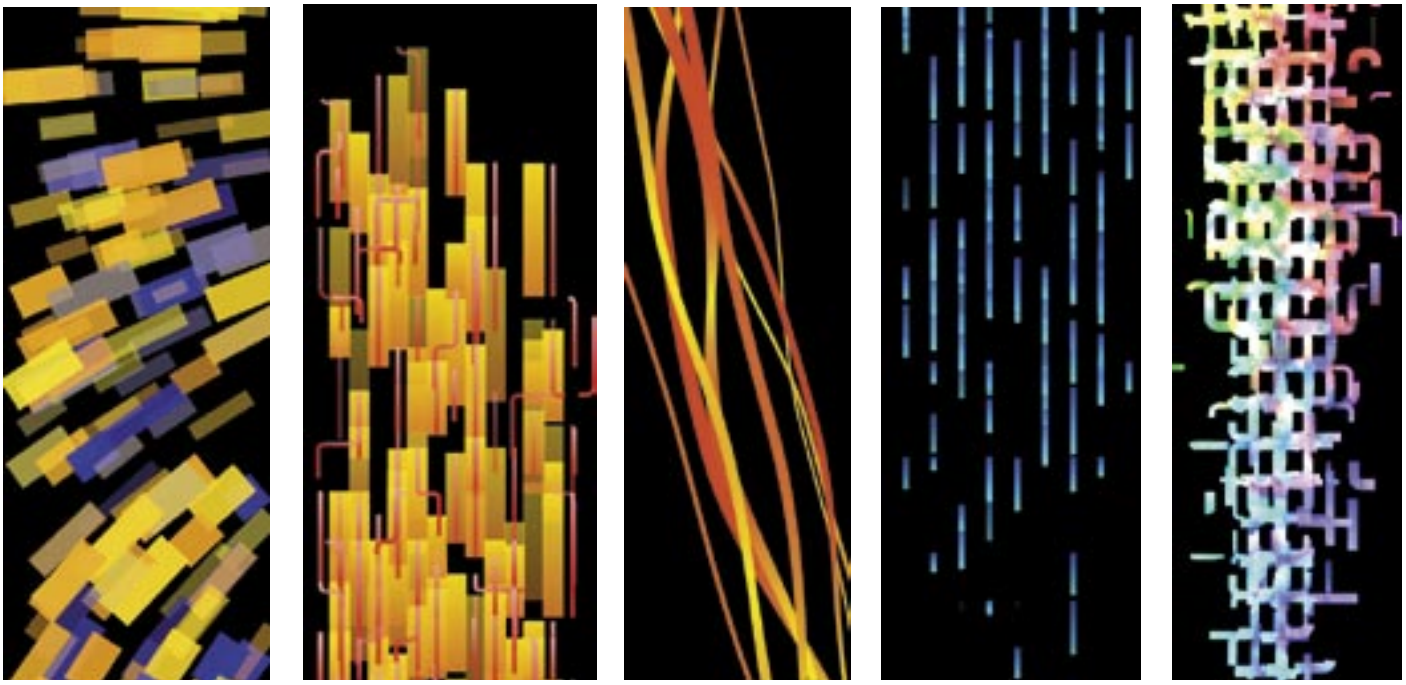
RED GIANT SOFTWARE™

www.redgiantsoftware.com

Geomancy Shapes manual

Shapes and lines for broadcast design.



Geomancy is a part of the new Text Anarchy package.

Introduction to Geomancy

Geomancy was designed to allow you to easily and quickly create and animate many geometric shapes. A common element in broadcast design in recent years has been floating lines, squares, circles, and other recognizable shapes.

While these elements are easy to create, animating many instances of them can be time consuming and tedious. In the case where they are simple background elements, this is not usually time well spent.

Geomancy gives you the control to produce a great number of these shapes quickly, and not have to worry about setting keyframes to animate them. By using a particle system and grid system to control the animation, we make it very easy to set the filter up, and then the filter take care of all the nitty gritty details of moving the shapes around.

It provides a very powerful and flexible way of accomplishing this oft asked for effect. As with our other set of particle system based plugins, Text Anarchy, Geomancy is designed give you more time to spend on designing and animating key components of your piece.

About This Manual

This manual discusses each of the filters separately. It also gives a Common Parameters section that explains the controls that are present in all of the filters.

Support & Registration

Registration occurs when you purchase the filter. We register you in our database using the contact information you supplied upon purchase, and the serial number we've given you. If you need a serial number, installer, or any other material support, just contact sales@redgiantsoftware.com or call 1 (260) 918-4505.

We hope that you find Geomancy to give you all the control you could want, while simple enough that you can set everything up in a few minutes. It's our desire to make sure you're satisfied with your purchase. If you have any questions, comments, or whatever, we'd love to hear them.

If you're having trouble with Geomancy, please make sure to go through the intro tutorials available at www.redgiantsoftware.com.

Installation: Macintosh

Geomancy is supported on Mac OS 10.3.9 +higher. We do not support OS 9.

After Effects

Launch the Geomancy installer. In the main window, you'll see a pop-up in the upper left corner that asks you to select your version of After Effects. Geomancy supports versions 4.1 and up. Make sure you choose correctly for the Mac operating system that you are running.

Once you've selected the correct version, in the lower part of the window is the installation destination. Click on 'Select Location' and navigate to your After Effects 'Plug-in' folder. **NOTE:** This is important. You need to specifically choose the 'Plug-ins' folder as your destination, or the plugin many not work correctly.

You are now ready to install. Click the 'Install' button.

Installation: Windows

Geomancy is supported in all current variants of Windows, from 98ME through XP Pro. We do support Windows Vista.

After Effects

Launch the Geomancy installer and click the 'Next' button until you get to the 'Locate Destination' screen. Click on 'Browse' and select the After Effects plug-in folder. Click 'Next'.

The next screen asks you which version of After Effects you'll be installing for. Select the appropriate version and click 'Next'. **NOTE:** This is important. You need to specifically choose the 'Plug-ins' folder as your destination, or the plugin many not work correctly.

You are ready to install. The installer will show a screen informing you of this. Click the 'Next' button to begin installation.

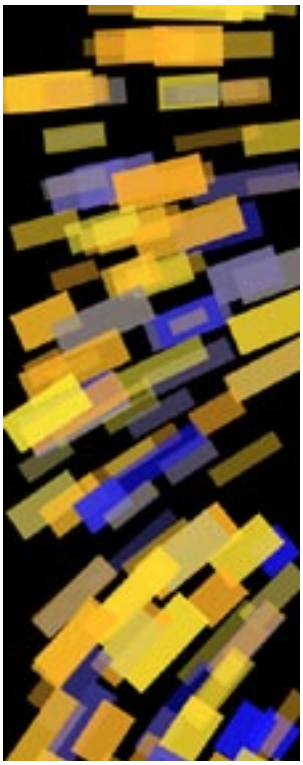
GridSquares

Welcome to GridSquares! We're going to explore the depths of this plugin, which is very powerful for creating random, animated elements. GridSquares was designed to make the animation of elements very easy. In this case it causes geometric shapes to be produced.

Like all of the Geomancy plugins, GridSquares is meant to simplify the process of creating these shapes. Making it easy to create sophisticated backgrounds, without spending hours fiddling with a million keyframes.

You're probably wondering how we think we can simplify this process. Well, by creating a particle system that is designed to do nothing but shapes, it makes it easy to customize the controls for making Squares, and controlling their behavior.

This gives you the ability to quickly, and, most importantly, easily set up the animation, get the look you want, and move on to other things. This ease is facilitated by the many Randomness controls that automatically randomize the appearance and behavior of shapes. They're really fun!



Some colorful squares courtesy of GridSquares, spinning as the Producer Point gets animated downwards along the Grid. Tweaking some of this stuff later may change your entire animation around, requiring you to tweak many other parameters.

The Secret Behind GridSquares

In order to do its magic, GridSquares first sets up a grid, then draws the squares into the spaces between the rows and columns. You then tell the squares how they're going to look, how to behave, and how rigidly they're going to stick to the grid.

It's all about setting up the grid, then telling the Squares to get a little crazy, and ignore it. Of course, you can tell your Squares how to behave as well.

The Grid itself can be set to any size, rotation, position, etc, and can be set up with any number of rows and columns. How you set the grid up is going to go a long way in determining how the animation is going to look, so make sure you have some idea of what you want to do, before you setting it up.

The rows and columns define what a 'Grid Space' is. Make sure you turn on 'Show Grid' so you know what you're getting as you adjust the setup parameters. Of course, you'll learn about all of this as you read the next few pages.

Grid Section

This section allows you to determine where the GridSquares particles sit against a (typically) invisible, underlying grid. The grid is your key to directing the shapes initially.

Height, Width

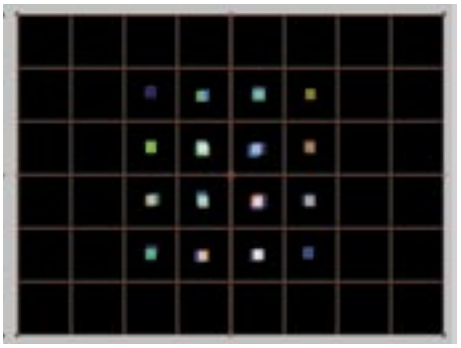
Should be pretty self explanatory what these guys do. Basically just sets the size of the grid in pixels.

You need to be careful about this, because if you set the grid size too small, the Squares will go off the edge. If you set the grid size too large, bigger than your comp for example, the animation will be slowed down unnecessarily. Unnecessary because you're telling GridSquares to keep track of squares and information you're never going to see. Obviously, if you're going to move the Grid position or rotate it, the squares might move into view.

Here, the grid is too small and doesn't cover the entire screen. Notice that the squares are the same size, on both grids. Grid size does not affect square size.

This is one of the things that affects the rendering speed, so don't make this any bigger than need it. If you just need a bunch of squares running across the width of the screen, at the top or bottom, don't make the grid the size of the entire comp. Just make it a 100 or so pixels high, and however wide the comp is. This will save the extra hit on computing power, a large grid would have given you.

Keep track of how big your producer point is going to be as well. If you're going to have a small producer point, it may not be necessary to have a large grid. Take a look at the Producer Point section for more info.



The grid is 1000x1000 on a 640x480 layer. This has no benefit, unless you're going to move or rotate the grid.

Make Size Match Layer Size

This just makes the grid match the size of the comp. If this is checked it ignores the Height and Width. If the Height and Width is set to 300 and 200, respectively, and the comp size is 640 x 480, turning this option on will make the grid size 640 x 480.

Position

This sets the center of the grid.

Rotation

Sets the rotation of the grid. This is different than rotating the squares themselves. If the grid is rotated, then all the squares (or whatever shape you've chosen), start off at an angle and the squares remain perpendicular to each other.

It looks a bit like taking a stack of pizza boxes, turning the entire stack 30 degrees. Each box is still directly on top of the other boxes, in such a way that relative to each other, there is no rotation.

This is not the case with square rotation in Square Setup. This would cause each box to be rotated 30 degrees, independently of the other boxes. As you can see in our image, this produces a different look.

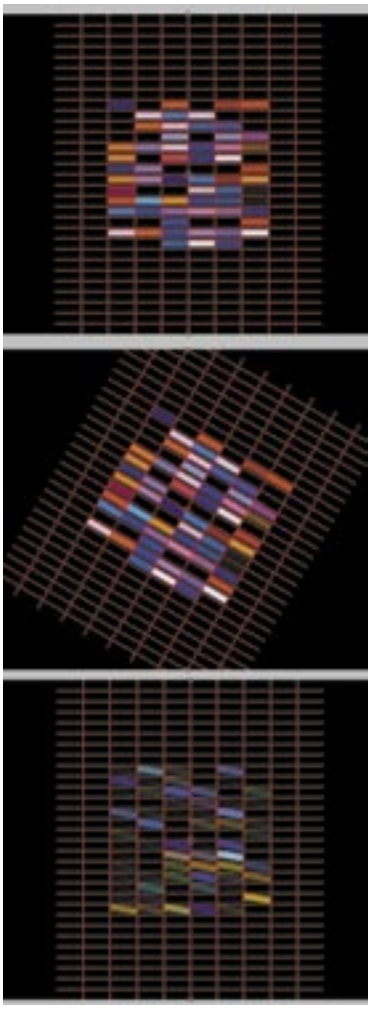
Rows, Columns

Rows and columns define the division of the grid. You can see them by turning on the 'Show Grid' parameter.

The squares are linked to the spaces between the rows and columns (grid spaces), so how many you have and how they're spread out (due to the size of the grid) will determine the structure of your animation. There are a variety of controls that we'll get into shortly that determine how rigidly the squares stick to the grid spaces, but, by and large, the divisions will set up the underlying foundation.

The rows and columns act as the Squares on a Tic Tac Toe board. The squares are only allowed to appear between the Squares, just like X's and O's on a tic tac toe board. If you have three rows and columns like you do with a Tic Tac Toe board, the squares really only have 9 places to appear.

There are ways to make the squares not follow the grid so strictly, and we'll go into those later.



This shows an example of both grid rotation and square rotation. The top image is the normal, unrotated squares. The middle image has the grid rotated. The bottom image is using the rotation attribute for the squares The difference is obvious. The squares are always going to be constrained to the grid. So you get a dramatically different effect if you rotate the squares themselves instead of the grid.

These are not animatable, since they're the underlying structure for the whole animation. Like any good foundation it needs to stay stable. There are other ways to change the look of the animation, but this has to remain the same, throughout. So give careful consideration to what you're going after when you set these up.

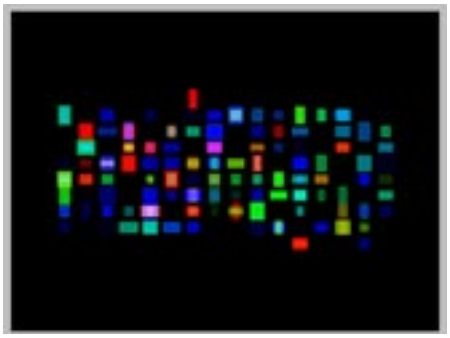
Again, selecting 'Show Grid' will make setting this part up easier.

Line Width

This just sets the widths of the Squares that 'Show Grid' causes to be rendered. Very useful at half or quarter resolution, when 1 pixel lines sometimes don't show up.

Producer Point Section

The producer point area defines where the squares start at. It can be as small as a single pixel or as large as the entire grid area.



Producer Point is a large value

The grid defines where the squares can go, and the Producer Point defines where they start on their journey. Consider the grid a range, and the Producer Point the home on the range. The Squares can go anywhere they want on the range, but they all have to start out back at home.

The size is set in pixels by the Height and Width parameters. While you can set these numbers higher than the height and width of the grid itself, GridSquares is smart enough to cap the Producer Point size at the size of the grid. There isn't any point in having squares created outside the grid as there are no rows or columns for them to be created in.

Producer Point is a small value

The position point determines where the center of the Producer Point is located on the grid. Again, you CAN position this off the grid, but Squares will only be created in the area that overlaps onto the grid. If no portion of the PP is on the grid, no Squares will be created.

If you have a large Producer Point, squares can be created anywhere within it. This creates a much different effect than if it's small, since in that case, all the Squares seem to be coming from the same point, and moving outward. If it's large, there is no central point, so Squares are created wherever, and head off in the direction (or directions) you have specified.

As long as the producer point touches a grid space, that grid space is capable of receiving squares. So, if we have a producer point that's 20x20, most likely it won't completely cover any grid spaces. However, it'll probably touch on 2 or 4 or them and all those will generate squares.

A large area produces a much more random effect, especially if Random Squares is checked in the Square Setup section (we'll get to that next).

You can also get some very different effects if Grid Space is selected instead of Pixel as the Unit. We'll go into the Unit pop-up in great detail shortly.

Square Setup Section

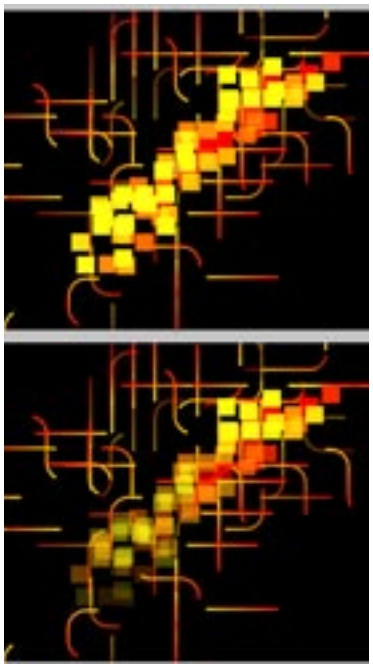
This section controls some of the behavior of your shape particles. Most of these parameters are tied into their movement and direction. The default particle is a square; thus, the name of this section.

Lock Birth Attributes

This attribute affects some of the parameters in the Square Attribute section. Specifically: Speed, Viscosity, and Opacity.

If this is checked, each square that gets created will remember it's settings for those attributes, and if you animate them, squares created before the change in values, will not be affected. For example, if you animate Width from 2 to 6, Squares created before the keyframes will always have a Width of 2. Squares created afterwards will have a Width of 6, so you'll have Squares with varying Widths on the screen at the same time.

If LBA is not checked, when you animate Width from 2 to 6, ALL squares on the screen, and all those created afterwards will be 6 pixels wide. This is how most of the attributes behave regardless of whether LBA is on or off.



An example of how Lock Birth Attributes works. In both these cases opacity has been animated from 0 to 100. In the top image LBA is not on. This causes all the squares to change opacity at the same time. We are looking at the end of the animation, so ALL squares are at 100% opacity.

Compare that to the bottom image, where LBA is on. Notice that the older squares retain their original opacity value. Even though newer squares are at 100% opacity, the older ones are at whatever opacity value they were originally created with. The squares at the bottom are almost totally transparent, while the ones in the middle are halfway transparent, and the ones at the top are completely opaque. The lines in the background were created using GridLines.

Line Unit

This paramter determines what measurement unit is used for the Squares, in the attribute section. There are two options: Pixel and Grid Space. Pixel is fairly obvious, but what about Grid Space?

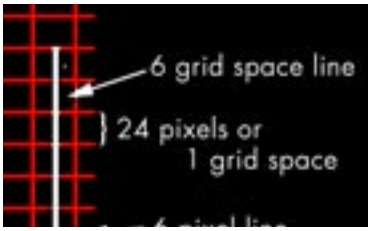
Grid Space is the space between rows or columns. The area between one column and the next, or one row and the next, is one grid space. If you have 10 columns in a 200 x 200 pixel grid, each grid space will be 20 x 20 pixels (200/10). If you had 4 columns, each grid space would be 50 x 50 pixels (200/4).

Obviously, you have to be careful about this, because just up and changing this will seriously change the animation. If you change it to Grid Space without adjusting the attributes, you may get extremely long render times.

If Minimum Length (in the Line Attributes Section) is set to 10, if pixels is selected the minimum length of a line is 10 pixels. So far, so good. If we now change to Grid Spaces, if a grid space is 50 pixels by 50 pixels, at a minimum of 10, the minimum goes from 10 pixels, to 500 pixels (10 (the minimum) x 50 (the size of a grid space)). This results in a lot of extra calculation, which will slow things down, while you're trying to change everything to a more normal amounts for the Grid Space unit.

So setup your attributes to something that makes sense for the particular unit, BEFORE switching over to it.

This is very key in achieving certain types of effects with the squares. Particularly when you want the squares to make the grid obvious. For example, you could move the producer point over the grid, and new squares would be only created in the grid spaces, which makes the grid much more apparent. There's a good tutorial on doing this on the web site.



Your choice of units can have a very significant on the look of your lines or squares, and how they animate. This example uses GridLines, but the concept is the same for GridSquares.

In this example, each grid space equals 24 pixels. If you had less rows and columns the number of pixels would be larger, if you had more rows and columns, the number would be smaller. Divide the size of the grid by the number of rows or columns. This gives you the pixel value of each grid space.

Rotation

This parameter sets the rotation for the squares. This causes each square to be rotated by this amount. See the Rotation parameter in the Grid section for a full discussion of the differences.

Random Squares Checkbox

This checkbox causes Squares to be sent off in all directions. You have no control over where they go, they get created and are sent off and running in some random direction. Actually, the Turning Controls still affect random Squares, so you do have some control over what they do, but not much.

With this selected, the Vertical Squares and Horizontal Squares pop-ups are ignored. The Squares are quite happy running about in any direction they choose and aren't about to be hampered by a couple silly pop-ups or a dial. With that said…

Direction

If Random Squares is not checked then the Squares are constrained by this dial and the series of checkboxes beneath it. Squares get spewed off in whatever direction is selected here.

You can either set them to go in one direction, or use the bi-directional checkbox to have they go in both directions.

Bi-Directional

As mentioned, this takes whatever direction is selected and sends squares running off in the opposite direction, as well as the selected direction

Constrain Along Horz and Vert Axis

This causes squares to only move up/down/left/right, locking them to the X and Y axis. Movement at an angle is thus prevented. This is particularly useful with Random Squares checked.

Outlines

The Outlines checkbox turns on outlines around the squares. If this is unchecked then there will be no outline on the particles. Makes sense. There are a number of Outline parameters that are interrelated.

Outlines Only

If you also have the Outlines Only checkbox selected, then no squares are displayed, just the outlines that go with the individual squares. You can get some really interesting looks and effects with the outlines, as you can adjust their look and have them jiggle around.

Outline Width, Randomness

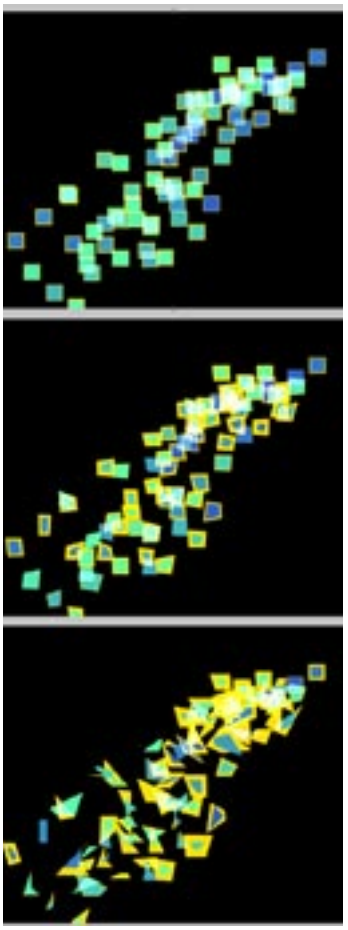
The Outline Width parameter determines how thick the outlines are. You can have them be a single pixel wide or so thick that it basically becomes a square. Of course, this can be animated, producing transition effects, or interesting displacement maps.

Outline Jitter

The Jitter parameters control how the corners of the outline jump around. The higher the Jitter setting the more the corners will move around and distort the square outline. This creates a cool 'wobbly' square type of look. This, like most other things, animate automatically, all you have to do is set up the amount you want it to distort and the Jitter Rate of Change.

Jitter Rate of Change, Randomness

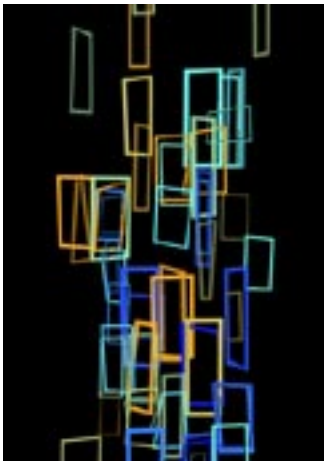
The Jitter Rate of Change controls how often the corners change the amount they're displaced by. This is extremely useful for avoiding any 'buzzing' if you're going out to video. If the squares are changing too rapidly, next thing you know you're sending your viewers into epileptic fits.



The crazy world of Outline jitters. From top to bottom, the jitter gets progressively worse. The top has no jitter at all, so the squares look smooth. The middle has a bit of jitter; you can see the outlines start to get a bit funky. They are still roughly square shaped, but definitely affected.

The bottom has outlines gone completely over the edge. The jitter value has been cranked up, and we're not even dealing with squares anymore in some cases. You can get some very funky dancing squares, by playing with the outlines and the jitter value.

Likewise, be careful about the Outline Width. A setting of 1, may be too small for broadcast video, and again, introduce buzzing. 2 or 4 is a much safer value when dealing with broadcast video.



An example of squares, with Outlines Only checked, and a Jitter value of 2.

Random Seed

The Random Seed simply sets the value that all the Randomness parameters use to determine their values. As long as the Seed remains the same, if you change a parameter and then go back to the original value, the animation will look the same. If you change the Seed, and go back to the original value, everything will look different.

Square Attribute Section

This section also controls the behavior of your shape particles. Most of these parameters determine the length of time that the shapes (such as a square) stay onscreen.

Minimum and Maximum Width/Height

All Squares are assigned two widths or lengths when they are created, a minimum and a maximum. The minimum is the least number of pixels a square will be when it appears on the screen. For example, if this is set to 10, then whenever a square appears on the screen it will be 10 pixels long. That is essentially it's starting length.

The maximum is what the square grows to. Once a square is created, it grows to reach the maximum width/length. If this is set to 35, and the minimum is set to 10, then the Squares will get created 10 pixels long, and will then grow to 35 pixels long.

The Horz. and Vert. growth speed determines how quickly the lines hit these limits. Be aware that the combination of LifeSpan, Growth Speed, and regular Speed, may cause the squares to disappear before ever reaching their maximum. This may be fine, but just be aware of it if the squares aren't quite getting big enough for you.

Birth Rate

Determines how many Squares are produced each frame. The higher the value, the more Squares you'll have.

Be careful about cranking this up, because this is another parameter that can seriously affect render times. The more Squares there are, the more stuff there is to keep track off, and the slower the filter renders.

This is especially true if the LifeSpan is set high. Once a square dies off, GridSquares doesn't need to keep track of it. However, if the Squares have a long LifeSpan and there's a high Birth Rate, things can get really slow. You have a lot of Squares being produced and they're lingering around for a long time.

LifeSpan

This controls, in frames, how long a square should exist on the screen before disappearing.

This can affect render times by having Squares linger around for a long time. This is fine if that's what you need, but if the Squares are going off the grid, and hence not visible, GridSquares is keeping track of a bunch of Squares it doesn't need to. Adjust this so that Squares are living as long as they need to, and not significantly longer.

Speed

Sets the speed the Squares move/grow at. It's pretty self-explanatory. The higher the setting, the faster the square will reach its maximum length, and the faster it will scoot around the screen afterwards.

Horizontal, Vertical Growth Speed

These parameters control the speed that the squares grow from their minimum width/height to their maximum width/height. Once a square has hit it's max width/height, it no longer grows.

This is in pixels/grid space per frame. So if you have it set really high, the squares will hit their maximum very quickly, unless the maximum is also very high.

Viscosity

Viscosity causes the Squares to slow down as they get further away from the producer point.

Normally, viscosity refers to the amount of resistance that comes from the thickness of a liquid or substance. For example, water has a higher viscosity than air, oil has a higher viscosity than water, mud has a higher viscosity than oil, and so on.

By increasing the Viscosity, you are essentially telling the Squares that they are moving through some substance with a certain amount of thickness. This will cause them to slow down, and if the Viscosity is high enough, they will stop completely.

Opacity

This determines how opaque or transparent the Squares are. The lower the setting the more transparent the Squares are, and the more you can see the background through them.

This is accomplished by adjusting the alpha channel, so if you save out a quicktime movie or photoshop file with an alpha channel, it will contain the transparency information from GridSquares. This allows you to take advantage of the transparency in other programs. Whether it's an editing program, 3D, or whatever.

Fade In, Fade Out

You can tell the Squares to slowly fade in or out over a given number of frames. In which case they will go from opaque (or whatever the opacity parameter is set to) to transparent and disappear.

This parameter is wholly dependant on the LifeSpan parameter, especially Fade Out. Fade Out will cause the square to start to fade as it gets close to the end of its LifeSpan. So if the LifeSpan is set to 60 frames, and Fade Out is set to 10 Frames, from frame 50 to 60, the square will fade to fully transparent. This can make the Squares exit from the screen a bit more graceful than just having it hit 60 and disappear.

Disappearing Squares tend to be somewhat jarring to the person watching, so it's best to fade the Squares out.

Fading In works the same way, if it's set to 10, over the first ten frames of it's life it will slowly go from completely transparent to whatever value Opacity is set to. Again, this allows the square to make a smoother entrance.

Color Section

It's a little difficult to figure out from the title... but this section controls the color of Geomancy's shape particles.

Color Chips, Gradient Bar

The color controls are pretty easy to deal with once you get your head around them. The color chips control what colors are in the gradient and where they're positioned. The gradient bar controls what colors the squares and outlines receive.

The squares can take the color from the bar sequentially or randomly. If they take it sequentially, then if the bar goes from red to blue to green, as new squares are created, they will be red, then blue, then green, then repeat. If the squares take their colors randomly, then each square just selects a color out of the complete range of the gradient bar, and what order the colors are in, doesn't make a difference.

Gradient Bar

Each color chip has three parameters: The color chip itself, an on/off checkbox, and a percentage slider.

The color chip is pretty self explanatory, just click on it and select a color, or use the eyedropper to select a color from the comp window or wherever.

The on/off checkbox, allows you to ignore the color chip. If you only want two colors in your gradient, there's no reason to mess around with all four of the color chips. Turn the last two of them off and just use colors 1 & 2. Likewise, if you want the squares to all be the same color, turn all the chips off except one.

Then there's the percentage slider. What this does is position the color on the gradient bar. The bar goes from 0 to 100 %. With 0 being at the very left, and 100 being at the opposite right. The higher the percentage the closer that color will be to the right edge.

If you have a gradient and you want it to blend smoothly from red to green to blue, you'd set red to 0, green to 50, blue to 100, and turn the fourth chip off.

Moving the colors around is pretty easy, just move the percentages. If you wanted your gradient to be solid red in the first half, then blend between green and blue, you'd set red to 50, green to 75, and blue to 100. With the start and end colors, if they are at anything other than 0 and 100, all the space before the start or after the end will just be a solid color. So with red set to 50, from 0 to 50 will be solid red. If we'd set blue to 75, from 75 to 100 would be solid blue, and so forth.

Take Color From Source

This checkbox causes the gradient to be ignored and all squares get their color from the underlying source image (the layer the filter is applied to). This is based on where the squares originate at, so if you have a very small producer point, most likely all squares will have the same color.

For example, if you have a 20 pixel by 20 pixel producer point, only that 20x20 area will be sampled for colors. If all the pixels in that area in the underlying photograph, movie, or whatever are the same color, all your squares will be the same color.

Blend Modes

This controls how the Squares lay on top of each other. These modes work like the transfer modes you're accustomed to in After Effects.

Color Cycle

This specifies how long it takes for the squares to go through the entire gradient. It's set in frames and as the squares are getting produced and pulling colors from the gradient, it's controlling how fast you move from 0 to 100. Once it hits 100, it flips over back to 0.

If Color Cycle was set to 200, and the gradient was red to blue, it would take 200 frames to go all the way from red to blue. Once it hit blue, it would turnover and start at red again, and repeat forever (or until your timeline ran out).

If you want to control when your squares start pulling a given color, be very aware of what this is set to. You can give your squares specific colors at specific times if you match up the gradient to this control. For example, set Color Cycle to 90, and set red to 33 in the gradient, blue to 66, and green to 100. That divides the gradient up into thirds, and since a third of 90, is 30, every 30 frames the squares will come out a different color.

Randomly Take Color From Gradient

With this checkbox selected, the color for the squares are taken randomly from the gradient. Color Cycle Speed is ignored.

Outline Color

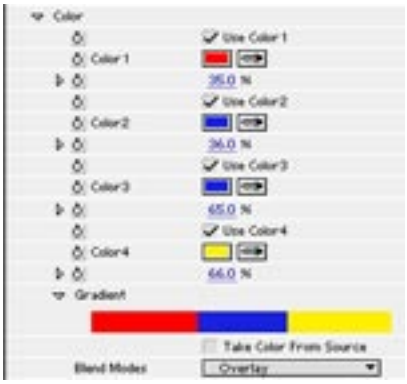
This defines what color the outlines are. For a complete explanation of outlines, see the Square Setup section. All outlines will be this color, unless…

Take Outline Color From Gradient

This checkbox causes the outlines to take their colors from the gradient. Usually, it will take the colors sequentially from the gradient, going from left to right. The Color Cycle Speed works with this as well.

As you can see, you can use any of the color chips to make the gradient. In this case we've used only two of them. Notice how before 10%, everything is red, and after 65%, everything is blue.

Here we've used all 4 color chips. Notice at the end, they yellow chip is pushed up against the orange chip. Since they're so close, there really isn't any blending.



We now take the no blending thing to an extreme. Using the two middle chips to section off a single color, we now have a 3 color gradient with no blending. This means you will only get those exact 3 colors. If there were blending, you could get some ugly, unexpected colors.

Outline Gradient Offset

This moves the point on the gradient at which the outlines are sampling their color from. This prevents outlines from having exactly the same color as the squares, making the outlines blend with the squares.

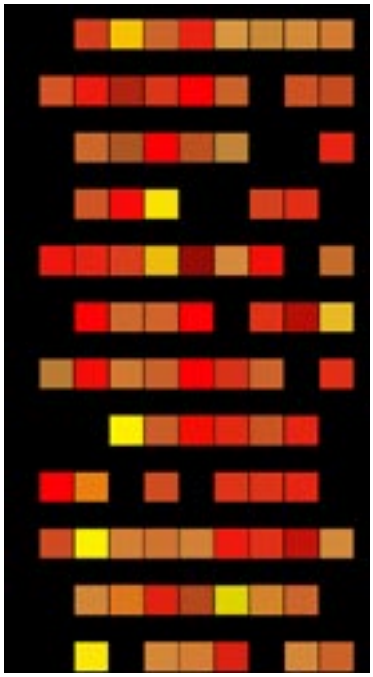
This is a percentage, so if it's set to 50%, the point that the outlines are sampling, is half way across the gradient from the point that the squares are sampling from.

Take Outline Color Randomly From Gradient

This just forces the outlines to randomly sample the gradient. This causes most of the other outline controls to be ignored. The outlines should randomly pick a spot on the gradient and sample from it.

Composite On Original

Composites the Squares on top of the original layer.



With the Grid Adherence set to zero, the squares follow the grid exactly. The Grid Adherence parameters make it very easy to confine squares to the grid, or have them completely chaotic and not follow the grid at all.

Grid Adherence

The controls in this section simply determine how closely the squares stick to the grid. These parameters are in many cases tied into behavior parameters from other sections, as you will see in a moment.

Horizontal, Vertical Grid Adherence

Grid Adherence is the maximum number pixels that the squares will be allowed to stray off the grid. Once a square is created, it's given a GA value, which doesn't change. GA can be animated, but the value changes only affected newly created squares, not squares already visible.

This allows you go have your lines form a very obvious grid or make it look like they are totally random with no underlying structure.The lower the Grid Adherence amount, the more tightly they will stay to the underlying grid. If you turn on 'Show Grid' in the you'll be able to see the underlying grid and get an idea how the squares are positioned.

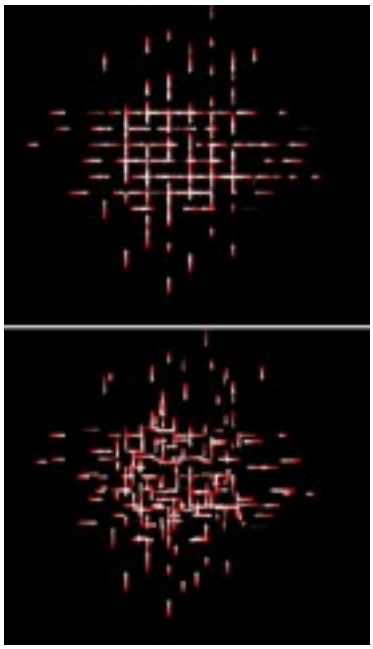
It doesn't matter how big or how many rows or columns the grid has. The squares will use the grid as a guide when they are produced. Unlike behavior in the GridLines plugin, the squares aren't constrained by the grid after they've been born.

Once they've been created, they're free to move in any direction they choose, unless Constrain Along Vertical and Horizontal Axis is checked (located in the Square Setup section).

In order to get the squares to stay put, you need to set the Speed parameter to 0, as well as the Growth Speed rates. All of these options are in the Square Attribute section.

It is possible for the squares to overlap, even if Grid Adherence is set to zero and speed is set to zero. If the squares are bigger than the grid spaces, the squares will overlap. As you can see in the sidebar, if the squares are smaller than the grid spaces, everything thing is nicely arranged. If the squares had been bigger than the grid spaces, you can see that there'd be a lot of overlap.

This might be an effect you want. However, just be careful when setting up the size of the squares and the size of the grid (and the number of rows and columns).



The results of Grid Adherence. The top image has Grid Adherence set to 0.The bottom has it set to 50. The higher the value, the more likely the squares will be pushed completely out of their spots on the grid.

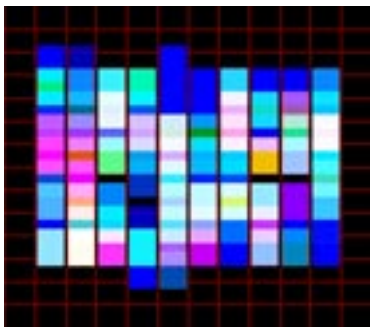
Horizontal, Vertical Grid Adherence Randomness

Grid Adherance Randomness, of course, just varies the amount that the line is off the grid, based on the Grid Adherance value.

Grid Adherance Probability

Probability determines the likelihood of a square being off the grid. If a square is striving to be off the grid, it has to get through the Probability parameter first. If it gets past that, then it has to get an amount determined.

The amount a square is off the grid is determined by [Grid Adherance] * [Grid Adherance Randomness] (unless Randomness is zero). Of course, Randomness varies from square to square, so there's no precise way of know this unless Randomness is 0.



What happens when squares get too big for their britches. In this case the height of the squares is larger than the height of the grid spaces, so you get overlapping squares. Which can created a nice effect. Just make sure it's the effect you want.

GridLines

Welcome to GridLines! This plugin is very powerful for creating random, animated elements. GridLines was designed to create the random animating lines that you see quite often in broadcast design. These lines aren't normally hard to produce, but it can be quite tedious to animate them.

GridLines is meant to simplify the process of creating these lines. Letting you, the hip, cutting edge designer, spend more time on the design of your project, and less time fiddling around with the technical details.

Like GridSquares, this plugin is based around a particle system that is designed to do nothing but lines. This makes it easy to customize the lines, control their behavior, get the look you want, and move on to other things. This ease is facilitated by Randomness controls that automatically randomize the appearance and behavior of shapes.

The Secret Behind GridLines

In order to do its magic, GridLines first sets up a grid, then draws the lines on top of it using the rows and columns for guides. You then tell the lines how they're going to look, how to behave, and how rigidly they're going to follow the grid. It's all about setting up the grid, then telling the lines to get a little crazy, and ignore it. Of course, you can tell your lines to behave as well.

The Grid itself can be set to any size, rotation, position, etc, and can be set up with any number of rows and columns. How you set up the grid goes a long way in determining how the animation is going to look. Make sure you have some idea of what you want to do before you jump in.

In this GridLines project, we've set the lines to take their color from an underlying image. The Grid defines an interesting structure. With the widely varying hicknesses, courtesy of the Thickness Randomness parameter, we end up with lines almost acting as brush strokes.

Grid Section

This section allows you to determine where the GridLines particles sit against a (typically) invisible, underlying grid. The grid is your key to directing the lines initially.

Height, Width

Should be pretty self explanatory what these guys do. Basically just sets the size of the grid in pixels.

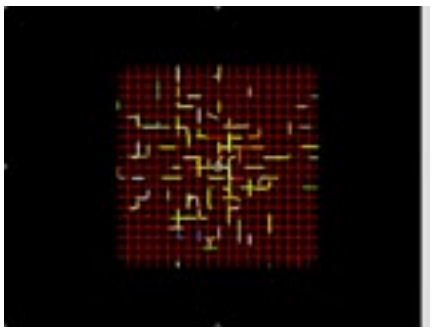
You need to be careful about this, because if you set the grid size too small, the lines will go off the edge. If you set the grid size too large, bigger than your comp for example, the animation will be slowed down unnecessarily. Unnecessary because you're telling GridLines to keep track of lines you're never going to see. Obviously,

if you're going to move the Grid position or rotate it, the lines might move into view. We've set the lines to take their color from an underlying image.

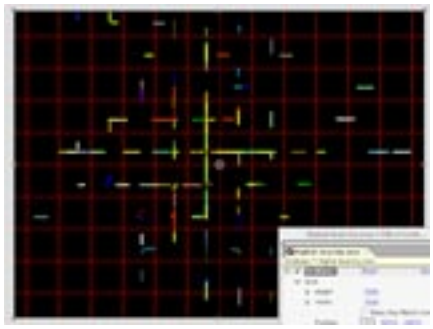
With the widely varying thicknesses, courtesy of the Thickness Randomness parameter, we end up with lines almost acting as brush strokes. The Grid creates an interesting structure.

Here, the grid is too small and doesn't cover the entire screen. The grid is 1000x1000 on a 640x480 layer. This has no benefit, unless you're going to move or rotate the grid.

Grid size is one of the things that affects the rendering speed, so don't make this any bigger than you need it. If you just need a bunch of lines running across the width of the screen, at the top or bottom, don't make the grid the size of the entire comp. Just make it a 100 or so pixels high, and however wide the comp is. This will save the extra hit on computing power, a large grid would have given you.



Here, the grid is too small and doesn't cover the entire screen.



The grid is 1000x1000 on a 640x480 layer. This has no benefit, unless you're going to move or rotate the grid.

Make Size Match Comp Size

This just makes the grid match the size of the comp. If this is checked it ignores the Height and Width. If the Height and Width is set to 300 and 200, respectively, and the comp size is 640 x 480, turning this option on will make the grid size 640 x 480.

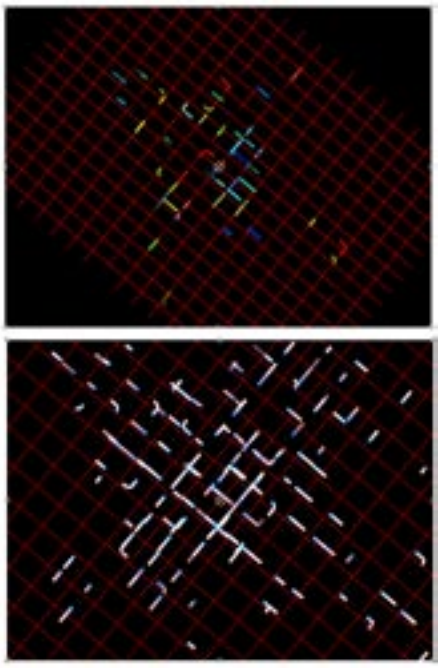
Position

This sets the center of the grid.

Rotation

Sets the rotation of the grid. Since all the lines follow the grid, it's not possible to rotate the lines themselves. To get around this, use the Grid Rotation. This allows you to create diagonal lines by setting the angle you want the grid to be at. This can be animated, and will rotate all lines on the grid by the same amount.

If you want to have diagonal lines and straight lines in the same project, you'll need to use two copies of the filter. Something to keep in mind, tho: If you're going to do this, make sure your grid is larger than your layer (See Grid Width/Height). If you have a 640x480 layer and the grid is set to 640x480 as well (covering the layer with lines), when you rotate it, around the corners you'll see the edge of the grid. See the illustration in the sidebar for an example.



The top image shows an example of the grid being rotated when it's the size of the layer. Notice the edges show in the corners. The bottom image, shows a rotation with the grid size being large enough to not have the edges show.

Rows, Columns

Rows and Columns define the division of the grid. You can see them by turning on the 'Show Grid' parameter. The lines will follow these divisions, so how many you have and how they're spread out (due to the size of the grid) will determine the structure of your animation.

There are a variety of controls that we'll get into shortly that determine how rigidly the lines follow these divisions (see the Grid Adherence and Turning Contorls sections), but, by and large, the divisions will set up the underlying foundation.

Think of the rows and columns as grooves in a flat piece of metal. If you pour liquid into these grooves, the liquid will only flow into the criss-crossing channels (assuming they don't overflow). This is basically what happens with the lines. The rows and columns determine where the lines will flow. If there's only two columns in a 400 pixel wide grid, there's really only two places lines are going to appear.

These are not animatable since they're the underlying structure for the whole animation. Like any good foundation it needs to stay stable. There are other ways to change the look of the animation, but this has to remain the same, throughout. Consider what you're going after when you set these up.

Again, selecting 'Show Grid' will make setting this part up easier.

Producer Point Section

This section controls the Producer Point. The Producer Point defines where the lines start at. This area can be as small as a single pixel or as large as the entire grid area.

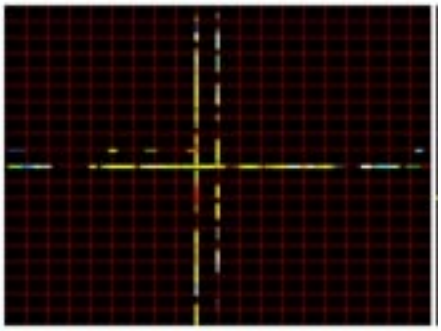
The grid defines where the lines can go, and the Producer Point defines where they start on their journey. Consider the grid a range, and the Producer Point the home on the range. The lines can go anywhere they want on the range, but they all have to start out back at home.

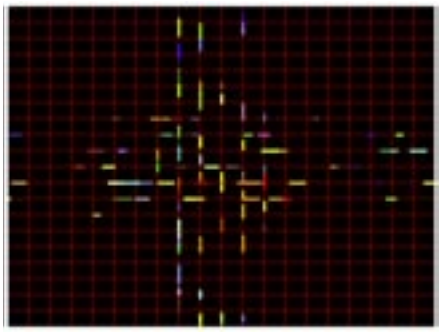
Height, Width

The size of the Producer Point is set in pixels by the Height and Width parameters. While you can set these numbers higher than the height and width of the grid itself, GridLines is smart enough to cap the Producer Point size at the size of the grid. There isn't any point in having lines created outside the grid as there are no rows or columns for them to be created in.

Position

The Position Point determines where the center of the Producer Point is located on the grid. Again, you CAN position this off the grid, but lines will only be created in the area that overlaps onto the grid. If no portion of the Position Point is on the grid, no lines will be created.



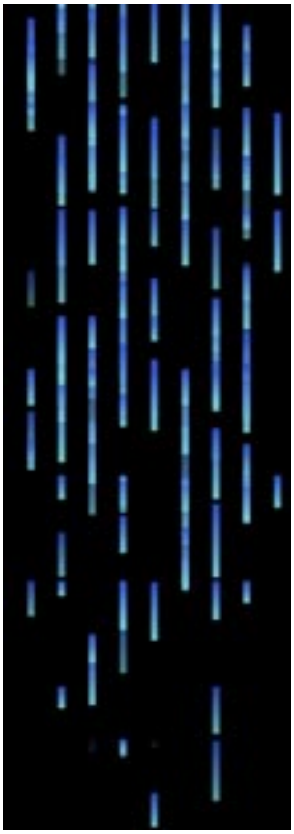


This shows a couple things. One, notice how the lines, align themselves to the grid, displayed by the red lines. By default all lines are limited to these 'grooves'.

Secondly, in the top image, the Producer Point is 20x20. In the bottom image it's 150x150. Notice that the lines are much more spread out in the right image. You may need to zoom in on the image if you're reading this in PDF format.

If you have a large Producer Point, lines can be created anywhere within it. This creates a much different effect than if it's small, since in that case, all the lines seem to be coming from the same point, and moving outward. If it's large, there is no central point, so lines are created wherever, and head off in the direction (or directions) you have specified.

A large area produces a much more random effect, especially if Random Lines is checked in the Line Setup section (we'll get to that next).



Constraining the movement to only downwards and selecting a blue gradient, gives us a 'rainy' sort of look. There are a variety of ways that you can control or constrain how the lines behave.

Line Setup Section

This section controls some of the behavior of your line particles. Generally, these parameters have to do with the type of line or the direction it moves.

Lock Birth Attributes

This attribute affects some of the parameters in the Line Attribute section. Specifically: Speed, Thickness, Viscosity, and Opacity. If this is checked, each line that gets created will remember its settings for those attributes, and if you animate those attributes, lines created before the change in values, will not be affected.

For example, if you animate Thickness from 2 to 6, lines created before the keyframes will always have a Thickness of 2. Lines created afterwards will have a Thickness of 6, so you'll have lines with varying Thicknesses on the screen at the same time.

If LBA is not checked, when you animate Thickness from 2 to 6, ALL lines on the screen, and all those created afterwards will be 6 pixels wide. This is how most of the attributes behave regardless of whether LBA is on or off.

Line Unit

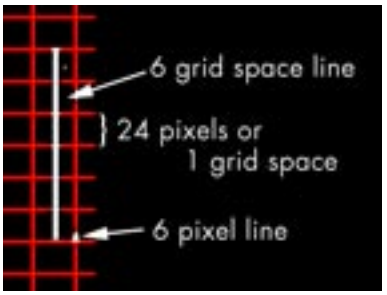
This determines what measurement unit is used for the lines, in the attribute section. There are two options: Pixel and Grid Space. Pixel is fairly obvious, but what about Grid Space?

Grid Space is the space between rows or columns. The area between one column and the next, or one row and the next, is one grid space. If you have 10 columns in a 200 x 200 pixel grid, each grid space will be 20 x 20 pixels (200/10). If you had 4 columns, each grid space would be 50 x 50 pixels (200 / 4).

Obviously, you have to be careful about this, because just up and changing this will seriously change the animation. If you change it to Grid Space without adjusting the attributes, you may get extremely long render times.

If Minimum Length (in the Line Attributes Section) is set to 10, if pixels is selected the minimum length of a line is 10 pixels. So far, so good. If we now change to Grid Spaces, if a grid space is 50 pixels by 50 pixels, at a minimum of 10, the minimum goes from 10 pixels, to 500 pixels (10 (the minimum) x 50 (the size of a grid space)).

This results in a lot of extra calculation, which will slow things down while you're trying to change everything to a more normal amounts for the Grid Space unit. So set up your attributes to something that makes sense for the particular unit, BEFORE switching over to it.



Your choice of Line Unit can have a very signficant on the look of your lines, and how they animate. In this example, each Grid Space equals 24 pixels.

If you had less rows and columns the number of pixels would be larger, if you had more rows and columns, the number would be smaller.

Divide the size of the grid by the number of rows or columns, to get the pixel value of each grid space.

Random Lines

This checkbox causes lines to be sent off in all directions. You have no control over where they go, they get created and are sent off and running in some random direction. Actually, the Turning Controls still affect random lines, so you do have some control over what they do, but not much.

With this selected, the Vertical Lines and Horizontal Lines pop-ups are ignored. The lines are quite happy running about in any direction they choose and aren't about to be hampered by a couple silly pop-ups. That said…

Vertical Lines, Horizontal Lines

If Random Lines is not checked, then the lines are constrained by these two pop-ups. One pop-up to control how the lines behave in the vertical direction, and one to control the horizontal direction.

You can either set them to go in one direction, or use bi-directional to have they go in both directions. In the case of vertical, that means up and down, with horizontal, it means right and left.

Having both of these set to bi-directional is essentially the same as having Random Lines selected, since lines are free to go off in any direction they choose.

If you have the Turn Probability parameter set to anything other than zero, the lines will still turn and go in the crossing direction. If you want the lines to occasionally turn, but to make sure they return to the original direction, set the Perpedindicular Max Length to a low amount.

Straight Lines Only

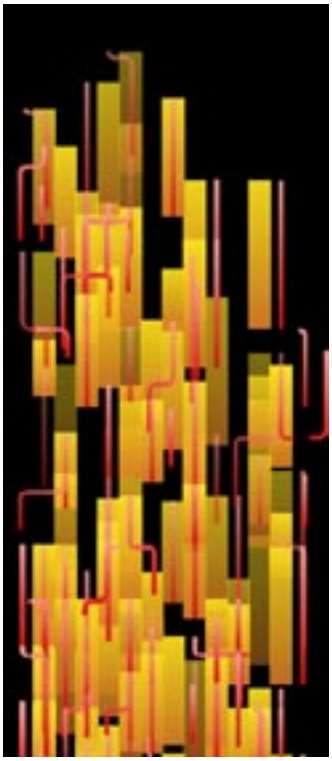
As you might guess, this causes the Turning Controls to be ignored and only straight lines are created. This works with both the pop-ups and the Random Lines checkbox.

This works better with a large producer point. If you have the producer point set to only a few pixels, the lines will essential be created right on top of one another and since they don't turn, you'll just end up with a few lines going out from the center. This may or may not be what you want, so be aware of it.

As you can see your choice of units can have a very significant effect on the look of your lines, and how they animate. In this example, each grid space equals 24 pixels. If you had less rows and columns the number of pixels would be larger, if you had more rows and columns, the number would be smaller. Divide the size of the grid by the number of rows or columns, to get the pixel value of each grid space.

Line Wrapping

This causes any line that goes off the screen to reappear on the other side of the screen as a continuation of its path.



Two instances of the plugin. The yellow lines were created by setting Thickness to be larger than the Grid Spaces. This results in lines overlapping. We also used the 'In Front' blend mode, which is a very useful blend mode for this plugin.

Line Attributes

This section also controls the behavior of your line particles. Some of these parameters determine the length of time that the lines stay onscreen. Others determine visual attributes of the lines.

Minimum Length, Maximum Length

All lines are assigned two lengths when they are created, a minimum and a maximum. The minimum is the least number of pixels a line will be when it appears on the screen. For example, if this is set to 10, then whenever a line appears on the screen it will be 10 pixels long. That is essentially it's starting length.

The maximum is what the line grows to. Once a line is created, it grows to reach the maximum length. If this is set to 35, and the minimum is set to 10, then the lines will get created 10 pixels long, and will then grow to 35 pixels long. Once it's hit 35, the entire line will start moving, since adding a pixel onto the front requires removing a pixels from the back to keep the length at 35.

The line doesn't move until it reaches it's maximum. Until then it just grows from wherever it was created from the producer point. To have the lines start moving immediately, set the min and max to be the same value.

Birth Rate

Determines how many lines are produced each frame. The higher the value, the more lines you'll have.

Be careful about cranking this up, because this is another parameter that can seriously affect render times. The more lines there are, the more stuff there is to keep track off, and the slower the filter renders.

This is especially true if the LifeSpan is set high. Once a line dies off, GridLines doesn't need to keep track of it. However, if the lines have a long LifeSpan and there's a high Birth Rate, things can get really slow. You have a lot of lines being produced and they're lingering around for a long time.

LifeSpan

This controls, in frames, how long a line should exist on the screen before disappearing. This can affect render times by having lines linger around for a long time. This is fine if that's what you need, but if the lines are going off the grid, and hence not visible, GridLines is keeping track of a bunch of lines it doesn't need to. Adjust this so that lines are living as long as they need to, and not significantly longer.

Speed

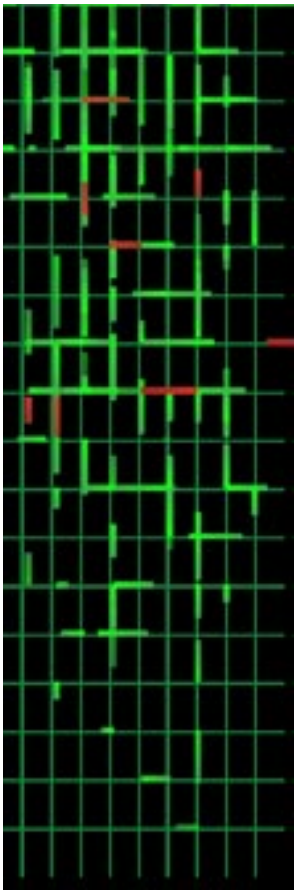
Sets the speed the lines move/grow at. Pretty self-explanatory, the higher the setting the faster the line will reach it's maximum length, and the faster it will scoot around the screen afterwards.

Thickness

Allows you to control the thickness or width of the lines.

Again, nothing particularly special about this one. The higher the value, the fatter the lines will be. Randomness is particularly useful with this one as it varies up the look of your composition.

for instance, in our example at right, we used two different instances of GrdiLines with different Thickness and Thickness Randomness. This created different colored lines that behave the same way and seem to interact with each other. animation quite a bit. If all the lines are the same Thickness, they tend to blend into one another. Varying line weights make the individual lines stand out.



By using two different instances of GridLine and varying their line Thickness, we created lines that behave the same way and seem to interact with each other.

Viscosity

Viscosity causes the lines to slow down as they get further away from the producer point.

Normally, viscosity refers to the amount of resistance that comes from the thickness of a liquid or substance. For example, water has a higher viscosity than air, oil has a higher viscosity than water, mud has a higher viscosity than oil, and so on. This allows you to have the lines slow down, without having to keyframe the speed.

By increasing the Viscosity, you are essentially telling the lines that they are moving through some substance with a certain amount of thickness. This will cause them to slow down as they move, and if the Viscosity is high enough, they will stop completely.

Opacity

This determines how opaque or transparent the lines are. The lower the setting the more transparent the lines are, and the more you can see the background through them.

This is accomplished by adjusting the alpha channel, so if you save out a QuickTime movie or photoshop file with an alpha channel, it will contain the transparency information from GridLines. This allows you to take advantage of the transparency in other programs whether it's an editing program, 3D, or whatever.

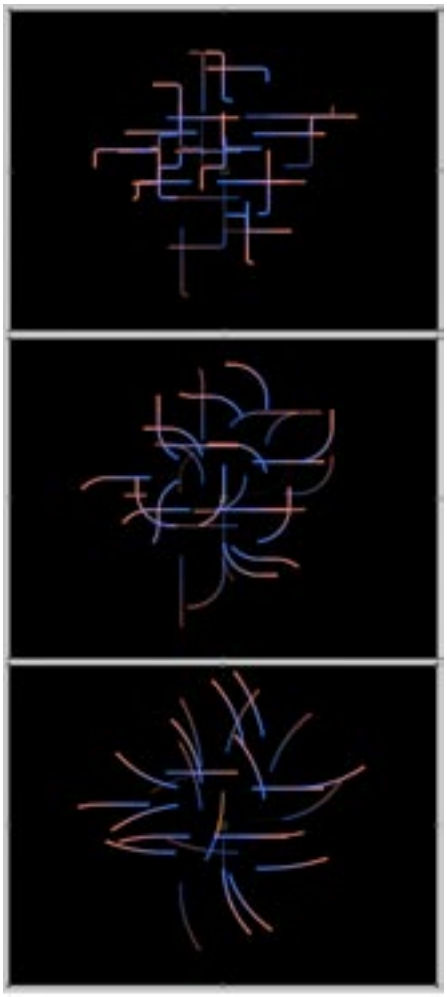
Fade In, Fade Out

You can tell the lines to slowly fade in or out over a given number of frames. In which case they will go from opaque (or whatever the opacity parameter is set to) to transparent and disappear.

This parameter is wholly dependant on the LifeSpan parameter, especially Fade Out. Fade Out will cause the line to start to fade as it gets close to the end of it's LifeSpan. So if the LifeSpan is set to 60 frames, and Fade Out is set to 10 Frames, from frame 50 to 60, the line will fade to fully transparent. This can make the lines exit from the screen a bit more graceful than just having it hit 60 and disappear.

Disappearing lines tend to be somewhat jarring to the person watching, so it's best to fade the lines out.

Fading In works the same way, if it's set to 10, over the first ten frames of it's life it will slowly go from completely transparent to whatever value Opacity is set to. Again, this allows the line to make a smoother entrance.



An example of Corner Size. From top to bottom: Corner Size = 5, CS = 50, and CS = 200. Notice that lines don't need to be larger than the corner. The entire line can be turning.

Corner Size

This defines the radius of the curve when the lines turn. The larger this is, the sooner the lines will start their turns, and the longer it takes for the lines to finish turning.

Color Section

This section, of course, controls the coloration of the line particles. There are really only three parameters, so it's a pretty simple way of working.

Start Color, End Color

These two color chips define the colors at the start and end of the lines. The colors are blended along the length of the lines.

Ignore End Color

Causes the lines to only recognize the start color, making all the lines that solid color.

Very useful if you want to animate the color of the lines over time, but want to keep them solid colors. You could do the same thing by animating the Start and End colors together, but this simplifies things a bit.

Blend Modes

How the lines lay on top of each other. These work like normal transfer modes.

A couple of modes that don't show up produce an interesting effect: Behind and In Front. Behind causes new lines to be created behind older ones, and In Front causes the new lines to be created on top of the older ones.

Composite On Original

Composites the lines on top of the original layer.

Grid Adherence

The controls in this section determine how closely the lines stick to their grid.

Grid Adherence Amount

Grid Adherence is the maximum number pixels that the lines will be allowed to stray off the grid. Once a line is created, it's given a GA value, which doesn't change. GA can be animated, but the value changes only affected newly created lines, not lines already visible.

This allows you go have your lines form a very obvious grid or make it look like they are totally random with no underlying structure.

The lower the Grid Adherence Amount, the more tightly the lines will stay to the underlying grid. If you turn on 'Show Grid' you'll be able to see the underlying grid and get an idea how the lines behave.

It doesn't matter how big or how many rows or columns the grid has. The lines will follow it and stick to the rows and columns within the range that you set in Grid Adherence.

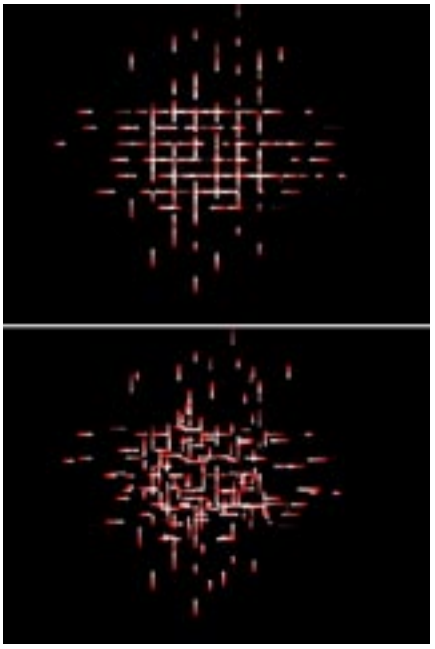
Grid Adherence Randomness

Grid Adherance Randomness, of course, just varies the amount that the line is off the grid, based on the Grid Adherance value.

Grid Adherance Probability

Probability determines the likelihood of a square being off the grid. If a square is striving to be off the grid, it has to get through the Probability parameter first. If it gets past that, then it has to get an amount determined.

The amount a square is off the grid is determined by [Grid Adherance] * [Grid Adherance Randomness] (unless Randomness is zero). Of course, Randomness varies from square to square, so there's no precise way of know this unless Randomness is 0.



A good example of what Grid Adherence does (or doesn't) do for you.

The top image has a Grid Adherence setting of zero. All the lines stick pretty well to the grid and, in this case, their movement is constrained to moving only horizontal or vertically. No turning.

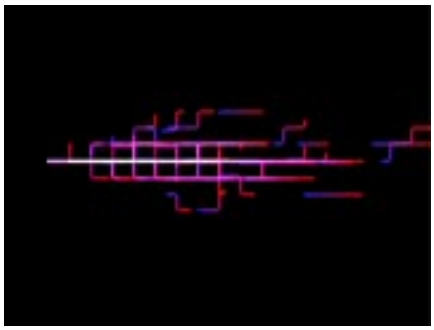
The bottom image, has a Grid Adherence setting of 10. You'll notice that the lines are pretty well scattered. If you're looking to make your grid obvious, set Grid Adherence to a low number. Otherwise, if you want a more random look, set it to a high number.

Turning Controls

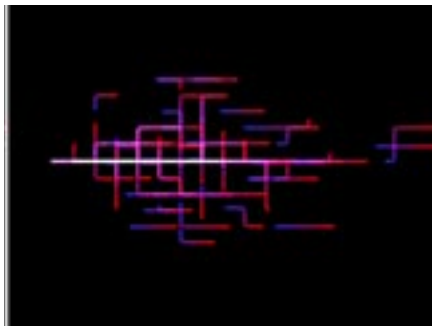
This section controls how the GridLines behave when they come to an intersection in the grid. Basically, they will either go straight or turn. Let's see how that works.

Turn Probability

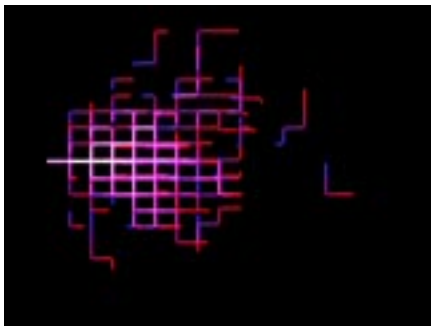
Sets the likelihood that a line will turn whenever it comes to an intersection of the columns and rows in the grid. If it's set to 25%, for example, one fourth of the times it comes to an intersection, it will turn.



An example of how Perpendicular Max Length (PML) works. This really works in conjunction with Turn Probability. If Turn Probability is low, if a line turns, it's unlikely to turn again so that it goes in it's original direction. PML acts as a line herder, making sure they keep going in the direction you want them to go. In the left image, PML is set to 1, meaning no line can go more than one grid space before turning back to it's original direction. On the right, the PML is 5. This results in a much wider spread of lines, as lines can go as far as 5 grid space before turning back. Here the Turn Probability is set to 5%, a very low value.



In this version, the PML is 5. This results in a much wider spread of lines, as lines can go as far as 5 grid spaces before turning back. Here the Turn Probability is set to 5%, a very low value.



Here the Turn Probability is set to 50%, and the PML is set to 10. Compare this to the other two previous examples. In the other two, the lines were pretty much headed in one direction, with a little variation (which increased as the PML increased). With this example, lines don't really have a clear direction. They're going in all sorts of directions, and the spread of the lines is huge. This may be what you want, but if you want your lines to stick to one direction, make sure both Turn Probability and PML are set very low.

Perpendicular Max Length

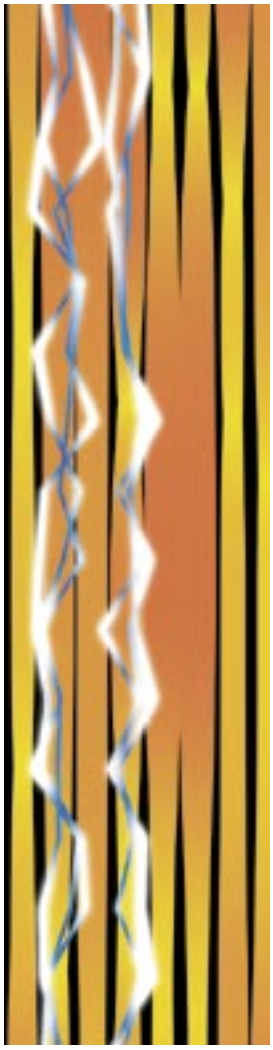
If the Turn Probability is low, lines will occasionally turn, but they won't turn back in the direction they were going. To prevent lines from never turning back in the direction you really want them to go, Perpendicular Max Length was introduced.

If a line turns, this sets the maximum number of grid spaces it can go before being forced to turn again. No matter what the Turn Probability is set to, after it's gone this number of grid spaces, it turns back to the direction it was going.

HairLines

HairLines is a filter designed to produce random lines. Squiggly flowing lines, straight lines, lines with no end, lines with two ends, all sorts of different types of lines.

Lines are something that broadcast designers use a lot for various types of effects, and as elements in a composited piece to add something visually interesting. While these lines are generally easy to create, animating them can be somewhat tedious. HairLines allows you to easily create lines and animating them without spending much time sweating the details.



Three instances of the HairLines plugin. Two high Frequency angular instances that have a glow applied to them. The orange/yellow background was created with a smooth, large Line Width and a low Fequency value. Read on to learn what these parameters mean.

The Secret Behind HairLines

Like the other Geomancy plugins, the secret behind the HairLines is its grid. The grid allows you to set up where the lines are and what they look like. It's all about setting up the grid, telling the lines to ignore or follow it, and how they should behave in doing so.

HairLines isn't technically a particle system like the other Geomancy filters. However the lines are affected by ripples, which behave similar to particles as they move through and distort the lines.

In fact, you could say that the other secret behind HairLines is its Ripples. They define the characteristics of a line jsut as much as the line characteristics do.

The lines and ripples have various attributes that can vary from line to line or ripple to ripple. Hence, the randomness parameters are in this filter as well. Lots of ways to get different looks and effects out of some simple lines.

Grid Section

This section allows you to determine where the HairLines particles sit against a typically invisible, underlying grid. (We'll continue with our term 'particles', though these could also be called units.) The grid is your key to directing the lines initially.

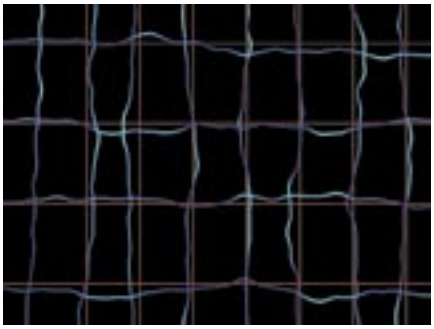
Height, Width

Sets the size of the grid in pixels.

You need to be careful about this parameter. If you set the grid size too small, the lines won't cover the entire screen. If you set the grid size too large, bigger than your comp for example, the animation will be slowed down unnecessarily.

We say 'unnecessary' because you're telling HairLines to keep track of lines, ripples and information you're never going to see. Obviously, if you're going to move the Grid position or rotate it, the lines might move into view.

This is one of the things that affects the rendering speed, so don't make this any bigger than need it. If you just need a bunch of lines running across the width of the screen, at the top or bottom, don't make the grid the size of the entire comp. Just make it a 100 or so pixels high, and however wide the comp is. This will save the extra hit on computing power, a large grid would have given you.



Unlike GridLines and GridSquares, the HairLines Grid doesn't constrain or control the lines. The grid simply defines how many Horizontal (rows) and Vertical (columns) lines there are. The lines are controlled by the ripples.

Make Size Match Comp Size

This just makes the grid match the size of the comp. If this is checked it ignores the Height and Width. If the Height and Width is set to 300 and 200, respectively, and the comp size is 640 x 480, turning this option on will make the grid size 640 x 480.

Position

This sets the center of the grid.

Rotation

Sets the rotation of the grid. Since all the lines follow the grid, it's not possible to rotate the lines themselves. This allows you to create diagonal lines by setting the angle of rotation you want the grid to be at. This can be animated, and will rotate all lines on the grid by the same amount.

If you want to have diagonal lines and straight lines in the same project, you'll need to use two copies of the filter.

Rows, Columns

Rows and Columns define the division of the grid. You can see them by turning on the Show Grid parameter.

Unlike GridLines, the lines don't really follow the rows and columns specifically. Rows and columns are really used to define the number of lines you want to have, and the lines aren't really constrained by them, as Gridline lines are. So if you want to increase/decrease the number of lines, increase/decrease the number of rows and columns.

You'll notice that there is no Grid Adherence set of parameters. The lines are controlled, for the most part by the ripples and that's the important thing to remember.

Show Grid

This option just shows the grid. Yup. Not as useful as in the other Geomancy filters, since HairLines don't follow the grid as specifically.

Line Spacing Randomness

This is as close to grid adherence as the HairLines filter gets. This parameter randomizes the distance between the lines. If you have Show Grid turned on and you crank this parameter up, you'll notice the lines move away from the Grid in random directions.

This can create a nice varied look to your animation or effect. Especially if there you're using angular lines or very small ripples, where the lines are basically straight. If they're straight, the grid will be more obvious, since the lines are straight and they're regularly spaced out.

Line Compression

This allows you to squeeze the lines together.

The Line Spacing Randomness parameter allows you to vary the distance between the lines. Line Compession allows you to push them all closer together around a center point.

Compression Controls. This is what the line compression does for you. The top image is without any compression. Just your standard lines sticking to the grid.

The middle image is with Line Compression all the way down to -100. Negative compression can create some very cool effects.

The bottom image has compression all the way up at 100. This should proably be called decompression, since it spreads the lines out. This is usually useful in situations where you want to animate the lines spreading out.

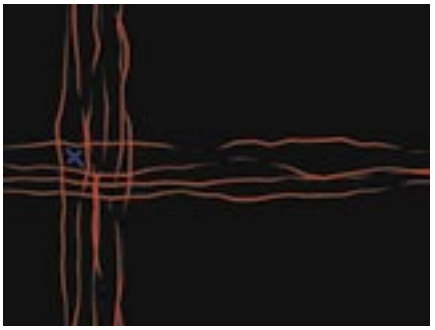
Compression Center

The Compression Center allows you to set this center point, which behaves a bit like an anchor point. All the lines will compress around this point. Usually, it's in the middle, so all the lines get evenly compressed. If there are 4 lines, 2 will compress towards the center from the left, and two will compress from the right.

If you adjust this to one side, say towards the left edge putting three lines to the right and one to the left, all three to the right will compress towards the left edge of the screen.

Actually, depending on how you set up the Line Compression, the lines will compress towards or away from the Compression Center.

If you set it to a negative amount, the lines will move towards the center, if you set it to a positive amount, the lines will move away from the center, spreading themselves out.



The Compression Center is where the blue X is. All the lines squash or scale around this point. Compression can yeild some really nice effects, so definitely spend some time playing with it.

Lines Section

This section controls some of the behavior of your line particles. These parameters have to do with the type of line or their visual properties.

Normal Line Width

This sets the width of the lines in pixels. If you remove the ripples and turn randomness to zero, all the lines will be this width. The ripples affect what size the lines are at any given point, and the randomness varies the default width between different lines.

Stressed Line Width

When the ripples distort a line, this sets the width of the line when it's distorted or stressed (imagine a rod of metal being bent). This parameter goes from -100 to 100 percent. In the positive direction, the stressed area of the line becomes bigger, in the negative direction the line becomes smaller.

The ripples will always change the size of the lines, but by increasing or decreasing the Stress, you'll make the size change more dramatic.

Normal Color, Stressed Color

The Line Widths have accompanying color chips which determine what color the line will be. With no ripples, the lines will just be the Normal Line Color. Where ripples distort the line, the line will become the Stressed Line Color. In between the colors blend together.

This is meant to reproduce, to some degree, the way you might see the color of metal change as it gets bent and twisted. The color on the bend, particularly if it heats up, is different than the normal metal. That's the idea at work here. Obviously you can make both chips the same color and make the lines all the same color, ripples or not.

Blend Amount

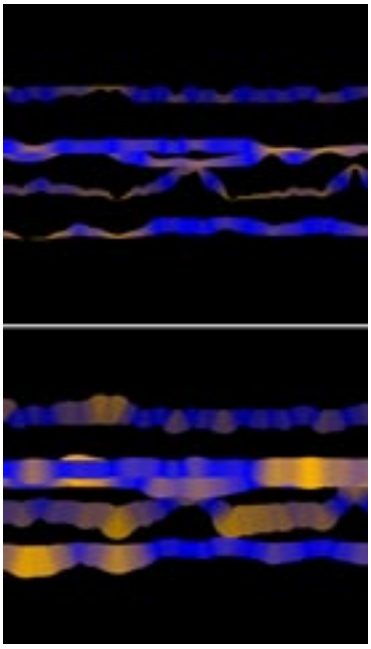
Determines how much the two colors blend together. This effectively moves the midpoint of the gradient. Setting it to 0% will cause the lines to only be the Normal Color and setting it to 100% results in lines that are only the stressed color. The closer it is to either extreme will result in a larger area of the line be taking up by the appropriate color.

Distortion

This works in conjunction with 'Affect' pop-up in the Ripples Section (see for more info). If that pop-up is set to Horizontal or Vertical, then this controls how much of an effect this causes. Setting it higher will increase the amount that the crossing lines are affected.

Opacity

Sets how opaque or transparent the lines are going to be. The lower the setting the more transparent each line will be. This behaves like normal Opacity in After Effects and should be familiar to you.



An example of how the Stressed Line Width affects the look of your lines.

The top image has Stressed Line Width to -100. and the bottom image has it set to 100.

Notice that the points where the lines curve are the 'stressed' points. They're different in color (orange), and with SLW set to it's extremes, they are thinner or fatter than the regular lines.

Composite On Original

Composites the lines on top of the original layer.

Tolerance

This is a very useful parameter because it allows you to quickly and easily get straight lines. It causes the HairLines to be made up of straight segments. The higher the Tolerance, the more segments are used to create the line. At 20, the line is very smooth, at 1 the lines are practically straight, or are made up of long straight lines.

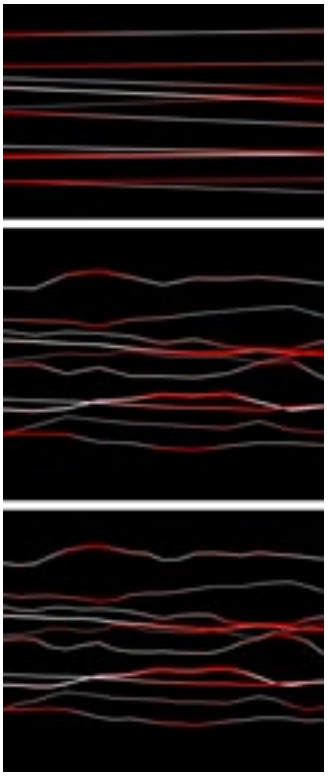
This can produce some great effects. For instance, take a look at the 'vert_lines.ffx' favorite that came in the Animation Presets.

Ripple size will affect this, even with Tolerance set to 1. With that Tolerance, the lines are usually reduced to single straight lines and the ripple causes the entire line to move. Normally, a large ripple would cause more distortion in the line, but in this case, since the resolution of the line is so low (only 1 segment), it just moves the line.

Once the Tolerance starts to get higher, particularly once it's up above 8 or so, the ripples cause the normal distortions. Of course, instead of smooth curves, the curves are angular, since they're made out of straight segments.

A good way to think about this is to imagine a circle made out of straight segments. A stop sign (an octagon) is really a low resolution circle made from 8 segments. If you increased the number of sides to 64 or 128, you'd be hard pressed to realize that it's made up of straight segments unless you looked at it closely. From even a short distance, it would appear as a smooth circle.

Such is the case here. With the Tolerance cranked up high enough, enough straight segments are produced (very small ones) to often give the appearance of a smooth line.

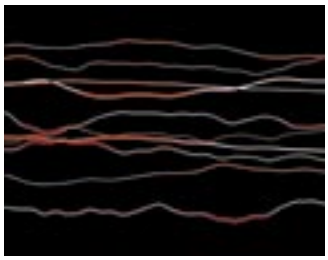


This is how Tolerance affects the lines. The top image is with Tolerance set to 1, which gives you virtually straight lines. The middle image has Tolerance set to 10, and the image at bottom is set to 20.

Notice as you go from left to right, the number of segments making up each line increases. This results in an almost smooth curve on the right.

Angular Lines

A high Tolerance doesn't guarantee a smooth line, but it comes close. However, if you really want a smooth line, turn Angular Lines off. Angular Lines, just as it sounds, makes the lines stay in straighter segments.



A bunch of normal lines, curving and flowing as is usual. Not very angular. Usually lines are affected by the ripples in a smooth way, and don't generally distort into sharp angles. The Angular Lines checkbox changes that.

Lines With Ends Section

Lines that are the length and width of the screen are great, but there's always a need for lines that are a bit smaller. You may want to simulate string cheese, sperm, or even pickup sticks (in fact, make sure to check out our Animation Presets/Favorites).

The controls in this section allow you to specify a line length and allow the lines to grow. This is quite different than the normal behavior, where the lines are simply there and get distorted by the ripples. These lines move, grow, and behave the way you tell them to. Don't you wish eerything in life was this easy?

Lines Have Ends

This checkbox activates the entire section. If this is not selected, none of the parameters in this section have any effect. The parameters in the 'Lines' section do still function as normal, whether or not Lines Have Ends is turned on.

Use Lines With Ends to create flowing lines which grow to reach their maximum Length.

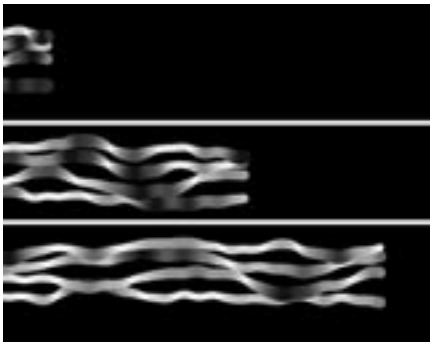
Lock Horizontal To Vertical

You'll notice that many of the parameters in this section actually have two parameters. One for the Vertical direction, and one for the horizontal direction. This parameter links all the Horizontal controls to the Vertical controls. This way you only have to change the Vertical controls to affect all the lines.

If you want the vertical lines to behave differently than the horizontal lines, leave this unchecked. The only difference will be that you'll have to move both the vertical and horizontal slider for each attribute, like length, or position.

Vertical, Horizontal Length

This sets the length of the column lines (vertical) and row lines (horizontal). This is the max length. The lines will grow into this based on the Grow speed. Starting off life as a pixel and eventually growing to whatever length you've set. You can move the lines around at any time, regardless of whether they are full length or not.



When lines have ends, you can set a Growth rate, and the lines will grow to reach the maximum length you set. The lines are still affected by the ripples while growing, so you end up with flowing, moving lines. This can produce some nice effects for hair, thread, and, uh, stringcheese.

Vertical, Horizontal Starting Point

This is where the lines start growing from. Even though these are position points, the Vertical Starting Point only works along the Y axis, and the Horizontal Starting Point only works along the X axis.

This is due to the way the lines are attached to the grid. While the lines aren't controlled by the grid, they are constrained, so you can't move Vertical lines left and right. You can only move them up and down. If you wanted to move the vertical lines left and right, you'd need to move the grid.

So… when moving the Vertical Starting Point around, keep in mind that only the Y axis value will have an effect. Whatever you enter into the X axis value will be ignored. The opposite is true for the Horizontal Starting Point. Only the X axis value matters, and the Y axis is ignored.

You can animate this, and move the lines around. However, they aren't affected by Line Wrapping (explained later).

Vertical, Horizontal Offset

This moves the lines by the amount in pixels. This provides an easy way to animate them. If Line Wrapping is turned on, the pixels will wrap to the other side of the screen when they get to an edge. For example, if the lines hit the top, and offset keeps animating, the lines will wrap around to the bottom and continue moving upwards.

Line Wrapping

Since this isn't a particle system, if you want lines to stay on the screen you'll need to use Line Wrapping. This makes sure your lines never disappear, unless you set opacity to 0 or something. This gives you a good way of getting repeating lines.

If Line Wrapping is not turned on, the lines will proceed to go off the edge of the screen as the offset is cranked up.

Vertical, Horizontal Growth Rate

These just determine how fast the lines grow to their maximum length. This is set in pixels.

There's not a whole lot to this parameter. Set up the lines up the way you want them, and set the growth rate, and let the lines grow. Obviously, the higher you set this the faster they'll grow and the faster they'll reach their maximum length.

Unfortunately there's no way to pre-roll the growth rate, so if you want the lines to start life fully grown, you'll need to move the layer backwards in time. For example, if you want them fully grown at the beginning of your timeline, you'll need to find the point that the lines are fully grown, and move that point to Time 00:00. If that point is 2:15, then you'd move the layer so that 2:15 on the layer lines up at Time 00:00.

Both angular (left image) and smooth (right) lines can have ends. You can take a look at these animating by applying the 'pickup sticks' and 'go-go-go' Animation Presets that came with the product.

Ripples Section

The ripples are at the heart of this filter. They're responsible for the look of the distortions and the way the filter behaves. Learning how to control these are essential to making the most of this filter.

Be very careful how you set these controls up. A high frequency and large size can result in glacial render times, especially if you have a lot of lines or very thick lines.

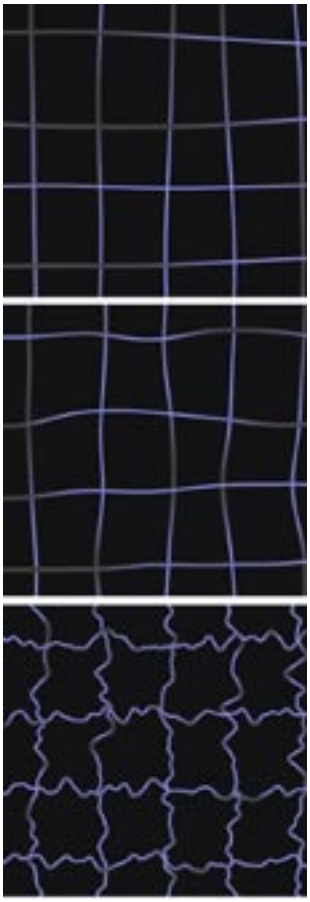
Frequency

This determines how many ripples the lines are affected by. You could also almost refer to this parameter as 'number of ripples'. That's not totally accurate, but it comes pretty close to what happens when this parameter is adjusted.

The higher this percentage, the more turbulent the line will look, due to the increase in the number of ripples. This is really just relative to the line. 0% still results in some ripples, but they're very smooth and far between. Cranking this up to 100% causes the line to be very chaotic, with many peaks and valleys, and ripples that are very close together.

If you want to create something that looks like the audio waveform of someone speaking, set this very high. If you're trying to create something that looks like a sine wave, set it relatively low.

Here we've set a low Frequency but a high Size amount. This produces undulating, liquid like distortions.



The top image has a Frequency of 10, the middle image is 60, and the bottom is set to 95. As you get close to 100, ripples really become pronounced.

Size

This is more like a magnitude control. It increases the height of the peaks of the ripples, and depths of the valleys.

The width of the ripples is controlled by the Frequency parameter. The lower the setting there results in wider ripples, since less of them are taking up the entire line. Size sets the scale along the Y axis.

An example of how Size does matter. To the left, Size is set to 10. In the midde, Size is set to 45. At right, Size set up to 100. By combining changes in Frequency, Size, and Turbulance, you can get a wide range of line distortions.

Speed

As you might guess, this controls how fast the ripples move through the line. The higher the number, the faster they move. This is essentially what causes the lines wiggle, and change shape. If you want the lines to appear to move in slow motion, set this really low, if you want to create lightning, set it very high.

Turbulance

Turbulance increases the amount of distortion in the lines. It doesn't increase the number of ripples, but it makes the ripples much more turbulent.

You'll notice little ripples on the ripples that are created by the Frequency parameter. The main ripples aren't really affected, but an extra level of distortion is added as this is increased.

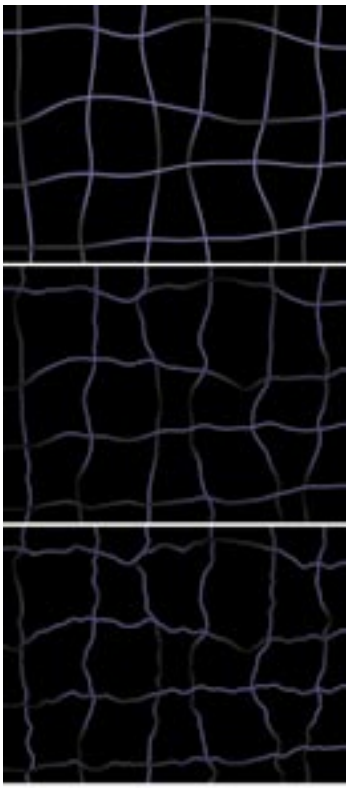
The higher you set this, the more 'sub' ripples occur on the main ripples, and the more chaotic and distorted the lines will look. If you want really smooth lines, set this to it's minimum value. Very for very chaotic, crazy looking lines, just crank this up to taste.

Affect pop-up

This pop-up controls whether the ripples affect just the horizontal lines, the vertical lines, or both.

This can be important when used with the Distortion parameter (located in the Lines section). If the ripples don't affect both Vertical and Horizontal lines, the lines that are affected will cause distortion in the lines that cross them. The amount of the distortion the crossing lines cause is determined by the Distortion parameter.

Two instances of HairLines. The bars or grid-like lines were created with a Size set to 1 and a high Frequency. The blue lines have a low Frequency and small Size.



Turbulance doesn't make for larger ripples, but it does increase the detail that occurs on the lines. You'll notice very smooth ripples in the top image. As you get to the bottom image, the ripples are much rougher.

These two parameters (Affect and Distortion) are useful when you want one set of lines to be the sole source of distortion for the other set of lines. Basically what happens is the ripples distort one set of lines (say, the vertical lines), then the vertical lines distort the horizontal lines that cross them.

Randomness Seed

This is the number that's used to generate any random numbers. If you want a slightly different look to your animation, try changing this. It will change the values around, make the ripples behave differently, and tweak any parameter that uses randomness.

Simply animating this parameter can create some neat effects. Take a look at the 'pickup_sticks.ffx' Animation Preset. Most of the animation is created by having hold keyframes every few frames, animating the Seed.

Displacement Map

This section, turns the lines into a displacement map. Allowing the lines to distort the underlying image. The three parameters act like a normal displacement filter. The Angle sets the direction that the original pixels are pushed around. The Amount sets how much they should be displaced by, and Soften, simply blurs the lines and eliminates the hard edge.