

Table of Contents

Click on any item below to view its page.

We recommend that all users print out the Concepts and Tutorials chapters and perform the provided tutorials.

[Introduction](#)

[Requirements](#)

[What's New](#)

[Overview](#)

[Concepts](#)

Tutorials

[QuickStart](#)

[QuickStart II](#)

[QuickStart III](#)

[Getting Deeper](#)

Reference

[User Interface](#)

[Color Shaders](#)

[Using Pictures and Movies](#)

[Animation](#)

[Compiled Trees & Recursion](#)

Component Reference

[Component Reference](#)

[1-in/1-out components](#)

[2-in/1-out components](#)

[2-in/2-out components](#)

[2-in/3-out components](#)

[3-in/1-out components](#)

[3-in/2-out components](#)

[3-in/3-out components](#)

[4-in/2-out components](#)

[4-in/3-out components](#)

Appendices

[QuickTime Notes](#)

[New in 2.0](#)

[Keyboard Shortcuts](#)

[Troubleshooting](#)

[F.A.Q.](#)

[Getting Help](#)

ArtMatic Pro User Guide

prepared by

[Edward Spiegel](#)

with thanks to Eric Wenger, Andy Leffler and Peter Miller.

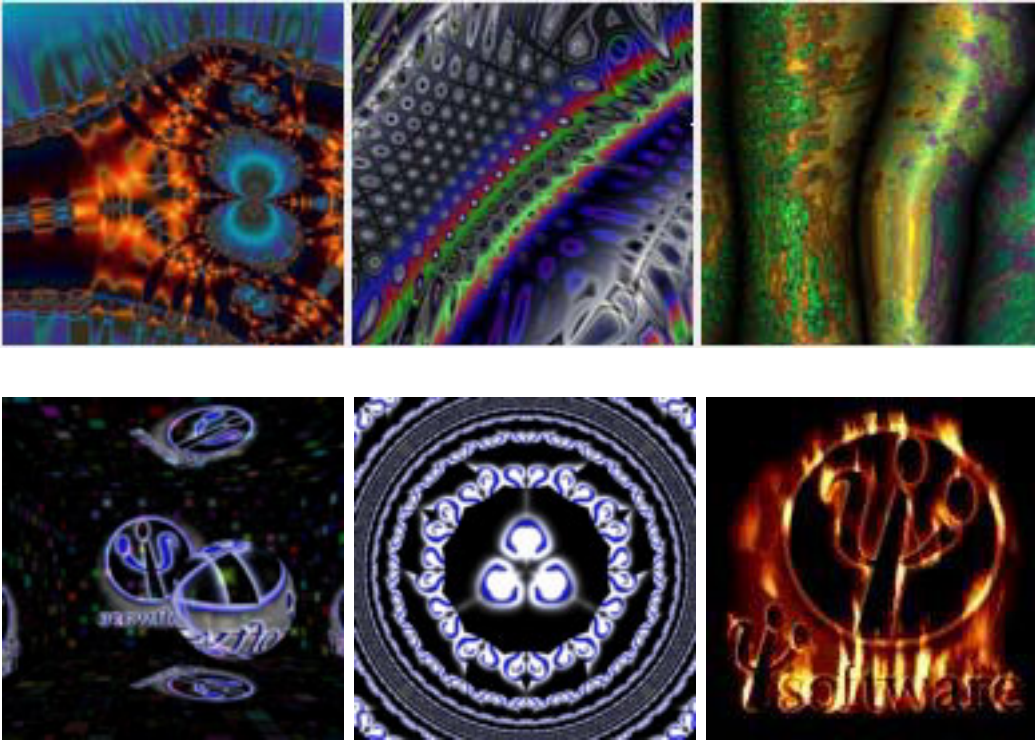
ArtMatic and MetaSynth are trademarks of U&I Software, LLC.

All rights reserved.

Bryce is a trademark of Corel.

© 2001, U & I Software

Welcome to ArtMatic !



ArtMatic is a unique kind of program: a modular, patchable graphics, animation and audio synthesizer. Its modular design and rich component set allow you to create stunning images, animation, video effects and sound. The applications for this tool are endless. Imagine creating fractal art, 3D scenes, and stunning video effects with the same tool! Anyone can create breathtakingly vibrant images, psychedelic animation, and exciting new sounds. Demanding pros and amateurs alike will find countless uses for this exciting application.

ArtMatic can create:

- **Pictures** - an incredible number of stunning pictures can be created during even a short session of a few minutes. Be sure to have enough disk space!
- **QuickTime Movies** - ArtMatic features several animation modes to render high resolution QuickTime movies.
- **Sounds** - Strange, exciting sounds can be automatically generated from ArtMatic's mathematical "structures". Don't worry, no mathematical skills required.

Here are just a few uses for ArtMatic:

- Create digital fine art,
- Create animation sequences for music and art video,
- Create unique video effects for processing QuickTime and DV movies,
- Create CD cover and label art,
- Create background images for web pages and brochures
- and much, much more.

How does it work?

While complex mathematical principles and sophisticated graphic functions underlie ArtMatic, you don't need to be a math whiz or experienced graphic designer to achieve amazing results with ArtMatic. Follow the simple steps described in the [QuickStart](#), and you will be creating ArtMatic images and animations in minutes. To get started, all you have to do is choose a structure from the **Structures** pop-up menu and click on the [Randomize All](#) button (the largest of the three dice) to explore a new picture-generating system.

If you want to torture your brain, you can think of each system as an Nth-dimensional universe where the image you see is a two-dimensional view from a particular point. Changing parameter values moves you to different locations in this rich universe. You will discover that every picture-generating system can create a vast array of striking images.

Learning to use ArtMatic

ArtMatic is a rich environment which can be explored in a number of ways. It can be used in a 'Zen' way in which you roll the dice and mutate to explore, or it can be used in a more directed fashion. We have provided a series of tutorials for those that want to master this amazing tool. We have also provided a wide range of example files that you can use as starting points for your exploration. **Be sure to explore** as many examples as you can--the range of images ArtMatic can create is truly astonishing -- so astonishing that it often surprises its own creators!

System Requirements

CPU: Macintosh PowerPC with 32 megabytes or more of memory.

System: 8.0 or higher with QuickTime 3.0 or higher.

Minimum Monitor Size: 800 * 600.

Monitor color depth: 32 (16 millions)

IMPORTANT! *Virtual memory can interfere with the sound features.* If you experience erratic behavior, open the Memory control panel and make sure that virtual memory is turned off. It should be noted that some OS upgrades turn this feature on automatically without informing the user.

Memory tip: ArtMatic can run in a memory partition as small as 12 megabytes. ArtMatic 2.0 has improved its memory handling when saving pictures. The application no longer needs a large memory partition to save large files. You may need to increase the memory partition if you are creating systems that use nested (compiled) trees or movie/picture inputs.

Performance note: ArtMatic will run on any Power Macintosh. With a slow processor (less than a 300 Mhz 604 chip) real-time animation and sound will be choppy and complex systems may be slow to render/display.

What's New in ArtMatic Pro 2.5

About This Chapter

This document provides a quick summary of the new features for users already familiar with ArtMatic 1.2 and ArtMatic Pro 2.0. ArtMatic Pro 2.5 introduces major changes to ArtMatic and ArtMatic Pro's architecture. Now, it is not only an extraordinary art synthesis tool but also a terrific image and video manipulation tool capable of creating spectacular image and video special effects using either ArtMatic's traditional gradient-based synthesized color or **true color**!

ArtMatic Pro is no longer just for the 'zen-mode' creation of images. Its new features open up astonishing capabilities for image design, and the new [Getting Deeper](#) advanced tutorials will help artists master this amazing new tool.

"Zen-mode explorers" will find that ArtMatic Pro can create an even wider range of images than before, and those users that seek greater control in manipulating and creating images and animation will find spectacularly expanded control and flexibility as well as a wealth of new primitives that include true 3D and true color (RGB) manipulation.

Here is a quick overview of the major changes introduced since ArtMatic Pro 2.0:

- ArtMatic structures now have a time-based third global input which systems can tap
- **True color** RGB color mode which allows for native color processing of color images and QuickTime movies,
- Four input pictures/movies per file,
- High quality animation mini-preview of animation in the main window (shortcut: command-h),
- New **compilation** feature permits a complete ArtMatic structure to be exported and used as a single component in other ArtMatic files,
- Recursion and iteration features permit the design of new fractals,
- True 3D components and textures for generating and manipulating 3D objects and solids,
- Expanded and improved shading model
- Many new components and component types,
- A host of new components for RGB and HLS manipulation,
- More flexible tree manipulation and construction functions
- **Connection dialog** for custom connections between components
- Parameter/function locking to make locked parameters immune from changes from the Mutations dialog or clicks on the small die.
- Pict/Movie popup for selecting and changing input movies and pictures
- [Input Movie Setup](#) command in the **Animation** menu
- Re-organized and upgraded [Preferences](#), [Edit Camera Path](#) and [Parameter Envelopes](#) dialogs
- Direct numeric entry of parameter values (via the [Parameter Envelopes](#) dialog)

ArtMatic 1.2 users may also want to read the appendix [Added In 2.0](#) for a summary of features found in ArtMatic Pro 2.0 that were not found in earlier versions of ArtMatic.

Important Compatibility Note for Users of ArtMatic Pro 2.0

We have tried to make ArtMatic Pro 2.5 compatible with files created in earlier versions. However, due to the way the complex, dynamical systems work and due to bug fixes in some components, some files will not appear the same in version 2.5 as in earlier versions.

It is strongly recommended that you make backup copies of all files created with older versions in case one or more of your files renders differently with version 2.5. We also recommend that you keep a copy of the ArtMatic version used to create those files in case you need to re-render them. In particular, systems that use the **sin x + sin y** or **sin x * sin y** 2D-scalar (2-in/1-out) components will be different. If the files have keyframes, the keyframes will no longer be as created and may need to be re-created from scratch.

Finding input pictures/movies. ArtMatic 2.5 has a new method for finding movie and picture files referenced by ArtMatic files. When ArtMatic can't find a movie or picture referenced by a file being opened, a message will appear in the **Tool Tips** area while it searches for the file. Pressing the Escape key will cause ArtMatic to give up looking for the file.

New Getting Deeper Tutorials

A new chapter has been added to the documentation which provides lessons and information that will be of interest to any user that wants to learn to control and manipulate ArtMatic Pro. [Click here to read the new chapter.](#)

NEW FEATURES

The rest of this chapter covers the most significant new additions to ArtMatic Pro 2.5. In addition to the changes covered here in detail, there are a wealth of new components.

Third Global Input (time/z-dimension/counter)

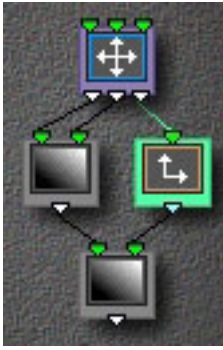
In earlier versions of ArtMatic, only two values could be fed into the top of the structure tree. Version 2.5 introduces a third global input which is fed to all systems that have a three-input tile at the top of the tree **or** a tile with an open third input in the interior of the tree. This third input (called the third global input or time) is a value which is steadily incremented over the course of an animation.

There are many ways to use and interpret the value which is actually a simple counter that starts at 0 and is incremented over the course of an animation. It can be viewed as a time input since its value increases steadily with the passage of time and provides a simple means of modulating the system over time. It can be used as a control source for those components that use the final input as a control source (**z Wipe** or **Packed RGB z Sort**, for example). It can also be used as a third spatial co-ordinate which increments over time. When used in this last way, you can think of the canvas as being a 2D slice of a three-dimensional space. With each incrementing of the third global input, the canvas is moved along the third dimension so that each frame is a slice from another point along the z-axis.

The third input's value is 0 at the beginning of an animation and a fixed value at the end that is the same in all systems regardless of the elapsed time. The value increases steadily between 0 and the fixed value over the course of the animation. To modify the counter's normal behavior, insert a 1D scalar (1-in/1-out) component in the tree to scale or alter the values. In this way, time can be made to cycle or accelerate or even change at random.

This new input is very handy for controlling an animation. Previously, when using keyframe animation, changes during an animation only occurred if the values of some parameter sliders were different in different keyframes. Now, time can be used to manipulate a system even if all its parameters stay constant.

A simple example. In the folder "Doc. Example Files" is a file called "Simple Time Explorer" which contains a simple system with 3-inputs at the top.



The topmost component simply scales the three incoming values (x co-ordinate, y co-ordinate, and time). The amount of scaling is provided by the A, B and C parameter sliders. For this example, no scaling is done. The x and y co-ordinates are passed unchanged to the $Ax+By+C$ component which simply generates a tilted plane. In this case, the parameters have been sent to send out 0 for all points (essentially a plane with no tilt). The last component is another $Ax+By+C$ component. There is one other component in the tree. It is the scale component which is connected to the rightmost (the z/time output) of the topmost component and whose output feeds the 'y input' of the final component in the system.

Click on the file's keyframe and click the **Add** button which creates a second identical keyframe. Now click the **Animate Keyframes** button. Note that the result is a gradual color change over time. In previous, versions of ArtMatic, animating between identical keyframes would also yield a static result. You can simulate ArtMatic's old behavior, by clicking on the scale component's tile and clicking Parameter A's lock icon then dragging the slider to 0. This causes the z-value to be a constant 0. If you now animate the keyframes, nothing will appear to happen. Notice that you can change the response to time by changing the 1-in/1-out component. For example, you can use the **Random** component to randomize the color change.



Important notes about 'time'. This new input has a few side effects which may seem surprising. **First**, it is important to keep in mind that the third global input is fed into open inputs that are in the middle of the tree (as in the picture at left); hence, time can influence trees that have only two inputs at the top. **Second**, the counter's value increments with the passage of time **even if it flows through locked tiles**. (Parameter locking only prevents a tile's parameters from changing. It has no influence on the values that flow through the component.) **Third**, when you add a keyframe in a system that makes use of the third global input, the image may change when the keyframe is added. This happens because, while you are editing, the third input has the time value of the most recently viewed keyframe (or the value it had when animation was stopped). ArtMatic is only able to calculate the correct time value when the keyframe is actually added.

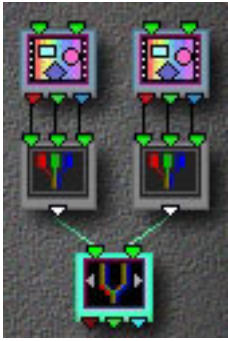
Examples. Many of the example files we have provided make use of this time input. Several examples are found in the folder **Doc. Example Files:Time Examples**. You can also search the main example library for "time" to find more examples.

True Color (RGB/HLS) Color Handling

In all earlier versions of ArtMatic, the final output color was strictly determined by the active gradient (palette) and the shading algorithm. As a result, input pictures and movies were always rendered with the colors of the gradient. In the new version, ArtMatic adds true color RGB color output which can directly generate 32-bit color output and which bypasses the file's gradient and shading algorithm. When the tree's final component has three outputs, the output values are not mapped to a color in the active gradient. Rather, the outputs are treated as the RGB (red, green, blue) values of the final image. This makes possible the creation of a wide range of images not possible in earlier versions. The most obvious impact is that it is now possible to process color pictures and movies while preserving their original colors. It is even possible to do advanced color processing and manipulation of these images to create amazing new special effects. ArtMatic even has new components that provide HLS and gamma processing.

Gradients & RGB color. Gradient-based shading is still used in all systems that terminate with a single output value (or a two-output tile--which is actually treated as two parallel one-output tiles). It also available in RGB-based systems through the use of new components which can either mix gradient-based and RGB branches of a tree or convert gradient-based branch to RGB. If the final component is not a three output component, the structure tree's output is fed into the shading component which determines how the output values are mapped to the colors of the selected gradient. If the final component has three outputs then the **RGB Color Model** is used.

New color components. Several new **movie/pict** components have been added to take advantage of this new color model. In addition to the 2D-scalar **movie/pict** component introduced in ArtMatic Pro, there are now 2-in/3-out **movie/pict** components for using the movie/picture's true colors.



A number of new components have been added for the manipulation true color images--including HLS (hue, luminance, saturation) manipulations. Because ArtMatic components are limited to four inputs per component, a new **Pack** component has been added which combines three input values into a single special value which can be passed to the special components that know about packed inputs. **Pack** makes a **packed component** with two inputs equivalent to a component with six inputs, which is necessary for components that mix two or three color pictures or movies. In the example structure shown at left, there are two identical parallel branches whose first component is the **color pict/movie** component which is followed by the **pack** component. The system is terminated by the **Packed RGB Crossfade** which takes two packed inputs and acts as a mixer for the two input images. Only components that have the word "packed" in their title know about packed inputs.

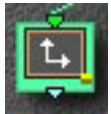
A number of new three output components (with two, three or four inputs) have been added for color processing, mixing and texture generation. Be sure to check out the example files to see the wide range of images and effects which are now possible.

New Shading Model and Shading Options Pop-up Menu



A number of refinements have been made to ArtMatic's original shading model (which is now called gradient-based coloring or shading) to provide additional control of the "fog" and "shadow" effects which previously were automatically assigned and only available to some shading algorithms. The names of some of the advanced shaders have been changed to reflect a refinement of the advanced shading algorithms. The **fog** and **shadow** effects that were available in only some algorithms are now universally available through the new **shading options** pop-up menu that is available below the main **shader pop-up menu**. These options (called **depth cueing** and **global shading**) are also available to RGB-based systems.

The advanced shading algorithms turn these new options on and off rather than having them built directly into the shader. This new interface makes fine control of these effects possible as well as making the effects available to all systems.



Component marked with depth cueing glyph

Depth cueing. Depth cueing simulates the color-filtering effect that the atmosphere has on the perception of distant objects. Depth cueing works by assigning a color (the first auxiliary color) to be associated with depth/distance. This color is superimposed on the image in relationship to the depth /distance. As the distance increases, so does the brightness and opacity of the depth color. ArtMatic now provides direct manipulation of depth cueing which can be used for both distance effects and purely decorative effects. Previously, there were several algorithms whose names contained the word **fog**. To be consistent with the new **shading options**, the shading algorithm names have been changed to use the more-technically-correct **depth cueing** which refers to a technique that simulates depth/distance effects.

Depth cueing color note. The depth cueing color (the first auxiliary color) is now always visible/available even if Depth Cueing is turned off. This color is used in RGB-based systems wherever the value **infinity** is encountered. Infinity is used for the areas outside the boundaries of 3D objects and where the **infinity gate** component is used.



Component marked with global shading glyph

Global shading (shadows). Another new shading option is **Global Shading** (or shadowing) that can be turned on via the options menu. Global shading creates shadows and light in the image by modulating the luminance (brightness) of the image's pixels. Where the shading component generates low values, the image will be dark (shadowed), and where it generates large values, the image will be brighter. As with depth cueing, you can now choose the component that provides the shadows (global shading). As in previous versions, the shaders with the words "global shade" make use of this technique. The shadow computation is no longer built into the shaders themselves. Rather, the algorithms with "global shade" in their names, simply turn the **Global Shading** option on in the **shading options** menu.

Controlling depth cueing and global shading. Previously, these effects were built directly into the shading algorithms, and ArtMatic

automatically picked the components it used to provide the depth cueing and global shading. Now, these effects are turned on and off via the **shading options** pop-up, and you can override the automatic selection and choose the components used to provide the effects. It is even possible to have branches of your tree whose only purpose is to provide depth and shadow effects. These options are available regardless of the active shading algorithm. To **turn on depth cueing or global shading**, pop up the options menu and choose a depth cueing or global shading option. To turn the options off, select either **Depth Cueing Off** or **Global Shading Off**.

Automatic or manual control. By default, ArtMatic selects the components that control the depth cueing and global shading (according to the rules described in the [Shaders](#) chapter of the manual). You can override the automatic selection by choosing **Component Sets Depth** or **Component Sets Shade** which causes the selected tile to be used to provide the effect. A small shaded glyph (as shown in the illustrations above) is used to indicate components that have been manually assigned. **Restoring automatic shade/depth control.** If you have used the **Component Sets Depth** or **Component Sets Shade** commands and want ArtMatic to restore its automatic depth/shading choice, select the **Fog and Shade Automatic** item from the **shading options** popup menu. Note that **Fog and Shade Automatic** may have a check mark even if depth and shading are turned off. The check mark merely indicates that if shading/depth are turned on that ArtMatic will choose the components automatically.

Shortcuts. Two keyboard shortcuts have been added to facilitate using these new features. Type **s** (not command-s) to turn on global shading and assign the selected component for use as the shading component. Type **d** to turn on depth cueing and assign the selected component the depth cueing function.


Tutorials. These features are explored in the [Shading Tutorial](#) found in the [QuickStart 3](#) chapter of this manual.

Mini-Preview

Type command-h (or choose **Preview** from the **Animation** menu) to see a small high-quality preview of the animation. Preview plays the animation using the last animation method (keyframe or random path) viewed. To change the animation type that is used by preview, press the **Animate Keyframes** or **Random Path Animation** button.

Multiple Input Picts/Movies Per File

ArtMatic Pro now allows each file to reference up to four input pictures or movies which makes possible an astounding range of special effects. ArtMatic can now be used to directly perform complex transition effects as well as to combine ArtMatic-generated images with existing pictures and movies. The range of possible effects is overwhelming. We recommend taking a look at the provided example files to get an idea of just some of the amazing things that you can now do. Keep in mind that anywhere that a picture is used as an input, a movie or DV stream can also be used. If you have video captures or QuickTime movies, we recommend that you explore the exciting effects that can be achieved with ArtMatic Pro by using your movies as input sources in any of the example systems we have provided.

We have modified the user interface for selecting input sources and assigning them within the system. A new **input source popup menu** () has been added to select and change the pictures in use by the system. Click and hold the mouse button to popup the menu. **To add a picture or movie as an input source**, select any empty slot from the new input source popup menu. **To replace a picture or movie** already in use by the ArtMatic system, hold down the option key, pop up the menu, and select any slot that is in use. A dialog box will appear that permits you to choose the picture to use in place of the existing picture. **To assign a picture to a particular tile**, select any tile that uses a movie/pict component then select the input source from the popup menu.

Parameter & Function Locking

Parameter locking. It is possible to lock parameters to prevent them from being changed by any operation that normally changes parameter values. Parameter changing operations include: random path animation, the Mutations dialog, keyframe animation, and the randomize parameters die (the lefthand die).

To lock all tiles, shift-click any unlocked parameter. **To unlock all tiles**, shift-click any locked parameter. This makes it possible to use the **Mutations** dialog to explore variations of only a few parameters.

Function locking. It is possible to lock the function assigned to a tile so that it is immune to operations that mutate function assignments (such as when you roll the big dice or use the **Mutations** window with the "mutate functions" option). To lock a tile so that its function will not be changed, lock all three parameter locks when the tile is selected. Note that even if the component has no parameters, all three locks must be locked to lock the tile.

Locking parameters and functions is a great way to explore ArtMatic systems. Parameter/function locking makes it possible to use mutations and the large die to explore subtle refinements of systems. For example, you might have a system that uses a few tiles to provide the texture for a surface. You could lock all of the parameters and functions of the system except for the tiles that provide the texture and then use the big die or the **Mutations** window to discover new textures created by function mutation that affects only the few components that provide the texture.

A note about parameter locking and keyframes. When parameters are locked, keyframe animation uses the last values assigned to the locked parameters when animating the system. As a result, any parameter changes stored between keyframes are ignored while the parameters are locked. The parameter changes are not lost however, just ignored. When the parameters are unlocked, any parameter changes stored in the keyframes will be honored during animation.

Compiled Trees

One of the most powerful features added in ArtMatic Pro 2.5 is the ability to export a structure tree as a compiled tree that can be used as a component in another ArtMatic file. This makes it possible to create extremely powerful building blocks of your own and to create extraordinary new images never before possible. Advanced users will find that they can even design their own fractal algorithms. Be forewarned, the use of compiled trees allows you to create structures that take a **very very** long time to calculate--especially on slow machines. We have supplied a large number of example files which make use of compiled trees and recommend that intermediate and advanced users explore them in some detail. The new [Compiled Trees & Iteration](#) chapter covers compiled trees and iteration in detail and provides a tutorial on the construction of iterative trees.

PLEASE NOTE! When exploring our example files, you should probably not adjust the **Recursion** or **Iteration** parameters of compiled trees. Each recursion level requires significant additional processing. Increasing the amount of recursion can completely tie up even a fairly power computer in some cases. So, we recommend that you leave these parameters alone unless you are adventurous and patient and have a good understanding of how recursion and compiled trees work.

How compiled trees work. The **File** menu has a new command called **Export Compiled**. When a structure is exported as a compiled tree, the structure tree and the file's keyframes are saved as a compiled tree. A compiled tree **does not** contain any color information (i.e. the gradient and shading algorithm are not part of the compiled tree). This tree can now be used as a component in another ArtMatic file. To use compiled trees, simply click on a selected tile to pop up the component popup menu and choose the **Open Compiled Tree** component. A dialog box will appear that allows you to select the available **relevant** compiled trees. A relevant tree is one that has the same number of inputs and outputs as the selected component. For example, if you export a tree that has two inlets at the top and two outlets at the bottom, it can only be used within tiles that have two inputs and two outputs. Note that compiled trees do not include camera path information, and, as a result, any movement performed via camera path animation is not stored in the compiled tree.

Keyframes and compiled trees. When compiled trees are used as components, their keyframes are imported and blended with those in the parent tree. This is a very powerful mechanism that allows you to build compiled trees that act as animation primitives or macros. If the parent tree has fewer keyframes than the subtree, new keyframes are added to the parent tree and adjusted so that the motion programmed in the subtree's keyframes is preserved.

Editing/viewing embedded compiled trees. When using a compiled tree as a component, the compiled tree is copied into the parent

and becomes independent of its original source. The subtree can be viewed and further modified from within the parent. Typing 'e' when a compiled tree component is selected allows you to edit and view the subtree. There are some restrictions when editing subtrees. A subtree's basic structure cannot be changed though you may change the components assigned to the subtree's tiles. To prevent subtree structure changes, all structure editing tools are disabled while editing a subtree. Type 'e' again to leave subtree editing mode.

Re-exporting compiled trees. While editing a subtree (as described above), it is possible to re-export the tree by choosing **Export Compiled**. Only the subtree is exported when in subtree editing mode.

Importing compiled trees. It is possible to create a new ArtMatic system derived from a compiled tree. To do this, choose **Import Compiled** from the file menu. The file's structure and keyframes will now match those of the compiled tree and replace whatever structure and keyframes were present prior to choosing the command. Note that compiled trees do not contain camera path information.

Recursion. While compiled trees are not re-entrant (a tree can't be compiled if it contains a compiled tree), many of the compiled tree components have a recursion parameter which feeds the components output back into the input a specified number of times. Recursion makes possible the creation of new fractal and fractal-like components. See the examples provided with ArtMatic Pro. By default, the recursion parameter should be set to 0 since recursion only makes sense with structures intended to be recursive.

Note: To avoid systems so complex they could bring even powerful systems to a halt, it is not possible to export a structure as a compiled tree if the structure already contains a compiled tree.

Tutorials. Many of these features are explored in the [Compiled Trees Tutorial](#) found in the [QuickStart 3](#) chapter of this manual.

Expanded Tree Editing Capabilities

ArtMatic Pro 2.5 provides considerably more flexibility when editing the structure trees than did ArtMatic Pro 2.0. Previously, ArtMatic did not like inconsistent (incomplete) trees (i.e. trees with unconnected inputs and outputs in the middle of the tree) and placed limitations on the editing operations you could perform.


While there are still some limitations, most have been removed since it is sometimes necessary to temporarily leave a tree in an inconsistent state. Removal of the limitations has made it possible to accidentally leave a tree in an inconsistent state which you should avoid. While incomplete trees are not forbidden, they may behave surprisingly when their parameters are mutated or animated or when used as compiled trees. So, it is a good idea to make sure that trees are complete except for those cases where the incompleteness is intentional.

In addition to the new tools described below, a few new commands have been added to the [Insert](#) menu. These commands are fairly self-explanatory and covered in detail in the [User Interface](#) chapter of the manual.

Troubleshooting note. If your tree behaves strangely, the first thing to do is to check for unconnected inputs in the middle of the tree. While unconnected inputs are sometimes allowable, they can lead to surprising results and should be avoided in most cases. The primary exception to this rule is that it is sometimes desirable to have an unconnected third input to a three-input function which allows time to be accessed directly from the inside of the tree. Techniques for connecting unconnected inputs are covered later in this chapter.

Structures Area



A new **Structures** area has been added to the right of the **canvas** to provide convenient access to the most commonly used editing commands (some of which correspond to commands in the **Insert** popup menu and some of which correspond to commands in the new **Replace** popup menu ). As with all of ArtMatic's tools, you can find a tool's name by mousing over the tool and observing the **Tool Tips** area at the lower-right of ArtMatic's window. Unless otherwise noted, these tools are activated by a single click.

The Tools



Delete selected - Delete the selected tile.



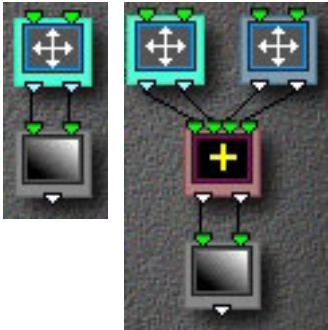
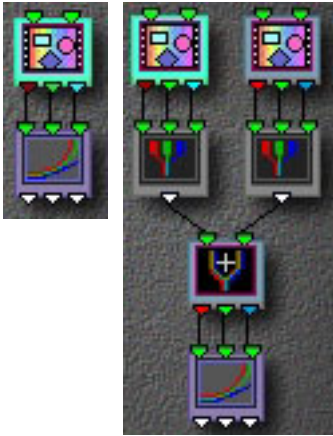
Insert After - Add a new tile after the selected one. The new tile will have as many inlets as the selected tile has outlets. The new tile will have the number of outlets necessary to connect it to the tile which follows it.



Insert Before - Add a new tile before the selected one.



Make group - Replace the selected component with a group that consists of a new identical component and another new component to mix the outputs of the original and the added components. If the selected component has three outputs then a pack component is added to each branch of the group as well as illustrated below.

	
A system before and after applying Make Group to a 2D vector tile.	A system before and after applying Make Group to a 3-in/3-out tile



Add branch -Add a new tile which branches off the selected tile. This command forks a branch at the selected tile.



Complete Tree - This handy tool adds the components necessary to mix any parallel branches whose outputs are open. An open output is one that is not connected to the input of any tile. Normally, a tree should have only one component with an open output. This tool saves a number of steps when creating structures that have parallel branches that you want to contribute to the final output. Tree

completion is discussed in greater detail [below](#) and in the user interface chapter.



Replace popup - Click and hold the mouse button to pop up a list of commands that can be used to replace the selected tile.

Disconnecting tiles. It is now possible to disconnect a component (and those that follow it) from the tree by using the **Insert** pop-up menu's **Disconnect** command. This is a handy way to break a long branch into two parallel branches.

Editing restrictions. There are a couple of editing limitations worth noting. There is a limit to the number of columns (four) and the number of rows (ten) that an ArtMatic structure can have. Compiled trees within the tree can be effectively used to work around this limitation.

Branch order "instability". While editing the tree structure, ArtMatic may occasionally switch the order of the branches. You should watch out for this behavior when deleting tiles from a branch. You can swap the position of two parallel branches (that are not connected at the bottom) by typing 'm' (for main component). Note that this only works if the branches are open at the bottom. If the two branches are joined at the bottom, delete the component that joins them before typing **m**.

Tree Completion

Because ArtMatic uses the output of a single component (the first component of the last row) to create an image, trees should be **complete** (i.e. have only one component with an unconnected output) except for cases where an unconnected output is used for depth cueing or global shading. Because you will often find yourself needing to connect parallel branches while editing tree structures (especially if you are creating a system with several pictures or 3D objects), the **complete tree** tool has been provided to automatically complete incomplete trees.

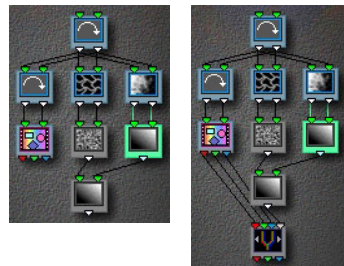
When the **complete tree** tool is clicked, ArtMatic will mix the branches of the tree by adding and connecting the appropriate components. For example, to mix **two parallel RGB** branches, ArtMatic will add any required **pack** components then mix the branches with a packed RGB mixer. **Mixed systems** that feature an RGB branch and a 1D gradient-based branch will be mixed with the appropriate 4-in/3-out mixing component. **Multiple 1D branches** will be mixed with a two or three to one mixer. In rare cases, ArtMatic may not be able to determine how to mix the system, in which case a beep will sound when the **complete tree** tool is pressed. If this happens, you will have to add some tiles yourself to complete the tree. If there is an RGB branch and a two output branch to mix, for instance, a **2D Scalar** (2-in/1-out) component should be added at the end of the two output branch. The table below shows a few examples of structures before and after application of this tool. You can often save yourself a lot of work by using this tool.

Two RGB Branches



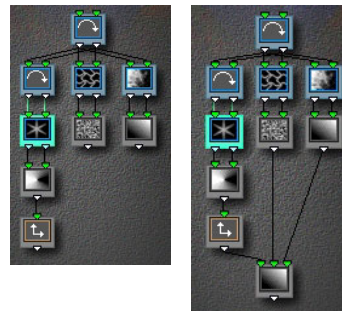
To combine two RGB branches, Artmatic adds the necessary pack components as well as the mixer.

RGB Branch and 1D Branch



ArtMatic mixes an RGB and a 1D branch with the 4-in/3-out version of **RGB Interpolate**.

Three 1D Branches

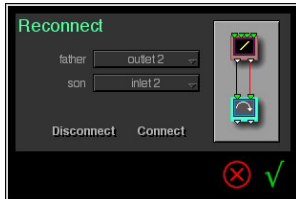


Three 1D branches are mixed with a 3D scalar component.

Connection Dialog

Re-connecting loose inputs and joining branches. There are a number of common editing operations which can leave tiles with unintentionally unconnected inputs, and there may be situations where you want to change the parent of a particular tile. There are two ways to change the connections between a child tile and a parent tile found higher on the tree. Automatic connection (**command-click**) forces an automatic connection from a child to a parent tile. The new connection dialog (**option-command-click**) allows for custom manual connection of tile inputs and outputs. Both methods require that the parent tile be higher on tree than the child tile and can be used to either connect loose or open inputs or to change a tile's parent.

Automatic connection. Automatic connection is the quickest way to change a tile's parent or force connection of unconnected inputs. To force an automatic connection, select a child tile then **command-click** a tile higher in the tree that will become the tile's parent. Be aware that choosing a parent tile from another branch will sometimes cause ArtMatic to re-arrange the tree's layout. **Undo** can be used to undo this automatic re-patching if the results are undesirable. ArtMatic does its best to determine whether you are trying to completely change the parentage of the child, or whether you are simply trying to connect and open input. In some cases, ArtMatic will not be able to determine an appropriate automatic connection. In such cases, use the **Connection Dialog** described below.



Connection Dialog. For greater control of child/parent connections, use the **connection dialog**. It allows you to create (or break) connections between any inputs and outputs of child/parent tiles and even to split parentage between tiles. To invoke it, click on the child tile and **option-command-click** any tile higher on the tree. The dialog displays the parent and child tiles and provides two ways of establishing connections.

1. Direct editing. Click on a parent output then click on the child's input to which to connect it.
2. Via the father/son pop-up menus. Choose the father's output, choose the son's input then press either the **connect** or the **disconnect** buttons.

An exercise: Use the connection dialog to horizontally flip an input image. **Hint:** start with the **RGB 1 Channel** structure (available from the structures popup) and use the connection dialog to re-arrange the connections between the first and second components in the tree.

Editing Tips

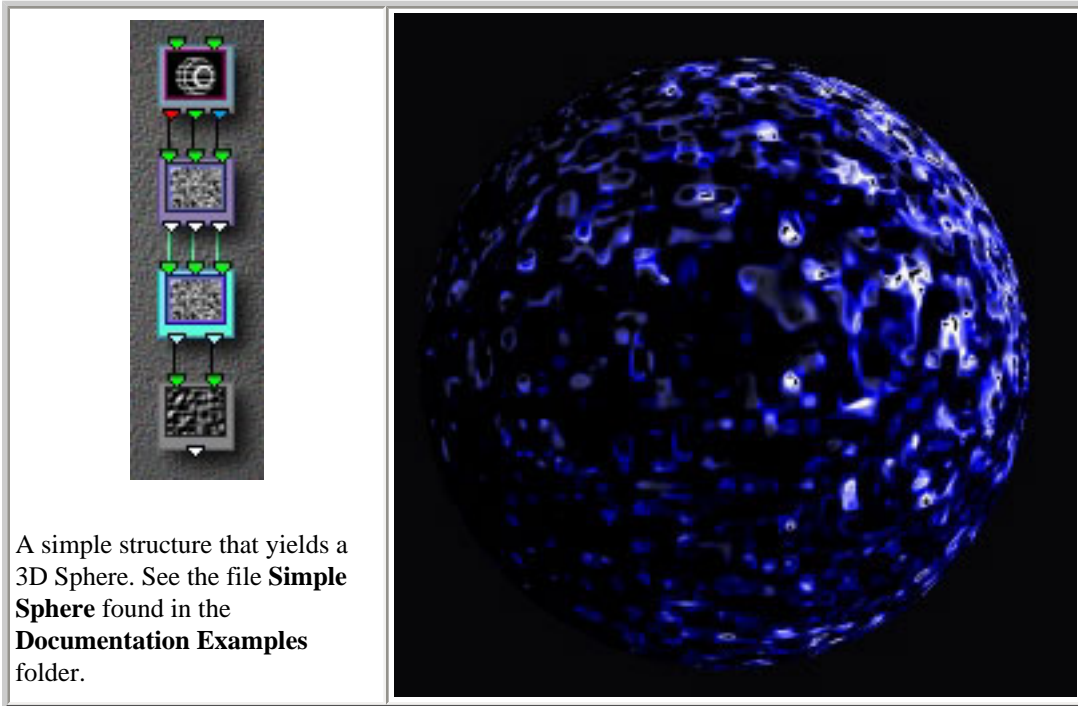
Joining branches at the top. If a tree has parallel branches you would like to connect at the top, simply choose the Insert menu's **Insert Global Rotation**. ArtMatic inserts a rotation tile at the top of the system to which both branches are connected. You can change the component to something other than rotation if you desire.

Adding filters. When building complex trees or experimenting with mutations or random path animation, you may find it helpful to insert filter components to restrict the range or modify the values being fed into some but not all of a component's inputs (or outputs). New editing commands have been added to make this possible. See the new [Inserting Filters](#) tutorial lesson to learn this powerful technique.

'True' 3D Objects

ArtMatic Pro 2.5 provides a number of components that generate 'real' 3D solids. The Examples folder contains some great examples of 3D objects which we recommend that you examine and dissect. These components (which include **3D Tube**, **3D Cube**, and **3D Sphere**) have either 2 or 3 inputs and 3 outputs. The three outputs for these systems provide the x, y and z co-ordinates of the solid being generated. The inputs that precede the 3D component define the space within which it is created (any warping of the space that

occurs before reaching the 3D tile will result in a warping of the generated solid--and any preceding **Scale** or **Rotation** components will rotate or scale the solid). The components which follow the 3D component define the solid's surface as in the example below. Because 3D object components define untextured space rather than a surface, they will generally be followed (at some point in the system) by a component or components to shade or texture the surface.



A simple exploration of 3D surfaces

In this section, we will explore some of the properties of ArtMatic 3D objects and some techniques to use with them. The image shown above was produced by the file "Simple Sphere" which is found in the **Documentation Examples** folder that accompanies this manual. Open the **Simple Sphere** example file. It simply has a **Sphere** component followed by several noise components which eventually reduce the three outputs to one so that ArtMatic can use the current gradient to color the surface. These components provide the image surface.

Click on the **Sphere** component's tile. Adjust its three parameter sliders and notice how it moves in the space in relation to the texture which covers it. When the sphere moves, it doesn't take the texture with it but moves within the texture's plane. It is as if you have a silk cloth with a textured pattern stretched on a frame before you and a small sphere in your hand. If you take the sphere, reach around the cloth and press the sphere so that its outline is visible through the cloth and slide the sphere, the sphere will be moving under the texture rather than taking it with it. Most of ArtMatic's 3D objects have this relationship to their shading/texture. There are a few 3D objects (whose names include the word **parametric**) which behave differently and take their textures with them.

The texture can also be animated. Click on any of the noise components and mouse over the parameter sliders to find the component's phase parameter. Adjust the fader and repeat for each of the noise components. Click on the **Add** button to add a keyframe. Continue adjusting the phases and adding keyframes. Now watch the animation by pressing the **Animate Keyframes** button. To see a higher quality (but smaller) preview of the animation, type **command-h**. Observe how the surface moves though the sphere stays in place. With some experimentation, you will find that you can simulate the rotation of a planet, simply by adjusting the phase parameters of the noise components.

Infinity and backgrounds

3D components generate the value **infinity** for areas outside of the solid (for such objects as spheres and cubes which have a well-defined outside). For simple systems such as that found in **Simple Sphere**, you can change the background against which the solid appears by setting the right-hand colors of the gradient in use. In more complex systems with multiple branches it is possible to have

the solid appear against a background provided by a picture, movie, or an ArtMatic subtree. The Documentation Examples folder provides several files for exploring these possibilities: **Cube & Pict** in which a 3D cube moves against a static background picture when animated and **Cube & Compiled** in which the background is synthesized by ArtMatic. Such systems can yield stunning animation.

There are a couple of features to note in these more complex examples. Both files make use of a particular 2-in/3-out component whose purpose is to take a non-RGB ArtMatic subtree and generate an RGB representation so that the branch can be mixed with a color picture or another subtree that generates RGB (true color). There are a couple of 2-in/3-out components that perform this function. For more details see the [Components Reference](#) chapters of this manual. The other key feature that both files have in common is that they use the **Packed RGB Crossfade** component to mix the background and foreground. This component can be used to mix the output of two branches into a single image. This component has just one parameter **Interpolate** which mixes the two images that feed it. This component however treats **infinity** specially. Wherever it sees infinity, the image is treated as transparent. In our example files, the left-hand image in the mix is the background and the right-hand image is the sphere or cube in the foreground. With interpolate set to its maximum value, the sphere or cube is opaque **except** for the areas of infinity which surround it.

Moving On

In combination, the new features vastly expand ArtMatic Pro's capabilities. To get the most out of ArtMatic, we recommend that you perform the advanced tutorials found in the new [Getting Deeper](#) chapter and browse the [Component Reference](#) chapters which have been improved and include detailed descriptions of the large number of new components. A number of new chapters have been added to the manual and all chapters have been updated and are worth revisiting for users that want to get the most out of ArtMatic Pro.

Overview

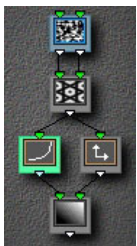
This chapter provides a brief Overview of ArtMatic. A more detailed overview is provided in the [Concepts](#) chapter.

ArtMatic is a unique program which uses mathematical functions to generate images, animation and sound. The program's design is such that one can create astounding images and animation without needing to know anything about math or the underlying functions by using its randomization and mutation features though if one learns about the components and tree structures one can design images and effects **possible with no other tool**. ArtMatic uses a simple, intuitive interface that can be learned in a few minutes but which will reward one with an endless variety of images. With some practice and a little systematic exploration, one can develop the ability to finely tune and control the images which ArtMatic creates. The provided tutorials progressively teach you how to get the most out of ArtMatic by starting with the basics and eventually introducing the concepts and methodology

Like fractal-based programs, ArtMatic relies on simple mathematical formulas that yield rich and complex results. ArtMatic, however, is much more than a fractal explorer (in fact, only a few of ArtMatic's functions are fractals). Its component trees are akin to the complex systems of chaos and complexity theory where modest changes in input can yield dramatic changes in output and where systems of mathematical equations generate a huge range of images from geometrical-type images to clouds of stars to bubbling, liquid surfaces to spinning planets to description-defying images of ineffable beauty.

Elements

ArtMatic's central display area is called the **canvas**. It can be thought of as a window on a vast world of which only a small portion is visible. New portions of this world can be brought into view by clicking on the canvas and dragging left, right, up or down. Using the zoom tools or the up and down arrows, you can zoom in or out. Some systems are so rich that merely zooming in or out or dragging the canvas can produce dramatically varying results.



At ArtMatic's core is the tree of mathematical equations called the **structure, structure tree or component tree** which is presented as a configuration of linked tiles called components. Each tile has a particular mathematical function (also called a **component**) associated with it. Clicking on a tile pops up a list of the available functions. When a tile is selected, its parameters (or settings) can be edited using the parameter sliders at the bottom of the page. There is a popup menu for choosing from a variety of basic structures which can be further modified (by adding and deleting tiles). Any particular structure together with a particular set of function assignments is called a **system**. Any particular system is capable of producing a wide array of images by either modifying the system's parameters.

The output of the structure tree is processed by a shading algorithm or **shader** which determines how the system is drawn and how colors are assigned. The colors used by the system are taken from a special kind of editable color palette called a **gradient or gradation**. ArtMatic provides simple yet powerful tools (described in the User Interface chapter of this manual) to create and modify these palettes. Note that there are different sets of shaders in the **Explore (black and white)** and **Sound** modes than when on either of ArtMatic's color pages.

A system's parameters can be changed in a number of ways. You can click on tiles and move the parameter sliders by hand, click on the left-hand die to jumble all the parameters (while leaving the component assignments unchanged), or use the **mutations** dialog which allows you to see many variations of the system at once.

In an abstract way, you can think of the area visible on the canvas as a location in a multi-dimensional universe. Every time a parameter changes or the canvas is magnified or offset, you are taken to a new location in the universe. ArtMatic features **keyframes** which allow you to store intriguing locations by clicking in any empty slot in the left-hand toolset (or by clicking the **Add** text button). Keyframes can also be used to create **animation** which can be saved as QuickTime movies.

Important note about keyframes! While keyframes have their own parameters and color gradients, all keyframes in a file use the same system (function assignments) and shader. If any function assignment or the shader is changed, each keyframe will change.

Modes/Pages



ArtMatic has four basic **modes** or **pages** which determine what sort of art it generates, whether black and white images, tiled color images, color images, or sounds. **Animation** can be rendered from all but the Sound page (where one renders sound files instead). At the top of the ArtMatic window are tabs which let you switch between pages. The tabs in left to right order are:

- **Explore (black and white)**: This mode is used to create grayscale images and animation. The grayscale transitions are smoother in this mode than when using a grayscale palette in the color mode. A limited number of shaders is available in this mode. **Tip**: Black and white animations can be used with movie editing programs to create astounding wipe and fade effects when these movies are used as mask layers.
- **Tiles (color)**: The current picture is displayed as a symmetrical tiled pattern that repeats horizontally and vertically. This is useful for creating patterns to be used for the Macintosh desktop or as Web page backgrounds.
- **Explore (color)**: This is ArtMatic's **default mode** and is used for creating color graphics and animation.
- **Sound**: ArtMatic can also be used to generate drones, waveforms and rhythmic sequences from animated systems. Sound is generated when the system is animated by deriving pitch information from the canvas as the sound generator circles over the canvas. More information about sound generation can be found in the [Sound Mode Tools](#) section the user interface section of this manual.

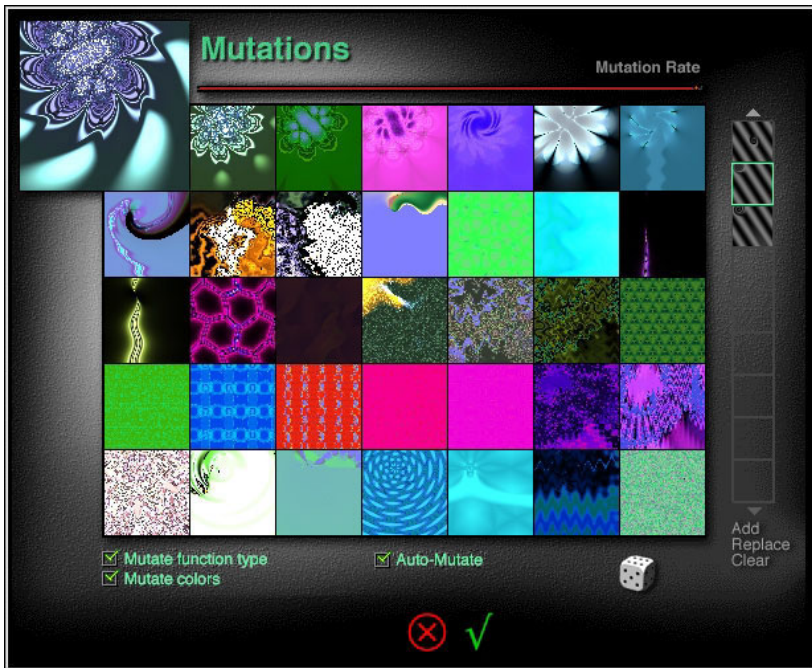
Animation

ArtMatic provides powerful tools for rendering stunning high-resolution abstract (and some not-so-abstract) animation. Note that due to the complexity of ArtMatic's images, real-time animation is done at a low resolution, but the QuickTime rendering is done at high resolution with anti-aliasing. There are two basic types of animations which it can create. **Keyframe animation** creates animation by morphing from one keyframe to another. **Random path animation** uses a pre-defined pseudo-random curve to modify the system's parameters.

This topic is covered in greater detail in the [Animation](#) chapter of this manual.

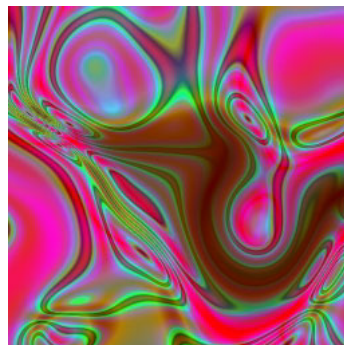
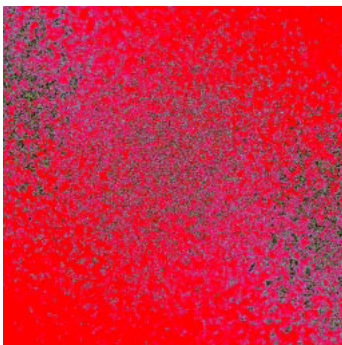
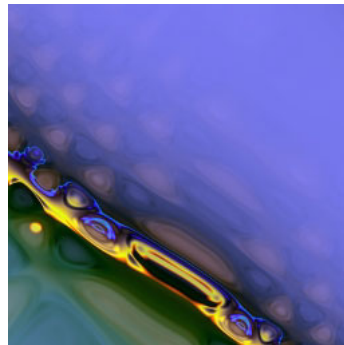
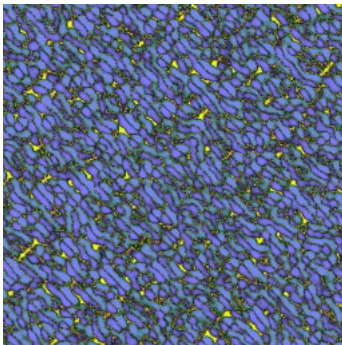
Exploring

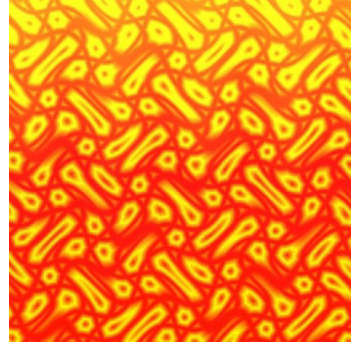
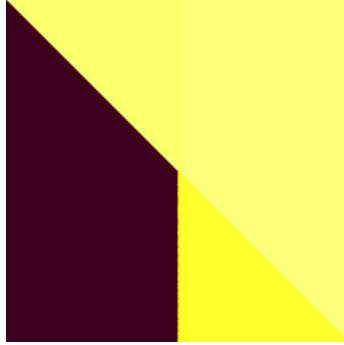
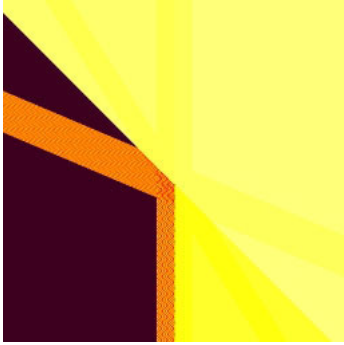
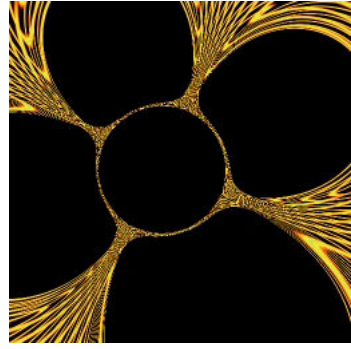
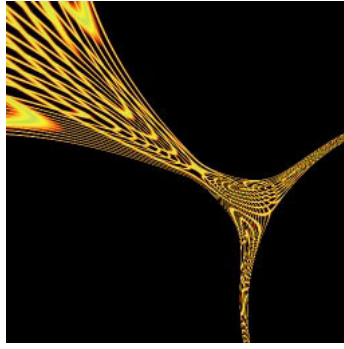
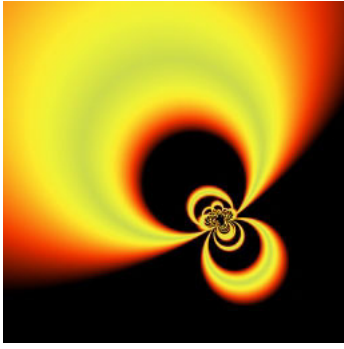
The best way to get a sense of what ArtMatic can do is to spend some time exploring on your own and examining the provided sample files which will give you an idea of the incredible variety of image and animation you can create. The [mutations](#) window is an invaluable aid for finding starting point. By turning all its options on, you can explore a vast variety of systems with just a few clicks of the mouse (remember that if function mutation is turned on all your keyframes will be affected).



Once you have found a starting point, save it as a keyframe by clicking in an empty keyframe slot. Use the small left-hand die button to randomize the parameters, Zoom in and out to explore the system from up close and afar. You will be amazed at the worlds that you will find when zoomed in very close and very far with some systems. When you are zoomed in close, it is useful to drag the canvas left, right, up and down. If the system includes fractal or iterative functions, you can increase the resolution using the iterations preference found in the [Preferences](#) dialog.

The table below will give you an idea of how rich a single ArtMatic setting can be. Each table row consists of images created by zooming in and out (and dragging the canvas left and right) on a system but making no parameter changes. Imagine how different the images would have been if the parameters had changed too!





Concepts

This chapter explains the concepts which underlie ArtMatic. The material sheds light on ArtMatic's inner-workings and can help you get the most out of the program. It is possible to create wonderful art, sound and animation without knowing the details covered in this chapter. The [Getting Deeper](#) chapter provides a series of lessons designed to deepen your understanding of the concepts presented in this chapter and to provide techniques for mastering ArtMatic .

Structures, Components and Functions

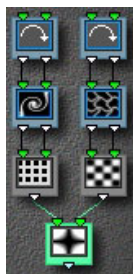
What is ArtMatic?

ArtMatic is a unique environment designed to allow anyone, regardless of their experience and expertise, to create astounding graphics, animation and sound. It can be used as a tool for 'Zen-mode' exploration where you roll the dice, mutate a system and discover images. Or, it can be used in a more directed manner where you design and modify systems to get particular images and effects. It is a program with many guises and applications. There are a number of different ways to answer the question "What is ArtMatic?" The one that is most meaningful will depend on your background. ArtMatic can be thought as:

- **A modular graphics synthesizer.** In a simple sense, ArtMatic is a patchable, modular graphics, animation and sound synthesizer--the graphics equivalent of a patch cord-based music synthesizer--whose component functions can be patched together in almost limitless ways to generate an astonishing range of images.
- **A visual programming language for image synthesis.** An ArtMatic "patch", or preset, is a mathematical program for processing the points of the visible plane and generating an image from the results.
- **A workshop for mathematically-based image creation.** At its core, an ArtMatic structure is a programmable system of equations. Users well-versed in mathematics can design powerful functions and equation systems that create an astonishing range of images. ArtMatic's components provide a wealth of mathematical primitives that can be chained together to implement a wide range of mathematical systems. ArtMatic is a great environment for discovering the beautiful, mysterious world of mathematics.

Structure/Component Tree/System

ArtMatic generates images by pumping the co-ordinates of the canvas' points (and, in some cases, a counter value) through an arrangement of function tiles which we call a **structure or component tree**. The component tree is a configurable arrangement of interconnected tiles. Each tile has an associate function, or component, that has up to three modifiable settings/parameters. **Keyframes** allow you to save snapshots of the parameter settings which can later be recalled or animated. Each tile has from one to four inlets and one to three outlets through which data flows.



A gradient-based component tree

Components are what we call the functions that can be assigned to tiles. The glyphs at the top and bottom of a tile are its inputs and outputs. Values enter through the top and leave through the bottom. A component is an equation which takes the incoming values, modifies them and sends them out via the outlets. The components that are available to a tile are determined by the number of the tile's inputs and outputs.

Due to the nature of the components we have provided, even very simple trees can create astonishingly complex images. Subtle changes to a tree's shape can have a dramatic impact on the image it generates. You can change the shape of the component tree using any of the tree editing tools and commands or by using the [Choose Structure pop-up menu](#). These commands and tools



An RGB-based tree

are covered in the user interface chapter and the various tutorial chapters.

Images can be manipulated in a number of ways: by changing the arrangement of tiles, by changing the functions associated with the tiles and by changing the settings of the individual component parameters. ArtMatic generates animation by modifying component parameters over time.

Periodically, the word **system** is used in this documentation. A system refers to the combination of a particular structure and a particular set of component assignments. For example, the same structure can have different function (tile) assignments. The different tile assignments will generally yield dramatically different results but can be said to have the same structure since the trees are the same shape.

Two types of systems. ArtMatic has two types of trees: gradient-based and RGB-based. The tree's type is determined by the number of inputs and outputs of the tree's last tile. **Gradient-based trees** use the active [gradient](#) for their colors and have a one- or two-output tile in the final position. **RGB-based trees** (RGB = red, green, blue) or "true color" trees have a three-output tile in final position whose output determines the color directly. The left, middle and right outlets are red, green, and blue respectively. RGB-based structures are frequently used when a picture or QuickTime movie is used as a tree input since RGB-based output provides the possibility of preserving the picture or movie's original colors. (RGB-based trees also allow you to perform spectacular color manipulation).

If the final tile has two outlets, the two output values are treated as two independent gradient-based images which are combined before the final color shading is applied.

Combining gradient-based and RGB-based systems. There are a number of special components that allow you to create trees that mix gradient and RGB-based color.

How images are calculated

Points of the canvas . You can think of the visible canvas as a downward-facing camera's view of a portion of an infinite grid or plane (the Cartesian plane you may remember from geometry class). Every visible pixel is treated as a point on the plane/grid. (Pixels are the individual points of which a picture is comprised.) When the canvas is in its default centered position, the center point is 0, 0; co-ordinate values increase to the right and up. The **component tree** acts as a giant equation that takes a visible point's co-ordinates as input and generates the color used to draw the point.

Data flow. ArtMatic feeds the x and y co-ordinates of the upper-left pixel (grid point) into the left and right inlets of the topmost component(s). These values are modified by the component and passed on to the next component which passes its output to the next component and so on until a value emerges from the final component. The value that emerges from the final outlet determines the color used for the associated grid point. This process is repeated for each visible point to create the final bitmap image. In some cases, the final component has two outlets. In this case, two separate images are created then blended to create the image you see.

Mapping the colors. Precisely how the final output value is mapped to a color depends on whether the tree is gradient-based or RGB-based. For RGB-based trees, the color is calculated directly from the three output values of the last tile. (The tile's three outlets represent the red, green and blue content respectively). If the tree is gradient-based, the color mapping is handled by the active color shading function (these are discussed in the chapter [Shaders](#)). Generally speaking, the lowest possible value in the system is mapped to the lefthand color of the gradient and higher values are mapped to the colors further right.

Manipulating the Visible Plane. To change the region of the plane/grid that is visible, click and drag the canvas left, right, up or down or use the zoom (magnification) tools. When exploring a system, it is a good idea to zoom in and zoom out on the system since the character of many systems varies dramatically when viewed from up close and from afar. Also, dragging the canvas left or right can reveal surprising details. The default center co-ordinates of a new system are 0,0 with the zoom set so that the x and y values

range from $-\pi$ to $+\pi$ (from minus to plus Pi). The [Edit Camera Path](#) dialog has a section which displays the current zoom level and the x,y co-ordinates of the canvas' center point.

Final or main component. Most systems have a single component in the last row of the tree because only a single component's output can be used to calculate the image. This component is sometimes called the main component. If there is more than one component on the last row, the leftmost component's output is used.

Streams. The chain of flow through the system is sometimes called a **stream**. Stream simply refers to the flow of numbers and values through a set of components. An RGB-stream or a 3D-stream simply refers to a series of components through which RGB or 3D values are flowing.

Third Global Input - Time Input

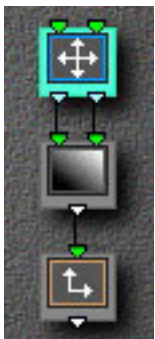
The description above applies to systems that have two inputs at the top since they are the most commonly encountered systems. A system may also have three inputs at the top **or** a tile with an open third input in the interior of the tree. This third input (called the third global input or time) is a counter value which is steadily incremented over the course of an animation. It is generally used when creating animation to allow time to modulate the system directly.

There are many ways to use and interpret the third global input which is just a simple counter that starts at 0 and is incremented over the course of an animation. It can be viewed as a time input since its value increases steadily with the passage of time and provides a simple means of modulating the system over time. It can be used as a control source for those components that use the final input as a control source (**z Wipe** or **Packed RGB z Sort**, for example). It can also be used as a third spatial co-ordinate which increments over time. When used in this last way, you can think of the canvas as being a 2D slice of a three-dimensional space. With each increment of the third global input, the canvas is moved along the third dimension so that each frame is a slice from another point along the z-axis.

At the end of an animation the counter has a fixed value called the **Termination Value** that is the same for all systems (regardless of the animation's duration). The third input's value increases steadily from 0 to the Termination Value over the course of the animation. To modify the counter's normal behavior, insert a 1D scalar (1-in/1-out) component in the tree to scale or alter the values. In this way, time can be made to cycle or accelerate or change at random.

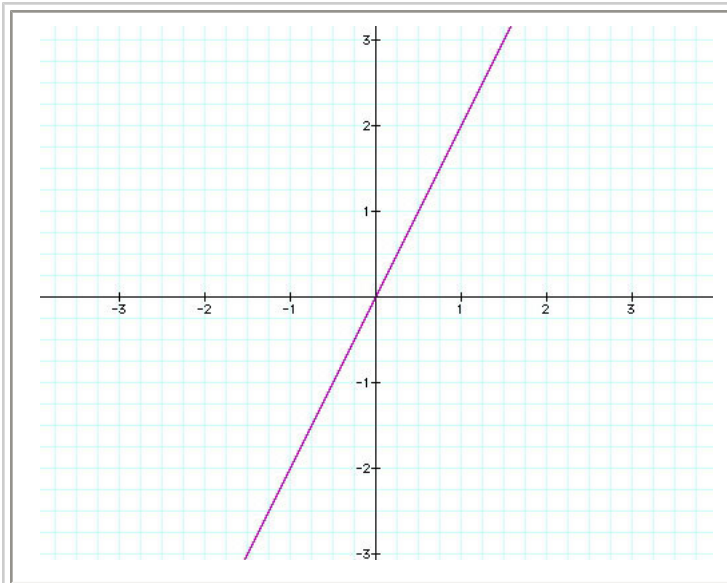
Interpreting Gradient-Based Systems

A gradient-based system can be thought of as giant equation that generates a 3-dimensional surface whose altitude is mapped to a color of the active gradient. (Mathematically speaking, the structure is a function such that $z = f(x,y)$.) In this sense, a gradient-based system is like a topographical or contour map. The tiles (components) generate the surface to be mapped, and the gradient and shader determine what colors are used to draw the surface. The shader has a large impact on the image's final appearance. Some shaders are progressive while others are cyclic and others are complex functions which provide 3D lighting effects. You can also think of the ArtMatic structure/component tree as a function in three dimensions $z = f(x,y)$ with the x and y values being the co-ordinates of the visible plane. Rather than plotting the z value as a third spatial dimension, ArtMatic plots the value as a color which is determined by the [shader](#), the active gradient and the shading options. Shaders are discussed in the [Shaders](#) chapter of this manual.

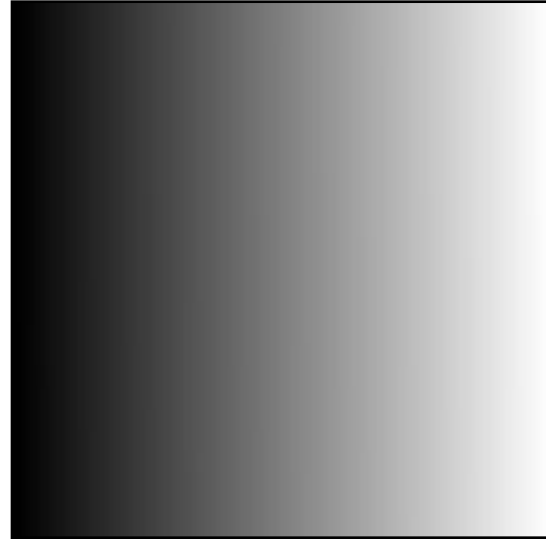


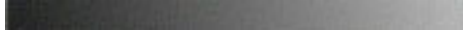
This is perhaps easiest to see if you take a simple structure with simple linear functions such as those in the tree to the left. The functions in the tree are the **Scale Function** ($Ax + B, Ay + C$), a scalar (1-out) component whose function is $z = Ax + By + C$, and a simple slope function ($Ax + B$). A, B, and C refer to the parameter sliders for each component. Since A, B, and C are different for each component, the following shorthand is used: A1 refers to the A parameter of component 1, A2 refers to the A parameter of component 2, etc. To create the simplest case, set A1 (the Scale function's x-multiplier) to 2. Set A2 and A3 to 1.0 so that they have no effect. And set B1 and B2 (the Scale function's y-multiplier) to 0 so that the final output value is really just a function of x. With the sliders set as described, we really have the function $z = 2x$. In the **Doc. Example Files** folder, there is a file named **Simplest Case** whose first keyframe is set up as described. Below is a graph which plots the function $z = 2x$ and next to it is ArtMatic's rendition using the linear shader.

Tip! Explore this simple project file and note the effect that parameter adjustments and the use of the different shading functions can have. Switch to the color page and explore what effect gradient (palette) changes have. Whenever you want to see what a particular function does by itself, it is helpful to return to this project and choose the function you want to explore from the component popup.



Graph of the function $z = 2x$



The picture above is ArtMatic's rendition of the function at left using the linear ceiled shader on the Black and White page. The Black and White page uses the following gradient  .

Note how low values of z are drawn using the left-hand colors and higher values are mapped to the right of the gradient.

3D Objects

There are a number of 3-in/3-out and 2-in/3-out components that generate 3D objects such as cubes, spheres, tunnels and rooms. To learn about using these 3D objects, see the [QuickStart III](#) tutorials and the [Getting Deeper](#) chapter of this manual as well as the detailed descriptions of the various 3D components in the [Component Reference](#) chapters.

In general, these components will be found towards the top of branches of the structure tree and will be followed by components which provide surface texture and coloring. There are two flavors of 3D objects: parametric and non-parametric. 3D object components that have 'parametric' in their name take the surface texture with them when they move through space. The other 3D object components behave differently; the objects they create move 'under' the texture, as if the texture were a patterned sheet under which they could move. This is covered in some detail in the tutorial and reference chapters.

There is another interesting property of most 3D component tiles. For all points beyond the object boundary, the value infinity is sent out of the object. In RGB-based systems, infinity is drawn with the first auxiliary color; in gradient-based systems, infinity is drawn using the rightmost color of the gradient. Components that mix the output of two or more 3D/RGB streams treat infinity as transparent. These issues are covered in the [Getting Deeper](#) chapter.

Added in version 2.53! Version 2.53 provided added control over 3D objects by placing ArtMatic's virtual camera in 3D space so that offset components move the virtual camera in three dimensions. If there are multiple offset components in the tree, the topmost component will be global and influence all 3D objects lower in the tree. Other offset components in the tree move objects individually in 3D space. This allows much more complex 3D animation than was previously possible-- especially because 3D objects within

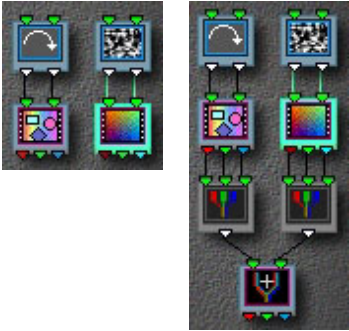
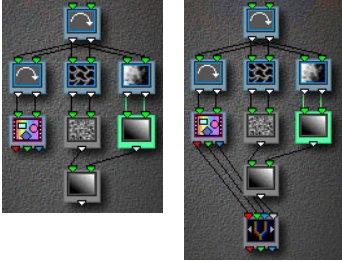
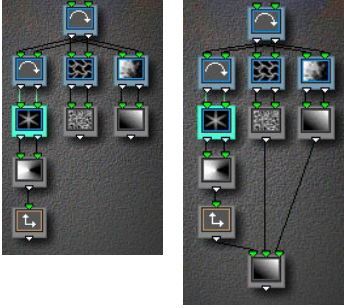
compiled trees are now sensitive to the parent systems 3D offsets.

Tree Structure Notes

Tree Completion

Because ArtMatic uses the output of a single component (the first component of the last row) to create an image, trees should be **complete** (i.e. have only one component with an unconnected output) except for cases where an unconnected output is used for depth cueing or global shading. Because you will often find yourself needing to connect parallel branches while editing tree structures (especially if you are creating a system with several pictures or 3D objects), the **complete tree** tool has been provided to automatically complete incomplete trees.

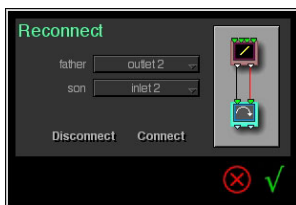
When the **complete tree** tool is clicked, ArtMatic will mix the branches of the tree by adding and connecting the appropriate components. For example, to mix **two parallel RGB** branches, ArtMatic will add any required **pack** components then mix the branches with a packed RGB mixer. **Mixed systems** that feature an RGB branch and a 1D gradient-based branch will be mixed with the appropriate 4-in/3-out mixing component. **Multiple 1D branches** will be mixed with a two or three to one mixer. In rare cases, ArtMatic may not be able to determine how to mix the system, in which case a beep will sound when the **complete tree** tool is pressed. If this happens, you will have to add some tiles yourself to complete the tree. If there is an RGB branch and a two output branch to mix, for instance, a **2D Scalar** (2-in/1-out) component should be added at the end of the two output branch. The table below shows a few examples of structures before and after application of this tool. You can often save yourself a lot of work by using this tool.

Two RGB Branches	RGB Branch and 1D Branch	Three 1D Branches
		
To combine two RGB branches, Artmatic adds the necessary pack components as well as the mixer.	ArtMatic mixes an RGB and a 1D branch with the 4-in/3-out version of RGB Interpolate .	Three 1D branches are mixed with a 3D scalar component.

Connection Dialog

Re-connecting loose inputs and joining branches. There are a number of common editing operations which can leave tiles with unintentionally unconnected inputs, and there may be situations where you want to change the parent of a particular tile. There are two ways to change the connections between a child tile and a parent tile found higher on the tree. Automatic connection (**command-click**) forces an automatic connection from a child to a parent tile. The new connection dialog (**option-command-click**) allows for custom manual connection of tile inputs and outputs. Both methods require that the parent tile be higher on tree than the child tile and can be used to either connect loose or open inputs or to change a tile's parent.

Automatic connection. Automatic connection is the quickest way to change a tile's parent or force connection of unconnected inputs . To force an automatic connection, select a child tile then **command-click** a tile higher in the tree that will become the tile's parent. Be aware that choosing a parent tile from another branch will sometimes cause ArtMatic to re-arrange the tree's layout. **Undo** can be used to undo this automatic re-patching if the results are undesirable. ArtMatic does its best to determine whether you are trying to completely change the parentage of the child, or whether you are simply trying to connect and open input. In some cases, ArtMatic will not be able to determine an appropriate automatic connection. In such cases, use the **Connection Dialog** described below.



Connection Dialog. For greater control of child/parent connections, use the **connection dialog**. It allows you to create (or break) connections between any inputs and outputs of child/parent tiles and even to split parentage between tiles. To invoke it, click on the child tile and **option-command-click** any tile higher on the tree. The dialog displays the parent and child tiles and provides two ways of establishing connections.

1. Direct editing. Click on a parent output then click on the child's input to which to connect it.
2. Via the father/son pop-up menus. Choose the father's output, choose the son's input then press either the **connect** or the **disconnect** buttons.

An exercise: Use the connection dialog to horizontally flip an input image. **Hint:** start with the **RGB 1 Channel** structure (available from the structures popup) and use the connection dialog to re-arrange the connections between the first and second components in the tree.

Advice

If some or all of this seemed confusing, don't despair. The [Getting Deeper](#) chapter will help you make sense of all this. After reading the other chapter and performing the [Getting Deeper](#) lessons, you may want to return to this chapter as it may be more meaningful on a second pass.

QuickStart - The Basics

This tutorial chapter introduces you to techniques for getting started with ArtMatic Pro, which is a very powerful tool that provides many methods and levels of exploration. We recommend that all users both read **and perform** each of the tutorial chapters. (Even experienced users will discover new techniques and exposure to new features.) This chapter guides you through exploration of ArtMatic but does not explain the hows-and-whys of ArtMatic. To learn more about how ArtMatic works, you will want to read the Overview and reference chapters of the manual.

We strongly encourage you to explore the many example files we have provided, since ArtMatic is capable of creating an extraordinary range of images which often surprises the most experienced users (including the program's creator, Eric Wenger).


ArtMatic can be used to both discover images (what we call "Zen-mode" exploration) and design images and fractals. It is also a very powerful tool for manipulating pictures and QuickTime movies. This first tutorial chapter introduces the basic techniques for discovering new images (or "systems"). The other tutorial chapters introduce you to techniques for manipulating the ArtMatic structure tree and for creating complex images.


Quickest Start Tutorial - The Basics

Before we start, move the mouse around, mousing over the various tools, and notice that there is green text in the lower-righthand corner that displays tool names and other helpful information. These Tool Tips are a great help when you are first learning ArtMatic. You will notice that, for the most part, all icons and text act as tools that you can click or click & drag to manipulate ArtMatic.

Find A Structure

Whether you are an expert or a novice, you will often want to let ArtMatic generate a system that will be the starting point of your exploration.

Choose a basic tree structure. Click on the **Choose Structure** selector  and choose a structure from the list which pops up. Don't worry about which structure to choose or what the structures' icons do. All of them produce amazing results.





Mutate the system. Click the large middle die  to shuffle the assignments of the system's tiles. If you aren't intrigued or enticed by what you see, click the die again. Keep in mind that if you zoom far in or far out you may discover surprising details not visible at the default zoom level. When you find something that captures your interest, it is time to move on to the exploration stage. Remember that the range of images possible from any one configuration of any single structure is mind-boggling. In the exploration phase, you will explore the possibilities of the configuration given you by the "big die".

Super-mutate - Control-click the large die to let ArtMatic both choose a random structure and shuffle the tile assignments.

Explore the structure


ArtMatic systems sometimes generate images whose visible details dramatically different when zoomed in or out or when the canvas area is scrolled.

Zoom in/Zoom out. There are several ways to zoom in and out on a system:

- **Zoom buttons**   - Press either of the zoom buttons   to continuously zoom in or out. While ArtMatic is zooming, it switches to low resolution rendering. As soon as you release the mouse button, ArtMatic re-renders the system at high resolution.
- **Plus and minus keys** - Press the '+' or '-' key on your keyboard to zoom in or out by a factor two.
- **Arrow keys** - Press the up or down arrows keys to zoom continuously.

Scroll the Canvas. The image display area is called the Canvas. Click and hold the mouse button anywhere in the Canvas and drag left or right to scroll it.



Using keyframes to save locations. When you find an intriguing location, click on the [keyframe palette's Add](#) button. A thumbnail of the image is added to the palette. Whenever you find an intriguing location, add it as a keyframe. Clicking on a keyframe recalls its settings. Keyframes can also be used to animate a system. All keyframes share the same tile/function assignments, structure, and color shader. So if you roll the big die or change the color shader, you will notice that all the keyframes will change.


Stroll a random path. Any one ArtMatic system has nearly infinite possibilities. [Random Path Animation](#) provides one method of exploring this huge image space by continuously changing the parameters of every tile in the system. Click on the Random Path Animation button  to start the random walk. When you find a position you like, click on the pause button and add a keyframe so that you can return to these settings. Continue your random stroll by pressing option-spacebar to resume.

Note realtime animation is done at low resolution since ArtMatic graphics are computationally very complex. When such animation is rendered to disk, the rendering is done with high resolution.

The speed of the random walk is controlled by the **Delta Time** setting. Smaller values result in faster changes and larger values result in slower changes. You can change the Delta Time setting in either the [Preferences](#) dialog or in the [QuickTime Export](#) dialog.

You can manually stroll the random path. Click on the Random Path Scroll button and drag left or right to manually stroll. *For fine control* of manual strolling, press and hold the option key then click and drag the Random Path Scroll button.

The small dice. By clicking either of the smaller dice, you can tweak the current system's settings. Click the die  at the left to shuffle the tiles' parameters. (The tile assignments are left alone but their settings are shuffled.) Click the die at the right  to randomize the current gradient's colors. When you find a variation you like, add it as a keyframe.

Change the gradient. The gradient is the color palette used to draw the system. Each keyframe can have its own gradient. To change the gradient, click on the Choose Colors tool  which pops up a menu of the gradient library. Mouse over the gradient and notice that the display changes. Click on any square to pop up the color picker. Choosing a color from the color picker replaces the old color with the selected one. There are several more tools available to manipulate gradients which are covered in detail in the [User Interface](#) chapter of this manual.

Note: The gradient is used in all systems whose last component has two or fewer outputs. If your system has a three-output component in the last row, you will need to choose another structure for gradient manipulation to have an effect.



Mutations Dialog. Choose [Mutations](#) from the Edit menu. Click on any of the thumbnails and watch the new mutations that are created. Turn on **Mutate Colors** and click on any thumbnail and note the color variations. Turn off the **Auto-Mutate** checkbox. You can now click on thumbnails without triggering new mutations. This makes it possible to add multiple mutations as keyframes. To add a mutation as a keyframe, click on the mutation then click on the **Add** text button. To create new mutations, you can turn **Auto-Mutate** on again or click on the die button or the large parent image found at the upper-left of the mutation set. Set the **Mutation Rate** control (near the top of the dialog) to its maximum value by clicking at the tool's right edge. Now, set the Mutation Rate near the minimum value and notice that the mutation is very subtle.

Note: Be careful of the [Mutate Function Type](#) checkbox. Function assignments are shared by all keyframes. If you turn this option on, all keyframes will be effected.


The **Mutations** dialog is a very powerful tool for exploring and refining ArtMatic systems. See the [User Interface](#) chapter for more information about its features.

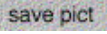
Animation

ArtMatic Pro can create incredible animation by morphing between keyframes or by animating the random path. Even if you think that you are only interested in still images, we recommend that you explore ArtMatic animation as it is capable of creating wondrous surprising images like nothing you have ever seen. You will notice that all of the files in the main examples library have two or more keyframes. When exploring these files, we recommend previewing the animation (as described below).

Animate Keyframes. If you haven't done so, create several keyframes as described in the steps above. To see a low-resolution, realtime approximation of the keyframe animation, click on the Animate Keyframes button . To see a higher-quality (but smaller) preview, type command-H (or choose *Preview* from the Animation menu). A high resolution QuickTime movie can be rendered by pressing the QuickTime Export button . Since QuickTime rendering can be time-consuming, we will leave this for you to explore on your own. **Note:** When this animation is created, it uses the time specified in the [QuickTime Export](#). You can change the animation's duration in the Preference dialog or in the QuickTime Export dialog. Animation is discussed in detail in the [Animation](#) chapter of this manual.

Viewing and Saving Your Work

View anti-aliased. Click on any keyframe to restore its settings. Click on the **Render Full Screen** tool . ArtMatic displays the image with high-quality anti-aliasing. Click anywhere to dismiss the image. Anti-aliasing is also applied when pictures are saved or animations are rendered.

Save your work. Use the File menu's **Save As** command to save the current system and keyframes in ArtMatic's native format. This format is more compact than saving a picture but can only be read by ArtMatic. To save the image as a high resolution, anti-aliased PICT format file which can be opened in other programs, click . When you click the [Save Pict](#) tool, the Save Picture dialog box is displayed, which lets you choose the dimensions of the image that you would like to save. You can choose dimensions from the dialog's popup menu or type in your own dimensions. (Numerical values can also be changed by clicking and dragging up or down.) If you enter your own dimensions, press the **Preview** button found below the image to force the preview to be drawn with the new aspect ratio.

Save your work now, if you like the system that you have created. In the next lessons, you will be using example files to continue your exploration.

Color Shaders




When an ArtMatic system has a one-output component on the last row, ArtMatic uses a color shading algorithm to map the tree's output to the colors of the active gradient. The term *gradient-color system* describes such systems (as opposed to RGB or true-color systems which have a three output component at the bottom of the system). The tools shown at left are used to determine the color mapping or shading. For this exploration, we will use a very simple system so that the relationship between the shading algorithm, the active gradient and the resulting image will be easy to see. We encourage you to repeat these steps with systems of your own creation.

In this lesson, we won't describe how the shading algorithms work; the shading algorithms are covered in detail in the [Shaders](#) chapter of this manual; some users may want to take a look at the Shaders chapter before proceeding. For this lesson, you don't need to understand why the image is drawn the way that it is. What is important is to notice that the same system can appear very different when drawn with different color shading algorithms (also called color shaders).

Open the file **Simplest Case Color** that is found in the Documentation Examples folder located in the same folder as the documentation.

Choose a shading algorithm. Click on the **Choose Color Shader** tool to pop up the color shader popup menu. A number of choices are available. Choose Cyclic Clut and observe the image. Now, choose each of the available choices up to and including *Procedural C*.

Auxiliary Colors. Choose the next option: *D: Log + Depth Cueing + Color Filters*. Notice the three color squares that appear above the Choose Color Shader tool. These auxiliary color are available for some complex color shaders and when some shading options (discussed below) are selected. Click and hold the mouse button on the topmost auxiliary color to pop up the color selector. Choose black from the 'popular colors' squares at the left edge of the color selector. After choosing black, choose other colors. Repeat this for each of the three auxiliary colors. Note that these colors are global. The shader and auxiliary colors are common to all of the keyframes in the file. If you change the shader or aux. colors, every keyframe in the file will be affected.

Shading Options. Below the Choose Color Shader tool is the Shading Options popup menu . Pop up the menu and notice that there are a number of options (all of which are covered in detail in the [Shaders](#) chapter). Notice that *Depth Cueing Small* has a checkmark next to it. Depth cueing uses the first auxiliary color to filter the image. The complex shading algorithms whose names contain 'Depth' all turn this option on. You can control the amount of depth cueing effect by choosing the other depth cueing 'weights'. Now, use the Choose Color Shader popup menu and choose *Cyclic Clut*. The depth cueing is automatically turned off. You can turn it on manually by choosing any of the depth cueing options in the Shading Options popup menu.

Take some time to explore your own creations or examples from the examples library, and see how the images are affected by changing the color shader and shading options.

Tiles, Parameters and Locks

As explained elsewhere in the manual, the ArtMatic structure tree is a modular graphics synthesizer made up of components (tiles) that synthesize an image. It can also be thought of as a chain of graphics filters or as a programmable mathematical system. Each tile has a function associated with it, and each function can have up to three parameters that influence its behavior. The images created by the system are a combination of the tree structure, its tile assignments and the tile parameters. The Mutation dialog and the left-hand die both mutate these function parameters, but you can also edit them directly.

Parameter sliders. Find a structure to explore by control-clicking the large die until you find a system that intrigues you. Click on the tree's tiles and notice that zero to three parameter sliders are available for the selected tile. When you mouse over a parameter slider, the Tool Tips area displays the parameter's name and its setting. Explore how the image changes as you manipulate the parameters. As you explore, add a couple of keyframes.

Parameter envelopes. If you click on the letter that appears to the left of the parameter slider, the Parameter Envelope dialog appears.

You can change the parameter's 'envelope' (the values it has over the course of an animation) using this dialog. You can also, directly enter numerical values by typing into the fields that appear at the bottom of the dialog box.

Note: For angle parameters, you can type the number of degrees (i.e. 30) and type 'd' to force ArtMatic to convert from degrees to ArtMatic's internal format.

Parameter locks. Notice that there are lock icons next to the parameters. You can use parameter locks to limit the parameters that are mutated by the system. **Shift-click** any unlocked parameter to lock all of the parameters. Selectively unlock some parameters. Open the **Mutations** dialog (make sure to turn off Mutate Function Type) and mutate the system. The system's mutations are restricted to modulations of the unlocked parameters. You can lock a tile's function assignment by locking all three parameter locks (you need to lock all three even if the function uses less than three parameters). The tile's function assignment won't change now if you roll the big die or use the Mutate Function Type option in the Mutations dialog. Try this for yourself. Shift-click any unlocked tile to lock all parameters and functions. Now, selectively unlock some parameters and open the Mutations dialog. Turn on the Mutate Function Type option.

To unlock all parameters, shift-click any locked parameter.


[Continue with the next tutorial by clicking here](#)

QuickStart II

More Features & U.I. Tour

This tutorial chapter provides a tour of useful ArtMatic features and some simple but helpful techniques. This chapter does not provide detailed explanations of these features; conceptual information is left to the other chapters of this manual. While not required to perform the lessons in this chapter, we recommend that you read the Intro, Overview and Concepts chapters of the manual (either before or after performing these lessons) as they will give you a deeper understanding of this program.


Lesson 1 - Pict/Movie Components

Among the most popular and powerful new additions to ArtMatic Pro are the  [Pict/Movie](#) components which make it possible to use pictures, QuickTime movies, and iMovie DV streams in ArtMatic structures. These components make dramatic special effects and image manipulation possible. There are two basic types of Pict/Movie components: 2-in/1-out and 2-in/3-out. Both component type have two inputs which are interpreted as coordinates for the pixels (points) in the picture. The 2-in/1-out version of the component generates the brightness of the pixel found at corresponding position (0,0 is the picture's center). The 2-in/3-out version of the component generates the red, green, and blue values (RGB) of the corresponding point. The main examples library includes many examples of systems that use picture and movie inputs.

1. Setting Up The Structure

Choose a simple structure. Choose **basic** from the **Structures** popup menu. The structure starts with a 2D vector (2-in/2-out) component. The second component is a 2D scalar component.

Set up the tree. For the first component in the tree, choose the **Rotate A** function. For the second component, choose the **Pict/Movie** component (note that this component is only available for tiles that have two inputs and one output or two inputs and three outputs).

Choose a picture. Unless you have replaced the file named **DefaultPicture**, you will see the U & I Software logo. By default, the picture named DefaultPicture (found in ArtMatic's home folder) is used when you use the **Pict/Movie** component. To use a different picture, click on the **Pict/Movie** tile to make it active. Now, mouse over the Choose Pict/Movie tool  and hold down the mouse button to pop up the Pict/Movie input selector. Choose any slot labeled **Open**. The Open File dialog appears to let you choose a picture. Choose any PICT format picture or QuickTime movie. If QuickTime translation is turned on, you will be able to choose any format picture. ArtMatic lets you use up to four different pictures or movies per file.

Replace the picture. To change the picture used by any slot, hold down the option key and (with the option key still held down) select any picture/movie from the popup. The Open File dialog is presented for you to choose another picture.


2. Simple Picture Manipulations

Tiling/Zooming. Zoom out. You will see tiles containing the picture. Click on the **Pict/Movie** tile. Move the mouse over the parameter sliders and observe the **Tool Tips** area at the lower right of the screen to see the parameter names. Click on the **tiling** parameter's slider and drag it back and forth to observe the effect. Experiment with the tile's other parameters. If you set the **tiling** to 0, you will notice that only a single image appears.


TIP! Whenever you find something you like, you can click on the **Add** button to save that location as a keyframe.

Image Distortion. Click on the first tile and choose the **Ripples** function (if your system has keyframes, they will all change since all keyframes share the same function assignments). Now, experiment with the parameter sliders and note how the ripples interact with the picture. If you save keyframes with different Ripple parameters, you can then play the sequence back by clicking on the **Animate Keyframes** button to have ArtMatic morph between the keyframes.

3. RGB True Color Pict/Movie Manipulations

Click on the last component in the system (it is a 1-in/1-out component) and delete it by clicking on the Delete tool . Hold down the option key and click on the Pict/Movie component to pop up the [Component Replacement](#) popup menu. Choose **Replace with Vector (3 out)**. You will now see a color version of the picture (if not, click on the component again to pop up the component selector and choose the Pict/Movie component). By adding more components after the Pict/Movie component, you can perform sophisticated color manipulation. Color manipulation is covered in the [Getting Deeper](#) advanced tutorials.

4. RGB Mixing and the Pack Component

There are several preset structures available from the Choose Structure popup menu that provide a great starting points for using pictures and movies in your ArtMatic projects. Choose **New** from the File menu. Choose the **RGB 3 Channels** system from the Choose Structure popup menu. This structure features three RGB branches which are mixed by the system's final component. Notice that each branch terminates with a **Pack** component . **Pack** combines three values into a single composite value (or stream) which can be interpreted by some components designed to mix RGB and 3D streams. Components with the term "packed" in them require that their inputs come from a Pack component.

In this system, the first and last branches use the Pict/Movie component and the middle branch generates an RGB texture. Click on the last component in the system. This component mixes the output of the three branches into a single image. Adjust its parameters, and notice how the image changes. Examine how the image changes if you use one of the other **Packed RGB ...** components as the final component.

Press the Animate Keyframes button. ArtMatic will automatically generate a couple of keyframes.

Experiment by modifying the system's parameters and adding components to the three branches (make sure to insert them **BEFORE** the Pack component). Try using a second picture for the righthand branch.

Shift-click any unlocked parameters to lock all the system's parameters. Selectively unlock parameters in the middle branch and use the dice to explore new possibilities. Be sure to preview the animation that results from your experiments.

Further Exploration

This lesson only touched the surface of what can be done with these components. If you have QuickTime movies or iMovie DV clips, we encourage you to explore using them as Pict/Movie sources. To get a deeper sense of what you can do, explore the main examples library; Pict/Movie input is useful for a wide range of applications that have to be seen to be believed. Look for the example folders with 'pict' in the name and example files with the words 'logo' or 'pict' in example file names to identify examples that use pictures as input sources.

Lesson 2 - QuickTour of the User Interface

This section gives a quick tour of some useful commands and shortcuts. For a complete guide to the user interface and shortcuts, see the chapters [User Interface](#) and [Keyboard Shortcuts](#).

1. Super randomize. Control-click on the large die. This action not only randomizes the component assignments but also randomly selects a tree structure.

2. Cycle shaders. Use the [and] keys to cycle through the available color shaders.

3. Tab through the tree. Press the tab key and notice that the next component tile in the tree becomes selected.

4. Cycle components. Press the left and right arrow keys and notice that the component (function) used for the tile changes. You can go through the entire list of components using the arrow keys. (Note that for historical purposes the order used by the keyboard is not identical to the order that they components appear in the menu).

TIP! Use steps 3 and 4 repeatedly to explore the possibilities found in a particular structure tree. Don't forget to zoom in and out to see what you find. A system can look very different when zoomed far in and far out. Zooming can be accomplished with the up and down arrow keys.

5. Add a keyframe. When you find something you like, type command-B to add a keyframe. Add a couple of them.

6. Replace a keyframe. Click on a keyframe and modify the parameter sliders for one or more tiles (components). Hold down the command key and click on any keyframe to replace it with the current settings.

7. Delete a keyframe. You can directly delete keyframes by holding down the option key and then clicking on the keyframe you want to delete.

8. Animate keyframes. Press the spacebar to start keyframe animation playback. Pressing the spacebar again will pause the playback.

9. Random path animation. Hold down the option key and press the spacebar to start random path animation. Press the spacebar again to pause. To resume random path playback, press option-spacebar again.

Lesson 3 - Tree Editing

ArtMatic Pro (unlike ArtMatic 1.2 and earlier) allows you to modify the structure of a file's structure tree. Very few restrictions are placed on the modifications you make. So, it is possible to create structures too big to be displayed in the available area. For the sake of this tutorial, we will keep things simple and show some basic modifications you can make. To get a more complete understanding of tree editing, you should read the [User Interface](#) chapter as there are extensive editing features beyond the scope of this tutorial.


1. Setting Up The Structure


Choose a simple structure. Choose the **basic** structure from the **Structures** popup menu. The structure starts with a 2D vector (two-in/two-out) component. The second component is a 2D scalar(two-in/one-out) component. Click Randomize All (the large die) until you find a pattern you like.

2. Insert Perspective & Rotation

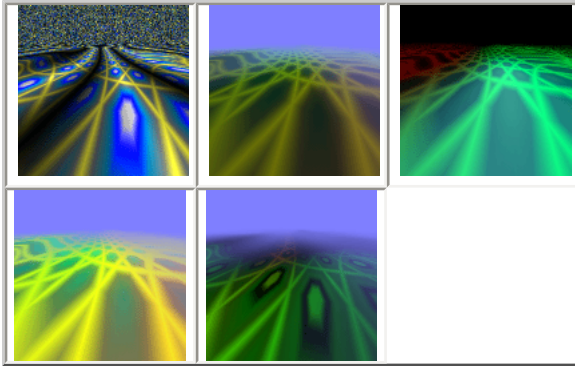
The **Insert** popup menu provides numerous commands for modifying the tree's structure.

Insert Perspective. Choose the **Insert Perspective** command. A group of tiles are added at the top the tree that provide a three-

dimensional perspective effect. Click on the **Perspective Clipped** tile  and adjust with the parameter slider (Perspective has only one adjustable parameter) and notice how the image plane appears to tilt. You will notice 'noise' in the distance at some settings.

Leave the plane tilted so that you can see the horizon. Click on the **Ax + By + C** tile  that feeds the Perspective Clipped tile and experiment with its parameters.

Change shaders. Use the [and] keys to cycle through all the shaders. You will notice that some shaders remove the distance 'noise' completely and others change its character dramatically as shown in the images below. You can remove the distance 'noise' with all color shaders by turning on depth cueing with the [Shading Options](#) popup menu.



3. Insert Global Rotation

Choose **Insert Global Rotation** from the Insert popup menu. ArtMatic Pro adds a Rotate component at the top of the system. If the rotation tile isn't active, click on it. Adjust its Angle parameter (A) and notice how you can rotate the entire system.

Enter an angle directly. Internally, ArtMatic uses a format called radians for angles. It is possible, however, to directly enter the rotation using degrees. Click on the **A** label next to the parameter slider to display the Parameter Envelope dialog. Click on the Angle parameter's numerical display (in the dialog) and type '45' (for 45 degrees) then type 'd'. ArtMatic converts the value from degrees to its internal format.

By animating the Angle parameter, global rotation can be used to animate rotation of the entire system. To do this, add a keyframe. Adjust the Rotation component's Angle parameter, and add a new keyframe, etc.

4. Component Replacement Popup

Make Group. When you option-click the active (selected) tile, the [Component Replacement](#) popup menu appears with options for replacing the tile. The exact options that are available depend on the number of inputs and outputs of the selected component. Click on the tile below the perspective tile to make it active. Option-click the tile to pop up the Component Replacement popup menu, and chose **Make Group** which replaces the tile with a group of tiles. Option-click the tile below the perspective tile (it will be a 2-in/2-out tile). Choose the **Make 3D Group** Click on any of the new tiles and use the left/right arrow keys to see how changing components affects the image.

Changing the number of inputs or outputs. Click on any tile to make it active. Option-click the tile to pop up the Component Replacement popup menu. Note that there are commands that allow you to replace the tile with another that has a different number of inputs and/or outputs. The exact options depend on the active tile. To change both the number of inputs and outputs, you will have to access this menu twice (once to change the number of inputs and once to change the number of outputs).

5. Inserting New Tiles

Click on any tile in the system to make it active. You can add a tile before or after the selected tile by clicking on the Insert Before or Insert After tools or by choosing the Insert Before or Insert After commands from the Insert popup menu.

Moving On

Obviously, these introductory tutorials have barely scratched the surface of what you can do with ArtMatic Pro. If you want to learn more about ArtMatic, here are a few suggestions:

- **Explore the example files** - we have supplied a large number of ArtMatic files which demonstrate a wide range of possibilities. Notice that each file has a number of keyframes. We recommend that you play the keyframe animation for each file by clicking on the **Animate Keyframes** button (or typing command-h to see a high-quality mini-preview).
 - **Tip:** A quick way to peruse the examples is to choose **Open** from the File menu and navigate the dialog to a folder that contains ArtMatic files. Rather than opening each file, use the arrow keys of your keyboard to visit different files. When a file is selected, you will see a thumbnail of its first keyframe.
- **Explore the documentation** - The rest of the documentation has full descriptions of ArtMatic Pro's features and lots of useful tips. Before continuing with the intermediate level tutorials, you should read the [Intro](#), [Overview](#) and [Concepts](#) chapters of the manual and browse the Component Reference chapters. We also recommend reading the [User Interface](#) chapter in order to familiarize yourself with all of ArtMatic's tools.
- **Perform the advanced tutorials** - the [QuickStart III](#) and [Getting Deeper](#) chapters of the manual provide valuable techniques and insights that will help you deepen your mastery and understanding of ArtMatic Pro.

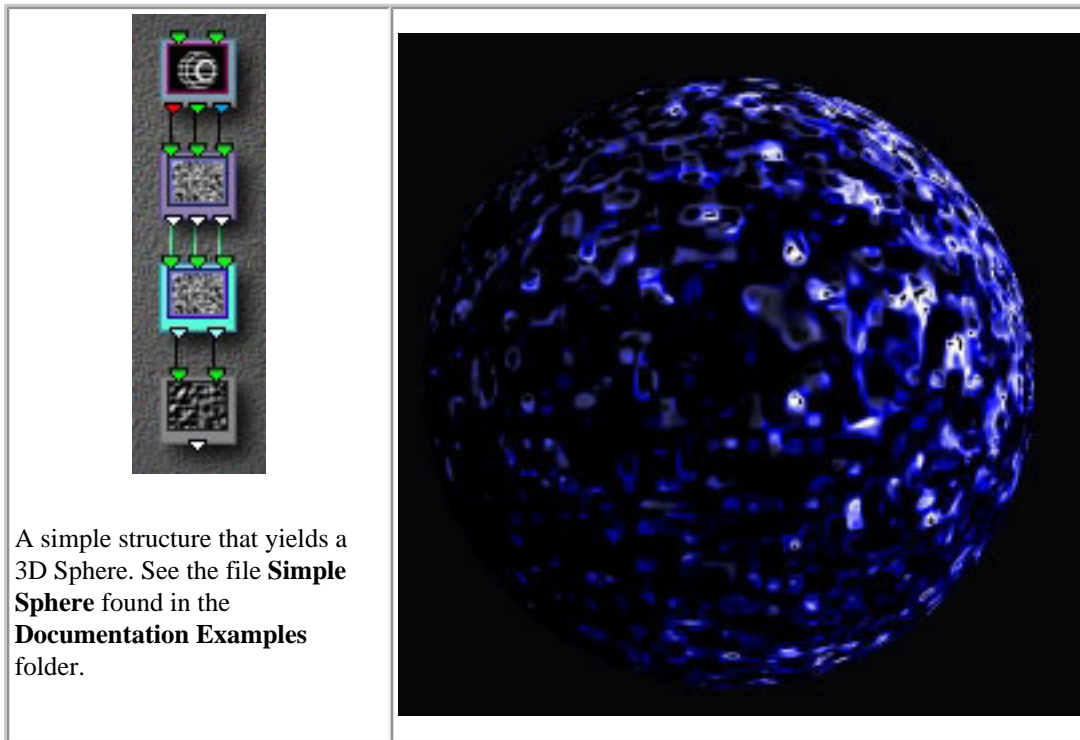
QuickStart III - Intermediate Level

Features

This tutorial chapter covers a number of useful techniques and tricks and provides an introduction to some advanced techniques. While not required, familiarity with the [Overview](#), [What's New](#), [User Interface](#), [Concepts](#), [Shaders](#) and [Component Reference](#) chapters will help you to get the most out of these simple lessons.

This chapter provides more background than the earlier tutorials but is still focused on giving you a tour of ArtMatic's features and giving you a sense of what ArtMatic Pro can accomplish. There is one additional tutorial chapter called Getting Deeper which provides both useful techniques and an introduction to concepts and methods that will benefit all artists seeking to get the most out of ArtMatic Pro.

Lesson 1 - 3D Object Basics



A simple structure that yields a 3D Sphere. See the file **Simple Sphere** found in the **Documentation Examples** folder.

A simple exploration of 3D surfaces

In this lesson, we will explore some of the properties of ArtMatic's 3D objects. The image shown above was produced by the file "Simple Sphere" which is found in the **Documentation Examples** folder that accompanies this manual. Open the **Simple Sphere** example file. The structure is simple: a **Sphere** component followed by several noise components which eventually reduce the three outputs to one so that ArtMatic can use the active gradient to color the surface. These noise components provide the sphere's surface.

Click on the **Sphere** component's tile. Adjust its three parameter sliders and notice how it moves in relation to the texture which covers it. When the sphere moves, it doesn't take the texture with it but moves within the texture's plane. It is as if you have a silk cloth with a textured pattern stretched on a frame before you and a small sphere in your hand. If you take the sphere, reach around the cloth and press the sphere so that its outline is visible through the cloth and slide the sphere, the sphere will be moving under the texture rather than taking it with it. Most of ArtMatic's 3D objects have this relationship to their shading/texture. There are a few 3D

objects (whose names include the word **parametric**) which behave differently and take their textures with them.

The texture can be animated to create a sense of motion. Click on either of the noise components and mouse over its parameter sliders to find the component's phase parameter. Adjust the fader and repeat for the other noise components. Click on the **Add** button to add a keyframe. Continue adjusting the phases and adding keyframes. Now watch the animation by pressing the **Animate Keyframes** button (or typing command-h to see a high-quality mini-preview). Observe how the surface moves though the sphere stays in place. With some experimentation, you will find that you can simulate planetary revolution by adjusting the noise components' phase parameters.

Infinity and backgrounds

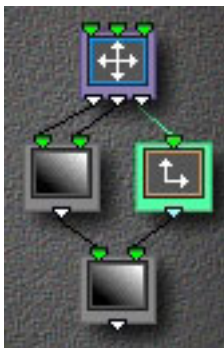
3D object components generate the value **infinity** for areas beyond object boundaries. In gradient-color systems, infinity is drawn with the rightmost color of the active gradient. In RGB (three output) systems, infinity is drawn with the first auxiliary color (the depth cueing color) and is transparent to RGB mixing components. In more complex systems with multiple branches it is possible to have the solid appear against a background provided by a picture, movie, or an ArtMatic subtree. The Documentation Examples folder provides several files for exploring these possibilities: **Cube & Pict** in which a 3D cube moves against a static background picture when animated and **Cube & Compiled** in which the background is synthesized by ArtMatic. Such systems can yield stunning animation.

There are a couple of features to note in these more complex examples. Both files make use of a particular 2-in/3-out component whose purpose is to take a non-RGB ArtMatic subtree and generate an RGB representation so that the branch can be mixed with a color picture or another subtree that generates RGB true color. There are a couple of 2-in/3-out components that perform this function. For more details see the [Component Reference](#) chapters of this manual. The other key feature that both files have in common is that they use the **Packed RGB Crossfade** component to mix the background and foreground. This component can be used to mix the output of two branches into a single image. This component has just one parameter **Interpolate** which mixes the two images that feed it. This component however treats **infinity** specially. Wherever it sees infinity, the image is treated as transparent. In our example files, the left-hand image in the mix is the background and the right-hand image is the sphere or cube in the foreground. With interpolate set to its maximum value, the sphere or cube is opaque **except** for the areas of infinity which surround it.

Lesson 2 - Time

When a system has three inputs at the top OR has an unconnected third input in its interior, the third input is a counter or time value. Time can be used to manipulate a system even if all its parameters stay constant.

A simple example. In the folder "Doc. Example Files" is a file called "Simple Time Explorer" which contains a simple system with 3-inputs at the top.



The topmost component simply scales the three incoming values (x co-ordinate, y co-ordinate, and time). The amount of scaling is provided by the A, B and C parameter sliders. For this example, no scaling is done. The x and y co-ordinates are passed unchanged to the **Ax+By+C** component which simply generates a tilted plane. In this case, the parameters have been sent to send out 0 for all points (essentially a plane with no tilt). The last component is another **Ax+By+C** component. There is one other component in the tree. It is the scale component which is connected to the rightmost (the z/time output) of the topmost component and whose output feeds the 'y input' of the final component in the system.

Click on the file's keyframe and click the **Add** button which creates a second identical keyframe. Now click the **Animate Keyframes** button. Note that the result is a gradual color change over time. In previous, versions of ArtMatic, animating between identical keyframes would also yield a static result. You can simulate ArtMatic's old behavior, by clicking on the scale component's tile and clicking Parameter A's lock icon then dragging the slider to 0. This causes the z-value to be a constant 0. If you now animate the keyframes, nothing will appear to happen. Notice that you can change the response to time by changing the 1-in/1-out component. For example, you can use the **Random** component to randomize the color change.



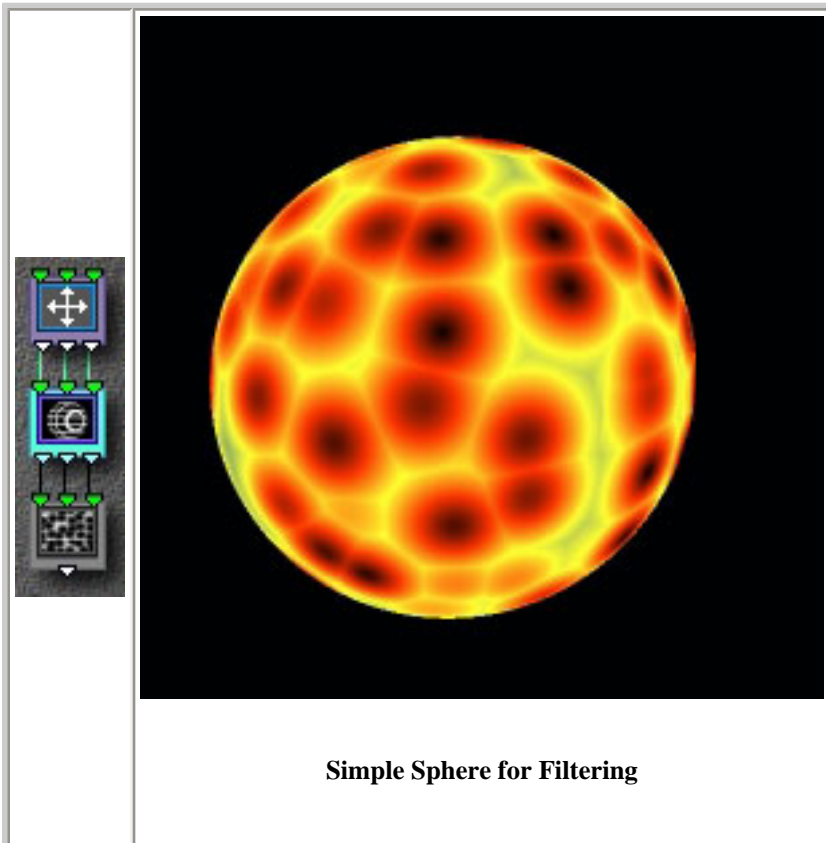
Important notes about 'time'. This input has a few side effects which may seem surprising. **First**, it is important to keep in mind that the third global input is fed into open inputs that are in the middle of the tree (as in the picture at left); hence, time can influence trees that have only two inputs at the top. **Second**, the counter's value increments with the passage of time **even if it flows through locked tiles.** (Parameter locking only prevents a tile's parameters from changing. It has no influence on the values that flow through the component.) **Third**, when you add a keyframe in a system that makes use of the third global input, the image may change when the keyframe is added. This happens because, while you are editing, the third input has the time value of the most recently viewed keyframe (or the value it had when animation was stopped). ArtMatic is only able to calculate the correct time value when the keyframe is actually added.

Examples. Many of the example files we have provided make use of this time input. Several examples are found in the folder **Doc. Example Files:Time Examples.** You can also search the main example library for "time" to find more examples.

Lesson 3 - Inserting Filters

When building complex trees or experimenting with mutations or random path animation, you may find it helpful to insert filter components to restrict the range or modify the values being fed into some but not all of a component's inputs (or outputs). This technique has many applications and is especially powerful when used in combination with the mutation dialog or the randomizing dice.

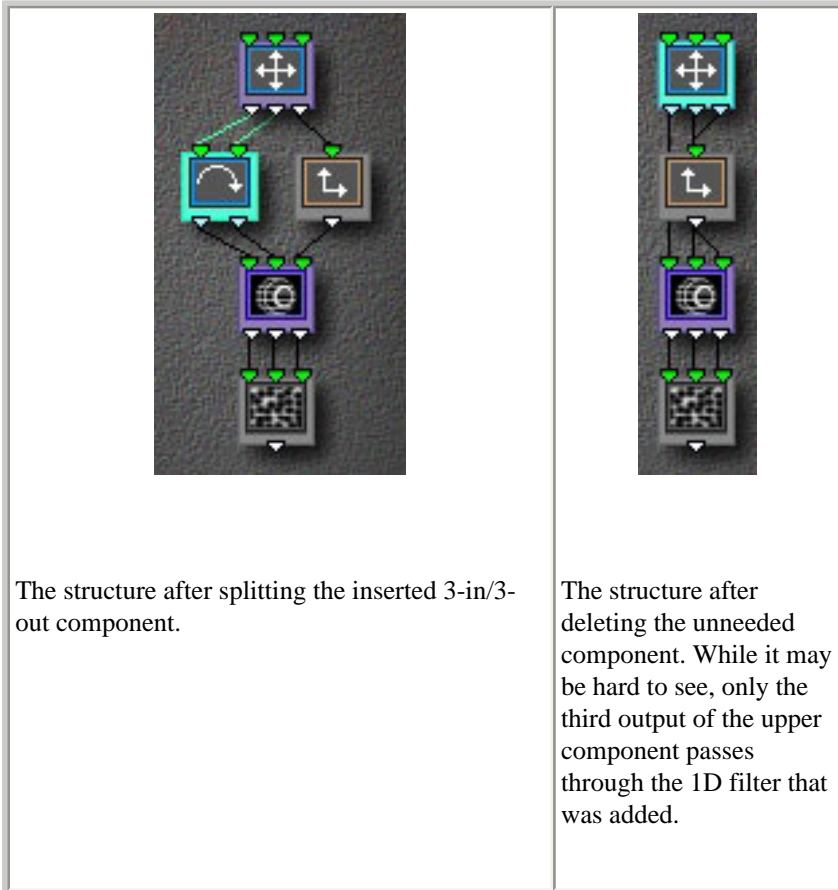
The system shown below is a 3D sphere system with three inputs at the top. As noted elsewhere, the third global input is a counter (or time value, if you will) which increments with each frame of an animation. Even if no parameters are changed, the passage of time during the animation will cause the sphere's size and texture to change.



In this structure, we can modify the counter values by inserting a 1D filter on the thread through which they flow. This structure is

found in a file called **Simple Sphere for Filtering** in the **Documentation Examples:QuickStart Files** folder. Add a few identical keyframes by clicking the **Add** button a few times in succession. Click the **Animate Keyframes** button and notice that the sphere continually recedes. The rate at which it recedes is influenced by the **Scale Z** parameter (the third) of the **Scale** component at the top of the system.

To add a filter, click on the Sphere component then click the **Insert Before** tool in the Structures area. A 3-in/3out component is inserted. Click on the inserted tile to select it. Option-click the tile and choose **Split Component** in the menu which pops up (this command is also found in the **Insert** menu which is found above the structure tree). Two components take the place of the component that was split: a 2-in/2-out tile on the left and a 1D filter on the right. Now, click on the leftmost of the new tiles and click the **Delete** tool in the Structures area to delete it.



There is now a filter on the z-output passed from the top of the system into the sphere component. Click on the 1D filter select it and click again to pop up the component selector. Choose **Sin x** from the menu. Animate the keyframes and notice that the sphere's distance now oscillates rather than continually recedes because the constantly increasing z-value output by the system is being fed through a sine function. Now, choose the **Random** function for the 1D filter and animate the keyframes. When exploring using the **mutations dialog** or **random path animation**, it is often useful to insert filters and to lock their parameters to restrict the values that flow through the tree.

Lesson 4 - Compiled Trees

This is very simple introduction to compiled trees, a very powerful feature introduced in ArtMatic Pro 2.5. The [Compiled Trees & Iteration](#) chapter provides additional information and exercises that we recommend performing.

In the documentation examples folder, find the file called **Compiled Source**. (It is in a folder called **Compiled Examples**.) Open the file. Play its keyframes to become familiar with them, and choose the **Export Compiled** command with a name such as **compiled**

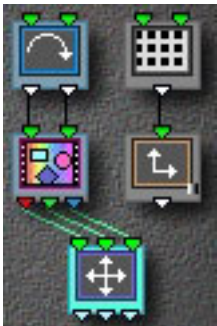
source.cp. Open the file **simple compiled example**. Click on the upper-left tile and choose the **Open Compiled Tree** component and choose the exported tree created in the previous step. Observe the complex animation that now results from a tree with just three tiles.

Now, create a new tree structure by choosing one from the Choose Structure popup menu or by control-clicking the large die. When you find a tree with a 2-in/1-out tile in it, explore what happens when you use Open Compiled Tree to replace it with the compiled tree we just created.

This is a great way to save favorite textures and shapes and incorporate them into new files. The ArtMatic examples library contains a large number of examples that demonstrate the power of this new feature. In the **Documentation Examples** folder, the file **planet with compiled bg** is one example of how compiled trees can be used to make complex textures easily available to your projects.

WARNING! Compiled trees have an Iteration or Recursion parameter which is locked by default. This parameter causes the tile to be 'looped' and, thus, can tie up even a fast processor. We recommend not adjusting this parameter until you have read the [Compiled Trees & Iteration](#) chapter which covers recursion and iteration in some detail.


Lesson 5 - Shading



To facilitate exploration of the new shading options, we have provide a simple system, **New Shader Explorer**, whose structure is shown at left. It is located in the **Doc. Examples Files:QuickStart Files** folder and is a convenient system for exploring the new shading options.

Open the file. Notice that the structure has a branch that is not connected to the final output. Components that are not connected to the final output allow you to provide shading and depth that is computed independently of the main image.

Notice that the righthand column's last component has a little shading glyph in the lower-right corner. This indicates that the component provides shading (shadow and lighting information) for the system.

Explore Global Shading. Pop up the Shading Options menu by clicking on its icon . Turn Global Shading Off and notice the change. Now, choose **Fog and Shade Automatic** from the Shading Options popup and turn Global Shading On. The image looks different than when you first opened it because ArtMatic is no longer using the Grid component's branch to provide the shadows. Click on the last component in the righthand column to select it. Type 's' (a shortcut for choosing Component Sets Shade from the Shading Options popup). The Grid branch is now providing global shading for the system. Adjust the parameters of the branch's components and note their effect. Where the branch generates low values, the image is dark. Where the branch generates large values, the image is bright.

Explore Depth Cueing. Turn Global Shading Off. Click on the last component in the righthand column to select it. Type 'd' (a shortcut for choosing Component Sets Depth from the Shading Options popup). Pop up the Shading Options popup and notice that **Depth Cueing Small** is on. The component you selected is now responsible for creating a depth fog effect. Change the depth cueing amount using the Shading Options Popup. The first auxiliary color is used to provide the fog. Change the color and see what happens. Explore different parameter settings on the depth cueing branch and explore what happens when you change the actual components used in the branch. Explore what happens when you also assign this branch to set the global shading (by clicking on a tile and typing 's').

To let ArtMatic choose the components that control global shading and depth, choose **Fog and Shade Automatic** from the Shading Options popup.

Another example: The **Planet and Stars** example file makes excellent use of a disconnected component. See [Getting Deeper](#) chapter for more information.

For detailed information about ArtMatic's shading functions, see the [Shaders](#) chapter of this manual.

Further Exploration

Hopefully, these tutorials have provided material that you can use to start exploring on your own. To get the most out of ArtMatic (though certainly not required), we highly recommend reading the [Getting Deeper](#) chapter of this manual which provides both useful techniques and a methodology that many users have found invaluable for learning to control ArtMatic.

We also can't emphasize enough the value of both exploring on your own **and** examining the files provided in the main examples library.

Getting Deeper: Lessons Towards a Deeper Understanding

ArtMatic Pro has come a long way from its beginning as a "Zen" tool for generating fascinating images. While ArtMatic Pro can still be used in the same Zen way as its predecessors, it has become a very powerful graphics synthesizer that can be guided and controlled by the artist. As ArtMatic's capabilities have increased and as our library of examples and artwork has grown, it has become more and more clear to more and more enthusiasts that ArtMatic can not only randomly generate striking images but that one can exercise a great deal of control and actually create art by design rather than chance. As people have come to this realization, there has been an increasing number of requests for a guide to designing images and animation.

This section of the documentation will help you develop your ArtMatic chops by providing some tips and techniques for methodically exploring ArtMatic. The aim will be to learn how to explore and analyze existing ArtMatic structures and images. Such explorations will help you develop the intuitions required to manipulate ArtMatic, design images and animations, and allow you to play an active role in the creation of ArtMatic art.

EXAMPLE FILES: The example files mentioned in this chapter are found in the **Getting Deeper Chapter** folder of the **Doc. Example Files** folder which is found in the same folder as the documentation.

Methods for Exploration

One of ArtMatic Pro's great strengths is that it allows many types of exploration which can lead to rewarding works. It is possible to create beautiful art with either very limited or very detailed understanding of ArtMatic's underlying principles. There are many different strategies which can be applied to creating works with ArtMatic. Below is a description of some strategies that have proved very productive. They are by no means mutually exclusive and are often used in combination.

The '**Zen**' strategy -- in which one mutates the system at random and zooms in and out to discover the wondrous images that can be found -- requires no understanding of the underlying principles, but even the most knowledgeable artists will find such exploration liberating and a source of raw material which can be further manipulated and molded. ArtMatic's 'imagespace' is so large that there are many worlds that will only be discovered through application of this type of exploration.

The '**template**' strategy is an effective strategy for both creating new works and for developing intuitions and deeper understanding. The template strategy is a method where you choose an existing ArtMatic file as a departure point for exploration/manipulation. The starting point could be one of the many example files that we have provided or a discovery of your own. If you are still developing your intuitions and understanding, you may spend a lot of time exploring the effect that particular parameter changes have and seeing how changes to the components used by particular tiles affects the system. As you become more advanced in your understanding, you will be able to use a system as a starting point and know what components to add or change to achieve the desired effect. This strategy is frequently used by even the most advanced users.

The '**strict construction**' strategy requires the greatest understanding. It is the strategy by which you construct a system from scratch to achieve a particular effect and requires a solid understanding of ArtMatic's principle's and components.

Finally, there is the '**modified Zen**' technique in which selective mutation and randomization is used explore the richness of a particular system that has been created with any of the preceding strategies. Beginners will find this technique an interesting way to discover new systems and to develop their intuitions.

- Start by locking all of a system's functions and parameters by shift-clicking the lock icon any unlocked parameter.
- Next, selectively unlock parameters. Only unlocked parameters will be influenced by mutations and randomization.
- Mutate the system to see how those mutations affect the system. Use any of these methods to mutate the system: random

path animation, the **Mutations Dialog**, and the randomizing dice. The Mutations Dialog is invaluable for such exploration. Locking all of a tile's parameters makes the tile immune from being changed in the **Mutations** dialog. Therefore, you can turn on the **Mutate Function Type** option in the Mutations dialog and see what effect mutations of an individual function or small number of functions has.

To successfully apply any of these strategies (other than the simple 'Zen' strategy), it is important to develop a feel for the underlying structure of ArtMatic systems. The remainder of this chapter is intended to help you develop a knack for finding the deep structure of ArtMatic systems. By performing the exercises provided in this chapter, you will learn the roles that the various types of components play, how to analyze ArtMatic systems, and a wealth of helpful techniques.

The Mysterious World of Complex Dynamical Systems

A comprehensive guide to image creation is beyond the scope of this manual for a number of reasons. While many simple ArtMatic systems produce easily predictable results, it is easy to create systems whose results have varying degrees of unpredictability. You can easily create or discover systems that will surprise even ArtMatic masters because one can create trees which implement complex mathematical or geometrical systems that have inherent unpredictability. One of the joys and--to some--frustrations of **complex dynamical systems** (the technical name for many ArtMatic systems) is that a system may inherently defy predictability due to its extreme lack of linearity. With such systems, it may not be possible to precisely predict how a system will respond to different input values and parameter settings. Oftentimes, the degree of unpredictability is only apparent when zooming far into or far out from a system or when modifying its settings. There is a lot of wonder to be discovered by chance. The trick is knowing how to harness those discoveries and mold them. It isn't necessary to understand the underlying mathematics to have great success with ArtMatic.

To precisely design the wide (nearly infinite) range of possible images from scratch would (even if it were possible) require a fairly deep understanding of a pretty tricky (and fairly new) branch of mathematics. This level of deterministic control is actually impossible because many systems defy predictability--and even credulity. Consider this: when Benoit Mandelbrot first ran a program to plot the equation for which he is best known (and which has captured the imagination of so many), he believed that the fuzzy irregular graph being plotted was the result of a bug in the computer hardware. Entire books have been devoted to exploring the Mandelbrot set and the images it can create, and the Mandelbrot Set is just a single ArtMatic component. Imagine how hard it would be to pre-design the kind of images generated by a combination of tiles each of which was as unpredictable as the Mandelbrot Set!

Suggested Reading. While not directly-related to ArtMatic, there are a few non-technical books which may be of interest to those who want to understand more about the kinds of mathematics and systems which underly ArtMatic. If nothing else, these books are entertaining and may give you a sense of why complex systems behave as they do. **Chaos** by James Gleick is a great introduction to chaos theory and its history and quite engagingly told. **Turbulent Mirror** by Briggs and Peat is an entertaining book which explores chaos theory with lots of helpful analogies. **Complexity** by M. Mitchell Waldrop picks up where "Chaos" leaves off. **Computers, Pattern, Chaos, and Beauty** by Clifford Pickover describes numerous equations and their visual pattern equivalents.

Exploring ArtMatic - Letter from Eric

One of the best ways to deepen your understanding is to explore favorite systems and examples. You can quickly develop a sense for the contributions made by the components within those systems, and -- as you explore more and more systems -- you will develop a sense of how to manipulate a system to create different images and effects. As you learn, you will find that you will be able to take elements from one system and use them in systems of your own design.

Ultimately, the key to creating ArtMatic art is to combine discovery, chance and design.

Before we dive into some techniques, a few words from ArtMatic's creator, Eric Wenger, may prove enlightening. In response to a letter from a user who was frustrated by not being able to better predict how ArtMatic systems behaved, Eric wrote:

If it can cheer you up, I don't understand ArtMatic either. I mean that as soon as several operators are connected and interact on each other the mathematical equation quickly becomes way to complex to be of any use.

But it does not matter.

ArtMatic lets me INTERACT and experiment with the system. I don't have to fully understand why it does this or that. I still can play with it and learn how it behaves. So, even if I don't quite understand the system mathematically, I can develop a FEEL for it. And now that I have some experience, I can more quickly find or create a structure and settings that yield interesting results.

The order of the system's transforms is important. You get very different results by inverting things. Complex system behavior can be mastered empirically.

What seems most helpful to me is to have a broad but clear understanding of the different classes or categories of transformations and how to interpret the inputs and outputs.

Understanding How Components Function in the Tree

When designing or trying to understand ArtMatic systems, it is generally helpful to understand the different ways that components can function in the tree. Where a function falls in the tree will influence how it behaves. A couple of categorization methods can help you to understand the structure and functioning of a particular tree. The rest of this chapter assumes that you have performed the [QuickStart](#) tutorials, are familiar with the basic concepts discussed in the [Concepts](#) chapter of this manual--especially the section which describes how images are calculated. It may also be useful to spend some time browsing the [Component Reference](#) chapter(s) to get a sense of the available functions. Detailed descriptions of all the available components is provided in the [Component Reference](#) section of the manual.

A note to users of ArtMatic Pro 2.0. If you haven't performed the tutorials in this version of the manual, we recommend that you at least read through them. They have been updated to include some new techniques, features and concepts that are likely to be of benefit.

Suggestion to all readers. We recommend that you print out this chapter to facilitate performing the tutorials. Many users have found it quite helpful to print out the **Component Reference** chapters as well; since you are likely to want to refer to them more and more as your understanding grows.

The Schema

When evaluating how a tree is put together, it is often helpful to identify the abstract structure of the tree. Much of this chapter is devoted to this topic since it is critical to the design and understanding of complex ArtMatic systems. We will use the term **Schema** to refer to the abstract structure (or metastructure) of ArtMatic systems. By using **Schema**, we can avoid having to use structure to refer to both the structure tree and the abstract structure.

Component Functional Categories

To understand a system, it is often helpful to identify the role that a particular tile plays in the tree. The functional categorization of components makes this task easier. It is important to keep in mind that many components can play multiple roles--the precise functional role being determined by the component's placement in the tree and its connections. You can use components in ways that resist categorization or defy their original intent and discover astounding new images and effects in the process. In order to get oriented, though, it is useful to think of components as falling into a few simple categories (even though the categorization is a bit of a fiction).

The basic functional categories are:

- **Space distortion functions** - these components distort and remap a system's geometry and generally have the same number of inputs and outputs. They take a 2-D or 3-D space and output a remapping of that space. The most basic space distortion functions are the **Scale** and **Rotate** functions which scale and rotate the incoming space respectively. The **Twirl** function is a more advanced component which distorts an incoming normal Euclidean space by warping it into a whirlpool, and there are even more complex remappings that defy description with words. You can think of these components as functions that remap the points of the incoming space to new locations. Typically, these components are the first ones in a system. They establish the geometry in which the subsequent surface, texture, space or object is drawn.
- **Surface/Texture generators/Shaders** - These components map a point in space to a particular altitude/color They take multiple input values (usually space co-ordinates) and generate a single output value. (Note: in the case of RGB-based shading functions, three outputs are the equivalent of a single output value since RGB requires three values -- red, green and blue -- to describe a single color.) These components provide texture for the incoming space or objects. The **Ax+By+C** (plane), **Ripples**, and **Grid** components are perhaps the most obvious surface generators. Technically, most ArtMatic images are surfaces though they might not appear as such. Most components that fall into this group may also act as mixers, and the particular context will determine whether the component is acting as a shader or a mixer.
- **Mixers** - these components mix the outputs of multiple branches/inputs (multiple surfaces, if you will) into a single composite surface or image. Many surface/texture generators can also function as mixers.
- **Filters** - these components will have the same number of inputs and outputs and modify or remap the incoming values. All of the 1D (one input/one output) functions are filters as are some multi input and output functions. For example, the 1D sine function will remap incoming values so that the output is restricted to values between -1 and 1; as a result, a steadily increasing input value results in cycling output values. If a sine filter is used at the output stage, the result is an undulating surface. By contrast, applying a random filter will result in the output value varying from the input value by a random amount.
- **Space translation functions** - these functions either take a 2-D space and generate a true 3-D object or take a true 3-D surface and generate a 2-D projection of the surface (such as when one draws a cube--a 3-D object--on paper--a 2-D representation). This category is really a special kind of space distortion function.
- **Colorspace functions** - These functions generally have 3 inputs and 3 outputs and translate the incoming points/pixels between different color representations such as RGB to HLS (and the reverse).
- **Pack** - This component enables the 3 values of an RGB component or a 3D component to be packed into a single packed value. Special functions that have 'packed' in their names can act on packed values (usually to mix them). This mechanism allows RGB/3D values to be passed through a single thread rather than requiring three threads.

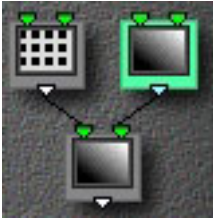

Component Categories by Connections

As noted earlier, the role played by a particular component in the tree is influenced by its placement in the tree and by the values that flow into it. The same component (function) can play very different roles given different inputs or placement in the tree. For example, the $Ax + By + C$ 2D scalar component generates a tilted plane when fed by the raw co-ordinates of ArtMatic's canvas (a 2D space) but acts as a mixer if its two inputs are fed by two independent single output components. The number of a tile's inputs and outputs determines the possible functional role played by the component. This section outlines the possible functional roles played by the different tile types. Note that there are several possible interpretations for some tile types. The **input/output interpretations** column indicates in shorthand what type of data is generated for a particular type of input. It indicates that when that input type is fed to the tile, the output is of the indicated type. For example, if an entry reads "**input**: single value | **output**: single value" it can be translated as "when the input is a single value, the output is a single value." Keep in mind that these are interpretations of a system--attempts at describing a mathematical system with words. Ultimately, the tree is simply a chain of mathematical equations.

The information in this table is not necessary to be productive with ArtMatic, but those that want to deepen their mastery and those that want to be able to analyze complex examples will find the information beneficial. This table (together with the detailed component descriptions found in the [Component Reference](#) chapters) will help you determine the role that a particular tile/component plays in a particular system.

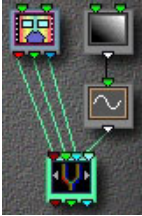

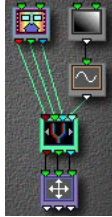

Note: Don't be intimidated if some of the cases don't make sense to you. Understanding every possible usage of the components is not necessary to develop a mastery of ArtMatic Pro. Some of the possible manipulations that can be done are extremely sophisticated and are pretty difficult to make sense of unless one has a thorough grounding in mathematics; you can skip over these cases **or** create simple systems that have those types of components and explore on your own. You will

often get a better feel for what is going on by exploring in this way than you will by reading a word description of what is going on.

Tile type	Input/output interpretations	Notes
1D (1-in/1-out)	input: single value output: single value [Filter]	<p>Main interpretation - filter: These components all act as filters whose function is to alter the range and behavior of the incoming values.</p> <p>Examples: Using a sine wave on a linear input causes the output to cycle. Apply a sine wave to a time input to generate cyclic behavior during animation.</p>
2D Scalar (2-in/1-out)	<p>input: 2-D space output: a value (a surface) [surface generator/shader]</p> <p>input: two independent values output: a value (a surface) [mixer]</p>	<p>Main interpretation - surface generator: Generally, these functions generate a surface by calculating a surface altitude from the input. In turn, the surface altitude determines the color when the system is a gradient-based (rather than RGB) system. In mathematical terms: $z = f(x,y)$.</p> <p>Secondary interpretation - mixer: When the inputs come from independent outputs (such as the outputs of parallel 2-in/1-out functions), the result is a mixing of the input surfaces.</p> <p>Some functions make equal sense in both interpretations. For example the component $Ax + By + C$ will generate an arbitrarily oriented plane when the inputs are those from a space, in which case, the parameter sliders will determine the orientation and offset of the plane. The function can also mix two independent surfaces to yield a single composite surface. In this case, the parameter sliders will determine the relative mix of the inputs.</p> <div style="display: flex; align-items: flex-start;"> <div style="flex: 1;">  </div> <div style="flex: 2;"> <p>Note that the system at the left has three 2D vector components. The upper-two demonstrate the main interpretation. The Grid component at the left generates a raised grid while the $Ax + By + C$ component to its right generates a tilted plane. The second, lower $Ax + By + C$ component mixes the two surfaces to create a superimposition of a grid on a tilted plane.</p> </div> <div style="flex: 0.5;">  </div> </div>

<p>2D Vector (2-in/2-out)</p>	<p>input: 2D Space output: 2D Space [space distortion]</p> <p>input: two independent values output: two independent values [filter]</p>	<p>Interpretation #1 - space distortion function: modifies (remaps) the incoming spatial system. Some distortions are quite radical (such as the complex number functions which remap infinity to zero and zero to infinity) while others are simple warpings of the space. The scaling and rotation functions are examples of the simplest space distortion functions. Space distortion functions tend to use both input values (x and y) to calculate both of the output values (i.e. both of the inputs influence both of the output values)</p> <p>Interpretation #2: Some 2D Vector components are the equivalent of two parallel 1D filters. In this case, the input x and y input values influence only their respective outputs. For example, Scale Ax + B, Ay + C scales the left and right inputs independently and can be used in place of two parallel 1D filters. Mirror 4 is another such function and behaves the same as two parallel absolute value functions so that x-out = ABS(x) and y-out = ABS(y).</p>
<p>2-in/3-out</p>	<p>input: 2D Space output: 3D Space [space translation/distortion]</p> <p>input: 2D Space output: RGB Color [RGB shader]</p>	<p>Interpretation #1: create a co-ordinate in 3D space. Components such as Sphere create 3D objects from an incoming 2D space while components such as 3D z set simply provide a third spatial co-ordinate for a two-output function so that it can be fed into a three-input function.</p> <p>Interpretation #2: Associate an RGB color with a point. For example, the RGB Pict/Movie maps a color picture to the incoming space. This is the RGB equivalent of a 2-in/1-out component.</p>
<p>3-in/1-out</p>	<p>[#1] input: 3D space output: a value (surface) [surface/texture generator]</p> <p>[#2] input: 2D space, 1D value output: a value [surface/texture generator]</p> <p>[#3] input: 2D space, time output: a value [surface/texture generator]</p> <p>[#4] input: three 1D values output: a value [mixer]</p> <p>[#5] input: RGB color output: single packed value that can be passed only to functions that accept packed input. [pack]</p>	<p>Interpret. #1 - surface/texture generator: Maps a value in 3-D space to a particular value. The effect is to provide the texture/shading for the 3-D object.</p> <div data-bbox="789 1121 1010 1346" data-label="Image"> </div> <p>Interpret. #2: When used in this fashion, the component creates a surface from the first two inputs (the 2D space) which is modulated by the third input. This type of use can yield very complex results. To see this, create a system such as the one at left and experiment with the last component's parameter sliders. In mathematical terms, this is a function $z = f(x,y)$ where the third input acts as a parameter within the equation.</p> <p>Interpret. #3 - surface generator: If the left two inputs are connected to a 2-output function and the third input is open (unconnected) then time will modulate the surface calculated from the incoming 2D space.</p> <p>Interpret. #4 - mixer: In this case, the component mixes three independent surfaces into a single composite surface just as the 2D Scalar components can mix two surfaces. This is easiest to see with the Ax + By + Cz component.</p> <p>Interpret. #5 - pack: This is a special case which applies only to the pack component. It merges 3 independent values into a single packed value that can be understood by components with the word 'packed' in their name.</p>

<p>3-in/2-out</p>	<p>input: 3D space output: 2D space [space translation]</p> <p>input: 2D space , 1D value (or time) output: 2D space [space distortion]</p> <p>input: three 1D values output: two 1D values</p>	<p>Main interpretation : Maps a point in 3D space onto a 2D plane. This is very similar to interpretation #1 of 3-in/1-out tiles. Oftentimes, the component provides the surface detail of the object though it may also have the effect of warping or projecting from a 3D space onto 2D space.</p> <p>Interpret. #2: In this case, the component acts as a space distortion function where the third input modulates the distortion. For example the z Rotate function uses the third input to provide the amount of rotation. Most of the 3-in/2-out components with z in their names perform this role.</p>
<p>3-in/3-out</p>	<p>input: 3D space output: 3D space</p> <p>[#2] input: colorspace output: colorspace</p> <p>[#3] input: 3D space output: RGB/HLS space</p> <p>[#4] input: three 1D values output: 3D space.</p> <p>[#5] input : three 1D values output: RGB /HLS color</p> <p>[#6] input: three 1D values output: three 1D values</p>	<p>Main interpretation: The primary interpretation tends to be 3D space transform functions which remap/modify the incoming 3D space. Some 3-in/3-out components are used to create 3D objects such as spheres, planes and tubes.</p> <p>Interp. #2: Convert RGB (red, green, blue) values to HLS (hue, luminance, saturation) values and vice versa. Sophisticated color manipulation is possible with this type of use. For example, you can put the RGB/HLS component after a color pict and follow it with a series of components that end with the HLS/RGB component. Several examples using this component are provided in the examples library--the examples all have "HLS" somewhere in their name.</p> <p>Interpr. #3: Pick a color for an incoming 3D point. Typically, the component will be one such as 3D Color Techno Noise which can provide a color texture for a 3D surface. This is the RGB equivalent of a 3-in/1-out 3D shading function.</p> <p>Interp. #4: The 3-in/3-out component can generate a 3D space from three independent components.</p> <p>Interp. #5: RGB or HLS colors can be synthesized from three independent values. This usage makes sophisticated color manipulation possible.</p> <p>Interp. #6: Some components act as 3 parallel 1D fileters. For example, 3D Scale and 3D Offset can be used to scale or offset the x, y, and z values independently.</p> <p>Note: As noted elsewhere, if the last component in the system has three outputs, then the shading algorithm is bypassed and direct RGB output is used.</p>

<p>4-in/2-out</p>	<p>input: two 2D spaces output: 2D space</p> <p>input: 4D space output: 2D space.</p> <p>input: 3D Space & a 1D value output: 2D space</p>	<p>Main interpretation: Create a 2D space by combining two 2D spaces such that:</p> $X_{out} = F(x1,x2) \text{ and } Y_{out} = F(y1,y2) \text{ (where the inputs are } x1,y1,x2,y2)$ <p>Interp. #2: Project a 4D space on to a 2D space.</p> <p>Interp. #3: In this case the behavior is a bit like the 3-in/2-out 'z' components where the fourth input modulates the projection of 3D space into 2D space.</p>
<p>4-in/3-out</p>	<p>input: RGB space + ArtMatic output color output: RGB space</p> <p>input: RGB space + 1D value output: RGB space</p>	<p>All of the 4-in/3-out components were designed to mix true color (RGB) pixels with "normal" ArtMatic gradient-based color. In all cases, the three leftmost inputs are intended to be fed from a component that generates RGB color and the output is an RGB image.</p> <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;">  <p>Main interpretation: This applies when the component is the last one on the tree as shown to the left. It acts as a mixer that mixes an RGB image with a gradient-based image. The rightmost input is processed with the current gradient and shading algorithm and mixed with the RGB image that feeds the three leftmost inputs. The result is an image such as the one shown to the right.</p> </div> <div style="text-align: center;">  </div> </div> <div style="display: flex; justify-content: space-around; align-items: flex-start; margin-top: 20px;"> <div style="text-align: center;">  <p>Interpretation #2 This interpretation applies whenever the component is not the last one in the system. In this case, the rightmost value is treated as a value that is added or multiplied with the RGB values BUT it is not processed with with the system's gradient OR shading algorithm. For example, if you were to append the 3D Scale component at the end of the structure shown above to create the structure shown at left, the result would be the image shown at right. In this case, the values coming into the rightmost input modify the luminance of the pixels in the RGB image.</p> </div> <div style="text-align: center;">  </div> </div>

Identifying the Structure of the System

When exploring a new system, it is a good idea to make note of the **schema**, its general abstract structure. The schema is a coarse outline of the tree that breaks it down into functional units. You will frequently find that very complex trees have very simple schemas with many tiles working together in functional groups. To a degree, all systems share the following "classic" schema: **geometry creation** (space transforms and distortions) followed by **surface/texture generation** followed by **shading/color manipulation**. Each of these schematic units might be quite complex, and, because tiles can be connected to multiple inputs and outputs, it can be difficult to isolate these abstract structural units. Even when it is difficult to discern the schema with precision, the attempt to identify it will help you gain insight into the tree's deeper structure. It should be pointed out that in a very complex tree, you might find individual branches of a tree that are schematically complete (i.e. they have this basic structure).

It probably bears repeating that the component categories are useful fictions that can help you get your mind around what is happening in a system. Components are really just mathematical functions that simply act on the numbers that pass through them. ArtMatic doesn't care whether the inputs make sense or whether they are spatial co-ordinates, RGB color values, etc. At a purely theoretical level the distinctions between the categories can be pretty fluid--and in some complex highly interconnected systems the structure may defy schematic analysis.

A word to the wise: don't become discouraged if your head starts to spin and a complex structure seems too much to analyze. Some structures simply defy detailed understanding. When this happens, let go of the need to understand and analyze, and let yourself explore the mysterious beautiful world the system creates.

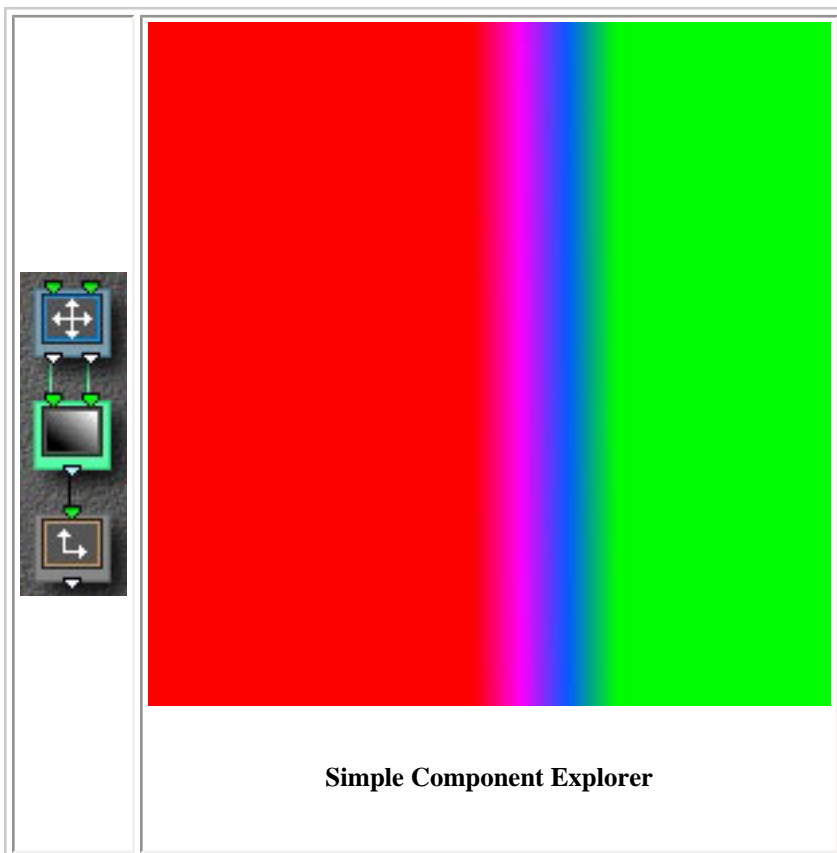
Tip: When exploring, you can use **compiled trees** to organize groups of components into functional units to create very complex space distortion groups or surface generators and then use them as parts of simpler trees. This technique will help you refine your sense of tree organization.

EXPLORING COMPONENT CATEGORIES

The following sections will familiarize you with simple cases of the different component categories in preparation for dissecting and understanding more complex structures.

EXPLORING A SIMPLE STRUCTURE

To get started, open the file **Simple Component Explorer**.



In this simple system there is a **space distortion** function (**Scale** $Ax+B$, $Ay+C$), followed by a **surface generator** ($Ax+By+C$) followed by a **filter** function which can be used to adjust the final output. As you explore other systems, you will notice that this basic schema is repeated over and over again though generally in more complex incarnations where a group of components provides the geometry followed by a group that generates a surface and finished off by the components that filter and shade the output. The gradient used by this system has red at the left and proceeds to pink to blue to green as values increase. In this simple case, the surface is a simple tilted plane whose altitude increases as a function of x . (Since we have set the y -scale of the second component to 0, the altitude increases as a linear function of x -- as x increases so does the altitude. The second component's B parameter can be set to a value other than 0 to allow the y -values to influence the system).

Space distortion functions. The first component of this system is the **Scale** component. Depending on the context, this component can actually act as either a space distortion (as it does in this example) or as two parallel one-dimensional filters. In this case, **Scale** remaps the incoming space (the plain old Cartesian plane) by multiplying the incoming x and y co-

ordinates by parameter A and adding an offset. Select the component's tile and explore the effect adjustments of parameters A (the scaling factor), B (the horizontal offset) and C (the vertical offset) have on the image. The image scaling is the result of the distortion (in this case, the scaling) of the system's underlying geometry.

Click on the **Scale** component's tile and hold down the mouse button to popup the **Component** popup menu. Choose **Rotate A**. This function remaps the incoming space by rotating and offsetting it. Experiment with adjustments to the parameter sliders. Now, change the component to the **Skew** function and experiment with the slides. All of the changes to the image that you see are the result of distorting the geometry that underlies the surface generated by the middle component.

An exercise: Very complex geometries can be created by using several spatial distortion functions in series. Explore this technique by choosing **Append Vector** from the **Insert** menu. Start with simple distortion combinations such as **Rotate** and **Scale** and then explore the other available 2D Vector components.

Another exercise: Explore what happens with some of the compiled trees that are provided in the main examples folder. To do this, choose the **Open Compiled Tree** component and then choose any 2-in/2-out [compiled tree](#). There is a folder in the main examples folder which contains a number of compiled trees. After choosing a compiled tree, you can take a look at its internal structure by typing **e**. You can return to the main structure by typing **e** again. NOTE: When choosing a compiled tree, ArtMatic will only let you choose a compiled tree with the same number of inputs and outputs as the the base tile.

Some of the space distortion functions (**Ripples** or **Radial Star**, for example) may appear to generate a surface. If this seems confusing, it may be useful to keep in mind that when you create a non-Euclidean space even simple surfaces such as a plane take on the characteristics of the new spatial system when projected back into our mundane Euclidean universe.

Surface/texture generators. These components generate an altitude (a z co-ordinate) from a point on a plane [in mathematical terms: $z=f(x,y)$]. Re-open the example file to restore it to its original settings. In the example file, the **Ax+By+C** component (which generates a simple tilted plane) is the surface generator. When the space is undistorted, the result is a simple inclined plane whose altitude increases as the x (horizontal) values rise.(Note: the gradient used in this example has red for low values and green for high altitudes.) Many of the 2D scalar functions are surface generators (some are mixers--and others can be either). Some of the surface generators (such as **Ax+By+C** or **Grid**) are great for creating large-scale surfaces such as planes and grids while others (like the various noise functions) are more generally used to create surface detail or textures.

Click on the **Ax+By+C** component to select it and then click again to popup the **Components** menu. Choose **Bubbles**. Zoom in on the image with the **Zoom** tool and explore what happens when you adjust the component's parameters. This component generates a bubbling surface from a simple plane. It may be easiest to see the surface by choosing a black and white gradient (or other gradient that is a smooth transition from black to another color). Choosing the **Derivative** component for the final filter component will make this even clearer. (As noted in the **Concepts** chapter, this trick doesn't work for functions like **Facet** whose output is discontinuous.)

An exercise: Choose the **Grid** function as the surface generator and choose the **Ax + B** function again as the final component. The Grid component is a great one to use when exploring because the interactions between it and any space distortion are quite easy to see. Explore what happens when you change the first component in the system to distort the space in which the grid is created. Explore many space distortions. When changing distortions, click on the **Random Path Animation** button. Random Path Animation is a great way to quickly explore the possibilities of the new system. Also, be sure to experiment with different gradients and zoom levels.

Note: The movie/pict components are sneaky functions that technically act as surface generators though that might not be intuitive.

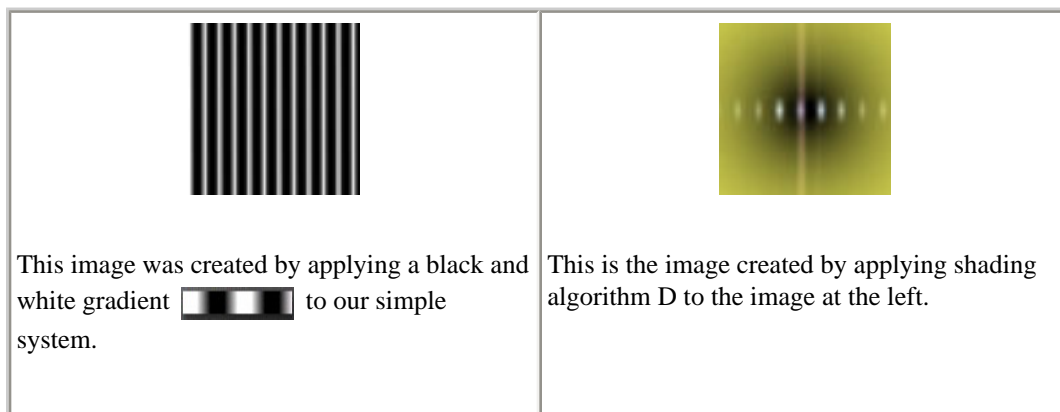
A note about RGB color. Gradient-color systems are surface-based. (I.e., the system generates a surface whose altitude determines the color.) In RGB color systems, there isn't a single scalar output value. Rather, the color is provided by shading/texture components that take spatial co-ordinates as input and generate RGB color to paint the corresponding pixels.

Filters. The final component in this system is a 1D filter, the **Ax + B** filter, which simply multiplies the input by a scaling factor and adds an offset. Notice that adjusting the component's offset parameter shifts the colors left or right. This is because the colors are mapped to altitude, and shifting the offset has the effect of adding a constant number to the altitude (which in turn shifts the colors used to draw it). Now, choose the **Sin X** component. The colors now cycle from red to pink across the canvas because a sin function only generates values from -1 to 1 no matter how high the input value.

You can use several filters in a row. Click the **Append** tool to add a filter after the **sin** filter. Make sure that the new filter is the **Ax + B** filter. Now, adjust the **Ax + B** component's offset parameter and notice that when it is set to the minimum value the entire image is red (that is because when you subtract a large enough number from the output of the sin function--whose output is -1 to 1 -- all the values are falling within the range mapped to the color red. Experiment by changing the function used by this tile and seeing how the change affects the image. Finally, use the **delete** tool to remove the component you just added.

Click on the **sin** function tile -- if it isn't already selected -- and adjust the frequency parameter and notice how the number of bands changes. Notice how phase shifts the bands to the left and the right and how amplitude changes relate to the range of colors output by the system.

An exercise: Spend some time manipulating the gradient used to draw the image both by creating your own and by choosing each of the gradients available in the gradient popup. Now, explore each of the shading algorithms available from the [Shaders](#) popup menu. Be sure to play with the auxiliary colors available for the complex shaders. The images below were generated from this system by simply changing the gradient and shading algorithm.



Mixers

Unlike the simple structure we used in the previous section, most ArtMatic systems will not consist of a single simple branch. A structure may have long parallel branches or branches that have sub-branches or even interconnected branches. In all these cases, there will be some components that mix the outputs of multiple branches or components. The example shown below (called **Simple Mixer** and found in the **Doc. Examples Folder**) is quite simple: a space transform (the **Rotation** component) with two branches (the **Grid** component on the left and the **Ripples** component on the right) that are mixed by the **Ax + By + Z** component. The coarse structure is a space transform followed by a surface generator. In this case the surface generation portion of the tree is made up of three tiles: the **Grid** and **Ripples** components and the final component in the tree which mixes the **Grid** and **Ripples** surfaces into a single composite surface with the simplest mixer in ArtMatic's arsenal, **Ax + By + Z**.

<p>The structure</p>	<p>When the Scale Y parameter is 0, only the left (X) input is passed out of the mixer.</p>	<p>When the Scale X parameter is 0, only the right (Y) input is passed out of the mixer.</p>	<p>When the scale values are non-zero for both inputs, the output is a mix of the two.</p>

When this component is connected to two independent inputs, this component mixes the left and right inputs into single surface. The contribution made by the inputs is controlled by parameter sliders A (Scale X) and B (Scale Y). When the Scale parameter is 0 the corresponding input has no influence on the output. Parameter C adds an offset to the resulting surface which causes the colors to be taken from further left on the gradient or further right, depending on the offset direction.

Exercise 1: Explore what happens when you change the component used as the mixer. While most of the 2D Scalar (2-in/1-out) functions were not designed as mixers, any can act as a mixer. The table below shows just a few examples created simply by changing the mixing component (and adjusting its parameters) and zooming in or out on the picture.

<p>Image created when using a Grid component as the mixer.</p>	<p>Image created using the Line component as the mixer.</p>	<p>Image created by using the Crystal Noise component as the mixer</p>



Exercise 2: Explore what happens when the two branches are disconnected at the top and use different space transforms. Once you have modified the structure, assign different transforms at the top of each branch and adjust all the parameters. When exploring the implications of such changes, it is generally useful to use the **Mutations** dialog. Make sure that Mutate Function Type and Mutate Colors are **off** otherwise it won't be clear what changes are caused by the parameter changes and which are caused by function or color reassignment.

Hints: Use the $Ax + By + Z$ component as the mixer so that it will be easier to see what is going on. To disconnect the branches at the top, click on the Rotate tile and delete it (using the Delete tool); then, for each branch, click on the uppermost tile and click on the **Prepend** tool which will insert a 2D Vector (2-in/2-out) component at the top of the branch as shown at left.

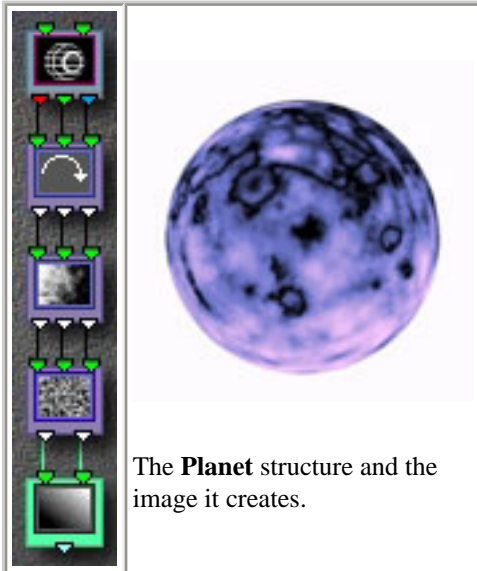
Space translation functions

Space translation functions translate between spatial systems with different numbers of dimensions. They are used to create true three-dimensional objects from 2D co-ordinates or to create a two dimensional projection of a 3D space/object onto a 2D space. This last case is what one does when drawing a sphere or cube on paper. The most frequently encountered components are the 2-in/3-out components such as **Sphere** that generate a 3D object from 2D co-ordinates.

Note: Not all components that have two inputs and three outputs function as space translation functions. For instance, the **packed RGB** components are very special functions that are used to mix RGB inputs into a single RGB-based image. The three outputs of RGB components do not correspond to three-spatial co-ordinates, together they represent the color a single pixel (point) in the image. An RGB 'triplet' is equivalent to a single value in a standard ArtMatic gradient' based image as both of them determine the color used to draw a single point. So, these RGB 2-in/3-out functions are equivalent to the 2-in/1-out of a gradient-color-based system.

Space translation components will generally be found towards the top of the tree in the group of components that defines the system's geometry. Any time that 3D objects or space are used, there will need to be a component that translates those three dimensions to either two dimensions (which will ultimately be reduced to the single value--or RGB triplet--used to determine the color used to draw the point) or to a single value (which can be a simple value or an RGB triplet). All functions which translate from higher to lower dimensions may also act as surface generators (or, in another way of thinking, the dimension reduction is an act of surface generation) which provide the texture that covers the surface.

Open the **Planet** example file found in **Doc. Examples Folder**. Notice that all but the final component manipulate space and so can be considered part of the geometry unit. The topmost **Sphere** component creates a true 3D sphere from the 2D co-ordinates fed into the system. The **Rotation** component that follows it can be used to rotate the sphere in true 3-D space. Notice that when you adjust the rotation component's parameters, the planet appears to rotate on its axis.



The **Planet** structure and the image it creates.

The two components that follow rotation are noise components. The first is a 3-in/3-out noise function which provides random distortions of the incoming co-ordinates. The next to last component is the second **space translation** component **Random 3D Noise** which both maps the 3D co-ordinates to 2D space and introduces additional distortion. As with all noise components, parameter A is amplitude and can be used to adjust the amount of distortion. So, it can be used to simply map from 3D to 2D space by setting the amplitude parameter to 0.

3D Noise Note. Both of these noise component are true 3D noise functions which means that as you move the sphere closer and further (by adjusting its z/depth parameter) the distortion varies. Click on the **Sphere** component and adjust parameter C which moves the sphere along the z axis and note how the texture changes as the sphere moves through space. 3D noise functions can be used to great effect when you want to create liquid, flame, cloud and space effects with real depth.

Exercise: Set the amplitude of both noise functions to 0 and notice that the texture disappears. Now, play with the amplitude settings and see what happens when only one

component's amplitude is set to 0. Try this with both noise components to see how each contributes to the texture.

The final component is the now familiar **Ax + By + Z** component which generates the final surface. If you are theoretically inclined, you can think of the planet that you see as really being a simple plane drawn in a very irregular non-Euclidean space created by the functions higher up in the tree.

Exercise: Explore what happens when you change the component used by the last tile.

Exercise: Replace the last tile with a 2-in/3-out tile and choose the **RGB Pict Movie** component. Set the amplitude of both noise functions to 0. Now, explore what happens when you adjust the amplitude of the noise. Even though, the RGB Pict

Movie has three outputs, this is not a space translation because the three outputs are really a single RGB value--it just so happens that a single RGB value is made up of three numbers the red, green, and blue values used to color a single point.

Note: An important feature of ArtMatic's 3D object components (sphere, cube, room, and tube) is that they return a special value, infinity, for all points beyond the object's boundary. When the image is drawn, the depth cueing color (even if depth cueing itself is off) is used to color any points that have the value infinity.

Colorspace Translation

As noted elsewhere in the documentation, ArtMatic Pro 2.5 has two basic color models that are used to determine the colors used to synthesize an image: gradient-color and RGB "true color". **Gradient-color** color maps a system's single output value to a color found in the active gradient (gradation); the method for mapping values to gradient colors is determined by the active shading algorithm chosen in the **Shaders** popup-menu. **RGB-based** color is used for all systems that terminate with three outputs. The color used to draw a point is determined by the three values generated by the tree which determine the blending of red, green and blue for that point. The choice in the **Shaders** popup menu does not influence the resulting image. Both RGB and gradient-color systems are influenced by the **shading options** (depth cueing and global shading). In addition to these two color models (or colorspaces), there is a third color model **HLS** (hue, luminance, saturation) which can be used inside a tree to manipulate color. The **HLS** model allows for color manipulation that is of special interest to video artists.

Gradient color (review). Open the **Simplest Case Color** example file. Gradient-based color is used to color the system because the tree ends with a single output value. If you choose another gradient using the **Choose Colors** popup menu, the colors in the image change since the gradient provides the raw colors used to create the image. Likewise, selecting a different shading algorithm from the **Shaders** popup menu will cause a change in the image.

Simple RGB. Delete the last component in the system (the 1-in/1-out tile). Option-click the last tile in the system (the 2-in/1-out tile) and chose **Replace with vector (3 out)**. Change the component used for the last tile to **RGB Color Shade**. Explore how the component's parameters influence the color (a detailed description of the component is found in the **Components** reference section of this manual). You will note that changing the active gradation has no influence on the image.

Gradient color-to-RGB. Gradient-based color is capable of creating astounding images and was the sole color model in earlier versions of ArtMatic. There are many cases, however, where you will want to perform RGB or HLS processing of a gradient-based image, or you may want to mix RGB and gradient-based branches of a tree. The **RGB Main Gradient** component (and the related **RGB Shaded Gradient** component) provides this translation. In the structure created in the previous paragraph ("Simple RGB"), choose **RGB Main Gradient** as the final component. Observe that the component uses the colors of the active gradient. Choose **Save As** from the **File** menu to save the current structure with the name "**Simple RGB Explorer**".

Manipulating RGB. Select the tree's last component (if it isn't already selected) and click the **Append** tool to add a 3-in/3-out component. Choose the **3D Scale** component for the new tile. Parameters A, B, and C now provide independent scaling of the red, green and blue values of the image. Adjust the parameters and note how you can manipulate the colors. Choose **Save** to save the file.

Exercise: Explore what happens when you use a component such as **3D Rotate** as the last parameter. This allows you to rotate the colorspace which can produce very nice effects. Any of the 3-in/3-out functions will now manipulate the colors of the image rather than its spatial characteristics.

Exercise: Open the "**color matrix multiply**" example file and read the accompanying notes. This example shows how you can use matrix multiplication to perform very powerful color manipulation and transforms. Create a 3-in/3-out compiled tree that performs the matrix multiplication, and use it as the last stage of RGB-based files.




RGB/HLS translation. If it isn't already open, open the file "**Simple RGB Explorer**" that you created earlier in this lesson. If it isn't already selected, click on the last component and append two more 3-in/3-out tiles to the structure. Choose the **RGB to HLS** component for the third tile and the **HLS to RGB** component as the last. Your structure should look similar to the one shown at left. The component(s) between the RGB/HLS translation components allow you to manipulate the image in the **HLS** (hue, luminance, saturation) colorspace. We encourage you to explore the wonderful color effects that can be created by manipulating the image in the HLS colorspace. We have provided many example systems that perform HLS manipulation. Search for 'HLS' to find the examples.

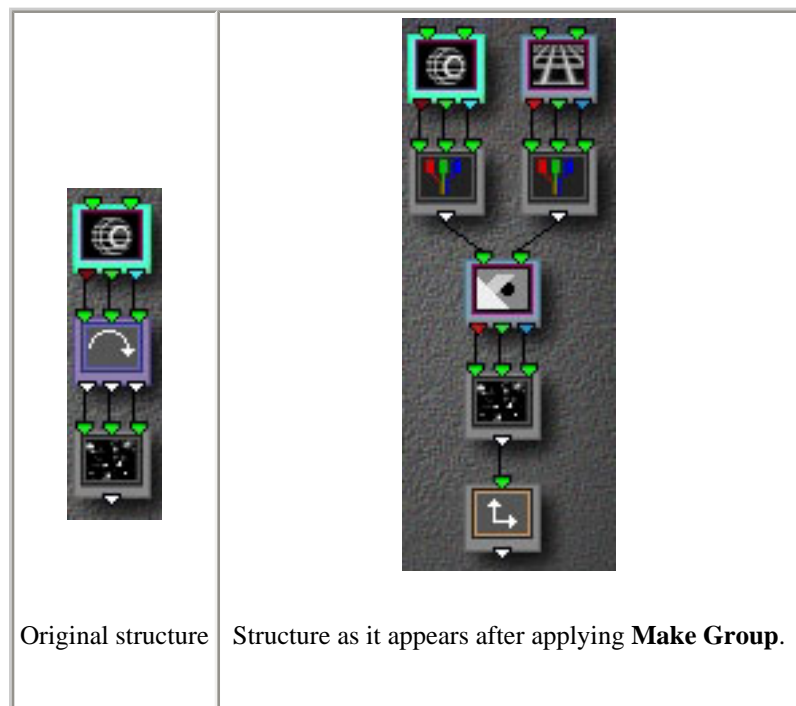
Mixing RGB and Gradient color. A number of **mixer** components have been provided to allow you to mix RGB-based branches and gradient-based branches of the structure tree. Most of the components that perform this function are 4-in/3-out components. Choose **New** from the File menu to create a new ArtMatic document. Choose the **RGB and Artmatic Shading** item from the **Structures** popup menu. The left branch of the tree is the RGB-based branch and the right branch is a gradient-based branch.

Exercise: Explore the different ways that the branches can be combined by changing the component assigned to the last tile. Note that only the components whose names start with **RGB** provide this sort of mixing of RGB and gradient-based color. With these components you can achieve effects as varied as simple mixing to alpha channel masking.

Pack/packed functions

ArtMatic tiles are limited to four input threads. As a result, it is necessary to pack outputs in cases where you want to mix 3D objects or RGB images/branches. (Otherwise, the tiles would require six or more inputs.) The parameter-less 3-in/1-out Pack (x,y,z) component performs this function. The single output of the Pack component is not a single value but a 'group' or 'list' (sometimes called a packed stream) that contains three values that can be passed to some functions through a single thread. Any function that has packed in its name needs to receive its input from a pack component. These functions allow you to mix RGB images or 3D objects.

Pack with 3D objects. Choose **New** from the File menu to create a new ArtMatic document. From the **Structures** popup menu choose **3D Sphere**, and click on the **sphere** component if it isn't already selected. Click on the **Make Group**  tool to create a parallel 3D object branch. Note that Make Group adds whatever elements are needed to mix the new component with the original selected component. When creating a 3D group, ArtMatic adds a **pack** component after each 3D component since the mixing of 3-out streams is always done with **packed** functions.



Some of the packed functions were designed primarily for mixing 3D objects and others for mixing RGB color streams. Click on the component that mixes the two **pack** components to select it. Click again to pop up the **component** popup menu and choose **Packed 3D Depth Sort**. This component mixes two packed 3D objects into a single 3D 'stream'.

Exercise: In the structure shown above, both objects are combined before the texture generating components are added. As a result, both objects share the same texture. Add some components to the structure so that the texture of the two objects is not identical.

Pack with RGB branches. **Pack** is also needed to be able to mix RGB images. Choose **RGB 2 Channels** or **RGB 3 Channels** from the **Structures** popup menu to create a structure that has several RGB branches which are packed and mixed together into a single RGB stream. When mixing packed streams, it is important to choose a mixing component appropriate to the type of 'stream' being mixed. See the [Component Reference](#) for detailed information about the available components.

DISSECTING ARTMATIC STRUCTURES

In the section [Identifying the Structure of the System](#) found earlier in this chapter, it was noted that most ArtMatic systems have the following **schema**: geometry components followed by surface/texture generation and completed with final color manipulation/shading. Each section (schematic unit) of a tree may have its own complex structure. For instance, a complex arrangement of components may be used to construct a complex geometry whose surface/texture generation is quite simple. The surface generation section of the tree might create and mix several independent surfaces into a single composite surface. Many trees have several parallel branches. Some branches may be complete substructures that can stand alone and which are joined with other complete branches by a mixing function.

One of the best ways to arrive at an understanding of how ArtMatic systems work is to dissect interesting systems. When examining a structure, start by observing some characteristics of the tree structure and identifying key features. Here are some questions that you can ask about any given system which may guide you to an understanding of the system:

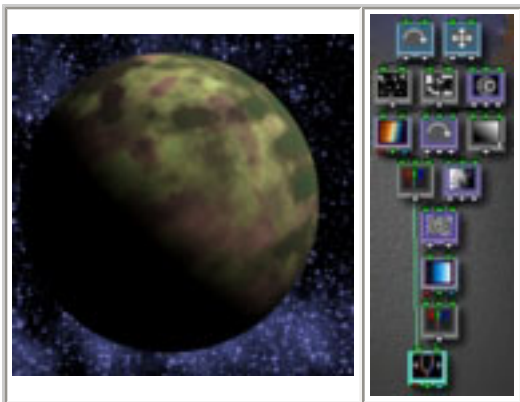
- If there are multiple branches, do they act as complete systems?
- How does each branch contribute to the image?
- How many inputs does the system have? If it has three inputs, observe how the third input (time) is used since it may cause the image to change over time even if all the parameters are locked!
- Are there any branches which are not connected to the final component in the system? If so, are they used for global shading or depth cueing? (If not, they are extraneous.)
- Are there any three-input components whose third input is unconnected?
- Which components are providing the texture for the image?
- Which components create the geometry (spatial system) or 3D objects in the image?
- Are global shading or depth cueing used?
- If shading/depth cueing is on, how does it contribute and what component is tied to it.
- Does the shading algorithm use auxiliary colors which contribute to the image?

In the following lessons, we will learn some techniques to help you answer these questions. As a general rule, a great way to start is by making a copy of the ArtMatic file then removing components (or arrangements of components) to see how their removal affects the image. It can also be useful to replace a complex component with a simple/neutral component whose effect you understand well.

Deconstructing the "Planets & Stars" example file

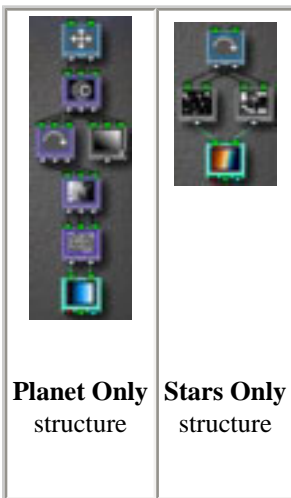
Note: All files referred to in this lesson are found in the folder **Planet & Stars f** which is found in the **Getting Deeper** folder found in the **Doc. Example Files** folder. This folder also includes solutions to some of the exercises.

Open the "**Planet & Stars**" example file whose image and structure are shown at left.



Observe the basic structure. The first thing to note is that the bottom-most component is a mixing component (**Packed RGB Crossfade**) that mixes RGB streams. This fact tells you that there are two branches, each of which generates an image/surface. Click on the last component and move the parameter slider to its far right position. The planet is opaque with the stars in the background. [The stars are visible because the planet is a 3D sphere. As you may recall, 3D objects are surrounded by infinity which is treated as invisible by RGB mixing components. As a result, the stars--which come from the left-hand branch--are visible. You will notice that as you move the parameter slider to the left the planet begins to be transparent.] Move the slider all the way to the left and notice that only the starry field and a shadow of the planet are visible. Hence, the left branch contributes the starry field, and the right branch provides the planet.

Question: Why is the shadow of the planet visible even though the planet itself isn't? If you click and hold the mouse button to pop up the **shading options** menu, you will notice that **global shading** is turned on. Furthermore, you will notice that there is a component in the system [Ax + By + C] which is found at the end of the rightmost branch] which is marked by the shading glyph. Notice that this component's output is not connected to any other component. So, this component (whose input is fed by the sphere) provides shadows (global shading) for the system. We will isolate the shading component later in this lesson.



Isolate the planet. Because the system is fairly crowded, we want to independently examine its two primary branches. One-by-one, click on each component of the left branch and delete it. This is generally easiest to do by working from the bottom back up to the top. Be aware that when you get near the top, the branches of the tree may switch places. Keep in mind that you are eliminating the branch that has the **Rotate** component at the top. When deconstructing trees, it is important to keep an eye out for the position-switching that sometimes occurs.

Once the 'starry field' branch has been eliminated, you will notice that the final two components in the tree are a mixing component that is receiving only one input and a pack component. Delete both of these components since they no longer serve any function. The image you see should be the planet against a blue background. Choose "**Save As**" and save the file with a new name such as "Planet only". We will explore this branch a bit later. The structure should match the one shown in the table to the left.

Isolate the starry field. Re-open the "**Planet & Stars**" example file. Now, eliminate the entire righthand branch by selecting and deleting each of its components. **IMPORTANT NOTE:** As you eliminate

components, the branches of the tree may switch places. Keep in mind that you want to eliminate all of the components in the branch that has the **Scale** and **Sphere** components at the top. At some point in the process, these may move to the left side. You should end up with the structure shown at the left. Recall that a component in the branch we eliminated was providing shading. Pop up the **shader options** menu and turn global shading off. Choose **Save As** and save the file as "**Stars Only**".

Explore the planet

Open the "**Planet only**" file created earlier. Now that we have removed a number of tiles from the original system, you can see that there is a component that isn't connected to the bottom of the system and is marked with the **shading** glyph. This component's only contribution to the system is to create the shadow that obscures the planet's lower left region. Turn global shading off via the **shading options** popup menu to remove the shadow. Since this component is no longer contributing to the system, you can delete it.

Most 3D objects (those that don't have 'parametric' in their names) have an interesting relationship with the texture that covers it. The object moves under the texture rather than taking the texture with it. Click on the **sphere** component and play with its parameters and observe how the texture stays in place while the object moves. Now, ask yourself the questions listed in the **ArtMatic Structure** section that preceded this lesson, and notice that the sphere is a 3-input component with an open third input. As a result, time provides the sphere's z (third spatial) co-ordinate. When the system is animated, time will move the sphere farther and farther away.

Exercise: Modify the system so that time does not move the sphere. The solution to this exercise is found in the file "**Planet No Time**".

Observe the system's schema. The upper three components are space distortion/translation components that create a 3D sphere. The next group of tiles is a series of 3D noise functions which generate the texture which covers the sphere, and the final component is a shading component.

Noise notes. As noted above, the planet's texture is provided by two 3D noise components. While these components are space distortion functions, their influence is limited to the space defined by the sphere, and so they act as shading/texture components. The texture they create is the result of the distortion of the space within the sphere; note that their influence on the system would be very different if they were placed before the sphere. 3D noise functions create very rich textures since they vary in each dimension. Manipulate the sphere's z position (parameter C) and notice how the texture changes and lends dynamism to the planet's atmosphere.

Click on each component in the system in succession and manipulate the component parameters to see how they influence the image. Many components have a 'nominal' setting which minimizes their influence on the system. For example, a scale parameter of 1 removes any scaling; a rotation of 0 removes any rotation. An amplitude of 0 removes the distortion introduced by the noise functions. Examine what happens when you set the amplitude of the two noise functions to 0. Explore how the shading component at the bottom interacts with the noise functions. Experiment with the use of different shading components such **RGB Linear Hues** and **RGB Radial Hues** and different noise functions.

Because the texture varies in all three dimensions, it will look very different if you remove the sphere object (and thus giving you a 2D noise plane). Delete the **sphere** object and notice how different the texture looks, especially as you zoom out. Choose **Undo** to restore the sphere component to the structure.

Explore Global Shade. Choose **Open** from the File menu and re-open the "**Planet only**" file without saving any changes. As noted before, the $Ax + By + C$ component provides the shadow in the image. Explore what happens when you assign other components to provide the shade. There are two methods to try.

- Simply choose another function for use by the tile that is set to provide the shade.
- Choose another tile in the system and type 's' (not command-s). Typing 's' is equivalent to choosing the **Component Sets Shade** command in the **shading options** popup. Try this with each tile in the system.

Explore the stars



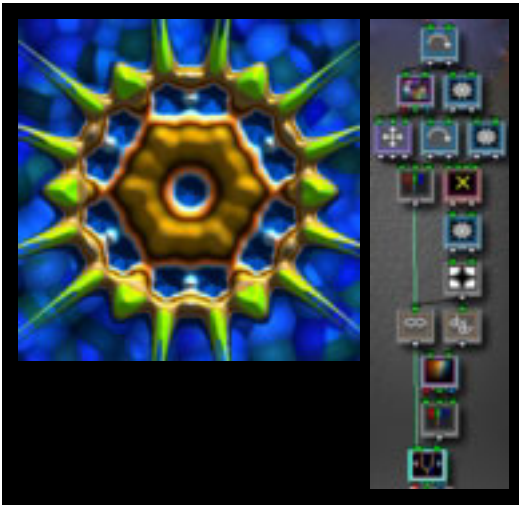
Open the "**Stars Only**" system you isolated earlier. This system is quite a bit simpler than the planet system. Examine its schema. The system has a space distortion function (**Rotate**) at the top which feeds two branches, each made up of a single surface/texture generation component. The final component, **RGB Main Gradient**, is both a shading component and a mixer since it receives input from two independent surfaces rather than from a space. The two inputs are added together then a corresponding color is chosen from the active gradient (the higher the value, the further to the right the color). You will notice that one component provides the "stars" and the other (Fractal Noise) provides the "clouds".

Exercise: Notice how you can change the balance of the clouds and the stars by adjusting the component's two scale parameters. Explore what happens when you choose different components for use at the texture generation stage. Also, explore what happens when you use the 2-in/1-out $Ax + By + C$ component in place of the **RGB Main Gradient** component. Can you find parameter settings for $Ax + By + C$ that creates a similar image to the original? (You may recall that the 2-in/3-out RGB functions are the RGB equivalents of 2-in/1-out functions.)

Tip: Option-drag the sliders to make fine adjustments.

Deconstructing an "Abstract Beast"

Note: All files referred to in this lesson are found in the folder **Abstract Beast f** which is found in the **Getting Deeper** folder found in the **Doc. Example Files** folder. This folder also includes solutions to some of the exercises. At left is a keyframe from and the structure of a file called (for lack of a better name) **Abstract Beast**.



In the previous lesson, we analyzed the system by breaking it down and isolating its parts. In this lesson, you will perform the analysis and breakdown on your own without step-by-step instructions since the process is will be very similar.

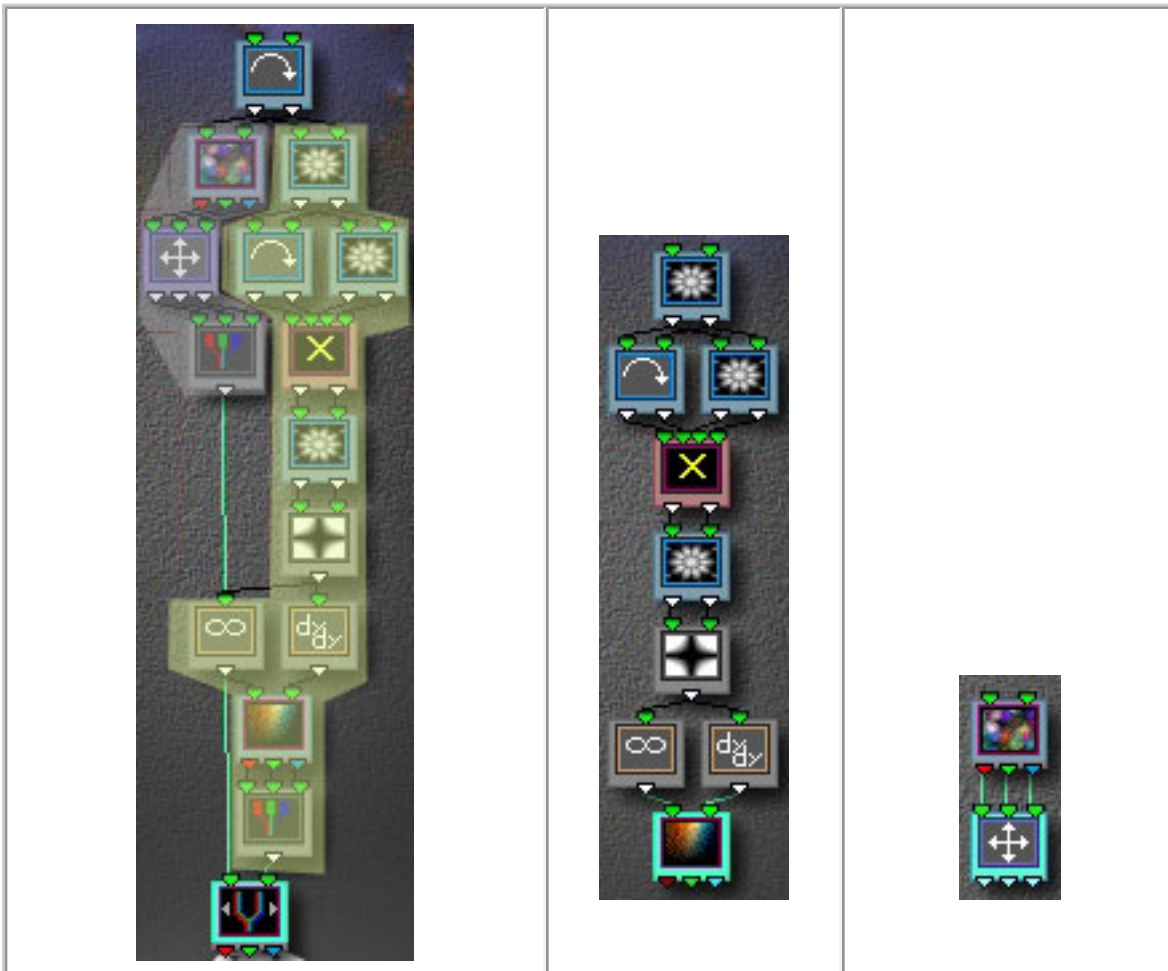
Exercise: Open the file and see if you can identify the system's schema. Remember to start by seeing if the tree behaves more like a single interconnected structure or a mixture of independent systems. If it is a mixture of independent systems, isolate them and save each to its own file.

Perform the exercise before proceeding. A discussion of the solution follows below.

NOTE: When deconstructing a system that contains pack components, there may be some surprises if the pack components are removed before removing the components they feed.

Solution: As in the previous example, this system is composed of two independent systems which are mixed at the bottom of the tree. There was a slight trick. There is a rotation component at the top of the tree which is attached to both branches. Oftentimes, a **Rotate** or **Scale** component or a group of tiles that act as a space distortion group are found at the top of a tree and feed two or more branches that are otherwise complete systems of their own. (Shared Rotate and Scale components at the top of the system facilitate global rotation and/or zooming when animating a system.) When the transform is a simple one (such as scaling or rotation), you can delete it from the tree before deconstructing it. By the same token, there might be a series of tiles which follow the mixer which provide color manipulation or shading; it is often helpful to remove these tiles when deconstructing a tree. Even better, isolate these tile groups and export them as compiled trees.

The table below shows the stages of the deconstruction.



The **Abstract Beast** structure with its two contributing branches highlighted. Highlighted in gray is the branch which provides the background.


The structure isolated from the righthand branch which creates 'the beast'.

The structure that isolated from the lefthand branch which provides the image background.

There isn't much to say about the structure that provides the background. It is simply a 3D color texture followed by a scaling component that allows for some adjustment of the background's colors. Observe how you can manipulate the background's color balance by adjusting the 3D Scale component's parameters.

Exercise #2: Examine the structure that is responsible for the 'beast' and see if you can identify its schema.

Commentary: The structure's upper 4 rows of tiles (all of which are 2D Vector -- 2-in/2-out -- components) create a very complex geometry while the lower three rows provide the shading. Most of the image's overall character/contour is created by the space distortion while the details and lighting/3D effects are created by the shading group at the end.

Explore the geometry. In a system such as this one whose component parts are somewhat complex, it is useful to explore the tree's parts on their own. When you explore a complex space, it is helpful to simplify the color mapping as the surface and space will be easier to discern than when a complex gradient is used. To proceed with the lesson, create a file that has only the 'beast' branch of the tree and a simple black-to-white gradient  for all its keyframes. (You can use the file **Beast Only** file as a starting point if you didn't create one during the previous exercise.) Save the file as **Beast Only B/W**. (See the file **Beast Only B/W** in the examples folder if you have any trouble). When you simplify a system in this fashion, high altitudes of the surface will be white and the furthest depths black. Since we are using a simple gradation between black and white, the contour of the space is easy to see.

Tip: To change the gradient used by all keyframes, hold down the shift key and then select the gradient from the gradient popup.

Since the geometry (the space) is pretty complex, it is useful to remove the group of tiles that provides the surface/shading. Keep in mind that there will generally be some sort of surface/shading (or mixing) component at the bottom. For exploring a complex space, it is a good idea to use the simplest surface/shading component there is which is $Ax + By + C$. Create a file that has just the space transforms followed by $Ax + By + C$. (See the file **Beast Only b/w #2** in the examples folder if you have any trouble).

Notice that, while the image's details have changed quite a bit, the overall character and contour are the same. In some systems, the space transforms (the geometry) will dominate the tree and provide most of the character; in some systems, the surface generator/shader stage will provide the character, and in yet other files, the image will be a complex interaction of the stages. When analyzing a tree, deconstructing it helps you to determine which aspects of the system's character come from which parts of the tree. When you later want to create something with that general character you can use such sub-structures as starting points.

Tip: When you find powerful substructures, save them as compiled trees so that you can easily use them as building blocks when creating new systems.

Find zero. When space becomes distorted, it is important to notice where the space has zero and where it has large values. The black-to-white gradient we are using for this exploration makes it especially easy to see these contours. Black areas correspond to 0, and the whiter the point, the higher its value. The choice of final shader, of course, determines how the x and y values will be combined to calculate the altitude. The $Ax + By + C$ component we are using generates 0 only when both values are zero (if parameter C is 0). You will notice that the spatial system we have created approaches 0 the further you are from the normal space's origin (the center of the canvas). Also note that the space has a complex symmetry derived from the **Branch N** component which is used several times in the tree.

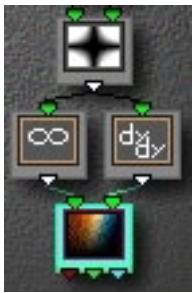
Exercise #3: Now that you have gained some expertise in exploring and analyzing systems, it is time to explore the space transforms and get a feel for how they interact. Below is a list of techniques, you should apply to your exploration:

- Pick a keyframe to use as an exploration base.
- Go through each tile in the system and manipulate its parameters to see how it responds within the system. Click on your base keyframe to return the parameters to their original settings before moving on to the next component.
- Examine what happens when you change the component used by a particular tile. In particular, see what happens if you replace the rotation component with a more complex one OR if you replace any complex component with a simpler one.
- Explore what happens when you change the component used by the 4-in/2-out tile. Note that some of 4-in/2-out components were intended to combine space systems (**Add** and **Complex Multiply** are two examples). Focus your attention on these choices when exploring this tile. More information is provided in the **Components Reference** chapters.


Exercise #4: In order to explore the shading components used in the system, it will be useful to create a system that contains the 'beast' space transformations in a compiled tile followed by the shading components. To do this, create a tree that has only the Beast space transforms and export it as a compiled tree; then create a new file that uses this compiled tree followed by the shading group. (We have provided the file **Beast Shader Explorer** in case you get stuck or want a glimpse of our solution.)

Hint: To create the final tree, create a tree with just the shading group, click on the top tile and use the **prepend with** tool to add a 2-in/2-out tile at the top then choose Open Compiled to import compiled tree.

Explore the shading.



The **Abstract Beast** system creates images with dramatic 3D lighting effects. The lighting and 3D texture are created by the shading part of the tree. The type of lighting and surface details found in this example are typical of systems that use the **2D Derivative** derivative-based component.

Open the file you created in exercise four (or the provided **Beast Shader Explorer**). Change the gradient to the black-to-white gradient  to make the surface altitudes more obvious (black is 0 and large values are white). As noted earlier, the example file's spatial system tends to zero the further one travels from the origin. At left is the group of tiles that provides the color and shading for the "Beast" system. The first tile, the **Multiply** function (which multiplies the inputs together to calculate the output value) provides the overall shading contour. Since it multiplies the inputs together, 0 is generated wherever either input is 0.

Explore the primary shader. The **Multiply** function could itself be used as the sole shader for the system. It provides the general surface contour (though the 3D effects and lighting are provided by the other components).

Exercise #5: To get a sense of how this first tile defines the overall contour of the surface/shading and how it functions **without** the other tiles in the group, remove the other tiles that make up the shading group. Notice that the resulting image, while still intriguing, lacks the detail and three-dimensionality of the real system. **Re-open** the the file without saving changes to restore the shaders deleted in the exercise.

Tip: To re-open the file, simply choose it from the **Recent Projects** menu and click No to the save changes dialog.

Explore what happens when you substitute other components for the **Multiply** component.

Tip: The most convenient way to substitute components is to select the tile and use the left/right arrow keys of your keyboard to cycle through the available components. You will notice that the order differs from the order in the menu.

Shortcut: you can use the tab key to change which tile is selected.

Explore other geometries. Restore the file to its original state by re-selecting **Multiply**. Explore how this shader group works with other geometries. To do this, simply click on the uppermost tile (which contains the compiled tree), and use the left and right arrow keys to cycle through the components. If you aren't familiar with how these components work, you may want to explore them on their own and come back to this lesson. While the overall shape and contours of the images change pretty dramatically, you will notice that there is a texture and lighting style that is common to them. This 'style' is provided by the shading group.

Using the compiled tree's blend parameter when exploring complex geometries. Re-open **Beast Shader Explorer** without saving changes. Click on the compiled tree tile at the top of the system. Set parameter A (**blend**) to 0. Setting blend to 0 essentially bypasses the compiled tree. The incoming space is passed through untransformed. This is a great way to get a sense of how the shading components behave on untransformed space. Don't forget this trick. By putting complex space transforms in a compiled tree, one can easily compare transformed and untransformed geometries.

Coloring component. The final component in the shading group is the **Shaded Main Gradient** component. It provides the color (and shadows) for the image. The component's first input selects a color from the active gradient and the second input determines the brightness (which in turn defines shadows and lighting). This second input acts like the **Global Shading** option available from **Shading Options** popup menu. Use the **Derivative** and **2D Derivative** components to feed this input achieve 3D-type lighting effects. This component is particularly well-suited for creating images that have 3D texture and lighting effects.

Explore how the choosing different color shaders influences the colors. Restrict your exploration to the components that are true shading components. (They are all grouped together in the components popup menu. The color noise and distortion functions don't fall into this group--as you will see if you pick one.)

Did you notice that the other shaders did not provide the same lighting and texture effects? Try the experiment again. This time, however, choose **Global Shading On** from the **Shading Options** popup menu; click on the **2D Derivative** component to select it and type 's'. This will make the **2D Derivative** component the shading component and allow it to provide shadows. This is necessary because the other color shaders don't treat the second input as shading and lighting information.


Re-open the file without saving changes to restore the file to its initial state.

Splitting the shader. The shading part of this system makes use of a useful and somewhat unusual technique. The output of the first component in this section (the **Multiply** component) is split by sending it to two parallel 1D filters (**Infinity Gate** and **2D Derivative**).

Containing an 'object' with Infinity. Notice that the background is the depth shading color (the first auxiliary color). **Shaded Main Gradient**, like all of the other RGB-based shading components treats infinity differently than it is treated in non-RGB systems. Infinity is treated as transparent by most RGB-based components. When ArtMatic encounters infinity at the output of a system, it is drawn with the depth-shading color. If infinity is encountered by an RGB mixing component, it is treated as transparent and pixels from the mixer's other inputs show through.

Infinity is very useful for limiting or defining space. Normally, only the 3D object components (sphere, room, etc.) generate infinity on their own (to define the region outside the object boundaries). You can use the **Infinity Gate** 1D component to define 'object' boundaries. The component lets you specify a value range; values within the range are sent out unchanged; infinity is substituted for values outside the range. Explore how you can alter the 'beast' boundaries by adjusting the minimum and maximum parameters of the **Infinity Gate**.

Lighting and Shadows. The **2D Derivative** component feeds the **Shaded Main Gradient's** shading/luminance input. **2D Derivative** and **Derivative** are special components that can provide 3D texture and lighting in a system. They behave somewhat differently from most other components. Their output is not determined simply by the input values but also by other components higher up on the tree. **2D Derivative** traverses the tree and generates values derived from the system's geometry.

Click on the **2D Derivative** component's tile  to select it. The component generates values which simulate two lighting sources whose angles are determined by the component's A and B parameters. Adjust its A and B parameters and notice how the direction of the shadows changes. The third parameter is an offset which is added to the output value. This function is particularly well-suited for use with the **Shaded Main Gradient** component since Shaded Main Gradient is designed to treat its second input as lighting information.

FURTHER EXPLORATIONS

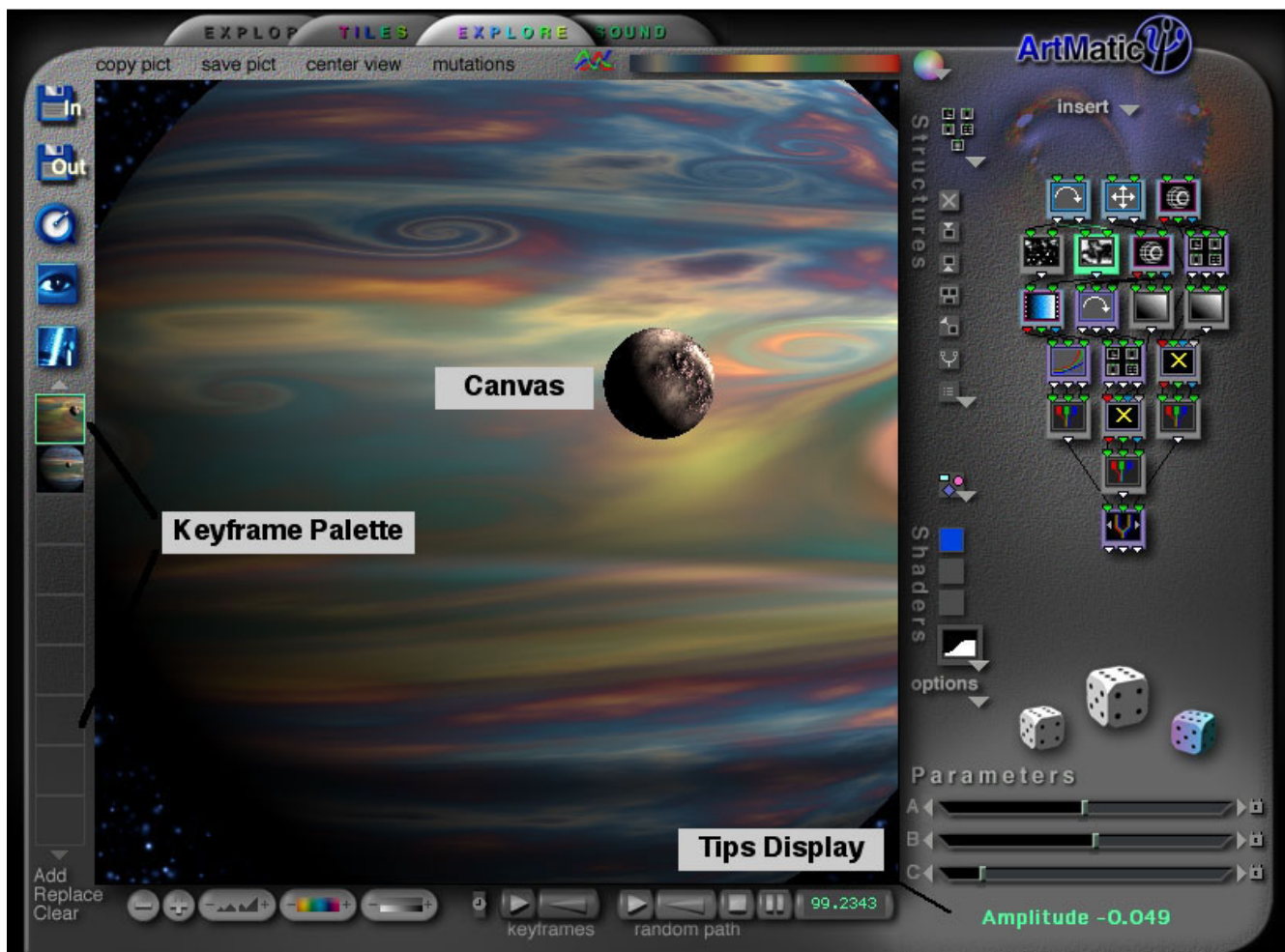
Hopefully, these lessons have provided you with a methodology that you can apply to the analysis of other ArtMatic systems. We encourage you to explore the example files that we have supplied and to use them as the starting points for your explorations.

Tip: To quickly explore the examples, choose **Open** from the File menu and guide the open dialog to the Examples directory. Rather than opening each file, simply click on each file to view its preview. When you find a preview that looks intriguing, go ahead and open the file.

Please let us know if this chapter helped you to understand ArtMatic Pro better and if there are particular example files that you would like included in future tutorials. Send your feedback to support1@uisoftware.com.

User Interface

This section describes ArtMatic's tools and menus. You can click on the user interface elements in the picture below to go directly to the item's description. Click on a section title to go directly to that section: [Modes](#) , [Tools](#), [Sound Mode Tools](#), [Menu commands](#).



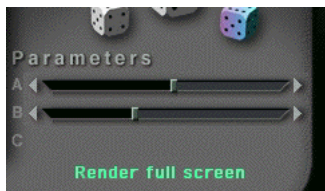
General User Interface Note

As you use ArtMatic, you will notice that nearly every icon acts as a tool or button. **Text Buttons** appear in many of ArtMatic's dialog buttons. Shadowed text is generally active. Holding down the shift key when performing many operations applies the operation to all of the keyframes in the file (These operations include: gradient selection, zooming and scrolling the canvas, and changing parameter values.)

Canvas

The central image display area is called the **Canvas**. Click on the canvas area and drag the mouse left, right, up or down to change the portion of the image plane that is visible. The magnification/zoom tools allow you to zoom in and out on the image. The image visible in the canvas is used when exporting pictures and animation.

Tips Display



As with all U & I Software applications, there is a dynamic **Tips Display** that displays helpful text about whatever tool the mouse is over.

The Tips Display is found in the lower-right corner of the ArtMatic window. We suggest that you mouse over all of ArtMatic's buttons and tools to become familiar with what is available.

Mode Tabs





Clicking on any of the tabs puts ArtMatic into the corresponding mode. ArtMatic's modes are:

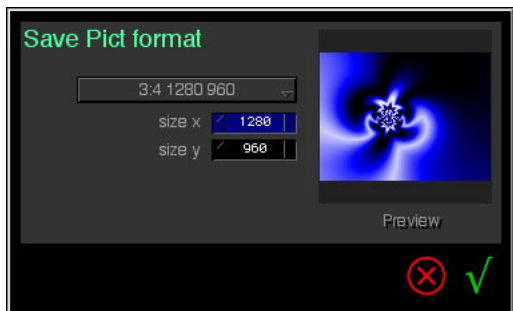
- **Explore (Black and White)** – Similar in functionality to the Explore (Color) mode. In black & white mode, ArtMatic uses a very smooth black to white gradation for all its rendering and offers a different selection of shaders than in color mode. Black and white mode is great for creating images and animations destined to be used as alpha masks. ArtMatic can be used to create animated masks that create mind-boggling wipes and fades when used with movie and QuickTime editing programs.
- **Tiles** – In this mode, the ArtMatic canvas' image is tiled. This mode is used for creating images that need to be tiled.
- **Explore (Color)** – ArtMatic's default mode. This mode is used for creating color graphics and animation.
- **Sound** – This mode lets you generate sounds with ArtMatic. The behavior of a few tools changes when ArtMatic is in Sound mode. Click here to read more about [Sound](#) mode.

TOOLS Top, Left, Right, Bottom

Top Toolset

Copy Pict  - Copies the 512 pixel by 512 pixel canvas area--as displayed--to the clipboard. To save an anti-aliased version of the image with user-selectable dimensions, use the **save pict** tool or the **Save Picture** menu command.

Save Pict  - Save a high-quality anti-aliased version of the canvas as a picture file with user-definable dimensions. When this command is selected, the **Save Picture** dialog box is invoked. After the ok icon button is selected, ArtMatic saves the picture to disk as a Macintosh PICT file. If the picture is large, Artmatic saves the picture in chunks which are displayed as they are being saved. Large pictures can take quite a while to save. You can abort the rendering by pressing the **escape key**. The dimensions of the image to be saved are displayed in the **size x** (horizontal size) and **size y** (vertical size) fields.



These numbers can be entered directly or modified by choosing a size from the dialog box's **size popup menu**. If new dimensions are entered directly, you must click on the **Preview button** for ArtMatic to recalculate the preview with the correct aspect ratio (dimensions).

Preview button. Re-calculates the picture preview to reflect the current dimensions.

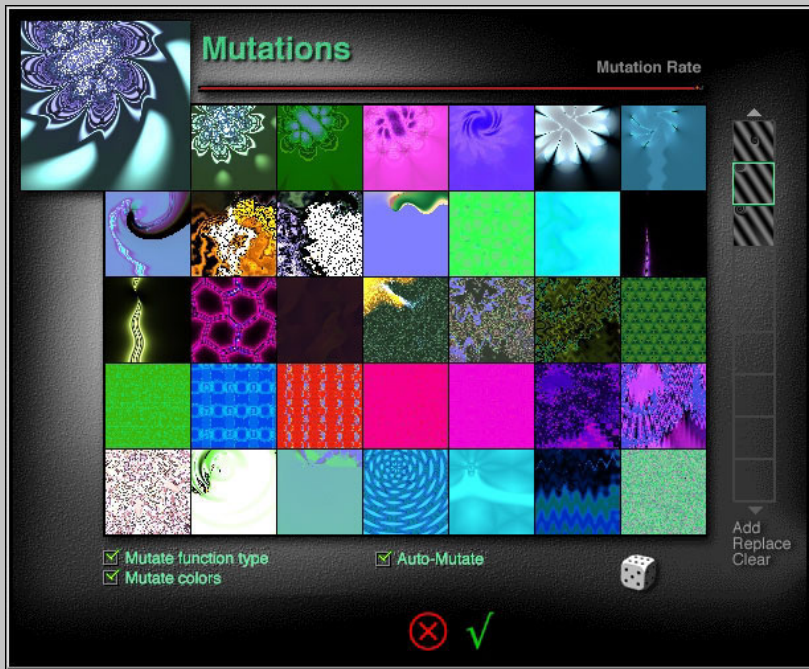
Note: Unlike previous versions of ArtMatic Pro does not require a large memory partition in order to save large images. Also, ArtMatic does not set a print resolution in the picture file. You can use almost any graphic editing program to set the print size of the image.

Note: ArtMatic only saves in PICT format. You can use any number of commercial and shareware applications to convert PICT files to other picture formats. ArtMatic can create pictures up to 10,000 pixels square.

Center View center view - Returns the picture to default co-ordinates and zoom value centered around zero. The default centered view ranges from $-\pi$ (π is 3.14.....) , π in both the x and y axes. Zooming in and out or moving sideways by clicking and dragging in the picture preview will change the view range. You can check the current zoom value and co-ordinates in the **Preferences dialog**.

Mutations mutations - Pops up the **Mutations Explorer** which can be used to quickly explore and mutate the possibilities of the current ArtMatic structure. The **Mutations Explorer** is one of ArtMatic's most powerful tools for exploring an ArtMatic structure. The Mutations Explorer uses *genetic* algorithms to generate mutations of a parent image.

The Mutations Explorer



When the mutations explorer first opens, the current canvas image is the **parent** (the large image in the upper-left). Clicking on the parent will generate a new set of mutations. ArtMatic creates mutations by randomizing unlocked parameter values and the gradient and/or function assignments (if the corresponding options are turned on). Clicking on any mutation (any "child"), makes that mutation the new parent. If the **auto-mutate** option is turned on, clicking on any mutation will make that child the parent and generate a new set of mutations. Clicking the OK icon button (the checkmark icon) sets the ArtMatic canvas to the parent image.

There are a number of settings in the explorer which influence its operation and the range of mutations generated:

Mutation Rate - This slider changes the mutation

rate. At the maximum value (the rightmost setting), the mutations can vary quite radically. At the minimum value, the mutations are more subtle.

Mutate Function - When this is turned on, ArtMatic changes the functions (components) assigned to the current structure tiles.

NOTE: when this is on any mutation will affect existing keyframe since the function assignments are shared by all keyframes in a file.

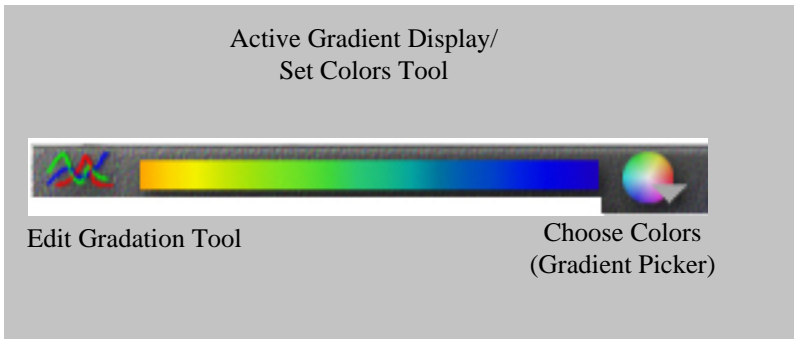
Mutate Colors - When this option is turned on, each mutation will have its own gradient.

Keyframe Palette - This area at the right-edge of the explorer operates just as the keyframe palette in the main window. The palettes allows you to add a number of keyframes from within the explorer.

Auto-Mutate - When this option is turned on, clicking on any child both makes the child the new parent **and** generates a new set of mutations. You can turn this option off in order to save more than one child in the same set of mutations as keyframes.

Die icon button - clicking on the die icon has the same effect as clicking on the parent in the upper-left corner of the window. A new set of mutations is generated.

Gradient Editing Tools



The last group of tools in the upper-toolbar provides gradient (palette) editing and selection capabilities. As noted in the [Shaders](#) chapter, gradients are only active in some systems. The gradient editing tools are not active in black-&-white explore mode.

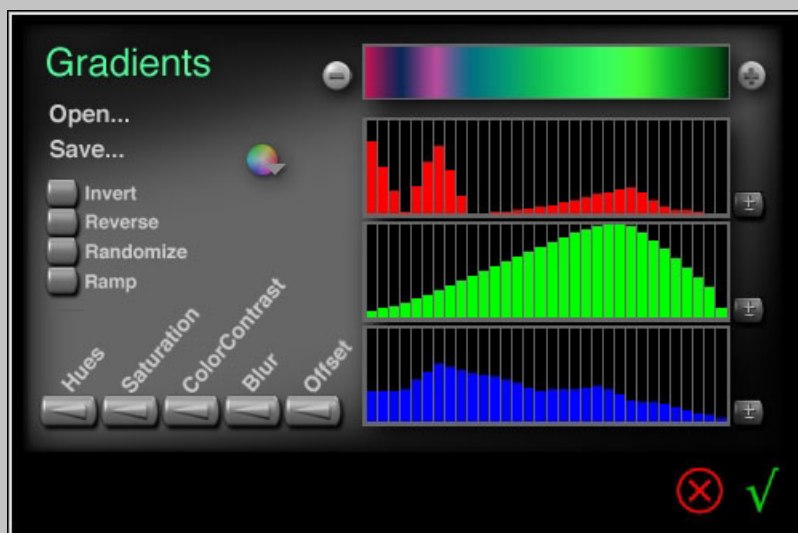
What is a gradient? A gradient or gradation is a special kind of color palette that allows you to easily create high-resolution complex color palettes. A gradient has a user-definable number of color slots. Each slot has its own color, and ArtMatic automatically generates all the colors that lie between adjacent slots so that with a few mouse clicks you can create rich palettes. For example, if you want to create a palette that goes from black to white with all the shades in between, you only need a gradient with two slots. Select black as the left-hand color and white as the right-hand color, and ArtMatic does the rest.

Each keyframe can have its own gradient. When you have created a gradient you like, you can store it as part of ArtMatic's gradient library. You can also export and import gradient libraries using the **Gradient Editor**.

Slots. Notice that the Gradient Editor (shown below) has three parallel rows of 'slots'. The rows determine the red, green and blue content of each gradient slot. ArtMatic interpolates between each of a gradient's slots to provide smooth transitions regardless of the number of slots. To add or remove slots to the gradient, click on the or tools. Click on a slot to change its value, or press and hold the mouse button while mousing over slots to change a number of them at one pass.

Edit Gradation - Pops up the Gradient Editor.

The Gradient Editor



- Click on or to add or remove slots. Colors can also be edited by clicking directly on the gradient display which behaves just like the [Set Colors tool](#) described later.

- Click and hold here to display a menu of your project's gradients.

Open - Open a previously saved gradient library.

Save - Save the current set of gradients as a library.

Invert - Invert the gradient's colors.


Reverse - Reverse the order of the gradient's colors.



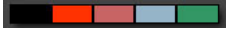
Randomize - Randomize the gradient's colors.

Ramp - Interpolate the slot values so that there is a steady ramp from the first to the last slot.

Hues/Saturation/ColorContrast - click and drag left or right to adjust the gradient's hue, saturation, or color contrast, blur and offset as appropriate.

Blur/Offset - Blur smooths the transitions between colors. Offset rotates the gradients slots.

 - Click and drag left or right to decrease or increase the values of a component color's (red, green or blue) slots.


Set Colors  The **Set Colors** tool lets you directly edit the colors of the current gradation. To invoke it, mouse over the gradient display and click. You will notice that the gradient's appearance changes when you move the mouse over it. For example, when you move the mouse over this gradient  , the display changes to  . Clicking on any color will pop down the **color picker** which is shown below. Holding down the shift key when selecting a gradient applies the gradient to all of a file's keyframes.

Color Picker




When the **color picker** is active, the cursor will change to an eyedropper and will pick up whatever color is below the cursor. You can choose a color from the palette or anywhere else on the screen by simply moving the eyedropper over the desired color and releasing the mouse button. **Note:** you can even pick up a color from a window that is open in another application by shrinking ArtMatic's windows to reveal the windows open in other applications.

To add or remove slots, use the [Gradient Editor](#) described earlier.

 **Choose Colors** - Pops up a menu of the gradients in the active gradient library. The popup menu also provides a **Store Gradation** command with which you can add the current gradation to the library. You can remove a gradation from the list by holding down the option key and then selecting the gradient from the list.

Left-Side Toolset

The left-side toolset varies slightly in different modes. In **Sound** mode, the **QuickTime Movie Export** and **Render Full Screen** tools are replaced by the **Save Sound** and **Play Sound Full Screen** tools.

 **Open Structure** (Command-O) - Open a previously saved ArtMatic structure file. Clicking on this button is equivalent to using the **Open** menu command. **Note:** ArtMatic cannot open picture files exported with the **Save Picture** command.

IMPORTANT NOTE FOR ARTMATIC 1.x USERS: you can open files created with ArtMatic 1.x; however because some of the mathematical functions have been modified, some files will not have the same appearance as in earlier versions of ArtMatic. Due to the nature of iterative mathematics complete compatibility with the older files is not possible.



Save Structure As - Save the current ArtMatic system (including all keyframes) to a new file. Clicking on this button is equivalent to using the **Save As** menu command. To save an existing file, use the **Save** menu command.

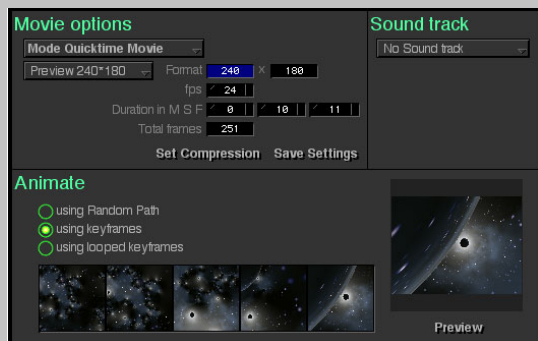


QuickTime Movie Export (not available in **Sound** mode) - Click here to render an animation as a high-quality animation or change the QuickTime parameters associated with this system. When you click on the button, the QuickTime Export dialog is presented. Rendered animation is much higher quality than real-time animation. Unlike the real-time animation generated in the main window, each frame of a rendered movie is a high-quality anti-aliased image. Keep in mind that animations can take up a lot of disk space and can take quite a while to render. Use of a high quality QuickTime compressor (accessed using the **Set Compression** text button) can greatly reduce the amount of disk space required to store a movie. To cancel rendering once it has started, press the escape key.

The quality of the rendering is related to the settings you choose in the export dialog. The quality of the playback is influenced by both the settings used when rendering **and** the playback capabilities of your computer. It is possible to render movies, for instance, at a higher frame rate or with a larger frame size than your computer can play back in real time. To learn more about ArtMatic animation and QuickTime, see the [Animation](#) and [QuickTime Notes](#) chapters of this manual. The **QuickTime Notes** chapter is recommended reading for anyone using ArtMatic to create animation.

Save Settings without rendering. You can change the animation settings (such as duration and frame rate) without rendering the movie by clicking the **Save Settings** button in the dialog and then clicking the cancel button.

QuickTime Export Dialog



Mode popup (movie or picture sequence) - Use the **mode popup** to determine whether an animation is rendered as a QuickTime movie or as a sequence of individual picture files (each sequentially numbered). Pict sequences are recognized by most movie editing programs and can be a convenient way of overcoming the 2 gigabyte file size limitation to which some versions of the system software are subject.

Format popup and fields - The dimensions of your animation are determined by the format fields. Located below the mode popup menu, the **format popup** provides a list of common frame sizes. You can either select a size from the popup or type in your own values. Be aware that large frame sizes are difficult for some computers to play back at high frame rates and may appear choppy when played back. Larger frames also take up more disk space. **DV FORMAT NOTE:** When a DV format is selected via the popup menu (either NTSC or PAL DV), ArtMatic Pro makes adjustments when rendering to account for the non-square DV pixels. As a result, the preview will look slightly stretched but the image will look right when the movie is played back on a DV playback device or with an application that adjusts for DV's non-square pixels. To bypass this adjustment, choose a non-DV format from the popup and type in the desired width and height.

fps (frames per second) - This is the frame rate of the rendered QuickTime movie. The larger this value is the more processor-intensive playback is and the larger the resulting file will be. Values of 10-15 generally give very nice results for computer playback. For NTSC broadcast-quality animation, enter **29** for the frame rate and turn on the **NTSC Drop Frame** option which appears to set the actual frame rate to 29.97 NTSC drop-frame (the American standard for video). **Note:** The **NTSC Drop Frame** option only appears when the frame rate is 29.

Duration in M S F (minutes, seconds, frames) - The length of the movie expressed in minutes, seconds and frames. Note that changing this value will result in a corresponding change to the **Total Frames** field.

Total frames - The length of the movie expressed in frames. Note that changing this value will result in a corresponding change to the **Duration in M S F** field. Note that changing the duration or the frame rate will result in a corresponding change to this field.

Set Compression text button - Clicking this button invokes the standard QuickTime compression dialog. This allows you to generate a movie that is compressed. The Sorenson option (available to users with QuickTime Pro) can yield space savings of 90%. In most cases, there is some loss of detail to the resulting animation. But this is generally compensated for by the enormous space savings. You may want to experiment with a short animation first to see which options yield the best results.

When choosing a compressor it is important to know what you will be doing with the animation. If the animation will be sent to a digital video device (DV) then you should choose a DV compressor. If you will do further editing of the movie in a movie editing application, you should use either the Animation compressor or DV compressor. The Sorenson compressor should only be used in cases where additional editing or mixing will not be performed. See [QuickTime Notes](#) for more information about choosing the correct QuickTime compressor

Save Settings text button - Press this button to save the current settings with your file. This button allows you to save settings changes without having to render the file. The settings are saved when you save your project even if you press the dialog's cancel button which allows you to change the settings without rendering the animation.

Animate Options - The selected option determines the content of the rendered movie. The dialog box displays frames from the beginning, middle and end of the animation as well as a mini-realtime preview. When you change the animation options, these frames and the preview will be adjusted accordingly.

- **Using Random Path** - This animation option performs the same sort of semi-random animation as when you press the [Random Path Animation](#) button. The Start Time and Animation Delta time fields let you specify where along the random path the animation starts and the animation's rate of change.
 - **Start time** - This is the [Random Path Time](#) used as the starting point for the animation.
 - **Delta time** - This is setting determines the rate at which the random path is traversed: the lower the value, the faster the pace of the random walk.
- **Using keyframes** - This is the most frequently used animation method. ArtMatic smoothly interpolates between the file's keyframes. The animation speed is determined by the duration setting. The time between keyframes is constant.
- **Using looped keyframes** - This method creates a smooth loop using the keyframes and will start and end with the system's first keyframe.

Preview text button - Clicking this button generates a realtime preview of the animation that will be exported. Note that the preview's duration may be longer than the actual rendering because some systems are too computationally intensive to be previewed in real time. To abort the preview, press and hold the mouse button or the ESCAPE key until the preview stops playing. Stopping the preview of a complex system may take a little while.


Sound Track Options - Clicking on this pop-up menu lets you choose an input sound to use as the animation soundtrack and, optionally, as the source of the animation. The options are:

- **No soundtrack**
- **Use ArtMatic Sound** - Generates the sound that would have been generated in Sound mode and uses that sound as the movie's soundtrack.
- **Use Sound File** - Prompts you for a Sound Designer II format file to include as the movie's soundtrack.
- **Animate by Sound** - Prompts you for a Sound Designer II format file to include as the movie's soundtrack and uses that sound file to create the animation according to the rules described for [Audio Input](#). When rendering, the only sound input that ArtMatic can use is a sound file.



Render Full Screen - Render a full-screen version of the canvas with anti-aliasing. Clicking anywhere on the screen will return you to normal viewing mode. This button is not available on the **Sound** page.

Shortcut: Pressing 'p' (without the command key) is equivalent to pressing this button. **To abort rendering,** press your keyboard's **escape** key.

 **Audio-In Control** – Click this tool to perform real-time animation modulated by the computer's sound manager input. Audio-in animation can be based on either keyframe or random path animation. Whichever animation mode (random path or keyframe) was most recently played will be used as the basis for the animation. **To stop the animation**, click anywhere or press the space bar of your keyboard. The real-time sound animation is provided for two purposes: to get a sense for the the system's responsiveness to sound input before rendering an animation using sound modulation and for entertainment. The real-time animation is not intended to be performance quality, but it can be quite entertaining.

The animation is performed by using a real-time 18-band FFT analysis of the computer's sound input to modulate the ArtMatic system's parameters. The mapping of frequencies to parameters is completely automatic and not customizable. This feature requires a fairly fast processor to work well. Computationally-intensive ArtMatic structures may not perform well with Audio-In Control. The overall sensitivity to the sound input can be modified in the [Preferences](#) dialog. Some structures will work best with low input sensitivity while others may require greater sensitivity.

ArtMatic uses your Mac's Sound Manager input device for the sound input. Use **Sound** or **Monitors & Sound** control panel or the **Sound control strip** to select the sound source (i.e the CD player or mic input, or other sound input device). If you use the Mac's builtin mic, you can have your voice or background music drive the picture. You can choose the Mac's CD drive then insert and play a CD and have the music "animate" the parameters.

Any ArtMatic picture can be animated with sound input, resulting in vastly different and often delightfully curious interpretations of the sound source. This real-time animation will give you an idea of how the system responds to sound. For a high-resolution sound animation, you can specify a soundtrack file in the [QuickTime Export Dialog](#) and choose the **Animate by Sound** option.

Note: Because this process is processor intensive, the real-time animation is performed at a much lower resolution than when QuickTime rendering is done.

Keyframe Display and Tools



ArtMatic files can store up to 16 keyframes which are displayed in the left-hand toolset. Since only 9 keyframes can be displayed at a time, there are scroll arrows at the top and bottom of the display which allow you to scroll through the file's keyframes. All of a file's keyframes share the same, structure (component tree), component (function) assignments and shader. If any of these change all keyframes are impacted.

Click an empty keyframe slot to add the current **canvas** as a keyframe (it will occupy the first empty keyframe slot). **Command-click** any keyframe to replace it with the current canvas. **Option-click** any keyframe to delete it.

The active preset is highlighted with a green frame. Keyframes may also be added, replaced and removed using the keyframe palette's text buttons:

- **Add** – Click on this button to add the current settings as a keyframe. The frame is added to the next empty slot. Keyframes may also be added by clicking on an empty slot. This has the same effect as choosing Add Keyframe from the Animation menu. Command-B can be used as a shortcut for this tool.
- **Replace** – Click on this button to replace the active preset with the current settings. **Command-click** any keyframe to replace it with the current settings. The command is the same as the Animation menu's Replace Keyframe command. Its shortcut is command-R.
- **Clear** – Remove the active preset. **Option-click** any keyframe to replace it.

More information about keyframes can be found in the [Animation](#) chapter of this manual.

Global changes to keyframes. Holding down the shift key when performing many operations applies the operation to

all of the keyframes in the file. (These operations include: gradient selection, zooming and scrolling the canvas, and changing parameter values.)

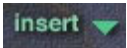
Right-Side Toolset

The right-side toolset consists of the structures area, the parameter sliders, and the shading options popup menus.

Structures Area

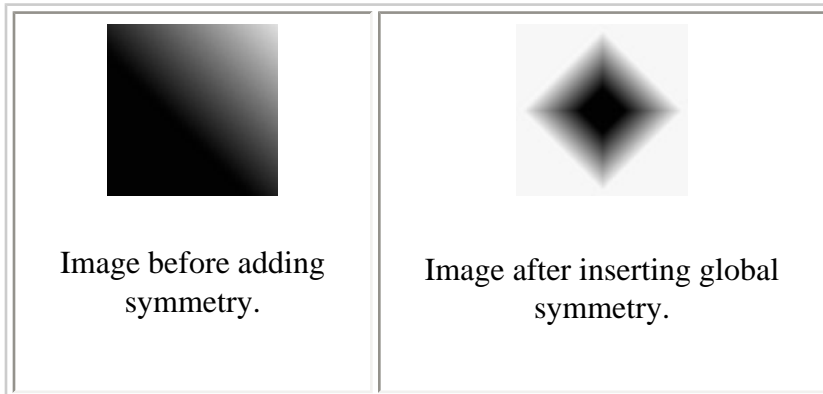


Choose Structure - Press this tool to pop up a menu of pre-defined structure trees. This is a quick way to get started building a structure. You can further modify the structure with the tools described later. This button allows you to choose a tree structure from ArtMatic's list. In general, the fewer components there are, the more quickly the system can be rendered. On the other hand, larger structures offer a richer array of images.



Insert Component pop-up menu - This pop-up menu provide commands for modifying the component tree (structure).

- **Insert global rotation** - Adds a [rotation component](#) at the top of the tree. Use this command if you want to rotate the image.
- **Insert global symmetry** - Adds the [mirrors 4](#) component at the top of the system to provide symmetrical tiling:

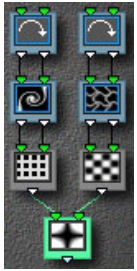


- **Insert perspective** - Adds a group of three tiles at the top of the structure that provide a three-dimensional perspective effect. To rotate and orient the plane, select the added $Ax + By + C$ tile and modify its A or B parameter. You can also adjust the amount of perspective using the Perspective tile's parameter A. For a less dramatic effect, you may zoom into the middle of the plane.
- **Disconnect** - Disconnects the selected tile from its predecessor. The selected tile (and any connected tiles after it) become a new branch of the structure tree.
- **Delete selected** - Deletes the selected tile. There may be cases where a tile cannot be deleted in which case the computer will beep.
- **Append 1D filters** - Appends a 1-in/1-out tile to each output of the selected tile.
- **Append Scalar (1 out)** - Inserts a one output tile after the selected one. **Note:** This command is only available when the selected tile has two or more outputs.
- **Append Vector (2 out)** - Inserts a two output tile after the selected tile. **Note:** This command is only available when the selected tile has two or more outputs.
- **Append 3D Vector (3 out)** - Inserts a three output tile after the selected one. This command is only available if the selected tile has two or more outputs.
- **Add branch** - Adds a new branch which forks off from the selected tile. The last component in the system is modified to accommodate the new branch.
- **Add Parallel branch** - Adds a new branch parallel to the selected tile.
- **Complete Tree** - Completes the structure tree by closing it and joining any loose branches. Learn more about tree

completion in the [Concepts](#) chapter.

- **Add Color Pict/Movie item** - Adds a color RGB Pict/Movie component after the selected tile. This command is only available when the selected tile has two outputs.

Structure/Component Tree



A file's **structure** (also called the **component tree**) is an arrangement of tiles (also called components) which is used to generate the displayed image. Each tile's outlets are fed into the inlets of the following tile. You choose a basic structure using the [Choose Structure](#) popup menu. The structure can be further modified by using the available editing tools.

Detailed information about the component tree and available functions is found in the [Concepts](#) chapter of this manual.

Note that the tile assignments (but not the tile parameters) are shared by all keyframes in the file.

Component Tiles

To change a tile's function, click on any tile and hold down the mouse button to pop up a menu of the available functions. The function list varies according to the number of a tile's inlets and outlets.

To modify a tile's settings, click on a tile and adjust the [parameter sliders](#).

Replacement/options popup. *Option-click* a tile to pop up the [Replace/Options](#) menu that features handy commands for modifying the selected tile. The menu is the same as the one associated with the [Replace](#) tool.

Modifying tile connections. There are two ways to change the connections between a child tile and a parent tile found higher on the tree. Automatic connection (**command-click**) forces an automatic connection from a child to a parent tile. The new connection dialog (**option-command-click**) allows for custom manual connection of tile inputs and outputs. Both methods require that the parent tile be higher on tree than the child tile and can be used to either connect loose or open inputs or to change a tile's parent.

Automatic connection. Automatic connection is the quickest way to change a tile's parent or force connection of unconnected inputs. To force an automatic connection, select a child tile then **command-click** a tile higher in the tree that will become the tile's parent. Be aware that choosing a parent tile from another branch will sometimes cause ArtMatic to re-arrange the tree's layout. **Undo** can be used to undo this automatic re-patching if the results are undesirable. ArtMatic does its best to determine whether you are trying to completely change the parentage of the child, or whether you are simply trying to connect and open input. In some cases, ArtMatic will not be able to determine an appropriate automatic connection. In such cases, use the **Connection Dialog** described below.

Connection Dialog





For greater control of child/parent connections, use the **connection dialog**. It allows you to create (or break) connections between any inputs and outputs of child/parent tiles and even to split parentage between tiles. To invoke it, click on the child tile and **option-command-click** any tile higher on the tree. The dialog displays the parent and child tiles and provides two ways of establishing connections.


1. Direct editing. Click on a parent output then click on the child's input to which to connect it.
2. Via the father/son pop-up menus. Choose the father's output, choose the son's input then press either the **connect** or the **disconnect** buttons.


An exercise: Use the connection dialog to horizontally flip an input image. **Hint:** start with the **RGB 1 Channel** structure (available from the structures popup) and use the connection dialog to re-arrange the connections between the first and second components in the tree.

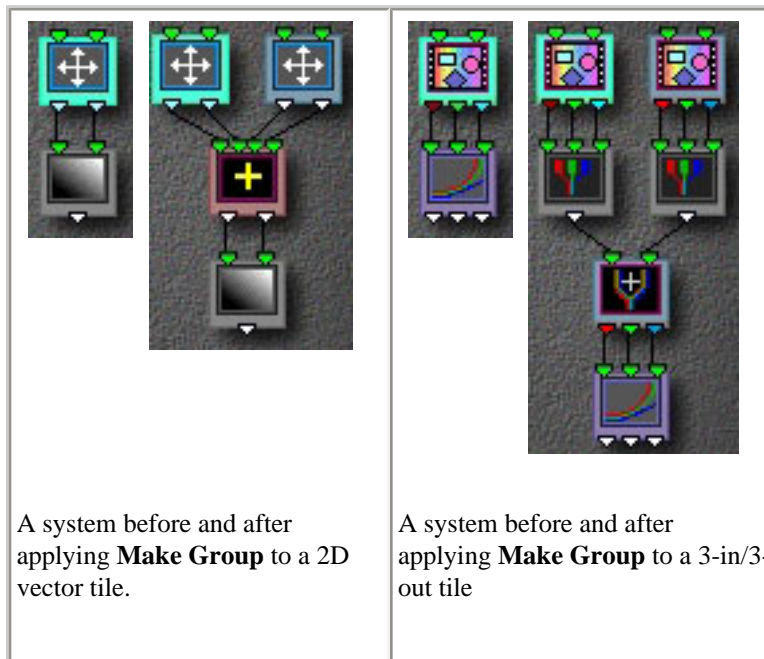
Tree Editing Tools


 **Delete selected** - Delete the selected tile.


 **Insert After** - Add a new tile after the selected one. The new tile will have as many inlets as the selected tile has outlets. The new tile will have the number of outlets necessary to connect it to the tile which follows it.


 **Insert Before** - Add a new tile before the selected one.

 **Make group** - Replace the selected component with a group that consists of a new identical component and another new component to mix the outputs of the original and the added components. If the selected component has three outputs then a pack component is added to each branch of the group as well as illustrated below.



 **Add branch** - Add a new tile which branches off the selected tile. This command forks a branch at the selected tile.

 **Complete Tree** - This handy tool adds the components necessary to mix any parallel branches whose outputs are open. An open output is one that is not connected to the input of any tile. Normally, a tree should have only one component with an open output. This tool saves a number of steps when creating structures that have parallel branches that you want to contribute to the final output. Tree completion is discussed in greater detail [below](#) and in the user interface chapter.

 **Replace popup** - Click and hold the mouse button to pop up a list of commands that can be used to edit the structure tree. Option-click any tile to pop up this list of commands. The available commands are:

- **Delete** - Delete the selected tile.
- **Insert After** - Add a new tile after the selected tile.
- **Insert Before** - Add a new tile before the selected tile.
- **Split component** - Split the selected component into several parallel components. A two-in/two-out tile, for example, is split

into two parallel 1-in/1-out tiles; a three-in/two-out tile is split into a two-in/1-out and a one-in/one-out tile. This command is only active if the selected tile has two or more outlets.

- **Make group** - Replace the selected tile with a group of tiles. This command is a shortcut for the [Make Group](#) tool.
- **Make perspective** - Add the tiles needed to make the selected tile part of a perspective group which creates a 3D perspective group similar to that created by the **Insert Perspective** command discussed earlier in this chapter. (This command is only active when the selected tile has two outlets).
- **Replace with scalar (1 out)** - Change the number of the selected tile's outlets to one. (Only available if the current number of inlets is compatible with one outlet.)
- **Replace with vector (2 out)** - Change the number of the selected tile's outlets to two. (Only available if the current number of inlets is compatible with two outlets.)
- **Replace with vector (3 out)** - Change the number of the selected tile's outlets to three. (Only available if the current number of inlets is compatible with three outlets.)
- **Replace with 1 in** - Change the number of the selected tile's inlets to one. (Only available if the current number of outlets is compatible with one inlet.)
- **Replace with 2 in** - Change the number of the selected tile's inlets to two. (Only available if the current number of outlets is compatible with two inlets.)
- **Replace with 3 in** - Change the number of the selected tile's inlets to three. (Only available if the current number of outlets is compatible with three inlets.)
- **Replace with 4 in** - Change the number of the selected tile's inlets to four. (Only available if the current number of outlets is compatible with four inlets.)
- **Replace with color pict** - Replace a selected two-in/one-out tile with a 2-in/3-out [RGB Pict/Movie](#) component.
- **Pack outputs** - Add a [Pack](#) component tile after the selected 3-out tile.

Editing Tips

Joining branches at the top. If a tree has parallel branches you would like to connect at the top, simply choose the Insert menu's **Insert Global Rotation**. ArtMatic inserts a rotation tile at the top of the system to which both branches are connected. You can change the component to something other than rotation if you desire.

Adding filters. When building complex trees or experimenting with mutations or random path animation, you may find it helpful to insert filter components to restrict the range or modify the values being fed into some but not all of a component's inputs (or outputs). New editing commands have been added to make this possible. See the new [Inserting Filters](#) tutorial lesson to learn this powerful technique.

Editing restrictions. There are a couple of editing limitations worth noting. There is a limit to the number of columns (four) and the number of rows (ten) that an ArtMatic structure can have. Compiled trees within the tree can be effectively used to work around this limitation.


Branch order "instability". While editing the tree structure, ArtMatic may occasionally switch the order of the branches. You should watch out for this behavior when deleting tiles from a branch. You can swap the position of two parallel branches (that are not connected at the bottom) by typing 'm' (for main component). Note that this only works if the branches are open at the bottom. If the two branches are joined at the bottom, delete the component that joins them before typing m. Use the Undo menu command to flip the order back to the original order.



Choose Pict/Movie tool [popup] - Select a picture or movie input from this popup to assign it to the selected Pict/Movie component tile. Choosing an empty slot prompts you to select a picture or movie which will be added as an available input source. Read more about this tool in the chapter [Using Pictures and Movies](#). Option-selecting an item from the popup prompts you for a file to use instead of the current one.




Randomize Parameters (small left-hand die) - Randomize the parameters of all the structure's tiles. Repeatedly clicking this tool is one way to explore the possibilities of a particular system.

 **Randomize All** ('r') (central die) - Randomize all of the structure's function assignments, parameter values and color assignments. **Control-click** randomizes the structure's shape as well as the functions assignments.

Note: If you have keyframes, using this tool will change each keyframe's appearance (since all keyframes in a file share the same tile assignments). You can use the **Undo** menu command to restore your keyframes' appearance if you don't like the results of using this tool. (Note that control-click can't be undone.)

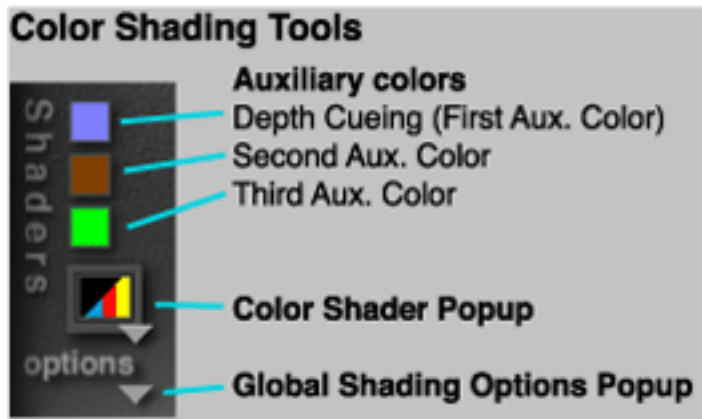
Randomize All is a way to explore the immense parameter space of ArtMatic's dynamical systems. When you find a promising system using this command, it is advisable to save a keyframe and then explore the system using either **Random Path Animation** or the **Random Path Scroll** tool. You can also use the **Mutations** dialog as a quick way to discover what happens in the current canvas' neighborhood. Sometimes a much nicer picture lies near by.

Shortcut: Pressing the 'r' key (without the command key) is equivalent to pressing this button.

 **Randomize Colors** (small right-hand die) - Mutate the active gradient.

Color Shading Tools

The next group of tools relates to the system's color mapping which is covered in detail in the [Shaders](#) chapter.



Auxiliary Colors - The auxiliary color squares are color pickers for choosing the auxiliary colors used by some shading algorithms. The number of auxiliary colors is determined by the shading algorithm. The first auxiliary color (the depth cueing color) is always active.

Color Shader popup menu - Click here to pop up a list of the color shading algorithms which are described in the [Shaders](#) chapter. The contents of this menu changes when ArtMatic is in **Sound** mode. The [sound shaders](#) are discussed in their own sections of this chapter.

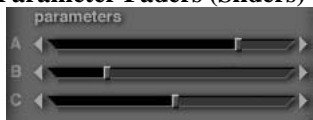
Shortcut: Use the [and] keys of your computer's keyboard to cycle through the shaders.

Shading Options popup menu - Click here to pop up a list of the global shading options. These are discussed in more detail in the [Shaders](#) chapter. The options are:

- **Depth Cueing Off** - Turn depth cueing (a distance/fog effect) off.
- **Depth Cueing Small** - Turn low-intensity depth cueing on.

- **Depth Cueing Medium** - Turn medium-intensity depth cueing on.
- **Depth Cueing Strong** - Turn high-intensity depth cueing on.
- **Global Shade Off** - Turn global shading off.
- **Global Shade On** - Turn global shading on. Global shading uses a component in the system to control the pixel luminance. By default, ArtMatic assigns the controlling component automatically.
- **Component Sets Depth** - (Shortcut: 'd') Override ArtMatic's automatic depth cueing control by assigning the currently selected tile to control the depth cueing.
- **Component Sets Shade** - (Shortcut: 's') Override ArtMatic's automatic shading control by assigning the currently selected tile to control the shading effect.
- **Fog and Shade Automatic** - Set both automatic control of depth cueing and global shading. Note that this does not turn depth cueing or global shading on if they are currently turned off.

Parameter Faders (Sliders) - Component (tile) parameters are modified using the parameter sliders and parameter envelopes. Each tile in a structure has from 0 to 3 parameters (settings). Each keyframe in a file can have different settings for these parameters. To modify a tile's parameters, click on the tile then manipulate the parameter sliders. To see what a parameter controls, move the mouse over the slider and note the text in the [Tips Display](#). Drag a slider to change its value or click on the arrows to make a small change. It is possible to edit parameter changes across keyframes. To do this, click on the parameter's label (A, B or C) to display the [Parameter Envelope](#) dialog box which is discussed in the [Animation](#) chapter.



There are several modifier keys which provide additional control when using the sliders.

Option-drag a slider to make very fine adjustments to the parameter's value (which is displayed in the [Tips Display](#) as you drag).

Option-click the arrows to make a fine adjustment.

Control-click a slider to set it to the default value for that parameter (which varies according to the particular function); this is convenient for setting the parameters such as Scale and Offset to 100% and 0 respectively.

Shift-change parameter value: Hold down the shift key when making a change to apply the change to all keyframes.

Shift-control click: Sets all keyframes to use the parameter's default value.

Shift-command click: Makes a parameter *ramp* which will ramp the parameter's value across its full range over the course of the current keyframes. The first keyframe will have the minimum value and the last will have the maximum value. Use this technique to create smooth transitions across all keyframes. For example, you can create consistent zooming or rotation using this technique. The parameter envelope can be edited by clicking on the parameter label (A, B, or C).



Parameter Locks - The parameter locks (found to the right of the parameter sliders) allow function parameters and component assignments to be locked to prevent them from being changed by operations that normally change parameter values. Parameter changing operations include: random path animation, the Mutations dialog, keyframe animation, and the randomize parameters die (the lefthand die).

To lock all tiles, shift-click any unlocked parameter. **To unlock all tiles**, shift-click any locked parameter. This makes it possible to use the **Mutations** dialog to explore variations of only a few parameters.

Function locking. It is possible to lock the function assigned to a tile so that it is immune to operations that mutate function assignments (such as when you roll the big dice or use the **Mutations** window with the "mutate functions" option). To lock a tile so that its function will not be changed, lock all three parameter locks when the tile is selected. Note that even if the component has no parameters, all three locks must be locked to lock the tile.

Locking parameters and functions is a great way to explore ArtMatic systems. Parameter/function locking makes it possible to use mutations and the large die to explore subtle refinements of systems. For example, you might have a system that uses a few tiles to provide the texture for a surface. You could lock all of the parameters and functions of the system except for the tiles that provide the texture and then use the big die or the **Mutations** window to discover new textures created by function mutation that affects only the few components that provide the texture.

A note about parameter locking and keyframes. When parameters are locked, keyframe animation uses the last values assigned to the locked parameters when animating the system. As a result, any parameter changes stored between keyframes are ignored while the parameters are locked. The parameter changes are not lost however; they are just ignored. When the parameters are unlocked, any parameter changes stored in the keyframes will be honored during animation. Parameter locking is intended for refined control or mutations and for investigating how a particular parameter effects the system. It is not intended as a method for making a global parameter change (i.e. for setting all keyframes to the same parameter value). To use the same parameter value in all keyframes, it is best to either shift-drag the parameter slider to the preferred setting (holding down the shift key while performing most editing operations shares the result among all keyframes) or to click on the parameter label and use the **Send To All** button in the [Parameter Envelopes](#) dialog.

Bottom Toolset


These tools provide additional control of the camera and gradients. When using these tools, you can apply the change to all keyframes by holding down the shift key while applying them.


A note about magnification and iterations: The **Max iterations for fractals** preference affects the resolution of fractal-based components (such as the Mandelbrot function). When this preference has a low value, there is limited resolution for the fractal. As a result, there will be limited detail as you zoom in. If you set this to a large value, you can zoom very far in and find beautiful detail. The drawback to a large setting of this value is that the computation time increases with the number of iterations.


Sound Mode Note: In **Sound** mode, the Zoom tools affect the sound output's pitch. To return to the key specified in the [Preferences](#) window, click on the **Center View** button at the top of the user interface.


 **Zoom Out** /  **Zoom In** - Press and hold these buttons to zoom the camera in or out.


Keyboard shortcuts: The '-' and '/' keys zoom in or out by 100%. The up and down arrows zoom continuously in/out. In the canvas, *command-option click* zooms the image out and *command-click* zooms in.

 **Zoom slider** - Click and drag left or right to gradually move the picture closer to or further away from you.


 **Color Hues** - Click and drag left or right on this button to adjust the the hues of the current gradient. By clicking and dragging with the option key down, you can modify the color saturation.


 **Color Contrast** - Click and drag left or right on this button to gradually change the picture's contrast. By clicking and dragging with the option key down, you can blur the gradient.

 **Duration** - Click and drag left or right to lengthen or shorten the animation's duration. Mouse over the icon to display the animation's duration in the Tool Tips area. For fine control of the duration use the [QuickTime Export](#) dialog.

 **Animate Keyframes** - Play a low-res realtime animation that interpolates between the keyframes. In order to play the animation in real time, ArtMatic uses a lower resolution than when it renders animation to QuickTime. See the [Animation](#) chapter of this manual for more information about keyframe animation,. **To stop the animation**, click anywhere or press the spacebar of your keyboard. A higher resolution preview is available via the **Animation** menu's [Preview](#) command.

Shortcut: pressing the spacebar starts/pauses/continues keyframe animation playback.


 **Interpolate Keyframes** - Click and drag left or right to scroll forwards and backwards through the keyframe animation. The **keyframe time** is displayed in the **Tips Display** area as you scroll. Keyframe time starts at time 0 and increments by one with each successive keyframe.


 **Random Path Animation** - Preview random path animation in realtime. Random path animation is performed by changing all parameter values according to a pseudo-random pattern. The speed at which the parameters change is determined by "Animation Delta Time" field found in the [QuickTime Export](#) dialog. Random path animation is a great way to explore the possibilities of a particular system as well as a way to create extraordinary animation. The realtime animation is a low-res rendering. A high-res non-realtime rendering can be done using the [QuickTime Export](#) dialog. When rendered via QuickTime, you can use a soundtrack to modulate the animation.


Shortcut: To start/pause/continue the animation, press option-spacebar on your keyboard.


To stop the animation, click anywhere or press option-spacebar on your keyboard. Pressing option-spacebar bar is equivalent to pressing the Pause button. **Note** that the [Random Path Time](#) is displayed as the animation plays.

ArtMatic Tip: The random path is actually a fixed pseudo-random path through the structure's parameter space. Since the path is predefined, it will produce the same results each time it is run on the same structure tree. The path is a parametric random curve in an N-dimensional parameter space where N is the number of the tree's component parameters.

 **Random Path Scroll** - Click and drag left or right to gradually move forwards and backwards along the random path. All parameters of the system will be changed simultaneously. This is a great way to find interesting locations and explore all the graphic possibilities of a system. Parameters are changed relatively fast (0.25 steps along the path) when dragging the scroller. To move in smaller increments, you can press the option key while you click and drag on the scroll icon.

 **Stop** - Stop the animation and resets the Random Path time to zero.

 **Pause** - Pause the animation, keeping the current value of the path time. To continue, press option-spacebar or the Random Path Animation button.

 **Random Path Time** - Displays the current location (in "parameter time") along the random path. You can click on this display and type in a start position. Using the **Random Path scroll** to travel along the curve will modify the current Random Path Time. You can use values you discover here as starting points when rendering by typing them into the [QuickTime Export](#) dialog's Start Time field.

Sound Mode Tools

ArtMatic is capable of creating sound files as well as pictures and animation. ArtMatic's sound generation technique is unique and capable of creating surprising and interesting sounds that are often used as starting points for further manipulation with U & I Software's MetaSynth electronic music composition and sound design application.

When playing an animation in sound mode, ArtMatic generates a realtime preview of the sound in addition to previewing the visual animation. The sound animation is quite CPU-intensive. ArtMatic gives preference to the sound generation (over the visual display) during sound animation. As a result, on slower machines (or with complex systems), the display may not be updated during sound animation. To stop the sound animation, press and hold the mouse button or the Escape key until the sound animation stops.

A note about cancelling/aborting playback: Because ArtMatic devotes most of the computer's processor to the intensive task of creating the realtime preview, cancelling can take a long time. You may need to hold the mouse button for 30 seconds or more. Be patient! ArtMatic will eventually stop playing the sound.

ArtMatic's **Sound mode** has a slightly different user interface than the other modes. The [QuickTime Export](#) tool is replaced by the **Save Sound File** tool which creates a sound file from the current structure using either Random Path or Keyframe animation (whichever was most recently performed). The **Render Full Screen** tool becomes the **Play Sound Full Screen** tool though the icon stays the same. The Shaders popup menu's choices change to a list of sound algorithms.


How ArtMatic Generates Sound


In **Sound mode**, ArtMatic generates sound by animating the system and deriving pitch and tonal information from the canvas as the sound generator circles over the canvas. In this mode it is useful to think of the canvas as a 3D terrain map where the colors on the right of the gradient represent high altitudes and the colors to the left represent low altitudes. Zooming in and out affects the pitch. The further out you zoom, the higher the resulting pitch will be. The reference pitch can be tuned in the **Preferences** dialog. The reference pitch is the pitch heard when the system is in the default centered view. For a system to maintain a pitch true to the reference pitch, you need to use zoom levels that are even multiples. A zoom level of two (a single click of the zoom in/out buttons) changes the pitch by one octave.

All of the sound algorithms except Direct Drone, calculate the sound by circling along a picture's surface at 440 cycles per second in the default centered view. The rate of circling determines the pitch and is determined by the zoom level. The further you zoom out, the faster the sound generator circles above the system and hence the higher the pitch becomes. The hills and valleys of the surface (as represented by the colors) define a waveform that evolves continuously as the picture's parameters change. The surface contours provide the sound's harmonics/overtones. The faster and smaller the distance between the peaks and valleys, the richer the resulting waveform will be. The overall difference between the peak heights and valley depths determines the overall volume.

The Direct Drone method treats the structure tree as a complex oscillator whose waveform is the output of the system (drawn as a waveform rather than a picture).

The following tools are only available after the **Sound** mode tab has been clicked. They replace the **QuickTime Export** and **Full Screen Preview** tools and the **Shaders** options available on the other pages.

 **Play Sound Full Screen** (lefthand toolset) Play the sound associated with the system while it evolves along either the random path or the keyframe path. The user interface and menu bar are hidden. This tool plays back using the last animation method used (keyframe or random path). If pressing the button results in no or brief animation then you should add some keyframes or increase the playback duration. You can click on the Random Path animation button to force ArtMatic to use Random Path animation the next time that you press the **Play Sound Full Screen** button. See [Modes](#) for more details.

 **Save Sound file** (left-hand toolset) - Save the sound as a Sound Designer II file. When you click on this button, a dialog box will appear, allowing you to specify a name for the file. Clicking on the Save button will close the dialog box and begin playing the sound. The sound you are hearing is being recorded to disk. Clicking anywhere on the screen will terminate the current sound playback and recording.




Sound Shaders (algorithms) - In **Sound mode**, the **Shaders** popup provides the choice of the sound algorithm used to create the sound. The algorithms are:

- **Drone:** produce a single continuous waveform with a stable pitch. To generate a drone, ArtMatic circles along the system surface (at its default zoom level) at a rate determined by the reference pitch (440 cycles per second for A) and the camera's zoom level. The hills and valleys of the surface determine the waveform content. The output value is fed through a sine filter before processing the sound to keep the output within audible range.

- **Rhythmic**: produce a rhythmically pulsing drone. The pulse's tempo is specified in the [Preferences](#) window. Clicking on the small dice will randomize the rhythmic pattern.
- **FM drones**: produce a drone modulated by intermediate functions in the structure tree. The effect is polyphonic.
- **Sequencer**: produce a looped short sequence of notes. To create a new sequence, click the small die.
- **Sine drones**: the system surface modulates the pitch of a sine wave whose frequency is specified by the tuning key in the [Preferences](#) window. This mode provides sounds that are harmonically less rich than the other modes, and the pitch by be detuned when the amplitude of the generated sound is high.
- **Direct drones**: Generates a single continuous waveform by traveling along a straight horizontal line (rather than circling) over the system surface. Essentially, this turns the structure tree into an oscillator whose waveform is determined by the output value (which is interpreted as amplitude). This mode works only for systems that are periodic and defined everywhere on the plane. If the system is non-periodic (such as those with the facet and mosaic functions), ArtMatic will use a circular trajectory similar to the **Drone** mode. Unlike the **Drone** mode, the output is not processed through a sine filter so care must be taken to avoid very high values and to avoid systems that have only positive values. A 1D component can be added at the bottom of the system to limit the final volume. This mode has the most interesting musical applications, **but** systems must be specially designed for it to get the best results. See the provided example files.

Menu Commands

File Menu

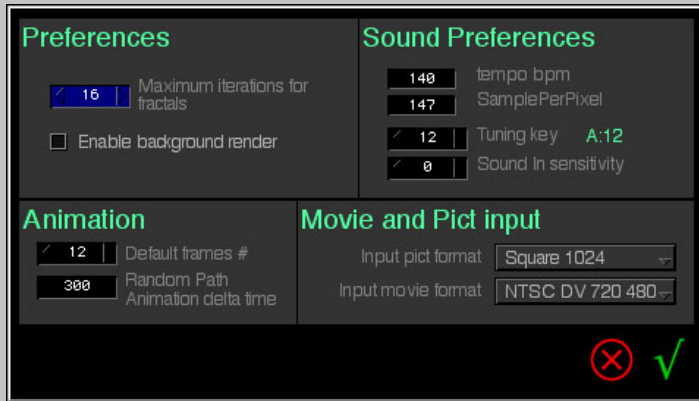
- **New** (command-N) - Create a new empty file with a randomly generated structure tree and gradient.
- **Open** (command-O) - Open an ArtMatic file. Also see .
- **Save** (command-S) - Save an ArtMatic file. Also see .
- **Save As** - Save the current file under a different name.
- **Save Picture** - Export the current canvas as a high-quality anti-aliased picture file. This is the same as clicking the [save pict](#) tool in the upper toolset.
- **Import Compiled** - Create a new ArtMatic system by importing a compiled tree. If the compiled tree has keyframes, they are imported. Learn about compiled trees in the chapter [Compiled Trees and Iteration](#).
- **Export Compiled** - Export the current system and its keyframes as a compiled tree which can be used by other ArtMatic files. A system can only be exported if it does not already contain a compiled tree. Learn about compiled trees in the chapter [Compiled Trees and Iteration](#).
- **Export Movie** - Export a QuickTime animation. This command performs the same action as the [QuickTime Export](#) tool  in the lefthand tool bar. See the [Animation](#) and [QuickTime Notes](#) chapters for more information about animation and QuickTime export considerations.
- **Quit** (Command-Q) - Quit the program. This feature is seldom used as quitting is the most difficult task when using ArtMatic.

Edit Menu

- **Undo** (command-z) - Undo the last change.
- **Copy Parameters** (command-x) - Copy the values of all system parameters to the clipboard.
- **Copy Pict** (command-c) - Copy the canvas image to the clipboard.
- **Paste Parameters** (command-v) - Paste copied parameters into the current system. This command assumes that the structure tree is the same as that from which the parameters were copied. Copy/Paste of parameters essentially copies/pastes a keyframe.
- **Clear**
- **Mutations (command-m)** - Invoke the Mutations dialog just as when you click on the [mutations](#) tool in the upper toolset.
- **Preferences** (command-p) - Invoke the Preferences dialog.

The Preferences Dialog

The Preferences dialog is where you set up a number of ArtMatic's settings. Most of these are used as default settings for new files as well as for the current file.



Maximum iterations for fractals - Determines the number of iterations used by fractal and iterative functions. The larger the value, the higher the resolution of these functions when zoomed in. Note that with large values images can take quite long to calculate. Setting this value high, lets you zoom far in on a fractal to discover the lovely details found at the "microscopic" level.

Enable background render - Turns on background rendering for QuickTime export. When this option is turned off, ArtMatic turns off the system event loop and completely takes over the computer during rendering. When this option is turned on, you can switch out of ArtMatic during rendering by clicking in the menubar

(which will also hide ArtMatic). *When using this option, there is a trick (described below) which will dramatically improve rendering speed.* To bring ArtMatic back to the foreground, choose it from the system menu located at the right side of the system menubar. Background rendering is great if you want to start rendering but can't leave it tied up with rendering for hours or days. Rendering will be somewhat slower when this option is on.

IMPORTANT TIP FOR BACKGROUND RENDERING! After starting the render, it will proceed fastest if you switch out of ArtMatic (by clicking in the menubar) and then bring ArtMatic back to the front. This may sound bizarre, but it is true. If you do this, rendering proceeds almost as fast as if background rendering were turned off.

Default frames - This is the default number of frames that will appear in the QuickTime export dialog. This value affects both the duration of sounds and keyframe animation. When previewing animation, the actual preview time may not be accurate because the computation time for systems is highly variable, but when rendering systems the time will be exact.

Animation delta time - Controls how fast an animation moves along the [random path](#).. The smaller the delta time, the faster the animation will be.

Sound Preferences - These settings apply to **Sound** mode and affect the way that an animation is translated into sound (see [Sound](#)).

Tempo bpm/SamplePerPixel - This tempo is used when sequence or rhythmic modes are chosen on the Sound page. The option to set tempo in samples per pixel is handy for generating a sequence to be synchronized with MetaSynth.

Tuning key - This will be the reference pitch when the view is centered. Change this value to change the pitch of the generated sound.

Sound In sensitivity - this setting determines how responsive the animation is to the sound input when [Audio In control](#) is used.

Movie and Pict Input Preferences - Movie and picture input is scaled to this size when used in a system. Read more about movie and picture input in the chapter [Using Pictures and Movies](#).

Animation Menu

- **Add Keyframe** - Add the current canvas as a keyframe.
- **Replace Keyframe** - Replace the selected keyframe with the current canvas. To directly replace a keyframe, command-click any keyframe to replace it with the current canvas.
- **Delete Keyframe** - Delete the selected keyframe.
- **Insert In-between Keyframe** (Command-I) - Insert a new keyframe after the selected keyframe. The new keyframe's parameters will be in between the selected keyframe's and the those of the following keyframe. This command can be useful for slowing down a particular passage and can be useful for creating coherent animations when used as described in the following recipe.

Recipe: Define a starting keyframe (which we will call K1). Define an end position and add it as a second keyframe (which we will call K2). Select K1 and choose **Insert In-between Keyframe**. Select the new second keyframe and modify it parameters slightly to yield a more complex path from K1 to the new frame. Click on the **Replace** text button to save the new settings for the second keyframe. Repeat these steps as many times as you like.

- **Guess Next Keyframe** (Command-G) - Generate a keyframe that continues the motion of the previous keyframes. This command is only available when the last keyframe is selected.
- **Copy Camera Path** - Copy the current camera path to the clipboard so that it can be pasted into another file.
- **Paste Camera Path** - Paste a copied camera path onto the current system.
- **Edit Camera Path...** - Edit the [camera path and position](#). Read more about the camera path and this dialog in the [Animation](#) chapter.
- **Preview** - Perform a high-quality mini-preview of the current file's animation. To stop the preview, press and hold the escape key or the mouse button until the preview stops. This may take a while with complex systems.
- **Input Movie Setup** - This dialog box provides access to the settings for any QuickTime movies used by the open project. Each movie in the project can have its own settings.

Input Movie Setup



Input Movie Selector - Select the movie whose settings you want to alter by choosing it from this popup menu.

Use movie's own time - This setting determines whether the input movie's duration is scaled or not. When this option is turned on, the input movie plays back at its own rate. When the option is off, the duration is scaled to the project's duration.

Play only at render time - Use this setting to turn off realtime playback of input movies.

Real-time playback can be very slow. So, it is often useful to turn it off while experimenting with a project.

Start movie at - Enter the position within the input movie where it should start playback. You can click anywhere in the time scroller to view different positions within the movie.

Set start time - Click this button to set the start time to the time being viewed.

- **Batch Render Recent Projects** - Render the projects in the Recent Projects menu as QuickTime movies. To remove items from the Recent Projects menu, hold down the option key and select the item to remove.

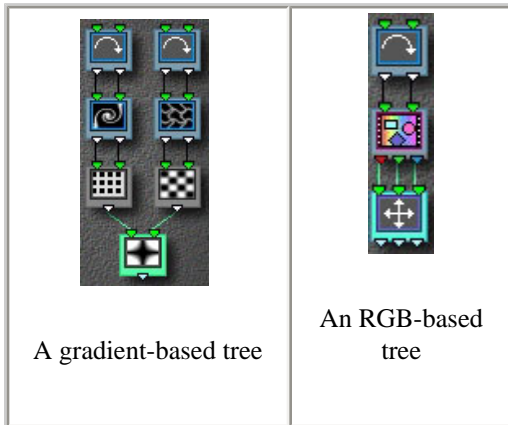
Recent Projects Menu

This is a list of the recently opened projects. The projects listed in this menu will be rendered when choosing the **Batch Render** menu command in the **Animation** menu. To remove an item from this menu, hold down the option key then select the project to remove (with the option key still held down).

Shaders (Color Shading Functions)

This chapter describes the **shading options** which apply to all ArtMatic systems and the **color shading algorithms** which determine the way that gradient-based systems are displayed. The material in this chapter is necessary for understanding how systems are drawn. These features are explored in the [Shading Tutorial](#) found in the [QuickStart 3](#) chapter of this manual and the [Getting Deeper](#) chapter as well.

Gradient-Based versus RGB-Based Systems




As noted in the [Concepts](#) chapter, ArtMatic has two types of trees: gradient-based and RGB-based. The tree's type is determined by the number of inputs and outputs of the tree's last tile.

Gradient-based trees

Gradient Editing Tools

Active Gradient Display/
Set Colors Tool



Edit Gradation Tool

Choose Colors
(Gradient Picker)

These [gradient tools](#) allow you to edit the gradient (palette/gradation) used to color gradient-based systems.

Gradient-based trees have a one or two output tile in the last row of the tree and are drawn with the colors of the active [gradient](#). How the output values are mapped to gradient colors is determined by a **color shading algorithm** (sometimes referred to as a shader or a color shader). If the final tile has two outlets, the two output values are treated as two independent gradient-based images which are

combined before the final color shading is applied.

RGB-based trees

RGB (RGB = red, green, blue) or "true color" trees have a three-output tile in final position whose output determines the color directly with the left, middle and right outlets determining the red, green, and blue values respectively. RGB-based structures will almost always have a component in the tree that maps a point's position to an RGB color value. The [RGB Pict/Movie](#) component, for instance, returns the source picture's color at the point defined by the component's inputs. There are RGB-based components that perform a large range of functions including texture generation, alpha masking, color mixing, and color space conversion.

RGB-based trees are often (but not exclusively) used to create special effects for processing picture and movie sources.

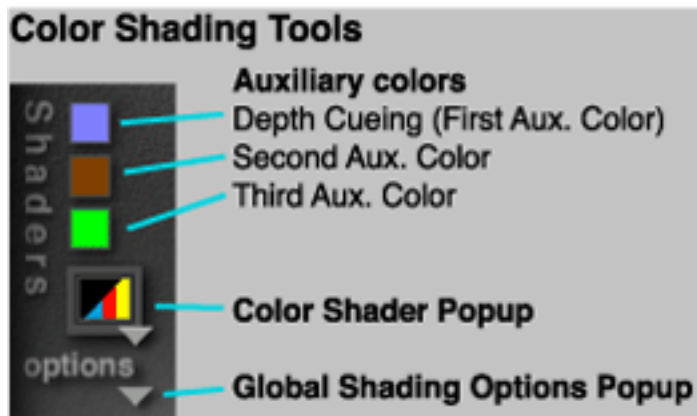
Combining gradient-based and RGB-based systems

There are a number of special components that allow you to create trees that mix gradient and RGB-based color. Most of these functions are [4-in/3-out](#) components (where the left three inputs are treated as RGB input and the fourth input is treated as gradient-based). There are also [2-in/3-out](#) components such as [RGB main gradient](#) and [Shaded main gradient](#) that generate RGB values from the active gradient. Many of the examples use these tools. The **RGB and ArtMatic shading** preset in the [Structures](#) popup menu mixes RGB-based and gradient-based shading.

Global Shading Options

In addition to the colors generated directly from the tree, there are global shading options (discussed later in this chapter) which influence both types of trees.

Color Shading Algorithms (Gradient Color)



A system whose last tile has one or two outlets is drawn using gradient-based coloring. The **color shading algorithm**, which is chosen from the **Color Shader Popup**, determines how the tree's output values are mapped to the colors of the active gradient and whether any of the three auxiliary colors are used. Some shaders also change the settings of the global shading options. The image generated by the structure tree can change dramatically when the shading algorithm is changed.

Complex vs. Simple Shaders

The first three shaders found in the Color Shader Popup are called **simple** shaders because only the final output value of the structure is used to determine a point's color. Simple shaders do not turn on the **global shading options** (discussed later in this chapter).

Complex shaders, on the other hand, calculate a base image using one of the simple shaders and further modify the shading using one or more auxiliary colors and/or by using components other than the final one to influence the shading. Complex shaders may also turn on one or more of the global shading options.

Auxiliary Values and Colors

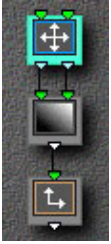
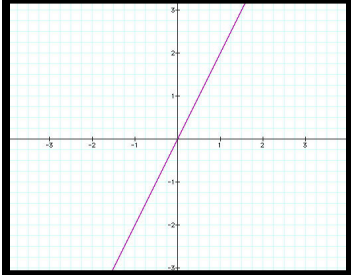
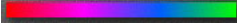

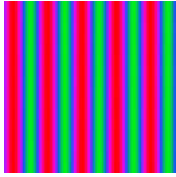
Some complex shaders use **auxiliary colors** to influence how the image is drawn. Clicking on any of the auxiliary colors (when they are active) pops up a color picker. The first auxiliary color is also called the depth cueing color (depth cueing is covered later in this chapter). Its square is always active even if the color isn't used by the current shader (because ArtMatic uses this color for the value infinity when it is encountered in RGB systems). The other color squares are only active when the color is used by the active shading algorithm.

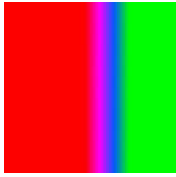
When auxiliary colors are used, a value in the tree other than the final output value is used to determine how an auxiliary color should influence the point being drawn. Each auxiliary color is associated with a value other than the output of the tree's final component. These values are called **auxiliary values** or **auxiliary inputs**. The precise algorithm by which ArtMatic chooses the auxiliary inputs is complex; in some cases, the automatic assignment can be overridden (discussed later in this chapter). When one auxiliary color is used, it is associated with the leftmost input of the tree's last two input tile. When two auxiliary colors are used, the left and right inputs of the last two-input tile are associated with them. When three auxiliary colors are used, the third color is associated with an input higher up on the tree.

Some algorithms use these auxiliary inputs to influence the image without making use of the auxiliary colors. For instance, an auxiliary input might be used to modulate the brightness of the image's pixels rather than the colors.

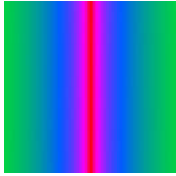
Color Shading Algorithms

The table below shows the same system (with the same parameter values and gradient) drawn with each of the available color shaders. The pictures were created using the file **Simplest Case Color** file found in the documentation examples folder. It is a color version of the system used in the [Concepts](#) chapter. This system is essentially the same as the function $z = ax + by + c$ (where $a = 2$; $b = 0$; and $c = 0$) which is a tilted plane.

 <p>Simplest Case Color structure tree</p>	 <p>At left is a 2-D graph of this function as viewed from the side. Recall that in ArtMatic the canvas looks straight down on the surface whereas the graph shows the surface as viewed from the side.</p>  <p>The gradient used for the renditions in the table.</p> <p>As you examine the images in the table, keep in mind that the values generated by the tree are very small to the left and increase to the right (as you can see from the graph). The output values are the same in all of the examples. What is different is how ArtMatic assigns a color to a particular value.</p>
SIMPLE SHADERS	
	 <p>Cyclic clut - This shader causes the colors to cycle throughout the value range. It applies a sine function whose output value is mapped to the gradient. Zero is mapped to the gradient's central color.</p> <p>Note: <i>Clut</i> stands for color lookup table.</p>

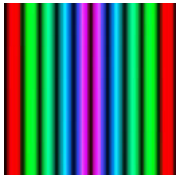


Ceiled linear clut - This shader uses a linear scale to map incoming values to the gradient. 0 is mapped to the gradient's central color. The shader has a ceiling value above which the incoming values are mapped to the rightmost color. There is also a floor value. Values below the floor are mapped to the leftmost color.



Logarithmic clut - This shader is symmetrical about zero (i.e. -5 and +5 map to the same color) and the transition between colors becomes more gradual as the incoming values increase. Mathematically-speaking, the logarithm of the absolute value of the tree's output value is mapped to the gradient. 0 selects the leftmost color.

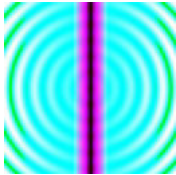
COMPLEX (COMPOUND) SHADERS



Procedural A
Base shader: logarithmic color shader
Auxiliary shading: pixel brightness determined by sine function
Auxiliary colors: none

This shader combines logarithmic and cyclic shading. The **Logarithmic Clut** algorithm determines the pixel hues. ArtMatic then imposes shadows by applying a sine function to vary the pixel brightness. Where the sine function is 0, the image is black and where the sine function is 1 the color is unchanged. The result is a sort of 3D cylindrical banding.

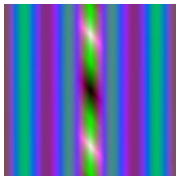
To make the most of this shader, use a two input tile for either the final or the penultimate tile. Watch how the shading changes as you manipulate that tile's inputs.



Procedural B
Base shader: logarithmic color shader
Auxiliary shading: aux. input 1 influences luminance, aux. input 2 determines color shift
Auxiliary colors: aux. color 2

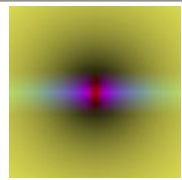
This shader uses two auxiliary inputs from the tree to modulate an image calculated using a logarithmic shading algorithm. The first auxiliary input provides luminance information and the second provides a color shift using the second auxiliary color. **Note:** the first auxiliary color is ignored by this shader.

The auxiliary color used for this example is



Procedural C
Base shader: logarithmic color shader
Auxiliary shading: aux. inputs one and two create a second image using a cyclic shader. Aux. input three controls the mix of the base and second image.
Auxiliary colors: none.

This algorithm calculates two intermediate images which are mixed together under the control of the third auxiliary input. The first intermediate image is calculated from the system using a logarithmic algorithm. The second intermediate image is calculated by feeding the first two auxiliary inputs into a sine-based shader. The third auxiliary input (from god knows where on the tree) controls the interpolation (mixing) of the two intermediate images.



Aux. colors
for sample
image.

D: Log + Depth Cueing + Color Filters

Base shader: logarithmic color shader

Auxiliary shading: color filtering using aux. colors and inputs

Auxiliary colors: all

Global shading options: Depth Cueing

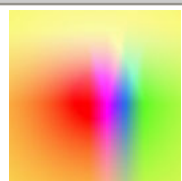
This shader calculates a base image using logarithmic shading, turns on depth cueing (discussed later in this chapter) and uses the second and third auxiliary colors to filter the image.

Two color filters (whose colors are provided by the second and third auxiliary colors) are calculated using two auxiliary values from the tree and applied to the base image. The filtering is done by multiplying the pixels of the filter with the pixels of the base image.

How are pixels multiplied? ArtMatic (like most computer graphics programs) defines a color as a mixture of red, green and blue (because computer monitors use red, green and blue dots to display color) and, as a result, a color image can be thought of as three images red, green and blue which are overlaid. So, for each pixel in an image there is a red, green and blue value associated with it.

To multiply the pixels of two images (of the same size), you multiply the red, green and blue values of the corresponding pixels. Consider two pixels, P1 and P2, whose rgb values are (0, 5.0, 8.0) for P1 and (2, 0.1, 0) respectively. The result of the multiplication is a pixel whose values are (0, 0.5, 0).

Black is the result of all three values being 0. White is created by having maximum red, green and blue values. So, multiplying by a black pixel results in black.



Aux. colors
for sample
image.

E: Linear+Depth Cueing+Lights

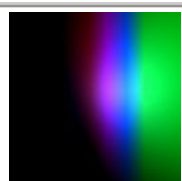
Base shader: logarithmic color shader

Auxiliary shading: color filtering using aux. colors and inputs

Auxiliary colors: all

This shader is based on linear shading and operates similarly to shader D except that the auxiliary color images are combined with the base image using addition rather than multiplication. This shader also turns on depth cueing.

Shading options: Depth Cueing



Aux. colors
for sample
image.


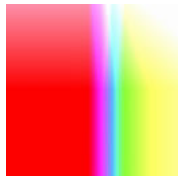
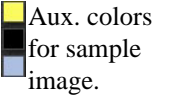


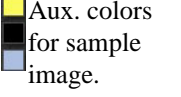

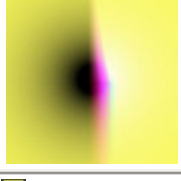
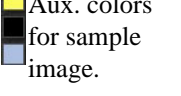

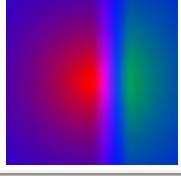
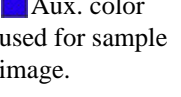
F: Linear+Directional Lights

Base shader: linear color shader

Auxiliary shading: directional lights controlled by the system's last vector function plus black shadows to emphasize lighting direction.

Auxiliary colors: aux. colors two and three

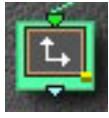
This shader creates directional lighting effects. The last vector function (i.e. the last function with two outputs) of the structure tree provides direction information for directional lights used to shade the image. A primary white light is shined on the surface while lights of the three auxiliary colors are shined from different directions. Black shadows are provided which emphasize the lighting effect.

	  <p>Aux. colors for sample image.</p>	<p>G: Linear+3 Lights Base shader:linear color shader Auxiliary shading: Auxiliary colors: aux. colors two and three</p> <p>This shader is based on linear shading. Three pre-defined slots from the structure tree are used to provide lighting information using three auxiliary colors. The rules for determining which auxiliary inputs are used are complex. If there is a three-input component then those inputs are used to add the auxiliary colors. If there is no three-input component then the last three inputs on the tree are used. If the auxiliary colors are all black then the image will be identical to one shaded with the ceiled linear clut.</p> <p>Tip: To see how this shader works, use an all black gradient and note the effects of changing the auxiliary colors.</p>
	  <p>Aux. colors for sample image.</p>	<p>H: Global Shade+3 Lights Base shader:linear color shader Auxiliary shading: aux. inputs provide shading and lighting information Auxiliary colors: all Shading options: Global Shade On</p> <p>This is another complex shader based on linear shading. The Global Shade option--discussed later in this chapter--is turned on which uses an auxiliary input to control the luminance (brightness) of the images pixels. Output of the final component provides global illumination which is multiplied against the linear-shaded image. Three auxiliary colors provide additional lighting.</p> <p>Tip: For many structures, this can yield dramatic three-dimensional effects especially when the last component in the tree is the derivative (dx) function.</p>
	  <p>Aux. colors for sample image.</p>	<p>I: Global Shade+Lights+Depth Cueing Base shader:linear color shader Auxiliary shading: aux. inputs Auxiliary colors: all Shading options: Depth cueing on</p> <p>This is another complex shader based on linear shading. It uses the last vector output function to provide depth cueing information and two other aux. inputs which control the application of auxiliary colors two and three.</p>
	  <p>Aux. color used for sample image.</p>	<p>J: Linear zero-based+Depth Cueing Base shader:logarithmic color shader Auxiliary shading: Auxiliary colors: depth cueing color (aux. color one)</p> <p>This shader uses is essentially equivalent to the linear shader with depth cueing turned on.</p> <p>Shading options: Depth cueing on</p>

Shading Options

In addition to the color shading **algorithms** (which are only relevant to gradient-based systems), there are two powerful **shading**

options which can be applied to all systems: depth cueing and global shading. Depth cueing provides fog and distance effects and global shade provides lighting and shadow control. Both of these options can be assigned either automatically by ArtMatic or manually by the user. These options are accessed via the [Shading Options](#) popup menu.



Component marked with depth cueing glyph

Depth cueing. Depth cueing simulates the color-filtering effect of the atmosphere (the same effect that causes distant objects to appear faint). Depth cueing works by assigning a color (the first auxiliary color) to be associated with depth/distance. By default, ArtMatic picks the component in the tree associated with distance (generally the first auxiliary input). but this assignment can be manually overridden (as explained below). This color is superimposed on the image in relationship to the depth /distance. As the distance increases, so does the brightness and opacity of the depth color. ArtMatic now provides direct manipulation of depth cueing which can be used for both distance effects and purely decorative effects. Previously, there were several algorithms whose names contained the word **fog**. To be consistent with the new **shading options**, the shading algorithm names have been changed to use the more-technically-correct **depth cueing** which refers to a technique that simulates depth/distance effects.

Depth cueing color note. The depth cueing color (the first auxiliary color) is now always visible/available even if Depth Cueing is turned off. This color is used in RGB-based systems wherever the value **infinity** is encountered. Infinity is used for the areas outside the boundaries of 3D objects and where the **infinity gate** component is used.



Component marked with global shading glyph

Global shading (shadows). Another new shading option is **Global Shading** (or shadowing) that can be turned on via the options menu. Global shading creates shadows and light in the image by modulating the luminance (brightness) of the image's pixels. Where the shading component generates low values, the image will be dark (shadowed), and where it generates large values, the image will be brighter. As with depth cueing, you can now choose the component that provides the shadows (global shading) as described below.

Note: The global shading glyph is the small marking in the icon's lower-right corner.




Controlling depth cueing and global shading. Previously, these effects were built directly into the shading algorithms, and ArtMatic automatically picked the components it used to provide the depth cueing and global shading. Now, these effects are turned on and off via the **shading options** pop-up, and you can override the automatic selection and choose the components used to provide the effects. It is even possible to have branches of your tree whose only purpose is to provide depth and shadow effects. These options are available regardless of the active shading algorithm. To **turn on depth cueing or global shading**, pop up the options menu and choose a depth cueing or global shading option. To turn the options off, select either **Depth Cueing Off** or **Global Shading Off**.

Automatic or manual control. By default, ArtMatic selects the components that control the depth cueing and global shading. You can override the automatic selection by choosing **Component Sets Depth** or **Component Sets Shade** which causes the selected tile to be used to provide the effect. A small shaded glyph (as shown in the illustrations above) is used to indicate components that have been manually assigned. **Restoring automatic shade/depth control.** If you have used the **Component Sets Depth** or **Component Sets Shade** commands and want ArtMatic to restore its automatic depth/shading choice, select the **Fog and Shade Automatic** item from the **shading options** popup menu. Note that **Fog and Shade Automatic** may have a check mark even if depth and shading are turned off. The check mark merely indicates that if shading/depth are turned on that ArtMatic will choose the components automatically.




Shortcuts. Two keyboard shortcuts have been added to facilitate using these new features. Type **s** (not command-s) to turn on global shading and assign the selected component for use as the shading component. Type **d** to turn on depth cueing and assign the selected component the depth cueing function.

Tutorials. These features are explored in the [Shading Tutorial](#) found in the [QuickStart 3](#) chapter of this manual.

Using Pictures and Movies

		
<p>This picture was used as in input source for each of the other images in this table and the next.</p>	<p>Logo processed with <i>FireChrome</i> example file.</p>	<p>Logo processed with <i>FireChrome</i> example file.</p>

In addition to synthesizing images and animation, ArtMatic can use up to four pictures and movies as input sources that can be processed by the structure tree. This capability is useful for an astounding range of effects and applications that will prove indispensable to both graphic and video artists. It can be used to create weird and wonderful effects and for simple but powerful effects that are difficult even with high-end graphics and video post-production tools. This chapter only touches the surface of what can be done. An entire book could be written about the possible applications. We have supplied a large number of examples in the examples library that we recommend perusing as they will provide both templates that you can use as starting points and insights about the applications of picture/movie input.








		
<p>Triple Pict Projections example file.</p>	<p>Burning Logo RGB example file. Note that the flames are synthesized by ArtMatic.</p>	

When movies are used as input sources, incredible video effects are possible since ArtMatic takes consecutive frames from movie inputs when it animates the system. This makes very complex fades, wipes, transitions, distortion effects possible as well as other remarkable effects that words cannot describe. As you examine the examples library, keep in mind that you can use movies as input anywhere you see pictures used as input.

The table below displays animation rendered from systems that have pictures or movies as input sources. To play an animation, double-

click its image.

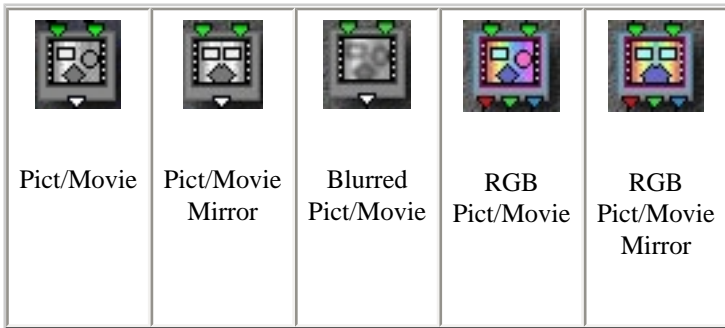
Please note that due to storage and download considerations, the examples are highly compressed animations whose resolution is quite a bit lower than the animation you will render.

<p>Picture Input Animation. The examples in this row are animations that use pictures as inputs.</p> 	 <p>Animation rendered from the <i>Triple Pict Projections</i> example file.</p>	 <p>Example rendered from the <i>Burning Logo RGB</i> example system.</p>	 <p>Example rendered from the <i>CPict FractSand Wipe</i> example file. Of course, the color logo could be replaced by a movies as an input source to create a dramatic fade-in.</p>
<p>Movie Input Animation. The examples in this row are animations that use QuickTime movies as inputs.</p>	 <p>Synthesized reflections. The source movie was a simple movie of the moon and clouds with no water! The water and reflections are completely synthesized by ArtMatic. See the <i>Reflection Waves</i> example file.</p>	 <p>Color matrix multiply. This example uses the same source movies as the example on the left (notice that there isn't any water in this one) and uses the <i>Color Matrix Multiply</i> example to provide HLS manipulation.</p>	 <p>Synthesized tunnel with movie walls. This example uses a movie of a rushing river as the input source. The <i>CPict 3D Tunnel Arch</i> example file was used for the processing.</p>

Pict/Movie Basics

Pict/Movie Components

Movies and pictures are inserted in systems by using the 2-in/1-out and 2-in/3-out Pict/Movie components whose icons are shown below.




All of the Pict/Movie components behave similarly. The tile's two input values are treated as the x and y (horizontal and vertical) coordinates of a pixel (point) within the picture or movie, and the output value is either the brightness (luminance) or RGB color of the corresponding pixel. Any spatial distortion introduced by the system will effectively distort the image--a property exploited in many of the provided example files. If the x and y values are beyond the image boundaries, the image may be tiled if the component is one that has a tiling parameter.

1-out Pict/Movie components. The 1-out components return the pixel brightness (luminance) which is eventually mapped (in gradient-color systems) to a color of the active gradient. If the gradient is black and white, the result is a grayscale version of the input source.

3-out (RGB) Pict/Movie components. The 3-out Movie/Pict components are for RGB output and color processing. They return the red, green, and blue values of the image's pixels. This makes it possible to access the picture's or movie's original colors and also to perform color processing since you can modulate the red, green and blue picture channels independently. It is also possible to convert the RGB values to HLS (hue luminance saturation) values for sophisticated color effects.

Input Sources

Default Picture. Any picture file or QuickTime movie can be used as an input source. By default, the *DefaultPicture* PICT file found in ArtMatic Pro's home folder is used for pict/movie tiles. You can change the default picture by moving any PICT format file into ArtMatic Pro's home folder and renaming it DefaultPicture.

Choose Pict/Movie tool. The **Choose Pict/Movie** tool  can be used to choose and change the input pictures and movies. To add a movie or picture to your project, click on the tool to pop up the picture/movie selector. Select any slot marked Open to bring up the file selection dialog box that allows you to choose any picture or QuickTime movie. To replace a picture or movie that is in use, hold down the option key and select the item to replace from the Choose Pict/Movie popup. The file selection dialog will be presented to choose the replacement.

Assigning input sources. To assign an input picture/movie to a Movie/Pict component tile, click on the tile to make it active then select the desired input source from the **Choose Pict/Movie** tool's  popup selector.

A note about input picture formats. For the best quality results, use PICT format files as picture inputs as their quality is much higher than jpeg images. If you have enabled the system's QuickTime translation options, any graphic file should be available in the file selection dialog.

Picture & Movie Input Buffers

ArtMatic loads movie frames and pictures into a memory buffer and scales them to the dimensions indicated in the [Preferences dialog](#) which provides different buffer dimensions for picture and movie inputs. Generally, you want to set the buffer dimensions to the same dimensions as the picture or movie (since the image will be scaled to fit the buffer). For non-square-pixel input movies, you may want to choose the square pixel equivalent depending on how you plan to use the movie input.

If you are going to use ArtMatic Pro for high-resolution image-processing, you may want to increase ArtMatic's memory allocation in the Macintosh Finder. If you are using still images as inputs for systems that will generate still images, you should use an input picture with large dimensions since this allows ArtMatic to perform high-quality anti-aliasing and will reduce scaling artifacts.

Parameters

All of the Pict/Movie components have the same A & B parameters (**scale** and **contrast**) with a third parameter, C, that varies by component. The **scale** parameter acts as a zoom control. The **contrast** parameter adjusts the contrast of the output values.

Keeping Track of Referenced Files

ArtMatic Pro does its best to track input source files. In some cases, ArtMatic will have to search for an input file (generally because it has moved) when a system is opened. When this happens a message, **Searching for File**, appears in the Tool Tips area. You can cancel the search by pressing the Escape key.

In general, we recommend that you keep the ArtMatic file and the picture/movie file it references in the same folder. You can also keep an alias to the picture or movie in the same folder as the ArtMatic file. If you keep an alias with the ArtMatic file, the alias should have the same name as the file itself (i.e. you should remove the " alias" sometimes added by the system). A **handy shortcut** for creating such an alias (in system 8.1 and later) is to hold down the command and option keys and drag the file icon to the desired location. This action creates an alias rather than moving/copying the file.

Movie Input

By using movies as inputs, ArtMatic can be turned into an effects generation tool for amateurs and professionals alike. Amateurs can use ArtMatic to create astounding transitions and effects previously only been available in high-end video effects software. Pros can use ArtMatic to design their own video effects and to easily achieve effects that would otherwise be difficult to create with other systems.

Movie Time Scaling and Start Position

When a movie file is used as an input for rendered animation, ArtMatic will take consecutive frames from the input file as it generates the frames of the animation. The Animation menu's [Input Movie Setup](#) command provides control for the start time within the movie and whether the input movie's duration is scaled or not. When **movie's own time** is selected, the movie's time is not scaled. If an animation is five seconds long, five seconds of the source animation will be used. If **movie's own time** is turned off, ArtMatic scale's the input movie's time to the rendered animation's duration. If the rendered animation's duration is shorter than the source's, for example, the content derived from the source will appear to play back faster than the original (great for creating pseudo-time-lapse animation)

Applications

There is a huge range of possible applications for pict/movie input--so many that we have only touched the surface in our own explorations. We have provided many examples of pict/movie input in the examples library and recommend that you examine them. Be sure to render some of the examples since the real-time preview can only hint at the astounding beauty of the rendered animation.

Here are just a few examples of pict/movie input applications:

- outrageous transition effects,

- dynamic color effects,
- displacement and distortion effects,
- video mixing and keying effects,
- morph inputs to/from ArtMatic textures,
- and a wealth of mindblowing transforms that have yet to be explored.

Search Algorithm

When searching for input pics and movies, ArtMatic starts the search with folders within the document folder then searches all folders one level higher than the document folder. If the movie/pict is still unfound, ArtMatic performs a global search on all disks. (To abort the search, press the escape key.)

Animation & QuickTime Movies

ArtMatic Pro can create astounding high-quality animation in addition to still images and sound. Not only can ArtMatic Pro create new QuickTime movies, but it can also be used as a special effects and video processing tool for existing QuickTime, iMovie and DV content. When animation is rendered, every single frame has the same level of detail and anti-aliasing as still pictures. ArtMatic animation can be breathtakingly beautiful and complex, and it can even be modulated by the sound track of your choice. Artists use ArtMatic animations both directly as video content and also as alpha (transparency) masks that are used to create extraordinary wipes, fades, and dissolves in their video projects.

Several types of animation are available keyframe-based, random path and realtime animation. All realtime animation is done at a low resolution due to the computational complexity of ArtMatic systems. Animation is rendered to QuickTime at high-resolution and each frame is fully anti-aliased.

The ArtMatic CD contains many examples of animation rendered with ArtMatic. Even if you think that you are not interested in using ArtMatic for animation, we recommend that you view these examples or render some of the example files that come with the download so that you can see the extraordinary beauty of ArtMatic animation.

Overview

How Animation Is Generated

ArtMatic creates animation by changing a system's parameters over time. Recall that an ArtMatic system is made up of interconnected components each of which has up to 3 parameter slider which can be modified directly or via the parameter envelope dialogs (discussed below). To understand how animation is generated, it is useful to imagine a mixing board with one slider or knob for each parameter of each component tile. During animation ArtMatic adjusts some or all of these sliders and creates movie frames as the settings changes. As you can imagine, this process can generate extraordinarily complex and fluid animation like nothing you have ever seen.

ArtMatic has two basic types of animation: keyframe and random path.

Keyframe Animation. Each keyframe represents a snapshot of all the component tiles' slider settings. When keyframe animation is created, it is as if an invisible hand is moving each slider from the settings of one keyframe to the settings of the next keyframe. The rate at which the changes take place is determined by the movie's duration. Time is evenly distributed between the keyframes. When exporting a movie, you can choose either linear keyframe animations or looped keyframe animation. Looped keyframe animation creates a loopable movie that uses the last keyframe as the midpoint of the movie and then interpolates a path to the return so that the movie begins and ends at the same point.

Random Path Animation. All of the system's parameters are modulated according to a pre-defined pseudo-random path designed to help you explore a wide-range of a system's possibilities. **Tip:** Random Path animation is often useful for finding keyframes. If you press the Random Path Animation button, you can use option-spacebar to pause and continue the animation. When the animation is paused, you can click the **Add** button to save the current image as a keyframe. The rate at which the animation changes is determined by the **Delta Time** setting which is found both in the [Preferences](#) dialog and in the [QuickTime Export dialog](#).

Shortcut: To start/pause/continue random path animation playback, press **option-spacebar**.

Previewing animation

There are a few different methods for previewing animation in real time with which you should be familiar. In all cases the preview will be lower resolution than rendered animation. (Because many ArtMatic systems are computation-intensive, it isn't possible to create full-resolution real-time previews). To stop the preview, press and hold either the mouse button or the escape key until the preview stops.

- **High quality mini-preview** - The best quality preview is achieved by typing command-h or choosing **Preview** from the Animation menu. This plays a reduced-size preview of very high quality. The Preview command can preview either keyframe or random path animation. By default, Preview displays keyframe animation. To preview random path animation, press the Random Path Animation Button then stop the animation.
- **Full-size (low res) preview** - Press the Animate Keyframes or Random Path Animation buttons to play a preview that fills the Canvas region. To accommodate this large size, the preview is done at a fairly low-resolution.
- **Audio-in Control** - this option is another full-size, low-resolution preview that uses the Sound Manager sound input to modulate the system in real time. This sort of previewing is generally done to get an idea of how a system responds to sound input, though it can also be quite entertaining as a realtime fractal color organ. For more about Audio-In Control see the [User Interface](#) chapter.
- **QuickTime Export preview** - A high-res mini preview can be performed in the QuickTime Export dialog by pressing the Preview button.

Preview limitations. While render-to-disk has excellent image and time resolution, realtime previewing has some limitations due to the computationally-intensive nature of ArtMatic systems of which you should be aware. The time accuracy of the preview is quite variable, especially in the mini-previews where resolution and smoothness of motion take precedence over time accuracy. The time accuracy is determined by computer processor speed and the ArtMatic system's computation needs. ArtMatic systems that contain recursive or iterative elements (such as some compiled trees) or some computationally-intensive components (such as the derivative-based components and some fractal components) may not be able to be computed in realtime. With such systems (which will seem quite slow while they are being edited), the best way to get a time-accurate preview is to do a test rendering of the system.

Test render tip. When performing a test render to get a time-accurate preview of the animation's motion, choose a low frame rate (such as 10 or 12 fps), a small frame size (such as 240 or 128 pixels square), and a 'fast' codec to minimize the rendering time. The Animation (with medium quality) and Video codecs are both fairly fast compressors that are convenient for this use. The Video codec only uses 16-bits of color information but is very convenient for these renderings. On the other hand, the Sorenson codec, which is able to create very high-quality compression, is computationally-intensive and not well-suited to this purpose.

Using QuickTime Movies in Animation

As previously mentioned, QuickTime movies and iMovie DV streams can be used as input sources in animations. When movies are used as input sources, ArtMatic chooses successive frames from the source movies, which makes ArtMatic a very powerful effects and video editing tool. This topic is covered in greater detail in the chapter [Using Pictures and Movies within ArtMatic](#).

Animation and Sound

Sound can be used to modulate ArtMatic animations when they are rendered, and a soundtrack can be added or generated (using the sound algorithm last used on the **Sound** page) when rendering. The sound track modulates the animation using the same technique as the realtime [Audio-In Control](#) only with much higher resolution. Audio modulation is a powerful technique to use for creating video for music videos and multimedia compositions.

Sound modulation is a bit of a hit-or-miss proposition at this time because ArtMatic's assignment of audio frequencies to particular parameters is not customizable. You can, however, use the parameter locks to prevent particular parameters from being modulated. The sensitivity is set with the [Sound In Sensitivity](#) preference.

TIP! One trick for incorporating audio modulation is to render a particular sequence once with sound modulation and once without. When editing the movie, insert interesting sections from the audio-modulated movie into the unmodulated movie to emphasize particular moments in the soundtrack. When creating animation for music, save short sections of the soundtrack as individual sound files -- or sections of individual tracks -- and animate these short sections rather than the entire song at once.

Wipes/Masks

In addition to creating content that will be used directly in movie and video projects, ArtMatic animations can be used to create astounding transition effects (wipes, fades and dissolves). There are two basic methods that you can use:

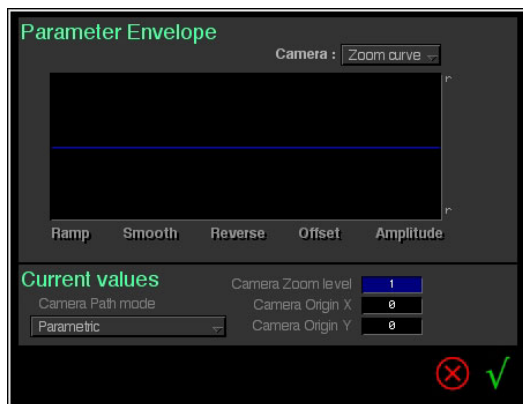
- ArtMatic can use external movie files and pictures as inputs and render the transition itself (which you will then drop into your movie project).
- A black and white ArtMatic animation can be rendered then used as an **alpha or transparency mask** in most advanced movie and video editing applications.

We have provided examples of both types of systems in both the examples library and in the **Doc. Examples** folder. Search the examples directory for files whose name includes wipe or transition. Note that where you see a picture used as an input, you can use a movie in its place.

Creating alpha masks. To create an animation appropriate for use as an alpha mask, you will want to create short animations where the first keyframe is all white and where the final keyframe all black (or vice versa). We have provided several examples of these in the examples library.

Editing Tools

Camera Path & Envelopes



Unlike ArtMatic 1.x, ArtMatic Pro has programmable zooms and offsets. When creating an animation, you can think of the ArtMatic Pro canvas as a camera viewfinder which looks down on the ArtMatic universe. Each keyframe has an associated camera position. (The camera position is not actually stored in the keyframe, however.) During keyframe animation, the camera travels from position to position over the course of the animation.

Zooming in and out on the canvas (using the zoom buttons or the up and down arrow keys) moves the camera position as does clicking and dragging the canvas (which changes the camera offset). Whenever a keyframe is added, the camera position is stored along with it. The camera position consists of three parameters: zoom level (height over the canvas), camera origin x (the position along the x-axis),

and camera origin y (the position along the y-axis). All keyframes contain zoom and offset information. Hence, you can use the zoom/magnification tools and you can drag in the **canvas**

Editing the camera path. The **Camera Path** refers to the path the camera takes over the course of an animation. It can be edited directly and also copied and pasted between projects. The Camera Path dialog provides a means for seeing and editing the camera path directly. To invoke it, chose the **Edit Camera Path** command from the Animation menu. When the dialog opens, the current camera position is displayed in the numerical fields at the bottom of the dialog box. You can type in new values or click on a value and drag up/down to change it. The upper-part of the dialog features a display of the camera path. Choose a position parameter (zoom level, x curve, or y curve) from the dialog's popup menu to edit that parameter's envelope. (An envelope is a graph of the parameter's values over the course of the animation.) Parameter envelopes are discussed in greater detail later in this chapter.

Camera Path Mode. ArtMatic has several different modes of camera motion that can be used to adjust the camera's zoom response in order to compensate for the non-linear response of many systems. Without these modes, zooms into our out of some systems could have undesirable acceleration or deceleration which would be difficult to work around. You can also use the camera path mode to create accelerated/decelerated responses in linear systems and to create other zoom responses that may be desirable. Most of these modes result in adjustments to the time between keyframes. The options are:

- **Parameteric** - Keyframes are guaranteed to be evenly spaced in time, and the zoom rate is determined by the distance traveled which can result in acceleration and/or deceleration over the course of the animation depending on the distance traveled between keyframes.
- **Constant Speed & Zoom** - ArtMatic tries to maintain a constant perceived zoom rate. In order to do this, the time between keyframes may be adjusted so that they are not evenly spaced.
- **Accelerated** - Accelerates the perceived zoom speed over the course of the animation. This mode may adjust the time between keyframes.
- **Decelerated** - Decelerate the perceived zoom speed over the course of the animation. This mode may adjust the time between keyframes.
- **Ease In Ease Out** - The zoom speed gradually accelerates then decelerates, and the time between keyframes may be adjusted. See the example file *Cosmos EaseIn EaseOut*.
- **Constant with inertia** - This mode more or less maintains a constant zoom speed with some added inertia which results in a smoother and more realistic zoom motion. See the file *Mandel Fire Zoom* for an example. Preview it with command-h for the best result.

Copying camera paths. Because you might want to use similar camera movements with different ArtMatic files, you can use the **Copy Camera Path** and **Paste Camera Path** commands to copy and paste camera paths between files.

Rotating the Camera/Canvas. For a number of reasons, rotating the canvas/camera is different than zooming and is not controlled by the camera path. To animate canvas rotation, simply add a Rotation component at the top of the system. Use the **Insert Global Rotation** command (found in the **Insert** popup menu) to add a rotation tile. You can then use **Parameter Envelopes** to animate the **Angle** parameter of the **Rotation** component which will appear at the top of the system.

Parameter Envelopes



ArtMatic provides parameter envelopes that control component parameter values across a project's keyframes. Each parameter of every component in a system can have an envelope. An envelope is a simple interface that shows a parameter's value for each keyframe.

To edit a parameter envelope, simply click on the letter next to any parameter slider which invokes the Parameter Envelope dialog shown at left. The green line represents the parameter's value and the horizontal axis represents time. The left end of the green line is the parameter's value for the first keyframe, and the right end of the line represents the value for the last keyframe. The line is broken by white markers which represent the intermediate keyframes. The numerical values displayed at the bottom of the window correspond to the current parameter settings.

You can directly enter numeric values for the current parameter values by typing a numerical value into any field or by clicking on a value and dragging up or down. You can even enter a value in degrees (useful for rotation parameters). **To enter a value in degrees,** type in the value then type 'd' to have ArtMatic translate the degrees into its native format (radians).

Editing the envelope. The envelope can be manipulated directly by clicking on any line segment and dragging up or down. You can also use any of the tools located at the bottom of the dialog. The tools are:


- **Ramp** - Click on this tool to make the envelope a diagonal.
- **Smooth** - Click on this tool to smooth sharp edges in the envelope contour.
- **Reverse** - Click on this tool to reverse the envelope.
- **Offset** - Click and drag left or right to offset the envelope up or down.
- **Amplitude** - Click and drag left or right to increase or decrease the difference between the start and the end values.

Modifying All Keyframes At Once

Parameter envelopes provide one method for manipulating several keyframes at the same time. It is also possible to use the **shift key** to apply some changes to all keyframes. Below is a list of the operations which can be applied to all keyframes by holding down the shift key:

- **Parameter slider change** (to set the parameter value for all keyframes)
- **Zoom button click** (to change the zoom level of all keyframes)
- **Gradient select/change** (to change the gradient for all keyframes). Note that picking a color rather than a gradient will apply the resulting gradient to all keyframes.

Rendering

To save an ArtMatic Pro animation as a high-quality QuickTime movie, click on the [QuickTime Export](#) tool . The QuickTime Export dialog has a number of options that allow you to set the animation type, frame rate, frame dimensions and compression method. The appropriate settings will depend on how the movie will be used. The [QuickTime Notes](#) appendix covers this topic in some detail and should be read by anyone interested in rendering animation since inappropriate settings can result in poor image quality, long rendering times, or poor playback.

Rendering takes time! You should be aware that the incredible quality of ArtMatic animation comes at a cost. Every single frame is a fully anti-aliased picture which has to be run through a QuickTime compressor when it is saved. Rendering movies can take a lot of time, but you will find that the quality of the end result is worth the wait.

Background rendering. Be aware that rendering can take quite a while. The amount of time is related to the computer's processor speed, the computational complexity of the current system, and the QuickTime Export settings. Fortunately, ArtMatic can render animation in the background. Background rendering can be turned on and off in the [Preferences](#) dialog box. When background rendering is turned on, clicking anywhere in the menubar will hide ArtMatic which will continue to render while it is hidden. To check your progress, you can choose ArtMatic from your Macintosh's Applications menu which is found at the right edge of the menubar.

Important note about background rendering: Due to a quirk in the OS, rendering can be a bit slow getting started when background rendering is turned on. There is a trick that vastly improves background rendering times. After you start rendering, click on the menubar (which will hide ArtMatic). Then, choose ArtMatic from the Applications menu to bring it back to the front. Rendering will now proceed at a much faster pace than before.

Batch rendering. It is possible to batch render a number of ArtMatic projects at once. The Animation menu's [Batch Render Recent Projects](#) command will cause all the projects listed in the Recent Projects menu to be rendered. It is often convenient to quit ArtMatic to clear the Recent Projects menu. You can then select the icons of all the files you want to render and drop them on ArtMatic's icon which will launch ArtMatic and add all the files to **Recent Projects** menu.

Test rendering. While working on a movie, it is often a good idea to first render a movie with a reduced frame rate and frame dimensions so that you can check it out before doing a time consuming full-resolution rendering. This step is important with some systems since the realtime preview of computationally-intensive systems may take longer to playback than the actual rendered animation.

TIP: You needn't render the entire animation. You can cancel a rendering by pressing the Escape key and open the partially rendered movie to have a sense of the animation's motion.

Canceling a render. To abort rendering, simply press the **escape key**.

Compiled Trees & Iteration

About this chapter. This chapter discusses both compiled trees and ArtMatic's looping/iteration features. The chapter assumes that you are familiar with the [Concepts](#) chapter and that you have performed the [QuickStart](#) tutorials. Some of the material in this chapter will be of interest to all users and some of it will primarily interest advanced users. The [Getting Deeper](#) tutorials contain lessons that will help most users to make use of compiled trees. All users should take a look at the example files that we have provided both in the main examples library and in the **Compiled Trees & Recursion** folder that is found in the **Doc. Example Files** folder found in the same folder as the documentation.

Introducing Compiled Trees


Compiled trees are complete structure trees that you export then use as a single tile in another ArtMatic system. The recursion parameter available for some compiled trees makes it possible to design new fractal algorithms. They are a very powerful addition to ArtMatic Pro and have a wide range of applications. Users with some knowledge of computer programming will recognize compiled trees as subroutines which can even be looped.

Uses & Applications of Compiled Trees

Here is a list of common applications for compiled trees:

- Create your own *primitives* - compile your favorite tile arrangements, making it a snap to use them in new structures.
- Create animation *macros* - compile rotation, offset and scaling groups with their keyframes to share consistent motion among your projects.
- Simplify complex trees - create compiled trees from portions of complex structures to aid in analyzing complex structures and building new ones.
- Create new fractal algorithms - some compiled trees have a recursion parameter that allows the creation of extraordinary new fractal images.

An Important Note About Compiled Trees

Compiled trees are represented by the following icon  when used in an ArtMatic system. **Be very careful** not to adjust compiled trees' iteration and recursion parameters unless you understand the implications well. The use of compiled trees allows you to create structures that take a **very very** long time to calculate--especially on slow machines or if the iteration/recursion parameters are adjusted. Each embedded tree is a complete structure in itself, and recursion causes that complete tree to be re-calculated a number of times.

When exploring our example files, you should probably not adjust the **Recursion** or **Iteration** parameters of compiled trees. Each recursion level requires significant additional processing. Increasing the amount of recursion can completely tie up even a fairly power computer in some cases. So, we recommend that you leave these parameters alone unless you are adventurous and patient and have a good understanding of how recursion and compiled trees work.

Compiled Tree Basics

Creating and making use of compiled trees is quite simple. We have provided quite a number of example files that make use of compiled trees and of compiled trees that you can use in your own projects. We recommend that you explore these examples as compiled trees open up tremendous possibilities for all users.

USING COMPILED TREES

To use compiled trees in a system, simply click on any selected tile to pop up the component popup menu and choose the **Open Compiled Tree** component. A dialog box will appear that allows you to select **relevant** compiled trees. A relevant tree is one that has the same number of inputs and outputs as the selected component. Only relevant trees will be visible in the Open Compiled Tree dialog box.

Keyframes and compiled trees. When compiled trees are used as components, their keyframes are imported and blended with those in the parent tree. This is a very powerful mechanism that allows you to build compiled trees that act as animation primitives or macros. If the parent tree has fewer keyframes than the subtree, new keyframes are added to the parent tree and adjusted so that the motion programmed in the subtree's keyframes is preserved. Note that compiled trees do not include camera path information, and, as a result, any movement performed via camera path animation is not stored in the compiled tree. If you would like to have camera-like movement stored in your compiled trees, use Scale and Offset components instead of Camera Path animation.

Editing/viewing embedded compiled trees. When using a compiled tree as a component, the compiled tree is copied into the parent and becomes independent of its original source. The subtree can be viewed and further modified from within the parent. Typing 'e' when a compiled tree component is selected allows you to edit and view the subtree. When you are editing a compiled tree from within its parent, choosing the Export Compiled command from the File menu causes just compiled tree to be exported (along with its new keyframes). There are some restrictions when editing subtrees. A subtree's structure cannot be changed though you may change the components assigned to the subtree's tiles. As a result, all structure editing tools are disabled while editing a subtree. Type 'e' again to leave subtree editing mode.

CREATING COMPILED TREES

Any ArtMatic structure tree (that doesn't itself contain a compiled tree) can be exported as a compiled tree by choosing the **File the Export Compiled** command from the File menu. When a structure is exported as a compiled tree, the structure tree and the file's keyframes are saved as a compiled tree. A compiled tree does not contain any color information (i.e. the gradient and shading algorithm are not part of the compiled tree) or any camera path information. This tree can now be used as a component in any other ArtMatic file.


Re-exporting compiled trees. Sometimes, you will find that you want to re-export a compiled tree after modifying it from within its parent. While editing a subtree (as described in the previous section), it is possible to re-export the tree by choosing **Export Compiled**. Only the subtree is exported when in compiled-edit mode.

Export limitations. There are some limitations on the trees which can be exported. Since compiled trees act as components within ArtMatic structures, a compiled tree must have a number of inputs and outputs which corresponds to possible tiles. For example, ArtMatic does not have 4-in/1-out tiles. Hence, it is not possible to export a tree that has four total inputs at the top and one input at the bottom. If you have such a structure, that you need to export, simply add a dummy component at the end with two outputs (such as the Scale component) that just passes the one value out unchanged and has a second output that serves no purpose. The second input to the dummy function can be left open.

CREATING A NEW SYSTEM FROM A COMPILED TREE

It is possible to create a new ArtMatic system by choosing the **Import Compiled** command from the File menu. This creates a new ArtMatic system whose structure and keyframes are derived from the compiled tree. The new structure and keyframes replace whatever structure and keyframes were present prior to choosing the command. Note that compiled trees do not contain camera path information.

COMPILED TREE PARAMETERS & RECURSION

There are essentially two types of compiled trees: recursive and non-recursive. When a compiled tree is used in a parent tree, it appears as a component with the following icon: . When the icon is selected, there are two parameters available. The name and precise function of the parameters depends on the the number of inputs and outputs the tile has. In all cases, the first parameter controls the amount of effect the tile has, and the second parameter controls the number of times the tile is looped. The parameters for both types of trees are discussed below.

A number of components (discussed later in this chapter) have been added to make it possible to take full advantage of the recursive and iterative capabilities of compiled trees. While the recursive/iterative properties of compiled trees are primarily of interest to advanced users and programmers, the other parameters (**Blend** and **Scale**) are useful to all users.

Recursive compiled trees

Parameter A: Blend

Parameter B: Recursion

Tiles that have the same number of inputs and outputs (or four inputs and two or three outputs, see below) are recursive trees. They may have their output values fed back into the tile inputs. The **Blend** parameter allows you to control the blending of the compiled tree's output with its parent tile's. When Blend is set to 1, the output is derived solely from the compiled tree. When Blend is 0, the compiled tree more or less passes the incoming values through without change. When Blend is between 0 and 1, the output is a blending of the compiled tree's output with the parent tile's output. The **Recursion** parameter, determines the number of levels of recursion (how many times the output is fed back into the tile). By default, the recursion parameter is 0 and is locked because the recursion parameter only makes sense with structures designed to be used recursively. Since recursion causes the tree to be calculated many times, care should be taken when setting the recursion level above 0.

4-in/2-out and 3-in/2-out recursion note. Recursion is handled slightly differently for these tiles than for the others. When the recursion is done, the output is fed back into the lefthand inputs. The righthand input(s) retain their initial value(s) during the recursion. The "un-recursed" input(s) can be useful as a controller or modulator for the tree.

A great many example files have been provided to demonstrate the use of recursion. All users can have fun exploring the examples and playing with the compiled structures to create new fractals. Advanced users will find that recursion enables them to create great new fractals from scratch. Fractal design is beyond the scope of this manual, but interested users can find a wealth of information about fractals and fractal algorithms in books and on the Web.


Non-recursive compiled trees and their parameters

Parameter A: Scale

Parameter B: Iterations

Tiles (other than 4-in/2-out and 3-in/2-out tiles) that have a different number of inputs and outputs are not recursive--the tiles' output cannot be fed back into their inputs. However, even non-recursive tiles can be iterative; they can be looped so that the calculations are performed a specified number of times, making it possible to create echoes and other repeating elements. The **Scale** parameter is a multiplier applied to the tile's output. When Scale is 0, the tile's output is 0. The **Iterations** parameter causes the tile to be looped the specified number of times. By default, this parameter is locked and has a value of 0 since only trees designed for iteration can make use of this parameter.

COMPILED TREES AND MUTATIONS

The **Mutations** dialog and dice can be used to explore mutations of both the parent system and its subtrees. Lock all three locks of a compiled tree's icon () to make the subtree immune to mutations. Otherwise, the unlocked parameters in the compiled tree will be mutated when the system itself is mutated.

Important note about Undo and compiled tree mutation. Mutation of subtrees is not undoable. You should save your file before exploring subtree mutations.

The die and compiled trees. When using the dice in a parent tree, the tiles inside of unlocked subtrees are mutated unless the compiled tree icon is locked. To protect them from mutation, lock compiled tree tile in the parent. When you are in **compiled edit mode** (after typing 'e'), the mutations caused by the large die are applied only to the subtree; the small die (randomize parameters) is applied to all trees in the system that aren't locked.

RESTRICTIONS & LIMITATIONS

Compiled trees do not have access to their parent structure. As a result, there are a few limitations relating to the components they contain.

- **No packed inputs.** Compiled trees can't process packed inputs. So, you shouldn't connect the output of a Pack component to the input of a compiled tree. A compiled tree can send out packed output, however.
- **Derivative-based functions can't access parent.** When embedded in a compiled tree, components such as Derivative and 2D Derivative can only access the components within the compiled tree. As a result, if you use these components in a compiled tree, the result is different than when they are in the parent tree. Note, however, that when these components are in the parent, they do have access to compiled trees that precede them.
- **Dice rolls not undoable** - As noted above, Undo cannot undo the effects of the dice in subtrees.
- **Left/right arrow keys** - the left/right arrow keys do not change the selected tile's function assignment when in compiled edit mode.
- **Copy/paste parameters.** Copy/paste parameters when in compiled edit mode is only meaningful when copying and pasting between compiled trees with identical structures.

TUTORIALS

A basic introduction to compiled trees is found in the [Compiled Trees Tutorial](#) of the [QuickStart 2](#) chapter of this manual. The [Getting Deeper](#) chapter features additional lessons that show you how to make use of compiled trees to explore complex structures.

Advanced Topics

The remainder of this chapter features information that will be of interest to advanced users and users with some background in programming. In the future, we hope to create tutorials that explore some of these topics and their applications in greater detail. These tutorials will be posted on our web site (<http://www.artmatic.com>) and announced to registered users. In the meantime, exploration of the example files will provide insights into the applications of the features discussed below.

RELEVANT TREES (INPUTS/OUTPUTS)


As mentioned earlier, when you choose a compiled tree to use for a tile, the dialog box only displays **relevant** trees. A relevant tree has the same number of inputs and outputs as the tile in which it is being used. When determining relevance, ArtMatic counts the total number of inputs at the top of the system and the number of outputs in the system's last row. If there are two 2-in components at the top of the tree, the resulting compiled tree will be a 4-in tree. Similarly, if the last row has two parallel 1-out tiles or a single 2-out tile, the tree will be a 2-out tree. Note that to be counted for the output, the tile must be in the actual last row of the tree. Tiles with loose outputs that are not in the last row are ignored when determining relevance.

Only possibly relevant trees can be exported. ArtMatic will only export trees that are relevant to an existing tile type. Because there are no 4-in/1-out tiles, for instance, ArtMatic will not be able to export a tree that has four inputs at the top and only one output.

ITERATION, RECURSION & ITERATIVE COMPONENTS

Without components designed with looping in mind, iteration and recursion would be of limited value. This section discusses the components created with iteration in mind.

Iteration Basics

Iteration (looping) is available in both the main structure tree and in its subtrees. In the main tree, iteration is performed via the 1-in/1-out, [Iterations](#) component (). When this component is found anywhere in a parent tree, it causes the tree to be looped the number

of times specified by the component's Iterations parameter. In subtrees, the recursion/iteration parameter of the compiled tree component performs the iteration and the Iterations component simply returns the iteration number.

When a tree or subtree is looped, the entire tree is traversed a number of times. The output of the [Iterations](#) component returns the number of iterations that have been performed multiplied times the value of its parameter A (i.e. current iteration number * multiplier value). The first time the tree is traversed, the value is 0. This value is incremented each time the tree is traversed.

By making use of this component, you can easily create a system that rotates or moves slightly with each iteration by connecting its output to the third input of a component like the 3-in/2-out [z Rotate](#) component that uses the third input as a control input. The output of the Iterations component should be connected to a component that influences the system; otherwise, each iteration will calculate the same image.

The Iteration component's input has no influence, but it should be connected to a parent to maintain structural consistency (and to avoid adding an extra input when used in compiled trees).

Advanced Tip - Nested Loops: "Nested" loops can be created using multiple instances of the Iterations component, but each instance must be on a different row of the tree. See the example files "Line Nested Loops", "Color Pict Nested Loops" for examples of nested loops. The example files are two variants of the same basic structure. The first instance of the Iterations parameter determines the number of echoes, and the second instance determines the number of lines/images to be echoed.

Be careful when nesting loops since the number of iterations builds up rapidly and can bring even the fastest machine to a halt.

You can think of iteration as a way of mixing **N** number of compiled trees where N is the number of iterations and the compiled tree is the iterated system itself. There are special memory components (discussed below) that act as mixers for these iterations.

The Iteration component should generally not be the first component in the tree (especially if the top row has more than one components).

Iteration Memory

Without some sort of memory functions (or accumulator), iteration would be of value only in recursive trees (where each iteration receives the value of the previous iteration). In non-recursive trees, you would use up a lot of processing cycles but only get the result of the final iteration. A few components (Memory Max, Memory Add and Memory Depth Sort) have been provided to allow the values of the iterations to influence each other. These components have no meaning in non-iterative trees.

Each of the memory components shares the same basic functionality. After the the first traversal of the tree, the component's input is simply stored in the component. On subsequent iterations, the new value and the previously stored value interact and the result is stored in the component. The nature of the interaction is determined by the particular memory component used.

Memory Max and **Memory Add** are available in 1-in/1-out, 3-in/3-out and 4-in/3-out flavors. **Memory Depth Sort** is available in 3-in/3-out and 4-in/3-out flavors. The 1-in/1-out flavors are typically used at the output stage (or end of a branch) of a gradient-color based system. The 3-out versions are used primarily for RGB or 3D data. There are also several 4-in/3-out memory functions which use the fourth input value as a control input for determining how to mix the current and store RGB values.

Memory Max retains whichever is larger: the new value or the previous value. When the iterations are done, you are this component will essentially have returned the largest value (for each point) of all the iterations.

4-in/3-out note: this flavor of the component is the same as the 3-in/3-out version with the addition of the fourth input which acts as a scale factor (multiplier) that is applied to the input values.

Memory Add simply adds the new value to the old value. It acts as an accumulator. The **Start Value** parameter is the initial value which is added to the first iteration. Because the effect accumulates over the iterations, you may sometimes want to set the Start Value to a low value so that the system doesn't attain large/max values everywhere. This component can be useful for superimposing many instances of a changing system. Memory Add can be especially useful for summing harmonics in systems designed for sound generation. In RGB systems, this component can be used to to achieve beautiful transparency effects. In 3D systems, you can achieve volumetric effects.

4-in/3-out note: this flavor of the component is the same as the 3-in/3-out version with the addition of the fourth input which acts as a scale factor (multiplier) that is applied to the input values.

Memory Depth Sort acts as a memory-based version of the other depth sort functions. Rather than performing a depth sort between two sets of inputs, the depth sort is performed between the current set of values and the stored values from the previous iteration. Depth Sort preserves the values that correspond to the foremost pixel when the three values are considered as 3D position. The foremost pixel corresponds to the value with the lowest valued z (third) co-ordinate. This component is useful for accumulating 3D objects in recursive and iterative trees.

Example files: See the files "Surfaces Z sort igate E" and "Surfaces Z sort igate" for sophisticated examples which use the component as a "z buffer" which builds up layers synthesized by the system's compiled tree.

4-in/3-Out Memory Functions: Memory Alpha, Memory w Min, Memory w Max. Each of these functions uses the component's fourth input (which we call 'w') as a sort of control input which is not passed back out of the component. Explore the examples library for examples of these components. The effects you can create with them defy description.

Memory Alpha mixes the current RGB values with the stored RGB values by interpreting the fourth input (w) as the transparency of the current values. A high 'w' value makes the current value (pixel) opaque. A 'w' value less than 0 makes the current value (pixel) completely transparent (and, hence, invisible). See the example file **MemAlpha Clouds** for an example of this powerful component. We recommend rendering the animation to get a sense of the incredible effects you can achieve with this component. One can simulate volumetric-3D rendering with this component.

Memory w Min compares the current w value with the stored w value and stores whichever set of values (R,G, B, and w) has the lowest value for w. See **Mwmin perlin noise** for one example of this component. This component is very much like Memory Depth Sort but uses 'w' (which is not passed out of the function) rather than the third input for the comparison.

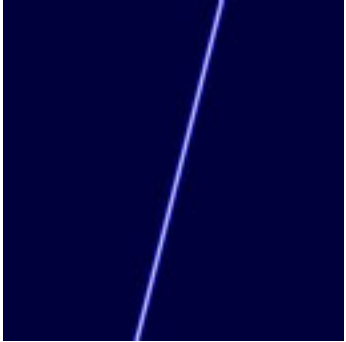

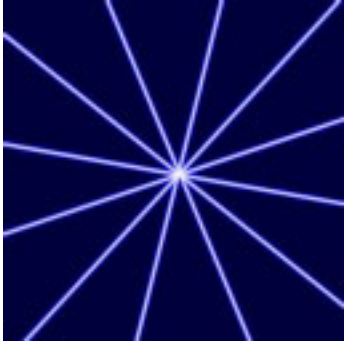
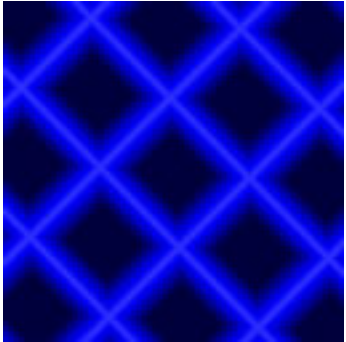
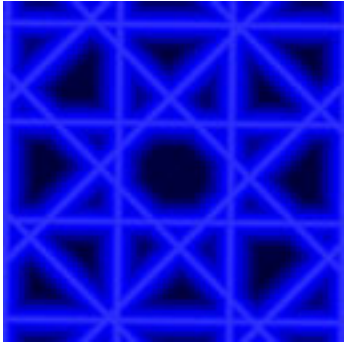
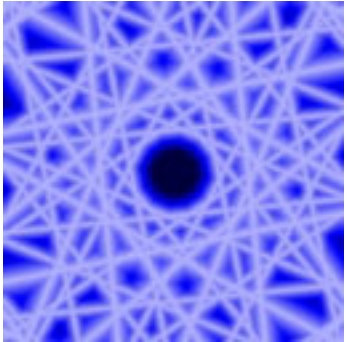
Memory w Max compares the current w value with the stored w value and stores whichever set of values (R,G, B, and w) has the largest value for w. Note that only the 'w' value is used for the comparison unlike the simple Memory Max component which uses all of the input values for the comparison.

IMPORTANT ADDITION! After this chapter was completed a number of components were modified to provide memory/accumulation functionality when used within a iterative systems. All of the mixing components with "packed" in their names are provided with memory when used in iterative systems. After the first iteration, the component's output is stored and mixed with the subsequent iteration's output which is stored and mixed with the subsequent iteration's output, and so on.


See the example file "AffineAlphaFeedback Textured" for an example of this capability.

A SIMPLE ITERATIVE EXAMPLE

This section takes you through the process of creating a simple iterative system that creates star-like patterns by iteratively rotating a line. As mentioned earlier, you can think of an iterative system as one that uses a memory component to mix of all of the iterations. So, iteration is used to create images that can be generated by successively modifying a system and mixing the iterations, as in the examples shown below:

	One iteration	Two iterations	Five iterations
Images created by drawing a line and rotating it in successive iterations.			
Images created by drawing a grid and rotating it in successive iterations.			

For this exercise, we will design a structure like that used to create the first set of examples in the table above. When you are first learning to put together iterative systems, it is a good idea to start by creating a non-iterative version of the system. To create this image, we need a simple system that draws a line and permits it to be rotated. See if you can design such a system before reading further. The solution is shown below.



The system shown at left is the simplest possible system to draw and rotate a line.

The next step in creating the system is to add the components needed to support iteration. Non-recursive iterative systems need to contain the following components to be meaningful: at least one Iteration component, a memory component to mix the iterations, and a component whose output can be influenced by the Iteration component.

Question: Do you know why the system needs to have a component that can be influenced by the Iteration component (in non-recursive systems)? It is needed because the parameter settings of all the tiles remain constant during the iterations. So, we need at least one component whose output will be influenced by the Iteration component. Otherwise, each iteration would generate the exact same values.

Before reading further, see if you can modify the simple structure shown above by adding the components required to perform meaningful iterative rotation.

Hints

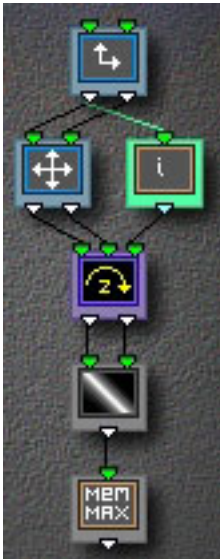

There are several 3-in/2-out components that perform the same function as 2-in/2-out components (such as rotation) and use the third input to control the component's effect. There are also 4-in/3-out functions that behave as 3-in/3-out components and use the fourth input to control the component's effect. These are perfect for Iterative control.

To change the number of inputs or outputs for a tile, hold down the option key and then click on the desired tile to pop up the tile options popup menu.

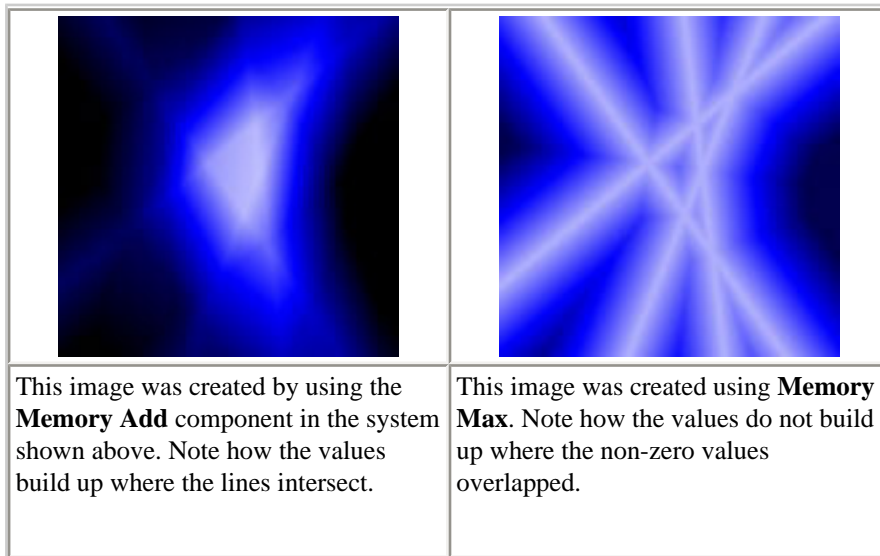
You may want to insert one or more neutral components (such as Scale or Scale & Offset) above the Rotation component to allow you to add a parallel branch whose output can feed a lower branch.

After adding the required elements, change the Iteration component's Iterations parameter to a non-zero value then adjust the other components parameters to manipulate the image.

The table below shows two possible solutions to the exercise. Note that the 2-in/2-out rotation component was replaced with the 3-in/2-out version which allows the rotation amount to be controlled via its third input which is fed by the Iteration component. You can use the same technique to provide iterative control of scaling and displacement.

<p>Two equivalent systems that provide iterative control of line rotation.</p>		
	<p>This is the stylistically preferable solution since it can be exported as a 2-in/1-out compiled tree.</p>	<p>This is the simplest solution. It is fine as a top-level tree but less preferable than the solution at left when used as a compiled tree since it would have a useless third input.</p>

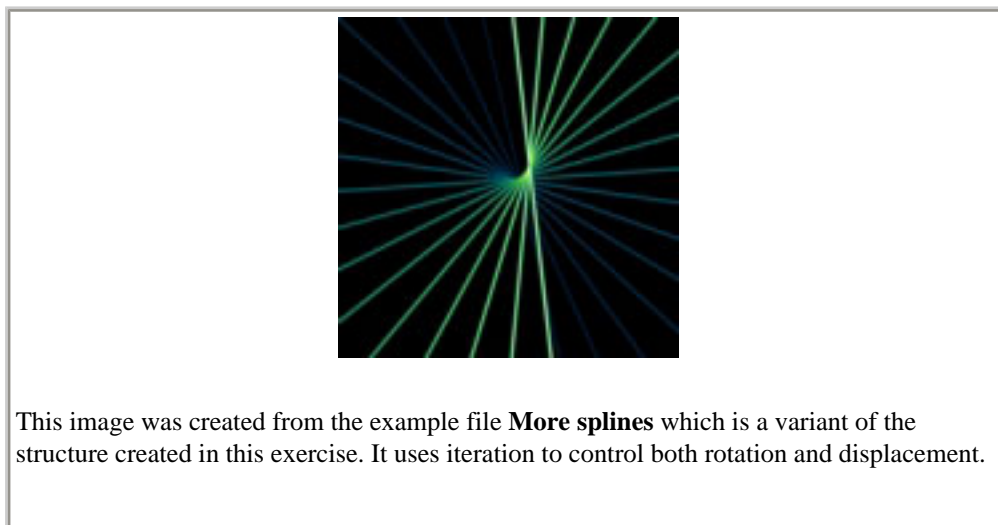
The choice of memory component depends on the desired effect. Using Memory Max provides 'opaque' mixing of the iterations since each pixel is the result of a single iteration. If Memory Add is used the value is cumulative. So values 'build up' where successive iterations are non-zero. The table below shows close-up renderings of the center of this system with Memory Add and Memory Max



This is a great system to export as a compiled tree. It can then be used as a 2-in/1-out component. It works both as a surface generator (for creating the starlike patterns pictured earlier) and as a mixer. Remember when using it as a compiled tree that you need to set the Iterations parameter of the compiled tree component itself since the Iterations component's Iterations parameter is ignored when it is inside a compiled tree.

As an exercise, explore the images that can be created by:

- substituting a displacement component for the rotation component,
- using **Grid** and other components in place of the **Line** component,
- adding components needed to provide iterative control of both rotation and displacement,
- inserting a 1D **Random** filter between the Iteration component and the rotation component.



TIPS & TECHNIQUES

- The Documentation Example Files folder contains a number of examples of compiled trees and iteration that are worth exploring to get some ideas about how it can be used. You should also explore the examples in the main examples library supplied with ArtMatic Pro.

- Some compiled trees only make sense when used within another tree. For example, a compiled tree that acts as a space transform is hard to fine-tune without surface and/or shading components that make use of it. When you develop such a tree, it can be tricky to find

meaningful parameters and keyframes. When working with such trees, create the basic tree structure and export it as a compiled tree then create a new system that makes use of the compiled tree. Use this parent tree to find meaningful keyframes and parameter values then re-export the subtree to make it available when creating other systems.

- Iteration is most successful in systems where there is a fair amount of contrast between the values/pixels/objects in the system and where the system is either offset, rotated or scaled between iterations.

- When exploring iterative systems, set the the iterations to 0 to see what a single iteration does. Now, adjust the Iteration component's Multiplier parameter to see how the component's value influences the systems.

Component Reference

The following chapters provide detailed information about ArtMatic's component functions. Every tile in a structure tree has a component (function) assigned to it. Component assignments can be changed by clicking on a tile to select it then clicking to pop up the Component Selector. The components that are available are determined by the number of inputs and outputs that the tile has. The Component Reference chapters are organized by tile connections (the number of inputs and outputs).

What are components?

You can think of ArtMatic's components as graphics filters, space distortion functions, signal processors, or mathematical functions. As noted elsewhere, components are really mathematical functions (or procedural rules) that take in values via the input threads and calculate new values which are passed out through the component's outlets. In the reference chapters, we provide the actual mathematical formulas used by the components where they are meaningful. It is not necessary to understand the mathematics involved, but users with some background in mathematics will find the formulae useful.

How to approach the Component Reference

The component descriptions make for pretty dry reading. But, we do recommend browsing through the chapters to get an idea of what is available. Where possible, we have provided simple examples and illustrations that will give you some ideas about how the various components can be used.

The [Getting Deeper](#) chapter of the manual provides a valuable framework for understanding how components work within structures and how to understand the structure trees themselves. We strongly recommend that you read the [Concepts](#) and [Getting Deeper](#) chapters as they will help to make sense of the component descriptions. You may find it useful to quickly browse through the component descriptions before reading the [Getting Deeper](#) chapter, but don't worry if you don't understand all of the descriptions. The Getting Deeper chapter will help you to make sense of the components and how they function.

Component Functions & Conventions

In the following sections, some functions are described as generating surfaces and others as distorting space and others as filters. These labels are intended merely as useful guidelines to give you an idea of the component's prototypical use, but these functions when combined together work in ways that no simple labels can capture. The wonderfully rich image universes generated by these complex dynamical systems are beyond description. While the individual components are quite simple, the results can be quite surprising, especially when a complex component tree is being used.

Complex numbers. You will notice that many components use complex numbers in their calculations. Complex numbers are numbers made up of a real and an imaginary part in the form of $a + bi$ where i is the imaginary number (the square root of -1). Complex numbers and operations involving them have many fascinating properties which are beyond the scope of this documentation. If you search the internet, you will find many sites that cover this topic.

Conventions and definitions. In the tables below, there are a few conventions worth noting in the formulas that appear. A, B and C refer to the values of parameter sliders A, B and C. x , y , and z refer to the inputs to component tiles. The first input (counting from the

left) is x , the second is called y , etc. Where the word **distance** is used, it refers to the **Euclidean distance** which is $\sqrt{x^2 + y^2}$ or $\sqrt{x^2 + y^2 + z^2}$.

About x , y , and z . Keep in mind that x , y and z are sometimes used in two very different ways. In discussing, the tree's inputs or the

input/output of some spatial components, x, y and z refer to actual spatial co-ordinates. When describing an individual component's inputs, x, y and z merely refer to the component's first, second and third inputs. In many cases, these values will not be spatial co-ordinates. For example, a 2-in/1-out function at the end of the tree might be receiving input from two separate branches of the tree; the component's left and right inputs are referred to as the x input and the y input when describing the component, but in this case the values being received aren't xy spatial co-ordinates; they are just two numbers to be acted on by the function.

Procedural functions. Some functions in the following sections are described as **procedural functions**. A procedural function does not have a simple mathematical formula but has a rule or algorithm that may change based on the function's inputs or parameter settings. For example, the **MinMax** component may choose either the minimum or the maximum of the two inputs or it may interpolate between the two values depending on the setting of parameter B.

Common Parameters

There are a number of component parameters that you will encounter over and over in the component descriptions. They are:

- **Amplitude** - amplitude generally controls the amount or strength of the effect. It often will effect the range of the effect as well.
- **Frequency** - frequency generally controls the spacing of repeated effects and cyclic functions. For example, in the Ripples component, the frequency determines ripples spacing.
- **Phase** - phase usually acts as an offset which shifts repeated patterns and cyclic functions up or down and/or left or right.
- **Scale** - scale acts as a multiplier which is usually applied to the component output.

Component Types

The following table shows the component types sorted by the number of inputs and outputs. Click on the underlined text to jump to the corresponding reference chapter. The Interpretation column summarizes possible functional categories of the various tile types. These categories are described in detail in the [Understanding Components](#) section of the **Getting Deeper** chapter of this manual.

# inputs/outputs	Common Interpretations
1-in/1-out	1D filter
2-in/1-out	2D scalar functions: surface generation, shading, mixing
2-in/2-out	2D vector functions: space distortion, parallel filtering
2-in/3-out	space translation, RGB shading
3-in/1-out	surface generation, mixing, packing
3-in/2-out	3D vector functions: space translation/distortion, texture generation
3-in/3-out	space distortion, colorspace translation, packed stream mixing






4-in/2-out	space distortion
4-in/3-out	RGB/3D mixing






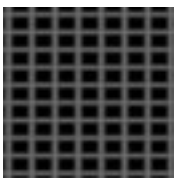
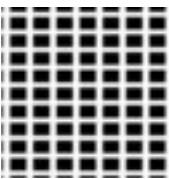

Filters: 1D Scalar Functions (1 in - 1 out)





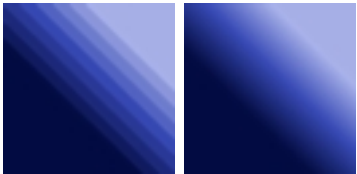

These scalar functions act as filters to modulate values. These components also provide basic mathematical functions that you can use to implement your own algorithms and formulas. In most cases, the component's name is the calculation performed by the component. An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone.






In the discussion below, **x** refers to the input value.









It is often useful to insert 1D components between other components (and at the end of the tree) in order to massage individual value 'streams'.






Function	Notes	Parameters
 Ax + B	This component multiplies the input and adds an offset. This component is a basic scaling filter whose formula is the same as the name. It is nice for adjusting the contrast and level of a system.	A: Scaling B: Offset
 x+A	Math primitive that adds a value supplied by the Offset parameter to the incoming value.	A: Offset
 A - x	This component inverts the slope of the incoming values. This component also provides a mechanism to invert the input values. When A is zero, this component yields -x. Formula; A - x	A: Level
 Abs x	This component produces the absolute value of the input (i.e. it strips the minus sign from the input). Tip: Add this component after the A/x component to create a neon line effect.	None
 A - ABS(x)	Useful mathematical primitive. Formula: A - ABS(x + B)	A: Size B: Offset


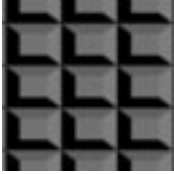


 <p>Smooth Abs x</p>	<p>This is a modified version of the absolute value function which is smoothed near the origin. (True Absolute Value has a sharp edge at 0,0). The parameter provides the degree of smoothing.</p>	<p>A: Amount</p>
 <p>Gaussian</p>	<p>This component uses a Gaussian transform (like a bell curve) on the input. The output of a Gaussian function at 0 is 1 and approaches 0 as x approaches infinity.</p> <p>Formula: $\exp(-nx^2)$ with n from 0 to 2</p>	<p>A: Power B: Phase C: Amplitude</p>
 <p>Plateau</p>	<p>This component is a gaussian-type filter that has an extended plateau on either side of 0. It is useful for delineating symmetrical shapes and creating edge effects. See the file Flying Rectangles to see how plateau can be used to provide rounded corners.</p> <p>Slope determines the sharpness of the plateau edge. Size determines the plateau width. Amplitude is a multiplier.</p>	<p>A: slope B: size C: amplitude</p>
 <p>x power A</p>	<p>This component gives a non-linear remapping of values. With the power parameter set to the minimum, you can get useful discontinuities such as well-defined islands and borders (depending on the input). The range of A is from the eighth root of x to the 4th power of x. If the incoming value is negative, the absolute value is taken before applying the function. Parameter B is applied as a divisor.</p> <p>Formula: $(x ^A)/B$</p> <p>Advanced tip: This component can be used to create metallic effects when applied to a value stream that is providing light intensity. For an example, see the file <i>Disgusting Shader</i>.</p>	<p>A: Power B: Scale</p>
 <p>Power Slope</p>	<p>An exponential curve with a step function-like effect except that it is continuous with rounded edges near the "step". This function when used at the final stage of a system tends to make the image very high contrast--black and white rather than shades of gray, for example.</p> <div data-bbox="289 1381 1182 1738" style="border: 1px solid black; padding: 10px;">   <p>Normal grid output. A grid with the Power Slope component filtering its output</p> </div>	<p>A: Size B: Phase C: Power</p>
 <p>A(1-exp(-x))</p>	<p>This component filters the input with an inverted Gaussian curve. It provides a convenient way to smoothly clip values that approach infinity. It is a little like log, but it has a steeper slope and sharper cutoff.</p>	<p>A: Amplitude</p>

 <p>Log</p>	<p>This filter applies the natural log to the input. It is good for decreasing the output range of functions that go to infinity. The amount parameter determines the steepness of the output curve.</p>	<p>A: Amount</p>
 <p>A/x</p>	<p>This component creates a non-linear inversion of the input.</p> <p>Formula: A/x</p>	<p>A: Amplitude</p>
 <p>A/(xx+A-1)</p>	<p>This component is similar to Gaussian but has a steeper curve.</p>	<p>A: Amplitude</p>
 <p>Quantize</p>	<p>Quantize the input into discrete steps. It turns a gradual gradient, for instance, into a series of color bars as in the illustration below. Use this function to create discontinuous, step-like functions. It is great for creating pixelization and color-reduction effects.</p> <p>Frequency determines the spacing of the quantization steps. The Amplitude determines the strength of the quantization. (With a value of 0, there is no quantization; with a maximum value, true quantization is performed.)</p>  <p>The image on the left is the result of applying the <i>Quantize</i> component to a gradual ramp (provided by the $Ax+By+C$ component). The image on the right is the unquantized image.</p>	<p>A: Freq. B: Amplitude</p>
 <p>Infinity Gate</p>	<p>This function allows you to replace finite values with infinity which has some special characteristics (i.e. it is replaced by the depth cueing color--in RGB systems--and is transparent when encountered by 'packed' mixers). Like the Floor component, this function can be used to define a range outside of which the values are remapped. When the input value less than the minimum or greater than the maximum the value infinity is sent out which makes those portions of the surface transparent to the 'packed' mixers.</p> <p>This is great for creating keying effects when using ArtMatic Pro to create video effects.</p>	<p>A: Maximum B: Minimum</p>

<p>Memory Functions</p>	<p>The following two components are Memory functions that allow images to be built in iterative and recursive trees by accumulating the iteration results. These components are only meaningful when used in iterative and recursive systems. The memory functions compare the incoming value to the stored value and store the result. This operation is performed for each iteration of the tree and the final output is the cumulative result of performing these operations for each visible pixel.</p> <p>This topic is covered in detail in the chapter Compiled Trees and Iteration which also provides simple tutorials for understanding these functions.</p>	
 <p>Memory Max</p>	<p>Memory Max retains whichever is larger: the new value or the previous value. When the iterations are done, you are this component will essentially have returned the largest value (for each point) of all the iterations.</p> <p>This component is only meaningful in iterative/recursive systems. This topic is covered in detail in the chapter Compiled Trees and Iteration which also provides simple tutorials for understanding these functions.</p>	
 <p>Memory Add</p>	<p>Memory Add simply adds the new value to the old value. It acts as an accumulator. The Start Value parameter is the initial value which is added to the first iteration. Because the effect accumulates over the iterations, you may sometimes want to set the Start Value to a low value so that the system doesn't attain large/max values everywhere. This component can be useful for superimposing many instances of a changing system. Memory Add can be especially useful for summing harmonics in systems designed for sound generation. In RGB systems, this component can be used to achieve beautiful transparency effects. In 3D systems, you can achieve volumetric effects.</p> <p>This component is only meaningful in iterative/recursive systems. This topic is covered in detail in the chapter Compiled Trees and Iteration which also provides simple tutorials for understanding these functions.</p>	<p>A: Start Value</p>
 <p>Sin x</p>	<p>This component applies a sine function to the output which renders its output cyclic. It is nice for creating wood-type textures and zebra patterns.</p>	<p>A: Frequency B: Phase C: Amplitude</p>
 <p>Power sin</p>	<p>This component is useful for creating banding effects and for adding harmonics on the sound page. This function is equivalent to passing the output through a power filter and then through the sin filter.</p> <p>Formula: $(\sin x)^A$</p>	<p>A: Frequency B: Phase C: Power</p>
 <p>Saw wave</p>	<p>This component uses a triangle wave function to remap the output. Similar to a sine wave but with sharper edges.</p>	<p>A: Frequency B: Phase C: Amplitude</p>






 <p>U wave</p>	<p>This component is a useful u-shaped periodic wave.</p>	<p>A: Frequency B: Phase C: Amplitude</p>
 <p>x + A sin Bx</p>	<p>This component modulates the input with a sine function rather than simply filtering with a sine function.</p> <p>Formula: $x + A \sin Bx$</p>	<p>A: Amplitude B: Frequency</p>
 <p>Random</p>	<p>This component generates truly random values unlike most of the ArtMatic's noise/random functions which generally provide randomized distortions of the input (rather than generating values unrelated to the incoming values).</p>	<p>A: Amplitude B: Frequency C: Phase</p>
 <p>Randomize</p>	<p>A randomizing component which provides random distortion of the input. This differs from the Random function which generates truly random values.</p>	<p>A: Amount B: Frequency</p>
 <p>Fractal Noise</p>	<p>A 1D version of the fractal noise algorithm. See the descriptions of the higher-dimensional fractal noise components for more information.</p> <p>Example file: Gaz Giant vortex study</p>	<p>A: Amplitude B: Frequency C: Phase</p>
 <p>x (Sin x)</p>	<p>This filter is a variation of the sine function.</p>	<p>A: Frequency B: Phase C: Amplitude</p>
 <p>A Power X</p>	<p>Another basic mathematical building block</p> <p>Formula: $B * (A^x)$</p>	<p>A: A B: Amplitude</p>
 <p>Sphere</p>	<p>The Sphere filter yields a sphere when applied after the distance component.</p> <p>Formula: $\sqrt{A - x^2}$</p>	<p>A: Radius</p>






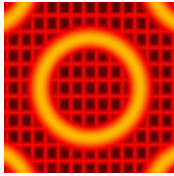
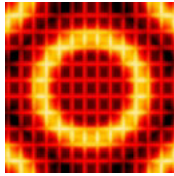
 <p>Cubic Clip</p>	<p>Cubic clip is similar to the Clip function but uses a cubic spline to maintain continuity at the clipping boundaries. It creates a smooth interpolations from 0 to A. Parameter B is an offset added to the incoming value before applying the cubic filter.</p> <p>Formula: $-2(x^3) - 3(x^2)$ for values that fall between 0 and A.</p>	<p>A: Amplitude B: Offset</p>
 <p>Clip</p>	<p>This filter clips the input when it passes outside of the range from 0 to the value defined by the CutOff parameter.</p>	<p>A: Contrast B: Offset C: CutOff</p>
 <p>Floor</p>	<p>Constrain the input to values between the Floor Level and the Roof Level. Any value below the Floor Level is assigned the floor level. Any value above the roof level is assigned the roof level.</p>	<p>A: Floor Level B: Roof Level</p>
 <p>sin(Bx)/sin(x)</p>	<p>This component is great for creating harmonics for structures designed for sound synthesis (when using the Direct Drones synthesis method). The component's equation is Sin(Bx)/sin(x).</p> <p>Amplitude determines the output amplitude. Frequency influences the harmonic spacing. Harmonics determines the number of harmonics.</p>	<p>A: Amplitude B: Harmonics C: Amplitude</p>
 <p>Derivative</p>	<p>This filter approximates the derivative of the function whose output feeds the derivative component. This component often emphasizes the 3D aspect of a system. Its output only makes sense when the input values are continuous (i.e. it does not work well with discontinuous functions such as Random Squares).</p> <p>Note: This component will behave differently when it is in a compiled tree than when it is in the main tree since it uses information other than that which comes in through its input. When this component is used (almost always in the lower part of tree), ArtMatic traverses the tree for the information that it needs to derive the lighting information. As a result, this component can't be used inside of compiled trees since the tree as a whole is hidden from compiled trees.</p> <p>Formula: $f(x + dx, y + dy) - f(x,y)$</p>	<p>A: Amplitude B: Offset</p>

 <p>2D Derivative</p>	 <p>Related to the derivative, this function provides lighting and 3D texture effects by scanning the entire tree to compute partial derivatives. Parameter A controls the direction of a light which shines across the horizontal axis, and parameter B controls a light which shines along the vertical axis. Note the 3D effect it has on a simple 2-in/1-out Grid.</p> <p>This component actually looks at the entire tree structure and not just the incoming values.</p> <p>Note: This component will behave differently when it is in a compiled tree than when it is in the main tree since it uses information other than that which comes in through its input. When this component is used (almost always in the lower part of tree), ArtMatic traverses the tree for the information that it needs to derive the lighting information. As a result, this component can't be used inside of compiled trees since the tree as a whole is hidden from compiled trees.</p> <p>Advanced note for the mathematically inclined: This component approximates the partial derivative along both the x and y axes of the system (dfx and dfy). The result is the dot product of parameters A and B with the approximated partial derivatives over x and y. This component is one of ArtMatic's most computationally-intensive components.</p>	<p>A: Light direction dx B: Light direction dy C: Offset</p>
 <p>Iterations</p>	<p>This component serves two functions. It can force a top-level tree to be looped (calculated several times in succession), and it acts as a counter whose value corresponds to the iterations that have been performed. The behavior is slightly different in top-level trees and in compiled trees.</p> <p>In top-level structure trees, it forces the tree to be looped the number of times specified by parameter B. In compiled trees (and also in top level trees), it returns the number of iterations that have been performed (multiplied by value of parameter A). In sub-trees, the number of iterations to perform is provided by the compiled tree component's Iterations/Recursion parameter.</p> <p>During each iteration, this component's value is: current iteration number * Parameter A</p> <p>Iteration is discussed in some detail in the chapter Compiled Trees and Iteration which also includes a tutorial that covers the construction of iterative trees.</p> <p>NOTE: In compiled trees parameter B is ignored as the compiled tree component performs the iterations.</p> <p>WARNING: This component should never be the first component in a tree.</p>	<p>A: Multiplier/Value B: Iterations</p>
 <p>Open Compiled Tree</p>	<p>Choose this component to use a 1-in/1-out compiled tree. Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine.</p> <p>Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.</p>	<p>A: Blend B: Recursion</p>

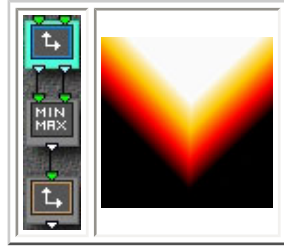
2D Scalar Functions (2-in/1-out)

These components perform a variety of functions. They are often used to mix the output of two separate components (or branches) or to provide a surface or texture when fed by the values coming from a single two-output component. An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone.

Function	Notes	Parameters
 Ax + By + C	<p>This component is useful both as a mixing component that mixes the output of two separate components and as a surface generator which can generate a tilted plane. If the inputs come from separate components, the component acts to mix their outputs with the scale parameters determining the relative contribution of the left and right inputs. If the inputs are the points of a plane (such as when it is the first component in a system) then this component effectively tilts the plane where A and B provide the tilt or rotation and C acts as a zoom. This is essentially a plane equation.</p> <p>This component can also be used to generate gradients such as the one pictured in its icon.</p> <p>Formula: $Ax + By + C$</p>	A: Scale X B: Scale Y C: Add
 Line	<p>A single "line" (a half-cylinder on its side) is generated when the input is undistorted space. This component is a commonly used primitive. It is also useful as a mixer for mixing the output of two components. The surface that is generated is actually a half-cylinder on its side and therefore has a height unlike the heightless lines of Euclidean geometry.</p>	A: Angle B: Amplitude C: Width
 Gaussian Dot	<p>Gaussian Dot is a basic building block that generates its maximum value at the origin ($x = 0, y = 0$) and whose output approaches zero as the input values approach infinity which makes it practical for constraining inputs that tend towards infinity. Practically-speaking, it is the inverse of the Distance component.</p> <p>Example: The example file "DC Double Affinity feedback" uses Gaussian Dot as the principal component of the compiled tree at the heart of the system. To view the compiled tree, click on its tile and type 'e'. Type 'e' again to return to the parent tree.</p>	A: Amplitude B: Size
 Lozenge	<p>A simple four-sided pyramid is generated when the input is undistorted space. As with Gaussian Dot this component reaches the maximum value at the origin (0,0) and reaches 0 as the inputs approach infinity. It can be used in many of the same situations as Gaussian Dot. It is frequently used as a primitive in recursive compiled trees. For instance, it can be used in place of the Gaussian Dot found in the example "DC Double Affinity feedback".</p>	A: Amplitude B: Width C: Height
 Multiply(x+A)*(y+B)	<p>This component generates interesting complex interactions of the two inputs. The output retains some aspects of the two inputs. While the equation is quite simple, the results are often complex and quite surprising.</p> <p>Formula: $(x+A)*(y+B)$</p>	A: Offset x B: Offset y

 <p>Distance</p>	<p>Calculate the Euclidean distance. The Distance component generates a sort of blunt-nosed cone when it is given the points of the plane as its input. If the 1D sphere component follows the distance component, the result is a perfect sphere. Parameters A and B offset the output and parameter C acts as a multiplier/zoom.</p> <p>Formula: $C\sqrt{(x+A)^2 + (y+B)^2}$</p>	<p>A: Offset x B: Offset y C: Amplitude</p>
 <p>Distance*(x + y)</p>	<p>This component provides shadow-like effects, and its value approaches 0 the further one gets from the origin (0, 0).</p> <p>Formula: $B(x+y)(e^{-A(x^2 + y^2)})$</p>	<p>A: Offset B: Amplitude</p>
 <p>Radial</p>	<p>When the input is undistorted space, this component's output is the angle between the x-axis and the input point. The output is symmetrical about the x-axis and yields a lighting effect when the input is the points of the plane. The frequency parameter controls the arc of the lighted and shadowed areas.</p>	<p>A: Frequency</p>
 <p>Min < - > Max</p>	<p>MinMax can be used as a mixer to combine two surfaces (outputs of two independent components as shown in the example). The two surfaces appear to be opaque unlike most of the other mixing components which tend to merge the inputs. The apparent opacity is the result of the fact that MinMax is a procedural function that chooses either the minimum or the maximum of the two incoming values rather than merging the values. It can also interpolate between the input values. When parameter slider B is at its minimum value, the output is the minimum value of the two inputs. When the parameter slider is at its maximum value, the component generates the maximum of the two values. Other values of parameter B generate an interpolation of the two inputs whose weighting (towards the minimum or the maximum) is determined by the slider's setting (at the midpoint it is the average of the two values). Parameter A is a multiplier which is applied to the chosen value. The Smoothing parameter is new in version 2.5 and controls the continuity of the transition between the two inputs. When it is set to 0, the function behaves as it did in earlier versions of ArtMatic.</p> <p>Uses: Min/Max tends to be used as a mixer since it selectively chooses values from one input or the other rather than combining the values. Notice the difference between the two images below. Both images mix the output from Grid with the output from Ripples.</p> <div data-bbox="383 1455 1219 1955" style="border: 1px solid black; padding: 10px;"> <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;">  <p>Structure used to create ex. 1.</p> </div> <div style="text-align: center;">  <p>Ex. 1: Note the ripples appear to be in front of the grid.</p> </div> <div style="text-align: center;">  <p>Ex. 2: Image that results when Ax+By+C is used to rather than MIN/MAX. Note that the images appear to be merged.</p> </div> </div> </div> <td data-bbox="1357 798 1524 1997"> <p>A: Amplitude B: Min to Max C: Smoothing</p> </td>	<p>A: Amplitude B: Min to Max C: Smoothing</p>

When this function is used as a surface generator (i.e. when its input is a space), the result tends to be edges and V type contours such as in the example below.

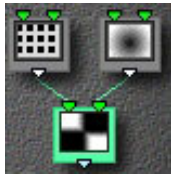
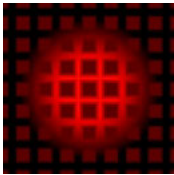
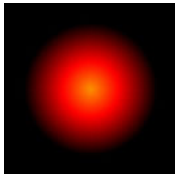


Min + Max

Like **MIN/MAX**, this component tends to be used to mix two independent surfaces rather than to generate a surface from an incoming space, and, like **MIN/MAX**, it selectively selects between the incoming values rather than combining them with a calculation. **Min + Max** is a procedural function that behaves differently for positive and negative input values. Either the minimum or the maximum value of the two inputs is selected, depending on the sign of the second input (*y*). When the second input is positive the maximum value is output. When the second input is negative, the minimum value is output. The component's icon is the image which results when the raw points of the plane are fed into the component.

None

When the inputs are two surfaces, you can get very interesting masking effects where the surfaces intersect. The effects are especially interesting when the second input has both positive and negative values.

 <p>The structure used to create the image on the right.</p>	 <p>Note that you can create a searchlight effect by animating the Distance component's offset parameters.</p>	 <p>The surface created by the Distance parameter by itself. The surface has very large values in the center and negative values at its edges.</p>
--	--	--









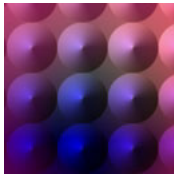
In the example above (from the example file "**Min + Max**"), the Distance component's amplitude parameter was set to -4 which results in a dome-like surface that approaches infinity at the center and turns negative at the edges. In the resulting image, notice how the grid's coloration is opposite inside and outside of the dome.



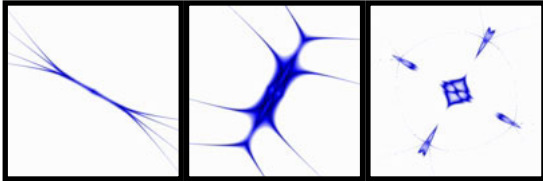











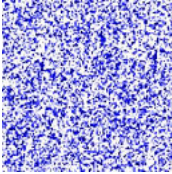

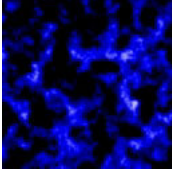


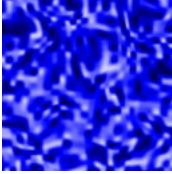
Difference


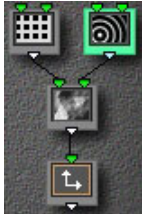

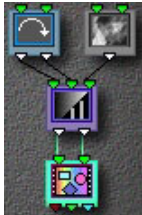

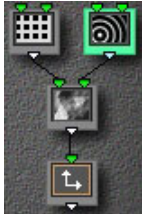

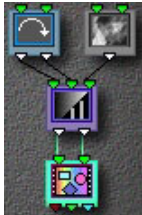

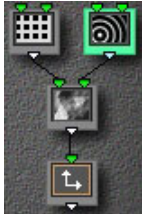

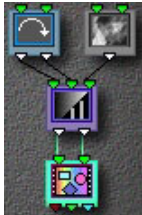


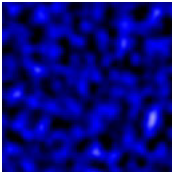

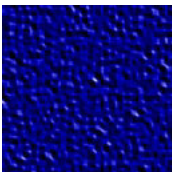


Difference calculates the absolute value of the difference of the two inputs. This is the same as Photoshop's difference mode.


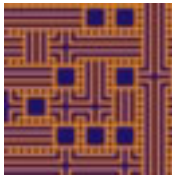



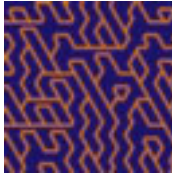


A:
Amplitude



 <p>Grid</p>	<p>This component generates a grid pattern when it is fed the points of the plane. It is a very useful function when examining how the various vector functions distort space. The scale parameters determine the spacing of the horizontal and vertical bars. The amplitude parameter determines the "height" or the "arc" of the grid bars.</p>	<p>A: Scale x B: Scale y C: Amplitude</p>
 <p>Hexa Grid</p>	<p>Hexa Grid generates a hexagonal grid which appears as an arrangement of hexagonal tiles. The frequency parameter determines the size/spacing of the grid and acts like a zoom function. The amplitude parameter determines the arc/height of the grid walls, and the phase parameter shifts the grid left/right.</p>	<p>A: Frequency B: Amplitude C: Phase</p>
 <p>Hexagons</p>	<p>Hexagons is typically used as a surface generator which creates a lattice of hexagons. The amplitude parameter acts as a contrast control while phase and frequency control the lattice's offset and spacing. This component can also yield interesting results when used as a mixer.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Sin x + Sin y</p>	<p>This component generates an egg-crate like surface when given undistorted space as its input. The frequency parameter controls the size of the surface's 'cells' and the phase parameters shift the pattern vertically and horizontally.</p> <p>Formula: $\sin x + \sin y$</p>	<p>A: Frequency B: Phase x C: Phase y</p>
 <p>Sin x * Sin y</p>	<p>The output of Sin x * Sin y is similar to that of the sin x + sin y though it has more of a checkerboard effect on a raw plane. Amplitude effects the steepness/color contour of the cell walls.</p> <p>Formula: $\sin x * \sin y$</p>	<p>A: Amplitude B: Phase x C: Phase y</p>
 <p>x + A Sin y</p>	<p>This component has a distortion effect and can have a wave or interference pattern-like effect when receiving the output from two independent scalar functions.</p>	<p>A: Amplitude B: Frequency</p>
 <p>Hexa sines</p>	<p>This component is similar to Sin x + Sin y performed on a hexagonal grid.</p>	<p>A: Amplitude</p>
 <p>Circles</p>	 <p>Circles generates an arrangement of tiled cones. The frequency parameter determines the spacing of the cones. The size parameter determines the size/shape of the bases; at the minimum value the base has a circular shape; at the maximum value, the cones appear to have square bases. The power parameter acts as a contrast/brightness control on the component's output. To see the output as cones, use the derivative function as the final component as in the image to the left.</p>	<p>A: Frequency B: Size C: Power</p>

 <p>Ripples</p>	<p>Ripples generates a pattern of tiled ripples when given a plane as the input. The ripples # parameter determines the number of concentric rings per tile. The size parameter determines the basic shape of the ripples. The frequency parameter determines the spacing of the ripple tiles.</p>	<p>A: Ripples # B: Size C: Frequency</p>
 <p>Dynamo</p>	<p>Dynamo is a fascinating component whose behavior is impossible to explain and whose formula is quite complex. It has a very non-linear response. The images below are all taken from a simple system whose only changes are to Dynamo's parameters and the zoom level.</p> 	<p>A: Rotation B: Phase x C: Phase y</p>
 <p>Craters</p>	<p>Craters is a texture function that is generally used to create realistic craters when creating planet-like objects. The example files include many simulated moons whose surface is covered with craters.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Random cones</p>	<p>Technically-speaking, this component generates an arrangement of randomly-sized and placed cones whose appearance can seem like motes or "space-dust". Amplitude controls the contrast/cone height. Bubble size controls the cone base size, and frequency controls the spacing and can act like a zoom control. This component involves fairly intensive calculations and can be a bit slow.</p>	<p>A: Amplitude B: Bubble size C: Frequency</p>
 <p>Star Field</p>	<p>Star Field generates a virtual, layered starfield that is great for creating cosmic textures and images as well as for bump textures. Animating the phase parameter creates a sense of depth by moving the layers at different speeds.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Facet</p>	<p>Facet is a discontinuous function that generates a mosaic-like surface. It uses an algorithm similar to bubbles but generates the same value (altitude) for each individual bubble.</p>	<p>A: Frequency B: Amplitude C: Phase</p>
 <p>Bubbles</p>	<p>Bubbles is almost the inverse of Random cones. It generates a surface of randomly placed spherical holes. A negative amplitude value turns the holes into bumps. This component's calculations are faster than random cones.</p>	<p>A: Amplitude B: Bubble size C: Frequency</p>

 <p>Techno</p>	<p>Techno is a function that recursively distributes differently-sized pyramids across the plane. The pyramids' base sizes differ but all have the same height. Frequency determines the size/spacing of the pyramids. Phase provides a horizontal offset. Power influences the pyramid height/contrast. The effect is like an aerial photograph of pyramids or city blocks. This component is generally used as a surface generator rather than as a mixer.</p> <p>Example files: Many of the ArtMatic Pro sample files use this component. Search for 'techno' to find some notable examples.</p>	<p>A: Frequency B: Phase C: Power</p>
 <p>Pyramids</p>	<p>Pyramids is a surface generation component very similar to Techno. The primary difference between the two is that the pyramid height is proportional to the base size.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Random Squares</p>	<p>This component generates discrete random rectangles when the component's input comes from the raw plane or another component with two outputs. The scale parameters determine the horizontal and vertical sizes of the rectangles and the amplitude is a multiplier to the output. Note that the output is discontinuous and does not tend to work well with the derivative component.</p>	<p>A: Scale X B: Scale Y C: Amplitude</p>
 <p>Random Noise</p>	 <p>This component generates random noise (like the snow on a television). Unlike Random Squares, this component's output while "noisy" is continuous with smooth transitions between the noise grains. This component is often used as a primitive for generating textures.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Fractal Noise</p>	 <p>This component is another flavor of noise that can be used for creating textures and for "texturizing" its input. This noise has a certain degree of structure which makes it great for creating marbled textures and clouds and other natural textures. This component is iterative and is sensitive to the Max iterations for fractals preference. The fractal noise component generates noise in different frequency ranges where the amplitude of the noise decreases as the frequency (value) of the input increases with a ratio of $1/f$ where f is the frequency/value input.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Multi Fractal Noise</p>	<p>Another flavor of Fractal Noise where the smoothness and fractal 'dimension' vary.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Low freq noise</p>	 <p>This is another flavor of random noise that is great for creating smoothly changing textures. The output is weighted towards lower values.</p>	<p>A: Amplitude B: Frequency C: Phase</p>

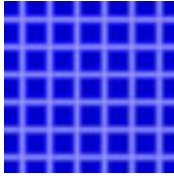
 <p>Crystal noise</p>	<p>This noise function has sharp edges and is useful both as a surface/texture generator (i.e. when the input is space) or as a mixer. Follow this component with Derivative to bring out the sharp edges. This component is a good primitive for mountains or torn paper or granite-like textures.</p> <p>Example: See "Crystal Noise z wipe" for a great use of this noise function to create a torn paper type effect.</p> <table border="1" data-bbox="383 405 1320 1066"> <tr> <td data-bbox="383 405 743 638"> <p>Crystal Noise used to mix a Grid and Ripples</p> </td> <td data-bbox="743 405 1127 638">  </td> <td data-bbox="1127 405 1320 638">  </td> </tr> <tr> <td data-bbox="383 638 743 1066"> <p>Crystal Noise used in a simplified version of the "Crystal Noise z wipe" example.</p> </td> <td data-bbox="743 638 1127 1066">  <p>Notice that the noise provides the 'holes' in the picture of the U&I logo</p> </td> <td data-bbox="1127 638 1320 1066">  </td> </tr> </table>	<p>Crystal Noise used to mix a Grid and Ripples</p>			<p>Crystal Noise used in a simplified version of the "Crystal Noise z wipe" example.</p>	 <p>Notice that the noise provides the 'holes' in the picture of the U&I logo</p>		<p>A: Amplitude B: Phase C: Frequency</p>
<p>Crystal Noise used to mix a Grid and Ripples</p>								
<p>Crystal Noise used in a simplified version of the "Crystal Noise z wipe" example.</p>	 <p>Notice that the noise provides the 'holes' in the picture of the U&I logo</p>							
 <p>Perlin noise</p>	 <p>Perlin noise is another special flavor of noise which can yield a variety of surface textures which can resemble stucco or cells in a petri and more.</p>	<p>A: Amplitude B: Level C: Frequency</p>						
 <p>Bump noise</p>	 <p>Bump noise has a hard-wired bump effect built in which gives the impression of raised and embossed textures. The generated surface is rougher than the other types of noise.</p>	<p>A: Amplitude B: Phase C: Frequency</p>						
 <p>Zebra noise</p>	<p>This component generates a zebra stripe-like pattern. The Amplitude parameter influences the waviness of the line. Thickness determines the line thickness and frequency determines the space between the stripes.</p>	<p>A: Amplitude B: Thickness C: Frequency</p>						
 <p>Fractal Lines</p>	<p>This component generates a system of veined lines and is actually the absolute value of fractal noise. It is great for creating marble-like textures.</p>	<p>A: Amplitude B: Phase C: Frequency</p>						

 <p>Smooth Entrelas</p>	 <p>This procedural component creates a randomized pattern that juxtaposes several predefined patterns. A typical pattern is shown below.</p> <p>The amplitude, phase and frequency control contrast, offset and spacing of the resulting pattern.</p> <p>Example files: "Bio Electronica" and "Entrelas z wipe" both use this component.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Circuitry</p>	 <p>This procedural component creates a randomized pattern that juxtaposes several predefined patterns. A typical pattern is shown below.</p> <p>Example files: "Fractal Trek" uses this texture in a recursive fractal voyage.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>ZigZag</p>	 <p>This procedural component creates a randomized pattern that juxtaposes several predefined patterns. A typical pattern is shown below.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Pict/Movie</p>	<p>This component lets you insert a picture file or movie (or iMovie DV Stream) into an ArtMatic system. The Size parameter scales the input image. When Size is 1, the input picture will fill the canvas at the default zoom level. (To set the zoom to the default, click the center view button.) The Tiling parameter determines the tiling of the input source. When Tiling is 0, no tiling is done. When tiling is greater than 0, the image will appear in repeated tiles if necessary to fill the canvas. The large the Tiling value, the closer together the tiles will be.</p> <p>The 2-in/1-out versions of this component use a black and white version of the input movie/pict and use the current gradient to color the resulting image. By preceding the tile with space distortion functions, you can warp and distort the image.</p> <p>There are several variations of the Pict/Movie component all of which are covered in depth in the chapter Using Pictures and Movies.</p> <p>Example Files: A large variety of examples are available. Search for files with UI, LOGO or PICT in their titles.</p>	<p>A: Size B: Contrast C: Tiling</p>
 <p>Pict/Movie Sym Mirror</p>	<p>This variation of the Pict/Movie component creates symmetrically mirrored tiles. For information about all of the Pict/Movie components, see the chapter Using Pictures and Movies.</p>	<p>A: Size B: Contrast</p>

 <p>Blurred Pict/Movie</p>	<p>This variation of the Pict/Movie component creates a blurred version of the input movie/pict. This component is usually in systems along with a parallel unblurred Pict/Movie tile. By combining the blurred and straight versions of the input source beautiful halo, chrome and other effects can be created. For more information about using pictures and movies as input sources, see the chapter Using Pictures and Movies.</p> <p>Example files: "Fire chrome" & "Light halo" are great examples which demonstrate how to use this component.</p>	<p>A: Size B: Contrast C: Blur</p>
 <p>Open Compiled Tree</p>	<p>Choose this component to use a 2-in/1-out compiled tree. Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine.</p> <p>Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.</p>	<p>A: Scale B: Iterations</p>




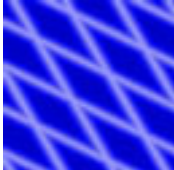


2D Vector Functions (2 in - 2 out)



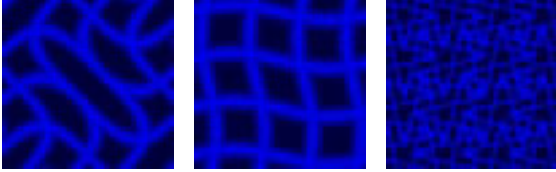
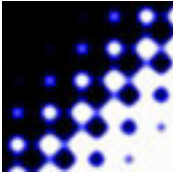

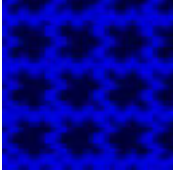










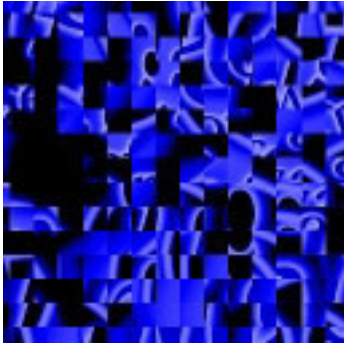
Basic grid generated by the structure shown at left


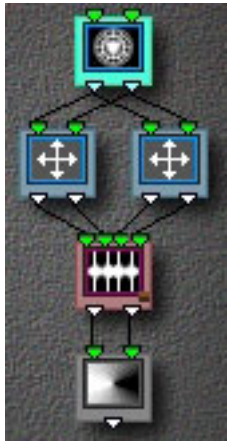







These functions are often used to distort or remap space. A good way to see what effect these functions have on a system is to use the **Basic** structure shown at the left with the **Grid** function as the second component and the **Ax + B** component (a simple scaling function which doesn't distort the output) at the last stage as shown in the picture to the left. Thumbnails of such a system are included for most descriptions in the table below.



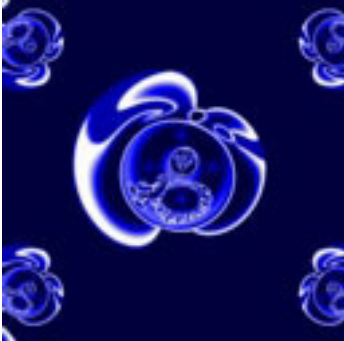






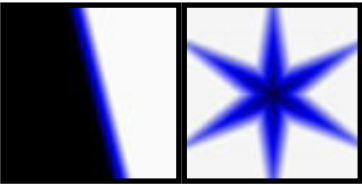
Some of these components can also act as two parallel 1D filters. An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone.




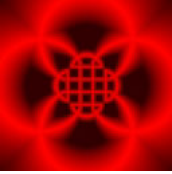





Function	Notes	Parameters
 Skew	 <p>Skew skews the incoming space. The skew angle parameter determines the angle of the skewing. Skew amount influences the amount of skewing, and scale zooms the space.</p>	A: Skew Angle B: Skew Amount C: Scale
 Scale Ax, By	<p>This simple primitive scales the two inputs independently. The scale parameters have a greater range than the similar Scale and Offset function. Note that the x output is solely a function off the x input, and the y output is solely a function of the y input, Hence, this component can be used as either a space transform function OR as two parallel 1D filters. When used as a space transform, this component acts as a magnification function.</p> <p>Left output: $A * x$ Right output: $B * x$</p>	A: Scale x B: Scale y
 Scale and Offset	<p>The Scale component simply scales the incoming space (by multiplying the x and w coordinates by the same factor) and allows it to be offset horizontally (the Offset x parameter) and/or vertically (the Offset y parameter). Note that a scale value of -1 inverts the space. Since the x output is a function of the x input and is not influenced by the y input, this component can be used as two parallel 1D filters.</p> <p>Left output: $(A * x) + B$ Right output : $(A * x) + C$</p>	A: Scale B: Offset x C: Offset y









 <p>Rotate A</p>	<p>This component rotates the space about a point. The angle parameter is the angle of rotation. The center x and center y parameters determine the point about which the space is rotated. By default, the center is 0,0. If the canvas has been shifted left or right, you may want to assign the center parameters to the values of the canvas center, if possible, when animating using this component.</p> <p>Tip: When this component is added to a system using the Insert popup menu's Insert Global Rotation command, the center parameters are automatically set to the system's center values (as displayed in the Camera Path dialog).</p>	<p>A: Angle B: Center x C: Center y</p>
 <p>$x + A \sin y,$ $y + A \sin x$</p>	<p>Distort space using sine waves. When the amplitude is 0, space is not distorted. A great variety of space distortions and patterns can be generated by adjusting the amplitude and frequency as demonstrated by the following pictures.</p>  <p>If a planar component $Ax+By+C$ is used in place of the grid component in the system illustrated above, the effect is an eggcrate-like effect as shown below:</p>  <p>Left (x) output equation: $x + A \sin y$ Right (y) output equation: $y + A \sin x$</p>	<p>A: Amplitude B: Frequency</p>
 <p>Waves</p>	 <p>This component bends space giving it a "waved" aspect. The distortion is created using the absolute value of the sine function. Frequency determines the spacing of the undulations. The amplitude parameters determine the amount of horizontal and distortion.</p>	<p>A: Frequency B: Amplitude x C: Amplitude y</p>
 <p>Sin warp</p>	<p>This component is very similar to the $x*A \sin y/y*A \sin x$ component and bends space with waves or an eggcrate-like effect.</p>	<p>A: Amplitude B: Frequency</p>
 <p>Sin x, sin y</p>	<p>This component tiles space by applying the sine function to the incoming values. The function's output always falls in the range of -1 to 1. This is a great way to create tiles and other repetitive patterns and textures.</p>	<p>A: Amplitude B: Frequency</p>



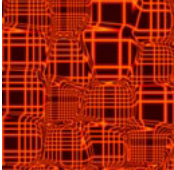

 <p>Hexagonal Sin</p>	<p>Like Sin x, sin y, this component breaks up space into tiles. In this case, space is broken up using a combination of sin transforms and hexagonal transforms.</p>	<p>A: Scale</p>
 <p>Hexagonal Mirrors</p>	<p>This component tiles space into hexagonal tiles.</p>	<p>A: Scale</p>
 <p>Hexagonal Tiles</p>	<p>This component creates a space of repeated hexagonal tiles. Each tile is a repetition of the space surrounding the origin of the incoming space. This is a great primitive for creating skin-like textures. Amplitude is the scale of the space within each tile while frequency controls the size of the tiles.</p> <p>Example files: The "Snake Skin" files provided typical examples.</p>	<p>A: Amplitude B: Offset C: Frequency</p>
 <p>Tile and Displace</p>	<p>This component generates a complex tiling of space. The displacement parameter creates non-repetitive tiles by applying different displacements within the tiles. When displacement is 0, the tiling is repetitive. Scale is the scaling of space within the tiles, and frequency determines the tile size.</p> <p>Tip: To create complex symmetry, follow this component with the Mirrors 4 component.</p>	<p>A: Displacement B: Scale C: Frequency</p>
 <p>Pixel prism</p>	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;">  </div> <div style="flex: 2; padding: 0 10px;"> <p>Pixel prism rearranges space by randomly breaking up the incoming space into little squares of uniform size. The effect is puzzle-like. To best visualize it, use the Movie or pict component after it as in the image to the left.</p> </div> <div style="flex: 1;">  </div> </div> <p>With displacement at 0 and scale at 0, there is no re-arrangement of the incoming space though there may be tiling (depending on the setting of the frequency parameter which determines the size of the squares).</p> <p>Tip: this component can yield particularly interesting results when used with the distance shaders.</p>	<p>A: Displacement B: Scale C: Frequency</p>

 <p>Disc Tiling</p>	<p>Warp space with an Escher-like tiling of space into a disc-shaped enclosed space. This is a great primitive for creating decorative figures.</p> <div data-bbox="332 226 1291 930" style="border: 1px solid black; padding: 5px;">    <p data-bbox="332 745 560 871">Structure of the example file "Chinese Fractal Sin z Plate"</p> <p data-bbox="576 640 917 808">An image created by the system shown at left. Notice how Disc Tiling encapsulates the space. The picture at right shows the untilted space.</p> <p data-bbox="933 640 1274 766">This is the image that results when the Disc Tiling component is removed from the structure at the left.</p> </div> <p>Example files: "Chinese Fractal Sin z Plate" .</p> <p>Tip: Try inserting the Disk Mirror component just before this one.</p>	<p>A: Smooth Edges B: Offset x C: Offset y</p>
 <p>Serpiensky tiles</p>	<p>Serpiensky tiles is an implementation of a classic fractal algorithm that starts with a triangle and recursively fills space with smaller copies of itself as shown in the component's icon. This component is sensitive to the Maximum Iterations for Fractals preference.</p>	<p>A: Amplitude B: Offset C: Frequency</p>
 <p>Ripples</p>	<p>This component distorts space with concentric circles. Amplitude provides the amount of distortion. With an amplitude of 0, there is no distortion. Frequency determines the spacing of the ripples.</p>	<p>A: Frequency B: Amplitude C: Phase</p>
 <p>Twirl</p>	<p>Twirl distorts space with a whirlpool pattern.</p> <p>Eric advises that you twirl everywhere, but you should avoid driving after doing so.</p>	<p>A: Scale B: Angle</p>
 <p>Multi twirls</p>	 <p>Multi twirls distorts space with randomly placed whirlpools. This is a great component for distorting pictures especially when creating animation. Note that this component can take a lot of time to render.</p>	<p>A: Frequency B: Phase C: Twirl</p>

 <p>Radial star</p>	<p>This component distorts space with radial undulates. Animate the outer phase parameter for nice exploding effects.</p>	<p>A: Outer phase B: Narrowness C: Amplitude</p>
 <p>Orbiters</p>	 <p>This is a complex component that re-arranges space into discrete, irregularly-shaped self-contained universes. It is like having little planet/universes each with its own co-ordinate system. The easiest way to see how this component re-arranges space is to follow it with the Movie or pict component as in the picture shown at left.</p>	<p>A: Rotation B: Offset C: Level</p>
 <p>Crystallize</p>	 <p>This component deforms space in a painterly way that makes it great for creating stylization effects for video processing. The image at left demonstrates its effect on the U&I Software logo.</p>	<p>A: Amplitude B: Frequency C: Phase</p>
 <p>Mirrors 3</p>	 <p>Mirror space into three parts with a 120 degree rotational symmetry. Use this and the other "mirror" components for kaleidoscopic effects. Scale scales the incoming space and the offset parameters provide horizontal and vertical offsets of the incoming space.</p>	<p>A: Scale B: Offset x C: Offset y</p>
 <p>Mirrors 4</p>	<p>This component creates a 4-way mirroring of the incoming space which is symmetrical about the x and y axes.</p> <p>Left output equation: Absolute value of x Right output equation: Absolute value of y</p>	<p>A: Scale B: Offset x C: Offset y</p>
 <p>Mirrors 6</p>	<p>This component creates a kaleidoscopic mirroring of space about three axes 120 degrees apart. Below are two images that demonstrate the effect. The image on the left does not have the component. The image on the right has this component added.</p> 	<p>A: Scale B: Offset x C: Offset y</p>

 <p>Branch N</p>	 <p>This component mirrors space as in the other "mirror" tiles but provides parametric control of the number of mirrors. The Branch # parameter allows you to achieve from 3 to 18 rotational symmetries. Unlike the other mirror functions, the axis symmetry is made seamless (continuous) which allows it to be used with the Derivative component (which abhors discontinuity). The offset parameters are the horizontal and vertical offsets of the incoming space.</p> <p>Example files: "Bio Electronica" and "Sea Star Creature".</p>	<p>A: Branch # B: Offset x C: Offset y</p>
 <p>Disk Mirror</p>	 <p>Disk Mirror provides a circular mirroring of the incoming space. This is another component that can be used to constrain space to prevent it from reaching infinity. The Disk Size parameter controls the size of the region that is mirrored. The image at left shows the result of placing the Disk Mirror component in front of a Grid.</p>	<p>A: Disk Size</p>
 <p>Complex inverse</p>	<p>Complex inverse inverts space using a complex number transformation so that infinity becomes 0 and 0 becomes infinity. The transformation divides x and y by distance.</p>	<p>A: Scale</p>
 <p>Complex Map</p>	<p>This component treats the x,y inputs as the real and imaginary parts of a complex number. As with Complex inverse, infinity becomes 0 and 0 becomes infinity. Four iterations of a recursive complex number equation are used to determine the output.</p>	<p>A: Real B: Imaginary</p>
 <p>Complex Map 2</p>	<p>This component is similar to Complex Map but uses a different equation which is a complex polynomial with a fixed number of iterations.</p>	<p>A: Real B: Imaginary</p>
 <p>Complex waves</p>	<p>This component warps the incoming space by applying a low frequency sine wave transform.</p>	<p>A: Scale B: Offset</p>
 <p>Random noise</p>	<p>This component provides continuous random distortions of space which can be subtle or extreme, depending on the amplitude setting.</p>	<p>A: Amplitude B: Phase</p>




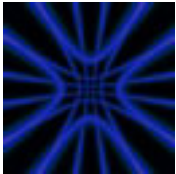
 <p>Turbulence 2D</p>	<p>This is a somewhat 'structured' noise that is sensitive to the Max iterations for fractals preference. This component is great for creating turbulence, fire, marble and clouds. See the notes for the scalar version of this component.</p> <p>This component was formerly called Fractal Noise.</p>	<p>A: Amplitude B: Scale C: Phase</p>
 <p>Low Freq Random</p>	<p>See the notes about the scalar version of this component.</p>	<p>A: Amplitude B: Frequency C: Phase</p>
 <p>Directional Random</p>	<p>This component distorts space directionally so that you can distort the system either vertically or horizontally or a combination of the two. The direction of the distortion is provided by the direction parameter. When direction is 2 or -2, space is distorted horizontally. When it is set to 0, the distortion is purely horizontal. In-between values, provide both vertical and horizontal distortion.</p>	<p>A: Amplitude B: Direction C: Phase</p>
 <p>Random Fractal Space</p>	<p>This noise function creates a structured noise that is similar to that created by Turbulence 2D. Unlike the other noise functions, Random Fractal Space generates truly random output (rather than adding randomized offsets to the incoming values) which results in the new coordinates being truly random. Place this component before a Grid to create veined textures, or place it before any other surface generator to create beautiful random textures.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Distance Mirror</p>	<p>This component inwardly mirrors distance beyond a certain radius and hence is another component that can be used for avoiding infinities.. This is useful for both cleaning up systems that are noisy as they approach infinity and for closing shapes.</p>	<p>A: Amplitude B: Mirror C: Distance</p>
 <p>Ring Space</p>	<p>This component warps space about several attractors. As with many of the complex number-based components, this component can be used to avoid infinities. The presence of the three attractors makes this component terrific for creating flower-like shapes.</p>	<p>A: Offset x B: Offset y C: Warp</p>
 <p>Polar Space</p>	<p>Use this component to interpret x and y as polar co-ordinates where x is the angle and y is distance.</p>	<p>A: Polar Scale B: Distance C: Scale</p>
 <p>Bubble Space</p>	<p>As with the Orbiters component, bubbles are create each of which is an enclosed space. Each bubble has 0 at its center, and each bubble is surrounded by 0. This component is terrific for creating non-repeating textures and patterns.</p>	<p>A: Amplitude B: Bubble size C: Frequency</p>








 <p>Facet Space</p>	 <p>Create a multitude of irregularly-spaced tiles of the incoming space. This creates 3D-like repetitions of the image created by the components that follow it as in the image below in which Facet Space is used before a movie/pict component.</p> <p>Scale determines the scaling applied to the incoming space. Delta size determines the range of sizes into which the space is tiled. With a Delta Size of 0, all the tiles are the same size. Large values provides a great variation of sizes. Frequency determines the tile spacing.</p> <p>Note: This component was added in version 2.53 and is different from the component called Facet Space in the original ArtMatic Pro release. The old component has been renamed <i>Facet Space Mirrors</i>.</p>	<p>A: Scale B: Delta size C: frequency</p>
<p>Facet Space Mirrors</p>	 <p>This component randomizes the incoming space and outputs a space with warped facets/tiles. The example at left shows the result of placing it in front of a Grid. The transitions between the facets is smooth which makes this a great tool for creating natural semi-random textures.</p> <p>Example file: "Monster's skins".</p> <p>Prior to version 2.53, this component was called <i>Facet Space</i>.</p>	<p>A: Scale B: Delta Size C: Frequency</p>
 <p>Open Compiled Tree</p>	<p>Choose this component to use a 2-in/2-out compiled tree. Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine.</p> <p>Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.</p> <p>The Blend parameter determines whether the compiled tree uses the incoming space for its input values (when Blend is at its maximum--which is the default) or the undistorted space or an interpolation of the values. The Recursion parameter lets you feed the output of the compiled tree back to itself to create your own fractals. Normally, recursion should be set to 0 as recursion is only meaningful with compiled trees designed to be recursive.</p>	<p>A: Blend B: Recursion</p>




2-in/3-Out Components

These components perform a variety of functions depending on how they are used in the structure tree. Typical uses are to translate 2D to 3D space, create 3D objects (spheres, cubes, rooms and the like), and to provide RGB colors which to shade the incoming space or objects. Many of the functions in this group, create 3D objects. Click here to learn more about [3D Objects](#).

An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone.

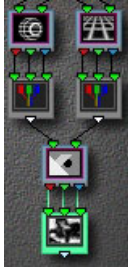
Function	Note	Parameters
 <p>3D zset</p>	<p>Use this component to create a third co-ordinate (whose value is determined by parameter A) that can be passed on to 3D functions or used as a controller value. The incoming x and y values are passed unchanged through the leftmost outputs. Parameter A's value is sent out the rightmost outlet. Animating this parameter is a great way to control solid textures such as 3D noise functions.</p> <p>Use this function at the top of the system if you want to use 3-in/3-out components near the top of the tree but do not want to use the system's third global input (time) which is incremented with every frame.</p>	A: z value
 <p>3D Axonometric</p>	<p>Use this component to create a third co-ordinate whose value is related to the incoming x,y values. Unlike 3D zset the third output (the z value) is not a constant value but calculated from the x and y values. The scale parameters are multiplied by the x and y inputs before passing them out.</p> <p>The outputs are calculated using the following formulas:</p> <p>Left output = Ax Middle output = By Right output = Ax/2 + By/2</p>	A: Scale x B: Scale y
 <p>Spherical</p>	<p>Spherical is a space transform function that creates a 3D space/object by projecting the incoming space onto a half-sphere. The effect is similar to the image that results from holding a mirrored ball over a picture. The image below shows the result of preceding the 3-in/1-out Grid component with this function.</p>  <p>The radius parameter controls the size of the half-sphere used for the projection. The smaller the radius is, the greater the resulting space distortion. This is a great component for radiant and supernova-type effects.</p> <p>Example files: "SphericalWipe", "Spherical Noise Wipe", "SphericalStainedGlass"</p>	A: Radius B: Offset x C: Offset y

 <p>3D plane</p>	<p>Create a 3D plane which can be banked and tilted. Like all 3D object components, the area outside the object is mapped to infinity (which is painted with the depth cueing color). The components which follow the plane component provide the surface's texture. Banking x controls the horizontal tilt, Plane offset provides an altitude offset, and z Slope controls the forward/back tilt.</p>	<p>A: Banking x B: Plane offset C: z Slope</p>
 <p>3D sphere</p>	<p>Create a sphere which can be moved along the x, y and/or z axes. The surface texture is created by the components which follow this component.</p> <p>Tip: Rotation can be simulated by using a noise component after the sphere and animating the noise component's phase parameter.</p> <p>Example Files: Search for "sphere" to find the many examples which use this component.</p>	<p>A: Offset x B: Offset y C: Offset z</p>
 <p>3D ellipsoid</p>	<p>3D ellipse primitive.</p>	<p>A: radius B: vertical skew C: offset z</p>
 <p>3D tube</p>	<p>Create a 3D tunnel whose radius and offset can be modified.</p> <p>Example Files: Search for "tube" to find the many examples which use this component.</p>	<p>A: Offset x B: Offset y C: Radius</p>
 <p>3D cube</p>	<p>Create a cube which can be rotated in space.</p> <p>Example Files: Search for "cube" to find the many examples which use this component.</p>	<p>A: Angle xy B: Angle zx C: Offset z</p>
 <p>3D box</p>	<p>A useful 3D object primitive. The size y parameter controls the size of the box top. The size zx parameter controls the box length. The offset z parameter controls the box offset along the z axis.</p>	<p>A: size y (face width) B: size zx (box length) C: offset z</p>
 <p>3D room</p>	<p>Create a 3D room. The z offset controls the distance to the back of the room. The angle zx controls the angle with which the rear walls are joined. The room size controls the size of the virtual room. The room has a finite size. At the maximum offset, you can see the room from the outside.</p>	<p>A: Offset z B: Angle zx C: Room size</p>

 <p>3D parametric cube</p>	<p>Create a cube whose texture 'sticks to it'. The non-parametric 3D objects move under the texture while a parametric object's covering texture moves with the object.</p> <p>Example files: "Parametric cube".</p>	<p>A: Angle xy B: Angle zx C: Offset z</p>
 <p>3D parametric room</p>	<p>Create a room whose surface texture sticks to it. (The non-parametric version moves 'under' the texture.) Unlike the non-parametric version, this room has infinite walls. So, you can't leave the room.</p> <p>Example files: "Parametric room". Note how the depth cueing in the example is controlled via a loose 1D filter connect to the room's z output.</p>	<p>A: Offset z B: Angle zx C: Room size</p>
 <p>3D parametric face</p>	<p>Simple 3D-object which is like a single face of a cube. The co-ordinates are parametric so the textures moves with the object.</p> <p>Use this object to perform 3D rotation of your movies and pictures.</p> <p>Example files: "Rotating face".</p>	<p>A: Angle xy B: Angle zx C: Offset z</p>
<p>Packed Mixers Family</p>	<p>The group of components whose names include 'packed' is used for mixing packed streams--usually from RGB components or branches. When used in iterative systems, these components have memory and in each iteration mix the stored value with the current iteration's value and then store the resulting value.</p> <p>See the chapter Compiled Trees and Iteration for more information about iterative systems and memory components.</p> <p>When infinity is encountered in one of the inputs, it is treated as transparent by these mixing functions.</p> <p>These functions can accept both packed and unpacked inputs. The first input acts as the background in iterative systems and is mixed with the final result.</p>	

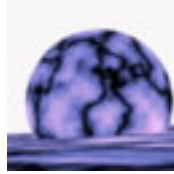


Packed depth sort



This component mixes two 3D objects into a single set of 3D outputs. The inputs should be the packed outputs of 3D objects as shown in the structure to the left (which is a simple modification of the **3D Sphere and Plane** item in the **Structures** popup menu).

NOTE: This component should always receive its input from **Pack** components (that receive their input from 3D object components). The algorithm is such that for each set of points received, the set with the lowest value of z wins; in other words, the co-ordinates closest to the viewer win.



None



Packed RGB crossfade

Mix two RGB color streams into a single image. When the parameter slider is all the way to the left, only the left input's image is visible (with one caveat noted below). When the slider is to the right, only the right input's image is visible (with one caveat). When the parameter is between the maximum and minimum values, a mix of the two images is created.

There is a situation in which portions of both images are visible even though the **Interpolate** parameter is at its extremes. If either input stream has a 3D object, then any points that have the value infinity (which means that they are outside the object's boundaries) will be transparent regardless of **Interpolate's** value. This allows you to have opaque 3D objects which appear against a background created by another RGB color stream.



Usage note: The inputs should come from **Pack** components that are connected to components whose outputs are RGB colors. This component is used by the **RGB 2 Channels** system found in the structures menu.

Tip: Not only can you mix two color streams, but you can also use this component to fade between two independent images/systems/pictures/movies.

Example files: "Cube & Pict" is just one of many of the example files that use this component.

A:
Interpolate



Packed RGB mul

Filter one color stream (image) with another by multiplying the corresponding RGB values of the incoming streams. **Amount** determines the strength of the filtering. **Offset** mixes back in some of the left input's image so that in conjunction with the **Amount** parameter, one can have very fine control over the filter effect.

RGB Math: Every RGB color is made up of three numbers which determine the amount of red, green and blue that make up a pixel's color. When doing multiplication, ArtMatic treats color values as ranging from 0 to 1. [0,0,0 is black. 1,1,1 is white. 1,0,0 is red. 0,1,0 is green, and so on). Unless all pixels are at maximum brightness, multiplication tends to reduce an image's brightness.

A: Amount
B: Offset



Packed RGB add

Mix two images (RGB color streams) by adding the values of the merged streams pixels. The two parameters control the relative contribution of the two images to the final mix.

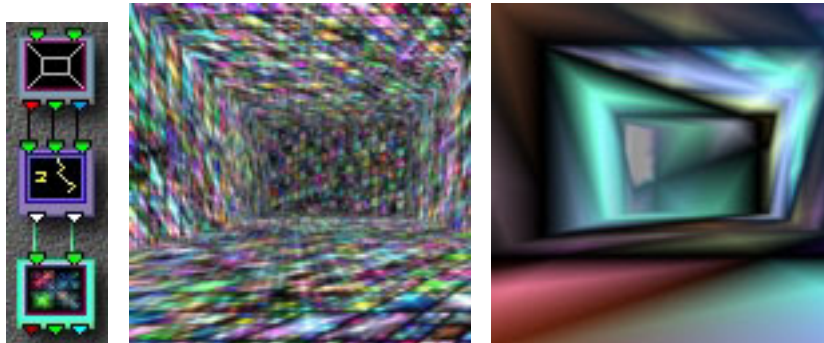
A: Level in1
B: Level in2





RGB Shaders Family





The RGB shaders are a group of components that are used to color a space with RGB-based color. Except where noted, these components should receive a space as input rather than inputs from independent surfaces.






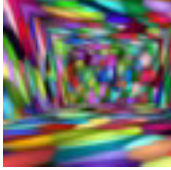

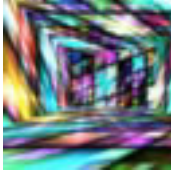

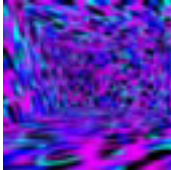

These RGB components are the RGB (true color) equivalents of the 2-in/1-out surface/texture functions used in gradient-based systems. In both cases, the components generate a single color-value based on the incoming points (co-ordinate pairs). In a gradient-based system, there is a single output which is mapped to a color of the current gradient. In the 2-in/3-out RGB shaders, the three output values (which we call an RGB triplet) together define a single RGB color. RGB components are generally used in the lower part of a tree that terminates with RGB output (three outputs). The shader need not be the last component since you might follow them with further color manipulation.





The nature of these shaders is such that they not only provide color but may also provide the illusion of space manipulation in addition to providing color. This is particularly noticeable when manipulating a noise function's frequency parameter. In the structure shown below, the only difference between the images is the setting of the frequency parameter.



 <p>RGB plain color</p>	<p>Generate a constant RGB color (or three constant values) determined solely by parameters A, B and C. The input values are completely ignored. The color is strictly a function of the parameter values and is the same for every incoming value. This is useful for creating solid color backgrounds or surfaces.</p> <p>Tip/Trick: Advanced users can use this component to generate three independent constant values which can be useful for a number of non-color manipulation applications.</p> <p>Example file: "Cube& Solid Color".</p>	<p>A: Red B: Green C: Blue</p>
 <p>Shaded plain color</p>	<p>Another useful RGB-based shader. The A, B and C parameters (red, green and blue respectively) define a target color. The left-input's value (the x-value) determines whether the output value is the target color (when the value is greater than 0) or its complement (when its value is less than 0). When the value is near 0, the luminosity of the color is reduced. (Hence, an x-value of 0 generates black.)</p> <p>The right-input value (the y-value) determines the luminosity (brightness) of the output color. Large values of y generate a shade of the color that approaches white.</p>	<p>A: red B: green C: blue</p>
 <p>RGB color shade</p>	<p>RGB Color Shade provides color shading by using the inputs to control the luminance of the generated color. The color (hue) is provided by parameter A, Hue; parameter B, Saturation, determines the maximum color saturation. The luminance (brightness) is determined by one or both input values. Parameter C, Luminance Pan, determines whether a point's brightness is determined by both input values or just one of them. When the parameter is at its minimum, only the left input is used; when the parameter is at its maximum setting, only the right input is used.</p> <p>Use this component to create subtle shadings of a single hue, such as when creating clouds, mist, or earth textures. This component can take input from either a space or two individual inputs.</p> <p>Example files: "Compiled Trees & Recursion:Advanced Compiled Examples:Planet with life Sunrise". This component is used in the file's subtrees to provide color for the space clouds, the star, and the planet's cloud cover.</p>	<p>A: Hue B: Saturation C: Luminance Pan x<->y</p>
 <p>RGB main gradient</p>	<p>Create RGB- color output using the colors of the active ArtMatic gradient. This component is equivalent to the 2-in/1-out $Ax + By + C$ component. The two input values are added together (after being multiplied by the appropriate scale parameter) and the associated color from the gradient is chosen. This handy component lets you mix ArtMatic gradient-based colors with RGB-based components. This technique is commonly used to mix ArtMatic-based structures with color pictures and movies.</p> <p>Just as with the 2-in/1-out $Ax + By + C$, you can use either a space as an input or two surfaces. When the input is two surfaces, the scale parameters provide a mix control.</p> <p>Example file "Entreelas z wipe".</p>	<p>A: Scale x B: Scale y</p>

 <p>Shaded main gradient</p>	<p>Create shaded RGB-color output using the colors of the current ArtMatic gradient. 3D shading and lighting effects can be easily created with this shader, especially when the second input is fed by the Derivative or 2D Derivative component. The first input selects the color and the second input controls the lighting/shading.</p> <p>Gradient scale determines the spacing/scale of the gradient. Luminance scale determines to what extent the y-values influence brightness and is similar to RGB color shade's Light/Hue variance parameter. Gradient offset adds an offset to the x-value before picking the color from the gradient.</p> <p>Tip: Set the Luminance scale parameter to something other than 0 to achieve effects similar to Artmatic's global shading. This technique is particularly effective when the 2D Derivative component feeds this input.</p> <p>Tip: It is often useful to feed this component's inputs from two parallel 1D filters that are connected to the same single-output parent. Learn more about this technique in the Getting Deeper chapter of this manual.</p> <p>Example file: "Abstract Beast". This example file is discussed in detail in the Getting Deeper chapter of this manual.</p>	<p>A: Gradient scale B: Luminance scale C: Gradient offset</p>
 <p>RGB color gradient</p>	<p>Create RGB-color output by projecting RGB color-complements across an axis defined by the three angle parameters. A wide variety of vibrant color and lighting effects can be achieved with this component. This is similar to the RGB radial hues component with the addition that it allows you to set up a custom color wheel.</p> <p>Example files: "Techno stained glass", "RGB Mixed synth"</p>	<p>A: Angle Red B: Angle Green C: Angle Blue</p>
 <p>RGB Concentric hues</p>	<p>Shade the image with a narrow adjustable range of hues that vary with the distance from the origin. The luminosity decreases in inverse proportion from the point's distance from the origin (0,0). Parameter A defines the 'base hue' used for shading, and Parameter B controls the color saturation. Parameter C determines the spacing of the hue variations.</p> <p>A concentric narrow adjustable range of hues. Luminosity becomes lower as the distance from zero increases (and then starts increasing again).</p> <p>Example file: "Monster's skins"</p>	<p>A: Hue Rotation B: Saturation C: Frequency</p>
 <p>RGB linear hues</p>	<p>Color the image by using x input value to define the pixel's hue. If the y input value is negative then the inverse hue is used.</p> <p>Example files: "Neuroweb Lights", "Differential system". In "Neuroweb Lights" this component's output is sent to a 3-in/3-out shading function so that it acts like a space distortion function.</p>	<p>A: Hue Rotation B: Saturation C: Frequency</p>





 <p>RGB radial hues</p>	<p>Color the image by choosing colors from a classic color wheel (shown in the component's icon). The two inputs are treated as an x/y co-ordinate pair. The angular position of the input point (the value pair) determines the hue. Parameter A rotates the color wheel and thus changes the color which corresponds to a particular angle. The Saturation parameter determines the luminance (brightness) of the colors. This component is frequently used to create vibrantly colored surfaces.</p> <p>When this component is used as the color shader, it is as if the original undistorted space (the flat plane) is a sheet of paper with the color wheel printed on it and the system's space distortions and remapping are really distortions and remapping of that sheet of paper with the color wheel on it. The more chaotic the distortions, the less like a color wheel the image will appear.</p>	<p>A: Hue Rotation B: Saturation</p>
 <p>RGB soft random hues</p>	<p>Color the incoming space with random modulations of a narrow range of hues. Naturalistic effects are achieved with a low value for the saturation parameter. At the minimum saturation, only a single hue is left. Hue rotation determines the base color for the variations. Saturation determines the average brightness of the resulting colors, and Frequency determines the density of the distortion.</p> <p>Example file: See "Pink turbulent marble" to see a range of this component's possibilities.</p>	<p>A: Hue Rotation B: Saturation C: Frequency</p>
 <p>RGB Random Fractal</p>	 <p>Randomly-generated color texture which is sensitive to the setting of the Max. Iterations for Fractals preference. The amplitude controls the saturation. A great texture for vibrant color textures.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Color Bubbles</p>	 <p>Color the space with randomly-sized and colored bubbles/facets.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>3D Color Pyramids</p>	 <p>Color the space with randomly-sized and colored pyramids.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>String noise</p>	 <p>Color the space with random colored 'strings'.</p>	<p>A: Amplitude B: Hues C: Frequency</p>
 <p>Techno Boxes</p>	<p>Techno boxes is decorative RGB shader that will generally be used to provide surface texture.</p>	<p>A: angle B: center x C: center y</p>








 <p>RGB Pict/Movie</p>	<p>This is the true color version of the Pict/Movie component. For more information about using pictures and movies as input sources, see the chapter Using Pictures and Movies.</p>	<p>A: Size B: Contrast C: Tiling</p>
 <p>RGB Pict/Movie Sym Tiled</p>	 <p>This variant of the color pict/movie component creates mirrored tiles of the picture when the picture is zoomed out. For more information about using pictures and movies as input sources, see the chapter Using Pictures and Movies.</p>	<p>A: Size B: Contrast</p>
 <p>Open compiled tree</p>	<p>Choose this component to use a 2-in/3-out compiled tree. Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine. Compiled Trees are covered in more detail in the Compiled Trees chapter of this manual.</p> <p>Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.</p>	<p>A: Scale B: Iterations</p>








3D Scalar Functions (space/field functions) (3 in - 1 out)






In the descriptions below, x, y and z refer to the components' first, second and third inputs. They are not necessarily x,y,z spatical co-ordinates since that depends entirely on the components which are feeding the inputs. These components sometimes generate a surface/texture from 3D co-ordinates and sometimes they mix three surfaces into a single composite surface. Many of these functions are essentially identical to the 2D versions of the same functions. For detailed explanations of those functions, see the 2D description.









An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone.




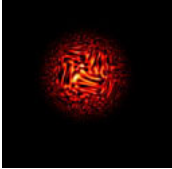



Function	Notes	Parameters
 Ax + By + Cz	<p>This component provides a weighted mix of the inputs. The three scale parameters act like 'level' controls which determine how much of each input is present in the output. An input is essentially ignored if the corresponding scale parameter is set to 0.</p> <p>Formula: $Ax + By + Cz$</p>	<p>A: Scale X B: Scale Y C: Scale Z</p>
 (Ax + By + Cz)/Norm	<p>This component can be used as mixing function and a 3D shading function. This component will contain the space by virtue of dividing the output by the distance which forces the output to approach zero as x, y, and z approach infinity.</p> <p>Formula: $\frac{(Ax + By + Cz)}{(\sqrt{x^2 + y^2 + z^2})}$</p>	<p>A: Scale X B: Scale Y C: Scale Z</p>
 Distance (x, y, z)	<p>This component is a common building block in complex structures. It is the 3-input version of the 2-in/1-out Distance component and calculates the Euclidean distance between the point defined by the x,y and z inputs and the origin.</p> <p>Formula: $(\sqrt{x^2 + y^2 + z^2})$</p>	<p>None</p>
 (x:y) * z	<p>This component mixes the x and y inputs by interpolating between them and uses the z input as a multiplier. This is a great way to animate a transition from one image to another using the z input as an intermediate distortion factor.</p> <p>The z input acts as a control value.</p>	<p>A: Balance x y B: Balance z</p>

 <p>x*y + z</p>	<p>This component is both a useful as a mixer and as a basic mathematical building block for those that are implementing specific equations. As a mixer, it is useful for cases where you want to filter two images together (i.e. using one as a mask) and mix another unmasked image with them.</p> <p>Example file: "Burning Logo RGB". Notice that the x-input to the function provides the flames. The y-input is a picture of the U& I logo which acts as a mask, and the z input adds in a color version of logo.</p> <p>Formula: $A(x * y) + B(z)$</p>	<p>A: Amplitude x*y B: Amplitude z</p>
 <p>x * y * z</p>	<p>This component creates a complex mix/filtering effect where the three inputs filter each other.</p> <p>Formula: $A(x * y * z)$</p>	<p>A: Scale</p>
 <p>z blend</p>	<p>This component mixes the x and y inputs using the z-input as an alpha channel/mask.</p>	<p>A: Amplitude</p>
 <p>Min <-> Max</p>	<p>This is essentially the same as the 2D version of this component but with the interpolation being done across three inputs. See the description of the 2D version for more information.</p> <p>The Smoothing parameter is new version 2.5 and controls the continuity of the transition between the inputs. When it is set to 0, the function behaves as it did in earlier ArtMatic versions.</p>	<p>A: Amplitude B: Min to Max C: Smoothing</p>
 <p>Max(x, y, z)</p>	<p>This procedural component passes out the maximum of the three inputs and multiplies it with parameter A's value.</p>	<p>A: Amplitude</p>
 <p>Z Sin shader</p>	<p>This component can generate complex shading/banding where the shading is controlled by the z input.</p> <p>Basic Formula: $\sin(x + y + Az)$</p> <p>Note that the basic equation is further modulated by $B * z$ and the contrast parameter.</p>	<p>A: Amplitude B: Z scale C: Contrast</p>
 <p>Z Circles</p>	<p>This component has a pointillation, pixellation or dither-type effect and only works well with some sorts of systems. If the first two inputs define a spatial position, the z value will effectively control the amount of pixellation.</p>	<p>A: Frequency B: Contrast C: Altitude</p>

 <p>Z Bubbles</p>	<p>This function distorts space with superimposed cylinders/bubbles. The interaction between the inputs can be quite complex. Animating the phase parameter can create fascinating mixing, morphing of the inputs and the distortion bubbles.</p>	<p>A: Scale B: Contrast C: Phase</p>
 <p>Fractal Lines</p>	<p>This is a variation of 2D fractal noise. The z input provides an offset that modulates the noise. For more about fractal noise, see the description of the 2D version.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Fractal Map</p>	<p>Fractal map is a complex number-based recursive fractal where the x and y inputs are treated as the real and imaginary parts of a complex number. The Z displace parameter is used to modulate parameter A in such a way that you can effectively modulate the "granularity" of the fractal with the z-input. The z -input can be used to explore changes in the fractals degree of chaos. In some ranges, the fractal behaves in an orderly fashion while in others it can appear quite chaotic.</p> <p>This component is sensitive to the Max iterations for fractals parameter.</p>	<p>A: Real B: Imaginary C: Z displace</p>
 <p>Mandelbrot set</p>	<p>This component is the classic Mandelbrot set. You can create interesting variations by feeding the output of a 2D space distortion function into the leftmost inputs of this component. The scale parameter simply acts as a zoom control. The z distortion parameter warps the resulting image, and the contrast parameter adjusts the color contours of the resulting output.</p> <p>This component is sensitive to the Max iterations for fractals parameter.</p>	<p>A: Scale B: Z distortion C: Contrast</p>
 <p>Julia sets</p>	<p>This component generates a Julia set. For the classic Julia Set (as shown in the component's icon), set parameter C to 0. When parameter C is anything other than 0, the z input is used to modulate the "granularity" of the fractal. The A and B parameters are multipliers applied to the x and y inputs respectively.</p> <p>This component is sensitive to the Max iterations for fractals parameter.</p> <p>Tip: Due to the limited output range of this component, the logarithm-based shaders often work best.</p>	<p>A: Real B: Imaginary C: Z displace</p>
 <p>Ferro magnet model</p>	<p>This is a complex fractal based on the magnetic field equations. The A and B parameters are multipliers for the x and y inputs. The Z displace parameter works similarly to the Fractal Map's and Julia Set's Z displace parameters.</p> <p>This component is sensitive to the Max iterations for fractals parameter.</p>	<p>A: Real seed B: Imaginary seed C: Z displace</p>
 <p>Pack(x,y,z)</p>	<p>Use this component to pack the output of 3 output functions into a single packet which can be passed to packed functions. The output of this function should only be passed to components that expect packed input.</p>	

 <p>3D Grid</p>	<p>This function creates a true 3D Grid (constructed of interlocking walls) when the inputs are a 3D space (as opposed to inputs from three independent branches). To get a sense of this as a 3D function, place it after the 3D Room (2-in/3-out) component and manipulate the room's offset z parameter which moves the room's back wall. (The "3-to-1 3D Explorer" example file has been set up to explore the 3-to-1 true 3D functions).</p> <p>This component can also be used as a mixer to mix the output of independent branches with interesting effect.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>3D Sin x * Sin y * Sin z</p>	<p>This component tends to create dimpled or egg-crate like textures. It can be used either as a mixer or as a 3D texture generator.</p> <p>Note: The number of parameters for this component have changed in ArtMatic Pro 2.5. As a result of this change, the keyframes of files made with earlier versions of ArtMatic Pro will need to be manually adjusted.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>3D Sin x + Sin y + Sin z</p>	<p>This component tends to create dimpled textures. It can be used either as a mixer or as a 3D texture generator.</p> <p>Note: The number of parameters for this component have changed in ArtMatic Pro 2.5. As a result of this change, the keyframes of files made with earlier versions of ArtMatic Pro will need to be manually adjusted.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
<p>3D Noise Functions</p>	<p>The next group of components is comprised of true 3D noise/texture functions. Noise functions are great for providing textures to cover the surfaces of objects as well as for creating standalone textures and images.</p> <p>Generally, these are used towards the bottom of branches to provide texture for 3D objects. A simple example file ("3-to-1 3D Explorer") has been provided to make it easy to explore these fascinating functions. To see the 3D effect, adjust parameters for the example's Room tile and notice how the texture of the back wall changes.</p> <p>TIP: When exploring, make sure to explore what happens when you use the Derivative function as the final component as this will bring out the texture of the noise functions.</p>	
 <p>3D Random</p>	<p>This component generates a 3D random space distortion and is related to the 2D random noise component. Great for granite-like textures!</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>3D Fractal noise</p>	<p>Three-input fractal noise algorithm. This component is sensitive to the Max. Iterations for Fractals preference. See the description of the 2D version for more information.</p>	<p>A: Amplitude B: Phase C: Frequency</p>





 3D Fractal Stucco	A variation of of the fractal noise algorithm that generates a stucco-like texture. See the <i>Shining Stucco Gelatin</i> example file. Note that the Power function provides a metallic effect.	A: Amplitude B: Phase C: Frequency
 3D Multi Fractal Noise	Another fractal noise variant with a turbulent aspect created by varying smoothness in the resulting texture.	A: Amplitude B: Phase C: Frequency
 3D spots	3D texture reminiscent of a star field.	A: Amplitude B: Phase C: Frequency
 3D Bubbles	3D version of the 2D Bubbles component. The texture is made up of irregularly shaped and tiled 'bubbles'. The texture is reminiscent of skin and cell textures.	A: Amplitude B: Phase C: Frequency
 3D hexa noise	3D version of the 2D Crystal Noise component. This is a structured fractal-based noise and is sensitive to the Max. Iterations for Fractals preference. Increasing the iterations, increases the detail. This is a very versatile texture function that can be used for everything from clouds to rock to torn paper. Be sure to experiment with different settings of the the Max. Iterations preference (a good range to explore is from 5 to 50). Also, see what happens when the Derivative function is used after this one.	A: Amplitude B: Phase C: Frequency
 3D Stratified Noise	A simple approximation of perturbed geological layers (a texture used widely in Bryce). The distortion parameter controls the texture's vertical displacement. Example file: "Solid Strata Time Room" shows how the 3D stratified noise intersects a Room. Note how the y and z co-ordinates are modulated by time to create a constantly moving texture.	A: Amplitude B: Distortion C: Frequency
 3D techno noise	True 3D version of the 2D Techno Noise component.	A: Amplitude B: Phase C: Frequency
 3D fractal lines	3D fractal-based texture made of randomly intersecting curving veins. Great for marble and other veined textures. This is a structured fractal-based noise and is sensitive to the Max. Iterations for Fractals preference. Increasing the iterations, increases the detail.	A: Amplitude B: Phase C: Frequency






 <p>3D Random lines</p>	<p>Similar to fractal lines but composed of randomly intersecting straight lines. Great for fibrous and geological textures. This is a structured fractal-based noise and is sensitive to the Max. Iterations for Fractals preference. Increasing the iterations, increases the detail.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Facet pers(pective)</p>	<p>This component generates a mosaic pattern with a perspective effect imposed by the z-input.</p>	<p>A: Frequency B: Z Scale C: Contrast</p>
 <p>Random pers(pective)</p>	 <p>This component provides smooth random distortions of the x and y inputs and uses the z input as a distance/perspective controller.</p> <p>Tip: To create an animatable planet/surface, use the distance component to feed the z input. Animating the phase parameter will provide surface undulations that resemble planet rotation. The image at left was created with the file Random Perspective which is found in the Doc. Example Files folder.</p> <p>If the z-input is fed by the Ax + By + C component, you will get a horizon-like effect.</p>	<p>A: Perspective B: Level C: Phase</p>
 <p>Turbulent pers(pective)</p>	<p>This component is similar to Random Pers but uses a turbulent noise rather than a random noise-type algorithm for generating its texture. This component works well for planet surfaces.</p>	<p>A: Perspective B: Level C: Phase</p>
 <p>Techno pers(pective)</p>	<p>This component is similar to Random Pers and Turbulent Pers but uses the techno texture. See the description of the 2D scalar techno component for more information about the techno texture.</p>	<p>A: Perspective B: Amplitude C: Phase</p>
 <p>Open Compiled Tree</p>	<p>Choose this component to use a 3-in/1-out compiled tree. Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine.</p> <p>Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.</p>	<p>A: Scale B: Iterations</p>






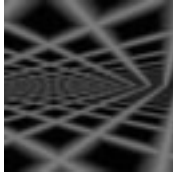
3D Vector Functions (3 in - 2 out functions)


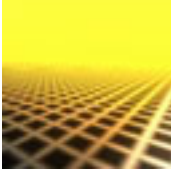







Like the other vector components, these **3D vector functions** warp or distort space when inputs **x** and **y** come from the raw plane or the output of another vector function. By convention, we refer to the third input to these components as **z**, but you should keep in mind that for many of these components the "z" input does not act as a spatial co-ordinate. The **z** input often acts as a control input that influences the transformation though in some cases it may act as a third spatial co-ordinate for components that transform 3D space into 2D space.






An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone.



Function	Notes	Parameters
 z Scale and Offset	<p>This component distorts the x and y inputs by scaling and offsetting the incoming values before passing them through to the left and right outputs respectively. The z input controls the amount of distortion. The A and B parameters control the x and y distortion and the C parameter scales the influence of the z input.</p>	<p>A: X distortion B: Y distortion C: Z Scale</p>
 z Displace	<p>This component uses the z input to distort the x and y inputs independently according to the displace X and displace Y parameters. It differs from z Scale and Offset in that the value of z is added to x and y rather than multiplying them. It has a skew effect on a simple plane</p>	<p>A: Displace X B: Displace Y C: Amplitude</p>
 z Multiply $x*z$, $y*z$	<p>Scale the x and y inputs using the third (z) input to control the scaling. This is also a basic mathematical building block for creating systems from your own mathematical formulas.</p> <p>Tip: When used at the top of the system, this component can be used to create time-controlled (rather than parameter-controlled) zooms. Because larger values of x, y and z make objects farther away, an increasing value of z makes objects appear smaller. To use this component to zoom in, insert a 1D A/x filter before the third input to this function as shown in the example file "z Multiply Zoom".</p> <p>Left output = $x * z$ Right output = $y * z$</p>	<p>None</p>
 z Rotate	<p>z Rotate uses the third input to control the rotation of the space defined by the x and y inputs. When the z input has a center value of 0, nice twirl effects can be created when animating the system.</p>	<p>A: Amplitude B: Phase</p>

 <p>z Weighted Multiply</p>	<p>This component multiplies the x and y inputs by the z input which can create complex interactions of the inputs. The amplitude determines the z input's influence. There is no influence when it is 0. The A and B parameters are offsets added to z before multiplying the x and y inputs. As with z Multiply, this component can be used to scale the x and y inputs under the control of another component.</p> <p>Left output: $(C * (z + A) * x)$ Right output: $(C * (z + B) * y)$</p> <p>The name of this component was changed in version 2.5. In earlier versions, it was called z Multiply. The new z Multiply provides parameter-less multiplication.</p>	<p>A: z phase x B: z phase y C: Amplitude</p>
 <p>Min(x,z), Min(y,z)</p>	<p>A procedural component that is useful as a primitive for building your own systems and implementing your own algorithms. This component simply sets the left and right outputs to the minimum of (x and z) and (y and z) respectively.</p> <p>Example File: Fractal Polynomial</p>	
 <p>Max(x z), Max(y z)</p>	<p>A procedural component that generates a co-ordinate pair made up of the maximum of x and z (for the left output) and the maximum of y and z for the right output. It is sometimes difficult to predict the effect of applying this effect before trying it as the results are highly dependent on the preceding components. It is very useful for creating interesting angular juxtapositions.</p> <p>Example file: "MAX 3D Room".</p>	<p>A: Phase x B: Phase y C: Phase z</p>
 <p>Max x y z, Min x y z</p>	<p>This component takes uses the maximum of the three inputs for output 1 and uses the minimum of the three for output 2. Great for creating weird spatial distortions. Results vary greatly depending on the inputs. The result is often a complex mixing of the inputs.</p>	<p>None</p>
 <p>Z Sin distort</p>	<p>This component is the same as z Scale and Offset with the addition of a sin filter applied to the third input. This is a nice component for creating periodic modulations of space (when the x and y inputs are received from the plane or from another vector component.)</p>	<p>A: Frequency B: Phase C: Amplitude</p>

 <p>Z Facet Space</p>	<p>A z-controlled version of the 2-in/2-out Facet Space. The z input controls the frequency of the tiling. Note that each tile has its own co-ordinate system centered at the parent space's origin (0,0). The Amplitude parameter controls the size of the tiles. The Delta Size is a scaling/zoom factor for the space enclosed in each tile, and Frequency controls the spacing of the tiles.</p> <p>This component is useful both as a space distortion function and as a texture generator. As a space distortion function, it acts like a 2D vector function whose frequency is controlled by the third input. Like the 2D Orbiters component, this one creates copies of the incoming space. When it acts as a space distortion component, it will be fed by a 2D space and a 1D component.</p> <p>When this component precedes components such as the various 3D objects (especially finite-sized objects like spheres and cubes), the result is a multitude of randomly spaced objects. When the component is used after objects, it provides cell-like textures.</p> <p>Example file: "3D Falling LogoBalls". Notice there are three layers of balls and that there is only one real sphere in each layer. The multitude of balls is created by the facet space which precedes the Sphere components.</p>	<p>A: Amplitude B: Delta size C: Frequency</p>
 <p>Z Waves</p>	<p>Z Waves is similar to the 2D sin warp but with the z-input modulating the displacement amount. This component can create ripple-like undulations, especially when the phase is animated. If you follow this component with the Movie or pict component, you can get interesting water-like distortion.</p>	<p>A: Amplitude B: Phase</p>
 <p>z Random waves</p>	<p>A space distortion function that creates liquid-like wave distortions under control of the z input. The z-input modulates the amplitude of the waves. The Frequency parameter controls not only the spacing of the waves but also where they start. By adjusting the frequency, you can create effects (as shown in the example file) where only a portion of the incoming space is distorted by waves.</p> <p>Example file: "Reflection waves", "Random Waves Moonlight"</p>	<p>A: Frequency B: Amplitude C: Phase</p>
 <p>z Ripples</p>	<p>A version of the 2D Ripples component that allows the ripples to be controlled by the z input.</p> <p>The Frequency parameter controls not only the spacing of the waves but also where they start.</p>	<p>A: Frequency B: Amplitude C: Phase</p>
 <p>Perspective Sym(metrical)</p>	 <p>This component can be used in some cases to create perspective-type effects. When set up properly, this component creates two opposed planes which converge towards a horizon. To work as a perspective tool, you should use the Insert popup menu's Insert Perspective command as it will supply the other components needed to give the effect. The z-input provides depth/tilt of the viewed plane. The amplitude parameter controls the tilt amount.</p>	<p>A: Amplitude</p>

 <p>Perspective Clipped</p>	 <p>This component is similar to Perspective Sym except that it doesn't mirror space about the horizon. When points reach a maximum value, they are clipped to the same distance. With this effect, it is often advisable to use depth cueing to reduce noise which is likely to appear in the far distance.</p>  <p>TIP: You can use this component to create 3D globes by using Insert Perspective and substituting the distance component for the upper-right component. You may also want to insert the Distance Mirror component after the perspective component to reduce the background noise. See the PerspectiveClip Globe file in the Doc. Example Files folder which accompanies this documentation.</p>	<p>A: Amplitude</p>
 <p>Parallel Projection</p>	<p>When the inputs describe a 3D surface, this component creates a 2D projection/representation. The process is similar to the process by which a cube can be projected onto a 2D plane (such as a sheet of paper) and represented as a configuration of lines. The two parameters serve to rotate and tilt the plane onto which the surface is projected.</p>	<p>A: Angle xy B: Angle uz</p>
 <p>Spherical Projection</p>	 <p>This component creates a virtual sphere onto which the surrounding space is projected. It can be a little bit like looking into a mirrored sphere and seeing the surrounding room captured in the mirror ball. Whether the resulting system actually has the appearance of a spherical projection depends on the inputs. It is often best to precede this component with one of the perspective components. The picture at left was created with the file Spherical Projection which can be found in the Doc. Example Files folder which accompanies this documentation.</p> <p>Tip: Take a look at the example file and manipulate the parameters of the topmost components and note how they are reflected in the spherical portion of the image.</p>	<p>A: Amplitude</p>
 <p>Complex inversion</p>	<p>This component is the same as the 2D version but uses 3D distance (the square root of x-squared plus y-squared plus z-squared). The z-input has a skew-like effect.</p>	<p>A: Scale B: Amplitude</p>
 <p>Multi bubbles</p>	<p>This component distorts space with a bump texture, and the z input provides the scale of the bumps or bubbles. This is great for stucco-like textures.</p>	<p>A: Amplitude B: Phase C: Z Scale</p>
 <p>Multi inverse</p>	<p>This is a complex component that creates islands of complex inversions., each one of which is a complete universe.</p>	<p>A: Amplitude B: Frequency C: Phase</p>

 <p>Strange web</p>	<p>This component distorts space with a complex number equation that has three attractors.. The z- input along with the z displace parameter determine the strength of the distortion.</p>	<p>A: Real B: Imaginary C: z Displace</p>
 <p>Random Fractal Space</p>	<p>True 3D random texture generator. The output is truly random (rather than simply a distortion of the input values). Generally, this component is used to provide textures for 3D objects as it both generates a random texture and creates a 2D representation of the 3D input. This component is similar to the turbulence components but the noise is truly random.</p> <p>Tips: Use this function after a Room to see the rich 3D textures. Also, adjust the Max. Iterations for Fractals preference to control the noise's detail.</p> <p>Example file: "Fractal vector noise"</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Random 3D Noise</p>	<p>3D version of random noise.</p>	<p>A: Amplitude B: Frequency C: Phase</p>
 <p>Turbulence 3D</p>	<p>True 3D turbulent noise function. Use this as you would Random Fractal Space which is described above.</p>	<p>A: Amplitude B: Scale C: Phase</p>
 <p>z Wipe</p>	<div data-bbox="349 1150 560 1528" data-label="Diagram"> </div> <p>Create a wipe or mask effect controlled by the z input. The component will usually take a 2D space input for the left inputs. The z input will come from a surface/texture generator that provides the wipe's contour. Where the z input is above value set by the threshold (parameter A), z Wipe generates infinity (which is transparent to the RGB mixing functions). Where z is below the threshold, the inputs are passed through and possibly distorted. Distortion controls how much the z value will affect the scaling of the x and y inputs before passing them through.</p> <p>Example files: "Simple z wipe", "Circuitry z wipe" and "Crystal Noise z wipe"</p> <div data-bbox="341 1596 1010 1780" data-label="Image"> </div> <p>Two stages of the wipe performed by the "Simple z wipe" example whose structure is shown at the left.</p>	<p>A: Threshold B: Distortion C: Frequency</p>




 <p>Gated Distance</p>	<p>Gated Distance is a routing component that sends its output out either the left or right outlet depending on the distance of the incoming point from the origin (0, 0). Use this component to provide different shading for different regions of an image. The x and y inputs are treated as spatial co-ordinates. Parameter A provides the threshold which is used for determining the routing. Amplitude x and Amplitude y are multipliers applied to the left and right outlets respectively.</p> <p>In the discussion below, D refers to the distance of the point x,y from the origin. The algorithm is such that when D is above the threshold (parameter A), the incoming z value is sent to the right outlet (0 is sent to the left). When D is below the the threshold, D is sent out the left outlet.</p> <p><i>When $D < \text{Threshold}$</i> left outlet = B * D right outlet = 0 <i>else (when $D \geq \text{Threshold}$)</i> left outlet = 0 right outlet = C * z</p> <p>Example files: <i>Gated distance basic</i> is a simple example. Note that the left branch controls the shading of the central area (the points closest to the origin) and the rightmost branch controls the shading of the outer region. Experiment with the threshold parameter to see how it influences the image. <i>polynomial LS D b</i> is an advanced example where Gated Distance (used along with Conditional Z Set) allows the points of the fractal which approach infinity to be shaded differently than the others.</p>	<p>A: Threshold B: Amplitude x C: Amplitude y</p>
 <p>Open Compiled Tree</p>	<p>Choose this component to use a 3-in/2-out compiled tree. Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine.</p> <p>Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.</p>	<p>A: Blend B: Recursion</p>





3-in/3-out Components



The components in this group cover a wide range of applications: 3D space distortion and object creation, color/light manipulation, and mixing. An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone.




The inputs to the components are referred to as x, y, and z for the left, middle and right inputs. Keep in mind that they may or may not actually be spatial co-ordinates. While many of these components were designed with particular applications in mind, it is worth exploring non-typical applications.






Tip: Try using the space manipulation components (rotate, sphere, etc.) as color manipulation tools. For example, insert **Rotate** or **Sphere** or **3D Random Vector** components between **RGB/HLS** and **HLS/RGB** components to manipulate image color.






Function		Parameters
 3D Scale	<p>Scale the three inputs independently. This component can be used as a space distortion function, or as three independent parallel 1D scale filters.</p> $\mathbf{x_{out}} = A * x$ $\mathbf{y_{out}} = B * y$ $\mathbf{z_{out}} = C * z$	A: Scale x B: Scale y C: Scale z
 3D Offset	<p>Add an offset to the incoming values.</p> $\mathbf{x_{out}} = A + x$ $\mathbf{y_{out}} = B + y$ $\mathbf{z_{out}} = C + z$	A: Offset x B: Offset y C: Offset z
 3D Rotate	<p>Rotate a three dimensional space about the x/y, x/z, or y/z axes. This component was designed to take a 3D space as its input but can also have interesting applications for color manipulation.</p> <p>Note: Rotation does not behave as three parallel, independent 1D components. Each input influences each of the output values. As a result, if this component is the first one in the system, the output values of each output will vary over time even if the parameters are locked since the z input will be constantly changing. So, Scale should be placed before Rotate if you wish to control the influence of time.</p>	A: Angle xy B: Angle xz C: Angle yz






<p>Normalise</p>	<p>This is a mathematical primitive which normalizes the incoming vectors. The vector's magnitude is set to unity and maintains its orientation. The magnitude parameter is a scaling factor applied to the resulting vector magnitude.</p> <p>Formulas:</p> $x_{out} = x / \sqrt{x^2 + y^2 + z^2}$ $y_{out} = y / \sqrt{x^2 + y^2 + z^2}$ $z_{out} = z / \sqrt{x^2 + y^2 + z^2}$	<p>A: Magnitude</p>
<p>3D Normal</p>	<p>3D derivative-based component that provides lighting effects. It uses the derivative to calculate the orientation of the incoming space/object. 3D Normal generally needs to be followed by the Ax + By + Cz (the dot-product) 3-in/1-out component to yield a meaningful result. The parameters control the light direction. This component must actually scan the entire structure tree; when used in a compiled tree, it will not be able to access the parent's tree and so will yield different results than when used in the parent itself.</p> <p>Using this component as an RGB-shader creates the illusion that red, green and blue lights are illuminating the object.</p> <p>Technically, this component returns the normalized vectors of the input. The normal is the vector perpendicular to the surface's tangent plane.</p> <p>Example Files: <i>Braque & Picasso & ArtMatic</i></p>	<p>A: Magnitude B: Distribution</p>
<p> 3D sphere A</p>	<p>Create a pseudo-3D sphere. This 3D object behaves a bit differently than the other 3D objects. It is not quite a true 3D sphere and infinity is not used as the value for the regions outside of the sphere as a result. The sphere appears against the same background that makes up the texture of its surface.</p>	<p>A: Angle xz B: Amplitude y C: Displace z</p>
<p> 3D sphere B</p>	<p>Create a true 3D sphere. The region outside of the sphere is given the value infinity (which is treated as transparent by many if the mixing components). To rotate the sphere, place a 3D rotation component after the sphere. To move the sphere in an orbit, place a rotation before the sphere.</p>	<p>A: Offset x B: Offset y C: Offset z</p>
<p> 3D plane</p>	<p>Create a 3D plane which can be banked and tilted. Like all 3D object components, the area outside the object are mapped to infinity (which is painted with the first auxiliary color -- the depth cueing color.) The components which follow the plane component provide the surface's texture. Banking x controls the horizontal tilt, Plane offset provides an altitude offset, and z Slope controls the forward/back tilt.</p>	<p>A: Banking x B: Plane Offset C: z Slope</p>
<p> 3D Tube</p>	<p>Create a 3D tunnel whose radius and offset can be modified.</p>	<p>A: Offset x B: Offset y C: Radius</p>

<p>3D elevation slices</p>	<p>This is an advanced component which makes certain kinds of 3D modelling possible. This component can be used to visualize 2D surfaces or terrains by iteratively drawing slices over the z-axis. Since true 3D modelling is beyond ArtMatic's current capabilities, this component was added to simulate 3D rendering of some systems. The third input must be connected to the 2-in/1-out function to be evaluated or 2-in/1-out compiled tree. The first two inputs determine the points to be evaluated.</p> <p>Scale acts as a magnification control. Slope controls angle of view of the 3D space. Offset controls the spacing of the slices on the z axis.</p> <p>Examples: Look for the word 'elevation' in the examples library for examples of this component.</p>	<p>A: Scale B: Slope C: Offset z</p>
<p>3D density slices</p>	<p>Another component for simulating 3D modelling. This component works by taking an input function and sampling it iteratively over the z-axis. It is a primitive sort of volumetric rendering. The slices are limited to a plane bounded by -Pi and +Pi on the z-axis.</p> <p>The third input must be connected to a 3-in/1-out component which will be the function to be evaluated iteratively.</p> <p>Examples: Look for the word 'density' in the examples library for examples of this component.</p>	
 <p>3D Random vector</p>	<p>3D space distortion function. Place this after a 3D object to provide surface texture. If you place a noise component before a 3D object, you will warp the space in which the object exists (and thus warp the object). This is a distortion-type noise whose output is made up of randomized modulations of the input values.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>3D Fractal Vector</p>	<p>Fractal version of the 3D Random vector component. Like the other fractal noise components, there are many frequencies of noise and the complexity/detail can be adjusted with the Max. Iterations for Fractals preference. This function is a great primitive for marble or clouds or any complicated non-repeating texture.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
<p>Memory Functions</p>	<p>The next several components (all with Memory in their names) are intended for use in (and are only meaningful in) iterative systems which are discussed in detail in the chapter Compiled Trees and Iteration.</p>	
<p>Memory Max</p>	<p>This component stores the maximum of the current iteration and the stored values with the fourth input scaling the incoming values.</p> <p>See the chapter Compiled Trees and Iteration for more information about iterative systems and memory components.</p>	

 Memory Min	Memory Min is something of a complement of Memory Max. Rather than retaining the maximum values, it retains the minimum values. This is useful for accumulating dark graphics against bright backgrounds in iterative systems.	
Memory Add	This component adds the value of each iteration to the value accumulated in the previous iteration with the fourth input scaling the incoming values. See the chapter Compiled Trees and Iteration for more information about iterative systems and memory components.	
Memory Depth Sort	Memory Depth Sort acts as a memory-based version of the other depth sort functions. Rather than performing a depth sort between two sets of inputs, the depth sort is performed between the current set of values and the stored values from the previous iteration. Depth Sort preserves the values that correspond to the frontmost pixel when the three values are considered as 3D position. The frontmost pixel corresponds to the value with the lowest valued z (third) co-ordinate. This component is useful for accumulating 3D objects in recursive and iterative trees.	
 Conditional z set	This is an advanced component which can be used to create 'levelset'-type shading when implementing classical-style fractals such as the Mandelbrot or Julia sets. When the distance (of the x, y inputs from the origin (0, 0)) is above the Threshold parameter or below the Minimum parameter the incoming z value is passed to the third outlet otherwise 0 is sent out the third outlet. x and y are passed through unchanged to the left and middle outlets. Hence, when x,y is within a specific range, the third outlet is non-zero and can be used for shading. Generally, the z-input will be connected to the output of the Iteration component in a system with memory components. In such a system, it is possible to color a point whose value is infinity with a color related to the number of iterations required to reach infinity. This component can also be used to create distance masking as in the <i>Clock</i> example file. Example files: <i>Fractal polynomial</i> demonstrates the levelset shading technique described above. <i>Clock</i> and <i>Clock Face</i> demonstrate distance masking.	A: Threshold B: Minimum
 3D Sines	3D math primitive which generates $\sin(x)$, $\sin(y)$ and $\sin(z)$. This component is equivalent to three parallel 1D sine filters.	A: Amplitude B: Phase C: Frequency

 <p>3D Color fractal noise</p>	<p>This component is the RGB equivalent of the 2-in/3-out Random Fractal noise function. The Amplitude augments the contrast of the R G & B channels. This component yields interesting effects when combined with other color texture functions. Color fractal noise can also be used as a space displacement function.</p> <p>Like other fractal noise, there are many frequencies of noise and the complexity/detail can be adjusted with the Max. Iterations for Fractals preference.</p> <p>Example files: "Solid Color Room". Notice that in this example, the noise is modulated with 3x3 matrix multiplication which is found in the compiled tree at the bottom of the system.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>3D Color techno noise</p>	<p>An RGB version of the 3-in/1-out 3D techno noise.</p> <p>Example files: "3D Color Techno River" and "Triple pict projection".</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>3D color spots</p>	<p>An RGB version of the 3-in/1-out 3D Spots noise function. The Chroma parameter interpolates between color and grayscale output.</p>	<p>A: Chroma B: Phase C: Frequency</p>
 <p>3D Color Strata stone</p>	<p>RGB version of the 3-in/1-out 3D stratified noise. The Chroma parameter interpolates between color and grayscale output.</p>	<p>A: Chroma B: Phase C: Frequency</p>
<p>3D Color Bubbles</p>	<p>RGB shading function which provides randomly shaped and colored bubbles.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
<p>3D Color Marble</p>	<p>3D color shading function that creates a veined, marble-like texture. Hue Rotation controls the color range of the resulting texture while Saturation controls the color saturation, and Frequency controls the spacing of the veins.</p>	<p>A: Hue Rotation B: Saturation C: Frequency</p>
<p>3D Color Perlin Noise</p>	<p>3D version of the perlin noise algorithm.</p>	<p>A: Hue Rotation B: Saturation C: Frequency</p>
 <p>RGB Gamma</p>	<p>Color manipulation function that provides independent control of the gamma (color response curve) of the red, green and blue channels.</p>	<p>A: Gamma Red B: Gamma Green C: Gamma Blue</p>

 <p>HLS to RGB</p>	<p>Colorspace function that converts an HLS-based color stream back to RGB. This component is almost always used at some point following an RGB to HLS component. By inserting components between this component and the HLS to RGB component, you can manipulate the hue, luminance and saturation of an image which make many interesting color manipulations possible.</p> <p>Example files: Search for HLS to find the many examples of HLS manipulation.</p>	<p>A: Hue Contrast B: Luminance Contrast C: Saturation Contrast</p>
 <p>RGB to HLS</p>	<p>Colorspace function that converts RGB-based color to HLS (hue, luminance, saturation). By inserting components between this component and the HLS to RGB component, you can manipulate the hue, luminance and saturation of an image which make many interesting color manipulations possible.</p> <p>Example files: Search for HLS to find the many examples of HLS manipulation.</p>	<p>A: Amplitude</p>
 <p>Colorize</p>	<p>A color filter function that lets you desaturate and tint an incoming RGB stream. With this component, you can simulate the look of “colorized” images and sepia tones, and it provides a useful control for the colors created by the various RGB texture primitives.</p> <p>When the Amount parameter is 0, there is no effect. When the slider is all the way to the right, the image is a tinted monochrome image whose tint is determined by parameters B & C. Parameters B and C (red balance and blue balance) determine the red and blue value for the colorization tint. When Amount is at the maximum and parameters B & C are 0, the resulting tint is green.</p>	<p>A: Amount B: Red Balance C: Blue Balance</p>
<p>Packed Mixer Family</p>	<p>The group of components whose names include 'packed' is used for mixing packed streams--usually from RGB components or branches. When used in iterative systems, these components have memory and in each iteration mix the stored value with the current iteration's value and then store the resulting value.</p> <p>See the chapter Compiled Trees and Iteration for more information about iterative systems and memory components.</p> <p>When infinity is encountered in one of the inputs, it is treated as transparent by these mixing functions.</p> <p>These functions can accept both packed and unpacked inputs. The first input acts as the background in iterative systems and is mixed with the final result.</p>	
 <p>Packed RGB crossfade</p>	<p>Crossfade between three RGB images. See the description of the 2-in/3-out version for a more detailed description. The parameters determine the contribution that the inputs make to the final output.</p>	<p>A: Interpolate AB B: Interpolate C</p>
 <p>Packed RGB alpha</p>	<p>Mix two RGB-based images while applying an alpha (transparency) mask provided by the third input. The third input will usually be a simple value (non-packed stream) and acts as an alpha channel for masking the inputs. If the third input is a packed stream then the effect is a complex mixing of the first two images treating the R, G and B layers of the third image as masks to apply to the red, green and blue layers separately.</p>	<p>A: A Threshold B: B Feather</p>




 <p>Packed RGB add</p>	<p>Perform an additive mix of the three inputs.</p>	<p>A: Level in1 B: Level in2 C: Level in3</p>
 <p>Packed RGB max</p>	<p>Mix three images by comparing the corresponding pixels of each image and picking the brightest pixel of the three at each pixel position. Interesting color effects are possible. Try using a picture of sky and clouds and watch the clouds jump to the foreground in front of the other images.</p>	<p>A: Level in1 B: Level in2 C: Level in3</p>
 <p>Packed Add & Mul</p>	<p>Perform an additive mix of inputs a and b and filter the result with input c. Parameters A and B control the mix of inputs a and b. Parameter C determines how much filtering input c provides.</p> <p>Example file: <i>Golden spinal fossil</i> is an example in which the c-input is used for specular highlights, the b-input provides diffuse illumination, and the a-input is the basic surface whose altitude determines the color.</p>	<p>A: Level A B: Level B C: Multiply C</p>
 <p>Packed depth sort</p>	<p>This component mixes three 3D objects into a single set of 3D outputs. The inputs should be the packed outputs of 3D objects as shown in the structure to the left (which is a simple modification of the 3D Sphere and Plane item in the Structures popup menu). See the documentation which covers the 2-in/3-out version for more information.</p> <p>NOTE: This component should always receive its input from Pack components (that receive their input from 3D object components). The algorithm is such that for each set of points received, the set with the lowest value of z wins; in other words, the co-ordinates closest to the viewer win.</p>	<p>A: z offset</p>
 <p>Open Compiled Tree</p>	<p>Choose this component to use a 3-in/3-out compiled tree. Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine.</p> <p>Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.</p>	<p>A: Scale B: Recursion</p>
















4D Vector Functions (4 in - 2 out)







These components provide a broad number of uses. An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone. In the descriptions below, **x** and **y** refer to the two left inputs and **x2** and **y2** refer to the rightmost inputs.




Complex number primitives. Many of these components are mathematical primitives for performing complex number arithmetic. The complex number inputs are called **Z** and **C**; **Z** refers to the two leftmost inputs, **x** and **y**, where **x** is the real part and **y** is the imaginary part of **Z**. Likewise, **C** is made up of the two rightmost inputs.

Function	Notes	Parameters
 Add Z + C	<p>Add mixes two 2D spaces by adding the respective x and y co-ordinates. Changes to the incoming space can sometimes have compound effects. For example, if two rotation components feed Add, changing the rotation parameter of either input will also cause changes to the scale of the picture.</p> <p>Tip: Check out the result of using this component with rotation and complex map as inputs.</p> <p>Left output: $A(x+x2)$ Right output: $A(y + y2)$</p>	A: Scale
 Complex multiply Z*C	<p>This component performs complex number multiplication by treating the inputs as two sets of complex numbers. The effect is quite dramatic and is highly dependent on the input functions. This component provides symmetry and creates interesting spatial deformation.</p> <p>The x and y inputs are treated as the real and imaginary parts of a complex number and the x2 and y2 inputs as parts of another such that:</p> $Z = x + iy$ $C = x2 + iy2$ <p>This component multiplies $Z * C$. The real part of the product is sent out the left outlet and the imaginary part is sent out the right.</p> <p>Left output: $(x * x2) - (y*y2)$ Right output: $(x * y2) + (y * x2)$</p>	None
 Complex divide Z/C	<p>Perform complex number division. Treats the inputs as a pair of complex numbers. The output is the quotient. As with all division if the inputs are identical, the output is a single color (i.e. a plane), but if you offset the values then you get interesting deformations.</p> <p>Left output is the real part of Z/C.</p> <p>Right output is the imaginary part of Z/C.</p>	None

 <p>Complex Z² + C</p>	<p>The Mandelbrot/Julia Set equation implemented as a space mixing/distortion component. This component will usually be used near the top of the tree and takes two spaces as input, treating them as complex numbers Z and C where the first input of each pair is the real part of the number and the second input of each part the imaginary part.</p> <p>The output of the component is a space (a quite complex one).</p> <p>Example files: "Yellow Pulsing juliaSet". The colorful and decorative "Fractalcandy" shows a modified Mandelbrot/Julia Set shaded by compiled tree that provides complex RGB shading.</p>	<p>A: Recursion B: Outer Slope</p>
 <p>Complex Z³ + C</p>	<p>Another Mandelbrot set relative. This is the cubed version of the Mandelbrot equation. Because this component is computationally intensive, you may want to lower the value of the Max Iterations for Fractals preference while exploring (and increase the value, if necessary, for final rendering).</p> <p>Example file: "Venitian Fractale SinDC". This example makes use of both the squared and cubed Mandelbrot relatives. The file also demonstrates how compiled trees can be used to implement complex shaders.</p>	<p>A: Recursion B: Outer Slope</p>
 <p>Complex C(Z + 1/z)</p>	<p>Complex number primitive and space distortion function. Explore it to discover fascinating fractal textures. This can also be used as a building block in compiled trees for creating new fractals.</p> <p>Equation = C * (Z + 1/Z)</p>	<p>A: Recursion</p>
 <p>Complex exp C e^Z</p>	<p>Complex number exponential function. This component is quite CPU-intensive and is a valuable building block for creating new fractals.</p>	<p>A: Recursion</p>
 <p>Complex C sin Z</p>	<p>Another building block for creating new fractals.</p> <p>Example files: "Complex C sin z" and "Chinese Fractal Sin z Plate"</p>	<p>A: Recursion B: Outer Slope</p>
 <p>Distance Divide</p>	<p>This component divides the first space by the distance of the second pair of inputs from the origin. It can have perspective-like effects. Note that because division is used, the component is not symmetrical so that the results change if the order of the inputs changes. Phase adds an offset to the distance value. Amount is an interpolation between the distance and 1. When the amount is zero, the distance is ignored.</p>	<p>A: Amount B: Phase</p>

 <p>Exp field</p>	<p>This component creates a composite space that has aspects of both input spaces. A field equation is applied to both pairs of inputs and the results are added together. The field equation is similar to a gaussian curve and hence the maximum output is found where the input value is 0 and approaches zero as the input reaches infinity.</p>	<p>A: Amplitude B: Warp</p>
 <p>ComplexMap 4D</p>	<p>Like the 2D Complex map, this component is a complex number polynomial. This version is recursive and is sensitive to the preference for Max iterations for fractals. This component creates particularly rich output.</p>	<p>A: Real B: Imaginary</p>
 <p>Esher disks</p>	<p>A complex space distortion function which tiles space with disks using the Mandelbrot set equation. The C input (the second input pair) act as the C constant in the Mandelbrot equation.</p>	<p>A: Offset x B: Offset y C: Radius</p>
 <p>Real multiply</p>	<p>This component multiplies the first space by the second using real number multiplication. This is yet another way produce a composite space that has aspects of the input spaces.</p> <p>Left output = $x * x2$ Right output = $y * y2$</p>	<p>A: Amplitude</p>
 <p>Add Normalized</p>	<p>Another useful mathematical primitive that is often used for implementing systems based on differential equations.</p> <p>The component's formula is:</p> $Z + \frac{C}{N}$ <p>where N is the magnitude of C (square root of $x^2 + y^2$).</p> <p>Example file: "Differential System".</p>	<p>A: Scale</p>
 <p>Interpolate</p>	<p>The left output is an interpolation of x and x2, and the right output is an interpolation of y and y2. This component offers a way to animate the transitions from one transform to another. The Interpolate parameter set to its minimum value generates the left-hand space and at the maximum value generates the right-hand space.</p>	<p>A: Interpolate</p>
 <p>Radial Interpolate</p>	<p>This component generates an interpolation from space A to space B where the distance from the origin determines the weighting towards space A or space B.</p> <p>Compatibility Note. ArtMatic Pro 2.5 uses undistorted space for the interpolation (unlike earlier versions). As a result, files created in earlier versions will appear different in 2.5 than in 2.0.</p>	<p>A: Amplitude</p>

 <p>Facet Mix</p>	<p>This component generates a composite space by randomly choosing facets/bubbles from one space or the other so that the resulting space has a mix of bubbles from both spaces. In this way, chunks of space B become mixed with space A. Amplitude is a zoom factor applied to the inputs before feeding them into the bubble algorithm. Phase is an offset factor. Frequency determines the "bubble" size.</p> <p>Renamed in ArtMatic Pro 2.5. This component was previously called Bubbles Mix.</p>	<p>A: Amplitude B: Phase C: Frequency</p>
 <p>Min < Max</p>	<p>This is a true minmax function that blends between the maximum or the minimum of the two spaces' co-ordinates.</p>	<p>A: Amplitude B: Min to Max</p>
 <p>Blended max</p>	<p>This component generates a weighted maximum. The actual math is a bit complicated as the implementation guarantees continuous (smooth) output. The resulting space will have some aspects of both input spaces. The balance parameter controls whether the function gives the precise maximum or a weighted value that favors either the left or the right input pair.</p> <p>Essentially, the component acts so that the left output=Max(x,x2) and the right output=Max(y,y2) though not strictly. With the balance set to 1, the input always wins. The effect of changes to the balance parameter can be quite surprising.</p> <p>This component creates interesting juxtapositions of space.</p>	<p>A: Balance</p>
 <p>Abs & minimum</p>	<p>This component provides an interesting way to combine spaces. The output is the absolute value of the minimum of x and x2 and y and y2. Amplitude is a multiplier applied to the output. Rotate B rotates the second space before calculating the minimum.</p>	<p>A: Amplitude B: Rotate B</p>
 <p>Y Compare</p>	<p>This is a procedural component that chooses the input pair Z or C whose y component has the highest value. This creates a composite space that is an interesting mix of the two inputs. The offset parameter is added to the two output values.</p> <p>Example file: "Y compare Space".</p>	<p>A: Offset</p>
 <p>Split Space</p>	<p>Create a composite space whose left half comes from the left inputs and whose right half comes from the right inputs. The split is made using Space A's co-ordinates. So, the split might not be a straight line. The offset moves the split point left or right. Advanced users will find this useful when building fractals.</p> <p>Example file: "Double Rotation fractal beast" makes use of this function within the compiled tree.</p>	<p>A: Offset</p>





 <p>Distance compare</p>	<p>Create a composite space by picking the point from Space A or Space B that is closest to its own origin. This is another primitive that can create fascinating composite spatial systems.</p> <p>Example file: "DC Double Affinity feedback" uses this component to create a fractal. "Space Compare" uses this component to mix a Radial Star space with a simple space.</p>	<p>None</p>
 <p>x from Z, y from C</p>	<p>Create a new space by taking the x co-ordinate from the Z input and the y co-ordinate from the C input. This is one of those components where the results are a bit unpredictable but often rewarding.</p> <p>Example file: "XfromZ Space"</p>	<p>None</p>
 <p>Open Compiled Tree</p>	<p>Choose this component to use a 4-in/2-out compiled tree. Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine.</p> <p>Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.</p>	<p>A: Blend B: Recursion</p>

4-in/3-out Components


These components are all mixers. Many of them were designed to mix RGB-based images with gradient-based images--such components treat the leftmost three inputs as an RGB image and the rightmost input as a gradient-based image (shaded with the active gradient). Some of 4-in/3-out components are essentially 3-in/3-out components with the fourth input acting as a controller for the component--these are especially useful in iterative systems since the output of the **i** component can be used to offset, scale or rotate the iterations.

An essential in-depth discussion of function categories is found in the [Getting Deeper](#) chapter and is recommended reading for everyone.


Input notation convention: By convention, the inputs are sometimes referred to as x,y,z and w where x is the leftmost input and w the rightmost. x,y and z does not refer to spatial co-ordinates except where explicitly stated.


Function	Notes	Parameters
RGB/Gradient Mixing Components	This group of components was primarily designed for mixing the output of RGB and gradient-based components and branches. The leftmost inputs should be an RGB 'stream', and the rightmost input is interpreted using the active gradient. The output is RGB. Choose RGB and ArtMatic shading from the Structures popup menu to create a structure set up to mix RGB and ArtMatic gradient-based shading. The active gradient is applied to the right input before the images are mixed and shading/depth cueing are applied to the combined image.	
 RGB mix	Mix an RGB branch of the tree with a gradient-based branch by adding the values of the corresponding pixels of the two input images. Parameters A and B control the relative level of the left and right images.	A: Level in1 B: Level in2
 RGB multiply	Mix an RGB branch of the tree with a gradient-based branch by multiplying the values of the corresponding pixels of the two input images. In essence, the RGB image is filtered by the ArtMatic image. The result is 0 (black) where either image is black (i.e. multiplying by 0 yields 0). Parameter A controls the strength of the filtering and B is a luminance (brightness) offset which is useful since multiplication can sometimes yield dark images.	A: Amount B: Offset
 RGB alpha	Mix the two input images and treat the second image as an Alpha mask. Only the pixels whose value is above the threshold are displayed. The feather parameter determines the number of pixels beyond the mask's boundaries which are displayed. When the threshold is at the maximum only image one is visible. When the threshold is at the minimum value, only image two is visible. This component is great for creating dramatic dissolve effects for movies and video. Tip: If the second image has cloud or star textures, you can effectively superimpose them on the first image. Example file: "Simple alpha dissolve"	A: Threshold B: Feather
 RGB Interpolate	Interpolate between the two images. When the slider is at the extreme left or right only the corresponding image will be seen except where the image has infinity . Infinity is transparent. This is essentially the same as the 2-in/3-out Packed RGB Crossfade .	A: Interpolate.

3-in + control	Each of the components in this group is equivalent to a simple 3-in/3-out component but provides a fourth input that acts as a control. For example, w Offset is equivalent to 3D Offset with the fourth (w) input controlling the offset amount. These components are often used in iterative system with the w-input being fed by the iteration component.	
w Offset	<p>Add an offset to the x,y and z inputs. The w-input acts as an offset multiplier/displacement vector.</p> <p>Formulas:</p> <p>left output = $x + (A * w)$ middle output = $y + (B * w)$ right output = $z + (C * w)$</p>	A: Offset x B: Offset y C: Offset z
w Scale	<p>Scale the x, y, and z inputs with w acting as a scale multiplier.</p> <p>Formulas:</p> <p>left output = $x * (1 - (A * w))$ middle output = $y * (1 - (A * w))$ right output = $z * (1 - (A * w))$</p> <p>When A=0 no scaling occurs.</p>	A: Amount B: Floor
w Rotate	Rotate the x,y,z inputs (treating them as spatial co-ordinates) with w acting as the rotation control. The parameters determine the angle of rotation which is multiplied by w.	A: Angle xy B: Angle xz C: Angle yz
Memory Components	These components are intended for use in (and are only meaningful in) iterative systems which are discussed in detail in the chapter Compiled Trees and Iteration .	
Memory Add	<p>This component adds the value of each iteration to the value accumulated in the previous iteration with the fourth input scaling the incoming values.</p> <p>See the chapter Compiled Trees and Iteration for more information about iterative systems and memory components.</p>	
Memory Max	<p>This component stores the maximum of the current iteration and the stored values with the fourth input scaling the incoming values.</p> <p>See the chapter Compiled Trees and Iteration for more information about iterative systems and memory components.</p>	

<p>Memory w Alpha</p>	<p>Memory Alpha mixes the current RGB values with the stored RGB values by interpreting the fourth input (w) as the transparency of the current values. A high 'w' value makes the current value (pixel) opaque. A 'w' value less than 0 makes the current value (pixel) completely transparent (and, hence, invisible). See the example file MemAlpha Clouds for an example of this powerful component. We recommend rendering the animation to get a sense of the incredible effects you can achieve with this component. One can simulate volumetric-3D rendering with this component.</p> <p>The parameters are the starting RGB color values. This component is great for simulating 3D volumetric rendering.</p> <p>Example file: Don't miss the <i>MemAlpha clouds</i> example. Render the movie, it is amazing!</p>	
<p>Memory w Min</p>	<p>Memory w Min compares the current w value with the stored w value and stores whichever set of values (R,G, B, and w) has the lowest value for w. This component is very much like Memory Depth Sort but uses 'w' (which is not passed out of the function) rather than the third input for the comparison.</p> <p>Example file: See <i>Mwmin perlin noise</i> for one example of this component.</p>	
<p>Memory w Max</p>	<p>Memory w Max compares the current w value with the stored w value and stores whichever set of values (R,G, B, and w) has the largest value for w. Note that only the 'w' value is used for the comparison unlike the simple Memory Max component which uses all of the input values for the comparison.</p>	
<p>Packed Mixing Components</p>	<p>This group of components combines 'packed' mixing functions with other common operations. They allow more compact trees which render more efficiently than trees which use multiple components to provide the same functionality. Most of these provide the same functionality as 3 or more primitive components.</p> <p>Note that in iterative systems these components have memory and behave like more complex memory functions. In iterative systems, the first input acts as a background image and its value is not changed by the iterations.</p> <p>Each of the three first inputs can be either packed or unpacked. The fourth input must be a simple unpacked stream.</p> <p>Notation: a, b, and c refer to the first, second and third inputs. These are used instead of x, y, z because the inputs are packed streams rather than simple values.</p>	
 <p>Packed blend * w & blend</p>	<p>This component is a procedural shader in which input A (the first input) is mixed with input B (the second input). The result is multiplied times the w (fourth) input and mixed with the C (third) input.</p> <p>Formula (pseudo-code): $\text{blend}(\text{blend}(A,B) * w, C)$</p> <p>Example file(s): <i>Udescribable C</i></p>	<p>A: Interpolate B B: Interpolate C C: Multiply w</p>

<p>Packed Add & Blend *w</p>	<p>This component is equivalent to a packed mix (using addition) of inputs a and b with the result being crossfaded with the c input. The resulting output is then scaled with w acting as a scaling factor.</p> <p>Parameter A controls how much of stream b is added to the base image. Parameter B controls the crossfading with stream c. Parameter C determines the influence of input w.</p> <p>Tip: w will usually be connected to a component that provides global illumination.</p> <p>Formula: $((a + A*b) \text{ crossfaded with } c) * w$</p>	<p>A: Add B B: Interpolate C C: Multiply w</p>
<p>Packed Add & Mul +w</p>	<p>Add inputs a and b. Multiply the result with input c and add w to the result</p> <p>Parameter A controls how much b is added to a. Parameter B controls how much c will multiply the result. Parameter C controls how much w is added at the end.</p> <p>w can be used to provide the shading and can created specular highlights.</p> <p>Formula: $((a + A*b) * c) + w$</p>	<p>A: Add B B: Mul C C: Add w</p>
<p>Packed Mul & Add *w</p>	<p>Multiply inputs a and b and add input c. Multiply the result by w.</p> <p>Parameter A controls the influence if input b. Parameter B controls the influence of input c. Parameter C controls how much w will multiply the result.</p> <p>w will usually be connected to the component that provides the global illumination. This component is used in the <i>Gated bilateral B metal ground</i> compiled tree which acts as a custom shader to provide a realistic illumination of the "beast".</p> <p>Formula: $((a * b) + c) * w$</p>	<p>A: Mul b B: Add c C: Mul w</p>
<p>Packed Add & Add *w</p>	<p>Add inputs a, b and c and multiply the result by w.</p> <p>Parameter A controls how much b is added to a. Parameter B controls how much c is added. Parameter C controls how much w will multiply the result.</p> <p>w will usually be connected to the component that provides the global illumination.</p> <p>Example file: "Liquid invasion"</p> <p>Formula: $(a + b + c) * w$</p>	<p>A: Add B B: Add C C: Mul w</p>

<p>Mixing components</p>	<p>These are additional mixing components that accept packed inputs (some of these require scalar - single value -- inputs for some inputs. All of these behave like memory components when used in iterative systems. When used in iterative systems, the first input acts as the background and does not iterate.</p>	
<p>Packed Depth Sort</p>	<p>Mix four 3D objects into a single 3D stream. This component is just like the other Packed Depth Sort components (2-in/3-out, 3-in/3-out) and requires that each input be a packed stream. Parameter A acts as an offset in the third (z) dimension. Each input should be a 3D object whose output has been packed.</p>	<p>A: z offset</p>
<p>Packed RGB w Min</p>	<p>This component is the RGB equivalent of the 3D Depth Sort components but uses the third and fourth inputs for the comparison that determines whether . The first two inputs must be packed i\\RGB streams. The third and fourth inputs must be single value streams. This component treats the third input as a fourth value for the packed data in input a, and the fourth input is treated as a fourth value for the packed data in input b.</p> <p>Either input a or input b's values are passed through the component depending on which stream's fourth value (either the third or fourth input) is smaller. When input c < input d then input a is passed through. When input d is the smallest value, input b is passed through.</p> <p>Example files: <i>Landscape 1 sky</i></p>	
 <p>Packed rgb w Min blend</p>	<p>This component is a variation of Packed rgb w Min except that the frontmost object (the component's second input) is transparent and fades into the background (the first input). This component is useful for 3D fades using the 3D parametric face component.</p> <p>Example file(s): Triple bascules blend, Tech flower room D, Tech Triple pict Pan B</p>	<p>A: Transparency (of input B)</p>
<p>Packed Alpha *w</p>	<p>Mix the a and b inputs (usually RGB inputs) and treat input c as an alpha mask. The w value is multiplied by the final output and will usually be connected to the branch that provides global illumination. Inputs c and w should both be scalar (non-packed) values.</p> <p>Parameter A determines the threshold over which masking occurs. Parameter B provides feathering at the mask border and determines the definition of the mask borders. Parameter C determines how much the w input influences the result.</p>	<p>A: Threshold B: Feather C: Multiply</p>
<p>Packed a*w alpha b</p>	<p>Mix the a and b inputs (usually RGB inputs) and treat input c as an alpha mask. The w</p> <p>Parameter A determines the threshold over which masking occurs. Parameter B provides feathering at the mask border and determines the definition of the mask borders. Parameter C determines how much the w input influences the result.</p> <p>w multiplies the source prior to the masking and makes shadow effects possible. Inputs c and w should both be scalar (non-packed) values.</p> <p>Example files: "Flying Shadows" contains a small iterative loop which feeds "Packed a*w alpha b". "Butterfly Shadows" uses the same technique to create a mathematical beast.</p>	<p>A: Threshold B: Feather C: Multiply</p>

Packed (a+w) Alpha b	Mix the a and b inputs (usually RGB inputs) and treat input c as an alpha mask. The w value is added to source a before masking. Example files: This component is used for the halo effect in "Flying Shadows Halo". "Electro Rain MemAlpha" illustrates the fact that the packed mixing components behave like the 'mem add' components for all layers except for the first. See also "Packed MemAlpha clouds".	A: Threshold B: Feather C: Add w
Packed Blend & Alpha	Perform a mix of inputs a and b and mix the result with input c while treating input w as an alpha mask. Inputs a, b and c should all be packed RGB values. Input w should be a scalar (single value) stream. Parameter A controls the mix of images a and b. Parameters B and C control the mask threshold and feathering.	A: Blend AB B: Threshold C: Feather
 Open Compiled Tree	Compiled trees are groups of tiles that can be used in place of single tiles as a kind of macro or subroutine.. Compiled Trees are covered in more detail in the Compiled Trees chapter of this manual. Compiled Trees and their parameters are covered in detail in the Compiled Trees chapter of this manual.	A: Scale B: Iterations

QuickTime Notes & FAQ

This chapter briefly provides some general information about QuickTime movies and some tips that you may find useful when rendering your animations to disk. If you have limited QuickTime experience, we recommend that you read this chapter since your choice of frame rate, frame dimensions and choice of compression (codec) are critical. Most of the information here is not specific to our applications and applies to most QuickTime movies. The chapter concludes with a FAQ (Frequently-Asked-Questions).

QuickTime Overview & Issues

QuickTime is an extremely flexible multimedia recording and playback architecture that can be used to create and display content appropriate for a wide range of applications from Internet delivery to Hollywood movies. Video and animation (referred to hereafter as video) are relatively demanding for a computer to play back (requiring much more of a computer's resources than sound playback and recording, for instance). Uncompressed video is also storage intensive since each frame is a complete picture. A raw 640 by 480 video frame or picture requires about 1/2 megabyte which amounts to a little over 13 Megabytes per **second** of full motion video. Fortunately, QuickTime provides a number of mechanisms for managing the storage and processor demands of QuickTime content.

QuickTime compression provides both lossless and lossy methods for compressing video and audio content in order to reduce the storage requirements. The choice of compressor (or codec) determines the amount of storage savings and influences the playback requirements. The frame rate and frame size influence both the storage and processor demands.

A common mistake made by QuickTime novices is to render movies that their computers have difficulty playing back. Determining the appropriate settings for your application is critical for success.

QuickTime Settings

ArtMatic is capable of creating professional-quality animation and QuickTime movies, but not all computers can play animation with those settings back. Before rendering an animation, you need to ask yourself the following questions:

- Is it going to be further edited or added to an existing video project?
- What will the playback mechanism be? (A computer, a DVD player, a CD-ROM?)
- Is it intended for web playback, computer playback, DV or DVD, or for use with VJ software such as [Videodelic](#)?
- If it is intended for computer playback, what range of processor speeds is it intended for?
- Do you have a program such as MediaCleaner or Cleaner available?

If the video is going to be edited or added to an existing video project, you will need to choose settings compatible with your video editor or project. In general, you will need to choose an archival-quality codec (compressor). Such a codec (Animation at its highest quality settings, for example) compresses the content in such a way that it can be passed through another codec after editing without degrading the signal. If you are going to be using a DV editing program, you will probably choose a DV codec and the frame rate and dimensions required by your project.

If the movies are intended for playback on a wide range of computers, you will need to choose a frame rate, frame dimensions, and codec that will allow smooth playback on the slowest machine that you plan to support.

If you are going to use the movies as content for a real-time video synthesizer, such as our [Videodelic](#) application, you will need to choose settings compatible with that application. For Videodelic, the very best results are with half-size frames (full-size may work well if you have a very fast computer), a high frame rate (since VDL doesn't play back every single frame, you can use frame rates that higher than you would if you were planning on playing the movie back with the QuickTime Player), and a compressor such as Animation or Motion JPEG A which doesn't strain the CPU.

There are no hard and fast rules for determining the best settings are. A little bit of experimentation goes a long way. For some situations, the choice is easy. If you are creating content that is going to be used with a video editor, you will use the same frame rate and dimensions and codec as your other content or the settings required by your video editing program.

Frame Rate, Frame Dimension and Codec

These are the settings that influence the storage and processor demands of your movies--as well as the rendering times.

Frame rate. Frame rate is a critical parameter--and the most commonly misunderstood. Because broadcast-quality video uses high frame rates (24 frames per second or more), most users assume that these frame rates are required for high-quality animation. If your animation is going to be transferred to video then you will want to render it at the target medium's frame rate. **However**, if the animation is going to be played back exclusively on computers, there is rarely any benefit to exceeding 12 or 15 frames per second since such frame rates play back very smoothly. (In fact, a lot of professional animation actually is done with 12 or 15 real frames per second with each frame being shot twice to satisfy the hardware playback rate.) In fact, using high a frame rate that is too high may make the playback less smooth, since the resulting data rate may exceed the computer's capabilities--especially when a CPU-intensive codec is used. (When creating content for Videodelic, you may want to use a high frame rate since Videodelic only plays the frames it needs--so higher frame rates give it more frames to choose from and hence smoother playback.)

Frame dimension. The frame dimension may be dictated by your project's needs. If you are rendering for DV, you need to decide whether to render at the DV pixel dimension or a square-pixel equivalent (this topic is covered later in this chapter). If the animation is intended for computer playback, the frame dimensions will need to be balanced against the frame rate, codec and playback power of the computers that will access it. Full-size frames won't play back at broadcast frame rates on most computers. 400 by 300, 320 by 240 and 240 by 180 are commonly used frame sizes. Some codecs, such as Sorenson, display reasonably well at twice the rendered size.

Codec. This is the most confusing and difficult choice to make if it isn't dictated by the video editing software you are using. In general, the codecs that offer the greatest storage savings are also the most processor-intensive for playback and aren't appropriate if the video will need to be edited or re-compressed. Sorenson, for instance, offers incredible space savings and very high quality output, but it places high demands on the processor--so the frame rate and frame dimension need to be adjusted accordingly. 12 or 15 frames per second at 320 by 240 play back well with Sorenson on a wide variety of machines and you often get files less than 1/10th the original size. At higher frame rates or with larger frame dimensions, older machines may not be able to play the movies. Also, video compressed with Sorenson shouldn't generally be compressed again since the quality degrades significantly on the second pass.

If you are going to edit the rendered animation, Animation at medium or high-quality is usually a good choice. Sorenson is often a good choice for compressing the final edit for distribution since it maintains high quality and provides enormous space savings. The very best quality post-production compression for computer playback can be achieved with the Sorenson Pro codec when used with Media Cleaner or Cleaner 5 from [Media 100](#).

There is a great web site called Codec Central where you can learn more about the available QuickTime codecs. It is located at <http://www.terran.com/CodecCentral>.

Square vs. Non-Square Pixels and Aspect Ratios

Computers have square pixels. So, an image that is 640 pixels wide by 480 pixels high has an aspect ratio of 4 to 3 when it is displayed--precisely the ratio of the pixel dimensions. On the other hand, some digital media (such as DV tape) have non-square pixels--the height and width of the pixels are different. As a result, the ratio of the pixel dimension does not match the aspect ratio of the played back image. In the U.S., for instance, NTSC DV format has 720 pixels wide by 480 pixels high BUT it plays back with a 4:3 aspect ratio making it equivalent to a 640 by 480 image.

There are a couple of different ways to deal with this situation. You can render your movies with the delivery medium's pixel dimensions (such as NTSC MiniDV's 720 by 480), in which case the image will look slightly different when played back on an actual DV device (such as a DV camcorder or deck). Or you can render the movie with the correct aspect ratio when viewed on the computer and let your video editing software provide the necessary compensation. (Most pro and semi-pro video editing software compensates for these dimensions automatically.) For example, if you are working with NTSC (U.S.) DV, you could render at **either** 720 by 540 or

640 by 480. See your video editing software's manual for details about its preferred import dimensions for computer-generated content.

When rendering animation, ArtMatic Pro will adjust for DV's non-square pixels, **wDV FORMAT NOTE** When a DV format is selected via the format popup menu in the QuickTime Export dialog. To bypass this adjustment, choose a non-DV format from the popup and type in the desired width and height.

Cross-platform Playback

For cross-platform playback, your movies need to be 'flattened'. Some video editing programs do this automatically when rendering. If the movie was created by ArtMatic or Videodelic, you need to open the file with the QuickTime Player and choose Save As with the 'Save Self-Contained' option to flatten it.

Frequently Asked Questions

[What is a Quicktime file and what is a codec?](#)

[If I use a codec will the image quality suffer?](#)

[Why is movie playback so jerky/unsteady?](#)

[Why is playback unsteady or the image pixelated?](#)

[What frame rate and frame size should I use?](#)

[What codec should I use?](#)

[Why doesn't my movie play back from web browsers after I upload them?](#)

[What settings do you recommend for use with MediaCleaner/Cleaner 5?](#)

[Should I let your apps do the compression or should I use MediaCleaner/Cleaner 5?](#)

What is a QuickTime file and what is a codec?

Unlike 'jpeg' or 'pict' or 'mpeg', '.mov' isn't a single data format. A movie file is a wrapper which contains movies and/or graphics and/or sound, etc. and any of that data can be encoded in a large number of formats. When animation and/or sound are saved to a movie, there are a number of formats those tracks can have. The translation of the tracks is handled by translators that are called CODECs (for compressor/decompressor). Even the data formats that don't use compression are translated by a sort of null codec.

If I use a codec does will the image quality suffer?

Not necessarily. Some codecs are 'lossless' which means that the decompressed frames are identical before and after compression (Animation at the highest quality setting is lossless). Some codecs are lossier than others.

Why is movie playback so jerky/unsteady?

In almost all cases, this is the result of some combination of the following factors:

1. The movie was rendered with a frame rate and/or frame size larger than the processor can play back in real time. If the frame rate is too high and/or the frames too large, your computer may simply not be able to play the movie smoothly. There is a complex set of factors that determines how smoothly a movie will play back that includes: frame size, frame rate, codec and processor speed. There are combinations of size/rate/codec that will defeat even the fastest G4! For solutions, read on.
2. The codec is too processor-intensive to accommodate the frame rate and/or frame size of the movie. It should be kept in mind that the codecs that offer the best image quality with a lot of space savings are also the most processor intensive. For example, Sorenson is an amazing compressor that can provide savings of 10 times or more disk space BUT it also needs a fair amount of processor power to do the decoding. So, if the frame rate is high and the frame dimensions are large, even a G4 won't be able to play it back smoothly. This is especially true of the DV codec. DV is an amazing compressor that saves

lots of disk space AND has almost no quality loss BUT smooth playback really requires the compression/decompression to be handled by hardware--so playback won't be smooth until you transfer the data back to DV tape.

3. A poor-quality codec was used or the quality setting was too low. The quality setting can have a large influence on the image quality.

Why is playback unsteady or the image is pixelated? I thought your apps generate is professional quality output.

Unsteady playback is usually the result of compressing at a higher data rate than your computer can play back. Even a 500 mHz G4 can choke on a 720 by 480 file with a 30 fps frame rate and a CPU-intensive high quality compressor like Sorenson (or others). The movie itself isn't unsmooth. It is the CPU's playback. If the movie isn't intended for transfer to video or DVD, the frame rate and frame dimensions should probably be reduced.

What frame rate and frame size should I use?

It depends on how the movie is intended to be played back. If the movie is destined strictly for DV tape or DVD, then you need to use the frame rate and frame dimensions required by the medium (for NTSC DV you need 29.97 drop frame with 720 by 480 -- or 640 by 480, depending on your video editing app). If you render with those settings, you will need to use an application such as Media Cleaner to render versions at lower frame rates and frame sizes for computer playback.

If the movies are destined for computer playback, frame rates of 12 to 15 frames per second is generally indistinguishable from higher frame rates. In fact, higher frame rates often play back with glitches you won't see at lower frame rates. You might be interested to know that an awful lot of movie animation is actually done at 12 or 15 real frames per second with every other frame being a duplicate of the previous one.

As for frame size, this is even more critical than frame rate. With frame dimensions of 640 by 480, for instance, my 400 mHz G3 will stutter on files with frame rates as low as 15 fps--while the same movie at 30 fps will play fine at 320 by 240. Common sizes that work well (depending on the machine playing them back) are 240 by 180 for movies that folks will download, 256 by 256, 320 by 240 and 400 by 304 and, 480 by 360.

TIP: Pick dimensions that are multiples of 16 in both directions since some codecs work best when this is the case.

What Codec should I use?

The answer really depends on what is going to happen with the movie after it is rendered. If you are planning on editing the movie with a movie editing app, you should use a codec such as Animation (at its high or best quality settings) or a DV codec if you are going to be using iMovie or FinalCut or EditDV or some other DV-based movie editor. While these codec generate larger files, the increased quality is a must--especially if the movie will ever be recompressed or edited since compressors like Sorenson that have few artifacts the first time around yield less than spectacular results on the second trip through the compressor.

Also, you should use one of the codecs mentioned in the previous paragraph if you will need to use an app like MediaCleaner/Cleaner to generate versions at multiple frame sizes or frame rates (such as when you need to create high-bandwidth and low-bandwidth versions).

If you are going to be playing the movies back from your computer and won't be recompressing them or editing them then Sorenson yields the best quality at its highest settings while also generating files that are about 1/10th the size of uncompressed files. It is pretty much the standard on the Mac.

There is a great web site with detailed information and examples of all the available codecs. It is found at:
<http://www.terran.com/CodecCentral>.

TIP: When performing test renders, try using the VIDEO codec. While this is not the highest quality codec, it is pretty fast and has reasonable space savings. I frequently use it while I am working on an animation so that I can

get a sense of how it is coming along. Its speed is a real convenience and the quality is acceptable for rough drafts.

Why doesn't my movie play back from web browsers after I upload them?

There is an easy fix, but first the reason: many web servers (and Windows machines) need the movies to be in 'flattened' form which has all the movie data arranged in a cross-platform-compatible format. When our apps render the movies, they aren't flattened since flattening adds to the rendering time AND reduces efficiency when editing the movies with some movie-editing applications.

To flatten the movie, open the file with QuickTime Player, choose Save As with the 'save as self-contained' option. This will "flatten" the movie so that it will play back from web servers, Windows machines, and Macs with equal ease.

What settings do you recommend for use with MediaCleaner/Cleaner?

I tend to turn off the Adaptive Noise Reduction since it seems that computer-generated animation tends to be blurred by it. Also, I like to use Sorenson Pro with the highest possible data rate AND the two-pass mode on. The two-pass mode dramatically reduces the artifacts so that most of the time the resulting movie is artifact free.

Should I let your apps do the compression or should I use MediaCleaner/Cleaner 5?

It all depends, if you have plenty of disk space and some time and Sorenson Pro, I would say let MediaCleaner/Cleaner do the work. The reason is that MediaCleaner/Cleaner can take advantage of some high quality options not available outside of their environment. In particular, the two pass mode improves the quality of the compressed movie dramatically.

What's New in 2.0

ArtMatic 2.0 is a major advance over earlier versions of the program. Since version 1.0 was first released, we have been hard at work on ArtMatic and have made it both easier to use, more flexible and more powerful. You will be astounded by the range of images, sounds and animations of which ArtMatic 2.0 is capable. There are new and improved graphic functions, editable structure trees, a powerful new interface for creating animation, and much more. We have improved and expanded the documentation and provided tutorials that will help you get the most of this incredible tool.

Here are the highlights:

- Improved user interface, making ArtMatic both easier to use and more powerful,
- Customizable ArtMatic component trees,
- New and improved graphics functions/components,
- Component functions are now organized by function type in the popups,
- New keyframes feature, lets you store 16 images/locations per structure,
- Keyframe-based animation,
- **Animation** menu with commands that unleash ArtMatic's power,
- Editable parameter paths and camera path for animations,
- New complex shading functions,
- Zoom and gradient changes can now be animated,
- Batch and background rendering,
- Disk-based image rendering that allows larger images to be saved,
- Improved memory handling (exported images no longer need to fit in memory)
- More powerful **Mutations** dialog,
- Ability to save gradient libraries,
- New sound algorithm
- and many, many more improvements and refinements.

Some Important New Features

There have been two user interface changes that users of older versions of ArtMatic should know about.

- **Aborting QuickTime rendering**

In ArtMatic 1.x, rendering was canceled by clicking the mouse button anywhere. In version 2.0, pressing the mouse only cancels rendering when background rendering (a new feature) is turned off. When background rendering is turned on, the Escape key (and commane-period) abort rendering.

- **Saving large pictures**

In earlier versions, ArtMatic needed a large memory partition to save a large picture. Picture rendering has been improved to allow ArtMatic to save pictures much larger than will fit in memory. As a result, it is no longer necessary to assign a lot of memory to ArtMatic to save large pictures.

Keyboard Shortcuts

There are a number of keyboard shortcuts which you may find convenient for manipulating ArtMatic.

Shortcut Key	Shortcut Name	Notes
up/down arrow keys	zoom	continuous zoom out/zoom in
left/right arrow keys	cycle functions	cycles to the next/previous component function for the selected tile. (Note: affects all keyframes)
/	zoom in (2x)	zoom in by a factor of 2
-	zoom out (2x)	zoom out by a factor of 2
tab key	select next	select the next tile in the structure
[]	cycle shaders	cycle to the next/previous shader. (Note: affects all keyframes)
R	randomize	rolls the large die.
spacebar	keyframe animation	start/pause/continue keyframe animation playback
option-spacebar	random path animation	start/pause/continue random path animation
escape key	abort/escape	abort rendering of large pictures or animation
e	toggle compiled edit	If a compiled tree icon is selected, this shortcut lets you view and edit the compiled tree. Typing the shortcut again returns to normal view.
d	component sets depth	Shortcut for the Component Sets Depth command of the Shading Options popup menu. It also turns on depth cueing.
d (after numeric entry)	make degrees	Type 'd' after entering a numerical value in a field of the Parameter Envelopes dialog to convert the value from degrees to radians (the format ArtMatic uses internally for angles).
s	component sets shade	Shortcut for the Component Sets Shade command of the Shading Options popup menu. It also turns Global Shading on.
p	render full screen	Equivalent to pressing the Render Full Screen icon.
m	make main output	Make the selected component (of the last row) the system's output component. This is a rarely used command with no user interface equivalent. It only has a meaning when the last row of a system has more than one component. Typing the shortcut will make the selected component the one which the system uses to calculate the canvas.
Special Clicks		
shift-parameter change		apply parameter change to all keyframes
shift-gradient change		apply change to all keyframes
shift-zoom click		apply change to all keyframes

control-parameter click	reset parameter	set the parameter to its default value
command-shift parameter click	ramp parameter	creates an envelope ramp that covers the full range of the parameter's possible values
control large die click	super randomize	randomly select a tree structure and randomize all components and parameters

Troubleshooting

This chapter describes common problems and their remedies. If your problem isn't covered, see the chapter [Getting Help](#) for your support options.

General

The system/structure responds surprisingly or misbehaves in some way

ArtMatic Pro's structure editing features make it possible for you to create inconsistent trees or to connect components that might not really make sense together. Here is a list of things to check if your structure behaves surprisingly:

- Check for open (unconnected) inputs and outputs in the system - if the third input of any three-input tile is not connected then the input will be fed by the third global input (sometimes called time) which will result in the system changing during an animation even if the parameters stay constant. If you have unintentionally left an input unconnected, follow the instructions in the **User Interface** chapter for forcing a connection between tiles.
- Check to see if there are any components with 'packed' in their name that are not being fed from a **pack** component.
- Check to see if a **pack** component's output is being fed to a component that does not have 'packed' in its name. Only some components understand the output of **pack** components.

The system/structure takes forever to redraw

Check to see if the system contains a compiled tree or an iteration component. If the number of iterations/recursion is set high enough (or the system complex enough), it might take a very long time to compute the system. While working with such systems, it is often advisable to set this value to a very small value and set it larger when rendering.

The canvas was left only partially drawn

When ArtMatic redraws the canvas, clicking the mouse will interrupt redraw. Generally, this behavior is desirable since you may want to change something without waiting for ArtMatic to finish drawing the canvas. You can force the canvas to be redrawn by clicking in it.

Changing the gradient has no effect on the system

Check the number of outputs of the last tile in the system. If this tile has three outputs then the system uses RGB rather than gradient shading. Take a look at the [Concepts](#) and [Shaders](#) chapters if you are unclear about the the RGB/gradient distinction.

The colors in my project changed unexpectedly when changing shaders

Check the global shading options to see if the Depth Cueing or Global Shading status changed unexpectedly. This can happen when inadvertently pressing a shortcut key such as 's' or 'd' which turn on shading options. Also, in some versions of ArtMatic, changing

shaders using the keyboard shortcuts results in the Depth Cueing or Global Shading settings not being reset when changing shaders.

Sound

I can't stop sound playback, what do I do?

Be patient; real-time sound animation can be very processor intensive and takes a while to abort. Press and hold the escape key (or press and hold the mouse button) until ArtMatic aborts the sound playback. This can take up to a minute on a very slow machine or with a CPU-intensive structure.

Sound output has dropouts or distortion or doesn't work

If your computer has a slow processor, virtual memory is turned on, or if the ArtMatic structure has computationally-intensive components, smooth audio playback in realtime may not be possible. When rendered, the sounds will not have these dropouts. If virtual memory is on, turning it off may help. You may also try eliminating processor-intensive components from the structure.

When I press the Audio-In Control icon, nothing (or very little) happens

Make sure that the sound input sensitivity is turned up and that you have selected an appropriate sound input device in the Sound control panel.


Sound input doesn't work, or I get an error when accessing sound input

Check to make sure that you have selected a sound input device in the Sound control panel and that you can record a sound there. In some versions of OS X, sound input does not work in compatibility mode, and you will get an error message.

If the ArtMatic structure is too computationally complex, sound input may not seem to work. Choose a simple structure from the Structures popup menu and see if sound input works with it. If it works fine then the problematic structure may be too complex for ArtMatic to modulate it with sound in real time.

Animation

When I press animate keyframes nothing or very little happens

This is generally the result of the animation duration being very short or a lack of keyframes. Check the duration of the animation by mousing over the [duration icon](#) . You can alter the duration by clicking on the duration tool and dragging left or right. You can also edit the duration in the [QuickTime Export](#) dialog.

Make sure that you have added keyframes. ArtMatic will create a couple of keyframes if there aren't already keyframes, but the animation may not be terribly exciting.

I get an error rendering animation larger than 2 gigabytes

Some versions of QuickTime (4.1.1 and earlier) and some OS (operating system) versions (pre-9.0.4) are not capable of creating QuickTime files larger than 2 gigabytes. To create files larger than 2 gigabytes, you will need system 9.0.4 or later **and** QuickTime 4.1.2 or later.

Also, make sure that the animation is being saved to a disk that was formatted with the Mac OS Extended option. If you select the disk's icon and choose the Get Info command, you can check to see whether the volume was formatted with the extended file system option.

We have verified that system 9.0.4 in combination with QuickTime Pro 5 will allow ArtMatic Pro to create animation larger than two gigabytes. This should also work with QuickTime 4.1.2.

Rendering seems very slow, what can I do to speed it up?

There are a number of factors which influence render time: the project's complexity, frame size, frame rate, and QuickTime codec. Some projects take a long time to render no matter what you do. The results are worth the wait. If the project uses movie input, the time taken for QuickTime to decompress the frames can significantly add to the render time.

We recommend doing low frame-rate (10 to 12), small frame-size renders before doing your final full quality render.

Using pictures & movies

How do you use a picture other than the U & I logo as the default?

Move the picture that you would like to use as the default into the ArtMatic folder and rename it "DefaultPicture". The picture should be in Macintosh PICT format.

How do you remove a picture from the popup?

If you hold down the option key and select a movie/picture from the popup, you can change the image used in that slot, but you can't empty the slot.

How do I use a picture within an ArtMatic system?

You need to use a Movie/Pict component (they come in 2-in/1-out and 2-in/3-out flavors). Use of these components is covered in the [Using Pictures and Movies](#) chapter and also in the tutorial chapters.

I rendered an animation that used a movie input and the motion was uneven.

Check the settings in the **Input Movie Setup** dialog. The movie can be played at its native speed or with an interpolated duration. The **Use Movie's Own Time** option ensures the smoothest motion since no frame interpolation is done.

Memory

ArtMatic can run in a memory partition as small as 10 megabytes. ArtMatic 2.5 has improved its memory handling when saving pictures. The application no longer needs a large memory partition to save large files.

You may need to increase the memory partition if you are creating systems that use nested (compiled) trees or use several movie/picture inputs.

ArtMatic Pro Frequently Asked Questions

Are ArtMatic Pro 2.5, 2.0 and ArtMatic 1.x compatible?

ArtMatic Pro can read ArtMatic 1.x files. Please note that some improvements in the component functions (such as increased parameter ranges and an improved random table generator) will result in a different appearance for some files. In most of these cases, some tweaking of parameters will fix matters. In some cases, primarily those that have random functions, the file will have a different appearance when rendered with ArtMatic Pro. Most ArtMatic Pro 2.0 files will appear the same in version 2.0 and version 2.5.

What are the differences between ArtMatic 1.x and ArtMatic Pro?

There are a large number of differences between these two versions. ArtMatic Pro has added a large number of features for professionals and power users. 16 keyframes can be stored per file. The maximum picture size has been dramatically increased. Tree structures can now be modified. Color change, zoom and camera path can be animated. Parameter envelopes have been added for increased animation control. More and improved component tiles have been added. New complex shaders have been added.

Version 2.5 introduces RGB truecolor mode which allows you to use pictures and movies as inputs in true color mode as well as to provided sophisticated color processing.

I recently bought ArtMatic 1.x, how can I upgrade?

ArtMatic Pro is a new version with many features for power users and does not replace ArtMatic 1.2. However, we are offering a very special price for registered ArtMatic 1.x owners.

The real-time animation looks "blocky". What's wrong?

Nothing is wrong. ArtMatic systems are computationally complex and high-quality output cannot be calculated fast enough for real-time animation. As a result, we use a lower resolution when doing real-time animation. However, when you render your animation with QuickTime, we use high resolution to create outstanding looking output.

How can I customize the audio input animation?

The mapping of sound frequencies to parameter values is not customizable though you can use the **sound in sensitivity** preference to control how sensitive ArtMatic to sound input.

What are the system requirements?

A PowerMacintosh computer with system 8.0 or later and a monitor with a resolution of at least 800 by 600.

On my iMac, some buttons look slightly 'cut off'?

The user interface needs a few pixels more than 800 by 600. All functions and tools are available and visible but the labels on the lowest row of tools may be slightly cut off.

Getting Help

While we have tried to design our applications to be easy to use and robust, there may be times when you have questions or problems and need help. Below are guidelines for receiving support. Please note that we strive to both provide high-quality responsive support and to keep our products affordable to the artists and educators that use them. To do this we need your help. You can help us by following the guidelines laid out below:

- **Check the manual.** 90% or more of the support questions we receive are answered in the documentation that comes with the software. Please take the time to look through the documentation before contacting us. In some cases, you may have to look in a couple of different chapters to find the answer. If you are browsing the pdf version of the documentation, use the Find command to search for keywords related to your question.
- **Perform the tutorials.** About half of the questions we are asked are covered by the tutorials. We recommend that you not just read but actually perform the tutorials. Sometimes concepts or methods are clear when performing a tutorial that might not be evident from just reading it. We update the tutorials in each new version of the documentation to cover frequently asked questions. So, even advanced users find that they pick up new techniques and tricks by performing them.
- **Check the web site.** Our web site is found at <http://www.uisoftware.com>. Both the product home pages and the support page are good places to look for answers to your questions. News of product updates and updated documentation and FAQs will be available there as well as additional tutorials and examples.
- **Join our email-based users group.** The UILIST email-based users group is a valuable resource for both getting questions answered and sharing tips, techniques and insights. The UILIST features many creative, talented artists that share discoveries and help each other out. The UILIST has also spawned a number of popular ventures such as the MetaCamp and the MetaSynthia CD-ROM magazine. The list can be joined by sending a blank email to: uisoftware-subscribe@topica.com
- **Search the archives.** Even if you haven't joined the user list, you can search its archives by visiting: <http://www.topica.com/lists/uisoftware/read>
- **Write to us.** In order to provide the highest-quality support, our development team, rather than tech support people, provides free email-based tech support. We strive for excellent responsiveness. Most questions are answered the day that they are received. We regret that we are not generally able to provide phone support, but we try to make up for this by providing responsive support from the people that actually created the software. Technical support can be contacted at support@uisoftware.com