DARKLING SIMULATIONS'

DARKTREE TEXTURES 2.0

User Guide
with DarkTree 2.5 Addendum

Revision 2- June 2, 2003

Acknowledgments

Darkling Simulations wants to take this opportunity to thank our design team for all the hard work and long hours they've put in to making DarkTree Textures 2.0/2.5 the first rate product it has turned out to be.

We would also like to thank our beta testers for their many insightful suggestions.

THANKS EVERYONE!

Browse our website at www.darksim.com

Darkling Simulations

181 Piedra Loop,

Los Alamos, N.M. 87544

USA

Phone: 505 672-0640

FAX: 505 672-1296

email support: support@darksim.com

Licensing Agreement

DarkTree Textures is licensed to you, not sold, by Darkling Simulations. You are granted the use of one copy of the software, which must be installed on one computer at a time only. This software may not be copied, or otherwise reproduced, except as expressly permitted under this agreement or as granted in writing by Darkling Simulations.

You are allowed to transfer the software to a single hard disk, provided you keep the original CD solely for backup or archival purposes. You may transfer the software with all written materials on a permanent basis. In such a case you must remove the software from your system immediately.

Darkling Simulations warrants that this program will perform as described in published specifications and in the documentation supplied in this package, with the provision that it is used on the computer hardware and operating system for which it was designed. Use of the DarkTree Textures software accompanying your license and its documentation are governed by the terms set forth in your license. Such use is at your sole risk. The software, the accompanying files and the documentation (including this file) are provided “AS IS” and without warranties as to performance of merchantability or any other warranties of any kind, whether express or implied. No warranty of fitness for a particular purpose is offered.

This software is provided by DARKLING SIMULATIONS “as is” and any express or implied warranties including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall DARKLING SIMULATIONS be liable for any direct , indirect, incidental, special, exemplary or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Contents

.....
8

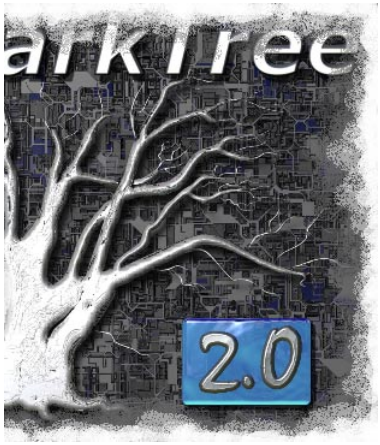
Part 1
The Preliminaries

Introduction

About the Manual

Before You Begin

Section 1.1
Introduction



1.1.1 About DarkTree 2.0

Welcome

Welcome and thank you for choosing DarkTree Textures 2.0. DarkTree is an intuitive application for creating high-end procedural shaders, called DarkTrees. While the software is designed primarily for 3D animators, game developers and visual effects mavens, it's friendly enough and fun enough for the casual user to enjoy.

What Are DarkTrees and How Are They Created?

A finished texture is referred to as a DarkTree. DarkTrees are built up from a selected group of algorithmic mini-textures or components linked together into a conceptual tree format. Linking simple components together to build complex DarkTrees ensures that your finished textures will be uniquely your own. You can design "real world" textures with built-in random variations that make them absolutely photo-realistic; textures reminiscent of some fantastic dreamscape; or complex, mathematically-perfect geometric designs. Whichever textures you need you can almost certainly create. A glance at the Texture Repository on our web site will convince you that this statement is no exaggeration.

With the exception of the Generator and Shader component classes, each DarkTree component comes in three discrete types: color, percent and bump. These three types are designed to be layered together, resulting in a perfectly integrated image that includes depth and shading. Although a specific DarkTree may include linked components of more than one type, the completed texture will always be classified as either a single or a shader type (see the note).

NOTE: The Shader component class is a special composite of the three basic types.

By the way, DarkTree Textures is shipped with a library of pre-made DarkTrees. This gives you the opportunity to examine the tree structures to see how they've been put together.

Exporting Your DarkTrees as Bitmaps or Shaders

Using the program's Bitmap Renderer you can export your finished DarkTrees, as algorithmically-generated bitmaps, to a wide selection of modeling/rendering applications. DarkTree Textures can generate bitmaps in a large number of formats, giving you the greatest possible choice of packages. (See Appendix A for a complete list of input/output formats.)

In addition though, we have developed specialized plug-ins, called Simbionts, for some of the more popular 3D modeling/rendering applications, allowing you to import DarkTrees directly as procedural shaders. (Aseparate plug-in must be written for each individual package before DarkTrees can be imported in this way.) Those Simbionts that are inlcuded with your DarkTree Textures purchase are located on your DarkTree CD and on our web site (www.darksim.com). We'll be adding new Simbionts to the website group from time to time.

NOTE: For a discussion on the advantages of using procedural shaders, see "Procedurally-Generated Texture Maps VS True Procedural Surfaces" on page 127.

What's New for Version 2.0?

Well, the DarkTree interface has undergone a sweeping face-lift. Included with the "lift", we've also integrated many new features. For example, the Texture Library is now a filtered file system browser. A taste of some other new features includes: a redesigned *DarkTree Color Browser*, a new *Spline Editor*, and the brand new *Tweaks* function. (We're especially proud of our *Tweaks* capabilities.)

One of our goals for 2.0 has been to streamline the DarkTree creation process for our users. So we've concentrated heavily on raising the level of interactivity. We've made a point of significantly improving the workflow and speed as well. As a simple example of this, now you can use a slider to change parameter values within a *Component Editor*, and view the dynamically updating image at the same time.

And finally we've added a variety of useful and, we hope, interesting new components, many of them specifically requested by DarkTree 1.0/1.1 users.

Section 1.2
About the Manual

1.2.1 The Manual Layout

We've divided the DarkTree manual into three parts, each with its own focus and purpose. The succeeding three paragraphs briefly discuss some examples of the issues covered in each of the three parts. The parts are:

Part 1: The Preliminaries

Part 2: The Interface Reference

Part 3: The Conceptual Discussions

Part 1: The Preliminaries (this part) takes care of such initial tasks as introducing you to DarkTree Textures. Also included here is a brief outline of the many changes in version 2.0, plus three schemes for learning the program - which includes a *locator guide* for the tutorials in the manual. The final sections of *Part 1* cover installation issues and how to get help for your puzzlements and problems.

The task of *Part 2: The Interface Reference* is to provide complete information on every window and button the interface has. It answers such questions as: "What is it for?", "Where is it located?", and "How do I use it?" The interface is divided into four discrete sections, each with its own set of tasks. *Section 2.1* explores the Texture Library, where you can arrange, preview, and relocate your DarkTrees within the file system. The second, *Section 2.2*, covers Examine window features. The Examine windows offer you the chance to view your DarkTrees as complete rendered images. Several new viewing controls have been added to the Examine Windows as well. *Section 2.3* discusses the DarkTree Editor. The Editor is the real workhorse of DarkTree Textures. Use it to design, build, and animate your textures. Several other specialized editors are accessed exclusively from within the DarkTree Editor, some of them new with version 2.0. And finally, the Bitmap Renderer discussion of *Section 2.4* completes the reference material. The Bitmap Renderer converts your DarkTrees to the bitmaps you'll need for importing into your modeling/rendering package (unless you are using one of our Simbiont plug-ins instead).

All four pieces of the Interface Reference have been upgraded, most of them significantly.

Part 3: The Conceptual Discussions is just that - a high-level discussion of the concepts behind DarkTree Textures. It covers several progressive subjects, such as: an in-depth exploration of the components and their classifications, what algorithmic textures are, advanced animation topics, how linking really works and how to maintain fine detail in your elevation maps, to name some of the information given here.

Although you can design some great textures just from what you learn in *Part 2* (especially if you do the DarkTree Editor tutorials), if you want to become an expert user capable of pushing DarkTree Textures to its limits, you must at least read through *The Conceptual Discussions* and complete the advanced tutorials you'll find there.

Besides the three parts, the manual is divided into further sections and subsections as well. The subsection numbering system can help you find a specific subject easily. As an example, consider this section, *1.2.1 The Manual Layout*. The first number refers to the part and there are three of those. The next number refers to a section, and those vary in size and complexity. The final number refers to a subsection, and each section typically has quite a few of those. So, this subsection has a number series of 1.2.1, which means part 1, section two, subsection one.

NOTE: The manual for version 1.0/1.1 included a fourth part, The Texture Reference. The reference is now online. Access it for specific components from within a Component Editor. Simply click on the blue-colored ***Help****.*

Suggestions For Learning DarkTree Textures

We've specifically designed the manual to help you learn how to use the program primarily from the tutorials, which is a lot more fun than reading through a bunch of text. The following learning suggestions are for three levels of expertise. The three levels are: the *Fast User*, the *Proficient User*, and the *Expert User*. These user titles do not imply anything about user ability. They're simply based on time constraints and what you want to get from the software.

The *Fast User* doesn't have time to learn DarkTree really well. He or she needs to be able to use it yesterday, so the following guide gives the quickest way to be "up and running". The *Proficient User* is interested in creating complex textures that take some serious thought to design, but again has to have something in hand yesterday. He or she needs enough information to understand how to use some of the less obvious component and program features. And finally, the suggestions for becoming an *Expert User* include even more reading (whoopee). The *Expert User* plans on exercising DarkTree Textures often, wants to know everything there is to know, and is prepared to push the software hard.

Here are our suggestions for becoming one of the above users.

THE FAST USER

Step 1. Work through the three DarkTree Editor tutorials in *Part 2*. These three tutorials cover using the Examine Window, *Tweaks* and an animation, while briefly covering the main DarkTree-creating chores. You can manage surprisingly well by just doing these tutorials, plus browsing through the information on each component you plan to use.

THE PROFICIENT USER

Step 1. Do the *Fast User* step.

Step 2. Read through *The Basic Concepts Of DarkTree Textures*, beginning on *page 143*. This is the first subsection of *Part 3: The Conceptual Discussions.*

Step 3. Read through *The Component Classifications beginning on page151*.

Step 4. Work through all of the tutorials in the section mentioned in Step 3. (There are quite a few tutorials in this section, but they're short. **:-)**)

THE EXPERT USER

Step 1. Complete the steps under *The Proficient User.*

Step 2. Read and understand the rest of *Part 3: The Conceptual Discussions.*

Step 3. A breakdown and analysis of some of the textures provided with the program is one of the very best ways to understand the non-obvious links you can make using specific components. Read about and really understand the components and component parameters that you find yourself using a lot.

Step 4. This step really gives some of the best advice. Try linking up components and parameters in every crazy way you can think of. It helps you get your mind out of linear mode, plus we guarantee that you'll find some effects you'll really like. This software gives you lots of freedom, so take advantage of it!

FOR ALL

For help with initially finding your way around the DarkTree Textures interface, go to the *main application menu*, and click on ***Help>Getting Started Tips>Enable***. The ***Enable*** option is a toggle that you can turn on and off by selecting it. When **Enable** is toggled on, the application will open a set of information windows.

NOTE: *Don't read through everything in Part 2: The Interface Reference. Just use it for answering specific questions, unless you like lots of reading.*

Figure 1.1
A beginner's information window for the DarkTree Editor.

1.2.2 About the Tutorials

A Few Words...

We've tried to design the tutorials to be informative, but also to be a pleasant learning experience. We especially want our users to be able to learn the graphical interface entirely through working the tutorials. The tutorials are designed as a series of steps but with lots of extra explanations, figures, and "asides" intermixed.

The tutorials are scattered throughout the manual, but always located after the subject that each is meant to teach you about. The following is the *Tutorial Locator Guide*. It can tell you what each tutorial teaches and exactly where in the manual it's located.

The Tutorial Locator Guide

To make it easier for you to choose the best tutorials for your needs, the following table lists what each tutorial teaches and where it is located.

Tutorial Locator Guide

Grimalkin Wood	***This tutorial teaches basic Editor controls while creating a simple wood texture.***	*page 109*
Metallic Pastiche	***You'll learn how to use the Tweaks facility, apply metallic attributes, plus further Editor controls.***	*page 116*
Rolling Smoke Ring	***This third Editor tutorial teaches you how to create an animated texture and make a movie of it in the Examine Window.***	*page 121*
Control Tutorial 1	***This tutorial teaches you how to get subtle color variations in a set of bricks.***	*page 156*
Control Tutorial 2	***You'll learn how to use one of the shuffle components to break up a pattern in each of a series of scales.***	*page 157*
Control Tutorial 3	***You'll learn to vary surface roughness in a set of bricks by indirectly linking to a Randomizer component.***	*page 159*
Deform Tutorial 1	***This tutorial teaches you how to create the refractive caustic lights on the bottom of a swimming pool.***	*page 161*
Deform Tutorial 2	***In this tutorial, you'll twist an external image thereby creating some neat effects.***	*page 162*
External Tutorial	***This tutorial demonstrates how to create a face of clouds, using an external face image.***	*page 163*
Generator Tutorial 1	***In this tutorial, you'll learn how to use a generator to control a pattern's shape.***	*page 165*
Generator Tutorial 2	***You'll learn how to use the Blend Function parameter in conjunction with a generator.***	*page 165*
Generator Tutorial 3	***With this tutorial, you'll learn how to use a generator to control the surface of some bump-type dots.***	*page 166*

Section 1.3
Before You Begin

1.3.1 Overview

Now that you have the DarkTree Textures package sitting on your desk staring back at you, let's run through a few preliminary checks. First you'll want to make sure nothing is missing from the box. And, you'll want to know where to call if the contents are incomplete.

Before you install, double-check the system requirements to make sure yours meet the minimum, at least.

Now, about the registration card we've provided. We're trying to gently henpeck at you to get it sent now so that we can get your information into our database.

And last, we've included a section on the help avenues that Darkling Simulations offers. We want to be able to answer your questions as quickly and accurately as possible, because we know your time is valuable!

NOTE: *If you've ordered DarkTree directly from our web site, we'll register you automatically.*

1.3.2 Checking the DarkTree Box

Each DarkTree 2.0 package contains the following essential items:

- *one CD*
- *one set of shrink-wrapped manual pages*
- *a binder for the manual pages*
- *one registration card for you to fill out and return*
- *a sticker with the registration number, located inside the back cover of the binder*

If, by chance, one of the box items is missing, you can inform us via email (write to *support@darksim.com*) or phone us at (505) 672-0640.

1.3.3 The System Requirements

The following is a list of the minimum system requirements you'll need to effectively run DarkTree Textures.

- *Windows 95, Windows 98, Windows ME, Windows 2000 or* Windows *NT4.0*
- *an Intel Pentium or AMD Athlon - with CPU running at 500 Mhz or greater (multiple CPUs supported)*
- *a 24-bit video display with a minimum of 800 X 600 resolution*
- *64 MB of RAM*
- *a CD-ROM drive.*
- *a mouse or other compatible pointing device*

1.3.4 Registration Information

The contents of the DarkTree package include a registration card and a sticker printed with the registration number. (The sticker is positioned inside the back cover of your folder.) You can either fill out the card, including the registration number, and send it to us or register online at our web site, www.darksim.com.

Darkling Simulations offers two ways for you to get help with a problem. If you cannot find an answer in the manual, you can select one of the following avenues. But please check the manual first.

Avenue 1: For the more technical or detailed questions, you can email us at *support@darksim.com.* We schedule some time to answer email questions on every working day. Email is probably the quickest way to get an answer from us, unless your question is very simple.

Avenue 2: For simple questions, you can call us if you prefer. Our phone support hours are Monday through Friday, between 10:00 am and 4:00 pm MST (Mountain Standard Time). For phone support, call (505) 672-0640.

When you email or call us for support, please give us the name under which DarkTree Textures is registered and the registration number.

And Finally - The Privacy Issue

We at Darkling Simulations have strong feelings about privacy. Therefore be assured that any information we ask for on the registration card is for our enlightenment only. We do not sell or share customer information with anyone.

Part 2
The Interface

The Texture Library

The Examine Window

The DarkTree Editor

The Bitmap Renderer

Section 2.1
The Texture Library

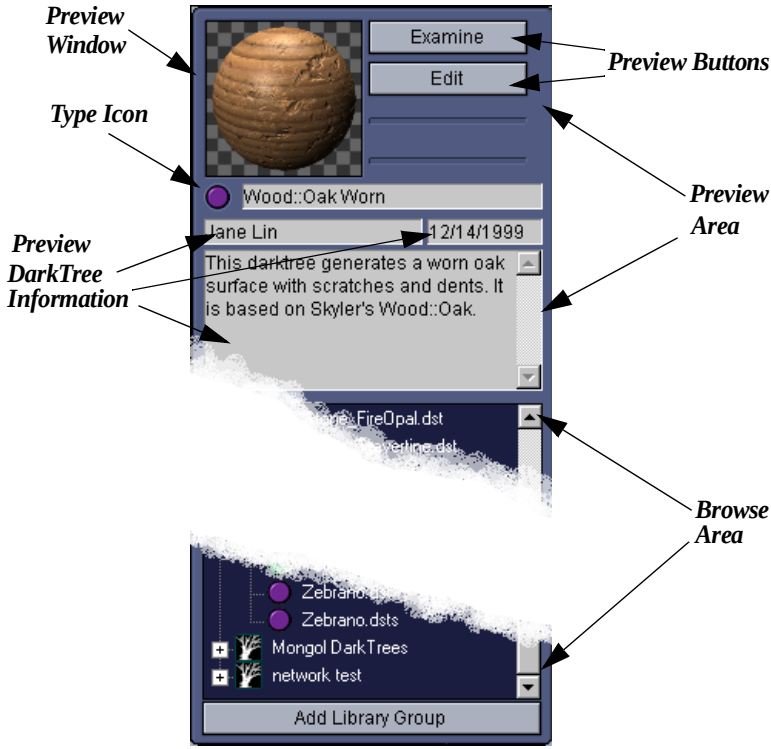


Figure 2.1

DarkTree's Texture Library. The areas that are labeled and pointed out in the figure are those that we discuss in this section.

2.1.1 Texture Library Overview

The Texture Library is a filtered file system browser similar to Windows Explorer®. It provides a convenient place to browse, organize and edit your DarkTrees. You'll find the pre-built DarkTrees included with this package located here as well. The Texture Library is intuitive to use, providing some useful shortcuts such as the ability to quickly load a DarkTree into the DarkTree Editor. We've included the Texture Library solely for your convenience; DarkTree Textures does not require that you use it. However, we think you'll find it tailor-made for keeping track of your quickly-expanding collection of textures.

The Texture Library occupies the left most lengthwise third of the main application window. The Library falls logically into two sections; the preview area (the top half), and the file browser area (the bottom half). *Figure 2.1* points out these two areas.

We'll cover the preview area next, followed by the file browser area in subsection *2.1.3*.

A VERY IMPORTANT WARNING!

The new Texture Library is very similar to Windows Explorer®. That being the case, the changes you make will affect the actual files and folders on your disk drive. Please consider carefully before deleting files and folders because the deletion will be permanent! In addition, some folders may contain files that you can't see, since the Library filters out those that are non-DarkTree-related. But when you delete a directory, those hidden files will be deleted as well.

2.1.2 The Preview Area

The preview area (check *Figure 2.1*) provides a place to view and review relevant information for any DarkTree file listed in the file browser area. Based on the preview, you can elect to open an Examine Window or a DarkTree Editor for the same texture.

Previewing a DarkTree

The *preview window* is a small, non-resizeable window that affords you a quick view of any DarkTree listed in the file browser area. The background for *preview windows* is a checkerboard pattern of two gray shades (the default colors). The background forms the edging you see around a planar display. Of course, even more of the checkerboard is visible if the display geometry is a sphere or cylinder.

Along with the texture itself, the preview area can display the following information.

- *A type icon, which tells you the texture type. The type icons are color-coded and accompany each file listed in the file browser area. If the DarkTree is animated, the type icon will also have a play button on its right side. All types can be animated.*
- *The DarkTree file name.*
- *The author (creator) of the previewed DarkTree.*
- *The date the DarkTree was last modified.*
- *Comments about the texture. Is it animated, for example.*

All of the above information except the file name comes from the *Properties page* on the *Edit Control Panel*. This information was entered by the artist at the time the DarkTree was created. Whatever information was keyed-in then is what you'll get. It may be very complete or very sketchy. The *Edit Control's* ***Properties*** tab is accessible from the DarkTree Editor.

NOTE: If you left-click once on a group name or the DarkTree icon, the preview area will display whatever group information the Library has.

Selecting a DarkTree

A single left-click on any DarkTree file listed in the file browser area will automatically render it in the *preview window.*

The Preview Buttons

In addition to the space allotted for previewing a DarkTree, the preview area has two buttons labeled ***Examine*** and ***Edit***, respectively. We'll discuss these buttons next.

The Examine Button

Clicking on the ***Examine*** button opens an instance of the Examine Window, and renders a copy of the DarkTree that is currently in the *preview window.* If you click the ***Examine*** button without first selecting a DarkTree file for preview, the Texture Library will still open an Examine Window. In that case, though, it will be the default window, a blue sphere. You can change the mapping geometry (a plane, cylinder, cube, frame, or sphere) with the Examine Window controls. But by default the opening geometry will be whatever you have set on the *Properties page* of the *Edit Control*.

You can open as many instances of the Examine Window as your system resources will allow.

NOTE: You can readily bypass the preview window and send a DarkTree file directly to the ExamineWindow for immediate display, provided you already have one open. Just left-click on a file listed in the file browser area, continue to hold the mouse button down, and drag the file directly onto the Examine Window.

The Edit Button

Clicking on the ***Edit*** button opens an instance of the DarkTree Editor window and loads a copy of the DarkTree that's currently in the *preview window.* If you click on the ***Edit*** button when there is no DarkTree in the *preview window,* nothing will happen. (See the following notes for two more ways to open a DarkTree Editor.)

The DarkTree Editor interface displays the tree structure of a DarkTree, and handles all of the many tasks associated with building one. By “tree structure” we mean the unique combination of components and links that make up each DarkTree. DarkTree Textures will only open one copy of the DarkTree Editor for each DarkTree file. If you attempt to open another copy with the same DarkTree, the already-opened copy will simply pop to the foreground, if it happens to be behind another window. *Section 2.3* covers the DarkTree Editor in great detail.

AN ALTERNATE WAY TO OPEN A DARKTREE EDITOR: Another faster way to open a DarkTree Editor with a specific DarkTree file is to simply left double-click on the DarkTree file.

OPENING AN EXAMINE WINDOW OR A DARKTREE EDITOR: Right-clicking on any DarkTree file in the file browser area opens a popup menu. The menu, among other things, lets you open an Examine Window or DarkTree Editor.

2.1.3 The File Browser Area

The Texture Library’s file browser area gives you a filtered view of the files in your Library groups (the system directories are chosen by you). That means that all non-DarkTree-related files have been filtered from your group folders and file lists. The file browser area capabilities include adding, deleting, editing, and re-scanning your Library groups, folders, and DarkTree files. In addition, you can move files to new locations and/or rename them. We’ll cover each browser capability in the following paragraphs. If you use Windows Explorer ®, you’ll notice a great similarity.

A VERY IMPORTANT WARNING

This same warning appears in the Texture Library Overview, but it’s so important that we’ve included it here as well.

We’ve already mentioned that the new Texture Library is very similar to Windows Explorer®. That being the case, the changes you make will affect the actual files and folders on your disk drive. Please consider carefully before deleting files and folders, because the deletion will be permanent! In addition, some folders may contain files that you can’t see, since the Library filters from view those that are non-DarkTree-related. But when you delete a directory, those hidden files will be deleted as well.

File Browser Groups

Groups are the Texture Library's root or top level directories. A group consists of a name, pointer to a system directory (selected by you), and a description (optional). You can organize your folders and DarkTree files into any reasonable number of groups. Practically speaking, though, you'll probably want a separate group for each discrete collection of DarkTrees you have. For example, you might want to create one group just for the DarkTrees that are included with this application, another group for your experimental DarkTrees, and yet another group for your company's DarkTree archive.

By right-clicking once on a group name or the accompanying DarkTree icon, you can access a popup *group menu* that has some extra options. *Figure 2.2* is a screen capture of this menu.

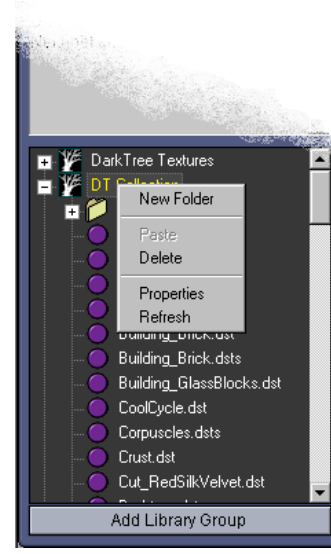


Figure 2.2
An example of a group menu.

NOTE: Deleting a group, unlike the file/folder deletion we've been warning you about, won't delete anything but the group's name, description and pointer.

Adding a New Group

The task of selecting a system directory and then creating a new group from it is one extended operation. Here's how to do it. First, click on the ***Add Library Group*** button located across the bottom of the Texture Library. This opens the *Browse for Folder* dialog. Select a directory that has DarkTree files within it (maybe in lower-level folders) and click the ***OK*** button. *Figure 2.3* is a screen capture showing the ***Add Library Group*** button and the *Browse for Folder* dialog.

After you select a directory and click ***OK***, the Library opens the *Edit Groups* dialog. *Figure 2.4* is a screen capture of this dialog, which has four controls. The first, labeled ***Group***, is a drop down list of all your current group names. ***Path***, which is read-only, gives the actual path location for the new group. The two final controls are for: (1) a new group name of your choice (this is solely for the Texture Library), and (2) any descriptive information or user notes you want to include with the group. If you don't enter a group name, your new group will bear the same name as the directory you've selected.

Once you select a directory and give it a group name, the Texture Library will scan all directory folders residing one or more levels below the group, looking for DarkTree files. DarkTree file endings now include ***dstc***, ***dstb***, ***dstp*** and ***dsts***, as well as the original ***dst***. Next the Texture Library will check each DarkTree file and assign a color-coded icon, based on type. Non-DarkTree-related files will be filtered out. When you're ready, click on the ***Apply*** button to close the *Edit Groups* dialog. (We'll discuss icon color-coding under *"Working with the DarkTree Folders & Files" on page 29.)*

If a group includes folders, which will usually be the case, those will be listed below the group name, once you've closed the dialog. If a folder doesn't contain any DarkTrees, the folder name will still be listed in the Library, but you won't be able to open it. Be careful not to delete such folders however, because you'll permanently lose whatever hidden files they may contain.

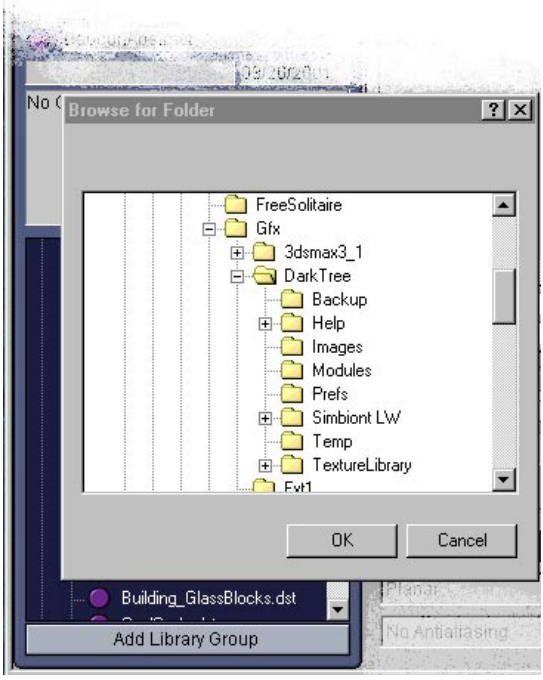


Figure 2.3
Click on the Browse for Folder dialog when you want to add a new group.

Deleting a Group

Deleting a group is simple and does not actually delete any of your files. Just right-click once on the group name to open the *group menu*, then select ***Delete***.

Editing Group Properties

If you want to add-to or change an existing group description or rename a group, you can easily do so by re-opening the *Edit Groups* dialog. Just right-click on a group name to open the popup *group menu*, then select ***Properties***. This will automatically open the *Edit Groups* dialog. Now you can edit the properties of any or all current groups by first selecting the group name from the ***Group*** drop down list, then making your changes in the ***Name*** and ***Description*** boxes.

Refreshing Groups and Folders

If you update or rearrange folders or DarkTree files outside of DarkTree Textures, the Texture Library won't automatically be able to detect those changes. But you can use the ***Refresh*** option to keep your file organization up to date. Just right-click once on a group or folder name to open the popup *group* or *folder menu*, then select ***Refresh***. The Library will scan everything below the group or folder you've selected as a starting point.

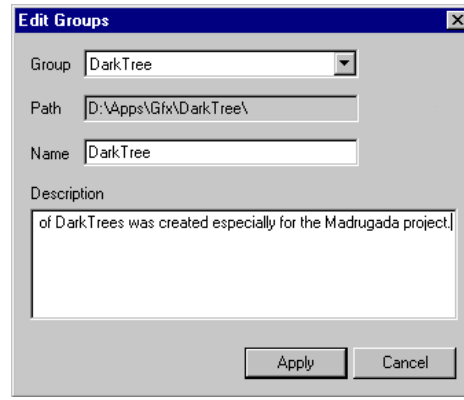


Figure 2.4

Edit Groups is the second dialog to open when you want to add a new group.

Working with the DarkTree Folders & Files

As with the Library groups, you'll want to perform file management operations on folders and DarkTree files. The remaining discussion for the Texture Library covers file and folder management capabilities.

Right-click once on a folder or file name to open the appropriate popup menu. These two menus (folder and file) have some special file-management options. *Figure 2.5* is a screen capture of the *folder menu,* and *Figure 2.7* shows the file menu. Each has options especially suited to its type.

First, though, here is a brief explanation of the color-coding for DarkTree file icons. A file icon precedes each DarkTree file name. By its color an icon announces the type of its associated DarkTree file. You don't have to preview a DarkTree file to learn its type.

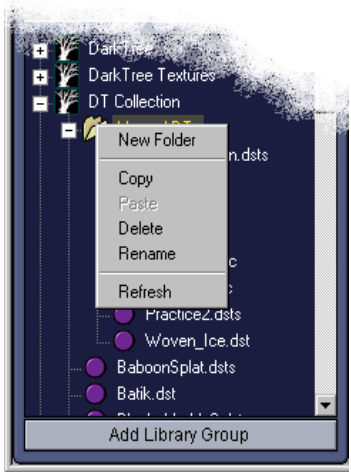


Figure 2.5
Example of a popup folder menu.

File Icon Color-Coding

- **green icon** – *The DarkTree is color-only.*
- **blue icon** - *The DarkTree is bump-only.*
- **gray icon** – *The DarkTree is percent-only. Percent DarkTrees are used for many special attributes, such as luminosity.*
- **purple icon** – *The DarkTree is fully-shaded. A fully-shaded DarkTree always has a DarkTree Shader component in the root position of the tree structure. DarkTree Shakers combine many attributes in a single DarkTree.*
- **question mark icon** - *The DarkTree file has been corrupted in some way.*

Moving Files & Folders

You can move files to another location by first selecting them, then dragging them to the new position. You can move folders in the same way but only one at a time. If you are moving a folder, the move will include all files within it as well. The new location can be another folder or even another group. Keep in mind that moving files and folders in the Texture Library will result in permanent changes to your file system.

You can move one file or a group of files in a single operation. The files can be consecutive or randomly located. Choose a consecutive group of files by holding down the ***Shift*** key while selecting the first and last file in the group. Holding down the ***Ctrl*** key allows you to select multiple files in the same way, but they need not be consecutive. File(s) or a single folder becomes highlighted when selected. The final step is to hold down the left mouse button and just drag the lot to the new location. Always drag multiple files by clicking on the selected file that has a box around it (called the caret item). Normally the box will enclose the last file of a group selection. If you try to drag a group of files by clicking on one with no box around it, you'll just

undo the group selection and end up with a single file selected. You'll know the move is successful after you've released the mouse button (dropped the files) on top of the highlighted target folder or group name.

NOTE: The Texture Library does not support use of the control key to copy one or more files while you're dragging them to a new location. This is one deviation from Windows Explorer® capabilities.

Deleting Files & Folders

It's easy to delete files and/or folders in the Texture Library but be careful because these deletions are permanent! You can delete by simply selecting a DarkTree file (or folder), then hitting the ***Delete*** key. Or you can open the popup *folder* or *file menu* and choose the ***Delete*** option. If you select a folder for deletion, any files in the folder (DarkTree files plus filtered files) will be deleted as well. You'll always get a warning message that you must respond to, before the Texture Library will proceed. Still, the safest way to remove unwanted files is to always delete one file at a time. Delete folders only after they are empty.

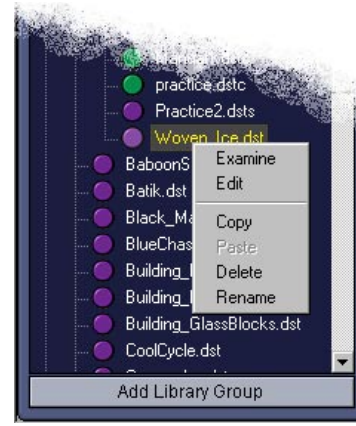


Figure 2.7
Example of a popup file menu.

Copying & Pasting DarkTree Files & Folders

Use the *file menu* for copy and paste operations on files. You can apply copy/paste to a single file, a group of files, or a folder-full of files. You can copy/paste just one folder at a time. Open the popup *folder menu* for folders. If you apply copy/paste to a folder full of files, the folder will be copied and pasted as well as all files within it. And whatever you select, the file(s) you want to copy or the target destination, it will always be highlighted with the color yellow.

With copy/paste operations, be sure to observe this precaution. You can only paste a file or folder that still exists! If you copy a file, delete the original, and then try to paste your copy, you'll be out of luck. The copy disappears when you delete the file. You will get a warning when you try to delete a copied file or folder, informing you of the consequences if you continue. That means you'll have to specifically tell the Library you want to proceed with the delete operation.

To begin, select the DarkTree file(s) or folder you want to copy. For a file group selection, follow the standard method described under *"Moving Files & Folders"* Next, right-click once on the caret item (the file name outlined with a box) to open the popup *file menu*. Choose the ***Copy*** option. To paste your copied file(s) or folder, first

right-click once on the target group or folder name to open the appropriate popup menu, and to show the Texture Library where you want the file(s) pasted. Finally, select ***Paste*** from the menu list. It isn't necessary to open the folder (or group) before pasting the copied items.

Renaming Files & Folders

The easiest way to rename a DarkTree file or a folder is to position the cursor on the current name and double-click slowly, using the left mouse button (pretty standard stuff). If you've clicked slowly enough, a box will surround the name. (Recall that a faster left double-click opens a DarkTree Editor.) Next, position the cursor within the box and type in your new name. This overwrites the original name. Move the cursor outside of the box and left click once or hit the ***Enter*** key to "set" the new name. An alternate method of renaming a file or folder is to right-click one time on the name you want to change, then select ***Rename*** from the ensuing popup *folder* or *file menu*.

If you change the file extension as well, you'll get a standard warning when you click outside the rename box or hit ***Return*** to "set" the new name. If you change the file extension to anything other than a DarkTree extension, the file will be filtered from the Texture Library list. And, furthermore, the operating system will think it's something it isn't and not know how to handle it.

Creating New Folders

To create new folders, right-click once on a group or folder name. Since the new folder will be placed one level below the name you clicked on, you'll want to open a *group menu* when the new folder is to be on the next level down. It doesn't matter whether the group or folder you select is open or not. When the *group* or *folder menu* appears, select ***New Folder***. By default, the first new folder will be named simply ***New Folder***. For multiple folders, the naming sequence is: ***New Folder(2)***, ***New Folder(3)***, and so on. After you create the number of folders you need, you can give them more appropriate names by selecting ***Rename*** from the same menu.

Section 2.2
The Examine Window

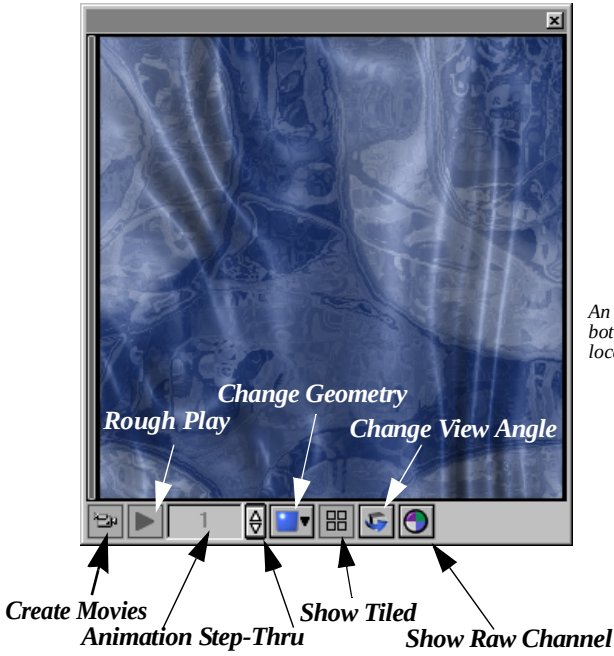


Figure 2.8
An Examine Window that has the bottom edge controls named and located for you.

2.2.1 Examine Window Overview

This section covers the Examine Windows, the second of the four major parts that make up DarkTree Textures. Examine Windows are intended to function in conjunction with the DarkTree Editor, as an aide to the DarkTree construction process. As you experiment with building your own textures, you'll want to keep at least one, and possibly more, Examine Windows open at the same time.

The usefulness of an Examine Window comes, for one thing, from the large number of viewing options it offers, and for another from the fact that you can resize it, keeping it small (so it's not in the way) to making it super-large (so you can't possibly miss anything). See *Figure 2.8* for a screen shot of an Examine Window with the icons along the bottom identified. In an Examine Window, you can:

- *Create AVI and/or Quicktime movies from your animated DarkTrees.*
- *View a quick and dirty rough render of an animated DarkTree.*
- *Step through an animated DarkTree, forward and backward.*

- *View your shader on different geometric shapes.*
- *Display a DarkTree as tiled, in a 2x2 grid.*
- *Reset the pitch and heading (for 3D geometry), so that you can view all sides of your DarkTree.*
- *View one raw DarkTree channel only, or all channels shaded.*
- *Use multiple Examine Windows to compare and contrast two or more DarkTrees.*
- *Save an image you like in a multitude of formats.*

In this section, we'll be covering each of the above listed options plus some others.

2.2.2 Examine Window Basics & Miscellany

This subsection covers basic Examine Window operations, as well as some other information that may prove to be of use.

Opening An Examine Window

You can open an Examine Window from two places. First, you can click on the ***Examine*** button beside the *preview window* in the Texture Library. (See a discussion of the preview window in *"Previewing a DarkTree" on page 24*.) When you open an Examine window from the Library, it will automatically render whichever texture is currently displayed in the *preview window.* If the *preview window* is empty, the Examine window will render a default blue sphere.

Second, an Examine Window may be opened from any popup *component menu* (just click on the ***Examine*** option). *Component menus* are accessible by right-clicking once on almost any component on the DarkTree Editor *workspace*. The Examine Window will open with a display of the image that's visible on the component. See *"Edit Options for Components and Wire Links" on page 67* for more information on *component menus*.

NOTE: You won't find the selection that opens an Examine Window on the component menus of either generator components or instances of the Time component.

NOTE2: You may open as many copies of the Examine Window as your system resources will allow.

Closing an Examine Window

Click once on the small "**x**" in the upper right-hand corner of the window. Closing the whole program will also close any open Examine Windows.

Loading a Single Channel or New DarkTree

With DarkTree Textures 2.0, an empty Examine Window automatically loads a greatly-expanded default shader component, the DarkTree Shader, all of whose channels are empty initially. The DarkTree Shader provides links for eighteen separate channels. All of these channels are based on the three key DarkTree types: color, percent, and bump. For example, you could link a percent-type component, subtree, or DarkTree to one of the percent shader channels and create glossiness, specular highlights, or a number of other interesting effects.

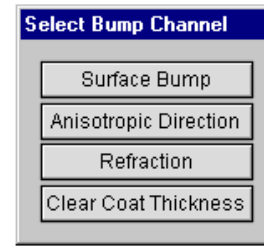


Figure 2.9

This popup dialog lets you select a bump-type channel to

Once you've opened an Examine Window, you can drag and drop a new DarkTree, subtree, or almost any single component from the Editor *workspace*. (Be sure to put the cursor on the component *image* before you drag.) In addition, you can drag and drop any file straight from the Texture Library list. If you load an unshaded DarkTree, subtree, or single component, a dialog will pop open with a list of possible compatible channels that it could be linked to. On the other hand if you drop a completely different shaded DarkTree onto the Examine Window, it's the same as opening a new Examine Window but with any changes in the settings that you've added. *Figure 2.9* is a screen capture of the dialog for a bump load.

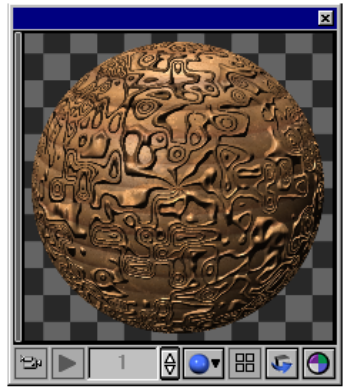


Figure 2.10

A shaded DarkTree overloaded with the bump-type channel from an unrelated DarkTree. This doesn't change the DarkTree in any way.

Figure 2.10 is a screen capture of a shader that has had its bump channel loaded with a completely unrelated bump-type subtree. This is the type of display you might get if you replace just one channel, rather than the entire shader.

NOTE: To drag a new component from the workspace to an Examine Window, be sure to drag from the component's image area, otherwise you'll move the component instead.

NOTE2: You cannot drag a texture from the Texture Library's preview window or the DarkTree Editor's Component Bin directly to an Examine window.

Resizing an Examine Window

You can resize an Examine Window in two ways. First, the window has four fixed sizes that are accessible under the ***View Size*** submenu on the popup *examine menu*. We'll discuss this choice more when we cover the *examine menu* options in the next subsection.

The second way employs standard window resizing. This just means that you can grab any window edge or corner with the cursor and stretch or shrink the whole window to the desired size.

Rendering and the Render Progress Meter

When an image is rendered, an Examine Window makes two distinct passes. The first pass performs a coarse rendering so that you'll have a quick and rough view of your image, and sometimes that's enough. The second pass is a pixel-level rendering. If the ***Antialias*** option is turned on, the window will perform a third, refining pass. You'll find the ***Antialias*** option on the *examine menu*, which we'll cover in the next subsection.

The *render progress meter* is a narrow green line that travels down the Examine Window's left edge and helps you keep track of rendering progress. Beginning at the top, the meter makes one complete pass for each rendering level, then turns gray when the job is completed. You can halt the rendering process by pressing your keyboard ***Esc*** key. If you do, the line will stop where it is and turn red, remaining that color until you begin rendering another texture.

Background Display

The background display is a checkerboard pattern rendered in two shades of gray. The pattern will be visible from behind DarkTrees that are partially transparent. You can use the *Global Preferences* dialog to change the colors of the checkerboard. *Appendix B* covers how to change the *Global Preferences.*

2.2.3 The Popup Examine Menu

Figure 2.12
The Examine Window's popup menu.

Overview

A single right-click somewhere on the image area of any Examine Window will bring up the *examine menu*. This menu has a list of options that deal mainly with the Examine Window itself, rather than the currently loaded shader. *Figure 2.12* is a screen capture of this small menu.

View Size

The ***View Size*** submenu lets you choose among three fixed window sizes: ***Small***, ***Medium***, ***Large*** and ***Custom***. But remember, you can resize an Examine Window by dragging an edge or corner as well. The smallest window size is especially useful

because you can line several Examine Windows up to compare/contrast images, and they still won't take up much room. However, the smallest window size can't display the option buttons, for obvious reasons.

NOTE: The default opening size for Examine Windows is medium, although you can reset this from the main application window, under ***Tools > Properties.***

Always on Top

Selecting ***Always on Top*** guarantees that an Examine Window will stay in front of all other windows that you open, with some exceptions. When short term dialog or editing windows pop open, the Examine Window will get lost until you have clicked the ***OK*** button. ***Always on Top*** is a toggle, and selecting or enabling it will put a check mark on the menu, next to the option. If ***Always on Top*** is disabled, the next window you open will cover your Examine Window, leaving it in limbo somewhere between the main application window and the last window you selected.

NOTE: Always on Top is enabled by default.

Save Image

Even though DarkTree Textures is not primarily a 2D package, from time to time you will most certainly create some special images that you'd like to save, just as they appear on the Examine Window. Selecting ***Save Image*** lets you do just that. Such files may not be of any use in a modeling/rendering applitcation, but they do make, for example, excellent wallpapers and web page designs. With this option you can save your special images in a large number of formats. *"Appendix A: Valid Formats for DarkTree Textures" on page 221* has a list of the formats available for Examine Window image files. Selecting ***Save Image*** calls up a standard file requester that lets you choose a destination and a name for your image files.

To ensure that an image looks its best, you might want to select ***Antialias*** (discussed next) before you start the process of writing to an image file. Some DarkTrees look much better when ***Antialias*** is enabled, but with other designs it doesn't make much difference. One caution should be noted. Be sure you wait until the Examine Window has finished its final render before you select ***Save Image.*** If you do not, you could end up with an only partially polished image, which is definitely not what you want. Count three rendering passes if you have enabled the antialiasing pass, and two if you haven't.

NOTE: For high-quality output, use the Bitmap Renderer to render and save DarkTree images.

Antialias

Enabling ***Antialias*** greatly enhances the rendered quality of most images. Complex images with small patterns generally benefit more than simpler ones. The antialiasing pass tends to slow the rendering time significantly (about 4/5 of rendering time is spent on this pass), because the Examine Window must sample five discrete points for each pixel. When enabled, you'll see a check mark next to the ***Antialias*** selection.

For more information on the rendering process, turn the pages back to "*Rendering and the Render Progress Meter*"

NOTE: Antialias is disabled by default for an obvious reason.

2.2.4 The Examine Window Viewing Controls

Overview

The DarkTree viewing controls (mostly buttons) are laid out horizontally across the lower edge of the Examine Windows. These controls give you just about any useful capability you could want for viewing static or animated DarkTrees. Our discussion begins with the left most control, covering each in turn as we move to the right, until we've covered all seven. Look back at *Figure 2.8* for a screen capture of an Examine Window that points out the location of each control.

Create Movies

Clicking on the *create movies* button begins the two-part process of making either an AVI or a QuickTime movie. This gives you the chance to view your animated DarkTree in real time, rather than simply stepping through frames one at a time. Movies created from the Examine Window are meant as a preview, not as a final animation. For a final-quality movie, render your animation frames in the DarkTree Bitmap Renderer. Or if you're using a Simbiont plug-in, you can simply import animations as DarkTree files.

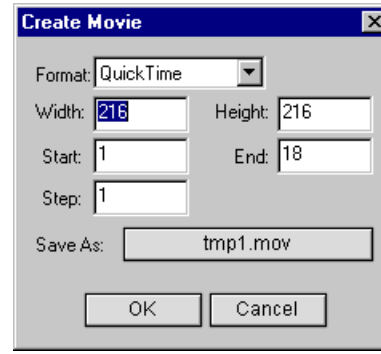


Figure 2.13
The Create Movie dialog lets you select one of two movie formats.

Create movies is only accessible if the Examine Window detects that your current DarkTree is animated. Otherwise all of the animation controls will be grayed-out. You can make a movie with any other of the Examine Window controls enabled as well. For instance, if the Examine Window is set to show just a single raw channel, your movie will be created using just that one raw channel.

Figure 2.13 is a screen capture of the ***Create Movie*** dialog window that opens after you've clicked on the *create movies* button. The following paragraphs explain each of the settings on this dialog.

The Create Movie Dialog

Format: This is a drop down list of movie formats. You can make movies with either AVI or QuickTime. AVI is a part of the Microsoft® Windows operating system, so you'll no doubt have it. QuickTime is an Apple Computer Inc. program and requires a separate installation (available for download at www.apple.com).

Format always defaults to the one you used for your last movie.

Width & Height: These values, which are in pixels, dictate the initial size settings for the movie. The initial or default settings are determined by the size of the current Examine Window. This will hold true each time you click on *create movies.* Hint: To speed up the movie-making process, select a smaller size here, say 100 by 100 pixels.

Start & Stop: Initially these are the start and end frame numbers for the current animated DarkTree. You can, however, reset the frame numbers to a subset of the total. Just be sure that the start frame is less than the end frame and that both are within the range of the current DarkTree settings.

Step: ***Step*** gives the number of frames between those you actually want in your animation. For instance, if you enter *2* for ***Step*** then every other frame will be rendered for the movie. Entering a larger step than *1* speeds up the animation process, often significantly. The default is *1*.

Filename: The ***Filename*** edit box gives you a chance to enter a file name other than the default one for your movie. AVI and QuickTime files always default to the DarkTree ***temp*** directory. When you finally close DarkTree Textures then reopen the program at some later time, any existing movie files are overwritten; so if you wish to save a movie file, give it a unique name to protect it.

The Video Compression Dialog

Clicking ***OK*** on *Create Movie* opens the second dialog window, *Video Compression*. The AVI and QuickTime versions of this dialog are somewhat different, but both are for setting the video compression for your movie. *Figure 2.13* is a screen capture showing these two dialog windows. The list of settings that you'll choose from are

those that are actually available on your system. Read through the AVI or QuickTime documentation if you need help understanding the differences among the compression codecs.

Clicking ***OK*** on the *Video Compression* dialog begins the movie creation process.

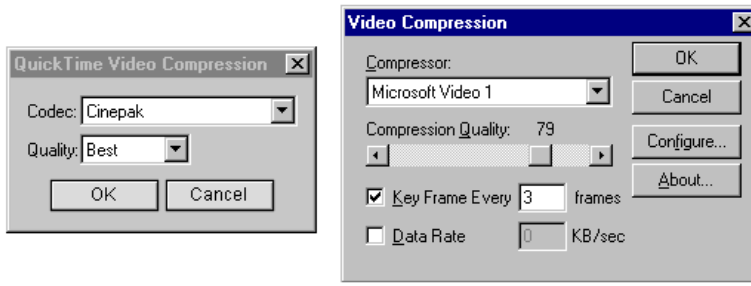


Figure 2.14

Screen captures of the QuickTime ® and AVI ® movie Video Compression format dialogs. Select one of these first before processing a DarkTree.

A Few Final Comments

After you begin the movie making process and upon the completion of each frame, you'll see it displayed in the Examine Window. If the frames are larger than the Examine Window, they'll be scaled to fit. (The resizing of the movie frames for the Examine Window won't affect the final ***Width*** and ***Height*** you set in the first dialog.) The *animation edit box* will keep track of which frame is currently being rendered (NOT the one currently visible but the next one). You can click on the ***Stop*** button at any time to interrupt the process. Any finished frames will still be saved for the movie.

Upon the movie's completion, the Windows Media Player ® or the QuickTime Player ® will open so that you can run the complete movie. The player will open with the frames it has, even if you interrupted the process.

Rough Play

The *rough play* button steps quickly through each frame of an animated DarkTree. To keep render times short, each frame is given just one rendering pass so you'll often see jagged edges.

A single click on the *rough play* button starts the animation, which proceeds from the current frame to the end frame and back to the current one. "Current frame" refers to the frame number specified on the ***Animation Step-Thru*** control. You can enter a different frame number at any time in the *animation edit box*. Just be sure to hit the ***Enter*** key to "set" the new frame number.

If you want to start *rough play* from a small Examine Window that doesn't display the controls along the lower edge, you can still do it by using the "***a***" key on your keyboard. If you want to stop the animation, hit ***Esc***. Enabling ***Antialiasing*** from the popup *examine menu* will not affect *rough play.* However, if you have specified a single raw channel render from the right most control, *show raw channels*, *rough play* will run through the animation in that mode.

NOTE: You can use the "a" key on your keyboard to initiate a rough animation as well.

Animation Step-Thru

The *animation step-thru* control lets you view the frames of your animation, one at a time. Unlike *rough play,* each frame is given two rendering passes by default. And if ***Antialias*** is enabled (from the popup *examine menu*) each frame will be given the third and most meticulous rendering pass as well.

The initial number in the *animation edit box* will be the representative frame for the animation (usually *1*), but you can enter any frame number you wish. Always press the ***Enter*** key to "set" a new frame number. Use the *zip slider* to move forward or backward through the animation quickly.

If your Examine Window is small and doesn't display the lower edge controls, use the keyboard arrow keys to move through the animation. The keyboard arrow keys will jump to the following frames:

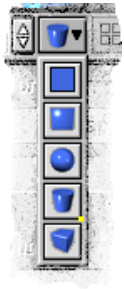
(arrow right) Advance one frame.

(arrow left) Go back one frame.

(up arrow) Advance to the end frame.

(down arrow) Return to the start frame.

Change Geometry



When you have a finished DarkTree targeted for a special model, you might want to view it on geometry that approximates the shape of your model before moving it to your modeling package. Or, you might simply prefer, for example, a cubic to a planar display. *Change geometry* is a visual drop down list of geometric shapes that you can apply your current DarkTree to. You can see a screen capture of this list directly to the left. DarkTree Textures supports the following mapping choices: *planar* (generally the default), *spherical*, *cylindrical, cubic*, and *frame*. All of these geometric shapes are three-dimensional, except for *frame* which is a

specialized two-dimensional mapping. Just left-click once on a shape to select it. Once you've made your selection, any subsequent DarkTree(s) you load into the same Examine Window will use the new geometry. When you open a new Examine Window, it will use the current shader's preferred mapping. That is, the one you set specifically on the *Properties page* or your default mapping.

Show Tiled

Clicking on the *show tiled* button displays the currently loaded DarkTree in a four-tile grid. Before you can use this control, you must set the mapping geometry to *frame*. *Frame* is a special two-dimensional geometry used expressly for tiling and video. The *show tiled* button will be locked if the geometry is set to anything else.

If one of the tiling components is a part of your DarkTree structure, *show tiled* can give you a good idea of what your texture will look like when it's repeated. You can use *show tile* with any DarkTree, but unless you have set the texture up specifically to be a tiled surface, it won't tile seamlessly.

Change View Angle

Change view angle works with any of the 3D mappings. If you have the current mapping set to *frame*, this button will be locked and unavailable until you change the Examine Window geometry to a 3D mapping.

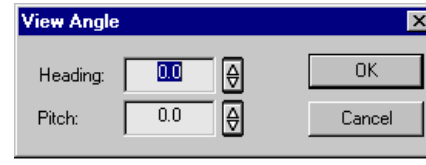


Figure 2.16
Dialog for changing mapping mode view angles.

Change geo angle opens a dialog (seen in *Figure 2.16*) that lets you enter a new ***Heading*** and/or ***Pitch***. Either type in a new angle or use the *zip slider.* Angles you enter are absolute. That is, all rotations are based off of the original object position. The Examine Window doesn't place restrictions on the values you enter for the angles. But for large entries the angle returns to its starting position every *360* degrees. ***Heading*** values turn the geometry left (positive values) or right (negative values), and ***Pitch*** values will tip the geometry either up (positive values) or down (negative values). Heading adjustments use the world Y-axis for rotations. Pitch is different in that it uses the model's X-axis to rotate about.

Keep in mind that planar geometry, while considered 3D by the Examine Window, has an infinitely thin edge (Z-axis depth). If you were to enter, for instance, a heading of *90* degrees, you would get an invisible edge (see *Figure 2.17* to clarify this).

NOTE: Cylindrical and cubic geometries are placed at an angle by default.

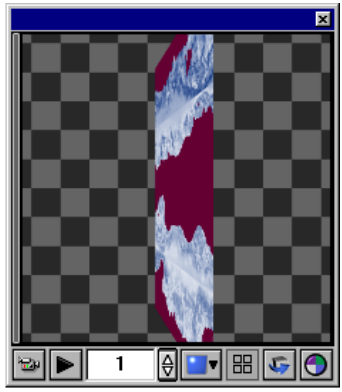


Figure 2.17
A planar mapping rotated by 80 degrees (heading). Notice how much the angle has changed the view? At 90 degrees you wouldn't be able to see the texture at all.

Show Raw Channels

A channel is defined as a displayable value or set of values that makes up a surface attribute, such as reflectivity or surface color. The DarkTree Shader accepts a maximum of eighteen channel links. Therefore, an Examine Window can render a maximum of eighteen channels at one time. Each channel accepts one of the three types of DarkTree: color, percent, or bump. This just means that the ***Surface Color*** and ***Environment*** channels take a color DarkTree, and so on. The next and final subject, *"Notes on the DarkTree Shader Channels"*, details the function of each channel.

The *show raw channels* control is designed to segregate one linked channel from the currently loaded DarkTree Shader, so that you can view it separately. The single channel is rendered "raw"; that is, it's an unshaded render. As an example of how this can affect the look of the mapping geometry, if you have spherical mapping set the rendered channel will look like a flat disk. The *type icon* located in front of each channel name denotes the link type (see the next note). And finally, there'll always be a check mark before the currently selected channel.

The *show raw channels* button opens a drop down list of all available shader channels. The channels that are actually linked to components or subtrees will have their names in a bolded font. The remaining channels will look grayed-out. For example, looking at *Figure 2.18*, you can see that only **Surface Color** and **Surface Bump** are linked in

this particular shader. You'll only rarely find many channels linked in a single shader, but many link choices gives you the freedom you need to create competitive special effects.

NOTE: Type icons are small colored circles that designate component, DarkTree, link, or parameter type by their color. A green type icon designates color, a blue one designates bump, and a gray one designates percent. A fourth icon color, purple, designates a shaded DarkTree.

You can use any other of the controls while the Examine Window is set to a single channel. Simply click on the top choice, **Shaded**, to return to a full shaded display. The following paragraphs give some information on the raw channel displays.

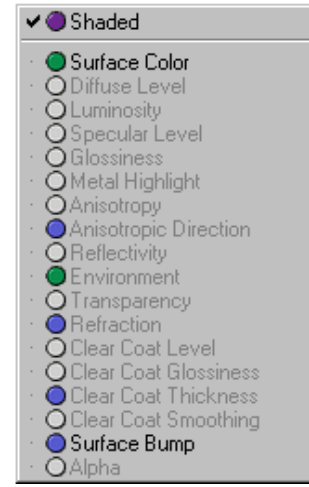


Figure 2.18
The show raw channels drop down list. Notice that the loaded shader has just two linked channels.

Raw Color & Percent Channel Displays

These two channel types will look just like their linked subroot component images but without the shading and lighting found on the images.

Raw Bump Channel Display

Displaying a raw bump channel, since it's unshaded, represents elevations as shades of gray. The higher elevations are white and near-white, and the deeper areas are dark gray to black. Your bump channel elevations will be distributed among 256 shades of gray (the maximum number available here).

Displaying your bump channel in this way can be a real eye-opener. That's because you can clearly see whether all levels of elevation are truly represented. For instance, if you have some large jumps in elevation and then some that are quite subtle (that is, varying only slightly in height), you may find that the channel is completely ignoring the small elevation jumps.

2.2.5 Notes on the DarkTree Shader Channels

This final Examine Window subsection lists each shader channel with its DarkTree type, then gives you a brief description of the effect each can contribute. Since every channel is listed with its basic type, the list won't be in the same order as in the *show raw channels* control list.

Color Channels

Surface Color - *This channel is reserved for the basic DarkTree color link.*

Environment - *The Environment channel is meant for an environment map link. For example, if your DarkTree is reflective, you'll want to link a suitable environment map to use for the reflection.*

Percent Channels

Diffuse Level - *The* ***Diffuse Level*** *quantifies how light is either absorbed or reflected by a surface. The lower the value the more light absorption, causing the surface to dim. As the value moves toward 100% the more light the object reflects, causing it to brighten.*

Luminosity - Luminosity *refers to a suffused, self-lit state. The more the value moves toward 100%, the brighter the object appears. Use* ***Luminosity*** *in place of ambient (surrounding) light.*

Specular Level - *The* ***Specular Level*** *determines how intense surface highlights from an exterior light source will be. This attribute works in conjunction with* ***Glossiness****.*

Glossiness - Glossiness *controls how concentrated or spread-out surface highlights will be. The more this value moves toward 100%, the more concentrated the spots will be.* ***Specular Level*** *controls the intensity of the light spots. Bright concentrated spots cause an object to look hard and shiny.*

Metal Highlight - *While non-metallic surfaces reflect the color of the light that shines upon them, metallic surfaces reflect light based on the color of the metal itself. For example, the specular highlights on an object made of gold will be golden in color.*

Anisotropy - *This attribute contributes a directed specular highlight. Because the highlight has direction,* ***Anisotropy*** *is good for brushed metal surfaces, records, and such.*

Reflectivity - *The level of reflectivity is what makes surfaces reflect their surroundings. The closer the value moves toward 100%, the clearer the reflection. Use an environment map in conjunction with this attribute to provide the reflection.*

Transparency - *A strongly transparent surface lets you see what is behind it. A background pattern or a scene with other objects can provide the background view.*

Clear Coat Level - *This attribute is a specular level, but for a surface that has been coated with a material akin to varnish. Thickly clear coated surfaces reflect or absorb light differently from non-coated objects. Use in conjunction with* ***Clear Coat Glossiness.***

Clear Coat Glossiness - Clear Coat Glossiness *sets the glossiness of the clear coat (see* ***Clear Coat Level****). An example of such an effect is metallic car paint.*

Clear Coat Smoothing *- This attribute gives the effect of partially or fully smoothing the Clear Coat on a bumpy surface.*

Alpha - *This channel is a standard masking channel. Values near 0% are transparent.*

Bump Channels

Surface Bump - *This channel is reserved for the basic DarkTree bump link.*

Anisotropic Direction - *Use this attribute to specify a brush direction for brushed (anisotropic) surfaces.*

Refraction *- Using the Index of Refraction, this attribute controls how much a transparent object bends light, giving a distorted view of the background.*

Clear Coat Thickness - *This attribute works with the other Clear Coat parameters. Use it to determine the thickness of the clear coat, with 0.0% representing no thickness. Clear coat thickness slightly changes the location of the clear coat specular highlights. This advanced feature's effects are subtle and dependent on the scale of your 3D scene.*

Section 2.3
The DarkTree Editor

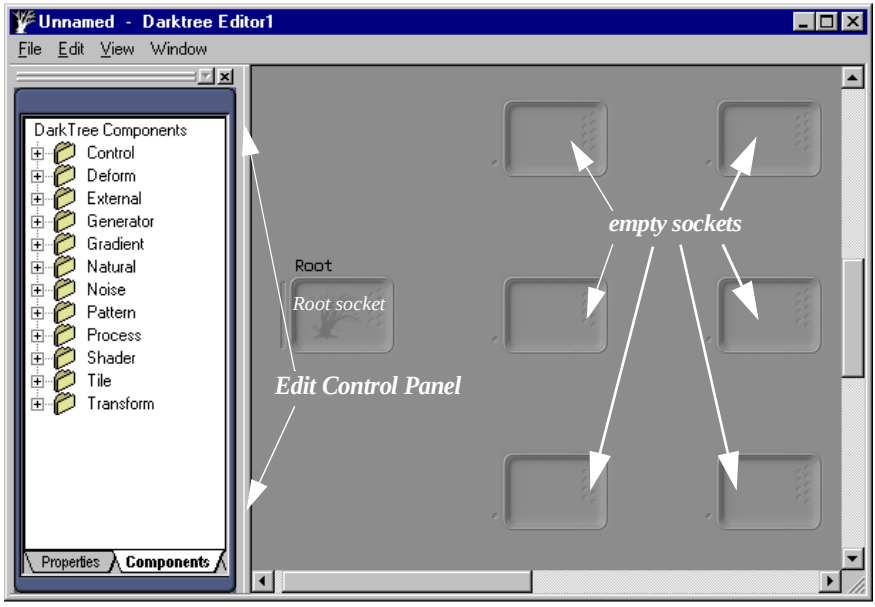


Figure 2.20

An empty DarkTree Editor showing the sockets where the components will be plugged in, the Root socket, which determines the DarkTree type and the Edit Control Panel. You can access the Component Bin, Properties page and Tweaks Editor from the tabs on the bottom of the panel.

2.3.1 DarkTree Editor Overview

The DarkTree Editor is the workhorse and the heart of the texture creation process. Here, you can design, build and display the structure of your DarkTrees. At the end of this section, we've provided some basic tutorials. Even though it's fairly simple to learn to build DarkTrees just from them, these sections discuss the interface extremely thoroughly. If you have additional questions after finishing the tutorials, the answer is here.

Since this is the most complex DarkTree Textures section, we've divided it into four pieces (not including the tutorials at the end of the section).

Piece 1: The subsections titled *"The DarkTree Editor Workspace"* through *"Parts of a Component Face"* have as their focus the task of familiarizing you with the Editor work areas. Specific subjects include: the *workspace* and socket grid, *workspace* ***View*** options, *Edit Control Panel*, and the breakdown of a component into its basic visible parts.

Piece 2: The next two subsections, *"The Basics of Tree Building"* and *"Edit Options for Components and Wire Links"* explain how to work with the components: how to plug them into the *workspace* sockets, how to connect them with wire links, and some editing options that affect them. After reading through these subsections, you'll have the knowledge to plug components into the *workspace* and link them into a DarkTree.

Piece 3: The third piece covers *"The Component Editors"* and *"The Generator Editors - Special Features"*. After completing this piece you'll understand the *Component Editors* and be able to apply transforms, change parameters, and recognize the differences between the regular *Editors* and the *Generator Editors*.

Piece 4: This final piece covers three utilities that are new with DarkTree Textures, version 2.0: the first subsection titled *"The Spline Editor"* covers the *Spline Editor*, *"Tweaks Power! The Tweaks Editor Page"* explains how to set up *Tweaks*, and *"The DarkTree Color Browser"* covers the new DarkTree color editor.

And to finish up, *"The DarkTree Editor Tutorials"* on *page 108* is a set of three tutorials that teach you how to use the Editor. We've included lots of information in these tutorials. But they're a relatively painless way to learn new software, and they're kind of fun too. We've also included a glossary at the end of the manual that has definitions for terms peculiar to DarkTree Textures and especially the DarkTree Editor, as well as some other terms you might not know.

So now let's move on to the first piece, beginning with the two easiest Editor tasks.

Opening and Closing the DarkTree Editor

Opening

You can either open an empty Editor or one that loads a DarkTree as part of the opening process.

You're only given one way to open an empty DarkTree Editor. Go to the *main application menu* and click on ***File>New***. Open an empty Editor when you want to begin building a DarkTree from scratch.

You're given three methods of opening an Editor that's preloaded with a DarkTree. All begin with the DarkTree selection. After making your choice, the DarkTree Editor will open with a display of the structure of the DarkTree you selected.

1. From the Texture Library list, click once on a DarkTree file to load it in the preview window. Next click on the Library's ***Edit*** button.

2. Left double-click on a DarkTree file in the Texture Library. The DarkTree Editor will open with the DarkTree you selected.

3. Click ***File>Open*** on the *main application menu*. Using a standard file requester, select the DarkTree you want to load onto an Editor. DarkTree files will have one of the following extensions: *dsts, dstc, dstp,* or *dstb*.

NOTE 1: The first time you bring up a DarkTree Editor window, it might seem rather small. However, after you have expanded it to a size you find convenient to work with, go to the Editor menu and click on ***Window>Save Layout****. You'll only need to do this one time, as* ***Save Layout*** *is persistent, even after you close DarkTree.*

NOTE 2: You can open as many instances of the DarkTree Editor as your system resources will allow.

Closing the DarkTree Editor

Close the Editor by choosing ***Close*** under ***File*** on the *Editor menu*. Alternatively, you can click on the **X** in the upper right corner of the Editor window.

2.3.2 The DarkTree Editor Workspace

What is the Workspace?

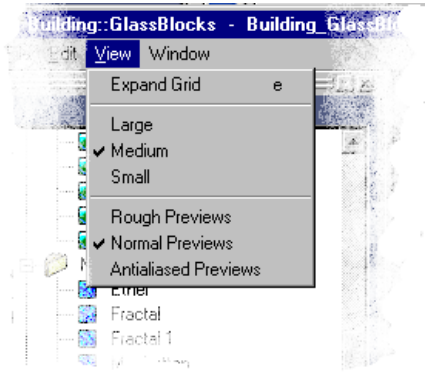


Figure 2.21
Viewing options for customizing the Editor workspace.

The screen capture at the beginning of the DarkTree Editor section, *Figure 2.20*, shows an empty Editor. Most of the Editor area is taken up by the *workspace*, a gray grid of depressed squares. The squares are referred to as "sockets", and you will plug components into them as you build up your DarkTrees. Actually, the word "plug" is merely used throughout the manual in a conceptual sense. In fact, you're simply dropping a component onto a socket. The root socket is by itself in the leftmost column. With the label **Root** just above the socket and the DarkTree

icon within, it's hard to miss. The root component, that is, the component you plug into the **Root** socket, will be a visual combination of all components linked to it. The root component's type determines the type of the DarkTree (color, percent, bump or shader).

We can define each column to the right of the **Root** socket as a level. For example, level *2* components are those in the column of sockets just to the right of the root component (the only level *1* socket). Level *3* is the next column to the right of level *2*, and so on. Some explanations in the manual refer to these levels, as a way of describing a particular position.

You can grab onto the *workspace* background (the plain gray area) with the left mouse button, then use the mouse to move the whole *workspace* up, down, or to either side. That way you can get a better view of your tree structure, especially if it's very large. Notice that the cursor changes to four directional arrows while you're actually in the process of dragging.

NOTE: Your keyboard Home key returns the Editor workspace to its opening position at any time.

Workspace View Options

The next three paragraphs cover a group of viewing options that affect the DarkTree Editor *workspace*. You'll find these selections on the *Editor menu*, under the ***View*** dropdown. *Figure 2.21* is a screen capture of this menu with the drop down open. We'll discuss each of the selections in the order that they're listed.

Temporarily Expanding The Grid

The first option under ***View*** is ***Expand Grid***. By default the DarkTree Editor maintains a buffer zone of two rows/columns of empty sockets. ***Expand Grid*** will temporarily increase that buffer zone by five additional rows and columns. The purpose for this temporary expansion is to provide extra room while you're editing components, moving a large subtree for instance. As soon as you delete, paste or perform a manual move on a component (all discussed in *"Edit Options for Components and Wire Links" on page 67*), the grid will return to its former size.

The shortcut key that expands the grid is ***Ctrl + e***.

Changing Socket Sizes

The next selection under ***View*** changes the size of the sockets on the *workspace*. Looking again at *Figure 2.21* you'll see the options ***Large***, ***Medium***, and ***Small***. The current socket size will be preceded by a check mark. The larger you make the sockets,

the more easily you can see image details. But larger sockets mean longer rendering times for the tree structure as well as fewer visible sockets on your monitor screen at one time, so your choice should be balanced by your style of working.

NOTE: You can choose your own default global preferences, such as socket size and some other options. This is covered in Appendix B.

Image Previews

The final ***View*** option gives you a choice of render quality for the component images that make up the tree structure. Render quality is determined by the number of samples the Editor takes for each pixel. For ***Rough Previews***, the Editor takes one sample for every four pixels and interpolates between them. While this is fast, the images remain rather fuzzy. ***Normal Previews*** is the default, taking one sample per pixel.You could call it a sort of middle-of-the-road preview choice. You'll find it more than satisfactory under most circumstances, since it's reasonably fast and produces quite nice images. The ***Antialiased Previews*** option produces refined images. In this case, five surrounding samples are taken for each pixel. This option, as you would imagine, makes the render a bit slower.

The Edit Control Panel

You'll find the *Edit Control Panel* directly to the left of the DarkTree Editor *workspace*. This panel has three tabbed pages that provide special support for the Editor. From *Edit Control* you can open (1) the *Properties page* which allows you to enter information about your DarkTree, (2) the *Component Bin*, which stores the components for building your DarkTrees and, (3) the *Tweaks Editor* page where you can create, modify or delete your tweaks. Of these three tabbed pages, we'll discuss two here: the *Properties page* and the *Component Bin.* Information on the *Tweaks Editor* is under *"Tweaks Power! The Tweaks Editor Page" on page 100.*

You can undock the *Edit Control Panel* by dragging the top bar on the panel (see *Figure 2.23*). Once undocked, you can resize and/or move the panel to a more out-of-the-way spot without affecting the DarkTree Editor. The panel will redock/resize itself on either the left or right edge of the DarkTree Editor. Hold down the ***Ctrl*** key to prevent redocking while you're moving the panel around. Selections from the ***Window*** option on the *Editor menu* let you show or hide the panel, or save its layout (position). Saving with ***Window>Save Layout*** not only means that the panel will be placed in the same position next time you open the DarkTree Editor, but also that the same *Component Bin* folders will be open.

The Properties Page

The leftmost tab opens the *Properties page* (See *Figure 2.23*). When you're ready to record descriptive information about a new DarkTree you've designed or set up an animation, click on this tab. The particulars you enter here will stay with the DarkTree file. Any time you preview a DarkTree in the Texture Library, some of this information is displayed along with the image. And each time hereafter that you load the same DarkTree into the DarkTree Editor, the *Properties page* loads the information you previously entered here. You can change the fields on this page any time you wish.

Let's touch on each of the *Properties* fields, even though many of them are self-explanatory.

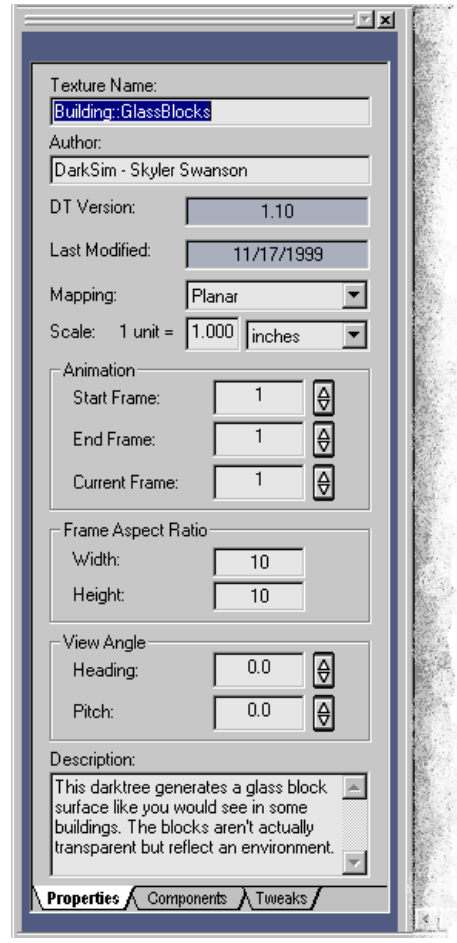


Figure 2.23

A screenshot of the Edit Control Panel open to the Properties page.

Texture Name:

This field is for a working or descriptive name, not the file name with its *dst** ending. When you preview a DarkTree in the Texture Library, it will include this field and some others as well.

Author:

That's you, or whoever designed the DarkTree.

DT Version:

Properties fills in the current DarkTree program version number each time you update (resave) a DarkTree.

Last Modified:

The DarkTree Editor will keep track of this information for you and record the current date each time you update (resave) your DarkTree.

Mapping:

Mapping defines the mapping geometry for the component images but it applies only within DarkTree Textures. DarkTree will use this mapping for the *Component Editor's update window* and the Texture Library's *preview window,* as well as for the DarkTree Editor an default Examine Window.

The default mapping is ***Planar***, unless you've changed the preferences but you can select ***Cylindrical***, ***Spherical***, ***Cubic*** or ***Frame*** as well. All selections, except ***Frame***, are true 3D mappings and that includes ***Planar***. (Think of ***Planar*** mapping as having an infinitely thin Z-axis depth.) If you wish to see your DarkTree tiled in a *Component Editor*, be sure to open this page and select ***Frame*** mapping first. Use ***Frame*** for all other 2D images as well, such as for image processing or banners for web work.

Scale:

Use this option to select a scale that best matches the DarkTree's final scene destination, if you can. The choices are ***inches***, ***feet***, ***miles***, ***millimeters***, ***centimeters***, ***meters*** and ***kilometers***. Modeling/rendering programs typically make you choose a scale for your model. If you set your texture's scale accurately, then when you import it to your modeling package using a Simbiont plug-in, it will fit on your 3D surface accurately.

You can set a default scale that will be persistent over DarkTree closings on the *Global Preferences* dialog. Just go to the *main application menu* and click on ***Tools>Properties***. Setting up your own global preferences is covered in *Appendix B* on *page 222*.

Animation:

This section is reserved for animated DarkTrees. When you change any of the following three fields, you'll see the new value(s) reflected in each Time component's *Component Editor*. You might have set up several Time components, each controlling a different aspect of the animation. But the values entered here are meant for the total maximum length of the animation. If you have a time-based generator, you'll see the values on its function plot change as well. A time-based generator is one that's linked to a Time component through its ***Input*** parameter.

Start Frame: Enter a beginning frame number here. The minimum start frame number is *0*. The default, however, is *1*.

End Frame: This is the final frame number for your animation. (***End Frame*** - (***Start Frame*** + 1)) defines the length of the animated DarkTree sequence. The default ending frame is *30* but the possible upper limit for this field is larger than you'll ever need.

Current Frame: By changing this frame number, you can scan the *workspace* and see exactly what each component does as the animation advances. When you're ready to select a final ***Current Frame***, choose one that can serve as a representative for the entire animation. This frame's image is the one you'll see on component faces when the DarkTree is open. You'll see it when you preview the animation in the Texture Library or display it in the Examine Window as well.

Frame Aspect Ratio:

This pair of fields is reserved for 2D output, and will only take affect when the DarkTree is in the ***Frame*** mapping mode. Changing a DarkTrees's aspect ratio simply changes what it looks like within DarkTree Textures. Its value lies in the fact that (as an example) you can design a DarkTree for video output with a ratio of, say, *720* by *486* pixels and make the DarkTree visually match that final ratio. *Figure 2.24* is a screen capture of a frame-mapped DarkTree image with just such an aspect ratio.

Figure 2.24
Screen capture of a texture in frame mode with an aspect ratio suitable for video production work.

Width: The target width of the DarkTree.

Height: The target height of the DarkTree.

View Angle:

View Angle lets you view your DarkTree from various angles by changing the position of the mapping geometry. The two fields use absolute angles and are specifically reserved for 3D mappings. **View Angle** will reset the angle for any mapping except ***Frame***, which is considered 2D. Keep in mind that if you specify something like a *90* degree heading or pitch for a ***Planar*** mapping, the result will be an infinitely thin line that you won't even be able to see. That's because, although ***Planar*** is a 3D mapping, its Z-axis has no depth.

View Angle values are not constrained in either the positive or negative direction. Each *360.0* degrees will simply return the angle to its beginning position. *Figure 2.25* shows adjusted view angles for both ***Cylindrical*** and ***Cubic*** mappings. **View Angle** controls are the same here as in the Examine Window.

Heading: Entering a new degree rotates the mapping geometry to the left or right.

Pitch: Entering a new degree rotates the mapping geometry up or down.

Figure 2.25
The angles for these two mapping modes are adjusted by default to give you a better view of the geometry.

Description:

Place descriptive information about your DarkTree here. And there's room to get quite chatty!

The Component Bin

The *Component Bin* is the repository for all components provided with DarkTree Textures. And there are a considerable number and variety of these! You can think of the components as the DarkTree building blocks. *Figure 2.26* shows a screen capture of the *Edit Control Panel* opened to the *Component Bin* page.

The components are arranged in classification folders. The classifications indicate qualities that apply to each member. For example, the ***Tile*** class includes every component that can make a DarkTree tile smoothly. Looking again at *Figure 2.26*, you can see that the ***Tile*** folder is open, revealing those components that perform this function. Open the folders by clicking on the little "+" icons.

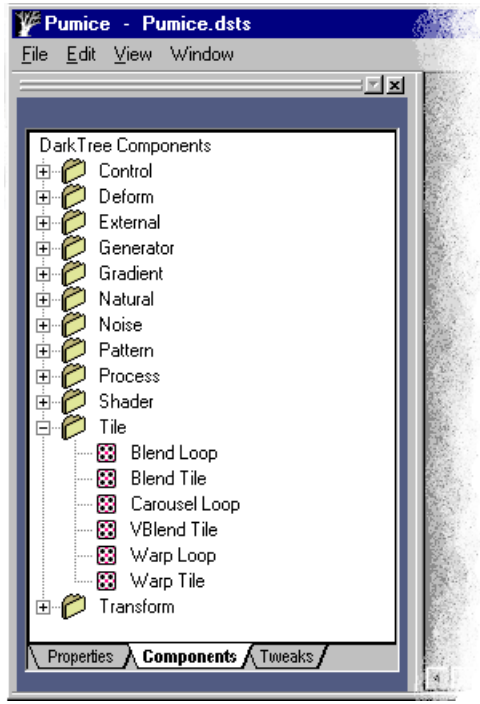


Figure 2.26
A screen capture of the Edit Control Panel open to the Component Bin page.

Components in each classification, with the exception of the *Generator* and *Shader* classes, and the Time and Audio wave components, can be plugged into a socket as either a color-, percent-, or bump-type. The generators and the Time component can only be plugged in as percent types. The DarkTree Shader component has its own separate class, called ***Shader***. The DarkTree Shader component is reserved exclusively for the **Root** position on the DarkTree Editor *workspace*.

For a complete discussion on the component classes, turn to *"The Classifications" on page 151*.

NOTE: To learn how to plug components into the sockets on the DarkTree Editor workspace, read "Plugging Components into the Sockets" on page 62.

NOTE: *You'll find a detailed description of the component classes in Part Three: "The Classifications" on page 151.*

2.3.3 Parts of a Component Face

When a DarkTree structure is laid out on the *workspace*, it's easy to recognize each plugged-in component by its visual representation. This representation incorporates several bits of information about the component, such as its type, the kind of pattern it produces (if it does), and so on. The entire representation is called a "component face". In the center of each component face is an image, the "component image". For example, the Weave component's image is a woven pattern, while the Sine Wave generator has a line graph of the sine wave function.

All component faces look much the same except for their names, icon colors and images. Discrete areas of a component face can give you access to subordinate menus and assist you with the linking process. Check out *Figure 2.27*, a breakdown of a

component face showing the various parts. This subsection explains each part of a component face and how it's useful.

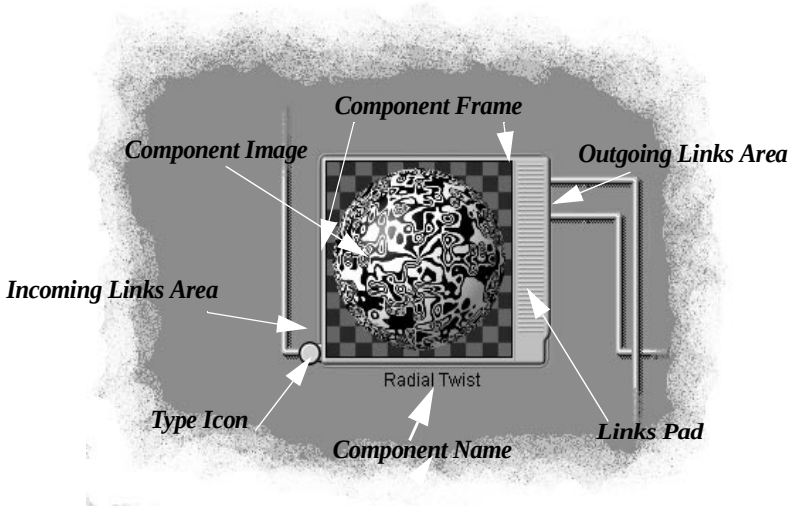


Figure 2.27

A component with labels and arrows pointing to the various parts of the component's "face".

The Component Name

The name of the component is printed here.

The Component Frame

The component frame, which is quite narrow except for the *links pad* section, surrounds the component image just as a picture frame surrounds an oil painting. With the cursor positioned anywhere on the frame, you can:

- *click once with the left mouse button to put the component in the "selected state". (This works with the image as well.)*
- *double-click with the left mouse button to open a Component Editor. (This works with the image as well.)*
- *click once with the left mouse button and hold it down to drag the component to a new socket.*
- *click once with the right mouse button anywhere, except the links pad, to open a customized popup menu. (This works with the image as well.)*

The Component Image

The center area of a component face usually displays a rendered image. What that image looks like is dependent on several factors. See *"Interpreting the Component Images"* which follows next, for a short discussion on this subject. With the cursor positioned on the image, you can:

- *click once with the left mouse button to put the component in the "selected state". (This works with the frame as well.)*
- *double-click with the left mouse button to open a Component Editor. (This works with the image as well.)*
- *drag the image onto an Examine Window to load it.*
- *click once with the right mouse button anywhere, except the links pad, to open a customized popup menu. (This works with the frame as well.)*

NOTE: The background area behind the component image is a two-tone checkerboard pattern. This pattern will always be visible in varying degrees, unless you choose Frame mapping.

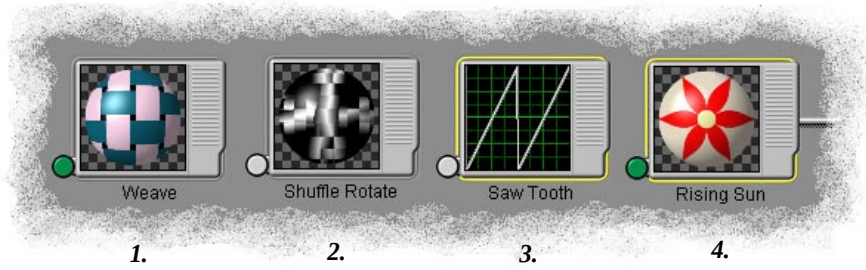


Figure 2.28

Four different types of component images. Read the next section to understand what each type of image means.

Interpreting the Component Images

As was just mentioned, when you select a component from the *Component Bin*, drag it onto the *workspace* and plug it into an empty socket, you'll see a rendered image on the component face. But what that image looks like depends on one of four special conditions. Those conditions are listed next.

1. The image might be the true representation of a single component. Components that can stand alone, that is that produce a specific pattern of their own, are in this category. For example, the Weave component has a woven pattern. The far left component in *Figure 2.28* shows a simple Weave image.

2. Or the component might be from a class that cannot stand alone, such as the Control class. In this case, the image will be a predetermined representative design. Look at the image for Shuffle Rotate (middle left) in *Figure 2.28*. Again if the component is linked to others, the image will be more complex.

3. Or the component might be an instance of Time or a member of the ***Generator*** class. In either case, its image will be a line graph. See the Saw Tooth generator (middle right) in *Figure 2.28* for an example.

4. Or the image might be the result of linking two or more components together. The far right screen capture of *Figure 2.28* shows a simple combination image of a single pattern (supplied by the Rising Sun component) and the S Curve generator. The ***Ray Size*** parameter (from Rising Sun) is linked to the generator. As you can see, the generator function has modified the default "shape" of the sun's rays.

The Links Pad

The *links pad* gives you access to the *links list.* With the cursor located somewhere on the pad, click once with the right mouse button to open the *links list*. This list gives the names of all component parameters that are linkable, and shows their current link status and type.

The Links List

When you click once with the right mouse button on a component's *links pad*, a customized *links list* pops open. This menu lists all of the linkable parameters for its component, including those that are linkable-only. (You won't see linkable-only parameters listed on the component's *Editor*.) You must use the *links list* in order to access the complete parameter list.

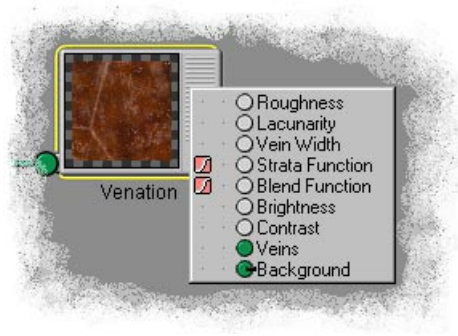


Figure 2.29
A screen capture of the links list for the Venation component.

Look at the *links list* in *Figure 2.29*. The color of the *type icon* (the small circle directly to the left of the parameter name) tells you each parameter's type (color, percent or bump). You'll often see percent parameters in all three component types.)

With DarkTree Textures 2.0, you'll see additional icons associated with some parameters on the *links list*. You'll find a more thorough discussion on the meaning of the new icons in *"Interpreting a Component's Links List" on page 64*.

Area for Outgoing Links

Each wire link has a parent end and a child end. Outgoing links (the parent end) leave from the *links pad* area. Each wire leaving this area represents a link from a parameter of the parent component, and goes directly to a child component. Both the parameter and component must be the same key type (color, percent or bump), because the child component sends data back to the linking parameter along this same wire (theoretically). Here's an example of what the linking does. Suppose you link the ***Bricks*** bump parameter to the bump Rough noise component. Bump Rough noise data is sent back to the parent parameter (***Bricks***). The result is a set of blocks, each individually textured with a non-repeating rough noise pattern. *Figure 2.30* is a visual example that demonstrates this result.

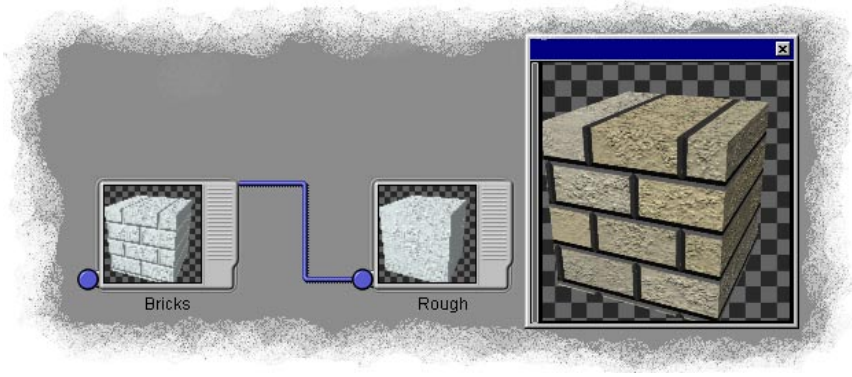


Figure 2.30

An example of how linking a parameter to a child component changes an attribute of the parent component.

Area for Incoming Links

The incoming links area is the child end of a wire link. The end of each wire must attach to a component of the same type as the parent parameter, for reasons explained in the last paragraph.

Theoretically any number of wires can attach to the incoming links area, but a parent parameter can attach to only one child component via one wire.

The Type Icon

You'll find *type icons* displayed within the DarkTree Editor in more than one place; two are within the bounds of this discussion. First, each component face displays a *type icon* for the component as a whole. Note the exact location on *Figure 2.27*. And second, when located beside a parameter name on the *links list* or on a *Component Editor*, the *type icon* indicates parameter type. You'll sometimes see a *shader icon*

(which is a purple type) attached to a component face but never to a parameter name. Components of the Shader class (there's just one component member in this class) are a special type reserved specifically for the **Root** socket. Shader parameters don't exist.

NOTE: The color of the type icon attached to the root component determines the type of the DarkTree.

If you see a *link bar* or *tweaks bar* across an icon, it means the parameter is currently linked or tweaked. Whether indicating the component or parameter type, the color-code for *type icons* is

- *Green for color types.*
- *Gray for percent types.*
- *Blue for bump types*.
- *Purple for shader types (component only)*.

2.3.4 The Basics of Tree Building

Well, okay, if you've read the previous parts, you've learned your way around the Editor *workspace*. The next steps are plugging components into the sockets and linking them into DarkTree structures or trees. You'll find that the *Component Bin* has a full supply of components to choose from, so let's get started.

Plugging Components into the Sockets

Building a tree begins with plugging components into the grid of sockets, a simple drag and drop operation. Be sure that the *Component Bin* is selected on the *Edit Control Panel*, so that you'll have access to the components. We covered the *Component Bin* on page *56*. Each folder that you see contains only those components that perform one type of function or give similar results. Here is a step by step guide on how to plug a component into a socket.

1. From the *Component Bin* open a folder by clicking on the "**+**" symbol preceding the folder name. Next choose a component by clicking once on its name or icon with the left mouse button. Keep the mouse button depressed.

NOTE: Remember, the component classes are used as the folder names on the Component Bin.

2. Continuing to hold the mouse button down, drag the component onto the Editor *workspace*. Notice the icon on the tip of your cursor. When you first begin dragging a component from the bin and as you drag it over occupied sockets, the cursor-tip icon looks like the screen capture on the left in *Figure 2.31*. As you drag the component over legal sockets, the icon on the cursor-tip changes. When it looks the same as the icon on the right in *Figure 2.31*, you're over a legal socket and can release the mouse button. Choose any legal socket you wish.

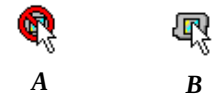


Figure 2.31
The No (A) and Yes (B) cursors for dragging components.

NOTE: Legal sockets are just empty sockets!

3. As soon as you drop a component onto a socket, the ***Paste As*** dialog opens and asks whether you want to paste the component as a color, percent or bump type. *Figure 2.32* is a screen capture of the ***Paste As*** dialog. The type you choose is generally dictated by the type of parameter you want to link to it. The type of component you select for the root determines the type of the entire DarkTree.

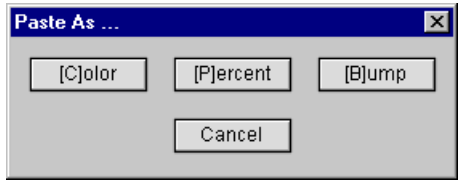


Figure 2.32
The Paste As dialog for pasting a component as a basic type.Some components, such as the Generators,don't need this dialog.

Components That Cannot Be Root

Any component of any class can legally occupy the **Root** socket except those of the Generator class or an instance of the Audio Wave or Time component. The Editor will not even save a DarkTree if one of the components just mentioned is installed as root**.** Several other components, Randomizer for instance, are legal as root but generally are not good choices since they need a parent component to drive them.

Basic Information on Making Wire Links

You'll find each plugged-in component's linkable parameter names on its *links list*. To be linkable, a parameter must be one of the three basic DarkTree types: color, percent or bump. Furthermore, each listed parameter and the component it links to must be of the same type. And finally, the linked-to component must be plugged into a socket one or more levels or columns of sockets to the right.

The subject of linking a parameter naturally divides into two sections: (1) the *links list* and its symbols and (2) a step by step guide to making the initial link.

Interpreting a Component's Links List

A brief description of the *links list* has already been given in *"Parts of a Component Face" on page 57*, but the coverage here is more complete because it includes DarkTree Textures version 2.0 additions. Open a component's *links list* by right-clicking on the *links pad* (the wide area on the right side of the component frame). Once the list is open, you'll see all the linkable parameters specific to that component. The list sometimes includes a parameter you won't see listed on the *Component Editor* for that component. This is a link-only parameter. A link-only parameter has just two options, either link it or go with its default setting; in other words, you cannot change its value.

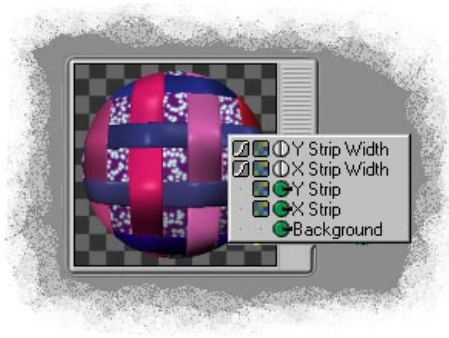


Figure 2.33

The links list for the Weave component, showing the three columns of icons possible for each parameter.

Figure 2.33 is a screen capture of the *links list* for the Weave component. Notice the *type icon* directly to the left of each parameter name. The icon is there to remind you that your link must be to a component of the same type. Notice, also, that the parameters ***Y Strip***, ***X Strip***, and ***Background*** are linked, and that ***X Strip Width*** and ***Y Strip Width*** are tweaked. The symbol that tells you this is a small black bar. When the bar is positioned horizontally on the right side of a *type icon* it's called the *links bar*, but when it's positioned vertically through the middle of the *type icon* (looking like a standard screw head), it's called a *tweaks bar.*

Looking to the left of the *type icon* column, notice that all but the last parameter have a second icon, which looks like a grid of random color values. This icon represents a regional parameter. Interpret its presence to mean that the parameter is especially suited to regional components. All Control class components are regional; that is, they give the quality of randomness to a parameter, based on its regions. You are not limited, however, to linking to a regional component. Components of the Control class, Fractal Noise and Fractal from the Generator class are the ones that will return regional data.

Again looking at *Figure 2.33*, you'll see that the ***Y Strip Width*** and ***X Strip Width*** parameters have a third icon with a flattened white S-curve design, in the left most column. These are the *function icons.* If a *function icon* has a pinkish-red background, you must link the accompanying parameter to a component of the Generator class. But if the icon's background color is gray (as in these two cases), you may link to either a generator or some other component. Note that linking a parameter without a *function icon* to a generator won't cause an error; it just won't give you a useful result.

Making Wire Links

The following is a simple step by step guide to making an initial wire link between some parameter and some component.

1. Open the *links list* menu by clicking on the *links pad* one time with the right mouse button. The *links pad* is the wide area on the right side of the component frame.

2. Select a parameter from the list by clicking once with the left mouse button. After you've selected a parameter, the *links list* will close automatically.

3. After making the selection, you'll have a stretchy wire with a *type icon* attached to the cursor-tip. This icon is a final reminder to link to a component of a "like" type. Next, using the cursor, stretch the wire-tip to the component you've chosen for the other end of your link. Click once with the left mouse button to attach the wire link. The link is made!

Links always begin on one socket level and are completed one or more levels to the right. In other words, component links go from a parent level number to a higher, child level number. Level numbers, a simple a way of pointing out socket locations on the *workspace*, were covered in *"The DarkTree Editor Workspace" on page 50.*

NOTE: If a parameter is tweaked, you must untweak it before you can select it from the links list.

Color-Coding for Wire Links

At a glance you can always tell the type of a link by the color of the wire. That's because DarkTree Textures uses the same color-code for links as for *type icons*. When selected a wire link always turns yellow, no matter which type it is. *Type icon* colors were covered on page *61*.

How Does Linking Change a Parameter's Value?

Linkable parameters get their values in one of three ways. Firstly if it isn't a link-only parameter, you can open a *Component Editor* and simply change the parameter value (or keep the default), in which case the parameter's status is "unlinked". Secondly, you can link a parameter to an existing tweak of the same type, in which case the parameter's status is "tweaked". Or thirdly, you can link a parent component parameter to a child component of the same type. In this case the parameter is considered "linked".

But what is this process really doing? When a link is made, the value generated by the child component is being substituted for a value either entered directly or tweaked. In other words, if you simply enter a value for a parameter, it will be the same for every point in texture space. The same situation applies to a tweaked parameter. But if you link a parent's parameter to a child component, the parameter receives dynamic values from the child component that are unique for each point in texture space! For example,

look at the component on the left in *Figure 2.34.* It shows a Rising Sun component whose percent parameter, ***Sun Size,*** was given a value of *50.0%.* Now look at the component on the right in the same figure. *Here* ***Sun Size*** is linked to a percent Rough noise component. Look at how this link affects the sun; it looks as though the sun has finally exploded. (Too bad for us Earthlings!) The ability to link parameters to components in this way is the linchpin that allows DarkTree Textures to create such powerful effects.

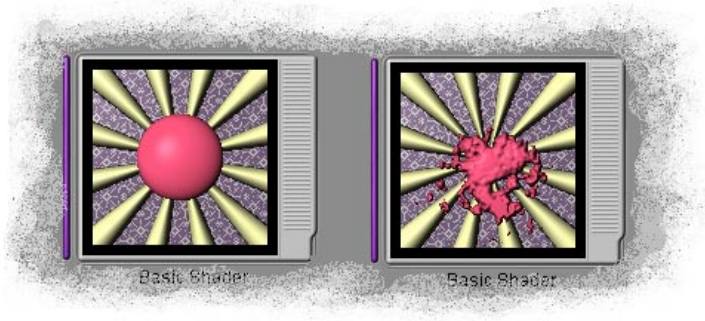


Figure 2.34
Another example of how linking a specific parameter to a component can dramatically change a texture.

Saving Your DarkTrees

For a DarkTree to be savable, some component that is not a generator or an instance of either the Time component or the Audio Wave component must be plugged into the **Root** socket. When a DarkTree is saved, everything else on the *workspace* is saved as well. That includes disconnected trees you may be doodling with off to one side.

A DarkTree is usually given two names: a descriptive name that's entered on the *Properties page*, and a file name that's saved to your system's directory structure. When you save a DarkTree, you don't have to add the file extension. The program will look at the root component type and choose the correct one for you. To save a DarkTree to the system directory, choose ***File>Save As*** from the *Editor menu.*

DarkTree file names all used to end in *dst.* This single file extension has been replaced in version 2.0 with four new ones. These are

- ***.dsts*** *(for DarkTrees that have a DarkTree Shader component in the Root socket)*
- ***.dstc*** *(for DarkTrees that have a color component in the Root socket)*
- ***.dstp*** *(for DarkTrees that have a percent component in the Root socket)*
- ***.dstb*** *(for DarkTrees that have a bump component in the Root socket)*

DarkTree Textures will continue to accept the old file extension, but new trees will be given one of the endings listed above. When you update a version 1.0/1.1 DarkTree, the program will create another copy with the same name but with the new file extension, and based on the type of the component in the **Root** socket, it knows what the new file extension will be. *"The Texture Library" on page 23* explains how to make your DarkTree files available through the Library.

2.3.5 Edit Options for Components and Wire Links

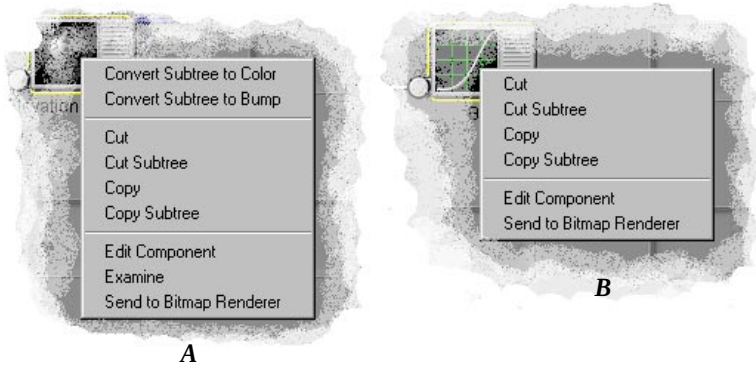


Figure 2.35

Two examples of the popup component menu. The menu on the left (A) was opened from a percent component; the one on the right (B) was opened from a generator. Notice that the menu on the right has fewer options.

The editing options covered in this subsection deal with moves, pastes, and deletes for components and with unlinks/relinks and deletes for wire links. Simply for this discussion, we've loosely divided the subsection into three segments or groups of options.

Most, but not all, of the editing options for components come from just two menus. These menus are the *component menus* and the *Editor menu*. The *component menus* are popup menus that you can access by right-clicking on any component or empty socket on the *workspace*. Instances of the *component menu* differ, because each is tailored to its component type. Check out *Figure 2.35* to get an idea of how these menus can vary.

The second menu is the *Editor menu*. This menu is located along the top edge of the DarkTree Editor window. For this subsection, we'll confine our discussion to those selections from the ***Edit*** drop down list of this menu. Look at an example screen capture of the *Editor menu* in *Figure 2.36*.

NOTE: You can also open a component menu from an empty socket, provided you have already performed a copy or cut operation.

Some Component Menu and Special Editing Options

You'll find the following three options on most *component menus*; however, none are available from the *Editor menu*. You'll see several references to the "opening component" in the following paragraphs. This simply refers to the component you right-clicked on to open a *component menu*.

First, we'll cover selections from the *component menus*. The remainder of this part of the discussion covers how to select and move components, and all about the DarkTree Editor's *dragging mode*.

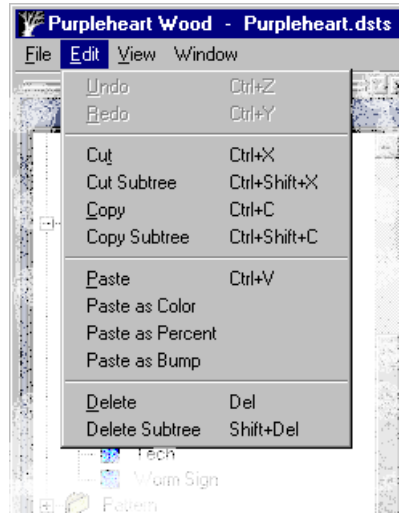


Figure 2.36
*A screen capture of the Editor menu options under the **Edit** drop down.*

Edit Component

Clicking on ***Edit Component***, which is found on most *component menus*, opens a *Component Editor* loaded with the specific parameters of the opening component. *Component Editors* generally have many features, and you'll find thorough coverage for them beginning on *page 76*.

NOTE: The more common way to open a Component Editor is to left double-click on the component image.

The Examine Window

The ***Examine*** menu selection, also listed on most *component menus*, opens an Examine Window with the opening component's image. However, you won't find this option on a menu opened from a Time component or any of the generators (what would the image look like?).

Examine Windows are an important and complex tool in the DarkTree Textures arsenal. The previous major section, beginning on page *33*, is devoted exclusively to their many viewing options.

NOTE: The component menus and the Texture Library are the only two places where you can access an Examine Window.

Send to Bitmap Renderer

And finally from the *component menus* is ***Send to Bitmap Renderer***. If you select it, the DarkTree Editor will send the opening texture and any children it has to the DarkTree Bitmap Renderer. Of course, if you select this option from the root component, then the entire DarkTree goes to the Renderer. As is the case with the ***Examine*** menu selection, a *component menu* accessed from a generator, or a Time or Audio Wave component won't have this option listed.

If you open a *component menu* from a DarkTree Shader component and then click on ***Send to Bitmap Renderer,*** you'll see a small dialog appear. It will have two choices: you can render the DarkTree as shaded (***Render Shaded***) or as single channels (***Render Channels***).

Selecting ***Render Channels*** will open the *Render Channels* dialog. This dialog splits out all of the channels that are linked to the Shader. *Render Channels* has the same rendering options as the Bitmap Renderer itself. The importance of the *Render Channels* dialog is that you can specify all output options for the separate channels at one time. The subsection titled *"Shaded & Single Channel Render" on page 129* gives detailed information on both of these dialogs and the separate rendering of shader channels.

Finally, *Render Channels* will iconify the DarkTree Editor and transfer the DarkTree to the Bitmap Renderer for final processing.

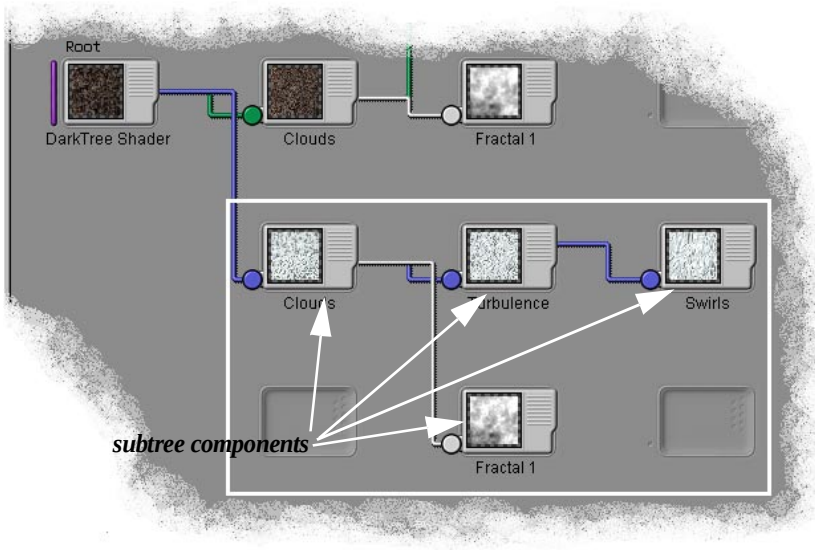


Figure 2.37

The screen capture of a DarkTree, highlighting an example of a subtree and its subroot.

Selecting a Socket or Component

"Select" is one editing option that you won't have to choose from a menu of some kind. Clicking on an empty socket or a plugged-in component places it in the selected state. After you make a selection, the Editor will outline the empty socket or component with bright yellow. The DarkTree Editor always uses the color yellow to signal the selected state.

If you want to select a subtree, simply choose a component (which then becomes the subroot) with linked children. *Figure 2.37* is an example of a subtree with its subroot component. Everything enclosed within the white rectangle is part of the subtree.

Moving Components & Subtrees

With a *move* operation you can relocate a single component or a subtree to new sockets on the *workspace*. After completing a *move*, the Editor will redraw any and all wire links from the new location.

Here's how to move components:

1. Choose a component subtree component and left-click anywhere on the its frame. Keep the mouse button down. Since the frame is narrow in most places, use the *links pad* area. Be sure to click on the frame; clicking on the image will not move the component(s).

2. Still holding the mouse button down, drag the component to the new legal (that is, empty) socket. To move an entire subtree, hold the ***Shift*** key down while you drag.

3. To plug the component into the new socket, just release the mouse button.

Pay special attention when moving an entire subtree, that you have enough empty sockets in the same configuration as the original. Otherwise the Editor will abort the *move*. See the next subject for an understanding of how the *dragging mode* can help you move components effectively.

Dragging Mode for Moves & Pastes

The Editor enters *dragging mode* specifically in conjunction with either a *move* or *paste* operation. Even though *dragging mode* works for single components as well, it hardly seems necessary. Therefore, we'll discuss this subject solely in conjunction with subtree *moves* and *pastes*.

Dragging mode is the DarkTrees Editor's way of helping you locate enough legal sockets in the original subtree configuration. This mode is especially helpful when you're working with large spread-out trees. Here's how it works! Whichever set of sockets you drag your subroot component across, the Editor takes to be a possible new location for your move or paste. And since the Editor knows the configuration of your subtree, it can determine whether you have enough empty sockets (in the correct configuration) to complete the operation.

Dragging mode uses two visual cues to help you select a legal set of sockets: highlight colors and cursor icons. Let's begin with the highlight colors. As you drag a component subroot over the *workspace*, notice that each socket is highlighted in either green or red. By highlighting sockets, the Editor is showing you the result if you were to paste the subroot at that particular location. The trick is to make sure all highlighted sockets are green. That ensures a legal *paste* for the whole subtree. If even one socket is highlighted in red when you click on the mouse button, the Editor will abort the *paste.* The same factors hold true for the *move* operation, of course. *Figure 2.38* is a screen capture of a proposed illegal *move*. The four-component subtree (each member is outlined in white) is moving one socket to the left, in the direction of the white arrows. The target socket that's outlined in black shows why the *move* will be illegal; the socket is already occupied by the Birdshot component, thus preventing Fractal Noise from occupying it.

With regard to the other visual cue, cursor-tip icons, take a look at *Figure 2.31*on *page 62*. It shows you the legal and illegal icons for the cursor. As you move the cursor across sockets, these icons give the same warnings as the socket highlighting.

When you have found the right socket set for your subtree, release the mouse button over the subroot socket if you're moving, or left-click it if you're pasting.

NOTE: You cannot drag a component or component subtree from one open DarkTree Editor to another. Use copy and paste instead.

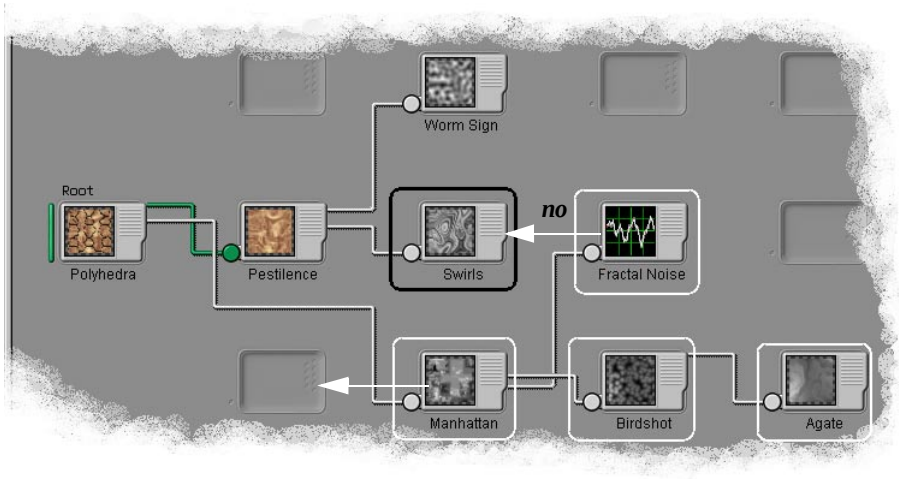


Figure 2.38

This example shows a four-component subtree that is about to move one socket to the left (hence the arrows). The move will be legal for three of the components, but the destination for the one outlined in black represents a conflict, since it's already occupied. Because one component's move will fail, the entire move will fail as well.

More Editor Application & Component Menu Options

This segment covers options on either the *component menus* or the *Editor menu* or both. The locations and how to open the menus are given in the opening paragraphs for this subsection.

Even though it's a repeat, a couple of definitions might be useful here. First, in this segment you'll find references to the "opening component". This is just the particular component you've right-clicked on to open a *component menu.* Second, for editing purposes the DarkTree Editor defines a subtree as a group of two or more linked components. The subroot component can be any component with a child or children linked to it.

Undo & Redo

Undo/***Redo*** is only available from the *Editor application menu*. ***Undo*** will back you out of your last operation, while ***Redo*** will reapply that last operation again. The ***Undo/Redo*** pair has a buffer of fifty levels, which provides you with plenty of leeway. If you cannot legally use either option (if the buffer is empty), they will be grayed-out.

Convert Subtree to

Some *component menus* give you the option of converting a subtree type, without moving it, to one of the two remaining key DarkTree types. For an example, look at the menu to the left back in *Figure 2.35.* Since the two selections are ***Convert Subtree to Color*** and ***Convert Subtree to Bump***, you know that the opening component was a percent-type.

If there are children linked to the component you select, the subtree will be converted. If you select a single, unlinked component, that's what will be converted. For a subtree, the Editor might not convert some linked components. One reason is that many parameters remain percent types, even if the component they belong with is a color or a bump-type.

Generators, the DarkTree Shader and a few other componenets will not have this selection on their menus. And you won't find it listed on the *Editor menu* either.

NOTE: When the Editor converts percent components to color, the initial colors will be grayscale. But when converting bump-type components to color, the Editor uses each component's default colors.

Cut & Cut Subtree

You'll find ***Cut*** and ***Cut Subtree*** on both menus. Begin by telling the DarkTree Editor which component or subtree you want to cut: ***Cut*** or ***Cut Subtree***. All "Cut" information is stored in the Editor's copy/paste buffer, and that includes wire links if you're cutting a subtree. Wire links to the cut component(s) will be disconnected.

In addition, you can cut selected components and component subtrees by using the following keyboard shortcut keys: ***Ctrl+x*** to cut single components and ***Ctrl+Shift+x*** for subtrees.

Copy & Copy Subtree

Copying a component or component subtree gives you the chance to install the exact same component(s) elsewhere on the *workspace* or on a different Editor altogether. Both menus include these options. ***Copy*** and ***Copy Subtree*** behave like the *cut* options, except that the original component(s) remain in their sockets. A *copy*, with any accompanying links, is stored in the Editor's copy/paste buffer.

The keyboard shortcuts for these options are ***Ctrl+c*** for single components and ***Ctrl+Shift+c*** for subtrees.

Delete & Delete Subtree

The *delete* options are listed only on the *Editor menu*. Select the component or subtree that you want to remove, then choose ***Delete*** or ***Delete Subtree*** from the menu. ***Delete*** does not save anything in the copy/paste buffer. Once you delete a component, it's a goner. (Well, actually you can undo your hasty deletes.)

Or you can use the keyboard shortcuts. To remove a single component, first select it and then use the ***Del*** or ***Delete*** key. For a component subtree, select the subroot component, then use ***Shift+Delete.***

Paste, Paste as Color, Paste as Percent & Paste as Bump

Both menus list the *paste* options in some form. So, after you've filled the copy/paste buffer using a *cut* or *copy* command on some component or subtree, you can proceed with a *paste*.

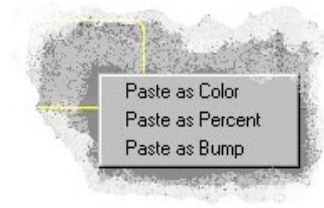


Figure 2.40
An example of a component menu that was opened from an empty socket.

To paste from a *component menu*, just right-click on an empty socket. When you open a *component menu* from an empty socket, the menu selections are all *paste* options, and then only if there is something in the copy/paste buffer. *Figure 2.40* shows a screen capture of a *component menu* that was opened from an empty socket. To do the same thing using the *Editor menu* select an empty socket first, then open the menu and choose one of the four paste options. If you select the plain ***Paste*** option, the *Paste As* dialog will pop up and ask you to choose among the three key DarkTree types.

If you're pasting an entire subtree, click on an empty socket for the subroot, making sure you have sufficient room for the entire subtree. And remember that if the subtree doesn't fit on your first attempt, the Editor goes into *dragging mode.* This gives you the chance to be guided by socket outline colors and icon types.

Editing the Wire Links

Selecting, moving, and deleting wire links is a bit different from performing the same operations on components. As you can see just by looking at the maze of wire links in a complex DarkTree, it's very easy to lose track of which component each link goes to. You can trace a specific wire link by left-clicking on it. That will select it, making it much easier to trace. Likewise, just looking at the wire links does not tell you which parameter each one is coming from. However, if you position the cursor somewhere on a wire link and right-click once, the parameter name and its details will appear (see *Figure 2.41*).

DarkTree Textures provides two ways to handle link editing: (1) you can go through the *links list*, which involves more or less going through the same operation each time; or (2) you can use direct link selection. You can only employ direct link selection after you've initially created a wire link using the *links list*. Creating the initial wire links is discussed in *"Basic Information on Making Wire Links" on page 63.*

We'll discuss these two methods under the headings that follow.

Figure 2.41

Right-click on a wire link to identify the component and the parameter it comes from.

Selecting a Wire Link

Using the Links List

First, open the *links list* by right-clicking once on the *links pad*. This will bring up a list of the linkable parameters belonging to the component. You can tell which parameters are linked because they have a black link bar on the *type icon*. Select a currently linked parameter from the list. As soon as you've made your selection, the wire link will turn bright yellow.

Direct Link Selection

Put the cursor anywhere on the target wire link, then left-click to select. The link will turn a bright yellow. This is obviously a shorter method, no?

Moving a Selected Wire Link

Using the Links List

First, you'll have to delete the highlighted link. So select it and hit your ***Del*** or ***Delete*** key. After deleting the current link use the regular linking process, which is explained in *"Basic Information on Making Wire Links" on page 63.*

Direct Link Selection

You need not bother selecting the link first when using the direct link selection. After positioning the cursor somewhere on the link, hold down the left mouse button, and drag the link to your new target component. Dropping it on the target component's *type icon* or image completes the link.

Deleting A Wire Link

Using The Links List

Open the *links list* and select the parameter whose link you wish to delete. The link will turn yellow (the selected state). Then, either use your keyboard's ***Del*** or ***Delete*** key or choose ***Delete*** from the *DarkTree Editor menu*.

Direct Link Selection

You can select a wire link and then hit your ***Delete*** key. Or you can select and drag the link to the background area and just drop it by releasing the mouse button. Since it's not over a component it will abort the linking process, thereby deleting the link.

2.3.6 The Component Editors

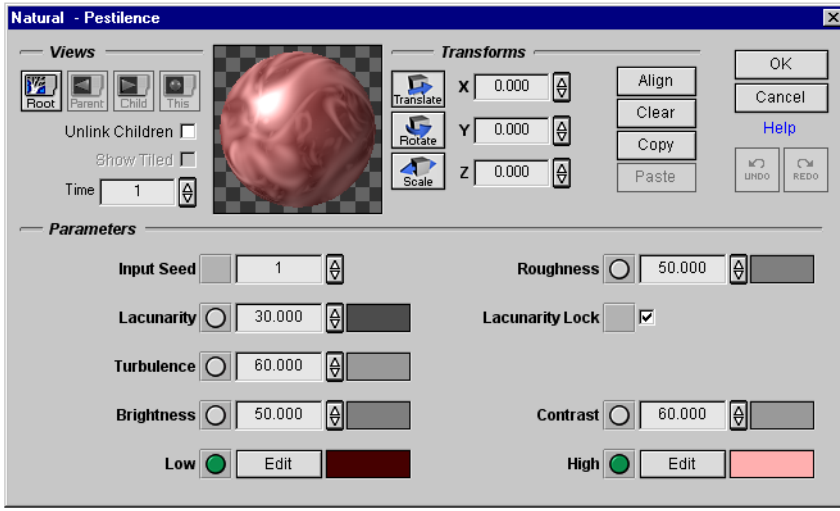


Figure 2.42

A screen capture of the Pestilence component's Editor. Each section of a Component Editor will be covered separately.

Overview

This subsection and the next, *"The Generator Editors - Special Features" on page 91*, are the third piece in our DarkTree Editor discussion. With the exception of the *Generator Editors* and the editor for the Time component, all *Component Editors* use the same format. *Generator Editors* have a slightly different format because each generator is a two-dimensional function. The Time component's editor is different because it includes the *Spline Editor*, and because it's specifically designed to handle texture animations. *Spline Editor* usage is covered in *"The Spline Editor" on page 92*. More thorough coverage of animations and the Time component can be found in *"Animating Your DarkTrees" on page 189*.

Each component has a series of user-settable parameters that defines how it looks. And each *Component Editor* is especially fitted to one specific component; that is, it's loaded with that component's particular set of parameters. However, a standard format divides all *Component Editors* into the same labeled areas, with each performing its own set of tasks. The areas are ***Views***, ***Transforms***, and ***Parameters***.

After examining the information given here, you'll be able to translate, rotate, and scale components, change component parameter values, and select a variety of texture views. So let's open a *Component Editor*, such as the one in *Figure 2.42*, and begin examining its features.

A Few Preliminary Options for the Editor

These few options are located on the upper right area. They aren't one of the areas mentioned above but simply a few standard controls that are mentioned here for the sake of completeness. *Figure 2.43* shows this small area.

Figure 2.43
The close/cancel area of a Component Editor.

Opening

Now there are two easy ways to open a *Component Editor*. The first is to choose any component that's plugged into a socket on the *workspace*, and left double-click on its image. An editor will open, customized for the selected component. The second way is to right-click once anywhere on a component image. This will open a customized *component menu* that lists ***Edit Component*** as an option.

Closing

Click ***OK*** or ***Cancel*** to close a *Component Editor*. ***OK*** closes it and saves your changes. ***Cancel*** forgets any changes you've made since opening and simply closes.

Accessing Online Help

Directly under ***Cancel*** is the word **Help** (written in blue). Clicking on it will link you directly to an html document. This document is the Component Reference, which has detailed information on the components and their parameters. The *Component Editor* you open the Reference from determines the component description you'll see first.

The Undo & Redo Buttons

Use ***Undo*** to reverse any changes you've made. And use ***Redo*** to reinstate them again. The *update window* immediately renders any changes brought about by ***Undo*** or ***Redo***. The levels of undo are virtually unlimited, but as soon as you close a *Component Editor* the buffer is cleared again.

*NOTE: The Component Editor undo/redo uses a different buffer and is independent from the **Undo** and **Redo** selections on the Editor application menu.*

The Views Area

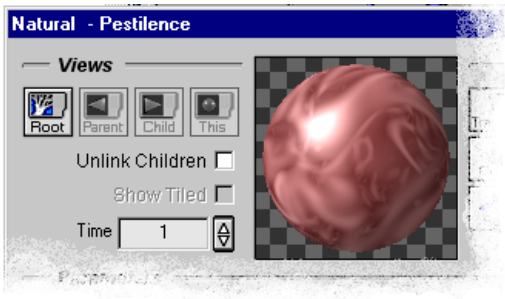


Figure 2.44
The Views area of a Component Editor.

The Update Window

Each time you change a parameter value or apply a transform, you'll see the resulting change in the *update window*. That's the small image window just to the right of the ***Views*** controls. The *update window* makes changes dynamically as you change a view, enter new parameter values or apply a new transform.

Under some circumstances you'll want to change the mapping geometry that's displayed in the *update window*. You have five geometry choices: planar, cylindrical, spherical, cubic, and frame. Frame puts the image into 2D space. The other mapping geometries are all considered 3D, even planar. First close the *Component Editor*, then change the mapping geometry on the *Properties* page. You can open this page from the *Edit Control Panel*, which is thoroughly covered beginning on page 52.

Figure 2.44 is a screen capture of the *update window* and ***Views*** area controls. What you see in the *update window* depends on the viewing choice you currently have selected. We'll cover ***Views*** next.

The Views Area Options

The *Component Editor* ***Views*** area offers several options, including a chance to step through an animation, stop and examine a single frame or see a tiled view of your texture. For your information, the viewing options are not persistent. So if you reopen the same *Component Editor*, the *update window* will display the (default) current component image again, regardless of the view chosen during the last session.

First, let's talk about the viewing buttons, then continue with the remaining ***Views*** options.

The Root Button

Clicking on this button puts an image of the root component in the *update window.* The root component shows you how your changes are affecting the DarkTree as a whole. If there isn't a component in the **Root** socket, the ***Root*** button will be grayed-out.

The Parent Button

A click on this button will put the image of the current component's parent into the *update window.* Another click will move the view to the next parent image. If you continue clicking on this button, it'll show successive parent images until you reach the DarkTree **Root**. If it isn't currently legal to click on this button, if the current image is the **Root** for instance, the ***Parent*** button will be grayed-out.

NOTE: A component may have more than one parent. DarkTree will select one to follow.

The Child Button

The ***Child*** button will be grayed-out if you're viewing the current component image. Once you've clicked on the ***Parent*** button, then it's legal to use the ***Child*** button. Each time you click on this button, the view moves one component closer to the current component. Once you reach the current component, the ***Child*** button will be grayed-out again since you cannot view the children of a current component.

The This Button

The ***This*** button simply puts an image of the current component into the update window (this is the default). If you're already viewing the current component, ***This*** will be grayed-out.

The Unlink Children Check Box

Unlink Children appears to remove the influence of any linked children. This check box doesn't actually unlink child components, but the image in the *update window* will be of the current component only.

NOTE: Keep in mind that if a component is linked to a child through its color parameters, checking ***Unlink Children*** *will produce an image that has reverted to its original component color(s).*

The Show Tiled Check Box

Show Tiled is for previewing DarkTrees that have one or more tiling components. Although you can get a tiled view of any image, it's unlikely to look seamless without a tiling component.

Before an image can be tiled, you must change the mapping geometry to ***Frame***. Change the geometry on the *Properties page*, which you can access from the *Edit Control Panel.*

The *Component Editor* keeps this option grayed-out until thegeometry is suitable for tiling, in other words until you've changed to ***Frame***. When you do check ***Show Tiled***, you'll see four tiles of the image in the *update window.*

The Time Edit Box

This option is specifically for animated DarkTrees. Whichever frame number you have set as ***Current Frame*** on the *Properties page* is the one you'll see initially in the *update window.* Use the zip slider to advance or go back a frame at a time. You can type in a frame number as well.

NOTE: You can apply any of the viewing options to any frame of a DarkTree animation.

The Transforms Area

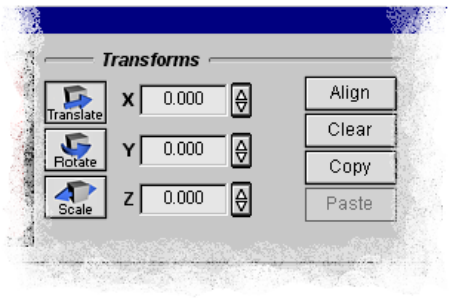


Figure 2.45
The Transforms area of all standard Component Editors look like this one.

Transforming Algorithmic Textures

Directly to the right of the *update window* is the ***Transforms*** area as shown in *Figure 2.45*. It's designed so that you can easily apply translations, rotations, and scalings to your textures. You can watch the component image in the *update window* change dynamically as you apply a transform. You cannot transform a generator function, components from the Control class, or the Time component.

Most texture components that come with the DarkTree Textures package are algorithmic. Without a little initial information, applying transforms to algorithmic textures can be confusing and the results not what you'd expect. If you haven't encountered this nomenclature before or are puzzled by some of your transform results, you can read an in-depth discussion on the subject in *"Transformations in Texture Space" on page 174.*

Algorithms are simply mathematical functions. And since the DarkTree algorithms are 3D (though a few don't vary along the Z-axis) you can think of these textures as solid blocks of material, essentially infinite in size. The portion of the block that you'll eventually apply to your model is generally the one that's visible in the *update window*. When you do apply a texture to a model, you're in effect carving the model's shape out of the texture block. And, when you apply a transform you're simply moving the texture block over to a new area (translating), or orienting the block differently (rotation), or shrinking/expanding the block design (scaling).

Since all DarkTree transforms are relative, we'll begin by briefly covering what that means. Following that, we'll discuss the three transforms and how to apply them. Finally, we'll talk about the other four buttons: ***Align***, ***Clear***, ***Copy***, and ***Paste,*** that accompany the transform buttons.

NOTE ON TRANSFORM VALUES: When a value is referred to as "unbounded", that simply means that there is no upper or lower limit set.

Notes On Relative Transforms

It's important to understand that all component transforms are relative. Making them so is the most flexible (and common) solution to a difficult programming problem. When a transform is relative, the value is reset to the original starting value, *0.0* for translate and rotate, and *100.0* for scale, each time. Think of it as transforming a component relative to your view, rather than to the global axes. Because the reset happens very quickly you might miss it, which can be disconcerting. And you may get the impression that the *Component Editor* didn't apply the transform at all. Don't worry, it did.

Translating a Texture

If you translate the texture block mentioned earlier, you're really just moving the view to a new area. *Figure 2.46* shows a texture pattern before and after we applied a translation of *+100.0* to the X-axis.

While the translation values are not bounded, you can keep track of how far you're moving if you consider that entering a value of *-100.00* for the X-axis will move the center of the block all the way to the left edge. And a value of *+100.0* applied to X moves the block's center the same distance to the right. Applying the same values in

Y means positive values move the texture block higher, and negative values move it lower, again halfway between the current center and the visible edge. And finally, for the Z-axis, positive values move the texture back into the screen; negatives move it out of the screen.

Figure 2.45 is a screen capture of all ***Transforms*** area buttons. When you're ready to translate a texture, click on the top button (it actually says ***Translate***). Next, enter a value in one of the axis input boxes: X, Y, or Z. Finally, pressing either the ***Return/Enter*** or ***Tab*** key will update your translation.

As soon as the *Component Editor* finishes applying the translation, it resets the input boxes to *0.0*.

Figure 2.46
An abstract texture design that is shown untranslated (A) and then translated (B) by 100.0 in X.

NOTE: *A translation in Z on a texture that doesn't vary in Z will not affect the image.*

Rotating A Texture

Rotating a texture revolves it about either the X-, Y-, or Z-axis. To apply a rotation, click once on the ***Rotate*** button. The input values are in degrees, with *360.0* being equal to one complete rotation. However, the input values, like those for translation, are unbounded; you can put in any number of degrees. Entering positive values rotates the image in a counter-clockwise direction; negative values result in a clockwise rotation. Pressing the ***Return***/***Enter*** or ***Tab*** key will update the texture and reset the value to *0.0*. For a visual example of a texture pattern before and after a *+38.0* rotation along the Z-axis, check out *Figure 2.47*.

NOTE: If you apply an X or Y rotation to a texture component that does not vary in Z, the program will rotate it, but the texture will begin to look funky or stretched out, rather quickly.

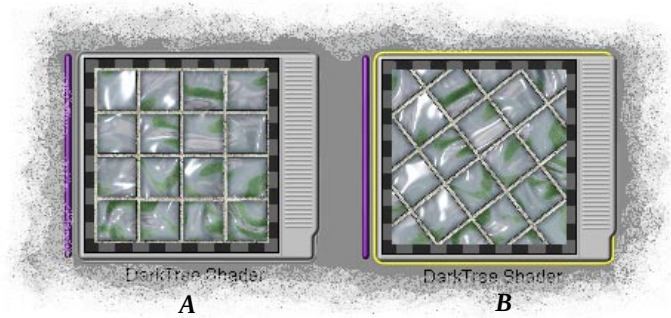


Figure 2.47
A Glass Block texture that's shown unrotated (A) and then rotated (B) by 38 degrees about the Z axis.

Scaling A Texture

Scaling acts like a camera lens, zooming in or out of a texture's pattern. You can think of this visually as shrinking a texture (zooming out), thus making the pattern smaller, or stretching a texture (zooming in), thus making the pattern larger. You can apply scaling in one of two ways: evenly or unevenly. *Even* scaling retains the texture's proportions as they are. *Uneven* scaling only changes one axis at a time, stretching or shrinking the texture pattern in comparison to the other two axes. *Figure 2.48* is a screen capture of the same two DarkTrees. The left one is not scaled at all, and the right one shows the same texture scaled unevenly (in Y only).

The start and reset value for scaling is *100.0%*. Entering a value greater than *100.0%* tells the *Component Editor* to zoom into the texture, causing its features to appear ever larger. Values less than *100.0%* down to *10.0%* cause a zooming away from the texture, so that its features appear to be increasingly smaller. Each time you apply a scaling, the result will be based on the current pattern size. For example, if you scale the component down to *10.0%*, the next time you enter *10.0%*, it will be ten percent of the first scaling. Scaling down requires some caution, because it doesn't take very many passes before the texture is reduced to a jumble of aliasing artifacts.

After you click on the ***Scale*** button, you'll see a small check box to the right of the X-axis input. (This check box won't be visible until you click the button for scaling.) To begin with, the check box will have a check mark in it. This enables even scaling. You'll only be able to enter values in the X-axis input box, but the scaling will apply equally to all three directions. If you prefer to scale unevenly, click in the box to

disable the even scaling feature. Now you can enter a value in any of the three axis input boxes. As with the other two transforms, pressing ***Return***/***Enter*** or ***Tab*** will update the transform and reset the value to *100.0*.

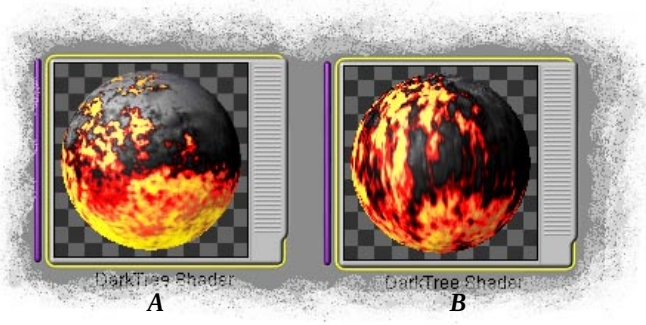


Figure 2.48
A fireball texture that's shown unscaled (A) and then scaled in Y (B).

The Align Button

The ***Align*** button brings the current component into alignment with its parent. That is, all scalings, translations, and rotations previously applied to the parent will now be applied to the child. If the current component was linked at the time that a translation was applied to the parent, it and any children linked to it were aligned automatically. So you won't need to bother with the ***Align*** button. If, however, you link a parent component that's been translated previously, to a child (or a subtree of children) then you can use the ***Align*** button to apply any transforms.

In the case of multiple parents, the child component can only be aligned with one of them; however *which* one it should be, is ambiguous. So try to avoid getting into this situation. The third *section of the manual* includes a thorough exploration of this problem. Look in *"Transformation Inheritance" on page 176*.

The Clear Button

The ***Clear*** button removes all transforms from the current component. In other words, using this button eliminates all rotations, translations, and scalings applied to the component since it was dragged onto the DarkTree Editor *workspace*.

The Copy & Paste Buttons

Copy uses a buffer to store the current component's transformations. This is done so that you can copy them to another component for purposes of alignment. ***Paste*** is grayed out until you've copied at least one transform. After you've made a copy of a

transform, you can paste it onto any other component(s) in the DarkTree. ***Paste*** works like ***Align*** but uses the transform information from the copy buffer, instead of the parent's transform.

In the case of the ***Paste*** button, remember that if a child was already linked at the time the current component received its transforms, that the child (any linked children) automatically received the same transforms. However, if you link a child or children after the transforms to the parent are completed, the child/children won't inherit those transforms. In this case, you'll have to paste them separately.

The Parameters Area

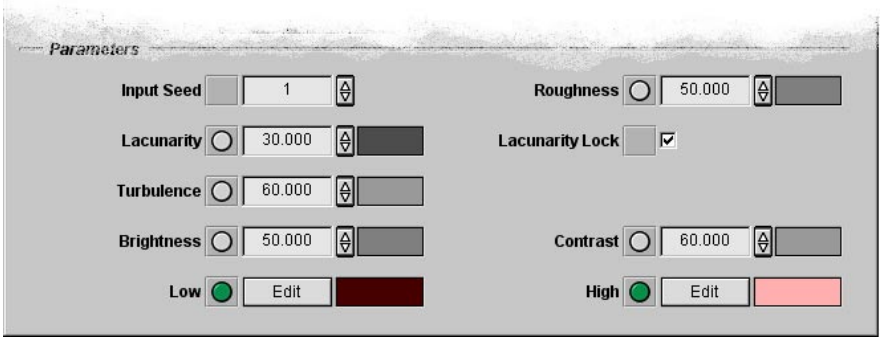


Figure 2.50

The Parameters section of a Component Editor. The number of parameters here is typical of many components.

Parameters Area Overview

Each DarkTree parameter defines one attribute or characteristic of a specific component. For example ***Width***, ***Height*** and ***Depth*** define geometric attributes of the Bricks component. Most parameters use numbers (integers, percents, or floating points) to define their characteristics. Others use color values, check boxes, or a drop down list of options. All parameters are set initially with default value. *Figure 2.50* is a screen capture of just the parameters area of a typical *Component Editor.*

The line for each parameter includes several pieces of information, such as link or tweak status. But how do you know what information a specific parameter includes? This section answers that question, and in fact tells you *almost* everything there is to know about the parameters.

The Parameter Display Format

All component parameters in the ***Parameters*** area more or less follow a standard format, with differences that are appropriate for each type. A breakdown of this format follows. Starting on the left, each parameter will have

- *A parameter name.*
- *A tweaks click pad, which is a plain gray box with a debossed perimeter. If the parameter is linkable, you'll see a type icon on the pad.*
- *An input control, such as an input box,* ***Edit*** *button, check box, drop down list or text box. The DarkTree parameter types will be treated separately later in this subsection. For currently linked parameters, the input space will have* ***Linked*** *printed in it. If the parameter is tweaked and not numberic,* ***Tweaked*** *will be printed in the input space. If the parameter is numeric and tweaked, the input box will be grayed-out but display the current tweaked value.*
- *A zip slider if the parameter input is numeric. (Zip sliders are covered soon.)*
- *A swatch, if the parameter is either a color- or a percent-type. (Bumps also have swatches but you won't see one until the parameter is linked.)*

The Tweaks Click Pad & Linkable Parameters

The *tweaks click pad* is a small gray square with a debossed perimeter. Nearly all parameters have one located just to the right of the name. We should briefly mention here that a tweak is a named macro setting that you can easily apply to any number of component parameters as long as they're of the correct type. Although most tweaks editing tasks are more easily handled by the *Tweaks Editor*, the *only* place you can assign a tweak to a specific parameter is from the click pad.

A t*ype icon* sitting on top of a *tweaks click pad* tells you that the parameter is linkable. And if you see a *link bar* you'll know that the parameter is currently linked. You can use a linkable parameter in one of three ways: (1) you can simply give it a value suitable to its type (a color for instance) or keep the default value; (2) you can link it to a child component of the same type; or (3) you can create a tweak for it. A *tweak bar* signals that the parameter is currently in the "tweaked" state.

Figure 2.52 is a screen capture of two parameters, one in the "linked" state and the other currently "tweaked". The linked state is signaled by a *link bar* on the *type icon*. This is a small black bar that's positioned horizontally on the right side of the icon. A

tweaks bar is a small black bar that vertically divides a *type icon* in half.

Figure 2.52
An example of a tweaked parameter (A) and a linked one (B).

The options available for a parameter at any given time depend on either its linked or tweaked state. Following is a list of options that *might* be available through the t*weaks click pad*, depending on the parameter's current state. *Figure 2.54* is a group of screen captures showing the various options under discussion.

- If the parameter is currently linked, you'll see this menu item: ***Unlink: [component name]***
- If you've already unlinked the parameter through the *tweaks click pad*, you'll see this menu item: ***Link: [component name]***.
- If you would like to create a new *tweak*, this menu item is always present: ***New Tweak:*** If a *tweak* of the proper type already exists, you'll see: ***Tweak With:*** followed by a list of *tweaks* you can choose from.
- If the parameter is currently *tweaked*, you'll see this menu item: ***Untweak: [tweak name]***.

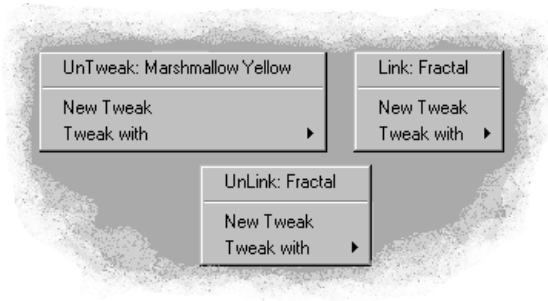


Figure 2.54
Options you might see after opening a tweaks click pad.

Using the DarkTree Zip Sliders

Any parameter that requires numeric input is equipped with a zip slider. Zip sliders, a custom DarkTree Textures control, behave exactly like the common sliders that many programs have. The input box updates dynamically as you zip through values, up or down.

The Parameter Input Formats

We dealt with the general parameter format with *"The Parameter Display Format" on page 86.* Component parameters come in eight formats: color, percent, bump, float, integer, drop down list, check box, and filename text box. You can make tweaks for all eight format types, but only three parameter types (color, percent, and bump) are linkable. In DarkTree Textures these three parameter types match the three key component types. *Tweak* values must be changed from the *Tweaks Editor.*

Next, you'll find a brief paragraph on each parameter format. These mention any special points and includes a visual example as well.

Color Parameters:

The color parameter is one of the three key DarkTree formats and is always linkable. These parameters will always have a green *type icon* on the *tweaks click pad*. To change a color, you can either click on the ***Edit*** button to open the *DarkTree Color Browser* or simply alter the HSV values with the cursor. No matter the method you select, you'll see the current color reflected on the color swatch and in the *update window.* If the color parameter is already linked, the color swatch will display a rendered image from the child component.

Here's how to change the HSV values with the cursor. To change the saturation (***S***), position the cursor over the color swatch and, with the left button depressed, move the cursor up or down. For changes to hue (***H***), move the cursor to the left or right. To change the color value (***V***), hold the shift key down while moving the cursor up or down. While you're changing a color, you'll see the above letters (***S*** and ***H*** or ***V*** and ***H***) when you hold down the left mouse button over the color swatch. This is simply a reminder.

Percent Parameters:

The range for all percent parameters, another key DarkTree format, is *0.0%* to *100.0%*. These parameters will always have a gray *type icon* on the *tweaks click pad*. Use the zip slider to enter a value quickly, and notice how the *update window* changes the component image in real time. Many component parameters are percents, both linkables and non-linkables being heavily represented. The gray level on the swatch corresponds to the input (or default) percent value (see the next note).

NOTE: The level of gray on the swatch is derived from assigning pure white a value of 100.0% and pure black a value of 0.0%. The percent value you enter will be equal to some gray shade in between black and white.

Bump Parameters:

Bump parameters are the third key DarkTree format. These parameters will always have a blue *type icon* on the *tweaks click pad*. Bumps do have swatches but they aren't visible until after you've linked to a child component. The reason is that an unlinked bump has a single elevation and, therefore, the swatch wouldn't be meaningful. Bump parameters require floating point (decimal) input. At *+/- 10000.0*, the value range is probably larger than you'll ever need. The DarkTree Editor interprets bump parameter values as elevation levels.

You'll find that bump parameters used as elevations only occur as part of a bump component. However, you'll eventually run into a color- or percent-type component that has one or more bump parameters. These parameters must be linked to bump components, as usual, but the parent component will interpret the incoming values as floats, not elevations.

NOTE: Be sure to read "Notes On Setting Bump Values" on page 90. It will make a difference in the quality of your rendered bump maps. And you'll find a more in-depth exploration of bumps and elevation ranges in "Part 3: The Conceptual Discussions", in the section titled "Working With Bumps" on page 182.

Floating Point Parameters:

Floating point parameters are not linkable and so, of course, don't have a *type icon*,or a swatch. Even though the value ranges for some floats are restricted (bounded), the maximum is the same one used for bump parameters. As with other numeric parameters, use the zip slider for speedy input.

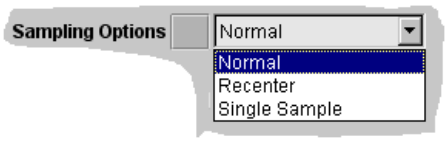
NOTE: For special case situations that require a linkable float parameter, read the information on bump parameters.

Integer Parameters:

The maximum range for integers is *0 – 10,000*. Integers are whole numbers; so the *Component Editor* won't allow you to enter a decimal point. Integers are never linkable!

Drop Down Lists:

DarkTree Textures employs the drop down list format when the parameter is a choice between two or more options. Just click on the right side of the box to make the list drop down, then make your selection.



Check Boxes:

Check boxes are really just toggles, a choice between enabling and disabling a parameter. A check mark in the box always means that the parameter is enabled.

Filename:

At present, components use this parameter for loading external files of some sort. First click on the ***File*** button, which opens a file requester. Browse for your directory and type in a file name, then select ***Close***. You'll see the entire file path name written out in the file name space. The *Component Editor* will immediately begin loading the external file for you.

Notes On Setting Bump Values

Bump components are floating point values representing elevations. Whereas the color and percent components have a specific value range, the bump components do not. The elevation value range is arbitrary. In other words, it's left entirely up to you. But you still need to put some thought into choosing a range for the following reason. Typical bump/elevation maps are saved as grayscale images. Each change in elevation is a shade of gray, with lighter shades representing the higher areas. The number of gray shades available for elevation maps is just 256, so your changes in elevation have to fit within that scale to be successful. DarkTree Textures will fit the gray shades to your chosen value range as well as it can. However, if your entire range is say, 0.0 to 1500.0, and you have included a cluster of very small changes in elevation somewhere within that larger range, your smaller elevations will surely be lost during the rendering process. There simply aren't enough shades of gray to handle such a large range plus the tight clump of values within it. This situation doesn't apply to textures that you use via one of the Simbiont plug-ins. However extreme bumps do not generally look good in this circumstance either, for other reasons.

See "*Section 3.4: Working With Bumps"* in "*Part 3: The Conceptual Discussions"* for a higher level but more in-depth, exploration of this problem and ways to get around it.

2.3.7 The Generator Editors - Special Features

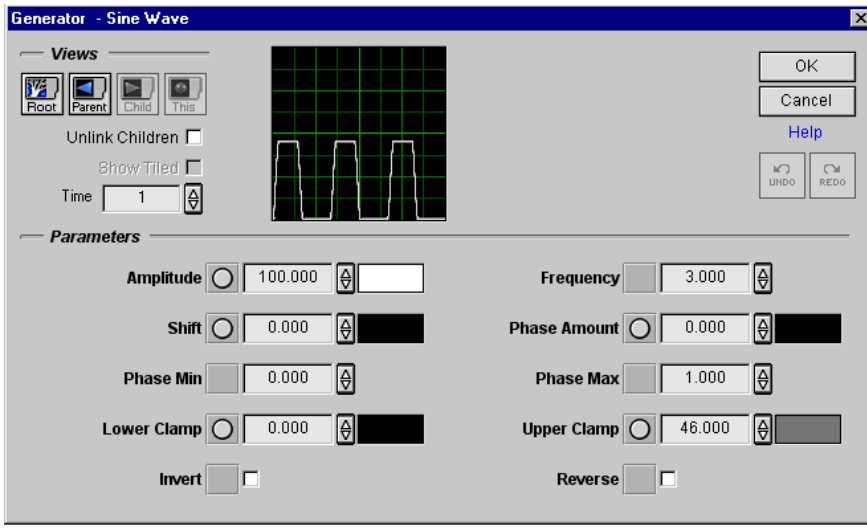


Figure 2.55

A screen capture of a Generator Editor. Most generators have several of the same common parameters, such as Amplitude, Frequency, and so forth.

Generator components are two-dimensional mathematical functions whose output is plotted rather than rendered. Because they control functions, *Generator Editors* have been modified from the standard *Component Editor.* For instance, *Generator Editors* have no need for a transforms area. *Figure 2.55* shows a screen capture of a typical *Generator Editor.*

Since generators are functions, the standard *Component Editor*'s *update window* has become a simple 2D plot (called the *plot window*). The vertical axis corresponds to output or percent values with a range of *0.0%* to *100.0%* and represents the output from the generator function itself. The horizontal axis represents input. The input originates from the linked parent parameter, which will be some percent-type, such as ***Stripe Width***, ***Sun Radius***, or ***Mortar Width***. The online *Texture Reference* clearly explains which parameters can be linked to generators, and there are many to choose from.

If the generator's ***Input*** parameter is linked to a Time component, the generator becomes time-based, and the *plot window* changes to reflect that it's now part of an animation. For time-based plots, the horizontal axis now represents time in the form of frame numbers. The default number of frames is always thirty, but you can increase or decrease that number on the *Edit Control*'s *Properties page* The vertical red line on

the time-based plot reflects the current frame (the frame being displayed in the *plot* or *update window*). The function itself is plotted in blue simply to reflect that it's now time-based.

Generator Editors have the same ***Views*** selections for the *plot window.* The ***This*** button will produce the plotted function; the other buttons will produce the same images they do in any standard *Component Editor. "The Component Editors" on page 76* thoroughly covers the viewing controls.

The ***Parameters*** area of a *Generator Editor* is just the same as it is for standard *Component Editors*.

2.3.8 The Spline Editor

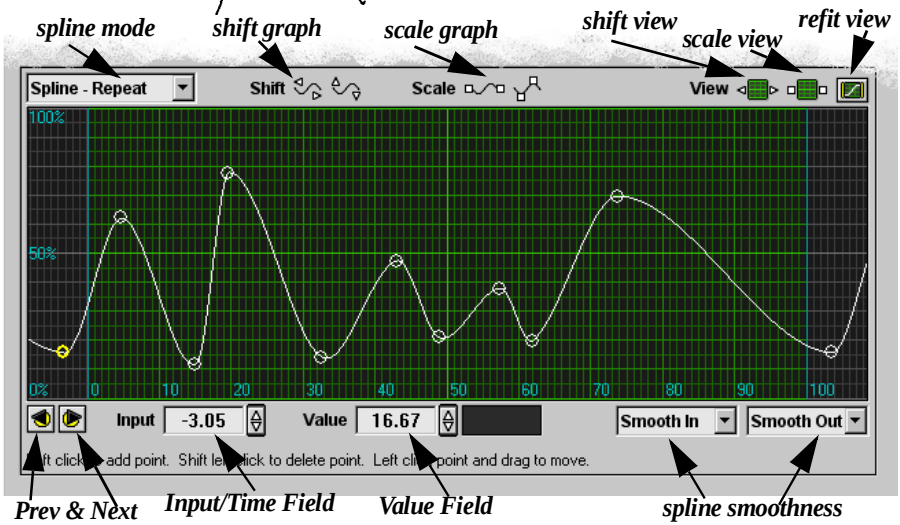


Figure 2.56
The Spline Editor with controls and other features named and located.

Overview

The *Spline Editor* gives you the option of defining your own spline curve. You'll find this editor both simple to understand and use, and very powerful. In spite of the fact that it's found only within the *Component Editors* of two components, it is an extremely important DarkTree tool! *Figure 2.56* is a screen capture of a *Spline Editor.* Notice that each control is named and located for you.

One of the two components that includes the *Spline Editor* is the Spline generator. With the Spline generator you can create your own function rather than having to rely on a pre-built one. *Figure 2.57* shows a tree structure that uses a Spline generator to define the top surface of a button. The second component that uses a *Spline Editor* is the Time component. Each instance of this component in a DarkTree enables you to animate some parameter. The example in *Figure 2.58* shows the tree structure of an animated DarkTree, *Exploding Corpuscles*. The figure includes some frames from the animation as well. Please refer to *"Animating Your DarkTrees" on page 189* for further coverage of this subject. For the following discussion, we'll be using the Spline generator component as our example.

NOTE: Throughout this discussion, you'll see the word(s)"curve" or "spline curve". This simply refers to the spline curve currently in the graph, whether it's a simple straight line or a complex series of waves.

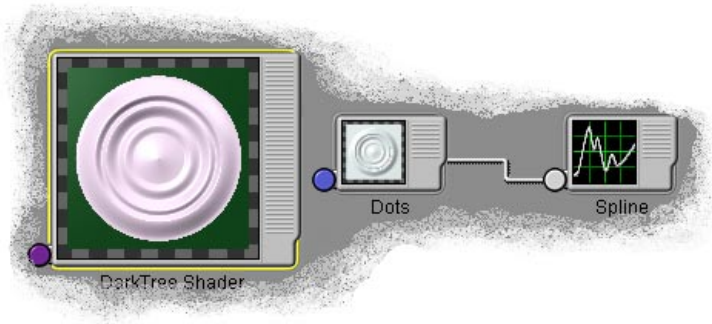


Figure 2.57

A Spline generator used to define the surface of a button. The button center corresponds to the right edge of the curve; the button's outer edge corresponds to the curve's left edge.

NOTE2: The Time component replaces the Trigger component of DarkTree version 1.0, and the Spline Editor replaces some tasks of the now defunct Trigger Bank.

Setting Up a Spline Graph & the Point-Tuning Controls

This segment touches upon some information about the graph area itself, then steps you through the building of a spline curve. All adjustments to this curve except adding, moving, and deleting control points are accomplished using the point-tuning controls. These controls operate on individual points, and are arranged below the graph. Refer back to *Figure 2.56* for the location of each *Spline Editor* control.

A Few Words About the Graph

The black graph area provides a place to build a spline curve. It's made up of two parts or areas: the middle or *active area,* with green grid lines and axis values; and the two ends, with grid lines and values drawn in gray and called the *inactive areas.* Parameters linked to a Spline generator will always make use of the part of the curve within the *active area*. However, in some circumstances, you'll want to continue your curve into the *inactive areas* (see the next note). In fact some of the controls, such as ***Spline-Repeat*** do this automatically. However, the editor can only utilize repeated curve values in the *inactive areas* under some specialized circumstances, for instance when you link up the ***Stripe Width*** parameter of the Stripes component.

The spline graph's vertical axis, designated as ***Value***, performs the same task for both components. The values for this axis are always percents (*0.0%* to *100.0%*), as the linked parameter must be as well. The Spline generator and Time component both return a percent value, based on the vertical position of the spline, for each request from the linked parameter of the parent component.

The horizontal axis numbers represent something different for each of the two component types. For the Spline generator, this axis is designated as ***Input***. It represents the same values that are passed down the tree through the linked parameter. For the Time component, this axis is labeled ***Time*** and represents the frame numbers for an animation (which, by default, initially runs from *1* to *30* frames). In each case, the vertical value for the same point is returned to the parent parameter. This gives the parent component the shape of the curve or animation, point by point. If you increaseor decrease the animation's length, the new frame range will automatically appear on the spline graph. Change the length of the animation from the *Edit Control*'s *Properties page*.

NOTE: After changing the length of an animation, a popup dialog will ask you if you want to scale the time to fit the new frame range. Click on ***Yes*** *to stretch or squeeze all Time components to fit the new length.*

NOTE: Starting your spline curve in the inactive area makes it much easier to create an ultra-smooth curve at the point where it enters the active area. You can use the same trick when leaving the active area, of course.

NOTE ON THE SPLINE GENERATOR: Any changes you make to the component parameters, which are not a part of the Spline Editor, won't be reflected in the spline graph. Look, instead, at the plot window near the top of the Component Editor for a complete and current update.

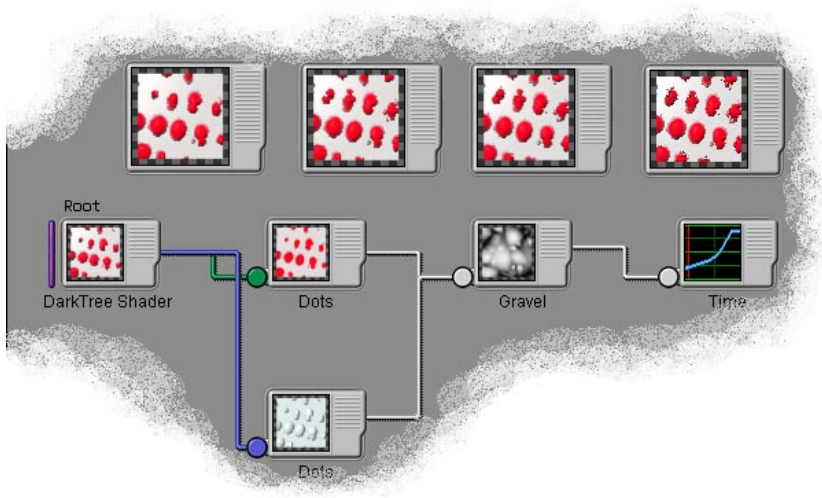


Figure 2.59
An 18 frame animation of exploding red corpuscles, using the Time component to define the animation path.

Adding, Moving, and Deleting Control Points

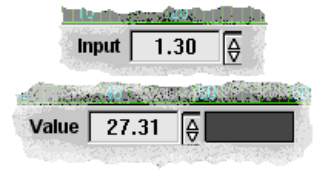
To Add: Control points define the shape of the spline curve. Add a control point whenever you want to change the direction of your curve, even slightly. To add a point, just left-click within the graph space wherever you want it. Notice that the cursor tip will have a little "**+**" on it. This just signifies that you're in the "adding-points" mode.

To Move: Moving control points is as easy as positioning the cursor on one. Left-click to select, then simply drag the point to the spot you've chosen. When dragging control points, notice the small four-way arrows surrounding the cursor tip. This just tells you that the cursor is currently in *dragging mode*. No two control points can have the same horizontal value; so be careful not to accidentally drag a point too far. If you do, the points will be reordered (switched). That is, they'll change position in relation to each other.

To Delete: To remove a control point, first hold the ***Shift*** key down, then left-click on a control point. A little red "***x***" at the cursor tip is a warning that the next point you select will be deleted. If you forget and select a point before holding down ***Shift***, delete won't work.

The Input/Time & Value Fields

You can select a control point simply by clicking on it. Or you can use the *prev* and *next* point buttons, covered in the next subsection. To reiterate, a control point will signal that it's currently selected by turning yellow and remaining that way until you select another one.



After clicking on a control point, you can check its exact position on the graph by inspecting the two fields labeled ***Input (***or ***Time*** for the Time component), and ***Value***. You can enter new numerical positions into either or both of these field at any time. A control point will change locations dynamically as you change its numerical position. You'll find these two fields left of center and below the graph.

Recall that the ***Input*** axis is the horizontal one whose values come from the parent component. This is only true for the Spline generator. For the Time component, the horizontal axis is labeled ***Time*** and the values are frame numbers. ***Value*** is the vertical axis and it's always the same. These are the numbers that will be your output; that is, they'll be returned to the linking parameter. The ***Value*** axis includes a percent swatch that shows visually, by the degree of grayness, what the height of a selected control point is (100.0% = white, 0.0% = black).

The Prev & Next Point Buttons

You can select and step through the curve's control points by using the *prev* and *next* point buttons. These buttons are located below and to the far left of the graph. After you select a point, you can enter new values into one or both numerical fields, drag it to a new location, or simply note the numerical position.

The Spline Smoothness Controls

By using the *spline smoothness* controls, you can adjust the way the spline connects to and departs from a control point. The two *spline smoothness* controls are located below and to the far right of the graph. These two drop down lists let you select one of two connecting line styles. The first control (the leftmost one) adjusts that part of the spline that connects to a selected point; the second control (the right most one) adjusts that part of the spline leading from a selected point.

The leftmost control's first drop down list selection, ***Smooth in***, causes the incoming spline to smoothly curve as it connects with a selected control point. The second line style is ***Linear in*** which, as you might guess, causes the incoming spline to connect to the control point with a straight line.

The second smoothness control adjusts a spline as it leaves a selected control point. The two list choices here are ***Smooth out*** and ***Linear out***. They make the same adjustments to an outgoing curve section as the first control does to an incoming one.

The Spline -Tuning Controls

You'll find the controls for adjusting the entire curve (as opposed to a single control point) all located above the graph area. Again, Figure 2.56 points out each control's location on the *Spline Editor.* The following paragraphs cover each of these controls.

Choosing a Spline Mode

The *Spline Editor* offers three graph modes. Each mode defines a different behavior for the spline curve. This drop down list is the left most control above the graph.

Spline - Hold is the first mode of the three. You'll find that you use this mode much more often than you do the remaining two. Selecting ***Spline-Hold*** keeps the spline curve just as you created it, except that the area before the first control point and after the last one are continued out as straight lines

Spline - Repeat is the second list choice. It tells the *Spline Editor* to repeat your spline curve configuration over and over. All of the curve between the first and last points is considered a part of the configuration, whether it's within the active area or outside of it. Subsequent to selecting ***Spline-Repeat***, moving one control point will move the same repeating control point as well. The repeat algorithm is set up such that the last point of your original spline curve will fit with the first connecting repeat point, making the entire series continuous. The curves repeat, for all intents and purposes forever, in the non-active areas.

Spline - Mirror is the third mode on the list and similar to ***Spline - Repeat,*** except the original curve is flip-flopped for each repeat. If you move any control point after specifying this mode, the matching repeat point for each mirrored curve will automatically move by the same amount. And just as with ***Spline-Repeat***, the *Spline Editor* will add mirrored curves within the active graph area, and continue them indefinitely in the non-active areas. ***Spline-Mirror*** is smoothly continuous (no breaks).

NOTE ON REPEAT AND MIRROR MODES: *Unless you plan your spline curve carefully, your repeat curves are likely to begin and/or end with a partial one within the active area.*

Shifting the Spline Curve

The spline curve *shift controls* move the curve either horizontally or vertically. The two *shift controls* are located above the graph, directly to the right of the word ***Shift***. The first control shifts the curve horizontally; the second shifts it vertically.

To shift the curve, position the cursor on or between the two mini-arrows and drag in the direction of an arrow. The spline curve shifts dynamically with the mouse's movement. Keep in mind that the spline curve cannot be shifted vertically above the ***Value*** boundary of *100.0%* or below the *0.0%* boundary. Attempting to drag the curve further will simply flatten it as control points hit the boundary. Once the curve is flattened, it will stay that way until you reconfigure it.

NOTE: Because the 2D plot (update window) near the top of the Component Editor is more compact, it's sometimes clearer to check it after a horizontal shift, just to see how much of your spline curve remains inside the graph's active area.

Scaling the Spline Curve

The *scaling controls* stretch or squeeze the entire spline in either the horizontal or vertical direction. You'll find these two controls located above the graph, directly to the right of the word ***Scale.*** The leftmost control scales the curve horizontally; the one to the right does the same in the vertical.

Scaling is always done about a control point that you select. After choosing your pivotal point (the one you want to scale about), position the cursor on or between the two mini-squares and hold down the left mouse button. Dragging the mouse either horizontally or vertically, depending on which scaling control you're using, changes the mini-squares to mini-arrows. The arrows indicate whether you're currently stretching or squeezing the spline curve. The control point remains stationary.

Vertical scale behaves exactly like the *vertical shift* control. That is, you cannot scale the spline curve above *100.0%* or below *0.0%*. As a control point hits one of these limits, it will stop scaling. And the spline curve will begin to flatten out if you continue in the same direction. Once the curve is flattened, it remains that way until you reconfigure it again.

The Spline Editor Viewing Controls

The three *Spline Editor* viewing controls adjust your horizontal view of the graph. These controls are grouped together above the graph and to the far right, directly after the word ***View***. Regardless of how a viewing control causes the graph to look, the spline curve is not actually affected in any way.

Next we'll briefly cover each of these options, beginning with the left most viewing control and moving to the right.

Shifting Your View of the Graph

The *shift view* control moves the entire graph to the left or right. Use this control to access graph areas outside the active area. For instance, you may want to add extra control points outside the active area or simply view those you've already placed.

To shift your view position the cursor on or between the mini-arrows, hold down the left mouse button, and move horizontally in the direction of one of the arrows. Moving the mouse either left or right shifts the entire graph as far as you want to go.

Scaling Your View of the Graph

The *scale view* control horizontally expands/contracts the view of the entire graph. The result is a zoom-in on some portion of your spline, or a zoom-out on the complete one. The *scale view* is especially useful for complex curves, since it gives you a chance to look at a single section by itself.

This control works like the graph scaling control. Position the cursor on or between the mini-squares and hold down the left mouse button. As you move the mouse horizontally, the squares change to mini-arrows that point in the direction you're currently scaling. Arrows pointing outward indicate that the view is being stretched; arrows pointing inward indicate that the view is being squeezed.

Refitting Your View of the Graph

The *refit view* control scales/shifts the graph area to include all of your control points. It's the farthest to the right of the three viewing controls.

A single mouse click is all you need to do to activate this viewing control. The *refit view* control creates a new view by taking the left most control point as the left viewing edge, and the control farthest to the right along the right edge. It doesn't matter whether a control point is within the graph's active area or off in the graph's hinterland (*inactive area*). It's still included in the *refit view.* If your control points are spread out, the graph will tend to look squeezed horizontally. A few close-together control points will produce a rather spread out view.

2.3.9 Tweaks Power! The Tweaks Editor Page

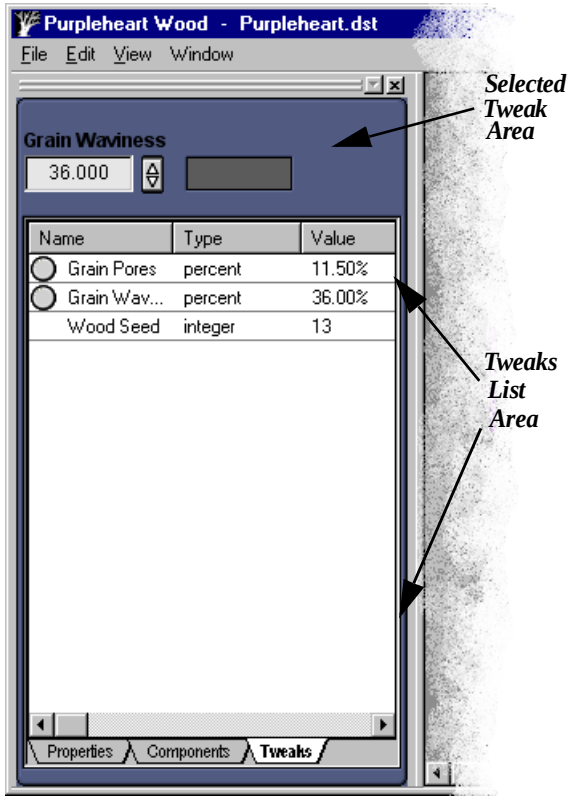


Figure 2.61
A screen capture of the Tweaks Editor showing the two main working areas.

Overview

The *Tweaks Editor*, a tabbed page found on the *Edit Control Panel*, has all of the controls you need to set up and edit the DarkTree *Tweaks*. You'll find *Edit Control* located along the left most edge of the DarkTree Editor. *Figure 2.61* is a screen shot of the *Tweaks Editor* page. The *Tweaks Editor* is divided into two areas: the *selected Tweak area*, where you can set or change a *Tweak* value and the *Tweaks list area*, where you can select a current *Tweak* or open popup menus that perform various tweak-related functions.

Tweaks are essentially macro controls for DarkTrees. You can potentially govern any number of component parameters with a single *Tweak*, and the parameters can be from any number of components. For example, suppose you've built a bricks shader that includes bump, color, and specular level (percent) links. The brick ***Width***, ***Height***, ***Row Shift***, and ***Mortar Width*** are obvious *Tweaks* choices, because each parameter value must match between types. And that way, if you decide to change ***Row Shift***, for instance, you can simply change its *Tweak* and all connected parameters are changed at the same time. The only rule here is that all parameters controlled by a *Tweak* must match it in type. All eight DarkTree parameter types support *Tweaks*.

Tweaks and the Simbionts

One of the most significant and important uses for *Tweaks* is associated with the new Simbiont plug-ins. Using these plug-ins, you can import DarkTrees as procedural shaders directly to various professional modeling/rendering packages, such as LightWave®. Once a DarkTree is imported into another package, any and all parameter adjustments are solely through the *Tweaks* it has. A DarkTree that has many tweaked parameters is very flexible because it's so easy to modify.

Selecting a Tweak from the List

To select a *Tweak* from the *Tweaks list*, simply left-click on either the *Tweak* name or *type icon* (the small colored circle). Although only the key parameter types (color, percent and bump) will have a *type icon*, you can select other *Tweak* parameter types by left-clicking on the space reserved for the *type icon*. For your convenience, both the *selected Tweak* and the *list area* are pointed out in *Figure 2.61*.

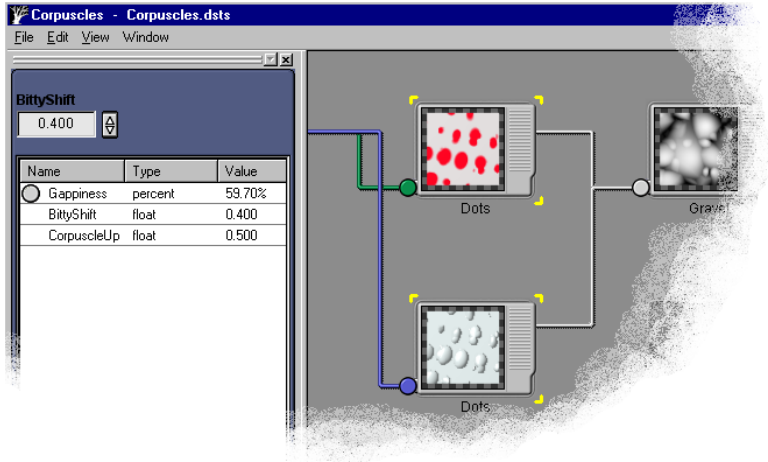


Figure 2.62

The corner marks signal that each component has a parameter with the same Tweak.

If you select a *Tweak* that's currently in use, look on the Editor *workspace* and you'll see corner tabs around those components that have parameters connected to the selected *Tweak*. Note these sets of corner tabs in *Figure 2.62* In this case, you can see that two components have parameters tweaked with ***BittyShift.*** The selected *Tweak*'s value is always displayed in the *selected tweak area.*

Creating a New Tweak

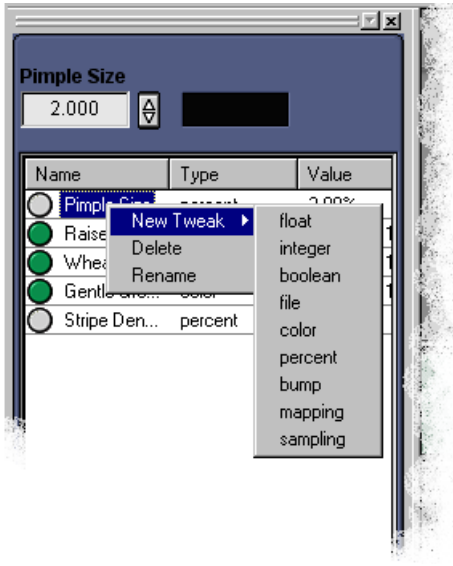


Figure 2.63
Two menus that assist in managing your Tweaks. If you choose NewTweak from the left menu, then you must select a type for it, using the right menu.

The following steps describe how to create a new *Tweak* with the *Tweaks Editor.* You can also do the same thing from within any *Component Editor.* That's covered in *"The Tweaks Click Pad & Linkable Parameters" on page 86*.

1. Right-click once on any name on the *Tweaks list*. If the list is empty, you can click on the white space. This calls up a small popup menu. If you open the menu from the *Tweaks list*, it will have three options. If you open it from the white space, you'll see just one option.

2. From the menu, select ***New Tweak***, which automatically opens a list of all eight parameter types. *Figure 2.63* is a screen capture of these two menus.

3. Left-click once on a parameter type to select it. For example, if you want to create a mapping type *Tweak*, select ***mapping*** from the parameter list. After you've chosen a parameter type, a dialog window opens so that you can give the *Tweak* a unique name. *Figure 2.64* is a screen capture of this dialog.

4. Now that you've given the new *Tweak* a type and a name, you should see it listed with the other *Tweaks* (if there are any). The next paragraphs explain how to assign a value to a *Tweak.*

Assigning or Adjusting a Tweak Value

To assign a value to a new *Tweak* or to modify an old one, first select the *Tweak* name from the list. Once you've done so, the *Tweaks Editor* will load the correct parameter type and current value onto the *selected Tweak area* shown in *Figure 2.61*. For example if the *Tweak* type is boolean, the *Tweaks Editor* will load a check box.

Since setting a color value is a little different than assigning other value types, we'll cover it first.

Setting Tweak Color Values: To change a color, you can either click on the ***Edit*** button to open the *DarkTree Color Browser,* or simply change the HSV values by dragging the cursor over the color swatch. No matter the method you select, you'll see the current color reflected on the color swatch. See *"The Parameter Input Formats" on page 88* for tips on changing the current color vaues with the cursor. Read *"Creating Your Own Active Colors" on page 106* to learn how to change a color in the *DarkTree Color Browser.*

Setting All Other Values: All other types of *Tweaks* values are set exactly as you would a parameter value in a *Component Editor.* That is, use the *zip slider* or type in a value.

Tweaking a Parameter

You cannot assign a parameter to a particular *Tweak* within the *Tweaks Editor.* To make such an assignment, you must find the component that has the parameter you want to tweak and open its *Component Editor*. From the *Component Editor,* left-click on the *Tweaks click pad* of the target parameter. A popup menu opens with a list of *Tweaks* of the correct parameter type. Simply select the *Tweak* you want to use. *"The Tweaks Click Pad & Linkable Parameters" on page 86* gives more detailed information on tweaking a parameter.

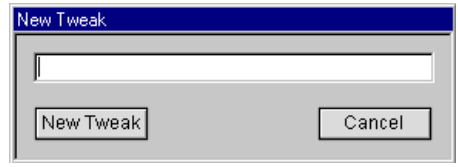


Figure 2.64
The New Tweak dialog where you can type in a unique name for a new Tweak.

Renaming or Deleting a Tweak

First right-click on the *Tweak* name you want to affect. This will open the first popup menu shown in *Figure 2.63.*

To Rename a Tweak: Select ***Rename*** from the popup menu just mentioned. Or you can double left-click slowly on the *Tweak* name. In either case, if you're successful a box will enclose the current *Tweak* name. Just type a new name in the box. This works just like a standard file browser.

To Delete a Tweak: Select ***Delete*** from the popup menu just mentioned. Or simply select the *Tweak* you want to remove and hit the ***Delete*** key on your keyboard. In either case, you can use ***Ctrl + z*** to recall the *Tweak* if you became a bit too free with your deletes.

2.3.10 The DarkTree Color Browser

Overview

The *DarkTree Color Browser* makes it easy to explore **RGB** and **HSV** color spaces or modes. Included with the *Color Browser* is an option for selecting colors directly from an external image as well. Loading an image makes it simple to lift colors directly when you're trying to match a particular one. For this discussion please refer to *Figure 2.65*, next, for pointers to the parts we'll cover in this section.

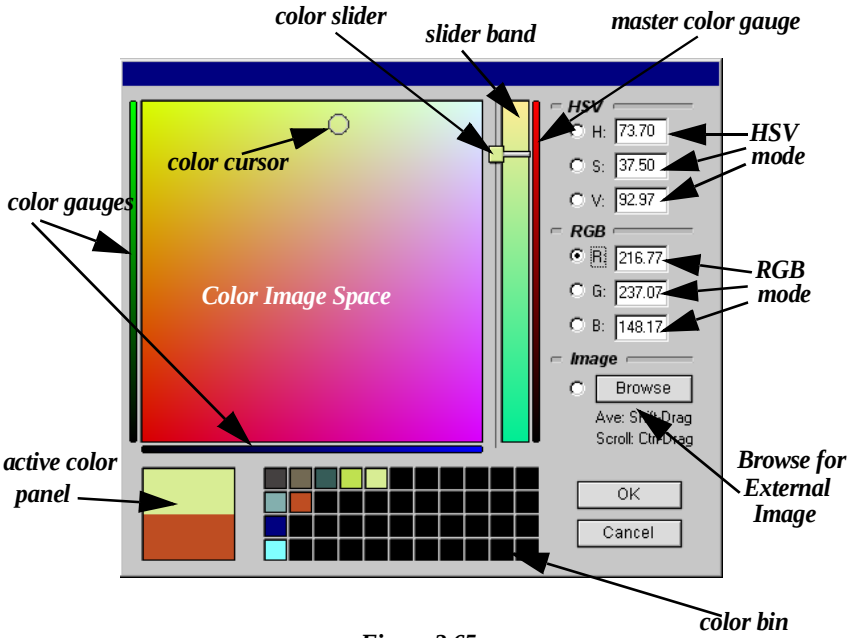


Figure 2.65

Key controls for the DarkTree Color Browser are named and located in the above image.

The Color Image/Space - Setting Up the Colors

Modes, Radio Buttons & Edit Boxes

The *DarkTree Color Browser* gives you a choice between two modes: **HSV** (hue, saturation, and value) and **RGB** (red, green, and blue). Each mode mixes colors differently. Visually think of a mode as a three-dimensional block of colors. The Z-axis represents the channel you've selected as the control; while the X- and Y- axes represent the two remaining color channels. When you look at a color mixture, you're simply seeing an XY slice through the **RGB** or **HSV** color block. You can choose a new *color channel* from either mode at any time, and the *Color Browser* will always find and display the slice that includes the current active color.

On the right side of the *Color Browser*, you'll see, running vertically, seven radio buttons and six edit boxes. The seventh button is for loading an image. Let's ignore images for the time being; we'll cover how to retrieve colors from those later in this section. That leaves six radio buttons, one for each letter (called a *color channel*). The *color channel* you select determines the mode, and therefore the color mix, you'll see displayed in the *color/image space*. The *color channel* you select becomes the control for the display, while the other two channels of the same mode vary in the *color/image space*.

NOTE: All of the edit boxes are dynamically updated whenever you move the color cursor. You can type values directly into the edit boxes as well, if you need a specific color and know its values.

The Color/Image Space & The Color Cursor

The *color/image space* displays a subset of the available colors. The small, open circle within the *color/image space* is the *color cursor.* The location of the *color cursor*, whether static or "on the move", defines the current active color. "*The Active Color Panel"* paragraph gives you more information on the current active color.

Anytime you click the mouse button within the *color/image space*, you activate the *color cursor.* You can select new colors in one of two ways: (1) by holding the mouse button down while moving the *color cursor* to another location, or (2) by simply clicking the mouse button once, anywhere in the *color/image space*. When you release the mouse button, the *color cursor* becomes static, again marking the location of the current active color. If you use the second method, the *color cursor* will jump to the latest spot.

The Color Gauges

The *Color Browser* has three *color gauges*, one for each channel of your selected mode, and each resembling a long narrow tube. Each *color gauge* displays the range for one color channel. For example, a gauge might display the levels of color saturation or the entire range for one color, depending on the current mode. The purpose of the color gauges is to help you keep track visually of the current levels for each color channel of the current mode. The gauges only change colors when you select another channel of either mode. To understand the gauges, it might be easiest just to use an example. So for our example, let's select **R** (red). *Figure 2.65* shows the *Color Browser* with **R** selected.

As soon as you select **R**, the color red becomes the control. And the *color/image space* now displays colors only in **RGB** mode. The gauge to the right of the *color slider* always displays the control color range. In our example, you'll see the entire range of the color red. The position of the *color slider* marks the current red level. Notice, as you move the *slider* to subtract or add more red, how the color mixture in the *color/image space* changes.

Using our example, the *color/image space* displays all green/blue color combinations that are possible, given the current red value. The *horizontal gauge* that runs just below the *color/image space* displays all blue levels. Moving the *color cursor* horizontally will add/subtract blue from the current active color. That is, the amount of green remains constant. The *vertical gauge*, directly to the left of the *color/image space*, shows all green levels that are possible. Moving the cursor straight up or down will add or subtract green from the current active color. Moving the cursor on an angle will, of course, change both the blue and green levels at the same time. Red, our control color for this example, will remain constant until you change it with the color slider.

The color gradient on the *slider band* displays all of the colors possible, given the position of the *color slider* in relation to the *control color gauge (*currently displaying red values) and the fixed values for both green and blue. The current active color provides those fixed green and blue values. Look at it like this. The color cursor lets you change both the blue and green levels, while the red level is held constant. The *color/image space* displays the resulting color mixture. The *color slider* lets you change the red level, while holding both green and blue constant. The color gradient on the *slider band* represents this result.

Creating Your Own Active Colors

The Active Color Panel

You'll find the *active color panel* below the *color/image space* and to the left of the *color bin.* Initially this panel is a single color; it's the color of the parameter you opened the *Browser* with. The lower half of the *active color panel* always displays the

original color. It remains there so that you can easily return to it simply by clicking on it, if you wish. Select a new color in two ways: by releasing the mouse button after moving the color cursor around in the *color/image space,* or by simply clicking anywhere within this space. As you choose new active colors, you'll see each one displayed on the upper half of the *active color panel*. Clicking on a color you've previously saved in the *color bin* will make it the active color again. When you close the *Color Browser,* whichever color is currently active will be your parameter color. If you decide to keep the original color, either click on the lower half of the *active color panel* or simply select ***Cancel***.

The Color Bin

The *color bin* is directly to the right of the *active color panel*. The bin lets you store commonly used colors or those you've gathered together for a particular DarkTree project. You can add a new color to the bin only when it's the currently active one, that is, when it's displayed on the *active color panel.* To add a color just click on it and, holding the mouse button down, drag it directly to a bin slot and drop it in place. If you want to save the original color, you can add it in the same way. As a color is being dragged to the bin, you'll see it represented visually as a neat little packet. Seeing the packet icon always insures that you've grabbed the color successfully. Replace a color in the bin by dragging and dropping a new color on top of it. If you want to rearrange your bin color collection, simply drag each color to a different empty slot. You can switch two colors as well, by simply dropping one on top of the other. This causes the displaced color to jump to the open bin slot vacated by the first color.

The bin color collection is persistent, remaining intact until you change it.

Retrieving Colors from an External Image

Loading an External Image

Use the ***Image*** radio button to load an external image. Working with an image can be extremely useful if you're trying to match colors from, for instance, a certain type of stone or a specific piece of wood. Once your image is loaded, you're set to select either discrete or averaged colors for your DarkTree project. Note that the radio button selects itself when you load an image (if you haven't already chosen it).

Start by clicking on the ***Browse*** button, which summons a standard file requester. Accepted file formats are the same as those that are legal for the DarkTree Bitmap Renderer. When the image is loaded initially, it will sometimes look fuzzy and downright unusable. If that happens the ***Fit*** option is probably selected. With ***Fit*** checked the image is automatically fitted to the *color/image space*, regardless of its dimensions. A new image is always loaded using the display option you selected last. To view and change these options, right click in the *color/image space* to pop up a set

of image display options. The options are*:* ***Fit***, ***100%***, ***200%***, ***400%***, and ***800%***. The percents refer to percent-of-zoom. Selecting ***800%*** will make it easy to see the color on each pixel. ***100%*** usually looks the most natural because it isn't zoomed.

However no matter which display option you choose, you'll probably find that often the entire image won't fit in the *color/image space*. So you'll need to move the image around to see all of it. To move your image to a part not currently visible, hold down the ***Ctrl*** key and left mouse button together, and simply scroll.

Selecting Image Colors

Select colors from an image as you would select a solid color. That is, either click on a color or move the color cursor around while holding down the mouse button. To average or mix image colors in a given area, hold down the ***Shift*** key and left mouse button together, and "scrub". Just as with a color mode, the changing (averaged) color is always visible on the upper half of the *active color Panel* as the current active color. Release the two buttons to stop the averaging operation.

When you're working with images, the edit boxes for both color modes are kept dynamically updated with image color values. You can collect image colors in the *Color Bin* or jot down the numerical values of colors you want to remember. And finally, you can select and de-select an image by clicking on the ***Image*** radio button (to select) and clicking on a color channel button (to de-select). The *Color Browser* has a long memory. It will remember the last image you loaded, even after you've closed DarkTree Textures.

2.3.11 The DarkTree Editor Tutorials

Tutorials Overview

The following three tutorials teach the basic skills you'll need to begin successfully building DarkTrees. The aim here is to give you sufficient knowledge to free you from reading large amounts of the Interface Reference. That way, you can save the manual reading for answering specific questions as they arise. To polish your skills and understand some of the finer points of texture creation, be sure to run through the several short tutorials in the Conceptual Section as well.

The tutorials here are arranged in steps, and because they teach basic DarkTree user skills, often include further comments that, we hope, add more clarification. If something is completely unclear, you can always look it up in the index. Keep in mind that each tutorial builds upon what you learned in the previous one.

Tutorial One - Building a Simple Wood DarkTree

For the first Editor tutorial, we'll create a simple three-component wood texture with some additional streakiness and a suggestion of pores. It's possible to make a surprisingly nice-looking wood with just three components. Later you can build extra features into the texture, making it even more interesting.

Skills:

- *opening a new DarkTree Editor*
- *choosing components from the Component Bin and plugging them into the workspace sockets*
- *opening the Component Editors and changing parameter values*
- *loading an external image into the Color Browser and "lifting" colors from it*
- *transforming a texture*
- *linking selected parameters to selected components of the same type*
- *viewing texture results in an Examine Window*

Tutorial One - Grimalkin Wood

1. Start DarkTree Textures, and after the main application window comes up, select ***File>New*** from the *main application menu*. ***New*** will open an empty DarkTree Editor.

Comments: *You can maximize/minimize the Editor window by grabbing an edge or corner, and holding the left mouse down, pulling the window to the size you want.*

2. Your first task is to learn how to drag a component onto the *workspace* (the gray area) and plug it in to a socket. On the left side of the Editor is the *Edit Control Panel*, which has three tabs along the bottom. Be sure the ***Components*** tab is selected, so that you can see the *Component Bin*. The bin has several folders, and depending on the Preferences, one or two might be open already. The folder names are the classifications for groups of similar components.

First open the *Shader* folder. This folder has just one component in it, the DarkTree Shader. Click on this component's name with the left mouse button, and holding down the button, drag the component to the **Root** socket. Simply release the mouse button to plug in the component.

3. We have two components left to install. So first find and open the *Natural* folder, and locate the Lumps component. Following the instructions in step 2, drag it to the *workspace*, and drop it over the socket immediately to the right of **Root**. Before the

Editor will plug in this component, it calls the ***Paste As*** dialog to find out which of the three key DarkTree types you want Lumps to be (color, percent, or bump). Select **[C]olor**, and the Editor will plug Lumps into the socket forthwith.

The final component for our DarkTree is called Agate. You'll find it in the *Natural* folder as well. Locate and drag Agate to the socket directly to the right of Lumps. Again, select **[C]olor** from the ***Paste As*** dialog. You could use the Birdshot component rather than Lumps, but only if you're making a 2D texture. Lumps is the 3D version of the same pattern as Birdshot.

Comments: *The DarkTree Editor was able to plug the Shader component into the Root socket without first asking for a key type, because shaders are a class in and of themselves. They're designed exclusively for the Root socket, having links for many subtree types. You can statically set several shader attributes as well (glossiness, transparency, and so forth).*

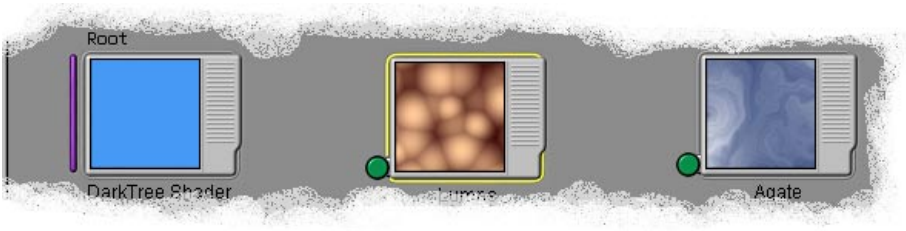


Figure 2.66
The three components that will become the wood texture.

4. The three components we need for this texture are all in place now, and *Figure 2.66* is a screen shot of them in place. The next step is to work with the component images and make them woody-looking. The color Agate component will provide the basic wood texture; color Lumps will supply additional streaks and a suggestion of wood pores. We'll make the texture a shader by linking Agate and Lumps to a DarkTree Shader component.

Let's work with Agate first. To begin, open Agate's *Component Editor* with a double left-click on its image. The open *Component Editor* you see lists the linkable and/or changeable parameters with their values, for this color component. For learning purposes, we'll give you the parameter values we liked for a good wood texture. But later, please experiment with different values and see what your own changes look like.

Using either the zip sliders (which function just like regular sliders) or the edit boxes, enter the following: ***Input Seed***: *14*, ***Roughness***: *65.0*, and ***Contrast***: *70.0*.

We're not done with Agate, so leave its *Component Editor* open for step 5.

Comments: *If you want extra information on what a particular parameter is meant for, click on the blue-colored* ***Help*** *button on the Component Editor. Doing so will open the online Component Reference.*

5. In this step, we'll use the *Component Editor's* ***Transforms*** section to stretch the Agate pattern until it looks like wood.

With the Agate *Component Editor* still open, go to the ***Transforms*** area (the upper middle). Click on the bottom button, labeled ***Scale***. You'll next see an enabled check box and that the Y and Z boxes are ghosted. This indicates that if you were to enter a value in the X box now, the scaling would be evenly applied in all three directions. But we don't want even scaling, so click in the check box to disable it. Type a value of *800.0%* in the Y box. This represents heavy duty scaling, but it will stretch the Agate pattern in the Y-direction so that it will approximate wood grain very well. *Figure 2.67 A & B* show before and after scaling versions of Agate. For this screen capture, we changed ***Low*** and ***High*** to black and white temporarily, simply because the grain is more visible.

Comment: *You can use the zip sliders to scale as well. Just remember that pulling up on the top of the slider will stretch the pattern out, while pulling down on the bottom of the slider will shrink the pattern, making it look smaller and busier. If you scale down too heavily, beware of aliasing artifacts. For even scaling, the same results apply, but to the entire texture. Keep your finger down on the scale button to see what your value is. It might take a minute to register.*

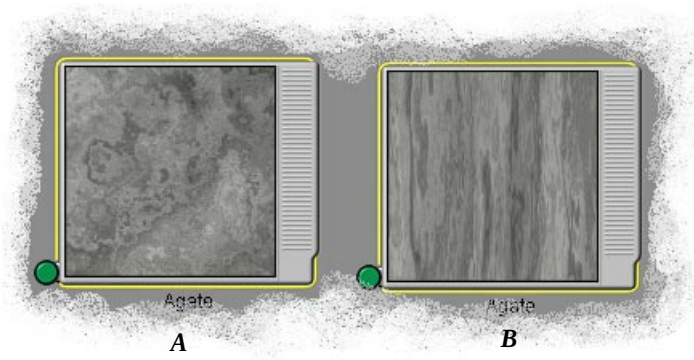


Figure 2.67
Agate before (A) and after (B) scaling in Y only. Note that, here, the image colors are black and white to make the pattern more visible.

6. With this step, we complete our work on color Agate. In the ***Parameters*** section of Agate's *Component Editor,* click on the **Edit** button for ***Low.*** This will open the *DarkTree Color Browser*. Click on the ***Browse*** button over on the right edge of the *Browser.* Look in **Textures/Tutorials/GrimWood.jpg** and open it. Now you should have a wood image loaded into the *Browser.* Move the cursor over the darkest-colored streak or area in the wood image and left-click one time. Simply for your information, the darkest color we found was RGB *129, 71, 23.* Before closing the *Browser*, use the cursor to drag a packet of dark color to the c*olor bin*. (you'll recognize the bin because it has several small squares or "bins" that are meant to hold any colors you wish to save.) Click ***OK*** to close the *Color Browser.*

And finally, click on the **Edit** button for the ***High*** parameter. The *Color Browser* should still have the wood image loaded. Find the very lightest streak or area in the image. Now go through the same process as you did in the last paragraph. The lightest color we found was RGB *200, 160, 108*.

At last you can close both the *Color Browser* and the *Component Editor* for Agate.

Comments: *To get rid of an image in the Color Browser, just click on any of the RGB or HSV radio buttons*.

7. Now we'll begin working with the color Lumps component. Again, we'll transform the pattern to better suit our purposes. It's a little harder to get Lumps looking just right, but with specific scale values and a little savvy, it should be easy enough.

First, open the color Lumps *Component Editor* as we did in step 4 with Agate. Click on the **Edit** button for the ***Lumps*** parameter and change the color to white. This is a temporary measure, but it's easier to see what the textural changes look like using a contrasting color. Next, change the following parameters: ***Input Seed***: *10*, ***Lump Size***: *5.0*, ***Density***: *10.0.*

We want to scale Lumps so that the balls are long and thin, but not so thin that they start to present aliasing problems. Let's begin by scaling the entire Lumps pattern down so that we have a good number of small, well-scattered balls. Using even scaling, enter *15.0%* or pull down with the zip slider until the pattern approximates *Figure 2.68*. Next, disable even scaling and stretch Lumps in Y by scaling up by about *1300.0%*.

Our final Lumps chore is to edit the ***Lumps*** parameter again, changing the color to the dark brown you saved in step 6.

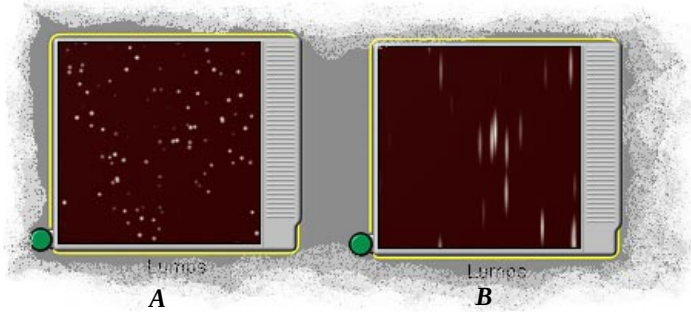


Figure 2.68

The Lumps component after even down-scaling to make the pattern small (A), then stretching the result in Y (B).

Comments: *For your information, when you're scaling unevenly, try to keep it down to one direction only. If you end up scaling differing values in two directions and the texture is 3D, you can distort it in ways you don't realize at first. For example, if you're working with a wood texture that you plan to use on a multiple-sided model, the wood end grains won't look like they should. The way to get around this is to pick one direction that you've already scaled, then match the same scaling in the third direction.*

8. Now, we're ready to link the components together into a single DarkTree shader. We'll link Lumps first.

Notice the wide ribbed area on the right side of the Lumps component frame. This is the *links pad*, and if you right-click once on it, the *links list* will open. Go ahead and do that. The *links list* has the names of all linkable parameters for a particular component, in this case Lumps. Left-click once on the ***Background*** parameter. This selects the parameter and closes the list but leaves you with a stretchy wire attached to the cursor tip. Drag the tip over the Agate component's image area and left-click again to attach. Or you can attach to the small colored circle as well. You've now completed your first link.

Comments: *The link you just made causes the wood pattern from the Agate component to be the new background for Lumps. If you look closely, you'll see that the streaks we scaled from the Lumps pattern are now part of the wood pattern.*

9. The instruction for this step completes the wood texture itself by making the final link. Open the DarkTree Shader *links list* and click on the top parameter, ***Surface Color***. Link this parameter to the Lumps component. Look at *Figure 2.69* to see a screenshot of the final tree.

Now we should view the wood texture on something bigger than a component image, so next we'll open an Examine Window and get a real look.

Comments: *Always remember that the type of the parameter from the links list that you click on must match the type of the component you stretch the wire link to. You can always determine a component's or parameter's type by looking at the color of the small circle (the type icon) associated with it. The type icon colors are: green for color components or parameters, gray for percents, and blue for bumps.*

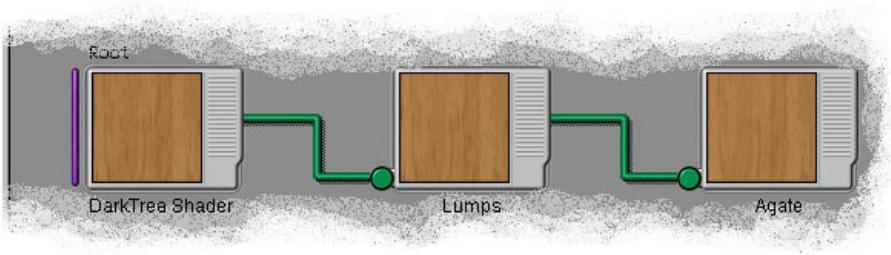


Figure 2.69
This screen capture shows the finished DarkTree for this tutorial.

10. This step instructs you in opening an Examine Window and getting a better look at what you've got. With the cursor centered over the DarkTree Shader component's image, right-click once. This opens a *component menu*. You can access a *component menu* from any component in the same way, although the options differ with the type of component.

From the menu find and click on ***Examine***, which opens an Examine Window and automatically loads the component image into it. Once an Examine Window is open, you can drag any updated image to it at any time. The Examine Window has several controls, too many to cover here, but you can turn to *"The Examine Window Viewing Controls" on page 39* to read about the viewing options.

You can find a copy of this DarkTree in **Textures/Tutorials/GrimalkinWood**.

Final Comments: *With a few simple adjustments and/or adding another component or two to your DarkTree, you can make some significant changes to the wood texture given here. For example, change the wood streaks simply by increasing the value for the* ***Lumps*** *parameter. Or try linking the Lumps component's* ***Blend Function*** *to one of the generators. Figure 2.70 is a screenshot of three quick (sorta) changes you can make to this texture. If your wood doesn't look the same after trying these variations, go ahead and try some experiments. Hint: Change*

the Agate component's Seed Input.

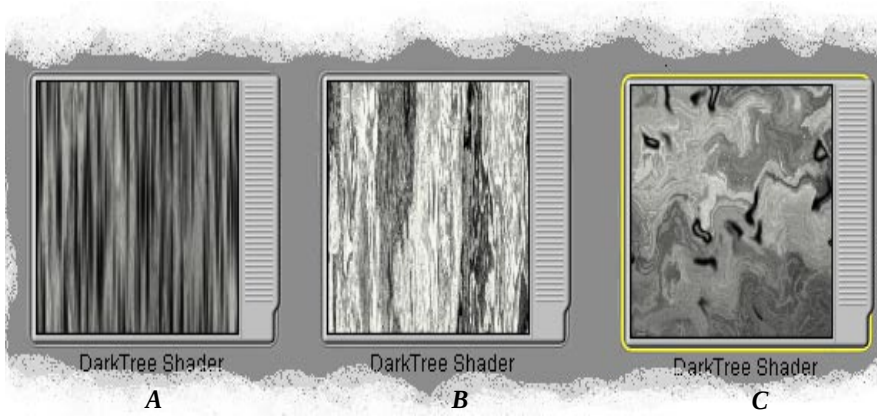


Figure 2.70

Three variations on the same wood theme. For A, the Lump Size parameter was changed to 50.0. For B, Agate's Blend Function parameter was linked to a Sine Wave generator. The third image, C, is more complex. A Turbulence component was linked to Lumps via its Background parameter. Lumps' Blend Function parameter was linked to a Bell generator. Then we changed some other parameter values as well; Agate's Input Seed is different, plus the scaling was cleared.

Tutorial Two: Building an Abstract Texture with Metallic Colors

This tutorial introduces you to the power and convenience of *Tweaks*. And, if you don't already know, shows you how easy it is to give a texture metallic attributes. Basically, we'll be combining two abstract patterns together and assigning specific colors to them, based on their multiplied grayscale values. You're also given a number of examples of how to make adjustments that can change certain aspects of the texture.

By the way, we won't re-explain those tasks you already learned in Tutorial 1, but we will reference them. At least the first time.

Skills:

- *creating Tweaks for Selected parameters*
- *making a texture metallic*
- *using a Process component*
- *changing Tweak values on the Tweaks Editor*
- *adjusting aspects of a texture using a generator*

Tutorial Two: Metallic Pastiche

1. In Step 1, Tutorial One showed you how to plug the components into their sockets. We'll use that information a lot for this tutorial, since we have seven components to plug into the Editor *workspace.* So let's get started.

After starting up a new DarkTree Editor, go to the *Component Bin* and open the Shader folder. Drag out the single component you see there, DarkTree Shader, and plug it into the **Root** socket. Next, open the Gradient folder and drag out a Mask Gradient component. Position it over the socket directly to the right of the **Root**. Paste it as color to plug it in. Fetch another Mask Gradient and drag it to the socket directly below the first Mask Gradient. Paste this one as bump.

Open the Process folder and drag a Multiply component onto the *workspace*. Drop it over the socket directly to the right of the color Mask Gradient component. Paste this component as a percent. Next, open the Generator folder and drag out the S Curve generator. This component goes to the socket directly below the Multiply component. Generator components are automatically pasted as percents.

We have just two components left to plug in. Find the Noise folder and drag out the Plasma component. Plug it into the socket directly to the right of the Multiply component. Specify percent for Plasma. And finally, from the Natural folder, fetch Agate to the socket directly below Plasma. It should be a percent as well.

All of the components we need for this DarkTree are now in position. You can see a screen shot of them in *Figure 2.71*. With the next step, we'll begin tweaking their parameters, trying ultimately for something interesting.

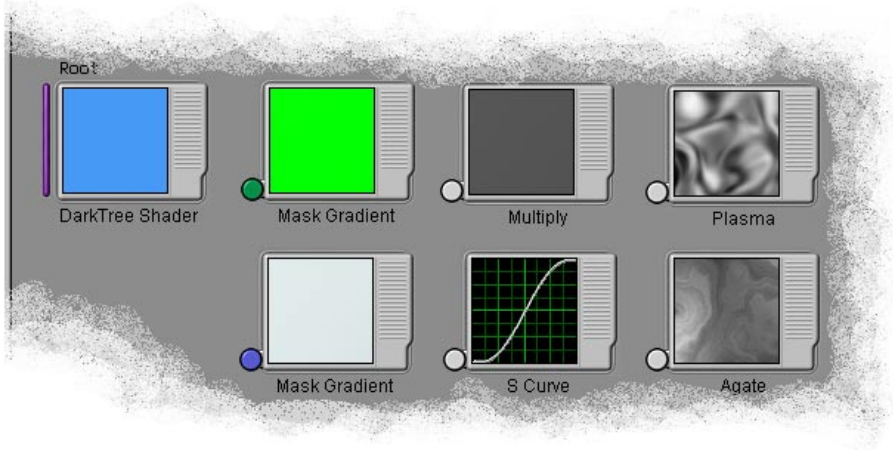


Figure 2.71
The complete set of components needed for this DarkTree.

2. In this step, we're going to begin by creating a couple of *Tweaks*. Later, you'll see how useful these can be.

First, open the Plasma *Component Editor* (covered in Step 4 of Tutorial One) and find the ***Input Seed*** parameter. Just to the left of the parameter name, you'll see a small gray, debossed square, the *Tweaks click pad*. Click on this pad and then on the ***New Tweak*** option listed on the button that immediately pops up. This should open the ***New Tweak*** dialog. Type in *Pattern Seed* for the *Tweak* name, then click ***New Tweak*** to close. You have just tweaked Plasma's ***Input Seed*** parameter. Its tweaked value is its current (default) value, but we can change that later. For now just close Plasma's *Component Editor*.

Following the instructions in the last paragraph, open the Agate *Component Editor.* Go to Agate's ***Input Seed*** parameter and, again, click on the *Tweaks click pad*. This time you'll see two options listed on the button: ***NewTweak*** and ***Tweak with***. Choose the second option, select *Pattern Seed*, then close this *Component Editor.* Now the ***Input Seed*** parameters from both Plasma and Agate are tweaked and share the same value.

Comments: *If a parameter is one of the three key DarkTree types (color, percent or bump), the Tweaks click pad will have a colored type icon on top.*

3. We have finished with Agate and Plasma for now, so let's go ahead and link them both to the Multiply component. If you need to, read step 8 again from Tutorial One.

Open the Multiply component's *links list*. Select the ***Percent A*** parameter, and link it to the Plasma component.

From the same *links list* select ***Percent B***, and link it to the Agate component. These two links combined make up our basic texture.

Comments: *The Process components, such as Multiply, do not create patterns of their own but instead modify the "look" of any linked child components. The Multiply component mixes Agate and Plasma together by multiplying the percent values from each component. The resulting values naturally never exceed 100.0%.*

4. To continue building our texture, let's next assign some new colors and create some *Tweaks* for the color Mask Gradient component. For your information, Step 6 of Tutorial One covers working with the DarkTree Color Browser.

First, open the *Component Editor* for color Mask Gradient. To simplify, we'll initially work with three colors only, but you can have as many as five. For the ***Number of Shells*** parameter, enter *3*. Using the RGB edit boxes in the DarkTree Color Browser, enter the following values for the following color parameters: ***Shell Value 1*** *= 214, 167, 85;* ***Shell Value 2*** *= 0, 150, 156* and ***Shell Value 3*** *= 127, 7, 245*. We chose gold, turquoise and violet for the colors, because they look good as metals. Keep this *Component Editor* open; we still have more to do.

Now we're going to tweak the shell position parameters, so that you can adjust the color areas later. Create a *Tweak* for the ***Shell Position 1*** parameter and name it *Gold*. Do the same for the ***Shell Position 2*** and ***Shell Position 3*** parameters, and name them *Turquoise* and *Violet*, respectively. You can close this *Component Editor* now.

Comments: *The shell position parameters are percent settings that match a color to a specific range of grayscale values. The grayscale values are supplied by the component that color Mask Gradient is linked to. For this tutorial, it's the Multiply component. By changing the shell position values, you can widen or narrow the areas for each color.*

5. We really don't need to make any changes to bump Mask Gradient at this time, but it's probably a good idea to have a basic understanding of what this component is doing. Read the next ***Comments*** section for some information on that.

Next let's finish linking the components together. First open the *links list* for color Mask Gradient, and select the ***Mask*** parameter. Link it to the Multiply component. The bump version of Mask Gradient has a ***Mask*** parameter as well. Select and link it to Multiply also. Now we have two ***Mask*** parameters linked to the same component.

Lastly, open the *links list* for the DarkTree Shader. Link ***Surface Color*** to the color version of Mask Gradient and ***Surface Bump*** to the bump version of the same component.

Comments: *The bump version of Mask Gradient assigns elevations based on the values of a linked percent texture (in this case supplied by the Multiply component). Each shell position has a percent value that corresponds to a grayscale level. The matching bump value gives an elevation for that grayscale level. And finally, the* ***Bump Scale*** *parameter gives the scaling factor that all other bump values are multiplied by. If you find you're running into aliasing problems due to elevations that are too steep (a common problem), reduce* ***Bump Scale****.*

6. For Step 6, we'll change several of the static percent-type settings in the DarkTree Shader component to give our three component colors a metallic cast.

After opening the *Component Editor* for the DarkTree Shader, change the following percent values: ***Metal Level*** *= 100.0*, ***Specular Level*** *= 95.00*, ***Diffuse Level*** *= 40.0* and ***Glossiness*** *= 5.0*.

Next open an Examine Window from the DarkTree Shader component, so you can get a better look at the texture so far. We covered Examine Windows briefly in Step 10 of Tutorial One. While the Examine Window's regular rendering will be good enough for a working view, you'll find that if you enable the antialiasing control the texture will

improve considerably. It does slow the render time down however. To enable antialiasing, right-click on the Examine Window's image area and select ***Antialias*** (what else?) from the popup menu.

Comments: *For future reference, if you wanted to create a partially metallic texture, you would first need to link a component designed to mask out the non-metallic areas. Then you would have to link a percent component for the Metal Level, and, if the metallic areas are large, at least a percent link to add the Specular Level.*

7. In this step, we're going to use the *Tweaks* you created back in Step 2 to change some parameters and consequently the abstract pattern we've been working with.

Look at the *Component Bin*, located to the left of the DarkTree Editor. Click on the bottom tab labeled ***Tweaks*** (the right most one). This opens the *Tweaks Editor* page. Now you should see a list of the *Tweaks* that you set up earlier.

From the *Tweaks Editor*, select *Pattern Seed* by clicking on either the name or *type icon*. The top part of the page always shows the currently selected *Tweak*, and here you can change its value. Try another number (integer) for *Pattern Seed*, and note how it alters the texture pattern. Drag the image to the Examine Window often to view the changes you're making. It might actually help to have two or even more smaller Examine Windows open so that you can compare pattern changes more easily. Once you find a seed that you like, we can adjust the color areas. We settled for a seed of *9*. *Figure 2.72* has three texture examples, including the one we liked best.

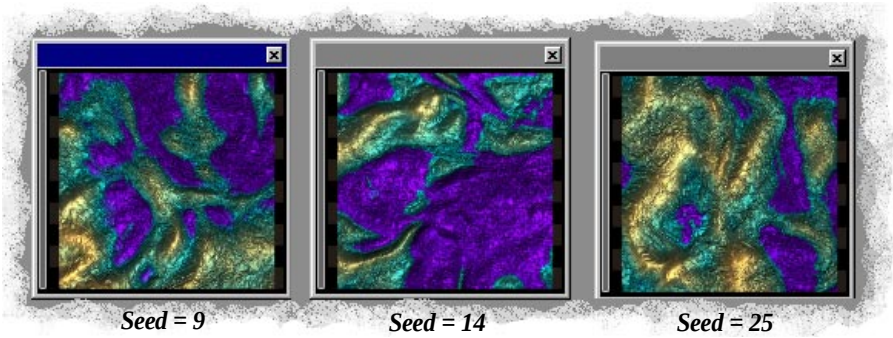


Figure 2.72
The same texture generated using three different Input Seed values.

Comments: *If you find the Agate contribution to be too rough looking for your tastes, even with* ***Antialiasing*** *enabled, go back into its Component Editor. Try lowering the default value for the* ***Roughness*** *parameter. Allowing too high a value for* ***Roughness*** *here is an invitation to aliasing problems.*

8. Let's say that you found a seed that gives you a nice abstract texture. After trying a few different numbers for the seed, you've no doubt noticed that some textures render with more equal amounts of each color than others. By adjusting the *Tweaked* values for *Gold*, *Turquoise* and *Violet*, you can get the color mix the way you like. For example, if you want more violet color showing, lower the value for *Violet*. In this case, you should keep this value higher than the next value (*Turquoise*), though. That's because ***Enable Clamps*** is checked. This toggle keeps the colors in order.

9. When you look at this texture in the Examine Window, notice that the blending between the shell colors is somewhat continuous. Let's try changing the blended area, making more discrete areas of color. First, open the *links list* for color Mask Gradient and select the ***Blend Function*** parameter. Link ***Blend Function*** to the S Curve generator. Now open the *Component Editor* for S Curve and change the ***Amplitude*** parameter. The more you increase ***Amplitude***, the narrower the blend area between each of the colors will be. Play around a little and see what happens. Try decreasing the ***Amplitude*** as well, making the colors blend together even more than they were. A value around *62.0* seems to be plenty for decreasing the blended area. A value of *24.0* produces a nice wide blending. The changes are fairly subtle unless you change the ***Amplitude*** drastically.

Figure 2.73 shows the completed DarkTree on the Editor *workspace*. You can find a copy of it in **Textures/Tutorials/Pastiche**.

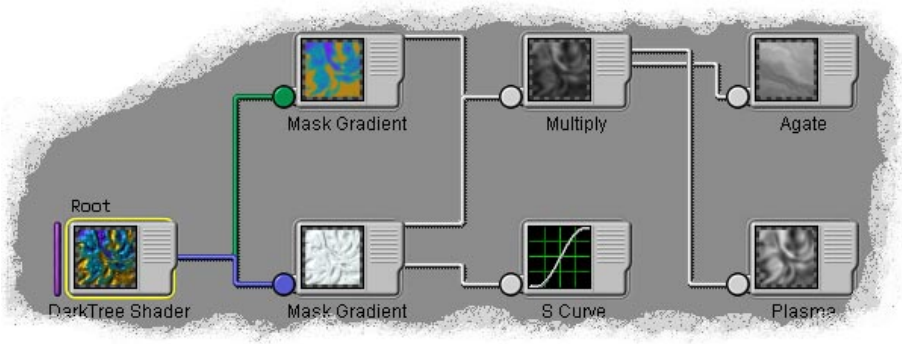


Figure 2.73
The final DarkTree for Editor Tutorial Two, Metallic Pastiche.

Comments: *With some of the seed patterns you may have noticed some of the large bump ridges are quite sharp. You can smooth those out a bit by dragging out a Bias generator, plugging it into the socket directly below S Curve and linking the bump Mask Gradient's* ***Blend Function*** *parameter to it.*

Tutorial Three: Creating an Animated Texture

The tutorial section for the DarkTree Editor wouldn't be complete without an animation tutorial. So for the third and final tutorial, we're going to build a smoke ring. The animation begins with no ring at all, just a blue background. But gradually an amorphous ring of smoke fades in to view. We'll finish by making a movie of this animation.

Skills:

- *working with a gradient cylindrical mapping*
- *using a Component Editor's update window to quickly track parameter changes*
- *animating more than one aspect in the same DarkTree*
- *changing the animation length and current frame on the Properties page*
- *creating a movie of the final animation*

Tutorial Three: A Rolling Smoke Ring

1. We'll construct this animation in two separate sections. First, we'll build the smoke ring itself and then add the rolling effect onto that.

To begin building the smoke ring, first open the Natural folder. Locate and drag the Clouds component to the **Root** socket and paste it as a color-type. Next, open the Gradient folder. Find Absolute Shells, drag it to a level-two socket (to the right of Clouds) and paste it as a percent-type. Last, drag out a Time component from the External folder, and plug it into a level-three socket. Time components are percent types only.

2. In this step, we'll begin by modifying some Absolute Shells parameters so that they'll better suit our purpose.

Start by opening the *Component Editor* for Absolute Shells. Locate the following parameters and change them to the values listed. Change ***Mapping Type*** to ***Cylindrical 2D***, ***Number of Shells*** to *3* and then disable ***Repeat Flag*** (remove the check from the check box). And finally, change the ***Shell Position 2*** and ***Shell Position 3*** parameters to make a large ring. ***Shell Position 3*** should be the maximum value of *100.0*; ***Shell Position 2*** should be about *60.0*. Leave ***Shell Position 1*** at its default value. We're finished here, so close the *Component Editor.*

3. Our next task is to link the parameter that controls cloud size to the Absolute Shells component.

Open the Cloud component's *links list* and click on ***Puff Size***. Notice that this parameter is a percent-type, even though the component type is color. Link ***Puff Size*** to Absolute Shells. Now when you look at the Cloud component image, you will see a ring of clouds. However, the ring looks overly thick for smoke. Let's work on that.

Reopen the *Component Editor* for Absolute Shells. Under the ***Views*** section, click on either the button labeled ***Root*** or the one labeled ***Parent***, in this case it doesn't matter which. The point is that we want to see the cloud ring in the *update window*, not just the gradient.

Use the zip slider (the small double arrow icon) to play around with the ***Shell Value 2*** parameter until you find a value that produces a decent-looking smoke ring. A value of around *50.0* looks good.

Comments: *It doesn't really matter what number you leave in* ***Shell Value 2*** *when you close the Component Editor, because we'll be linking it to the Time component.*

4. We have almost finished with the first section of this texture. What remains is to animate the smoke ring so that it begins with nothing, then an amorphous ring of smoke gradually fades in.

First open the *Component Editor* for Time. Notice the time graph and that the default number of frames is thirty. That's fine for now. (The frame numbers are along the bottom edge of the graph.) We want the animation to begin with no ring (*0.0*) and then gradually produce a smoke ring over the course of the animation. Ending the animation at *100.0* will give us that overly thick-looking ring by the last frame. Therefore we want to reduce the upper limit of the function to *50.0*. (Remember when we tested values for ***Shell Value 2*** in step 3?) The easiest way to change the upper limit is to first change the view of the graph so that the end points are more clearly visible. Click on the right most icon along the top edge of the graph. This will re-adjust the graph view so that you can see the end points more easily. Grab the right end point with the cursor and pull it down vertically to the graph's halfway line. Close the *Component Editor*.

The final task for this step is to open the Absolute Shells *links list* and select ***Shell Value 2***. Link this parameter to the Time component. This completes the first or smoke ring section of our animation. But let's take one more step and look at it in the Examine Window, before continuing with the second or "rolling" part of the DarkTree.

Figure 2.74 is a screen capture of the component layout so far.

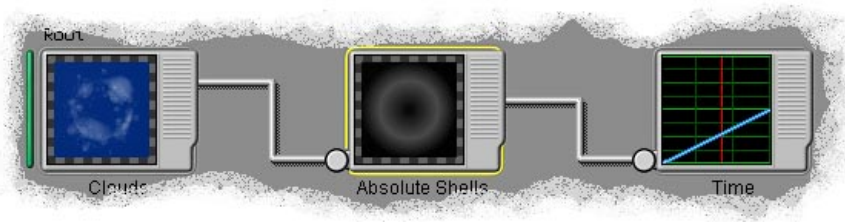


Figure 2.74
The first part of the smoke ring tutorial.

Comments: *After linking up Time, notice now that when you look at the Clouds component image the smoke ring that was there has now disappeared. That's because the current frame number for each animation is set to 1 by default. If not seeing the smoke ring bothers you, click on the Properties tab in the same place you clicked on theTweaks tab in step 7 of the second tutorial. Find the* ***Animation*** *section of the Properties page and change* ***Current Frame*** *to a number between 15 and 20.*

5. To assess the animation so far, we should view a rough render of it in the Examine Window.

Use the Clouds *component menu* to open an Examine Window, and select ***Surface Color*** from the popup dialog that appears. (Check back to step 10 of Tutorial One if you need a memory refresher on how to do this!) If you changed the ***Current Frame*** for the animation in step 4, you'll see it listed on the bottom of the Window. Using the zip slider, change that number to *1* again. Or you can type it in.

To run a *rough render* of the animation so far, just left-click on the button control with the arrow icon. The *rough render* control is second from the left, located on the bottom edge of the Examine Window. You can also step through the animation, using the zip slider.

Comments: *You might like to try an adjustment or two for practice. If so, open the Clouds Component Editor and change the* ***Density*** *and* ***Roughness*** *parameters. First, though, go to the* ***Views*** *section of the Component Editor and change the* ***Time*** *edit box to 30. That way when you try some changes, you can see what the smoke ring will look like at the final frame.*

6. Let's animate the smoke so that it rolls independently as the ring fades in. That requires a separate Time component.

But first, holding the ***Shift*** key on your keyboard down, left-click on the right edge of the component frame (the wider area). Keeping the ***Shift*** key depressed, drag the entire subtree one socket to the right.

Next, open the Transform folder and drag the Translate component over the **Root** socket just vacated by Clouds. Paste it as a color-type. Go ahead and drag out another Time component next. Plug it into the second-level socket directly above color Clouds.

7. For this step, we'll look at the parameter that supplies the rolling aspect for the animation.

From the Translate component's *links list*, find and link the ***Background*** parameter to Clouds.

Next, open the *Component Editor* for color Translate. Notice how the ***Z Amount*** parameter is set to *1.0*? As the ***Percent Moved*** parameter changes from 0.0% and up to 100.0%, the component or subtree image that Translate is linked to will vary in the Z direction (into/out of the screen) up to the value given to ***Z Amount***. This action will supply the rolling effect.

While the *Component Editor* for Translate is still open, play with changing the ***Percent Moved*** value in the same way you did with ***Shell Value 2*** in step 4,. Watch the *update window* as you change ***Percent Moved***, just to get an idea of how the image changes.

And now, let's make the final wire link. Open the *links list* for Translate again and select ***Percent Moved***. Link this parameter to the Time component, thus animating (finally) the rolling smoke effect.

Figure 2.75 is a screen shot of the completed DarkTree.

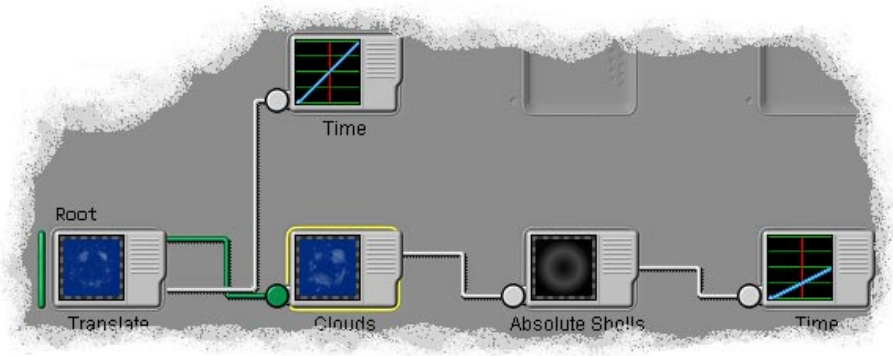


Figure 2.75
The completed DarkTree for the Rolling Smoke Ring tutorial.

Comments: *As you can see simply by looking at the function image on the second Time component, it's just a straight line that moves from 0.0% to 100.0% over the course of the thirty frame animation. That is exactly what we want, so there isn't any need to open the Component Editor for Time. Just for your own information however, you can change the animation's path to be whatever you want just by manipulating the Time component's spline graph.*

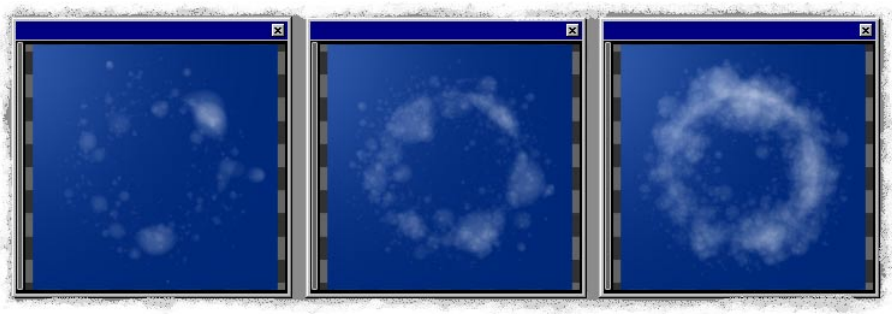
8. We should take a look at the completed animation; after all it did take some energy to make it. But, this time, let's create a movie of it - make it look a bit more professional.

Following the instructions in step 5, open a new Examine Window, or use the last one. If you use the one from step 5, drag the root component's image onto the Window, again selecting ***Surface Color*** from the popup dialog. And you might want to run through a *rough render* of the completed animation before we make the movie.

The movie's animation finishes very quickly with only thirty frames. However we can improve the animation just by doubling the number of frames. To change the animation length, open the *Properties page* as you did in the step 7 comments section, and change ***End Frame*** to *60*. A popup window will query you about rescaling. Click ***OK*** when it does. When you make changes to a DarkTree, such as we just did, always drag the latest root image to the Examine Window.

And now, let's make a short movie of our animation. On the Examine Window, select the first button control located on the bottom left. This opens the ***Create Movie*** dialog. Select ***AVI*** as your format. ***Width*** and ***Height*** you might want to lower to *100* each, since it will shorten the time it takes to make the movie. If you want to keep the movie, enter a unique name in the ***Save As*** box. And last, click ***Continue***. The next dialog that appears allows you to select a compression format. Systems don't always have the same choices, so just use the default scheme, and click ***OK***.

Your can find a copy of this animated DarkTree in **Textures/Tutorials/Rolling Smoke Ring**. The next screen capture, *Figure 2.76*, shows three frames from the animation.



frame 25 ***frame 37*** ***frame 54***

Figure 2.76

A few frames from the Rolling Smoke Ring animation.

Section 2.4
The Bitmap Renderer

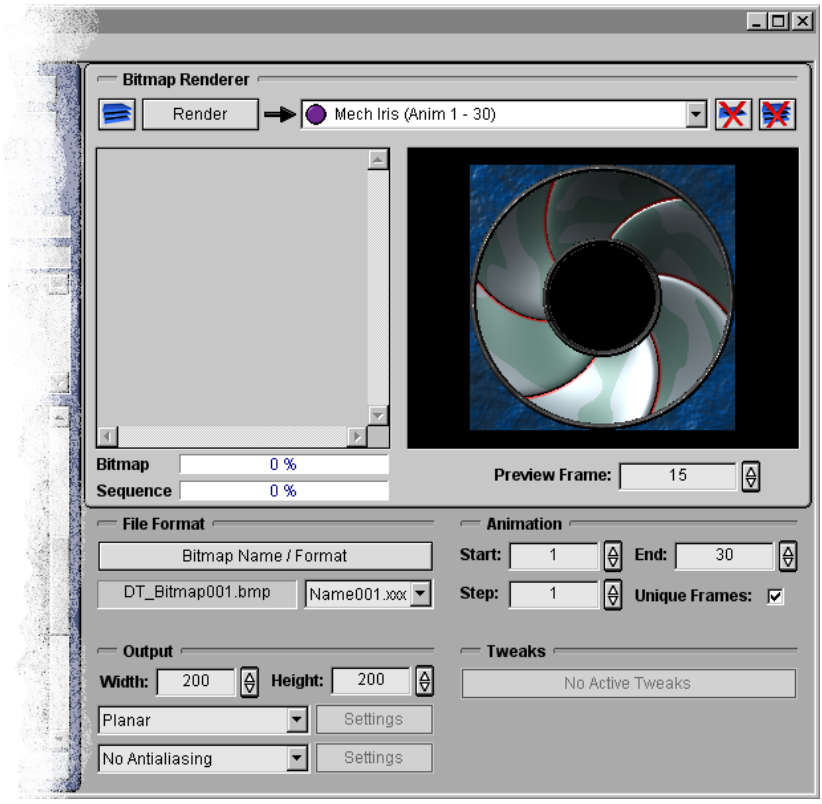


Figure 2.77
A screen capture of the DarkTree Bitmap Renderer.

2.4.1 Bitmap Renderer Overview

Overview

The Bitmap Renderer, fourth and final part of DarkTree Textures, handles the task of generating the texture maps. The Renderer, along with the Texture Library, makes up the main application window. You can see a screen capture of the Renderer in *Figure 2.77*.

DarkTree Textures version 2.0 has added some advanced features to the Bitmap Renderer. In addition to the standard 3D mappings, the Renderer now includes a simple-to-use API so that you can program your own customized mappings. And you can now change a component's attributes on the fly, through its tweakable parameters. Another new option allows the Renderer to dynamically split out and separately render each linked channel of a shaded DarkTree. The Renderer now includes a very large number of additional bitmap file formats as well.

The Bitmap Renderer separates logically into several areas of control, such as *File Format*, *Output file options*, and *Animation* to mention just a few. Our discussion for the Bitmap Renderer will cover these areas in the order that you'll most likely use them. First, though, read the paragraphs on procedural texture maps versus procedural surfaces that follow next. They explain when you'll want to use the Bitmap Renderer for your output and why.

Procedurally-Generated Texture Maps VS True Procedural Surfaces

If one of our Simbiont plug-ins was designed for the modeling/rendering application you generally use, you'll be able to import DarkTrees as dst* files. The next two paragraphs discuss the pros and cons of using DarkTrees as either procedural shaders or as procedurally-generated texture maps. Certainly both have the advantage under certain circumstances.

We have designed the Bitmap Renderer to write your DarkTrees out as texture maps. Although a texture map, also called a bitmap, is static, it does involve less render time. That's only because it will be rendered just once though. And it allows you to add some hand-painted effects as well. Texture maps must be wrapped or projected onto 3D objects, which almost always introduces stretching and seaming. However, with texture maps you're less likely to get aliasing artifacts, although you do sometimes get blurry or pixelated results. Texture maps do use up substantially more memory. But time constraints can often make texture maps the better choice because of their extra speed.

Procedural shaders are true 3D volumetric textures. You don't need the Renderer for procedural shaders since they're imported directly, via a Simbiont, into a 3D application as standard DarkTree files. Procedurals will impact render time because they're dynamic - rendering each frame of your animation (for example) on the fly each time your 3D scene renders. Consequently, procedural shaders have the superior render quality. Because procedural shaders are 3D volumes, you'll never have problems with tiling or mapping seams. You also have the option of tweaking the shaders within your 3D application, making them even more flexible. Because procedural shader files can be very detailed, you might get aliasing or scintillation problems. These problems can be fixed by making procedural textures detail-appropriate for the rendering size and by using antialiasing. And shaders require less memory than the texture map files. Best of all, procedural shaders are dynamic,

meaning they can react to scene conditions and be key framed to change over time. If you have the time and a Simbiont plug-in that matches your modeling application, the best results definitely rest with procedural shaders.

2.4.2 Setting Up a Texture for Rendering

This part of the Bitmap Renderer covers the most information and is, therefore, the largest. It takes you from loading textures through setting all of the basic parameters you'll need to specify before actually beginning a render. Note that most discussions include a figure that isolates just that part of the Renderer we'll be covering.

Loading Textures onto the Bitmap Renderer

You can load textures onto the Bitmap Renderer in several ways. First, from the DarkTree Editor open a *component menu* from nearly any component and select ***Send to Bitmap Renderer***. The component you choose, and whatever subtree is linked to it, will be automatically loaded onto the Renderer. Most of the time you'll want to select the root component so that the complete DarkTree will be rendered.

Or you can simply drag a component preview image from the Editor. Again, you'll usually want to drag the root component so that the entire DarkTree is included. When you drag a texture, be sure you drop it on the *image preview window* (discussed shortly). For a third way of loading a texture, place the cursor on a component on the Editor *workspace* and hit ***Ctrl + r*** or, if you want to iconify the DarkTree Editor as well, use ***Ctrl + Shift + r***.

And last, you can drag a DarkTree file straight from the Texture Library list onto the Bitmap Renderer's *image preview window.*

NOTE: Since complete DarkTrees, subtrees, and single components are all legal input for the Bitmap Renderer, the terms "DarkTrees", "trees", and "textures" are used interchangeably to mean acceptable input.

NOTE: You won't be able to drag any of the generator components or instances of the Time or Audio Wave components onto the Renderer.

Shaded & Single Channel Render

A “shaded” DarkTree or “shader” is called that because it has a DarkTree Shader component for its root. Shader files are easily recognized because of the purple *type icon* associated with them. When you load a shaded DarkTree, the first thing the Renderer wants to know is, “Do you want to render this DarkTree as shaded or do you want its linked channels to be split out and rendered as separate texture maps?” To get your answer, a small dialog that has these two main selections will popup. *Figure 2.78* is a screen capture of this dialog.

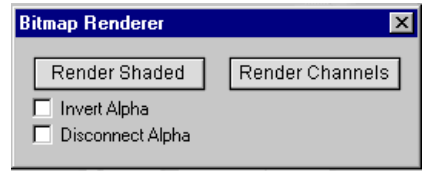


Figure 2.78
The first dialog that pops up when you load a shaded texture into the Renderer.

Clicking on the ***Render Shaded*** button loads the shader onto the Bitmap Renderer and renders it in the *image preview window.* In the *Figure 2.78* dialog are two small check boxes, ***Invert Alpha*** and ***Disconnect Alpha***; they are associated with the shaded render button only. If the check boxes are not grayed-out, the Renderer has detected a linked alpha channel. Check ***Invert Alpha*** is you want to invert the alpha channel; if you don’t want the alpha channel to affect the color channel, check ***Disconnect Alpha***.

Clicking on ***Render Channels*** means you’ve chosen to split out the shader channels instead, and opens the *Render Channels* dialog. (View a screen capture of this dialog in *Figure 2.79*.) *Render Channels* has all the same basic render settings as the Bitmap Renderer. The idea here is that by selecting a filename, file format, mapping type, and so forth on this dialog, you are able to choose the render settings for all channels at one time. If you use the Bitmap Renderer to do the same thing, you must select the options for each channel file you wanted rendered, separately. Since the settings are exactly the same as those on the Bitmap Renderer, each will be covered as part of our general discussion on the Bitmap Renderer.

Notice in*Figure 2.79* that the top two-thirds of the dialog space is taken up with a list of the DarkTree Shader channels. The Renderer is able to detect which channels of a particular shader are linked. Looking at the screen capture in the figure, notice that most of the channels are grayed-out (meaning no link). Those that do have a link, in this case ***Surface Color*** and ***Surface Bump***, have a check mark by the link name as well.

The text in the **Channel Suffix** box, "*_color*" and "*_bump*", will become part of the file name when the channels are finally rendered. If you don't need a texture map of a specific linked channel, simply click in the check box to disable it.

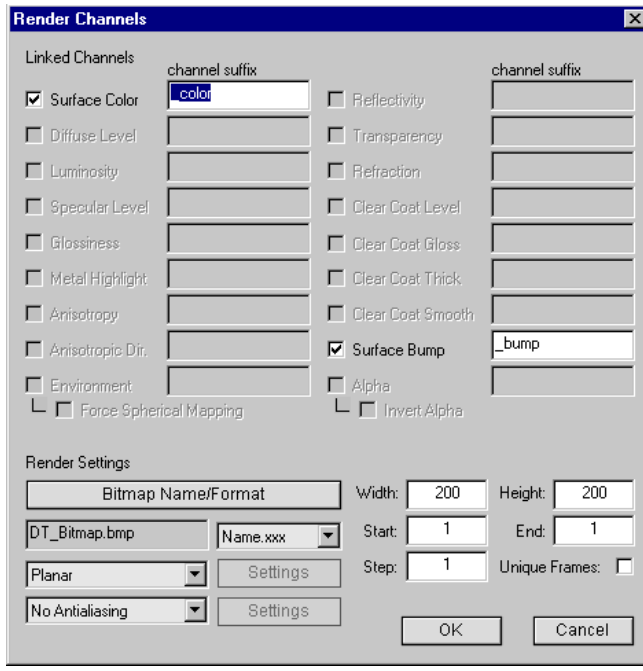


Figure 2.79
A screen capture of the Render Channels dialog. For this texture, the Renderer has detected two linked channels: color and bump.

The *Render Channels* dialog records the full shader compliment of eighteen channels. In addition, if the DarkTree has an ***Environment*** link, you can enable ***Force Spherical*** mapping for it. Forcing spherical mapping for this channel does not affect the mapping mode you have set for the other channels. And if the Renderer detects an alpha channel, you can enable ***Invert Alpha***. (Remember that the alpha options on the first dialog window, shown in *Figure 2.78*, are strictly for shaded renders.)"*Notes on the DarkTree Shader Channels" on page 45* has a short description of each channel's attribute.

NOTE: An example of a single channel output file name is: "MyTexture_Color.bmp", for a color output texture map.

Texture and Render Previews

The Image Preview Window

Located on the right half of the Renderer is the *image preview window. Figure 2.81* is a screen capture of this part of the Bitmap Renderer. To successfully load a texture, you must drag it directly onto this window (unless you're using the keyboard key method or a *component menu*. If the currently selected mapping mode in the Renderer is spherical, cylindrical, or cubic, the image will be an "unwrapped" view. And when you actually begin the rendering process, the *image preview window* will redraw the current texture image as the file is being written out.

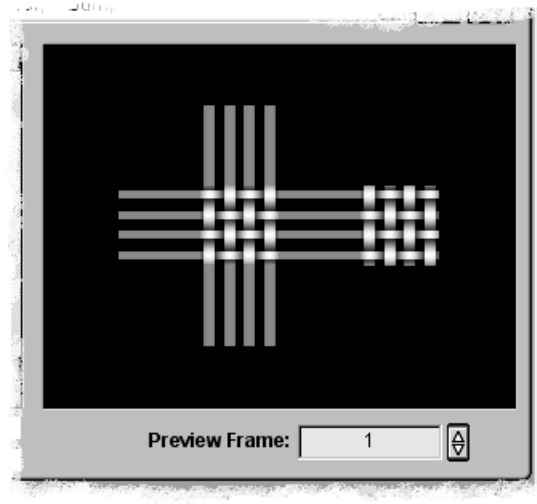


Figure 2.81
A screenshot of the image preview window and Preview Frame control. The loaded woven texture is ready for rendering in cubic mapping mode.

If you have a list of several DarkTrees ready for rendering, the *image preview window* will render the last one you loaded. If the currently selected DarkTree is to be rendered as a fully shaded bitmap, all of the DarkTree Shader attributes will be present on the rendered image. If, on the other hand, the DarkTree is a single type, a subtree, or even a single component, the image will be rendered raw (unshaded).

Figure 2.81 shows a raw cubic render of a bump DarkTree. The varying bump heights are represented by grayscale values.

The Preview Frame

You can preview a different frame of an animation by entering the number here. The frame number is constrained by the start and end frame.

When first loaded, the frame number will be the one that was set on the *Properties page* of the *Edit Control Panel.* You can brush up on *Properties page* information by turning to *"The Properties Page" on page 53.*

Output File Format

This segment of the Bitmap Renderer, seen in *Figure 2.83*, has the options for selecting a file format, a name and a location for your texture maps.

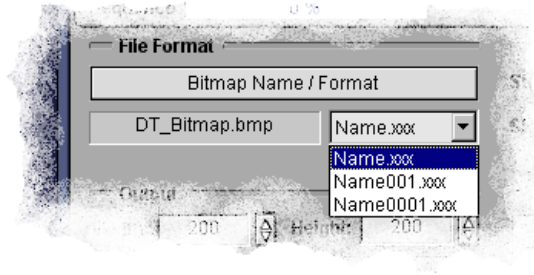


Figure 2.83
A screenshot of the File Format segment of the Renderer, where you can specify a name, location, and file format for your output.

Bitmap Name / Format Button

Clicking on this long top most button calls up a standard file browser dialog. Here, you can enter an output location and unique name for the texture map(s), and select a bitmap file type from the drop down list. The Renderer now supports a large number of output file formats. You'll find a complete list of these formats on *Appendix A, page 221*. Some extensions include a second drop down list titled ***Subfile Type***. A few extensions let you set a ***QFactor*** (quality factor). Quality factor values range from *2* to *255*, where *2* is the highest quality and *255* is the most compression.

Name Format Drop Down

The name format drop down lists some choices for the output file format. (Notice that this list is open in *Figure 2.83*.) Selections include two formats that will add consecutive numbers to each output file in a sequence. These are meant for animations, for example: "*name001.xxx*", "*name002.xxx*", "*name003.xxx*", and so on. If you're simply rendering a single texture map, you'll prefer the simpler "*name.xxx*" format.

Example Format Text Box

To the left of the *name format drop down* is a non-editable text box. It's there simply to remind you of the output filename and extension you currently have selected. This box will dynamically update whenever you make a change in name or extension.

Output Specifications

Figure 2.84 is a screen capture of the Renderer's *Output* segment. Here, you can enter the specifications for the output texture maps.

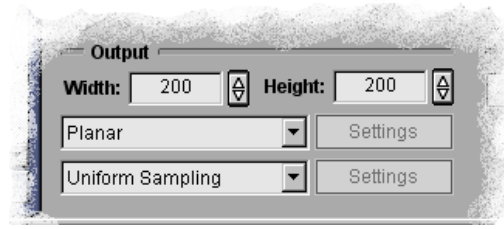


Figure 2.84
The Renderer's Output segment, where you can specify output size, mapping mode and antialiasing scheme for your bitmaps.

Width & Height Specs

The ***Width*** and ***Height*** boxes let you specify a size for your texture map(s). These two parameter values are always in pixels. The minimum size is sixteen pixels for each direction; the maximum size is constrained by your system memory. Try to select a size that will most closely approximate the size of the object you want to apply the texture map to.

If the size you select isn't too large, you can check it visually. So as soon as you've set ***Width*** and ***Height***, look at the *image preview window.* The Renderer will redraw the current output image there, in the new size. If the image will fit in this window, then you can rely on it being visually accurate for size. However for larger sizes, the image will be shrunk until it fits in the window. Tab over from ***Width*** to ***Height***, then hit the ***Return*** key to force new input to take affect.

Figure 2.85
Examples of unwrapped planar, cylindrical, and spherical mapping modes.

The Mapping Drop Down

DarkTree Textures currently supports four 3D mappings: ***Planar***, ***Cylindrical***, ***Spherical*** and ***Cubic***. Try to choose a mode that most closely resembles the shape of the object you want to apply the texture map to. The mappings are each rendered

"unwrapped". This doesn't have any effect on planar mapping, but for the other three modes the image is pre-stretched/shrunk to fit the intended shape. To illustrate unwrapped mapping differences, *Figure 2.85*is a screen capture of the planar, cylindrical, and spherical modes. You can see a cubic mapping in *Figure 2.81*.For cubic mapping, you can click on the ***Settings*** button and open the *Cubic Output Properties* dialog (*Figure 2.87*). On this dialog, you can specify output order and a key letter or word as part of your output file name so that you can keep track more easily. ***Cubic*** mapping writes six output files for each frame. In the future, other mapping types are likely to have additional settings as well. If so, you'll be able to select the ***Settings*** button, which is otherwise grayed-out.

In addition, the Renderer supports an API, so that you can develop custom output mappings and antialiasing schemes for other shapes that interest you. You'll find information on how to set this up on on the DarkTree Textures web site (www.darksim.com)

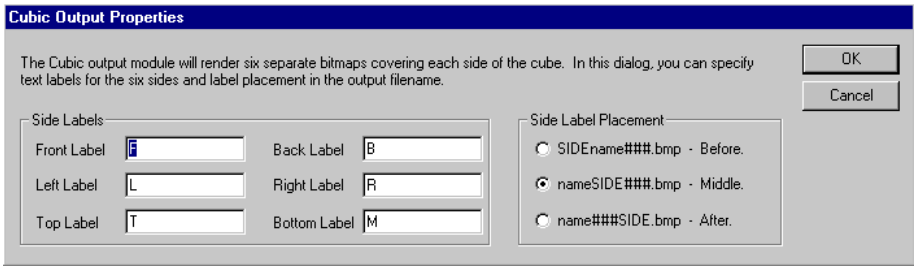


Figure 2.87

This dialog lets you choose one of three naming schemes for the six output file per frame that you'll get when you specify the cubic mapping mode.

The Antialiasing Drop Down

This drop down list has four selections for the antialiasing level: ***No Antialiasing*** and three antialiasing schemes to pick from. For antialiasing schemes you can choose from ***Jittered-16 sample***, ***Jittered-4 sample*** and ***Uniform-5 sample***. For the two jittered options, the final color is averaged from randomly-scattered samples taken from each pixel (16 or 4 samples per pixel). The ***Uniform-5 sample*** scheme is just what it appears to be. Five uniform color samples are taken for each pixel, then a color is averaged from those.

If you want to do a quick bitmap, it hardly makes sense to bother with antialiasing since it slows the process down, perhaps considerably. However for final production work, you'll probably want to choose one of the antialiasing selections so that you can get the sharpest output possible.

DarkTree Textures has an API available for anyone who wishes to program his or her own antialiasing algorithms. You can find information on how to program the API on our web site: www.darksim.com.

2.4.3 Other Texture Options

This segment of the Bitmap Renderer has the extra controls needed to set up animated output (see *Figure 2.88*). You'll also find information here on changing your tweaked parameters for the current output. Read on!

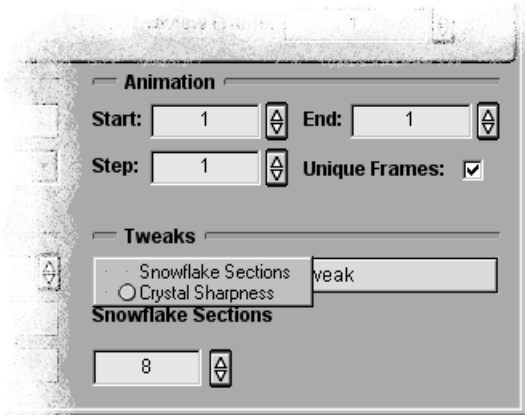


Figure 2.88
A screenshot of the special options segment of the Renderer. It includes both the animations and Tweaks controls.

Setting Up Animated Output

Start & End

The values in ***Start*** and ***End*** specify the first and last frames for an animation. As each new DarkTree is dropped onto the *image preview window*, the Renderer automatically detects and fills in these numbers. If the DarkTree is a single frame (not animated), both will be *1*. You can enter new frame numbers for ***Start*** and ***End***, as long as they fall within the actual number of frames, and ***Start*** is less than or equal to ***End.*** Doing so will result in a subset of the entire animation.

Step

Step tells the Renderer whether to render every frame of an animation, in which case it should remain at the default value of *1*, or whether to skip some frames. For example, if you give ***Step*** a value of *5*, then frames *1, 6, 11*,... and so on will be rendered.

Unique Frames

This control, when enabled, will render a frame only if it's different from the previous frame in an animation. For example, suppose you want to render an animation that runs from *1* to *100* frames but doesn't actually begin animating until frame *50*, whereupon the animation changes smoothly until the final frame. If you check ***Unique Frames***, then just frames *1* and *50* to *100* will be rendered.

Transient Tweaking

The Tweaks Button

The Renderer's *Tweaks* area gives you the power to modify any tweaked parameter that a selected DarkTree has. However even though you can, for example, change a DarkTree color parameter (provided it's tweaked) and render a texture in the new shade, the change is transient. Any modifications you make to a tweaked parameter are only "active" for the render. The DarkTree itself remains unchanged.

If the selected or current DarkTree doesn't have any tweaked parameters, the *tweaks button* will simply be labeled ***No Active Tweaks***, and that will be that! If, on the other hand, the currently selected DarkTree has one or more tweaked parameters, the *tweaks button* will be labeled ***Select Tweak***.

To see the list of tweaks for a DarkTree, just click on the ***Select Tweak*** button.*Figure 2.89* shows you the two tweaks accompanying the SnowFlake texture. If you choose one of these tweaks, you'll be able to change it. In the figure, note that the integer-type tweak, ***Snowflake Sections***, has been selected already. As you can see, its name and current value are visible on the left, near the bottom of this segment of the Renderer.

2.4.4 Rendering Your Texture Maps

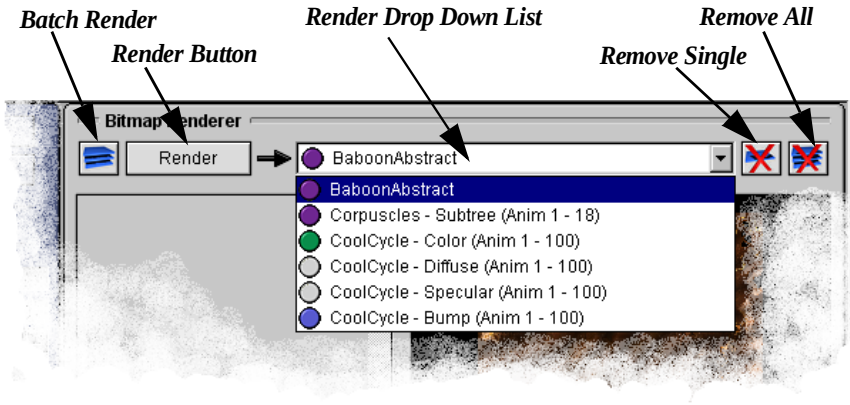


Figure 2.89

A screenshot of the rendering controls. Notice that the render drop down list is open, showing all currently loaded textures.

This final piece of the Renderer controls the point of the exercise - rendering out the texture maps you have set up. First we'll cover the rendering control options. This is the place to initiate the render itself. The second part explains the render log and progress meters. Their function is to give progress reports during and at the completion of the renders. *Figure 2.91* has three examples of *render log* output.

Render Control

As you can see from *Figure 2.89*, the render controls are arranged across the top edge of the Bitmap Renderer. The figure points out and labels these controls for you as well. So, beginning with the left most button and traveling to the right, we have:

The Batch Render Button

The *batch render button*, located farthest to the left, toggles the *Render button* between batch and single renderers *Render button* is just to the right of the *batch render button*. You can easily see which type of render is currently in affect, because clicking on the *batch render button* will retitle the ***Render*** button with the word ***Batch***.

For a batch render, the Bitmap Renderer will process each DarkTree that you currently have loaded. Rendering begins with the first texture on the *render drop down list* and continues until the last one has been processed. You can go away and do something else, then return to a completed job.

The Render Button

Selecting this button will start a render for either the highlighted (selected) texture on the *render drop down list* (this is a single render) or a batch render. The type of render currently active is determined by the name on the button. If it's ***Render***, then one texture will be rendered; if it's ***Batch***, then all textures currently loaded on the Bitmap Renderer will be processed. The *Render button* is located second from the left in the render control segment.

The Render Drop Down List

When you open this list, you'll see all of the textures you have loaded on the Bitmap Renderer. (*Figure 2.89* shows this list in the open position.) The information listed for each DarkTree is as follows:

- *a type icon that identifies the texture type.*
- *a DarkTree name.*
- *an identifier after the name (usually). If you're rendering split out channels from a shaded DarkTree, you'll see "-type" where type refers to the channel type (color, bump, etc.). If you've loaded a DarkTree that has not been split into channels, you'll see "-Subtree" after the name (unless it was dragged from the Texture Library). If the texture is an animation, you'll see the frame range. The render list in Figure 2.89 shows examples of each of these identifiers.*

The Remove Single Button

The *remove single button* removes the currently selected texture and its output specifications from the Bitmap Renderer. If a shaded DarkTree has been split into channels, the button removes a single channel texture. You'll find this button second from the right in the render control segment.

The Remove All Button

The *remove all button* erases all textures and their information from the Bitmap Renderer. This button is the last control on the right.

Rendering & Progress Information

Output information is divided among three controls: the *render log*, the *Bitmap render bar*, and the *Sequence render bar*. Coverage for each of these controls follows.

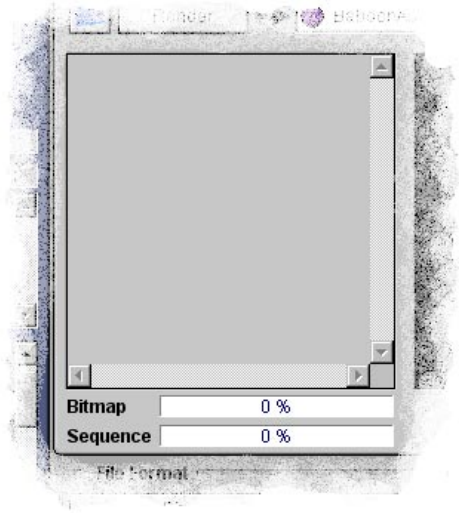


Figure 2.90

The Rendering & Progress Information segment of the Renderer, which consists of the render log, Bitmap progress bar and the Sequence progress bar.

The Render Log

Once the rendering process begins, the *render log* (the large area in *Figure 2.90*) will continually display updated information on the current render's progress. When the render is completed, the *render log* will list the total rendering time as well. From left to right, the log displays information in the following format.

(1) [frame number / bitmap number] - the render log displays this information as "F0001/B1"

(2) [rendering time] - displayed in the form "(hours:minutes:seconds)"

(3) [texture name] - the DarkTree name just as you see it on the *render drop down list*

(4) [optional information] - "-Subtree" if the DarkTree was loaded from the Editor, "-channel type" if the DarkTree is a split-out shader

(5) [total rendering time] - on a separate line and only when the render is completed.

Here are a few examples of render log output, just to give you an idea of how this information looks for different types of texture.

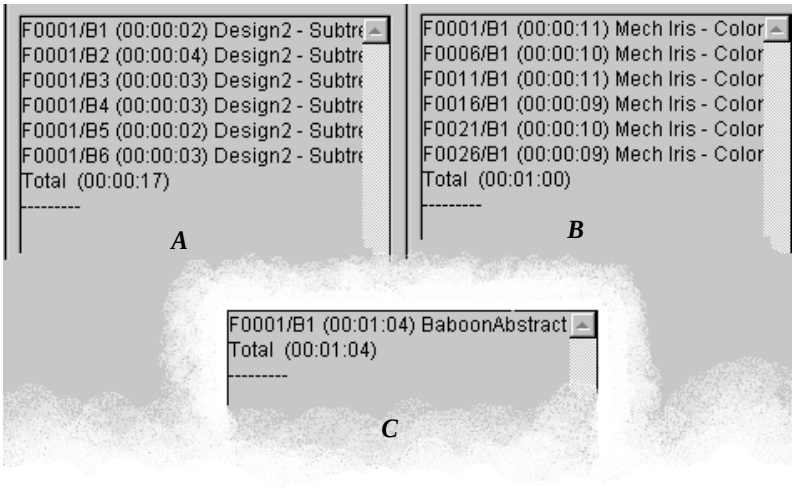


Figure 2.91

Here are three sets of output from the render log. Output A is one frame of cubic output; output B is an animation of the color channel from a shaded texture. The animation is set to render every sixth frame. An finally, output C is a shaded DarkTree that was loaded from the Texture Library. You can tell that this is so because the render log lists just the name of the shader with no additional comment

The Bitmap Progress Bar

This bar, labeled ***Bitmap***, tracks the progress of the currently rendering texture or bitmap. The information is given as percent complete. The ***Bitmap*** progress bar is located below the render log.

The Sequence Progress Bar

This bar, labeled ***Sequence***, tracks the progress of an animation sequence. This information, like that from the ***Bitmap*** *progress bar*, is given as a percentage of the total animation. For example, if you're half-way through the second frame of a ten frame animation, you should see ***15%*** written in the bar. You'll find the ***Sequence*** *progress bar* below the *render log.*

NOTE: Both progress bars will have the same percentage, if the rendering texture is a single frame.

Part 3
The Conceptual Discussions

Basic Concepts of DarkTree Textures

The Component Classifications

Transformations in Texture Space

Working With Bumps

Animating Your DarkTrees

Working With Generators

Notes on the Simbionts

Section 3.1
Basic Concepts of DarkTree Textures

3.1.1 Section Overview

This first section of *The Conceptual Discussions* is an in-depth discussion of the terminology and concepts behind the DarkTrees, and the components they are made of. All of Part 3, and especially this section, are definitely worth taking a little time to read and digest.

The following subsection begins by defining exactly what a procedural texture is. Subsequent sections go on to explain how procedural textures are encapsulated into texture components, which are ultimately combined together to build the texture-generating structures called DarkTrees.

3.1.2 Algorithmic Textures

DarkTree Textures uses procedural textures to generate its color, percent, and bump texture maps. You can think of a procedural, also called an algorithm, as a mathematical recipe. Thus, procedural textures are simply mathematical recipes that tell the computer how to create the colors and elevations (bump) of a textured surface.

Shaders are more complex textures, since they include full lighting and surface attribute algorithms that fully "shade" a surface. Procedural textures that include shading attributes are generally referred to as procedural shaders. DarkTree Textures has a very comprehensive shader component for those users who wish to render fully-shaded images. The version 2.0 DarkTree Shader component allows users to link up to eighteen separate attributes for a single shaded texture. The DarkTree Shader component is designed exclusively for the **Root** socket position.

Procedural textures have a number of advantages over standard texture mapping. They can generate high- quality results at almost any resolution. They avoid the false lighting cues that occur when using photo textures. And they can animate in ways that cannot be accomplished by other means. They can simulate animated natural phenomenon such as smoke, fire and water, or unnatural plasma and ethereal effects that simply cannot be drawn or filmed. DarkTree Textures' procedural textures have all of these advantages without being directly tied to any one rendering package.

3.1.3 The Texture Components

Every procedural texture in DarkTree Textures is encapsulated in a flexible, object-oriented structure. These structures are called components, and they are the building blocks of DarkTree Textures. Components allow the user to combine a group of procedural textures in a number of ways. The result is the capability to produce a virtually unlimited number of unique textures.

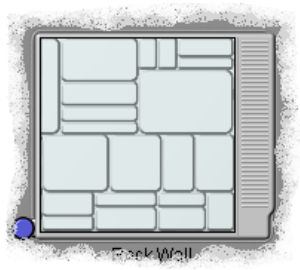


Figure 3.1
Here is a single component. This one is called Rock Wall.

Visually each procedural texture is represented on the DarkTree Editor *workspace* as a square gadget called a component. Each component has a render preview image at its center that shows what the internal texture looks like with the current settings. Figure 3.1 is a screen capture of a single component. See *"The Basics of Tree Building" on page 62* to learn how to place and edit individual texture components.

Actually, many of the DarkTree components are not really designated as procedural textures. Rather, they are components that modify and enhance the pattern-generating components, making the combined whole considerably more powerful. See the *"The Classifications" on page 151* for an in-depth discussion of the different classes of texture components you'll find in DarkTree Textures.

3.1.4 Component Types

The most impressive looking textures combine matched color and bump channels. More specialized textures contain percent or grayscale channels to control surface attributes such as luminosity, reflectivity, transparency, specular level, and so on. High-quality 3D rendering demands that these surface attributes complement one another. To support this, DarkTree Textures has three basic or key versions of each texture component: color, percent, and bump. Each component shares most of its user-settable parameters or controls between the three types, and each component type is easily converted to either of the other two types. This means that DarkTree Textures can generate perfectly matching color, percent and bump textures that work together to make very realistic-looking surfaces. The user then has the option of linking all three key DarkTree types to the DarkTree Shader mentioned earlier.

Some specialized texture components are restricted to a single key type. For example, the components of the Generator class are 2D functions. And they only make sense as percent components. This discrepancy generally makes no difference in the ability to match color, percent and bump textures because of the way these specialized textures work.

You can easily determine the type of a component in the DarkTree Editor by looking at the small colored disk, called a *type icon*, attached to the component's lower left-hand corner. A green icon tells you that the component's type is color, a gray icon means the component's type is percent, and a blue icon indicates a bump type component. The *type icon* for the DarkTree Shader component is a special type, colored purple.

NOTE: A percent-type component is the same as a grayscale texture. We call them percent textures because they usually control the degree of something. For example, in most renderers, grayscale or percent components control things like the degree of reflectivity, transparency, luminosity, shininess, etc. Percent components are also used extensively to control other components.

3.1.5 Linking Components Together

We mentioned in the previous subsection that the DarkTree components can be combined so that they work together. This is accomplished with a process called "linking" (*seeFigure 3.2*). Each component has a set of parameters for controlling attributes such as density, color, roughness and so on. These parameters, which are usually set by the user, control the "look" of the texture.

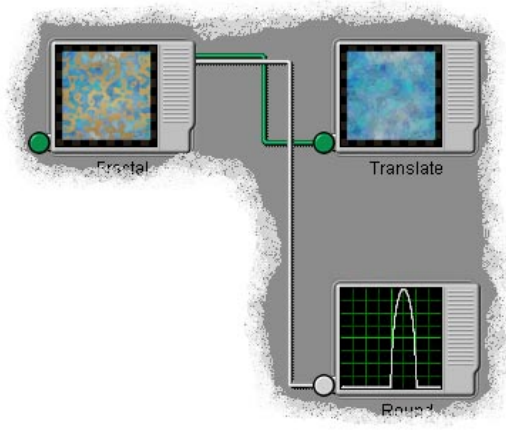


Figure 3.2
An example of three components linked together, each through the parent component's parameters.

The majority of the parameters are linkable. This means that you can do more than simply set a static value for the parameter. You can link it to or replace it with another component as well. For example, you can link the brick color of the Bricks component to a color Randomizer component. Now each brick has a slightly different shade. And/or, you can link the ***Mortar Width*** parameter to a percent Noise component to add an uneven, rough-cut look to your bricks.

On the DarkTree Editor, links are represented by colored wires that connect components. These are referred to as wire links. The wires themselves are color-coded to indicate the type of the link (again, color, percent or bump). The color-coding for wires is the same as the color-coding for the *type icons*. That is, green wires represent color links; gray wires represent percent links, and blue wires represent bump links. The type of the parameter that originates the link must be the same as the type of the component at the link's destination. See *"The Basics of Tree Building" on page 62,* for instructions on how to link parameters and components together.

3.1.6 Texture Trees

A collection of texture components, combined together with links, is called a DarkTree. The name "tree" really comes from computer science and mathematics, where similar conceptual structures are studied. A tree structure looks like a real tree in that it begins with a root and branches out at each level, similar to a real tree with its branches. A DarkTree is our own brand of tree structure, designed specifically for texture generation.

The three basic types of DarkTrees match our three different texture component types. A DarkTree generates textures of the same type as itself. In other words, a color DarkTree generates color texture maps, a percent DarkTree generates grayscale texture maps, and a bump DarkTree generates grayscale elevation or bump maps. Note that it is important that these channels be initially separated into different subtrees if you will be linking them to a DarkTree Shader component or separate DarkTrees if you won't be linking to a Shader. This allows you to purposefully mismatch the color, percent, and bump channels, which is often necessary for creating realistic textured surfaces.

The type of a DarkTree is determined by the type of the component in the root position of the DarkTree. The word "root" is another computer science and mathematics term that means the origin of the tree. In DarkTree Textures, this means the left most component of a tree structure. The structure of a DarkTree is built and displayed on the DarkTree Editor. Each tree structure is displayed lying on its side, which provides the maximum *workspace* area. The tree origin or top component is labeled **Root**. The resulting DarkTree always looks like the root component but modified by the other components that are linked to it.

Other important tree terminology to know is the parent and child relationship between linked components. If component A has one of its parameters linked to component B, then A is called the parent of B, and B is called the child of A. This terminology is

borrowed from family tree charts where the child is linked to the parent by bloodline, but may also have a child of its own. These terms are an easy way to keep track of which component in a tree you are referring to.

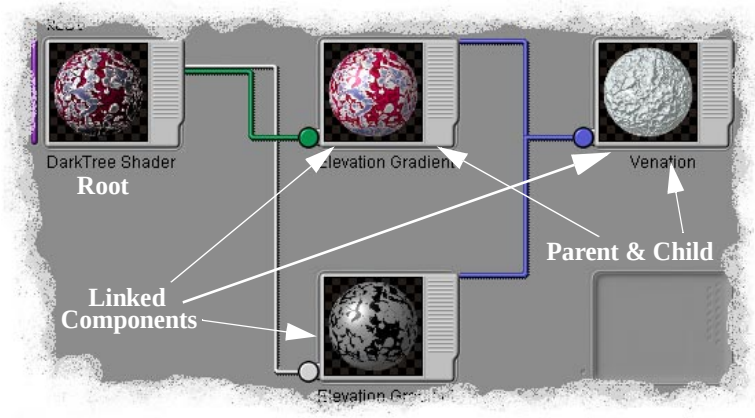


Figure 3.3

An screen shot of a DarkTree open on the Editor workspace showing the names and positions of the components.

Figure 3.3 graphically shows a DarkTree, with pointers to its various parts. You can see what a DarkTree looks like by editing any of the textures in the Texture Library. When the DarkTree is opened in a new DarkTree Editor, the root component is in the left most socket and is labeled, as you would expect, **Root**. Parent components are always to the left of the child components they link to.

NOTE: The root component has no parent, only children.

NOTE2: A single component in the root socket is technically still considered a DarkTree, even if it has no children.

3.1.7 Texture Types Working Together

The real power of DarkTree Textures is in how the three key component types (color, percent and bump) work together. As you may have noticed, color DarkTrees often don't just consist of color components. A DarkTree can contain any number of color, percent and bump components, working together in concert to produce the texture you want.

Percent-type components are very important in the construction of most DarkTrees. The component parameters have been adjusted to fit into the *0%* to *100%* range. This means that parameters such as the width of a stripe, the sharpness of a scale, or the

density of lumps can be controlled by percent child textures that are linked to those parameters. This naturally gives you a great deal of control over how your final texture will look. It's a good thing.

3.1.8 Converting DarkTrees to Texture Maps

The design and creation of textures in DarkTree Textures is done using procedural texture trees. But to make these textures useful, they must be saved as procedurally-generated bitmaps using the Bitmap Renderer. DarkTree provides four mapping modes for saving textures: planar, cylindrical, cubic and spherical. These four modes correspond to the four most common methods of projecting textures onto 3D surfaces. For non-3D applications, planar-mapped textures are generally what you'll want.

Cylindrical and spherical texture maps are specially designed to be projected, or wrapped, onto 3D surfaces. Cubic mapping is typically the mode of choice for environments. You can use planar mapping for the truly flat surfaces (planes) in your 3D scenes. And cylindrical projections are useful for mapping textures onto tube-shaped objects. Spherical projections are a good method for mapping textures onto spheres, as well as onto complex shapes.

Usually, when a texture map is projected onto an object with a cylindrical or spherical projection, there is a seam in the back where the edges of the map meet. In addition, spherical projections squeeze the top and bottom of the texture map in order to more closely fit a sphere. This results in polar distortion. DarkTree Textures' cylindrical and spherical mapping modes avoid these problems by creating seamless and pre-distorted texture maps.

3.1.9 Texturing Solutions - Procedural Shaders

If you typically use one of the more popular modeling/rendering packages, chances are good that a DarkTree plug-in can meet your needs. The DarkTree plug-ins, called Simbionts, allow you to import your DarkTrees directly to a 3D application as procedural shaders. Having the ability to import DarkTrees without having to render them as texture maps first is the best of all possible texturing solutions. Check our website to determine whether or not one of the Simbionts matches your favorite 3D application.

Since procedural shaders are true 3D volumetric textures, you won't have to think in terms of wrapping them onto 3D objects. Rather, you can think in terms of actually carving each object out of a solid block of texture (no tiling, stretching or seams). And, using *Tweaks*, you'll be able to adjust the textures in a variety of ways, even though they've already been imported to other software.

Procedural shaders do render more slowly, since the information is being calculated on the fly while your scene renders. But the texture quality is excellent. Read *"Procedurally-Generated Texture Maps VS True Procedural Surfaces" on page 127* for more in-depth coverage of the pros and cons of using a Simbiont plug-in.

3.1.10 What You Get

What all this gets you is a very flexible and powerful texture-generation environment for creating surfaces, processing images, and concocting special effects. We have worked very hard to make the interface easy to use and fast to learn. Understanding how to combine texture components to get what you want will take a little longer, but like most good artists' tools, DarkTree Textures gives you the power and leaves you to create the art. And like most good tools, the best way to learn is to simply play around - a lot. The rest of the *Conceptual Discussions* will give you a better idea of how to control the components and build the DarkTrees you're looking for.

Section 3.2
The Component Classifications

3.2.1 Section Overview

The texture components available in DarkTree Textures are grouped into classifications, based on how they function and the kinds of results they produce. Understanding the purpose for the different classifications and how they work is the key to knowing how to design your own DarkTrees. This section examines each component class separately, explaining the purpose and reasoning behind each classification and how it fits into the overall design.

At the end of this section, you'll find a number of short tutorials that demonstrate some of the more interesting component classifications in action. Each tutorial teaches you how linking components in a specific way can help you to achieve the most unusual and/or realistic effects. None of these tutorials go into great detail about how to perform some of the DarkTree Editor tasks. If you find that to be a problem, please run through the *"The DarkTree Editor Tutorials"* beginning on *page 108* first.

NOTE: In the DarkTree Editor, texture components are stowed in folders in the Component Bin. The classifications do double-duty as the folder names.

The Role of the Modifier Components

Some of the components that DarkTree Textures provides are strictly for modifying other components and, therefore, do not generate useful textures of their own. These modifier components are designed to work in close combination with regular stand-alone textures to multiply their power and flexibility. The Control, Deform, Generator, Process, and Transform classes consist of modifier textures only. The Time component from the External class is also a modifier. All other textures generate their own patterns.

NOTE: Most modifier components do generate representative images on their own, simply to make their purposes more easily identifiable. For example, the Randomizer component from the Control class displays a grid of randomized colors to represent its job, which is to generate random colors for regular patterns.

3.2.2 The Classifications

The Controls

Components of the Control class bring the quality of randomness to otherwise regular texture parameters. Most of the texture components in the Gradient and Pattern classes, as well as a few of the Natural textures, generate regular patterns with distinct regions. Control textures have the ability to randomize any color, value or transform for textures that have these regions. For example, you can make every stone in the Flagstones texture a different color or shade.

Control components are modifiers, and as such must be linked to other components to be useful. In order for a Control component to work properly, it must be linked to the regional parameter of a parent component. That is, a component that is positioned to its left on the Editor *workspace*. In other words, the Control component must be a child or descendent of the component whose regions are to be randomized. In general, any parameter that defines a value for a region, such as a brick, a ray or a gradient shell, can take advantage of a link to a Control component. When you open the *links list* of any component, you can easily recognize its regional parameters (if any) because each has a *regional icon* associated with it. *Control Tutorial 1* on *page 156* and *Control Tutorial 2* on *page 157* are basic examples of Control component usage.

A Control component and the parameter it's randomizing need not be next to each other. In other words, any number of other linked components may separate the two. When a Control component receives a request for a random color, percent, bump or transform, it begins searching for any regional parameter that it might be linked to. First it checks to see if its link to the parent is through a regional parameter. If it isn't, the Control component checks its parent's link to its parent and so on, all the way to the **Root** level. A Control component uses the regional information from the first regional parameter it finds, to generate its random value. *"Control Tutorial 3" on page 159* demonstrates the situation where a Control component is indirectly linked to a regional parameter, and *Figure 3.6* shows what the completed component layout looks like on the Editor *workspace*.

NOTE: If a Control component is not linked directly or indirectly to a regional parameter, it will simply generate a single value.

The Deforms

The Deform components' task is to distort other linked components or linked component subtrees. This important component class allows you to take any regular pattern in DarkTree Textures and subtly warp it until it begins to look organic. If

required, it can distort a pattern beyond recognition. Deform components work by distorting your view of the texture they are linked to, in the same way that a warped lens would deform your view of the world.

Deform components are modifiers and so cannot make useful patterns on their own, even though they've each been assigned a simple image for display on the component faces. For the deformation to work properly, the component or component subtree you wish to distort must be linked to the ***Background*** parameter of the Deform component. *"Deform Tutorial 1" on page 161* and *"Deform Tutorial 2" on page 162* show two basic examples of how to use the Deforms to best advantage.

The Externals

Components of the External class allow the user to bring in information from outside DarkTree Textures. This class includes components that allow for one or more external image files to be read-in, or for the dimension of time to be introduced. The new addition then becomes part of your DarkTree. (You'll find a detailed description of the Time component in *"Animation Basics" on page 189*.) With DarkTree Textures version 2.0, the External class includes several new components, for instance: Audio Wave, Cache and Poster Board. The Time component replaces the version 1.0/1.1 Trigger component.

The external image files can be anything from another rendered DarkTree to a famous painting. The Image component reads-in one external image. Image Sequence is basically the same component, except that it reads in a series of images. With these two components, you can either use the DarkTree Editor to apply effects to the images, or you can use the images to control linked textures. *"External Tutorial 1" on page 163* teaches you how to use an image to control the distribution of a cloud texture.

The Generators

The Generator components are basic 2D functions. They're used to control many aspects of many textures, such as bevel profiles, time rates, blend functions, and geometric shapes. *Generator Tutorials 1, 2* and *3*, beginning on *page 165* show generators controlling geometry, blend and bevel profiles respectively.

Generators are modifiers as well, and so must be linked to other textures to be useful. Just as Control components must be linked to regional parameters, generators must be linked to function parameters, such as ***Center*** from the Stripes component. You can have any number of texture components between a Generator component and the function parameter controlling it. A generator looks for its closest linked function parameter to determine what values to generate for any given point. First it checks to see if the link to its parent is through a function parameter. If it's not, the Control component checks its parent's link to its parent and so on, all the way to the root of the

DarkTree. If it can't find a function parameter, the generator will produce a constant value. Function parameters are clearly associated with a *function icon* on many of the component *links lists* where they occur. Actually, a function parameter may have one of two types of *function icon*: one signaling that it can be linked to a function, the other signalling that it can only be linked to a function.

If a generator's ***Input*** parameter is linked to a Time component, the generator function becomes time-based. And in this case, the generator no longer needs to be linked to a function parameter. Components of the Generator class have their own special *Generator Editors*. Each of the *Generator Editors* includes a large graph. You can tell a generator has become a time-based function when its plot turns blue. White plots are regular generators; blue plots are time-based generators.

NOTE: Generators are percent-types only, since they don't make sense as color- or bump- types. Generator components cannot be DarkTree roots, nor can you drag one to the Renderer or the Examine Window.

The Gradients

Components of the Gradient class provide smooth transitions (or gradients) between color, percent or bump values. In fact percent-type gradients, when used to control other components, are an extremely useful way of providing a smooth transition of values. For example, you could use a Linear Gradient component to create the colors in a sunset.

The Naturals

The Natural components produce only natural-looking textures, such as Clouds. Natural components have some of the most complicated algorithms in DarkTree Textures. They are especially important for creating realistic textures. It's worth your time to explore the many different "looks" you can get out of these texture components. One example is the first DarkTree Editor tutorial on *page 108* which uses an Agate component to create a basic wood texture.

The Noises

Components of this class are the *realism worker bees* of DarkTree Textures. Like components of the Control class, Noise components create randomness in otherwise perfect textures. And, as you know, the real world is seldom perfect. Mortar lines are never completely straight and paint is never totally uniform. The longer a real-world surface is out in the weather, the more seemingly random changes occur. The Noise components are great for helping you create that realistic look for your textures.

The Patterns

The Pattern class consists of regular shapes. These textures are meant to be fundamental and so were designed from basic shapes, such as stripes and checks. Patterns are actually very flexible and can add a great deal of control to your DarkTrees. They commonly contain regional parameters for linking to the Control components, and, subsequently, randomizing the blocks of the pattern. They also have function parameters that can be linked to generators to control attributes like geometry and bevel profiles. And, they have geometry parameters that can be linked to Noise components, which help to create more realistic results. Use Patterns for giving regular structure to textures that are meant to look man-made.

The Processes

Components of the Process class perform image processing operations. Process components are modifiers, and as such, can modify the "look" of child textures. They do not, however, create any patterns of their own. Most Process components, such as Add, Composite and Darken, should be familiar to anyone who has worked with an image processor or compositing application. *"Process Tutorial 1" on page 168* uses the example of weathering to teach you how to use these components.

A number of the Process components are uniquely powerful when they're used to combine or alter bump textures, which they treat as elevations. For example, the Add component takes the elevations of two child bump components, then adds them together. This allows you to layer one bump on top of another. For instance, the Maximum component compares the elevations of two child components and chooses the larger of the two, allowing you to embed one bump into another. *"Process Tutorial 2" on page 169* is an example of a bump Process component in action.

The Shaders

Use the Shader components for creating fully-rendered or shaded texture output. Usually, textures are generated as raw color, percent and bump texture maps, which can then be read into another 3D application and used to define surface attributes. Shader components combine color, percent and bump textures to create fully-rendered surfaces. This can be especially useful for web pages, game textures, and other 2D applications.

In the current version of DarkTree Textures, there's just one Shader component, the DarkTree Shader. It uses the Phong illumination model with two light sources. DarkTree Shader allows you to link or statically set a value for ***Surface Color***, ***Surface Bump***, ***Reflectivity***, ***Luminosity*** and a wide variety of other surface attributes, eighteen in all.

Most of the finished DarkTrees included with DarkTree Textures make use of the DarkTree Shader. This provides you with a fully-shaded view of each texture, without the need to drag multiple channels to an Examine Window. If you wish to render the channels of a shaded texture to independent color, percent, and bump bitmaps for use in a 3D application, the links coming out of the DarkTree Shader will indicate which texture subtree belongs with which attribute.

*NOTE: The DarkTree Shader should only be plugged into the **Root** socket and no other.*

The Tiles

The Tile class of components is basically a set of tools for creating tiled bitmaps. Tiling includes both spatial tiling, what you would normally think of as tiling, and looping animations, which tile over time. You can even combine the two to get animated looping tiles. Tiling methods in this class focus either on blending edges together or warping texture space to get a truly seamless tile. One of the methods will be the most suitable choice, depending on the situation.

Tile components should generally be linked to the root of a subtree (subroot). If you're making a fully-shaded tile, you'll want to tile each shader channel (color, bump, specular level, and so forth) immediately before linking it to the DarkTree Shader component. For example, you could link the DarkTree Shader's ***Surface Color*** parameter to a tile component, which would in turn have it's ***Background*** parameter linked to the color subtree you want tiled. Aspect controls are available in case you would like to make non-square tiles. While designing a tiled texture, be sure to set the mapping mode to ***Frame***, which enables the tiled preview mode. This lets you see how the tiled result will look at each step in the creation process.

The Transforms

Use the Transform class of components to apply transformations to single components or component subtrees. The Transforms are just like those you might apply to any individual texture from within a *Component Editor*, except that these Transform components can be uneven or change over time. To set up a Transform component correctly, make sure it's in the parent position and linked to a child texture or texture subtree through its ***Background*** parameter. Each Transform component has this parameter. *"Transform Tutorial 1" on page 172* goes through the example of using the Translate component in an animation.

NOTE: Animated translation through components from the Noise or Natural classes is the basis for many of the more interesting special effects that you can achieve with DarkTree Textures.

3.2.3 The Control Tutorials

Control Tutorial 1

This tutorial demonstrates how you can use the Randomizer component to control the color of a brick pattern.

Randomizing the Bricks

1. For a start, let's drag out the components we're going to use for this tutorial.

First, go to the *Component Bin* and open the Pattern folder. Drag out and plug the Bricks component into the **Root** socket; paste it as a color-type. Second, open the Control folder. Locate and drag the Randomizer component onto the *workspace*, then plug it into the socket directly to the right of **Root**. Paste it as a color- type also.

2. In this step, you'll take care of a couple of preliminary chores: scaling Bricks by *50%* and linking components.

Open the Bricks *Component Editor* and click on the ***Scale*** button. Look for the scale button in the ***Transforms*** area. It's the lowest of the three transform buttons. Notice the check box that now appears. When this box is enabled, as it should be when it's first opened, it ensures that your scaling will be even. You'll only be able to enter a value in the X-input box, but the scaling will be for all three directions. Enter *50.0*, then press the ***Return*** key to update. This should zoom away from the bricks by the value given, and has the effect of showing more but smaller bricks in the *update window.* Close this *Component Editor.*

Next, link the ***Brick*** parameter to the Randomizer component. Note that Randomizer is now controlling the brick colors.

3. In this step, you'll play around with Randomizer's color range to see what color effects you can get for the bricks.

First though, let's get a more relevant view for the upcoming changes. Go to the ***Views*** area in the Randomizer *Component Editor* and click on the button labeled ***Root.*** Now, you'll be able to see how your color changes affect the brick pattern as a whole.

Next, look at the Randomizer component parameters. As you can see, the colors are controlled with the HSV color model. **Hue**, **Saturation**, and **Value** input runs from *0.0* to *100.0* for each. Randomizer will randomly assign a color in the hue, saturation, and value ranges to each brick. Try several values until you find a range that you like.

4. The last task is to simply play around with the seed value, noting how it affects the final image.

In Randomizer's *Component Editor*, you'll see a parameter named ***Input Seed***. Use the zip-slider (the small double-pointed control) to change the seed to some other positive integers. Try lucky thirteen. Notice how changing the seed value changes the color distribution for the bricks.

This completes the first tutorial for components of the Control classification. *Figure 3.4* shows the component arrangement for this tutorial. Two more Control tutorials follow, each with additional valuable lessons. By the way, the finished DarkTree from this tutorial is located in the **Textures/Tutorials** folder. The DarkTree name is **Random Bricks**.

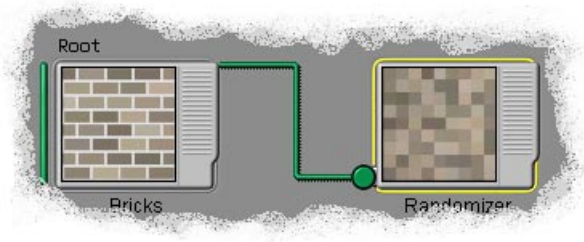


Figure 3.4
The components from the Randomizing the Bricks tutorial. This is a small but complete color-type DarkTree.

Control Tutorial 2

This tutorial uses a control component to break up a pattern, which in turn greatly improves the DarkTree. As the tutorial progresses, you'll be asked to put up two Examine Windows for comparison purposes.

Shuffling the Scales

1. In this step you'll plug in the components for the first comparison.

Go to the *Component Bin*, and open the Pattern folder. Drag out the Scales component, and drop it on the **Root** socket. Paste it as a color-type. Next open the Natural folder. Drag the Blobs component to a level two socket, directly to the right of the root, and drop it. Paste it as a color-type, as well.

Finally, drag out or copy another Scales component. Plug it into the socket directly below the Blobs component. Paste it as a bump-type.

2. In this step, you'll just reset some values for the Scales components.

On the color Scales *Component Editor*, click on the ***Edit*** button for the ***Edge*** parameter to open the *Color Browser.* Darken the edge color to *110, 110, 110*, or something similar, by entering the values in the **RG** and **B** boxes. Then change the ***Scale*** parameter to some light color. This will make viewing a little nicer when we have the two comparisons set up.

3. In this third step, we'll finish the first part of the comparison we're developing.

First, make a wire link between the color ***Scale*** parameter and the color Blobs component. Next, open an Examine Window, using the color Scales *component menu*. Select ***Surface Color*** from the popup dialog. And for depth, drag the bump Scales component to the Window as well. Select ***Surface Bump*** from the popup dialog this time.

Now observe the Examine Window image. Do you notice how the Blobs cross the individual scales? Because of the crossing problem, the scales don't look separate or real. Let's try to improve them.

4. In this step, you'll rearrange the components, and add a control component to the DarkTree.

Using the Blobs' component frame, drag it one socket to the right (level three). Open the Control folder on the *Component Bin* and drag out Shuffle Translate. Plug it into the socket just vacated by Blobs. Paste Shuffle Translate as a color-type.

Next, relink the color ***Scale*** parameter to the Shuffle Translate component. And then link the ***Background*** parameter from Shuffle Translate to the Blobs component.

5. This short step takes care of the final editing chores for the tutorial.

Open the *Component Editor* for Shuffle Translate. Change the values for the parameters ***X Amount*** and ***Y Amount*** to *2.0*. This tells Shuffle Translate to randomly translate each scale region **+** or **–** *2.0* units in both the X and Y directions. **6.** Step 6 completes this tutorial.

Open another Examine Window using the same method as in Step 3, and place it beside the first one. Drag bump Scales onto the window, again selecting ***Surface Bump*** from the Examine popup dialog. Note the improvement in the image. Shuffle Translate translates the texture (in this case, Blobs) within each region of a regional component (in this case, Scales). The effect is that the regions or scales appear to be

independent of one another.*Figure 3.5* shows the final component arrangement on the Editor *workspace*. And a copy of the finished DarkTree for this tutorial is in the **Textures/Tutorials** folder. It's named **Shuffle Scales**.

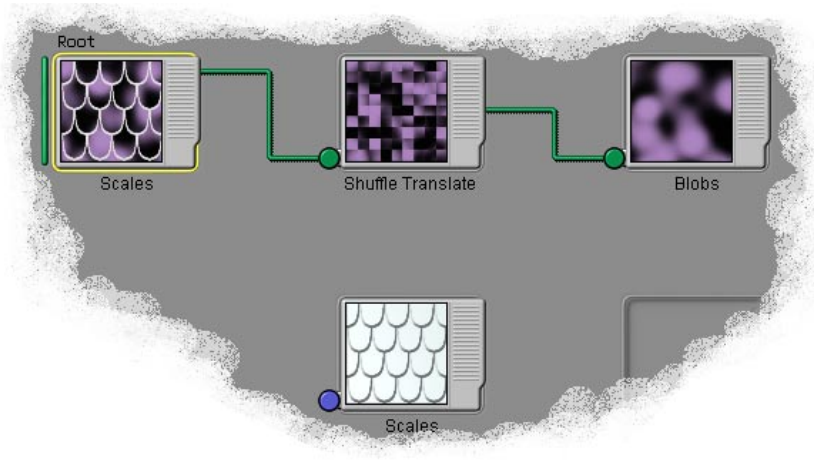


Figure 3.5
The component positioning for the Shuffling the Scales tutorial.

Control Tutorial 3

This tutorial demonstrates how a Control component, in this case the color Randomizer, can "do its job" just as well when one or more other components are separating the control from its target component.

Randomizing By Indirect Link

1. In Step 1, you'll drag out and plug in the components needed for this tutorial.

From the *Component Bin*, open the Pattern folder. Locate good old Bricks and drag it onto the Editor *workspace*. Drop on the **Root** socket, and paste it as a color-type. Next, open the Natural folder and drag Venation to the socket directly to the right of **Root**. Paste it as a color-type, as well.

And last, open the Control folder and drag our old friend Randomizer to a level-three socket, directly to the right of Venation. Paste it as a percent-type.

2. For this step, you'll be asked to make a couple of changes to the Bricks component.

Open the *Component Editor* for color Bricks, and use the ***Scale*** button to make more bricks visible.That is, enter *50*, which should give you twice as many bricks. Be sure the scaling lock is enabled, so that the scaling will be even.

While the Bricks *Component Editor* is still open, enter *5* in the format input-box for the ***Mortar Width*** parameter. This will make the mortar lines very narrow. Go ahead and close the *Component Editor* now.

3. You'll make the wire links in this step.

First open the Bricks *links list*, select the ***Brick*** color parameter, and link it to the Venation component. Next, link Venation's ***Roughness*** parameter to the percent Randomizer component.

4. For this final step, you'll examine the DarkTree rendered image. And we'll explain some points about this type of tree configuration that should help you to get the most out of components of the Control class. For the final tree, check out *Figure 3.6*

First, open an Examine Window from the Bricks *component menu* and, when it comes up, select ***Surface Color*** from the popup dialog. Now resize the window so that you can get a more detailed view of the image.

In *Control Tutorial 1*, we used a color Randomizer to vary the HSV color values for each brick. Here, we're using a percent Randomizer to vary the degree of roughness in each brick. ***Brick*** is a regional color parameter for the Randomizer component. And as you can see, the Randomizer is not linked to it directly. Rather the link is indirect, through Venation. Such a linking setup allows Randomizer to randomly vary the ***Roughness*** parameter of Venation, on a per brick basis.

A Control component will seek through as many levels of parents as is necessary, until it finds a regional parameter. If there is more than one such parameter, a Control component will use the first one it encounters. This feature can be very powerful. And it's definitely worth the time it takes to understand it.

A DarkTree of this tutorial is in the **Textures/Tutorials** folder under the name **Indirect Random**.

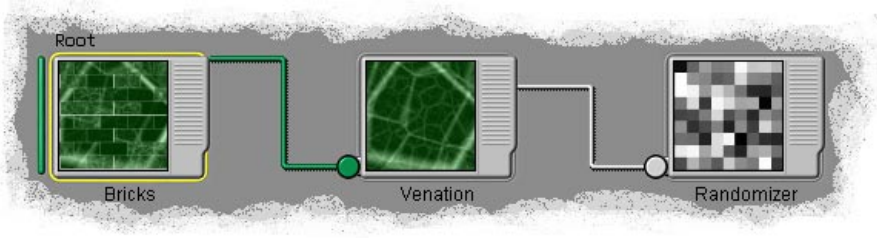


Figure 3.6
DarkTree workspace layout for the Randomizing by Indirect Link tutorial.

3.2.4 The Deform Tutorials

The Deform class of components allows you to take any evenly-shaped or linear patterns and subtly warp them until they begin to take on an organic look or, if exaggerated, a wildly distorted one. This first tutorial teaches you how to create a look similar to the refractive caustic light effects on the bottom of a swimming pool.

Deform Tutorial 1

Creating Organic Pool Lights

1. In this step, you'll just choose the components for the DarkTree, and get them plugged into their respective sockets.

From the *Component Bin*, open the Deform folder. Drag the Warp component to the Editor *workspace*, and plug it into the **Root** socket. Paste it as a color-type. Next, drag out Venation from the Natural folder, and plug it into a level two socket, beside Warp. Paste this component as a color type as well.

2. You'll simply be directed to edit a component for this short step.

Open the *Component Editor* for Venation. Set the ***Background*** parameter to a strong dark blue shade, such as RGB 0, 0, 128. And set the ***Vein*** parameter color to a cyan shade, for instance RGB 0, 255, 255., then close the *Component Editor.* The strong colors you've added here make it easier to see the subtle variations in the finished DarkTree. This time, play around a bit with the ***Input Seed*** to get a different arrangement of veins (*5* looks pretty good)

3. For this step, you'll finish the building chores.

Before adding the link, open an Examine Window from the Venation *component menu* and select ***Surface Color***. Notice how linear the Venation pattern looks. We'll open another Examine Window shortly, so that you can compare/contrast this Venation with one that's been warped.

Now, link the ***Background*** parameter of Warp to Venation. Open another Examine Window from Warp (making the same selection as before from the popup dialog) and set the two windows side by side. Notice how the linked Venation texture has a more twisted, organic look. It's better but there's still more we can change.

4. This step wraps up the tutorial by changing more parameters and, hopefully, further improving the DarkTree.

Open the *Component Editor* for Warp. Crank the values of the ***Density*** and ***Distortion*** parameters up to their maximum of *100*. Drag the new DarkTree to the second Examine Window and compare it with the original. The swirls and twists that have been added to Venation really do look like the caustic light effects that water makes on a swimming pool bottom on a hot, bright summer's day.

You can study the component configuration for this DarkTree in *Figure 3.7* And the finished texture from this tutorial is in the **Textures**/**Tutorials** folder under the name **Pool Lights**.

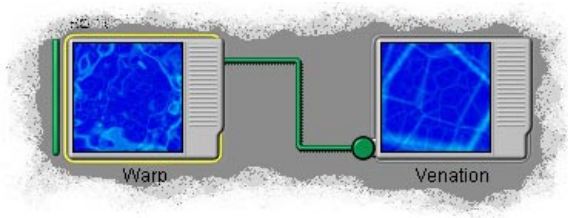


Figure 3.7
The components from the Organic Pool Lights tutorial.

Deform Tutorial 2

This tutorial teaches you how to use a deformer in conjunction with a percent component, usually of the Natural or Noise class, to twist an external image about its center. The results of this type of deformation can range from quite interesting to exceedingly beautiful when you begin with a subtly colored image.

Twisting an Image

1. As usual, for Step 1 we'll just get the components plugged into their respective sockets.

In the *Component Bin*, drag Radial Twist from the Deform folder to the **Root** socket. Paste it as a color-type. Next, open the Natural folder and drag the Clouds component to a level two socket, say, directly to the right of the **Root**. Paste this one as a percent-type. Lastly, open the External folder and select the Image component. Plug it into a level two socket as well, and paste this one as a color-type.

2. For this step, you'll have to find the Image directory and select an image.

First, open the *Component Editor* for color Image. Click on the button labeled ***File***, which opens a standard Windows file requester. Find and open the DarkTree directory, and below that you'll see the ***Image*** directory. Open it, and double click on one of the several image files you'll see there. Doing so will load it into the Image component.

If you can see the black background around the image edges, you can change the ***Background*** parameter to other than black, if you wish.

3. In this step, we'll begin the linking process.

First, link the ***Background*** parameter of Radial Twist to the color Image file that you selected and loaded in the last step. Notice how the image is now twisting around its center.

4. Now we'll make another link, thus finishing the tutorial.

Link Radial Twist's ***Distortion*** parameter to the percent Clouds component. Pretty cool results, don't you think? Try using some other percent parameters, especially from the Noise and Natural folders. Rounded or smooth-edged components work best here. Using sharp-edged components as the percent link to the ***Distortion*** parameter give very unsatisfactory results. *Figure 3.8* is a *workspace* screenshot of this DarkTree.

A DarkTree of this tutorial is in the **Textures/Tutorials** folder under the name of **Image Twist.**

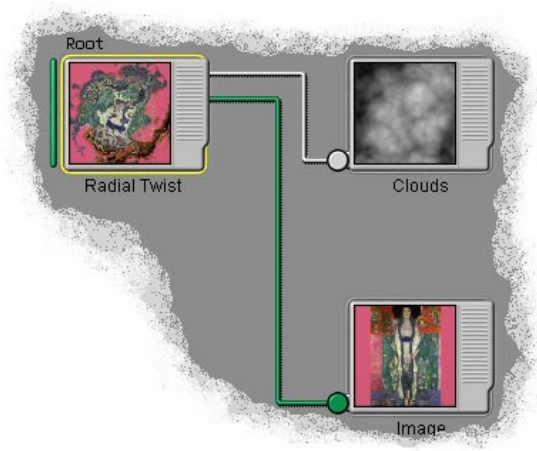


Figure 3.8
The components and their proper positioning for the Twisting an Image tutorial.

3.2.5 The External Tutorial

External Tutorial 1

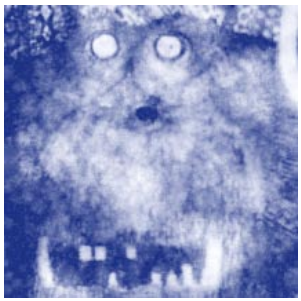
Clouded Face

1. For this step, we'll get the components plugged into their respective sockets, and then select/load an external image.

Open the Natural folder in the *Component Bin*. Select the Clouds component and plug it into the **Root**. Paste it as a color-type. Next, open the External folder, and choose the Image component. Drag it out and plug it in to a level two socket. Paste this component as a percent-type.

Open the *Component Editor* for Image, then press the ***File*** button. This will open a standard Windows file requester. Find the directory where you have DarkTree Textures installed. There you'll see a subdirectory named **Images**. Select ***Face.bmp.*** The Image component will build a percent version of the requested file. The requested file, however, is a color file.

2. This step handles the linking and makes some suggestions about how you can adjust the ***Clouds*** parameter to get a cloudier face.



BOO!
Figure 3.9
The clouded monkey face from the External Tutorial 1.

To begin, open the *links list* for the **Root** component, Clouds, and choose ***Puff Size***. Link it to the Image component.

Here are some hints on manipulating the parameters of the Clouds component, so that you can get a more realistic cloudy face. Increasing the value of the ***Brightness*** parameter will lighten the entire DarkTree. And increasing the ***Contrast*** parameter will sharpen the blending between the two colors that make up the clouds and background. Increasing the values of the ***Roughness*** parameter gives a higher level of detail and more layers of puffs. ***Density***, of course, refers to the density of the clouds. Try different values for these parameters until you're satisfied.

For an image, one with sharply-defined features works less well here than one that's a bit more wispy.

You'll find a DarkTree example of this tutorial, named **Clouded Face**, in the **Textures**/**Tutorials** folder in the Texture Library. And the following figure, *Figure 3.10*, is a screenshot of the Editor *workspace* with the open DarkTree from this tutorial.

Figure 3.10
The components for the Clouded Face tutorial. You can choose from a number of external images for this tutorial.

3.2.6 The Generator Tutorials

Generator Tutorial 1

This very simple tutorial demonstrates how a generator component can control the shape of a pattern.

Rick Rack Stripes

1. Step 1 instructs you to plug the components into various sockets.

From the Pattern folder in the *Component Bin*, choose the Stripes component and drag it to the **Root** socket. Paste it as a color-type. Next, from the Generator folder, choose Sine Wave. Plug it into a level two socket, directly to the right of **Root**.

2. This step tells you what to link, essentially completing the DarkTree.

From the Stripes *links list*, select the ***Center*** parameter. (Notice that ***Center*** is a percent-type but also a function parameter.) Link it to Sine Wave.

3. Take a good look at the Stripes pattern. It's easy to see how the generator is affecting the shape of the center stripe, because it causes the stripes to follow the shape of a sine wave. This is a very basic tutorial, but it demonstrates quite well how a generator component can change the shape of a linked percent parameter. *Figure 3.11* (following) shows the opened DarkTree for this simple tutorial.

There's an example DarkTree file for this tutorial, named **Rick Rack**, in the Texture Library in the **Textures**/**Tutorials** folder.

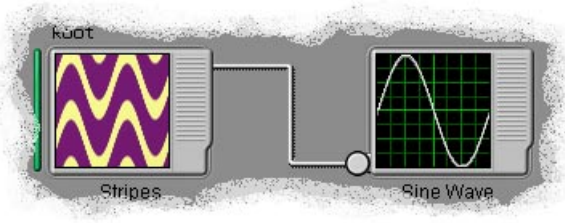


Figure 3.11
The two components that make up the color-type DarkTree tutorial, Rick Rack Stripes.

Generator Tutorial 2

This demonstration of the Generator class illustrates what happens when a generator is linked to the ***Blend Function*** parameter of another component.

Fuzzy Mortar

1. First, as usual, we'll plug in the components for this tutorial.

From the *Component Bin*, open the Pattern folder and select Flagstones. Plug the component into the **Root** socket, pasting it as a color-type. Next, open the Generator folder and choose Gain. Plug this component into a level two socket.

2. Step 2 takes care of the basic linking/parameter-editing chores.

Open the *Component Editor* for Flagstones and change the ***Mortar Width*** parameter to *20.0*. This makes the mortar very thick. Close this *Editor*. Before doing anything else, open an Examine Window from the Flagstones *component menu*. Select ***Surface Color*** from the popup dialog. We'll use this basic view of the flagstones as a reference.

Now open the *links list* for the same component, and link the ***Blend Function*** parameter to the Gain generator. Open a new Examine Window, also from the Flagstones *component menu*, for comparison with the first one. You can see the finished DarkTree laid out on the Editor *workspace* in *Figure 3.12*.

3. In this step, we'll analyze what's going on.

Notice how the mortar blends with the stone in the second Examine Window. The width of the mortar defines the area in which the Gain generator is active. Because the Gain function is fairly steep, the transition is sharp. When you look at Gain, keep in mind that the lower function values translate to the Flagstone color and that the higher values translate to the mortar color.

Let's try some other values for the ***Gain*** parameter. First, open the *Generator Editor* and select the ***Root*** button from the ***Views*** area. This way you can glance at the *update window* as you make your changes. Second, replace the current ***Gain*** parameter value with some new ones. Do you see how you have control over the sharpness or steepness of the mortar blending?

A DarkTree for this tutorial is in the Texture Library. It's in the **Textures**/**Tutorials** folder under the name **Fuzzy Mortar**.

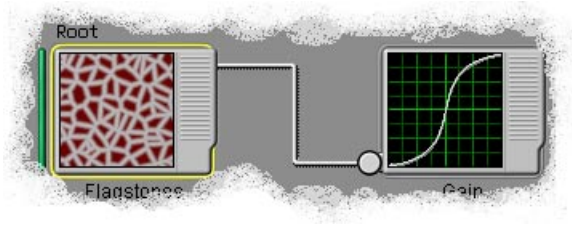


Figure 3.12
The two components for the Fuzzy Mortar tutorial.

Generator Tutorial 3

This tutorial illustrates a third use of the versatile Generator class. Here, we'll use a generator to control the shapes of a grid of dots.

Profiling the Dots

1. In this step, we'll retrieve the components we need.

As with many of the classification tutorials, we only need two components from the *Component Bin*. First open the Pattern folder, and drag out the Dots component. Install it in the **Root** socket, and paste it as a bump-type. Second, we'll need the Sine Wave component from the Generator folder. Drag it onto the Editor *workspace*, and plug it into a level two socket.

2. We'll handle the linking first in this step, followed by some remarks for analysis.

Select the ***Bevel Profile*** parameter from the Dots component *links list*. Link it to the Sine Wave generator. That's it!

Actually, it's a good idea to open an Examine Window now, so that you can get a closer look at the grid. Open it as you've been instructed to in the other tutorials. After looking at the dots closely, notice how their profiles or shapes follow the shape of a sine wave? Remember, the line plot of a sine wave that you see as the component image on the Editor *workspace* is only half of the dot profile. The other half is a mirror image of the first. If you look at a dot center, you'll see that there is a smaller and lower bump. That center bump matches the right side part of the sine wave plot.

If you opened an Examine Window, try looking at this tree as a raw elevation map. To do that, click on the button farthest to the right, along the bottom of the Examine Window. Then choose ***Surface Bump*** from the drop down menu.

Also, you might like to substitute some of the other generators for Sine Wave. Not all of the generators are suitable as bevel profilers, but Bell, Noise and Round are reasonable.

You'll find an example DarkTree for this tutorial, named **Dot Profiles**, in the **Textures**/**Tutorials** folder that is in the Texture Library. And you can see the open DarkTree from this tutorial in *Figure 3.13*.

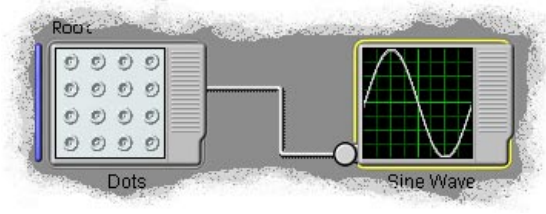


Figure 3.13
Component positioning for the Profiling the Dots tutorial.

3.2.7 The Process Tutorials

Process Tutorial 1

This is the first of two process tutorials. It uses a process component to dirty or darken another component.

Darkening the Rising Sun

1. Following our usual division of chores for the tutorials, we'll drag out and plug in all of the components for the DarkTree first.

To begin, from the *Component Bin* locate and open the Process folder. Drag the Darken component out, then plug it into the **Root**. Paste it as a color-type. Next open the Pattern folder, and fetch the Rising Sun component. Plug it into a level two socket, and paste it as a color-type as well. Last, open the Noise folder and locate Fractal 1. Plug it into a level two socket, and paste it as a percent-type.

2. We'll begin the linking process in this step.

From the Darken component *links list*, select the ***Background*** parameter and link it to the Rising Sun component. Notice how the sun and its rays are darkening a little?

3. In Step 3, we'll continue giving Rising Sun the darkening treatment.

Choose the ***Mask*** parameter from the Darken *links list*, and link it to Fractal 1. This link gives Fractal 1control over the dirty or sooty look that the Rising Sun is taking on.

4. For this final step, we'll streak the soot a bit..

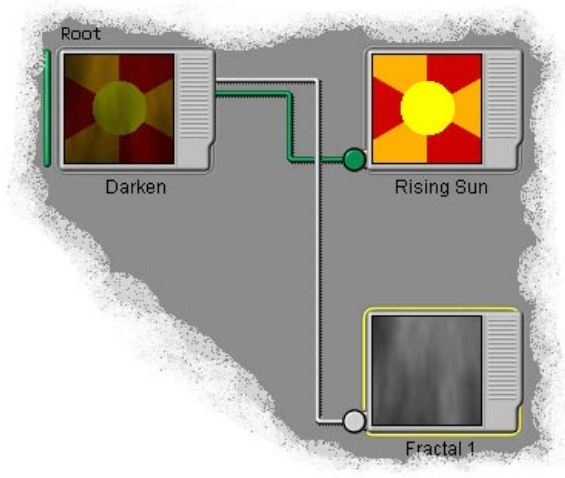


Figure 3.14
The three components for the Darkening the Rising Sun tutorial and their proper positioning.

Open the *Component Editor* for Fractal 1. Click on the ***Scale*** button, and disable the even-scaling check box (remove the check mark). Now enter *300.00* into the Y-axis input box for the scaling. You can try playing around with the ***Brightness*** and ***Contrast*** parameters of Fractal 1 to sharpen or vary the darkening effect on this pattern.

Figure 3.14 shows the arrangement of components for this DarkTree. And you'll find the sample DarkTree file for this tutorial in the Texture Library. The name is **Dark Sun** and it's in the **Textures/Tutorials** folder in the Texture Library.

Process Tutorial 2

This example from the Process class shows you how to slice off the tops of your bumps. Many of the process components are quite useful in manipulating bump textures. Here **Minimum** is used to a chip out areas of a flat plane.

Truncated Bumps

1. As always, Step 1 tells you what components you'll need for this short tutorial.

From the *Component Bin*, open the Process folder and select the Minimum component. Plug it into the **Root** socket, and paste it as a bump-type. Next, select Pestilence from the Natural folder. This component should be plugged into a level two socket, and pasted as a bump-type also.

2. In Step 2 we'll link the two components, then talk about what you can do with this type of Process component.

There's only one link to make for this tutorial. Open the *links list* for Minimum, and select the ***Bump A*** parameter. Use this parameter to make a link to Pestilence.

You might have trouble seeing what's happening to the bump. If so, open an Examine Window from the Root component's *component menu*. Select ***Surface Bump*** from the popup dialog. Vary the value for the ***Bump B*** parameter of Minimum to see how it affects the image. Change the cut-off point by raising or lowering ***Bump B***. If you want to do a color version to match the bump tree, use the Minimum Switch component. *Figure 3.16* is a screen capture showing this DarkTree on the Editor *workspace*.

In the Texture Library, you'll find a sample DarkTree for this tutorial named **Cut Bumps**. It's located in the **Textures/Tutorials** folder.

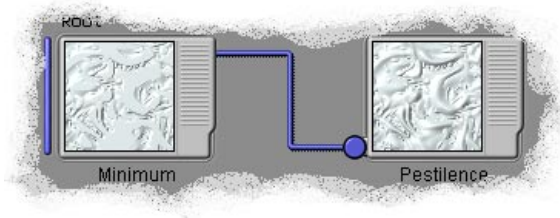


Figure 3.16
Positioning the two components for the Truncated Bumps tutorial.

3.2.8 The Tile Tutorial

Creating a Seamless Tiled DarkTree

Tiled Blobs

This tutorial demonstrates how to make a texture that will tile seamlessly.

1. First, we want to be able to view the texture as it looks when tiled. Therefore, open the *Properties page* by clicking on the ***Properties*** tab at the bottom of the *Edit Control Panel*. (More than likely the panel is currently showing the *Component Bin* page.) Find the ***Mapping*** section and change the mode to ***Frame***. Frame mode is the only one that allows tiled previews.

2. Now let's place the components we'll need for this texture. But first, change the *Properties page* back to the *Component Bin* by clicking on the ***Components*** tab.

Open the Shader folder and drag out the DarkTree Shader. Drop it on the **Root** socket. Next, open the Tile folder. Find and drag the WarpTile component to the level-two socket directly to the right of **Root**. Paste it as a color-type. To a level-three socket, directly to the right of color Warp Tile, paste a color-type Blobs component from the Natural folder.

3. In this step, we'll link up the components from step 2, to create our DarkTree.

Open the DarkTree Shader *links list* and find the ***Surface Color*** parameter. Link it to the color Warp Tile component. Now open the Warp Tile *links list*, select ***Background*** and link it to the color Blobs component.

4. Next, let's make some adjustments to the Warp Tile component.

Open the *Component Editor* for Warp Tile by double-clicking anywhere on the component. Look at the *update window* first. Notice that the Lumps image appears distorted. The reason for this is that Warp Tile works by distorting the texture space (Blobs in this case) through its linking ***Background*** parameter. This creates a seamless tile without doing any edge blending.

Try the ***Ivarian Warp*** toggle and play with the ***Symmetry*** and ***Scale Factor*** parameters until you get a look that you like. Since we're building a square tile, you should leave the ***Aspect Width*** and ***Aspect Height*** parameter values as they are. Under the ***Views*** section of this *Component Editor*, check (turn on) the ***Show Tiled*** check box. This will display a four-tile group so you can see what the tiling actually looks like. Close the Warp Tile *Component Editor*.

Hint: *You'll nearly always want to use 3D components in conjunction with Warp Tile. 2D components do not work well.*

5. In this step, we're going to make some adjustments to color Blobs.

Open the *Component Editor* for color Blobs. First, turn on the ***Show Tiled*** toggle, like you did in the previous step. Looking at the Blobs' *update window,* you'll notice that the texture still is not tiling as we wish. Click on the ***Parent*** button in the ***Views*** area. This will basically let you make your adjustments to color Blobs, while previewing the results from the perspective of the parent component, Warp Tile.

Freely adjust the Blobs component parameters while keeping an eye on the *update window.* Don't forget to play with the transforms as well. Sometimes a slightly different position for a texture improves the tiling.

6. And finally, let's add some bump to our tile to give it a bit of depth. That basically will finish this tutorial.

After closing the Blobs *Component Editor*, right-click one time on the Warp Tile component. When its *component menu* comes up, select ***Copy Subtree.*** Next right-click on the level-two socket directly below Warp Tile to open the empty socket's *component menu*. Choose ***Paste As Bump***. Now you have a tiling bump subtree that perfectly matches the current parameter values that you've set for the two color components. Open the DarkTree Shader *links list* and select ***Surface Bump***. Link this parameter to the bump version of Warp Tile.

To get a real look at your newly-created tile, open an Examine Window using the DarkTree Shader's *component menu*. To see the tiled version of your texture, click on the third button from the right along the Examine Window's lower edge. You'll see a four-square tiled group display.

Figure 3.17, next, is a screenshot of the finished tiling texture. The final DarkTree can be opened from the **Textures/Tutorials** folder. The DarkTree's name is **Tiled Blobs**.

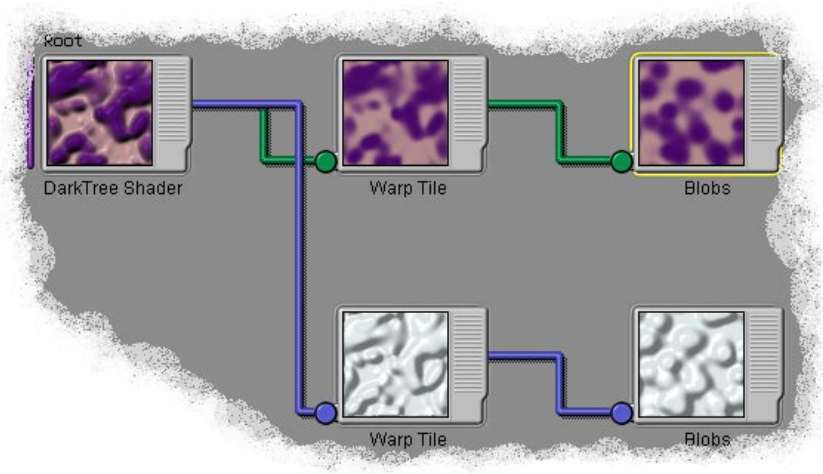


Figure 3.17
The components for the Tiled Blobs tutorial positioned on the DarkTree Editor workspace.

3.2.9 Transform Tutorial

Transform Tutorial 1

This tutorial teaches you how to create a short animation that moves into, and up through, a block of agate using the Transform class component, Translate.

Moving Through the Agate

1. For this first step, we'll plug in the components where they belong on the Editor *workspace*.

From the *Component Bin*, open the Transform folder and select the Translate component. Drag it out and plug it into the **Root** socket, and paste it as a color-type. Next, open the Natural folder, choose Agate and plug it into a level two socket - to the right of **Root**. Paste this component as a color-type as well. And finally, open the External folder and select the Time component. Plug it into another level two socket.

2. Let's go ahead and link the components together for the animation first.

Open the *links list* for Translate and make two wire links. The first link goes from the ***Percent Moved*** parameter to the Time component. The second goes from the ***Background*** parameter to Agate.

3.Since the default animation length is thirty frames and that is exactly the number of frames that we want, you won't have to adjust the Time component at all for this animation. So for this step, we'll add values that tell the animation how far we want to go into and upward through the agate block.

Next open the *Component Editor* for Translate. Change the ***Y Amount*** value to *0.5*. That means that over the length of the animation, travel in the Y direction will be half of the present height of *1.0*. Make sure the ***Z Amount*** is *1.0*. This makes the travel distance for Z one unit block of texture space, over the thirty frames of the animation.

4. Now, let's look at the animation.

First, open an Examine Window from the Translate *component menu*. Choose ***Surface Color*** from the popup dialog. Use the *rough play* button to step through the thirty frames of the animation. Of the controls arranged along the bottom of the Examine Window, this button is second from the left (the button with the arrow on it).

Notice how the animation is moving in two directions at the same time? You can see the animation moving in the Y-direction easily enough. And, you can detect the Z translation by noticing how the texture itself changes as the animation moves into the agate material.

There's a DarkTree that accompanies this tutorial, named **Inside Agate**. It's located in the Texture Library, in the **Textures**/**Tutorials** folder. And *Figure 3.18* which follows, shows the component arrangement for this DarkTree.

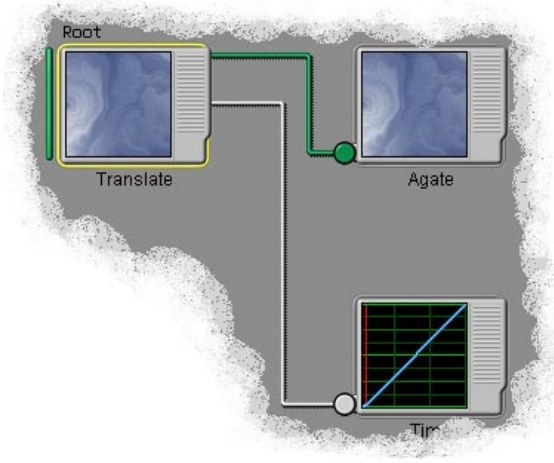


Figure 3.18
The three components that make up the Moving Through the Agate tutorial.

Section 3.3
Transformations in Texture Space

3.3.1 Section Overview

This section covers a number of important points associated with transforming algorithmic texture space, and the complex issues that must naturally be addressed when transformations are applied to the units (components, in this case) that are part of a tree structure.

For many users, transformations performed on components in relative texture space can be quite confusing, especially when the process is not familiar. So, we'll teach you a useful way to visualize texture space, and then show you how to envision translations, rotations, and scalings the way that DarkTree Textures employs them.

You'll also learn something of the problems associated with inheritance and multiple parent ambiguity. These are problems that naturally arise out of using tree structures. The section ends with two tutorials that will help you to better understand parent-to-child inheritance.

3.3.2 Algorithmic Texture Space

The concept of texture space and transformations in texture space can be quite confusing at first. When you look at a DarkTree texture map or rendering, it will appear as a flat, two-dimensional pattern. The images are really only a small piece of a much larger texture space that has been defined by DarkTree's algorithms. We use transformations to change the view of the texture space, either by moving or re-orienting it.

One of the best ways to understand algorithmic texture space is to think of it as being a virtually infinite block of material. For example think of a stone texture as a mass of stone, or a wood texture as a big block of wood. The difference between these textures and real ones is simply that the colors of these materials come from mathematical recipes rather than a conglomerate of sand particles or wood fibers. The plane, sphere or cylinder that you see rendered in the DarkTree Textures previews and Examine Windows is really just a small cut out piece of this algorithmic block of material. It's as though an artist took a chisel to this large block filled with colors and patterns, and carved out a perfect sphere (or plane, cube or cylinder) for you to look at and use in your own work.

3.3.3 Moving in Texture Space

To change your view within algorithmic texture space, use transformations. In keeping with most applications that allow movement in 3D space, DarkTree Textures provides translation, rotation, and scaling as its fundamental transform operations. Translation lets you move anywhere within the texture space. You can use rotation to view the texture space from any angle. And scale allows you to squeeze or stretch texture space, so that you can view more or less of it.

Also in keeping with most 3D applications, all transformations are done with respect to the X-axis, the Y-axis or the Z-axis. These axes provide standard directions for applying translate, rotate, and scale operations. The X-axis extends horizontally, the Y-axis extends vertically and the Z-axis extends into and out of your screen.

Note that some of the DarkTree components are two-dimensional. That means these textures vary in the X and Y directions but not in the Z. As a result, you won't see any changes if you translate along the Z-axis. Also, when viewed in the planar mapping mode, rotations about the X and Y axes will make these textures look like they're stretching rather than rotating. Two-dimensional components are labeled as such in their online help texture reference descriptions.

NOTE: Where they make sense, basic transformations are available for all texture components.

NOTE2: The Transform class of components provides a way to animate an entire texture subtree through space.

NOTE3: The "shuffle" components of the Control class (Shuffle Scale, Shuffle Translate, and Shuffle Rotate) allow the random transformation of individual regions, if the linking parameters are regional.

Translation

To translate basically means to move. Translating along the Y-axis moves your texture view up (+) or down (-). Translating along the X-axis moves your texture view to the left (-) or to the right (+). And translating along the Z-axis moves your texture view in (+) or out (-) of your monitor viewing plane. Translating along the Z-axis can sometimes be confusing because you won't see motion in any particular direction. However you can usually see the texture changing.

NOTE: Translation through Noise and Natural textures can create many varied and interesting special effects.

Rotation

Rotation allows you to turn or rotate your view within the texture space. Rotations are specified in degrees about the X, Y or Z-axis. For example, rotating *90* degrees about the Z-axis will cause the texture to pivot one quarter of the way about the center of the view as you watch it on the monitor.

NOTE: Rotating your texture about X or Y while in the planar mapping mode can look a little confusing. Since the X and Y axes lie in the same plane as your view, you'll see the edges of your texture change rapidly while the center moves more slowly. It's most noticeable during animated rotations.

Scale

Scale operations let you squeeze and stretch texture space. Scalings are uniform by default. In other words, scaling happens equally in all directions. A special check box allows you to change that and scale each of the three axes independently.

Scale values should be thought of as percent changes in size. In DarkTree Textures, scale values of *100%* leave the texture space unchanged. This is the value that scale always resets to. Values of greater than *100%* make the texture space stretch and the texture pattern look larger. Scale values of between *10%* and *100%* make the texture space shrink, condensing the texture. A value of *200%* will stretch the texture space to twice its present size; while a scale value of *50%* will compress texture space by half. Scale values of *0%* are never allowed because they would cause the space to be compressed into a single point. Keep in mind that while scaling a texture up does make the pattern larger, you'll actually be viewing a smaller section of it because of the *update window's* fixed size. Scaling down lets you view more of a pattern, even though it is smaller.

Scaling is an important part of exploring the many possible "looks" a texture can have. And strangely enough, textures can often look significantly different depending on the scale. Non-uniform scaling along an axis allows you to change a texture significantly. For example, a stretched-out Lumps texture can look like muscle fibers. And the simple Fractal noise component that has been stretched can look like the weathering on a fighter plane. It's definitely worth your time to explore some of the Noise and Natural components at various scales and proportions.

3.3.4 Transformation Inheritance

Any time a transformation is applied to a component, the same transformation is applied to the children of that component. This is called "transformation inheritance". Every child texture inherits the transformations of its parents. However, parent

components do not inherit their children's transformations. Thus transforming a child texture won't affect its parent. Likewise, a child component will not inherit parent transformations that were performed prior to linking. The ***Align*** button in the Component Editor allows you to update the transformations of a newly-linked child component, should you want to align it with its parent.

Transformation inheritance ought be familiar to anyone who has worked with hierarchical 3D objects. We'll use the following example to illustrate what we're talking about. A model of the human figure has the body as its root object and the limbs as child objects. If you move the body, all the limbs (children) receive the same transformation, and so move with the body. However, if you move a limb, say an arm, the hand (a child) will move with the arm, but the body (the parent) will not be affected.

Just as transformation inheritance makes working with 3D models much easier, it makes transforming DarkTrees much easier as well. Once you have all the textures in a component subtree aligned the way you want them, you only have to transform the root of the subtree (the subroot) to move the entire texture. *"The Inheritance Tutorial" on page 178* is one of the Transformation Tutorials, and illustrates transformation inheritance in action.

3.3.5 Relative Transforms

Transformations done within DarkTree Textures are applied relative to your view of texture space. In other words, the center point of your view is always the origin of your next transformation, regardless of where the real center of the texture space is. The positive Y-axis is always up, the positive X-axis is always to the right, and the positive Z-axis is always in, regardless of the previous rotations that might have taken place. This allows you to freely apply transformations to your textures, without worrying about where you really are in texture space. You simply apply transforms until you like what you see.

One problem with these view-relative transformations is that you don't really know where you are in texture space. That's where the ***Align*** and ***Clear*** operations come in. ***Clear*** will take you back to your original view of texture space by removing all transformations. ***Align*** copies a parent's transformations and applies them locally, bringing the target component into line with its parent. Any previous transformations on the child are wiped out. The transform-related ***Copy*** and ***Paste*** operations give you the additional ability to copy the transformed location of one component to another, allowing you to align completely unconnected textures.

*NOTE: The common parameter **Lacunarity** is a factor that scales the levels of a fractalized pattern. **Lacunarity** does not scale relative to your view, as described above, but scales based on the true origin of the*

*texture space. Because of this, the transformed location of a component with an animated **Lacunarity** parameter will affect the look of the animation differently. In general, you'll want to keep such a component close to the origin (that is, don't translate). Online help, accessible from most Component Editors, has more detailed information on this parameter.*

3.3.6 Multiple-Parent Ambiguity

While editing your DarkTrees, one thing to watch out for is multiple-parent ambiguity. If a child component in the DarkTree Editor is linked to two or more parents, then only one of the parent's transforms can be rendered on the component image. It becomes unclear which parent's transform will affect the child, if a component has multiple parents (with differing transforms). When rendering such a component, only one image is being rendered so only one parent's inherited transform can be shown. The DarkTree Editor arbitrarily chooses one parent's inheritance for the rendering. The ***Multiple Parent Ambiguity Tutorial*** illustrates this point.

Multiple-parent ambiguity is really only a problem when displaying DarkTrees in the DarkTree Editor. Internally, DarkTree Textures doesn't have any problem at all determining which parent to inherit from during rendering. It's only when editing the transforms of a component with multiple parents that you may get confused. In general, it's better to avoid this problem by making sure that if you choose to link more than one parent to a component you keep all the parent component's transformations identical.

*NOTE: The **Align** operation also becomes ambiguous when there are multiple parents. In this case, **Align** will arbitrarily align the texture with just one of the parents*

3.3.7 Transformations Tutorials

The Inheritance Tutorial

This tutorial demonstrates transformation inheritance. After completing it, you'll be able to clearly see two basic points: (1) that each child texture inherits the transformations of its parents, and (2) that parent components do not inherit their children's transformations.

1. For Step 1, we'll fetch the two components that we're going to be using. Then we'll change the Editor mapping mode to make the transformations and their effects on the components clearer.

First, open the Pattern folder on the *Component Bin*. Select the Rising Sun and drag it to the **Root**. Plug it in, and paste it as a color-type. Plug another Rising Sun into a level two socket. Paste this one as a color-type also.

Next, select the *Properties* tab on the *Edit Control Panel*, find the **Mapping** control and choose ***Spherical***. This will change all of the component images on the *workspace* to spherical from the current planar

2. For this step we'll be making some changes to the **Root** component, again so that the inheritance will be clearer.

Open the **Root** *Component Editor*. Change both the ***Sun Size*** and the ***Ray Size*** parameters to *30.0*. Next, edit the color parameters, ***Sun*** and ***Ray***, and give them differing shades of blue. When the two Rising Sun components are linked, you'll be able to distinguish the level two Rising Sun from the one in the **Root** position.

3. And now for the linking!

From the Rising Sun in the **Root** socket, open the *links list*. Choose ***Background*** for the link to the second level Rising Sun.

4. In this step, you'll be demonstrating the first point to yourself. The point in question is that each child texture inherits the transformations of its parents.

Open the *Component Editor* for **Root**. Rotate the Rising Sun about each axis several times. Rotate in increments of ten degrees at a time, while watching the component change in the *update window*. Notice how the child Rising Sun (the yellow and red one) rotates with the **Root** Rising Sun? It's inheriting its parent's rotations.

5. Step 5 demonstrates the second point.

Open the *Component Editor* for the level two Rising Sun. Before you do anything else, change the view to **Root** so you can get the full picture concerning what's going on. Next, go through the same incremental rotations, several times for each of the three axes. Do you see how the blue Rising Sun just floats on top, while the level two (red/yellow) Rising Sun moves underneath? This is demonstrating point two, that parents don't inherit their child's transformations.

Figure 3.19 shows the component layout for this DarkTree. And you'll find the sample DarkTree that accompanies this tutorial in the Texture Library, in the **Textures/Tutorials** folder. It's called **Inherit.**

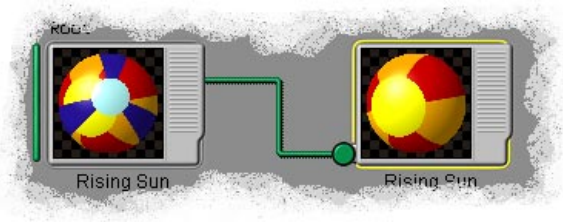


Figure 3.19
The Inheritance tutorial DarkTree showing the results of this problem.

Multiple Parent Ambiguity Tutorial

The following brief tutorial illustrates the problem of multiple parent ambiguity, a situation where a child component has two parents. While it does inherit information from both parents, it will only render the inheritance it has received from one of them.

1. This step tells you which components to drag to the *workspace*, where to plug them in, and so forth.

First open the Noise folder on the *Component Bin*. Drag two Fractal components onto the *workspace*, one at a time. For this tutorial, we aren't going to plug anything into the **Root** socket. So plug the first Fractal into a level two socket, and the second into another level two socket directly below the first. Make both color-types. And last, from the open Pattern folder, select a Rising Sun component, and plug it into a level three socket. Paste it as a color-type also.

2. Now let's take care of the wire links!

Open the *links list* of each Fractal component, and make wire links between the two ***High*** parameters and the Rising Sun component. Each Fractal component image should now reflect a rising sun pattern.

3. In this step, we'll perform different translations on the two Fractal components.

Open the *Component Editor* for one of the Fractals, and translate it by *50.0* along the X-axis. Next, open the *Component Editor* for the other one, and translate it by *50.0* along the Y-axis.

Now exit the *Component Editor* and look at the linked components. As you can clearly see by looking at the Rising Sun image (the child component in this case), only the inheritance from one Fractal is displayed.

4. This step demonstrates what happens when you remove the link to a parent. The parent should be the one whose transformation is currently being displayed by the child component.

Delete the wire link between one of the ***High*** parameters and the Rising Sun. Make sure it's the one whose inheritance is currently rendered on the Rising Sun image. Notice how the child component image changes, and begins displaying the other parent's inheritance?

The accompanying DarkTree for this tutorial is in the Texture Library, in the **Textures**/**Tutorials** folder. The name is **Ambiguity.** And *Figure 3.20*, which follows next, shows the component layout you should see on the *workspace* after finishing this tutorial.

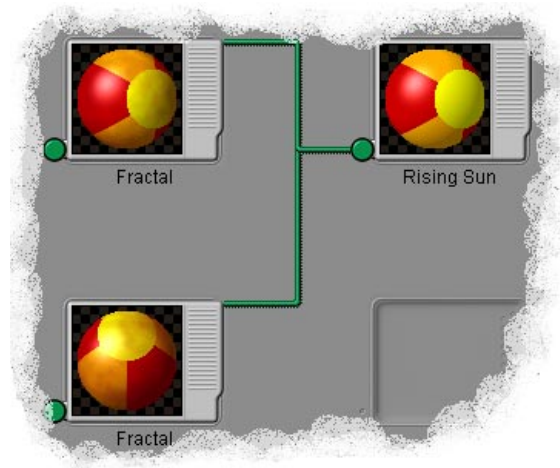


Figure 3.20
The color-type DarkTree from the Multiple Parent Ambiguity tutorial.

Section 3.4
Working With Bumps

Section Overview

Creating high-quality bump textures is a very important aspect of designing realistic textured surfaces. Bump components are generally quite intuitive and easy to use, even if you don't understand how they work. In some ways, working with bump textures is the same as working with color or percent textures. However, to take full advantage of DarkTree Textures' bump manipulating capabilities, it's important to understand how bump textures are created and rendered. This section explains the concepts behind bump textures, and gives you some reliable methods for working with and combining them.

3.4.1 Elevations Define Bumps

Bump textures are defined by elevations. Just as color-type components generate a unique color for every point in a color texture, bump texture components generate a unique elevation for every point in a bump texture. Those elevations are used to determine the curvature of a surface during shading, and result in a bumpy-looking skin. You can think of a bump texture as being like a topographical map of a bumpy surface. The Examine Window renderer and the DarkTree Shader component use that map to determine how directly the light source shines on each point of the surface, and hence, how bright each point on the surface should be.

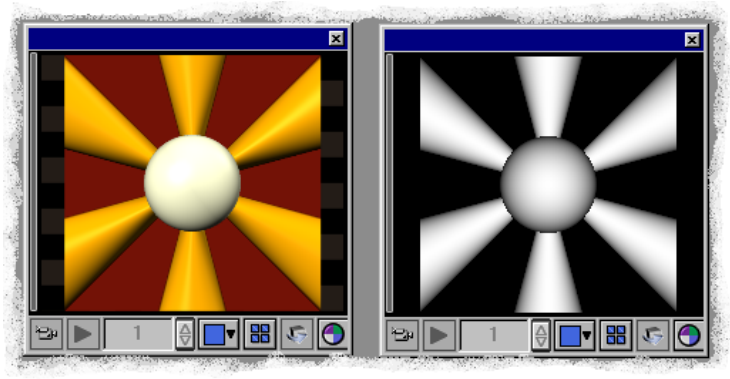


Figure 3.21
A Rising Sun component. On the left is the shaded DarkTree version; on the right is an elevation map of the bump-type component.

NOTE: Some texture components take advantage of the fact that bump textures are simply elevations. For example, the Maximum component from the Process class compares the elevations of two child bump textures, and chooses the higher of the two.

3.4.2 Bump Parameters and Bump Scale

When editing a bump component, you'll need to set some bump-type parameters. As explained earlier, DarkTree Textures defines bumps as elevations. As a result, bump parameters are elevation values. For example, the Fractal bump component has two bump parameters: ***Low*** and ***High***. The ***Low*** parameter defines the elevation of the lowest point in Fractal, and ***High*** defines the highest. The larger the range between the two parameters, the more extreme your bump will be.

As a general rule, you should set the bump parameters between *-1.0* and *+1.0*. DarkTree Textures' previews and shaders look best with elevation differences of *1.0* or less, although you aren't constrained to that. Bump textures that have small-scale detail usually look better with even smaller elevation ranges. Larger value ranges are generally too extreme and, therefore, won't look as good.

Most bump textures include a bump scaling parameter, ***Bump Scale***. After the appropriate algorithm determines the elevation for a given point in a bump component, it's then multiplied by ***Bump Scale***. This allows you to increase or decrease the bump component's elevations through a single parameter. A ***Bump Scale*** of *0.5* reduces the elevation range of a bump by a full *50%*. Values greater than *1.0* increase a bump component's elevation range. And negative ***Bump Scale*** values invert a component's bumps.

3.4.3 Maintaining Detail

When the Bitmap Renderer creates a bump map, small details sometimes get lost in the process. In this subsection, we'll explain why this happens and what can be done to fix the problem. If you're rendering shaded textures exclusively and don't need to create raw bump maps, you can skip this section.

The Renderer saves bump maps as grayscale images. Each point in a grayscale bump map defines an elevation. Dark gray to black points represent the lower elevations, and light gray to white points represent the higher areas. Black, of course, represents the very lowest point of your elevation, and white represents the highest. Three-dimensional applications use these elevations to make two determinations. They use them either to determine the amount of displacement of the 3D object's surface (called displacement mapping) or to determine how much to perturb the object's surface normals, for shading (called bump mapping).

Internally, DarkTree Textures uses a virtually unlimited range of elevations to define surface bumps. Bump maps on the other hand are 8-bit, or 256 level grayscale images. Since each shade of gray represents a different elevation, bump maps only provide 256 elevations. When the program renders a bump DarkTree, the lowest and highest elevations in the texture are assigned to black and white respectively. All 254 shades of gray now represent equally distributed elevations between the lowest (colored black) and highest (colored white) elevations of the bump texture. If a bump DarkTree contains both large and small elevations, the smaller ones tend to get lost. *Figure 3.23* illustrates this problem. The 256 levels of the grayscale bump map will be used to define the large elevation range, leaving very few or no levels to define the smaller bump elevations.

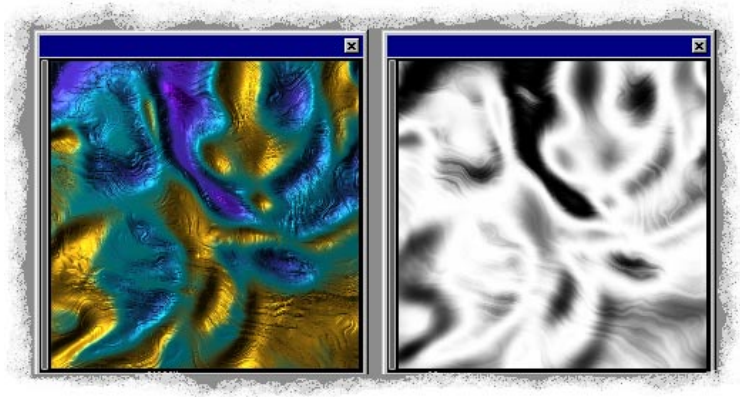


Figure 3.23

This figure shows, on the right, a raw elevation map of the shaded DarkTree on the left. Note that not all of the fine detail shows up in the raw elevation map.

The key to maintaining adequate detail in your bump maps is to control the elevation ranges of each component in your bump DarkTree. As mentioned in the previous subsection, each bump component allows you to explicitly set upper and lower elevation bounds. As long as you keep your elevation ranges similar in both size and location, you won't have any detail problems when rendering your bump maps. The Examine Window provides raw elevation rendering that can help you detect and identify any loss in detail. If you can easily see the important details in the raw elevation mode, your rendered bump map will be fine. If any part of the bump texture doesn't show up, you'll need to adjust your component's elevation ranges. The *"The Bump Detail Tutorial" on page 187* demonstrates how to detect and fix bump scaling problems.

3.4.4 Manipulating Bumps

A number of texture components rely on the fact that bumps are defined by elevations. These components are particularly useful in the construction of bump DarkTrees. Following are examples of such components and how they rely on the concept of elevations to create their effects.

Comparing Bumps (Maximum and Minimum)

The bump version of both the Maximum and Minimum components compare the elevations of their children. Maximum takes the larger of the child elevations for any given point. And of course, Minimum takes the smaller of its children's elevations. These two Process class components make it easy to embed one bump texture into another.*Figure 3.24* shows two bump components that have been combined using

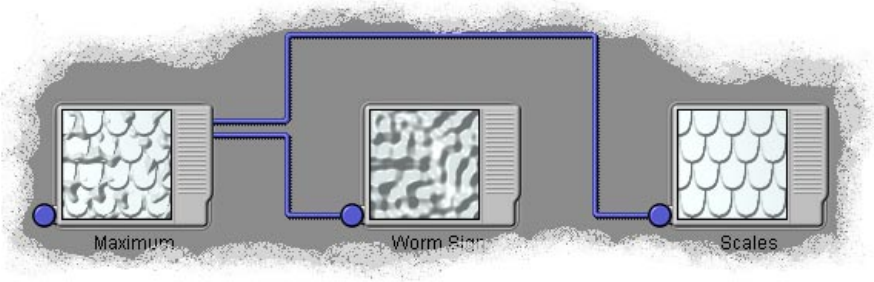


Figure 3.24

This DarkTree shows how the Maximum component combines Scales and Worm Sign.to create some infected-looking, possibly bleeding, scales.

Maximum.

Combining Bumps (Add, Multiply, and Subtract)

The Process class components Add, Multiply and Subtract apply basic mathematical operations to their bump texture elevations.

ADD: Add takes the elevations of two child bump components and adds them together. This is great for layering one bump texture on top of another, without altering the basic appearance of either algorithmic pattern.

SUBTRACT: Subtract takes the elevations of the second child bump component and subtracts them from the first. This allows you to carve out the pattern of one texture from the pattern of another.

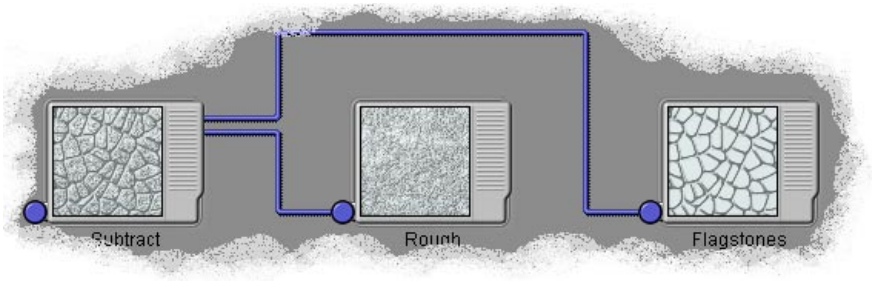


Figure 3.25

An example of using a Subtract component to create a nice rough flagstone texture.

MULTIPLY: The Multiply component actually produces the same type of results as the ***Bump Scale*** parameter that's found in most bump components. The elevations of the individual bump children are multiplied together. In effect, the child bump components scale each other's elevations.

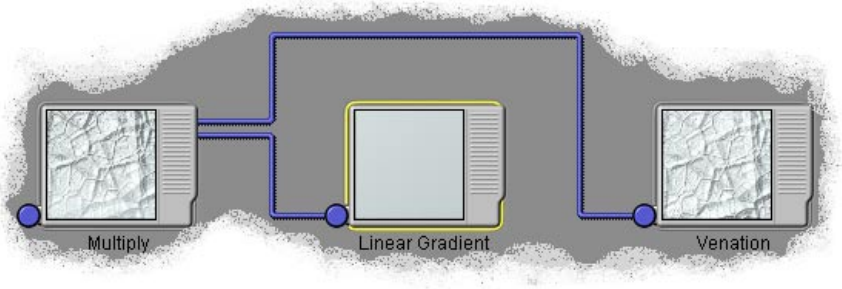


Figure 3.26

An example of using Multiply to mix the Venation and Linear Gradient components. The result is a gradual fading of the veined pattern.

Coloring Bumps

In some cases, bump DarkTrees aren't easily reproducible as percent or color DarkTrees. This is where the Elevation Gradient component comes in.

ELEVATION GRADIENT: Elevation Gradient allows you to set up to five colored layers, at various elevations. The color for a given point is then determined by the elevation of that point in the child bump texture. In essence, you can use the Elevation Gradient component to convert bump textures to color or percent textures.

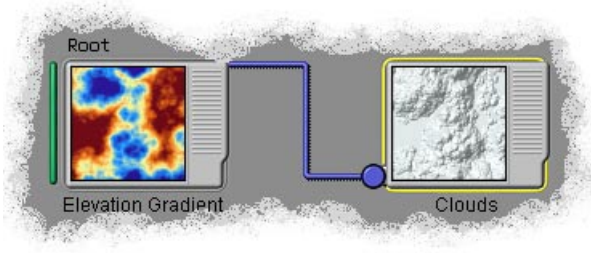


Figure 3.27
This screenshot is an example of an Elevation Gradient component being used to color the varying heights on a bump-type Clouds component.

3.4.5 Bumps as Controls

In a couple of places in DarkTree Textures, bump-type components are used as controls instead of bump-generating elevations. Specifically the Transform class components, Rotate, Scale, and Translate, use linkable bump parameters to control the amount of transform they apply. With these components, the controlling bump parameters are used as values rather than elevations. We don't use percent parameters in this case because they are bounded to a specific range. Bump parameters, however, can be set to any reasonable value.

NOTE: You can create some interesting melting and stretching effects by using uneven transformations. Controlling the amount of transform causes some parts of the resulting texture to transform farther than others.

3.4.6 A Bump Tutorial

The Bump Detail Tutorial

This tutorial shows you how to check the elevations on your bump trees, and how to ensure that you'll end up with the level of detail you expect. With *Figure 3.28*, you can check a screen capture of the finished DarkTree for this tutorial

1. This step places the components for the tutorial.

From the *Component Bin*, open the Pattern folder, and select the Bricks component. Drag it to the **Root** socket, pasting it as a bump-type. Next open the Noise folder. Drag out the Rough component and plug it into a level- two socket. Paste Rough as a bump-type as well.

2. Now we'll give the bricks some surface roughness.

Open the *links list* for Bricks, and select the ***Brick*** parameter. Make a wire link from this parameter to the Rough component. Notice how the rough texture shows up on the bricks now?

3. In this step, we'll check to see exactly how well the existing rough texture will be preserved after this DarkTree is written to a bitmap.

First, open an Examine Window from the Bricks *component menu*. Select ***Surface Bump*** from the popup dialog. Click on the button that is farthest to the right along the lower edge of the Examine Window. Specify ***Surface Bump*** from the (now) open drop down list. This tells the Examine Window to render your DarkTree as a raw elevation map. Notice that little to none of the surface roughness shows up. The raw elevation mode gives you quite an accurate picture of how much bump detail will still be visible after the DarkTree is rendered to a bitmap.

4. Step 4 shows you some adjustments you can make to solve this elevation problem.

Open the *Component Editor* for Bricks and note the ***Bump Scale*** parameter. The default value for this parameter in the Bricks component is much larger than the default value given to the same parameter in the Rough component. So first, reduce Brick's ***Bump Scale*** to *0.4*. Close the Bricks *Component Editor,* and open the *Component Editor* for Rough. Increase its ***Bump Scale*** parameter to the same amount, *0.4*.

Now drag the **Root** to the Examine Window again. Notice how much more bump texture is preserved on the brick faces. Be sure to always check your bump trees by using the raw elevation option in the Examine Window before generating any bump-type bitmaps.

The DarkTree that accompanies this tutorial is in the Texture Library, in the **Textures**/**Tutorials** folder. Its name is **Detail**. And following is a screen capture of this DarkTree, laid out on the DarkTree Editor.

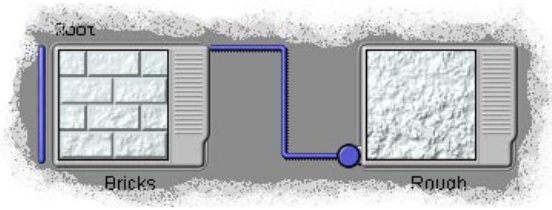


Figure 3.28
The component arrangement for the Bump Detail tutorial.

Section 3.5
Animating Your DarkTrees

Overview

DarkTree Textures has some simple but very powerful texture animation abilities. This section provides a basic explanation for how DarkTrees are animated, as well as giving you some tips on managing more complex animations. For example, we'll explore four separate methods for creating seamless looping animations.

3.5.1 Animation Basics

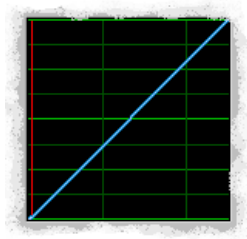


Figure 3.29
A default linear time plot from the Time component.

With version 2.0, animating DarkTrees is extremely easy. All you have to do is link the parameter that you want to animate to a Time component. You may use any number of Time components for a single DarkTree and/or link several parameters to a single Time component.

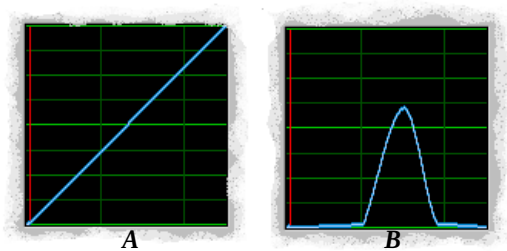
Time components display 2D graphs, just as generators do (see the default in *Figure 3.29*). Time graphs are colored blue to indicate that they are in the time domain (regular generator plots are white). The horizontal axis represents the frame range in a DarkTree's animation and is called the *Time Axis*. The left edge is the first frame, and the right edge is the last frame in the animation. By default, that range is thirty frames. But you can easily increase or decrease that number by clicking on the ***Properties*** tab on the *Edit Control Panel*. The vertical axis, referred to as the *Output Axis*, shows the percent value that the Time component will output for each frame. The lowest point vertically is *0.0%*; the highest is *100.0%*. Note that Time components output only a single percent value for any given frame number.

NOTE: In DarkTree Textures, all animated events are set in terms of frames. How this is interpreted in real time is dependent on how the animation is played back. For example, video typically requires thirty frames per second; thus thirty frames are required for every second of video playback.

If your animated DarkTree includes two or more Time components but you don't want the animated portion of each parameter to be of the same duration, you should still make sure each instance of Time has the same frame range. *Figure 3.30* shows you how to set up such an animation.

Figure 3.30

This set of plots controls two parameter animations for a single DarkTree. Both are 30 frames but plot (B) begins animating at frame 10 with a value of 0.0%, rising to a value of 60%, then dropping back to 0.0% at frame 19. In other words,the animation for the B parameter is much shorter.



You may notice that the Time component is percent-type only. If you want to animate color or bump, you can use a Composite component. Just set the ***Color A*** parameter to your first color (or color combination) and ***Color B*** to your second one. Then link Composite's ***Mask*** parameter to your Time component. If you want to animate between multiple colors, use the Mask Gradient, with Time linked to the ***Mask*** parameter. One final alternative for animating color is to use the Hue Rotate component with Time linked via the *Amount* parameter. This will give you a smooth animation of all color hues. *Figure 3.31* is an example of a color animation linkup.

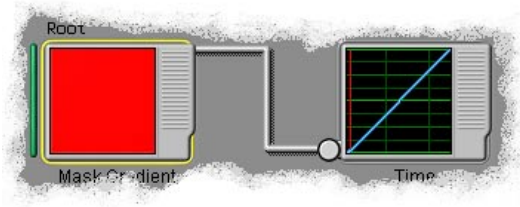


Figure 3.31

A simple example of how to animate a color. Mask Gradient has five possible colors (each of which could be linked to a color DarkTree), and it is linked through the ***Mask*** *parameter. During the course of the animation,Time will cycle through all five colors (or links).*

3.5.2 Non-Linear Animations

By default Time components are linear. In other words, they start at *0.0%* for the first frame and proceed in a straight line to *100.0%* for the final frame, with no variations. If you wish to create a non-linear animation, edit the Time component's *Spline Editor*. You'll find the *Spline Editor* within Time's *Component Editor*. For more in-depth information on these editors, refer to "*The Spline Editor" on page 92*. Another non-linear alternative is to link the Time component to a generator, which gives you an animation that matches a function curve. To do this, the generator's ***Input*** parameter must be linked to a linear (default) Time component. The generator plot will turn blue to indicate that it is now a time-based function and will generate a single value for each frame of the animation. Note that the generator's function curve will be compressed or stretched to fit with the linear time range of the Time component.

You'll find that some generators are particularly useful for animations. The S Curve generator is great for smooth ease-in/ease-out transitions. The Noise and Fractal Noise generators are quite useful for jittery animation work (check out *Figure 3.32*). The Sine Wave generator is all-around excellent for any kind of smooth waving, in and out. Saw Tooth is great for making an action repeat over and over at a linear rate. And finally, the Square Wave generator is very good for generating pulses, like blinking lights.

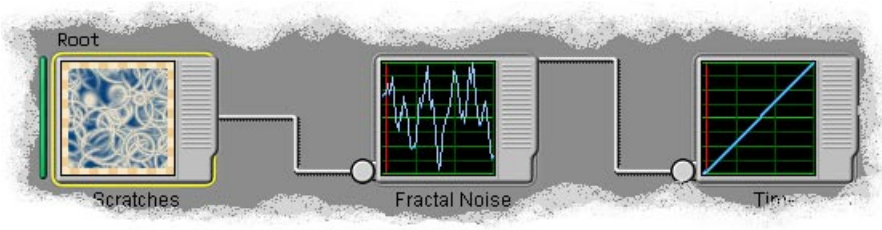


Figure 3.32

Here's a rough example of a jittery animation. Animating the Curviness parameter of the Scratches pattern gives us a clump of wriggling nematodes.

When you link a Time component to a generator, make sure that the Time output starts at *0.0%* and ends at *100.0%*, without variations in-between. If you set a smaller percent value range, the full generator curve will not be reproduced in the time domain. And finally, if you use a Time component's *Spline Editor* to create a non-linear curve, then add a link from some generator's ***Input*** parameter, you will be mixing two curves together. The results can be very confusing! As an example of what you might end up with, check out *Figure 3.33*. Try to avoid doing this.

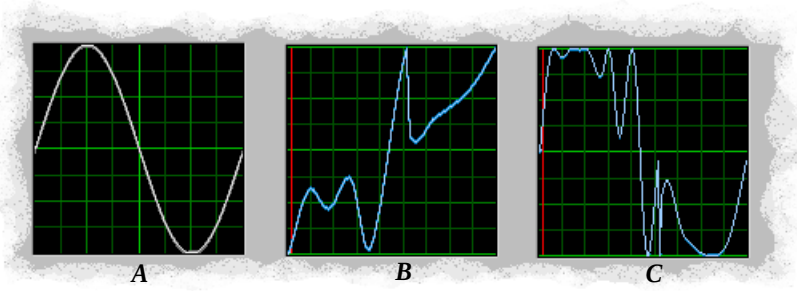


Figure 3.33

Here's an example of the type of curve (C) you could end up with if you combine a generator function (A), and a fancy curve you created with the Time component's Spline Editor (B).

3.5.3 Seamless Looping

Overview

For many applications, most notably web graphics and computer game effects, it's useful to create seamlessly looping animations. DarkTree Textures provides several methods as well as "loop" components that let you quickly set up a looping animation. Here, we'll discuss a couple of methods that you can use, alone or in any combination, to make both seamlessly looping texture surfaces and effects.

To create a seamlessly looping animation, all you need is to smoothly change your texture from one state to another and back again, over a given period of time or frame range. Basically, just make your first frame match your last frame. Three common methods of changing a procedural texture's state over time are through a Transformation component, parameter cycling, or phase rotation.

NOTE: When making a looping animation, the easiest way to design it is by making the first and last frames identical. When you render out the animation, make sure you don't render the last frame (or else toss it), since it's the same as the first frame. For example, if you want a thirty frame looping animation, set it up as a thirty-one frame animation and don't render the thirty-first frame. If you forget and render that last frame, you'll get a very obvious pause or hiccup in your animation.

Looping with a Transformation

Probably the most useful method of animating a DarkTree is to transform the sampling plane (the plane you see) through the 3D volume of texture space. By translating through texture space in the Z-direction, you'll get a boiling motion from the texture, without any apparent directional motion. The DarkTree Editor Tutorial Three, on *page 121* is an example of this approach. Translating in the X- and Y-directions will make the texture slide either left or right, or up or down. Since getting transformations to loop seamlessly is a little tricky, DarkTree Textures has added three looping components, all found in the Tile folder, that make the task easy.

The following paragraphs give you a brief rundown on each of those looping components.

Blend Loop

This component works by blending two offset layers of a texture together as they translate. Blend Loop supplies both offset layers, and they're linked from its ***Background*** parameter. Each layer fades as it translates, so that it can be reset to its

original location without being noticed. By setting values for the ***X Amount***, ***Y Amount*** and ***Z Amount*** parameters, you control how far in each direction the texture is being translated. Note that you must hook the ***Time*** parameter to a Time component to get the texture to animate. See the next figure.

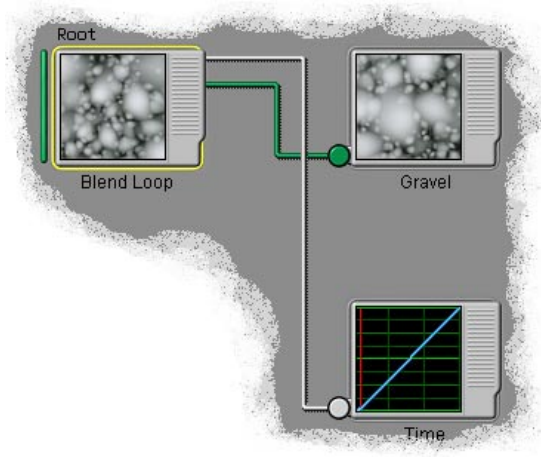


Figure 3.34
A simple example of a Blend Loop animation.

Warp Loop

This component works by translating in a waving motion through the texture space that's linked via the ***Background*** parameter. Warp Loop smoothly waves from the origin to the location specified by the ***X Amount***, ***Y Amount*** and ***Z Amount*** parameters. As the sampling plane moves back and forth through texture space, you'll see an obvious deceleration and then acceleration as the plane changes directions. This effect can be broken up so that it cannot be seen, by hooking a smooth percent texture to Warp Loop's ***Distortion*** parameter. In this context the word "smooth" means a texture that doesn't have sharp changes. Note that you must link Warp Loop's ***Time*** parameter to a Time component to get the texture to animate. Look at *Figure 3.35* for an example of this type of animation.

Carousel Loop

This component takes a cylindrical slice of the component texture that's linked from the ***Background*** parameter. It then slides that slice across the viewing plane from right to left. You can also get Carousel Loop to tile seamlessly in the X-direction by setting the ***Speed*** parameter to *0.0%*.. As with the other looping tile components, you must hook this one's ***Time*** parameter to a Time component to start the texture animating.

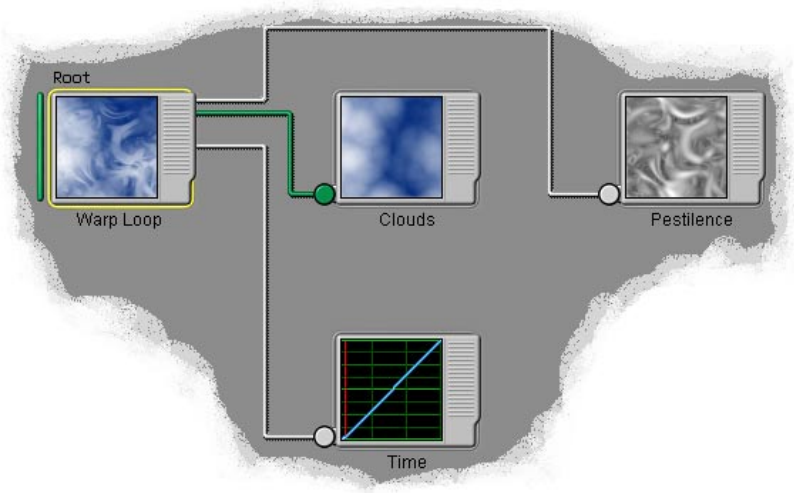


Figure 3.35

An example of a link configuration for Warp Loop. The Pestilence component represents the "smooth" link, but this link could just as easily be a complete percent subtree.

Rotate

One final component that you can use to make a seamlessly looping animation is the Rotate component, from the Transform folder. To use Rotate for a seamlessly looping animation, all you have to do is make sure you specify full *360* degree rotations. The rotation amounts are normalized for Rotate, so that a value of *1.0* is equivalent to *360* degrees. This way any whole number will give you full rotations that will loop. Rotate is certainly not as useful as the looping tile components, but it can be perfect for vortex effects. Note that you must link the ***Percent Moved*** parameter to a Time component before Rotate will animate.

Parameter Cycling

Another method for seamless animation is to cycle parameters over time. To do this you must first choose a parameter that affects the "look" of your texture, say ***Lacunarity*** for example, then animate it smoothly from one value to another and back again. To get the animation to cycle smoothly, use a repeating Time curve or a cycling

time-based generator. A cycling generator is one whose function values are the same on both the left and right sides. When you link such a generator through its ***Input*** parameter to a Time component, it becomes a time-based generator that will loop.

The best and smoothest of the cycling generators is Sine Wave. Sine Wave gently oscillates back and forth between its high and low values. It will continue to cycle as long as its ***Frequency*** parameter is a whole number. The Bell generator is another good repeating function that generates a single high spike. Both the Noise and Fractal Noise generators will cycle as well, provided their respective ***Repeat Flag*** parameters are enabled (checked). The remaining generators can be made to cycle, but they're generally too rough to look good in most animations. *Figure 3.36* is a simple setup of this type of animation.

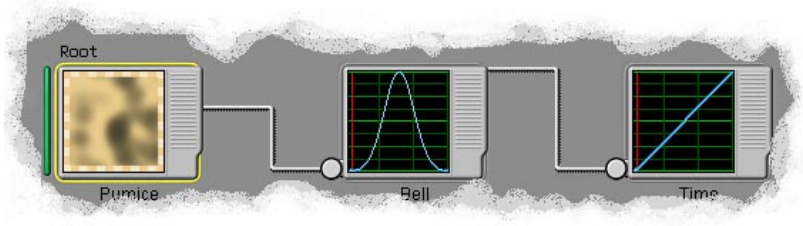


Figure 3.36
Here is a simple example of a parameter cycling DarkTree.

NOTE: Cyclic variations, such as those produced by the Sine Wave generator, can look rather obvious in the way they wave back and forth. You can break this pattern up by linking the generator's ***Phase Amount*** *parameter to a smooth noisy texture. This will cause the cycling rate for any one point in the texture to be out of phase, obscuring the regular cyclic "look".*

Phase Rotation

A last method for creating looping animations is to animate the ***Phase Amount*** parameter of a generator that has been linked to some component's ***Blend Function*** or ***Strata Function***. Both of these parameters can change the "look" of a Noise or Natural class component significantly. By animating a generator's ***Phase Amount*** parameter

that's linked from a ***Blend Function*** or a ***Strata Function***, the texture pattern will animate smoothly with no apparent direction. This method is quite useful for force field type effects.

NOTE: For the Phase Rotation method to work properly, your generator must repeat when ***Phase Amount*** *is animated. To insure that this happens, make sure that the* ***Frequency***, ***Phase Min*** *and* ***Phase Max*** *parameters are all whole numbers (1, 2, 3, 4,...). For some generators, you'll have to enable a* ***Repeat Flag*** *check box as well.*

Section 3.6
Working With Generators

3.6.1 Generator Basics

Overview

Generators are a unique and useful class of components. Even though there are many different generators, they all have some basic similarities. For instance, most have a subset of the same parameters, and generators are all mathematical functions. Their similarities make it easier to learn how to use them. This section covers all aspects of working with generators.

Before getting into the controls used to change a generator's appearance, let's take a close look at one to see what it looks like and how it works. Finally let's go over some of the terminology that we'll employ for this discussion.

Generator vs. Function

Making a distinction between the term "generator" and the term "function" is important. Generator is the name of a class of components. But we use generator when referring to a component that's a generator as well. A function is the mathematical formula that is, for all practical purposes, inside a generator. We'll use this term when we want to focus on the values within a generator. It's a slight distinction but it will be helpful later on. For example, the function inside a generator can take on any value at all, but the generator always clamps its values between *0.0%* and *100.0%*.

A Generator Preview

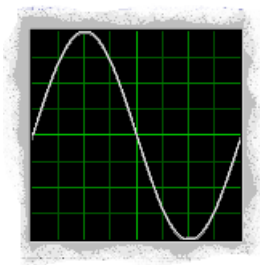


Figure 3.37
A Sine Wave generator plot.

Generators are unique because they're functions instead of 2D or 3D images. Because they're functions, generators are displayed in the *plot windows* as graphs, not images. Let's identify the different parts of a graph, using a Sine Wave generator for our example.*Figure 3.37* shows a default Sine Wave generator plot.

In the *Generator Editors* each graph has: a vertical axis, a horizontal axis, a green grid on a black background, and a white line tracing the path of the function. We'll look at each part in detail.

The green grid is a visual aide that makes adjusting the plot easier. It's much simpler to determine the value of a generator at a particular point and to see small changes when grid lines are present.

The white line plots what the generator function looks like. For each input value, the line shows the generator's output value. As you follow the line from left to right, you can see how the plot changes as the input value increases. The meaning of the generator shape depends on how it's being used. It might be the shape for a bevel, or the width of the mortar between some bricks.

Although the axes aren't explicitly labeled, the vertical axis is the *output axis*. It shows the generator's output values. The axis itself is located along the left most edge of the graph. The plot uses the vertical distance of the white line to determine the generator's value. Wherever the function line appears at the very bottom of a plot, the value is *0.0%*. And wherever it's at the top, the value is *100.0%*. Generators never have values that are less than *0.0%* or greater than *100.0%*. The values of the function are clamped so that they always lie inside this range.

The horizontal axis is the *input axis*, and is used to represent the generator's input values. The axis itself is located along the bottom edge of the graph. The input value begins with *0.0%* on the left, and ends with *100.0%* on the right. The graph shows what the generator looks like as the input value smoothly increases from *0.0%* to *100.0%*.

NOTE: Generator Editor graphs (plot windows) are always read from left to right. So for the Sine Wave generator, the graph starts at 50.0%, increases to 100.0%, goes down to 0.0%, and then completes one cycle at 50.0% again.

But where do the input values come from? Well, they can come from one of two places. If you link the ***Input*** parameter of a generator to another component (or another generator), the input values will come from the linked component. Linking the ***Input*** parameter to other generators is useful for combining more than one (this is covered later). If the ***Input*** parameter isn't linked, then the generator looks up the tree to find a parameter that supports functions. The first linked component parameter it finds that supports functions will supply input values to the generator.

Repeating Functions and Input Value Ranges

The *plot window* shows that part of a function whose input value ranges from *0.0%* to *100.0%*. This is the part of the function that makes up the generator. In most cases it is exactly the range you'll need. So as far as you're concerned, the graph shows the entire function. But it's important to realize that the function is defined outside this region as well. When you adjust the ***Phase Min, Phase Max*** and ***Phase Amount*** parameters, you're sliding a different region of the function into the *plot window.* Also,

some component parameters that support generators can supply input values that lie outside the *0.0%* to *100.0%* range. A couple of examples are the ***Center*** and ***Width*** parameters of the Stripes component.

Let's discuss how functions behave outside the *0.0%* to *100.0%* range so you'll have a better idea of how the generator responds in these situations. Some functions are cyclical, which means they repeat the same pattern over and over. An example of a cyclical function is the Sine Wave generator. The repeating pattern is called a cycle of the function. Cyclical functions repeat the same pattern outside the *0.0%* to *100.0%* range.

Other functions aren't cyclical and, by default, do not repeat the same pattern over and over. Such functions, Bell for instance, have a ***Repeat Flag*** check box allows the function to cycle multiple times. With the flag disabled (not checked - the default), the value at *0.0%* will be held for any input value less than *0.0*. The value of the function at *100.0%* will be held for any input value greater than *0.0*. When the flag is enabled, the function behaves like a cyclical function - repeating the same pattern over and over. You won't see the repeats in the *plot window* nor will you be able to use them effectively with the majority of the parameters you might link, until you change either the phase parameters (***Phase Max***, ***Phase Min*** and ***Phase Amount***) or the ***Frequency*** parameter. At that point, the cycles can be very useful.

The Noise generator is not cyclical, but doesn't repeat in the manner described above either. When ***Repeat Flag*** is not checked, Noise generates values outside the *0.0%* to *100.0%* range, but the patterns never repeat. It'll simply generate more noise values similar to those in the graph window. When ***Repeat Flag*** is checked, the value at *0.0%* will be the same as the value at *100.0%*. This is required for some functional parameters, because in certain cases, when a functional parameter defines the outside edge of a pattern there will be a big discontinuity, if the values are different.

NOTE: The Add, Subtract, and Multiply generators are simple operations and not considered to be either cyclical or non-cyclical.

3.6.2 Generator Controls

All generators, other than those that are simple math operations, such as Add, Multiply, and Subtract, share a common set of controlling parameters. These parameters are, in some cases, the only ones controlling the generator. In the remaining cases, these same parameters still make up the majority of the controls. In this section we'll describe the controlling parameters and explain them in detail. They all behave identically in all generators.

The Amplitude and Shift Parameters

The ***Amplitude*** and ***Shift*** parameters work together to define the range of output values for a generator. The value for ***Shift*** defines the center or horizontal axis of the function. The value given ***Amplitude*** defines the amount the function will change relative to the center axis. *Figure 3.38* shows visually how each of these parameters affects a generator. The left most generator in the figure is the reference. The middle figure shows a decrease in ***Amplitude***, and the right most figure shows a added decrease in ***Shift*** on the middle Sine Wave.

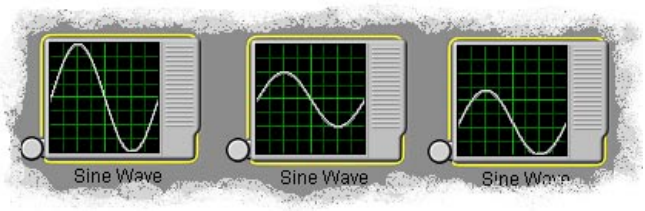


Figure 3.38

Three sine wave plots. The left most plot shows the default, the middle one has a decreased Amplitude and the right most has a lesser value for Shift.

To determine the *maximum* value of a function, add the ***Amplitude*** to the ***Shift***. To determine a function's *minimum* value, subtract the ***Amplitude*** from the ***Shift.*** (Remember, you must clamp these values at *0.0* and *100.0* for the generator.) If you know what you want the maximum and minimum values to be for a generator, it's straightforward to determine the ***Amplitude*** and ***Shift***. ***Shift*** will be half of the sum of the maximum and minimum (max + min divided by 2). ***Amplitude*** will be half of the difference between the maximum and minimum (max – min divided by 2). It's worth noting that if ***Shift*** and ***Amplitude*** are the same value, then the minimum value of the generator will be *0.0*, and the maximum value will be twice the value for ***Amplitude***. Most generators default to an ***Amplitude*** of *50.0%* and a ***Shift*** of *50.0%*, which allows both to vary from *0.0%* to *100.0%*.

The Frequency Parameter

The ***Frequency*** parameter controls how quickly a function changes. The higher the value given to ***Frequency***, the faster the function changes. This is similar to scaling a 2D or 3D texture. *Figure 3.40* shows how the frequency affects the ubiquitous Sine Wave generator. Note that the generator on the right has a higher frequency than the one on the left.

You can use most generators to make a DarkTree cycle over time, although not all generators will do so smoothly. An example might be making a Sun expand and contract over a specified number of frames. For cyclical generators, setting the

parameter to an integer, such as *1.0, 2.0, 3.0* etc., will cause the generator's first and last values to be equal. That's important if you want the generator to cycle or loop evenly. For the non-cyclical generators, with the exception of Noise, make sure ***Repeat Flag*** is enabled and the ***Frequency*** is given some multiple of two. If you don't give ***Frequency*** a value that's a multiple of two, the first and last values won't be equal. For the Noise generator, simply enable the ***Repeat Flag***.

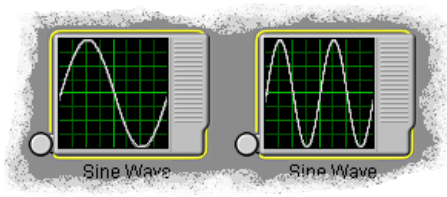


Figure 3.40
A default sine wave plot on the left shows an increased value for Frequency, on the right.

"Repeating Functions and Input Value Ranges" on page 198 defines cyclical and non-cyclical generators.

Phase Max, Phase Min and Phase Amount

The phase parameters move the function from side to side. ***Phase Min*** represents the lower phase limit, and ***Phase Max*** represents the upper phase limit. The two define a phase range. ***Phase Amount*** represents a point between the two. *Figure 3.42* shows how these parameter values affect a generator. The component on the left has no phase shift, while the one the right has a positive shift.

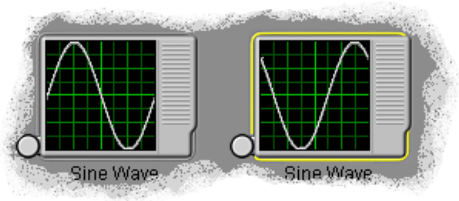


Figure 3.42
On the left is a default sine wave plot. The plot on the right has a positive value added for the Phase Amount parameter.

It's possible to shift the function the entire width of the *plot window* by changing the ***Phase Amount*** parameter. And for animations (for instance), you can set the number of cycles or parts of a cycle by subtracting ***Phase Min*** from ***Phase Max.*** For example, if you set ***Phase Max*** to *10.0* and ***Phase Min*** to *5.0,* then as ***Phase Amount*** moves from *0.0%* to *100.0%* that will be five complete cycles (assuming a frequency of 1.0). Remember, a cyclical generator does not have (or need) a ***Repeat Flag*** check box.

Lower Clamp and Upper Clamp

The default clamp values for all generators are *0.0%* to *100.0%*, but you can clamp values within the default area (more than *0.0%* and/or less than *100.0%*). To make the changes, set the ***Upper Clamp*** to the maximum value you've chosen, and the ***Lower Clamp*** to your minimum value.

3.6.3 Combining and Controlling Generators

This subsection describes some useful techniques you can use when working with generators.

In most cases, you can adjust the parameters of a generator to get the function that you're looking for by simply adjusting the common parameters described in the previous section. To change a generator further, remember that you can invert it using the ***Invert*** check box. (Enabling ***Invert*** will flip the function around the X-axis.) Using the ***Root*** button under the ***Views*** section of the *Generator Editor* can be very useful when working with generators, just to keep a running view of how changes to a generator are affecting the whole DarkTree.

One of the first things you might find yourself wanting to do is to mirror a generator plot. Checking the **Invert** check box can sometimes accomplish this, but in other cases you'll have to reverse the function's direction. Luckily, with the new ***Reverse*** check box (as of version 2.0) this isn't a difficult thing to do. Enabling ***Reverse*** actually flips the function about the Y-axis. *Figure 3.43* shows the plot of a normal generator and the same generator plot inverted or/and reversed.

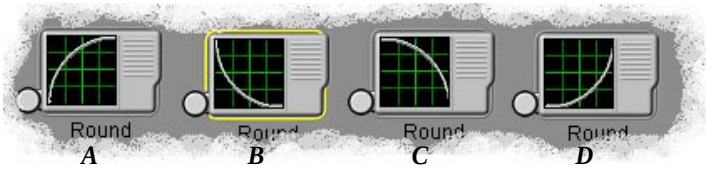


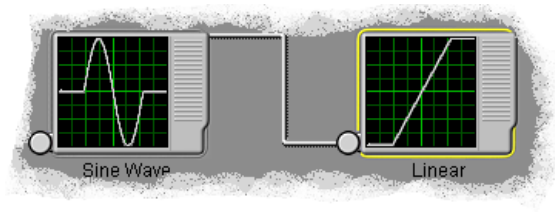
Figure 3.43

A Bias function plot shows the default (A), with Inverse enabled (B), C shows the function with Reverse enabled; while D shows both Reverse and Invert enabled.

Another useful trick is to be able to select any piece of a generator and place it anywhere you want inside the plot window. By using the ***Amplitude***, ***Shift***, ***Frequency***, ***Phase Min***, ***Phase Max*** and ***Phase Amount*** parameters, you should be able to get any part of a generator to completely fill the *plot window*. To place a part of the function in a specific location within the *plot window*, drag out a Linear generator and adjust it so that it ramps up exactly where you want the part of the other generator to begin. Now link the other generator's ***Input*** parameter to the Linear generator. *Figure 3.44* shows you an example of this arrangement.

At some point you might want to layer two generators, one on top of the other. One common method is to layer the Noise generator on top of a regular one. Use a noise generator to break up even patterns. Try Add to put the two generator layers together. The Subtract generator also works well for layering. To avoid clipping, you might have to scale one or both of the layered generators. You can use the ***Amplitude*** and ***Shift*** parameters to scale the generator down.

Figure 3.44
This plot shows an example of using the Linear generator to cause Sine Wave to ramp up in exactly the place on the plot that you choose.



*NOTE: You don't have to use a **Linear** generator to control the blending, although it works well.*

Section 3.7
Notes on the Simbionts

3.7.1 General Information on Simbionts

What are the Simbionts?

The Simbionts are a set of independent stand-alone plugins, each designed and written for a specific commercial modeling/rendering package. Using a Simbiont, you can import DarkTrees as procedural shaders directly into your rendering software. You'll find a discussion on the many advantages of importing your textures in this way on page *127*, under the heading *"Procedurally-Generated Texture Maps VS True Procedural Surfaces"*.

We try to select modeling applications that are among the more heavily used for our plugins, just so that you, our customers, can get the most mileage from the DarkTrees you have designed.

Simbiont Capabilities

- *Because of the new Tweaks tool, you can change many of a DarkTree's attributes from within your rendering package. When you design a DarkTree, simply create a Tweak for any of the parameters that you might want to adjust later. You can create a Tweak for any of the eight parameter types supported in the DarkTree components. One of the Simbionts' greatest advantages lies in the fact that, given a number of tweaked parameters, you can completely change a texture's total "look", whether the original is your own or from the DarkTree Simbiont texture set.*
- *With some of the Simbionts, you can animate a previously static DarkTree. Using SimbiontMAX as an example, you can animate a texture using 3d studio max's Track View capabilites.*
- *You can use the camera distance to change a DarkTree. For example, you can make the texture fade into the distance, or conversely create microscopic details for a closeup.*
- *If the modeling software supports it, you can design a texture so that it's sized automatically to fit a model.*

- *The Simbionts support network rendering. That is, they don't require an additional license for rendering on each render node or machine.*

3.7.2 Designing DarkTrees for the Simbionts

How to Design Map Textures

Map textures are single-type DarkTrees that can be automatically converted among color, percent and bump types. When designing map textures in DarkTree, there are a few specific rules you should follow. The first is that the tree should be a color-type; that is, a color component should be in the root socket. Because map textures need to be automatically converted between types, their *Tweaks* must be set up in a certain way. Here's how.

For a start, tweak the common parameters as you normally would. Color parameters, however, must be treated specially. Instead of tweaking them directly, use a Color Convert (Process class) component to do the job. To tweak a color parameter, begin by linking it to a Color Convert component. Next, tweak the Color Convert's ***Background*** parameter.This allows the color texture to be converted to percent- and bump-types, while keeping the color *Tweak* controls connected to, and able to control, the converted versions. For conversion to percent and bump, Color Convert internally uses the color value (of HSV). Figure 3.45, below, is an example of a correctly tweaked map texture.

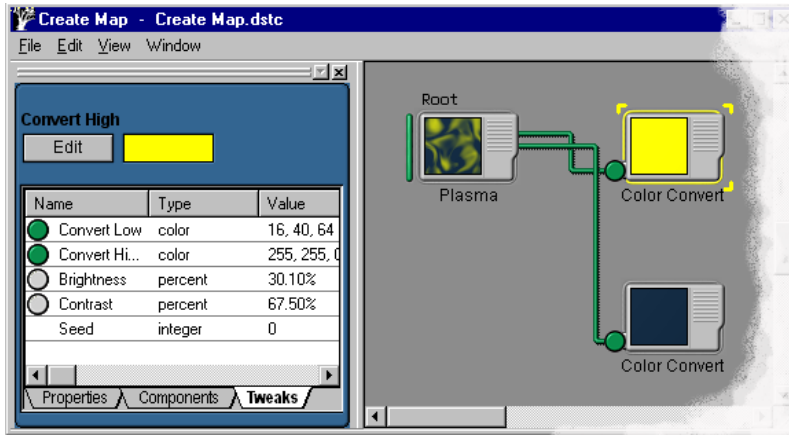


Figure 3.45
An example of a correctly tweaked map texture that uses a Color Convert component for each linked color.

Surface Distance

Setup is critical for getting good results from textures that vary based on the surface's distance from the camera. That is, DarkTrees that use the Surface Distance component. With a complex texture, setting the ***Near-*** and ***Far Threshold*** parameters by eye sometimes is very difficult. As an aide to setup, try using a *calibration* texture to get good final distance values. Figure 3.46 is a nice example of a recommended calibration texture. It's simply a Surface Distance component with its ***Blend Function*** parameter linked to a Gain generator that has *its* ***Gain*** parameter set to 100%. By using a calibration texture, you can set up distinct bands of color. Using bright colors, of course, helps as well because you can easily see what's happening. Note that the texture in Figure 3.46 has three distinct regions.(Even though the color doesn't show in the figure, you can see the color value.) The darkest area defines points closer than the ***Near Threshold***, the medium area is for points between the ***Near-*** and ***Far Threshold*** parameters, and the lightest area is for points beyond the ***Far Threshold***. Use this texture to set up these parameter values, then switch to your final distance-based material.

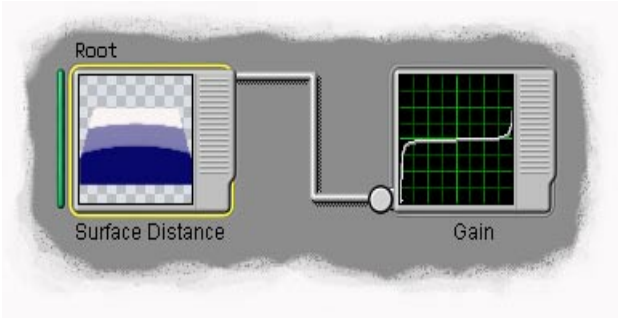


Figure 3.46
An example of a recommended calibration texture.

NOTE: In order to make all three color values visible in the Surface Distance component preview for Figure 3.46, we had to change the Near- and Far Distance parameters from their 0% and 100% defaults, respectively. It really isn't necessary for you to do that, since the texture will work correctly as it is. It just isn't as visible in DarkTree because of the small viewing area.

3.7.3 In Conclusion

The DarkTree Simbont collection includes SimbiontLW for LightWave, SimbtiontMAX for 3d studio max, SimbiontTS for trueSpace and SimbiontAM for Animation:Master. The two most recent Darkling Simulations plugins are: SimbiontC4D for Cinema 4D and SimbiontRM for Renderman.

Beginning with late Summer 2002, you can download any or all Simbionts free of charge from our web site at **www.darksim.com**.

Part 4
DarkTree 2.5 Addendum

Section 4.1

4.1.1 Introduction

The DarkTree 2.5 release adds tools for those programmers who would like to write their own components and plugins for DarkTree. Therefore, updates and additions are geared toward this purpose, with just a few cosmetic changes and some additions to the program. The following paragraphs cover these changes.

4.1.2 New Menu Bar Options

The Scripts Utility for Python

Python is a great, easy-to-use scripting language that you programmers among the DarkTree community can use to write scripts for managing your DarkTree files in a variety of ways. Python is available without charge and downloadable (as of the Summer 2003) from either www.python.org or www.activestate.com.

The ***Scripts*** utility for Python is accessable from both the menu bar on the DarkTree user interface (***Tools*** > ***Scripts*** > ***list-of-Python-scripts***) and from the DarkTree Editor menu bar (also under ***Tools***). You must first choose a DarkTree for a script to act upon by either selecting one from the DT Library list or by loading one into a DT Editor. Make sure to save the most current version of your DarkTree first. Otherwise you'll get a warning message and won't be able to activate the script.

You'll find the Python script files in the "Scripts" folder of your DarkTree installation. DarkTree calls scripts with the DarkTree file name as the first argument. If you prefer that some script names do not appear on the ***Scripts*** menu, simply begin those file names with an underscore ("_").

DarkTree 2.5 ships with two Python scripts: ***Split_DT*** and ***Export_Renderman***.

- ***Split_DT*** splits a DarkTree shader into separate single-type DT files. The number of files you'll end up with is dependent upon the number of subtree links the Shader component has. The split-out files will be put into the same directory as the shader you began with. All split-out DT files will have the shader file name but end with the specific type and channel delineation added, (i.e., *name_color.dstc* for color).
- ***Export_Renderman*** generates a Renderman shader from a DarkTree file and a shader template. Please download SimbiontRM from our website, www.darksim.com, for more detailed information.

The Event Log

The menu bar on the DarkTree main interface now includes an internal event log (***Tools*** **>** ***Event Log***) that you can enable and set to one of four levels. Enabling the ***Event Log*** can give you an idea of what's happening with DarkTree internally. The most important reason that we've added the ***Event Log*** is to give new component and plugin programmers a way to track down problems between their code and DarkTree.

From least important to most important, the four levels of warning are:

- ***Information*** - *This event level covers general information of interest.*
- ***Warning*** - *A warning event is usually something unexpected but not fatal.*
- ***Error*** - *Errors are more severe problems but not those that are likely to cause a DarkTree crash.*
- ***Fatal*** - *Fatal errors indicate a serious enough problem to probably crash DarkTree*

Regardless of the event level you choose, you'll get messages about higher level events as well. In other words, if you set the ***Event Log*** to ***Warning***, you'll get all ***Error*** and ***Fatal*** information too. Since an ***Event Log*** popup will appear each time DarkTree generates a new message, you might want to set the log to ***Fatal*** or disable it entirely until you really need it for debugging. While enabled, DarkTree keeps a complete list of events.

NOTE: Since DarkTree can detect fatal errors, you may wonder why a program crash cannot be avoided. Actually it can, but fatal errors are almost exclusively caused by some problem with a plugin component.

To abort a crash, additional checks for every component would have to be put in place. This would slow down rendering significantly.

Component Paths

Component Paths, found on the main interface menu bar (***Tools*** **>** ***Component Paths***), is another useful utility for anyone writing new DT components. Directory paths are the way DarkTree locates the new components you have written. Figure 4.1 is an image of the ***Component Paths*** dialog. Note the four buttons along the top of the window. Click the leftmost

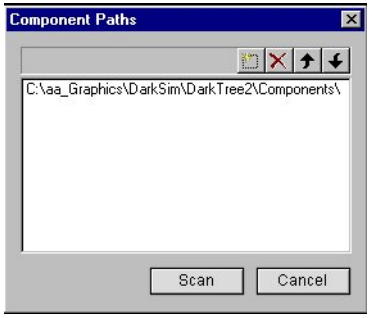


Figure 4.1
An image of the Component Paths dialog.

(*New*) button to browse and add new directory paths. The second button from the left deletes the currently selected path. The two right most buttons move the currently selected path up or down on the list, giving you a chance to rearrange path importance. After you've added one or more new paths, click the ***Scan*** button so that DarkTree can scan them for plugin components and duplicates. You'll only need to do this once. DarkTree 2.5 will list any repeat components it finds and also tell you which version it will use. You may add any number of new paths.

When you first click on ***Component Paths*** , one path should already be listed. This is the directory path to all standard components that ship with DarkTree 2.5, except the Shader component (which is embedded).

4.1.3 Additions & Changes to the Component Editors

The Help Button

The blue **Help** button has been moved to a more visible location. Clicking on it still brings up a description of the component currently loaded into the *Component Editor.*

The Update Window

The *update window*, found on all *Component Editors*, is now larger for easier viewing. As was always true, the *update window* represents *x* square units of texture (with *x* equaling some number, usually *1*). The units (inches, meters and so forth) are the same as with version 2.0 and can be set on *Properties* along with a number for the units. The *Properties* discussion begins on page 53.

The Views Scaling Tool

The *update window* discussion leads to a useful new feature on the *Component Editors*. Without actually scaling a texture, you can use the new ***Views*** *scaling tool* to see what the texture in the *update window* would look like over a larger (or smaller) area. For example, suppose you have set your texture size to 3.0 meters. You would like to see how it will look over a wider area, say on a 7.5 meter cube. Just use the zip slider and scale up, to 7.5 meters. Now what you see in the *update window* is your texture over a larger area. To reset the scale again (to 3.0 meters in this case), just start scaling back down. The zip slider will "stick" at the original number of units set earlier in *Properites.*

Figure 4.2 shows a texture scaled to 7.5 meters.

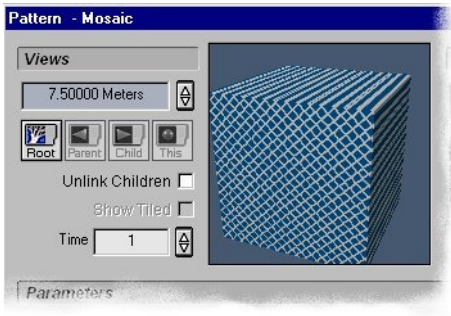


Figure 4.2
This is image from the Views section of the Component Editor shows the new scaling tool.

NOTE: Some modeling / rendering packages keep the texture at the scale DarkTree provides. Others do not. You can preserve the scale set in DarkTree with our plugins for 3d studio max, LightWave and trueSpace.

Glossary

8-Bit Color: An image format that assigns 8 bits of color data per pixel which works out to two hundred fifty-six colors.

24-Bit Color: An image format that assigns 24 bits of color data per pixel which works out to sixteen million colors.

Algorithm: A mathematical recipe or procedure.

Algorithmic Texture: A mathematical recipe or procedure that tells the computer how to create the colors, gray scale values, and bumps for a textured surface.

Aliasing Artifacts: Texture areas of jagged silhouette edges caused by too much detail for the number of pixels on the display device. This can be particularly troublesome in animated sequences, because the jagged areas appear to "crawl".

Alpha Channel: A channel that is traditionally reserved for masks.

Anisotropic Effects: An directional effect that works in conjunction with certain attributes. The DarkTree Shader has two such channels: one that controls a directed specular highlight and the other that is used to specify a brushed bump effect.

Antialiasing: A process used to increase the quality of a rendered image by taking multiple samples for each pixel, and combining them into a single value.

Aspect Ratio: A ratio of width to height. For example, a width of 100 dots and a height of 50 dots has an aspect ratio of 2 to 1.

AVI: A movie format that is included with the Microsoft© Windows operating system.

Background Display: A checkerboard pattern that is visible beneath the DarkTree displays on the Texture Library and component images. The background is changeable from Global Preferences.

Bitmap: A computer representation of an image or picture defined by individual pixels, which are arranged in rows and columns.

Bitmap Renderer, Renderer: One of the four major pieces that make up the DarkTree Textures software interface.

Bump: One of the three key or basic types for DarkTrees, single components and parameters. Bump-types are visually identified by a blue *type icon.* This texture type is used to make bump maps, elevation maps, or displacement maps.

Bump Map: The result of saving a bump texture, and the same as an elevation map.

Channel: A single surface attribute (luminosity, transparency, surface bump) that may be defined by a DarkTree or subtree. The Examine Window loads a default shader; but one or any number of single channels can replace one (or more) of the shader channels.

Check Box: A small toggle that has two options: on or off. When you see a check mark in the box, the option is enabled (on). Use the left mouse button to disable it.

Child/Parent Relationship: The relationship between two linked components. The parent has a parameter linked to its child. In the DarkTree Editor, the parent is to the left of the child.

Class, Classification: A DarkTree Textures component grouping. Components are classified based on how they function and what kind of results they can produce.

Clear Coat Channels: Four channels in the DarkTree Shader component. They all have to do with the special effects arising from a heavy coating of something akin to clear varnish.

Color: One of the three key or basic types for DarkTrees, single components and parameters. Color-types are visually identified by a green *type icon.* This texture type is used to make color texture maps.

Color-Coding: This occurs on *type icons* and wire links. The colors and designations are: green for color, gray for percent and blue for bump. The purple-colored type icons are used strictly for shaded DarkTrees and just one component: the DarkTree Shader.

Component Face: The visible surface of a component and consisting of several parts.

Component, Texture Component: Used interchangeably. A flexible object-oriented structure that contains an algorithmic texture.

Component Menu: A menu that is accessed by a single right-click on any component. *Component menu*s are tailored to the component from which they are opened.

DarkTree Editor, Editor: One of the four major pieces that make up the DarkTree Textures software interface. The place where DarkTrees are edited.

DarkTree Shader: A special DarkTree component type that includes eighteen linkable channels. The links are still made from the three basic DarkTree types: color, percent and bump. A purple-colored icon is attached to the shader type.

DarkTrees, Trees: A collection of components connected to each other with wire links into a conceptual tree shape. The texture generating structures in DarkTree Textures.

Dialog Box, Dialog Window: A common program window that allows you to select a variety of options. A dialog box appears on the screen in response to some user action and remains until you choose one of its options, which closes it.

Diffuse Level: The degree that light is absorbed or reflected by an object's surface.

Direct Link Selection: A linking shortcut whereby you can click on a link to select it. You can then drag the end of the link to another component, or see which parameter the link begins with.

Drag And Drop: A common user interface operation whereby you can drag an object to a different place on the screen and drop it there.

Drop Down List: A common user interface control that allows you to click on an arrow and access a list of options. At that point you can select one of the list members.

Edit Control Panel: Part of the DarkTree Editor, this detachable panel has three tabbed pages: the *Component Bin*, *Tweaks Editor* and *Properties page*

Environment map: An image used as the reflection for a reflective surface, in place of a real environment.

Examine Window: One of the four major pieces that make up the DarkTree Textures interface. The Examine Window allows larger and shaded views of DarkTrees. New with version 2.0, the Examine Window has an expanded number of controls, such as the *rough play* control. for a quick view of your animations.

File Extension: An ending for a file that gives the format or type of the file. For example, *bmp* is a common file format. See Appendix A for a list of the output file formats that DarkTree supports.

File Icons: *Type icons* placed before each file name in the Texture Library. They indicate the DarkTree type: color, percent, bump or shaded. There's also an icon for corrupted files.

Floating Point: A decimal or real number made up of a whole part, a decimal point, and a fractional part.

Glossiness: The glossiness attribute controls how concentrated or spread out surface hot spots will be. Used in conjunction with the specular level.

Gradient: A smooth transition from one value to another. A DarkTree Textures component class.

Grayscale: Black, white and all graduated levels of gray. Percent-type components make grayscale images.

Grid: The rows and columns of sockets on the Editor *workspace*.

Heading: Rotating a 3D object to the left or right changes its heading.

HSV: One method of defining a color space where each color is defined by a hue, a saturation level, and a value level.

Hue: The basic color in the HSV color scheme.

Image: In DarkTree Textures, the visual representation of a DarkTree, or an external image. See Bitmap.

Interpolation: A mathematical blending between two values or levels.

Levels, Tree Levels: The columns of sockets on the DarkTree Editor *workspace.* Level one is the **Root** socket, level two is the next column to the right, and so on.

Library Groups: Top level directories in the Texture Library. Library Groups come from the system directories.

Library List: The list of folders and DarkTree files in the Texture Library.

Linking: In the DarkTree Editor, a process for connecting a component parameter to a component of the same type.

Luminosity: The steady, suffused and self-lit state of a 3D object's surface.

Mapping Modes: Projection shapes that define how a texture is applied to a 3D object. DarkTree Textures supports four modes: planar, spherical, cubic and cylindrical.

Metal Highlight: A DarkTree Shader channel that controls metallic effects.

Online Component Reference: The Component Reference is now completely online. To access information about a specific component, click on the blue-colored ***Help*** that you'll find on most *Component Editors*.

Percent: One of the three key or basic types for DarkTrees, single components and parameters. Percent-types are visually identified by a gray *type icon.* Percent-types are used to create diffuse, specular, luminosity, reflectivity, alpha masks, etc.

Pitch: Rotating a 3D object up or down changes its pitch.

Pixel: A picture element or single point on a screen or image.

Pixilated: To be led astray, as though by pixies; confused, bewildered and/or intoxicated.**:-)**

Preview Window: A small window that is part of the Texture Library. It displays the image of the currently selected DarkTree.

Progress Bar: A graphical display of some program action's progress. DarkTree Textures uses blue meters for its progress bars.

QuickTime: A popular movie format from Apple Computer, Inc.

Raw Channel: An unshaded render of a single channel, such as surface color.

Raw Texture Map: An unshaded texture map.

Raw Elevation Map: An unshaded bump map in which elevations correspond to shades of gray.

Reflectivity: The degree that a surface mirrors or reflects its surroundings.

Refraction: This DarkTree Shader channel controls the degree to which a transparent object bends light. This depends on the material it's intended to mimic.

Rendering: The process of turning a DarkTree into an image (or horses into glue).**:-)**

Resolution: The density or number of pixels in an image.

RGB: One method of defining a color space, where a red value, green value, and blue value are mixed to define a color.

Root: On an Editor *workspace*, the left most component position from which all other links branch out.

Saturation: This controls the purity of a color and goes from the strongest possible purity at one end of the scale, gradually adding more white to the color as the opposite end is reached. Part of the HSV color model.

Screen Shot, Screen Capture: A snapshot taken directly from a monitor screen.

Shaded Preview: A texture image with lighting effects applied; a shaded DarkTree.

Shaders: A subprogram that combines surface attributes and lighting to color or shade an object's surface.

Sibling: A component that shares the same parent with another component.

Specular Level, Specular Intensity: The specular level determines how intense surface hot spots from an exterior light source will be.

Simbionts: A group of plug-ins, each especially written for a particular commercial modeling/rendering package. Simbionts allow you to import DarkTrees directly, without creating bitmaps of the textures first.

Sockets: On the Editor *workspace*, the rows and columns of depressed squares that the components are plugged into.

Subroot: The topmost component of a conceptual tree branch. The root of a subtree.

Subtree: A branch of a conceptual tree. A component and all of its children, grandchildren, and so on. The type of a subtree is determined by the type of its subroot.

Surface Attributes: Properties of a surface, such as bumpiness, color, glossiness, etc.

Tabbed Sheet, Tabbed Page: A common user interface that allows you to click on one of a group of labeled tabs. Doing so will bring the sheet or page with the labeled tab to the front.

Texture Library, Library: One of the four major pieces that make up the DarkTree Textures software package. This is where DarkTrees are organized and stored.

Toggle: A small graphical area, i.e. a box, that can be clicked on or off with the left mouse button. (See check box.)

Transparency: A DarkTree Shader channel, transparency has the property of transmitting light without appreciable scattering, so that objects situated behind are completely visible.

Tree Structure: The shape or design of a conceptual tree, visible within the DarkTree Editor.

Tweaks: Macro controls for DarkTrees that allow the user to control more than one parameter with the same value. All parameters for a single Tweak must be the same type (check box, color, float, etc.).

Type: In DarkTree Textures, the three kinds of basic DarkTrees you can build: color, percent, and bump. DarkTree Textures also has a specialized type, the shader.

Type Icon: A small color-coded disk that tells the user the type of a DarkTree, component, or component parameter. The three key or basic types are: green for color, gray for percent and blue for bump. The purple-colored *type icon* is associated with the shader type.

Update Window: The small window that is part of each *Component Editor*. The *update window* can be set to show the root, parent or this image.

Value: The third element of the HSV color model. Value controls the balance between brightness and darkness.

Wire Link: A link that's drawn between a component parameter and a child component. The child component is located in a column to the right and must be of the same type. Wire links are colored by type (see Color- Coding).

Workspace: The grid-of-sockets area of a DarkTree Editor. This is where tree building takes place.

Appendix A: Valid Formats for DarkTree Textures

The following list of formats are, as of DarkTree Textures version 2.0, supported output formats for bitmaps generated by the DarkTree Bitmap Renderer. In addition, these same formats are all valid output choices for the ***Save Image*** option on the Examine Window *popup menu*. And, finally, you can load image files that use any of these formats from the Image and Image Sequence components.

Some of the formats, such as JPG 24 bit, include choices for a subfile type and a Q Factor. Q Factors (quality factors) run from *2* to *255*. An entry of *2* is the best and has the least compression. JPG, to use the same example, has four subfile types: *YUV 4:4:4* to *Progressive 4:4:4*. The subfile types and Q Factor settings are not indicated on the following list. However if you select a format that has these additional options, the appropriate dialog will appear automatically when you make a selection.

IFF - 8 bit and 24 bit

JPG - 24 bit

FPX - 24 bit

Mac PICT - 8 bit and 24 bit

PSD - 8 bit and 24 bit

CServe PNG - 8 bit, 24 bit and 32 bit

SGI - 24 bit and 32 bit

TIF - 8 bit, 24 bit and 32 bit

TGA - 8 bit, 16 bit, 24 bit and 32 bit

Win BMP - 8 bit, 16 bit and 24 bit

XPM - 16 bit, 24 bit and 32 bit

PCX - 8 bit and 24 bit

Appendix B: Setting Global Preferences

Set up global preferences on the *DarkTree Preferences* dialog. This dialog lets you change the "look" of the DarkTree Editor and the Examine Windows to better suit your own way of working. The settings on this dialog are defaults. That is, they define how newly-opened DarkTree Editors and Examine Windows appear. The options you set here will remain in effect until you change them. Access DarkTree Preferences from the *main application menu* under ***Tools>Properties***.

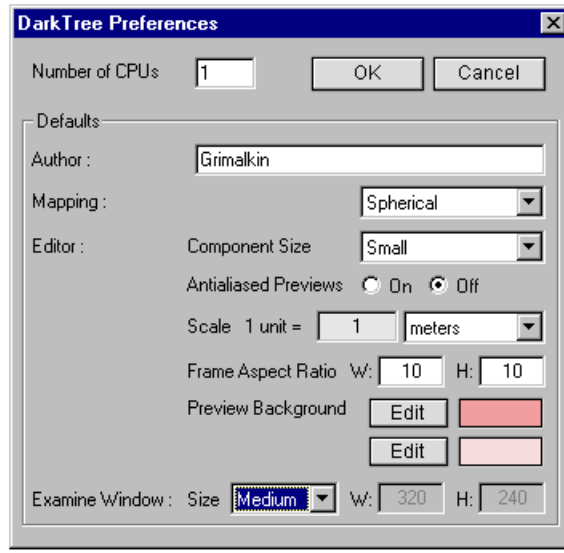


Figure 4.3
The DarkTree Preferences dialog. Access this dialog from the main application menu.

You can change many of the same options found here on the *Properties page* of the *Edit Control Panel.* However the *Properties page* settings alter a specific DarkTree only.

The following is a rundown of the options you can change on the DarkTree Preferences dialog.

Number of CPUs: The first time you run DarkTree Textures, the program will assess and set the number of CPUs that you have on your system. In the case of a multiple-CPU system, you might want to set ***Number of CPUs*** to fewer than you actually have. If so, you can key-in a number. Don't, however, set a higher number here than you can back up with actual CPUs, thinking it will improve program efficiency. The effect will be just the opposite.

Author: When you first install DarkTree Textures, you'll be asked for an owner's name. That is the name you will see here as the default. But suppose you want to author your DarkTrees under the pseudonym of "Spider". Change the **Author** to Spider, then later when you enter information on the *Properties page* about a DarkTree, you'll always see Spider in the author's box.

Mapping: Set the mapping mode based on how you like to see most of your DarkTrees displayed. For the DarkTrees that look better on another shape (mode), change the mapping on the *Properties page.* The mapping choices are: ***Frame***, ***Planar***, ***Spherical***, ***Cylindrical***, and ***Cubic***. All of these modes are 3D except ***Frame***. Use the frame selection to display tiled textures.

The following settings affect just the DarkTree Editor.

Component Size: The size you select here will resize the the components and sockets on the Editor *workspace*. Choose among ***Small***, ***Medium*** and ***Large*** from the drop down list.

NOTE: You can resize the components and sockets for a single open DarkTree Editor by making your selection from the Editor menu, under ***Views.***

Antialiased Previews: The ***On*** or ***Off*** radio buttons control whether or not your component images on the Editor *workspaces* will be antialiased. The standard rendered previews are quite good, but if you want the finest quality rendering, even while building DarkTrees, then you'll want to select ***On***. Keep in mind that if you like your component size to be large and select ***On*** as well, the component renders will take a little extra time.

NOTE: From the Editor menu you can select among ***Rough***, ***Medium*** *and* ***Antialiased*** *under the* ***Views*** *option. Only the open DarkTree Editor will be affected if you select from the Editor menu.*

Scale: This option lets you choose a scale you like to work with regularly. Select from: ***inches***, ***feet***, ***miles***, ***millimeters***, ***centimeters***, ***meters*** and ***kilometers***. Depending upon the modeling/rendering software you'll be importing your DarkTrees to, some of our Simbiont plug-ins are able to support the texture scale you set here.

Frame Aspect Ratio: This option is for frame mapping only. Enter a new aspect ratio if you work primarily with uneven sizes, with video for instance. You can change the aspect ratio for individual DarkTrees from the *Properties page*.

Preview Background: You can change the colors, but not the pattern, for the background checkerboard at any time. Clicking on each **Edit** button calls up the *DarkTree Color Browser*.

NOTE: This global preference setting is the only change that happens immediately. It will not re-render the backgrounds of components already situated on the workspace, but any new renders will have the new background colors.

The last preference applies just to the Examine Windows.

Size: This option refers to the default size for each newly-opened Examine Window. Choose among ***Small***, ***Medium***, ***Large*** and ***Custom***. If you select ***Custom***, the two edit boxes will no longer be grayed-out, and you can enter the sizes you prefer. To give you an idea of window sizes, a medium-sized Examine Window is 216 by 216 units.

NOTE: You can, of course, re-size individual Examine Windows by either selecting another window size from the Examine popup menu or by dragging an edge or corner.

Index

D

U

V

W

Z