Texture Editor

for Windows®, first public Release

User Guide V. 1.0

Contents

Chapter 1
Texture Editor Overview

Texture Editor – What the f**k is that ?

The Texture Editor is a powerful tool to generate procedural quadratic textures.
You can use them for your stunning 3D scenes, demos, games etc, wherever textures are needed.
Use the Texture Editor to create:

- color texture maps
- bump maps
- specular / diffuse maps
- reflection / refraction maps
- transparency maps
- displacement / height maps

Why to use procedural textures instead of static images ?

- Procedurals tile seamlessly
- Procedurals can be easily reused and modified
- Procedurals do not contain any lighting artifacts
- Procedurals can be generated in high resolutions (up to 1024 x 1024) pixels
- Procedurals can be used to generate many different textures of the same scheme (see illustration 1). This is useful in large scale mass renderings.
  We will provide an effective mechanism to generate texture masses in later versions.

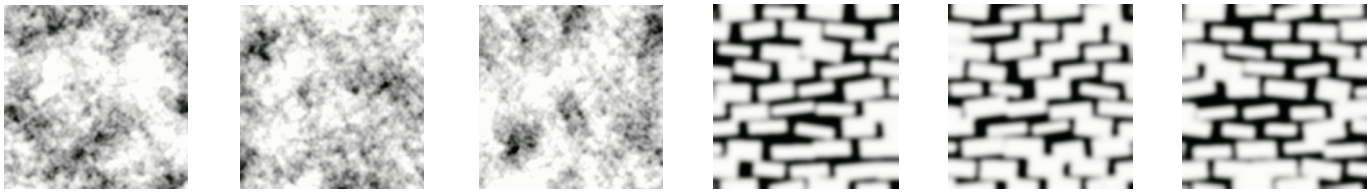


illustration 1.1 & 1.2: different noise & brick patterns generated each with the same parameter set, but different random seeds.

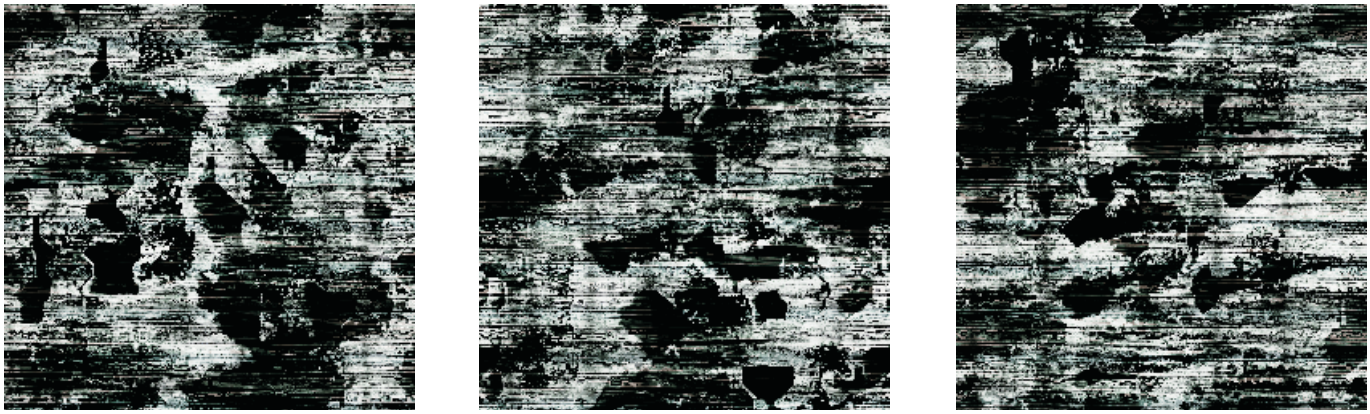


illustration 1.3: three textures - with similar but not identical patterns.

Chapter 1
Texture Editor Overview

Let's start with some basics..

Each texture consists of 9 layers and each layer can contain a generator (perlin noise, cellular..).
can add up to 4 filters to modify the generated layer.
All these layers are blended together, similar to those in Photoshop™ and your texture is finished.
You can generate up to 16 textures in one, let's say session, and those 16 textures form one texture set. Within this set you can use one texture as input for another texture (directly or indirectly as input for a filter). Well that sound's quite complex, luckily illustration 2 shows the concept.

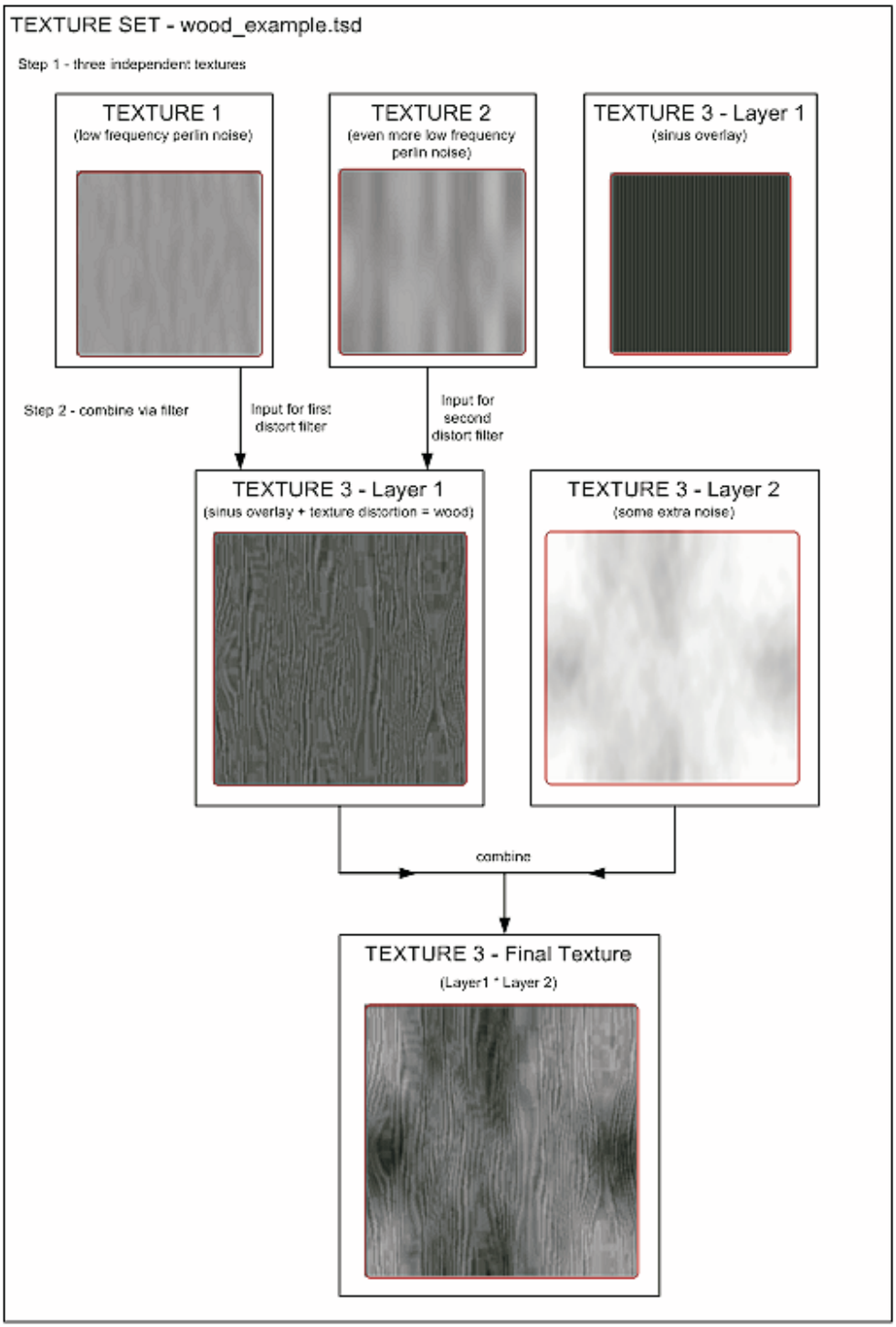


illustration 2: basic concept of procedural texture generation

Features

- interactive real-time texture generator
- OpenGL 3D Preview
- Code Library for TSD file import and texture generation

Requirements

Minimum Requirements

- x86 CPU with MMX Instruction Support (Pentium Pro, AMD K6)
- OS: MS Windows 2000 / XP.
  (Will not run properly under Windows 95, Windows98; not tested on Windows ME, Windows NT)
- 1024x768 pixels screen resolution with 16 million colors (32 bit color depth)
- 128 MB RAM

Recommended Requirements

- very fast x86 CPU for real-time editing (2 GHz and more).
- 1280x1024 pixels screen resolution with 32bit color depth
- 256 MB RAM

Chapter 2
Texture Editor Basics

Main View

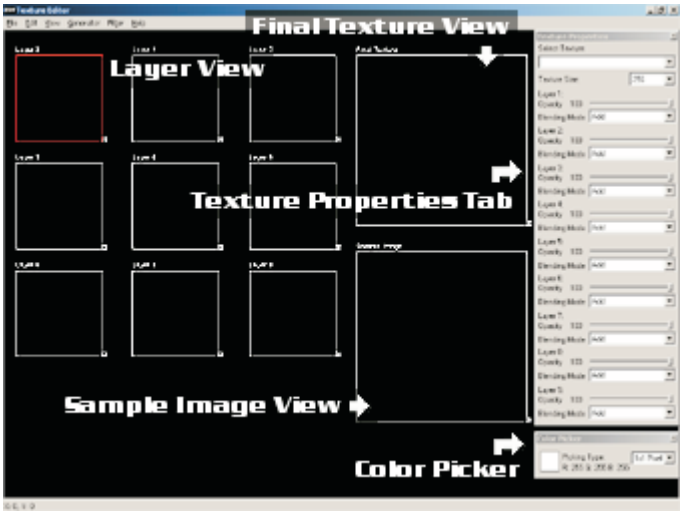


illustration 3: the main view & basic elements

Set layer as active

Click with the left mouse button in a layer rectangle to select a layer. The layer has now a red rectangle. All operations are applied to the active layer. By using the arrow keys, you can also switch quickly to another layer.

Move layer

By moving your mouse pointer over the top of a layer the background color turns into a dark blue. Now, push and hold the left mouse button over the layer to move it.

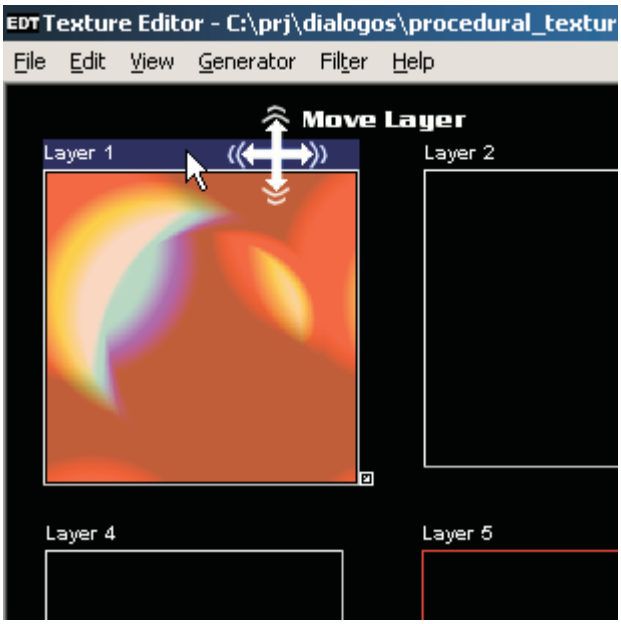


illustration 4: layer movement

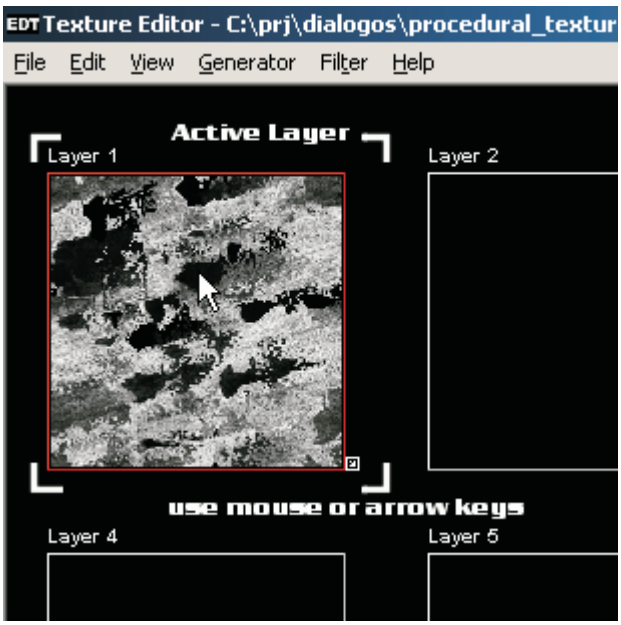
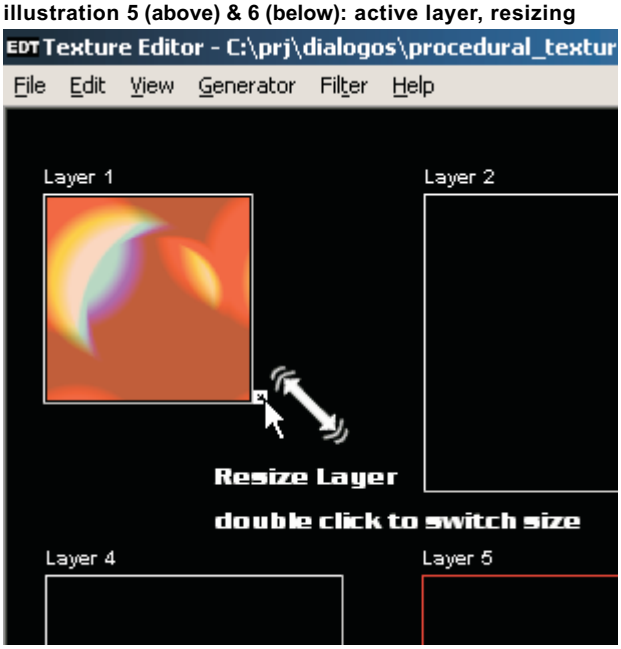


illustration 5 (above) & 6 (below): active layer, resizing



Resize layer

Click with the left mouse button on the small arrow on the bottom left side of the layer to resize a layer.

By double clicking on it you can switch to the original (texture) size of the layer.

Full Size Preview

If you want to see the layer in its original size push the right mouse button over any of the views. A new window pops up which can be closed simply by pushing again the left mouse button over the preview window.

Chapter 2
Texture Editor Basics

The Texture Properties Tab

Here you select the texture you want to edit, setup the texture size and control layer blending. Texture Size can be varied from 16 x 16 up to 1024 x 1024.
For each layer you define opacity and an appropriate blending mode. The blending of the first layer in use is always set to add no matter if you setup a different mode.
There are four different blending modes available: *Add, Multiply, Overlay, Difference*.

Add:	layers are added, the result is clamped to the range (0..255).
Multiply:	multiply layers the result gets darker
Overlay:	pixels with intensities over 50% get brighter, pixels with lower intensities darker
Difference:	layers are subtracted the result is clamped

If you fill an empty layer with a generator the appropriate generator name appears in the property tab.

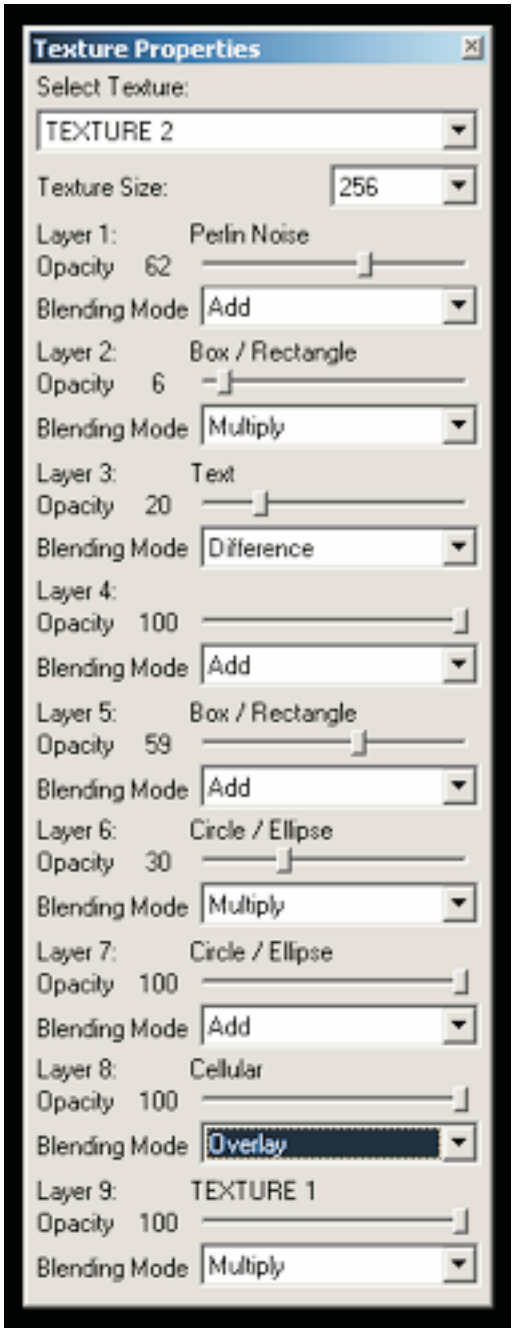


illustration 7: texture properies tab

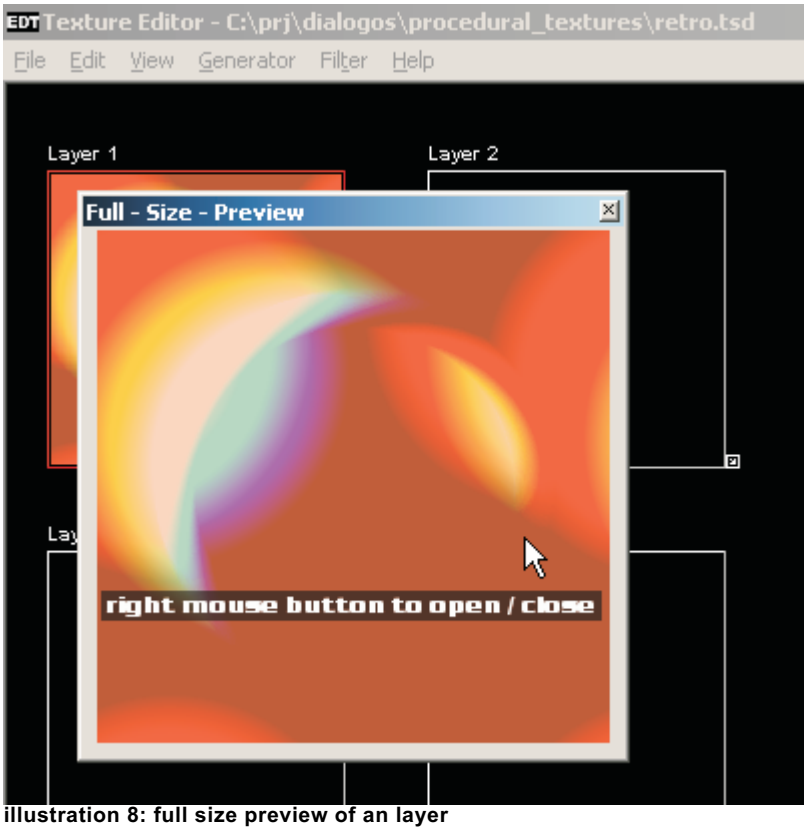


illustration 8: full size preview of an layer

Chapter 3
Generators

Introduction

Generators are the most important part of the Texture Editor, although there are only 8 different generators at the moment (Yeah I'm going to extend that later) you can do quite a lot of things. Experiment as much as you can and improve visual output by adding filters.
Since in most cases you need a while to get the right parameters, efficient editing is important. Try to use as much keyboard shortcuts as you can.

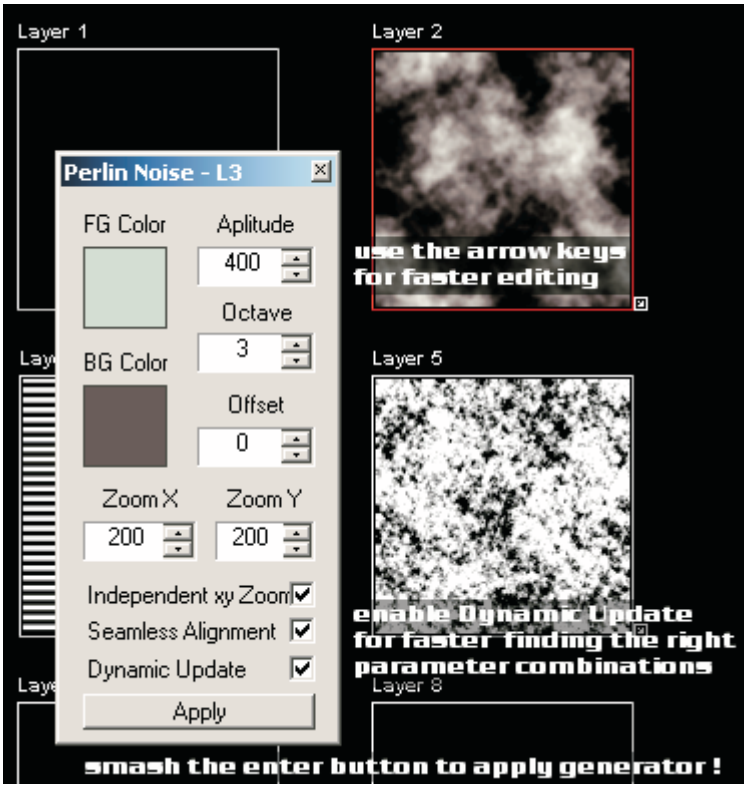


illustration 9: perlin noise generator

Every generator dialog has a "Dynamic Update" checkbox. When Dynamic Update is enabled the generator is applied to the actual layer every time you change a generator parameter. This can be quite useful when trying to find the right parameter combination but it can also get really slow when used on large textures (512 x 512, 1024 x 1024). For testing purposes it is recommend to turn down the texture size.
You can also hit the enter button instead of clicking every time with the mouse on the apply button.

In many cases you will find edit boxes where you have to enter numbers. You can increase or decrease the numbers by clicking with to mouse on the up/down arrows or you can use the up/down arrow keys for faster editing.

Often you have to enter position values who refer to the coordinate system used in this application. [0,0] refers always to the upper left corner of your layer.

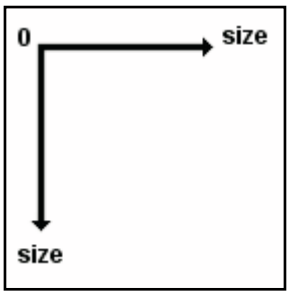


illustration 10: coordinate system

Sinus Overlay

Quite simple one to generate gradients, dot patterns, horizontal & vertical lines.
With the right scaling & position values, the texture gets tileable.
Combine several Sinus Overlay generators with different parameters and colors to get cool patterns.
Foreground and background color are averaged together, so high contrast can not be achieved, use contrast filter or USM (UnSharpMask) filter to gain higher contrast.

Parameters:

Size X horizontal scale – higher values higher scaling

Size Y vertical scale – higher values – higher scaling

Position X horizontal position

Position Y vertical position

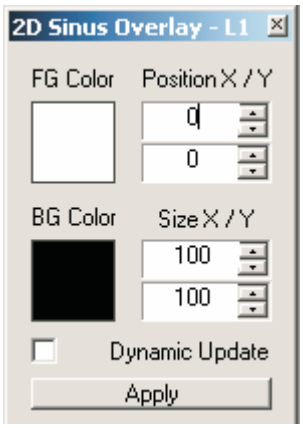


illustration 11: sinus overlay filter

Chapter 3
Generators

Perlin Noise

A very popular noise generator, all frequency bands are summed together, supports independent horizontal & vertical zoom factors, and can be used in tileable and un-tileable mode.

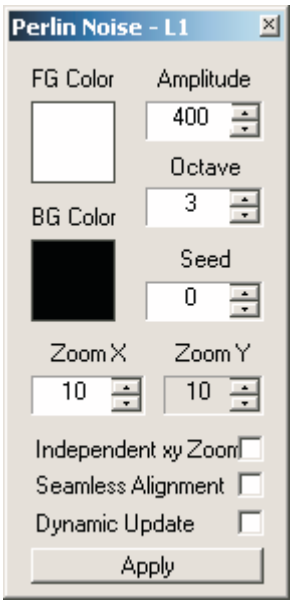


illustration 12: perlin noise generator

Parameters:

Amplitude	higher values result in higher contrast. It is only used if Octave > 1
Octave	number of frequency bands. Since using more frequency bands increases computation time, try to use as few as possible. At a certain amount, which depends upon the used zoom factors, the visual impact is not noticeable.
Seed	some kind of offset to produce different patterns
Zoom X	self-explaining
Zoom Y	self-explaining
Seamless Alignment	produces a tileable texture

Circle / Ellipse

Generates a circle or an ellipse.

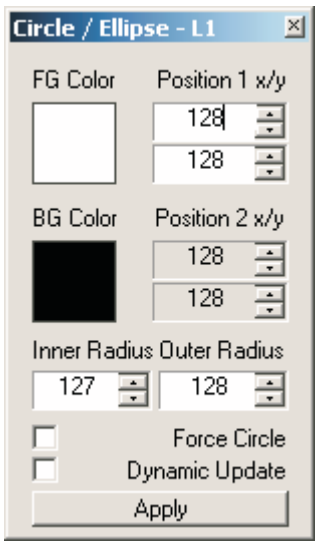


illustration 13: circle / ellipse generator

Parameters:
Self explaining - use different position values to produce an ellipse. If ellipse disappears, use a higher radius or reduce distance between the positions.

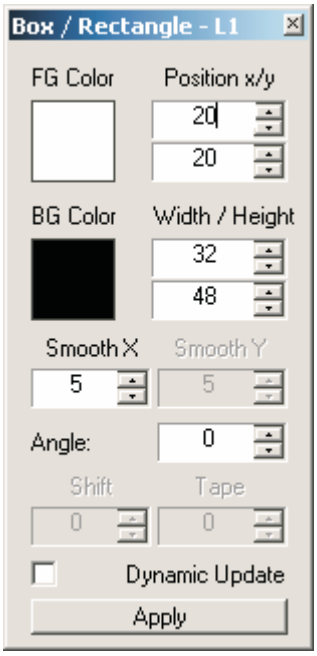


illustration 14:box generator

Box / Rectangle

Generates a box or a rectangle. The greyed input fields are reserved for further versions.

Parameters:

Position X	horizontal position from the middle of the box
Position Y	vertical position from the middle of the box
Width	self explaining
Height	self explaining
Angle	in tenth of degrees (0 – 3600)
Smooth X	in pixels - produces smoother edges

Chapter 3
Generators

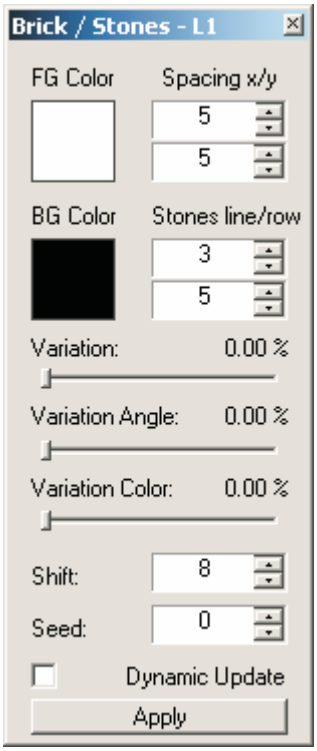


illustration 15:brick generator

Brick / Stones

Generates a tileable brick / stone pattern.

Parameters:

Space X	mortar width in pixels
Space Y	mortar height in pixels
Stones line	number of stones per line – the exact number of stones can vary sometimes.
Stones row	number of stones per row – the exact number of stones can vary sometimes.
Variation	varies the width, height and stone positions
Variation Angle	varies the angle of stones.
Variation Color	varies the color intensities of stones.
Shift	shift each line by the number of pixels into right direction
Seed	random seed used for variation, variation angle and color variation

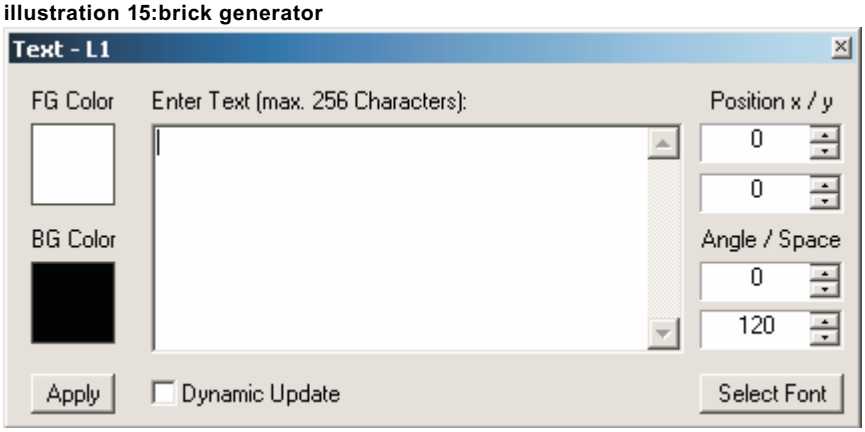


illustration 16: text generator

Text

Generates text with up to 255 characters.
The background color is only used within the bounding rectangle of the characters.

Parameters:

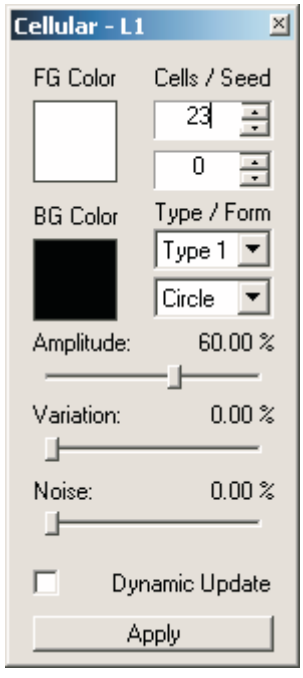
Space distance between text lines
Angle in tenth of degrees (0 – 3600)

Cellular

Generates a wide range of fancy cellular patterns. The texture is tileable (in most cases)

Parameters:

Cells	number of cells, randomly placed within the texture. High cell numbers produce slightly performance penalties.
Seed	some kind of offset to produce different patterns.
Type	Type 1 : the classical cell structure Type 2 : fancy pattern, use low amplitude values otherwise the texture get filled with the foreground color Type 3: similar to Type 1, no gradients



Chapter 3
Generators

Form	Circle:	generates circular patterns
	Box:	generates square patterns
	Star:	generates star patterns
Amplitude	high Amplitude – much foreground color, adjust according to the different types & forms	
Variation	every cell gets some kind of different radius	
Noise	to add extra noise	

Texture Source

Adds an existing texture as layer. This is helpful to add filters to an entire texture or if 9 layers are not enough to produce a texture. Note that infinite recursions are not allowed. Example: texture B uses texture A as input layer and texture A uses Texture B. So texture B would calculate texture A, and texture A would calculate Texture B . As this would never end and since it is really stupid we do not allow this, instead you get the error message: "Recursive texture definitions are not allowed !"
Note that wherever it is possible to specify source textures such error messages can appear.

Synthesize

An image synthesizer. Use this to produce larger textures from smaller input textures. The synthesizer picks random blocks from the input image and randomly assembles it to a larger texture, without resizing. Try to experiment a bit with this generator, you can get fancy and great looking results. It can also be used to modify a texture without changing its size. Tileable versions of untileable patterns can be created with ease using this generator.

Parameters:

Block Size	size of the random patterns
Overlap Ratio	specifies the overlap region of the blocks, higher overlap ratio produces harder edges. Overlap ratios that are lower than its corresponding Block Size results in hard edges without overlap region.
Seed	some kind of offset to produce different patterns.
Quality	high quality produces perfect matching blocks and low quality produces more random blocks. Note that high quality can produce significant performance penalties.
Seamless Alignment	if switched to on, a tileable texture is produced it only works with high and high-end quality settings.

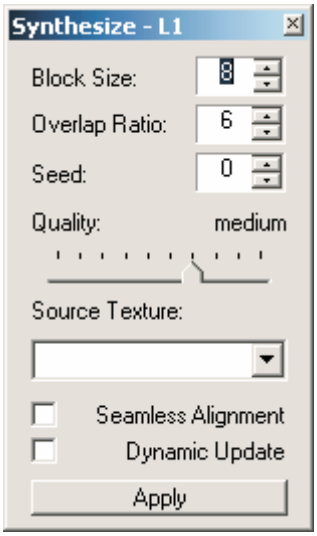


illustration 18: texture synthesizer

Chapter 4
Filters

Introduction

Filters can greatly improve the visual output. As said before filters can only be applied to layers. If you want to apply a filter on a whole texture, you have to do the following:

1. Create a new texture. (File-> New Texture)
2. Enter a name.
3. Go to: (Generator->Texture Source)
4. Select the desired texture
5. Apply filter

Existing filters can be edited by clicking Filter->Edit
At the current status only the last filter can be deleted (Filter->Delete).
Some Filters require a source texture, recursive texture definitions are not allowed (see Chapter 3, Texture Source).

Blur

A Gaussian blur filter. Edges are clamped, so tiling artifacts can be produced when blurring too much.

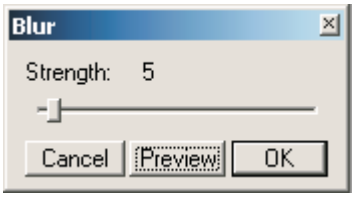


illustration 19: blur filter

Parameters:
Strength Blur pixel radius

Unsharp Mask (USM)

illustration 20: USM filter

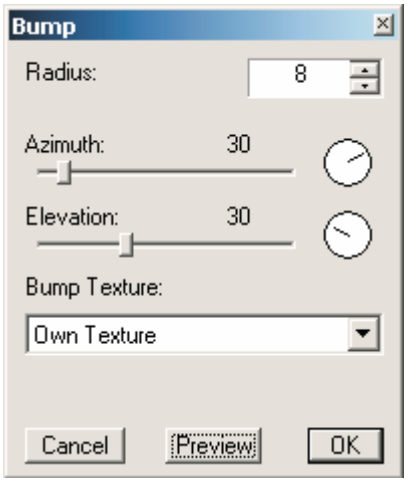
Use it to sharpen mid to high contrast. This filter locates surrounding pixels by differences in lightness values that you specify, and it increases their contrast by the amount you select by changing the Strength parameter.

Parameters:

Radius	small radius sharpens high frequencies, higher radius sharpens low frequencies.
Strength	sharpening strength
Clipping	all pixels under this intensity threshold are not sharpened

Bump

This filter produces some kind of bump or emboss effect. You can also use a different texture than the bump source. Make sure that the source texture does not use (in direct or indirect way) the texture in which the filter is applied. See Infinite recursion Problem explained in Texture Source Generator.



Chapter 4
Filters

Parameters:

Radius strength of the bump effect

Azimuth Azimuth angle in degrees (0..360), specifies light source direction

Elevation Elevation angle in degrees (0..90), specifies light source direction

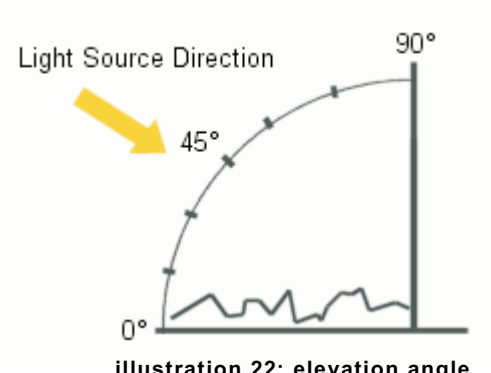


illustration 22: elevation angle describing the incoming light source onto a surface

Solarize

Inverts all pixels which have intensities greater or equal to the solarize threshold. A threshold of zero is used to negate the entire layer.

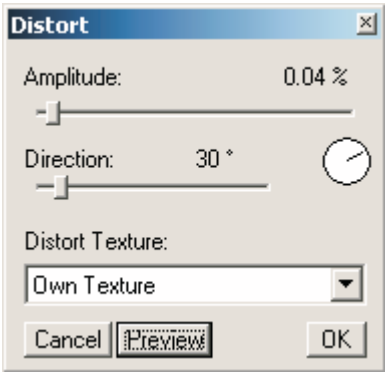


illustration 23: distort filter

Distort

Shifts pixels in a specified direction, the shift amount is specified by the source image. The first pixel in the layer is shifted by the intensity of the first pixel in the source image. Source and destination can be identical, every color component is processed seperatly. Use this to produce wood, marble or fancy LSD like effects. If you use grey-scaled source images the color components do not drift apart. The layer texture remains tileable, pixels that pass the right edge get to the left side and vice versa. The same applies to bottom and top.

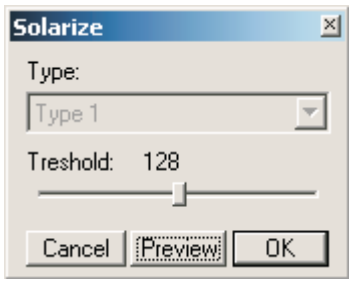


illustration 24: Solarize filter - the type parameter is reserved for further versions

Parameters:

Amplitude global shift amount

Direction shift angle in degrees (0..360)

Brightness / Contrast / Gamma:

Adjusts brightness, contrast and gamma, there is nothing more to say ;)

Chapter 5
Useful Tools

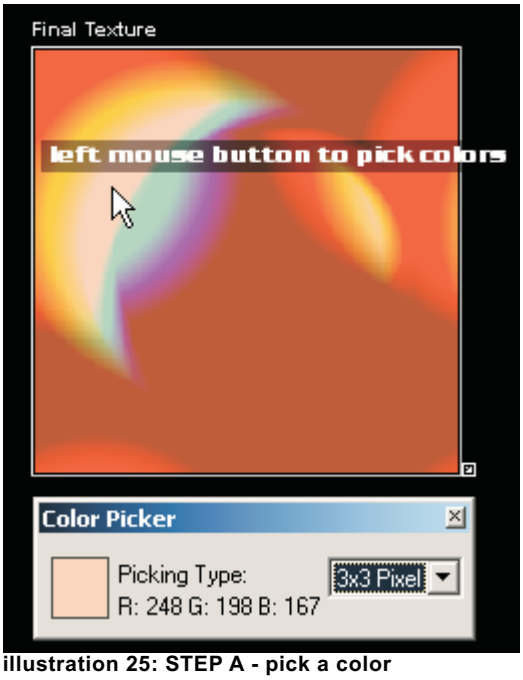


illustration 25: STEP A - pick a color

Color Picking Tool

Pick colors by clicking left mouse button into Final Texture View or into Sample Image.
Note that a Sample Image has to be loaded before colors can be picked. In Sample Image Difference View, the picking isn't allowed. You can also average picking colors by choosing picking fields of 3x3 up to 9x9 Pixels.
Once a Color is chosen you can easily drop it in a color field of a generator by pressing left mouse button.

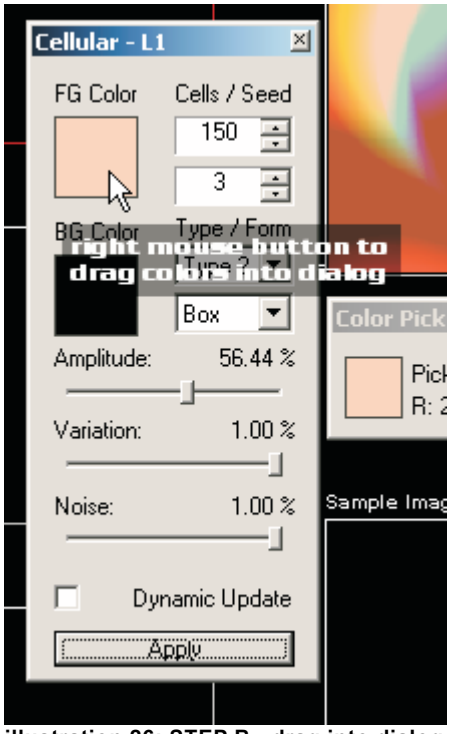


illustration 26: STEP B - drag into dialog window

The 3D Preview

Here you can see:
- How your newly generated texture looks on different mapping schemes
- Appearance with basic lighting
- tiling of your texture
- How it looks as an environment map

To open the 3D Preview click on View -> 3D Preview. Three basic mapping types can be choosen: Box, Cylinder, Sphere.
By dragging the left mouse button in the preview window you can rotate the object, use the right button to zoom in / out. Note that the window is resizable.

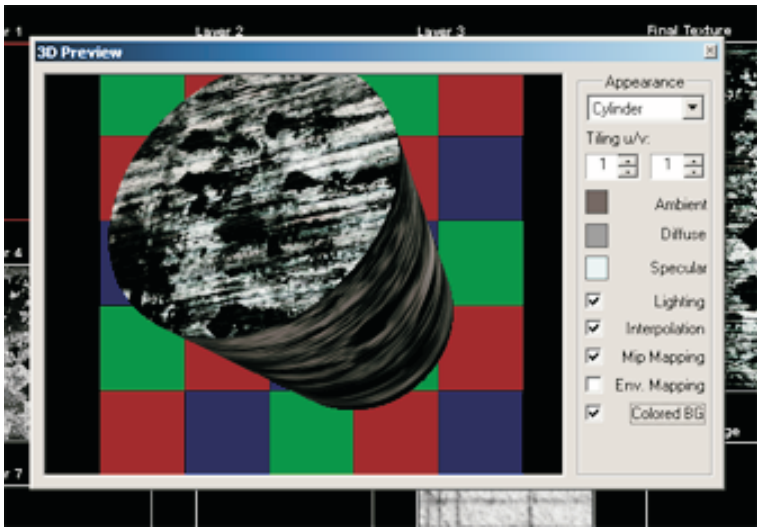


illustration 27: 3D Preview in action.

Sample Image

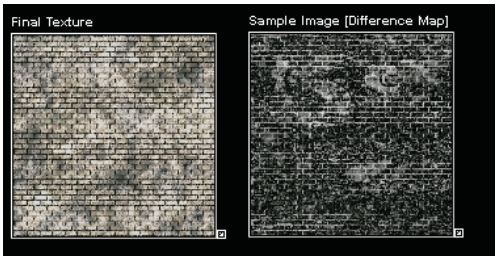


illustration 28: Final Texture & Differance Map of a photo and the (Final) Texture

This is useful when you have to rebuild procedural textures from existing bitmaps. At the current Version only uncompressed Windows Bitmaps are supported . All bitmaps must have n x n size which means the bitmap must be a quadratic one. In Difference Viewing Mode you see the difference of the sample image and the final texture. (high intensity = high difference)

Appendix A
Complete Operations List

File

New Texture	Create new, empty texture.
Delete Texture	Delete selected texture. The name of the selected texture can be seen in the properties tab.
Load Texture Set Load	Load complete texture set
Save Texture Set	Save complete texture set
Save Texture Set as	
Export Current Texture	
RAW File	Save current texture image as raw data (without any header information)
Windows Bitmap	Save current texture image as uncompressed windows bitmap
Delete Texture Set	Delete whole texture set.
Load Sample Image	Load Sample image (see Chapter 5)
Exit	Exit Application

Edit

Undo	Undo last operation. Second undo undos the undo operation (redo).
Cut Layer	Cut selected layer and places it on the clippoard.
Copy Layer	Copy selected layer.
Paste Layer	Paste clipboard content into selected layer.

View

Texture Properties	Show/hide texture properties tab.
3D Preview	Show/hide 3D Preview.
Color Picker	Show/hide color picker
Status Bar	Show/hide status bar (at the bottom).
Sample Image – Normal	Sample image normal mode (see Chapter 5)
Sample Image – Difference	Sample Image difference mode (see Chapter 5)
Align Layers	Align all views.

Generator

Sinus Overlay	see detailed generator description in Chapter 3
Perlin Noise	
Circle	
Box	
Brick	
Text	
Cellular	
Texture Source	
Synthesize	

Filter

Add	add one of the following filters
Blur	see detailed filter description in Chapter 4
Unsharp Mask	
Bump	
Solarize	
Distort	
Brightness / Contrast / Gamma	
Edit	modify already applied filters.
Delete	delete last filter on filter stack

Help

About EDT...	show Copyright and author information.

Appendix B
Short API Library Reference

Texture EDiTor - Library

This is a simple library to generate procedural textures directly in your application. The concept is quite straightforward:

1. Load as many TSD files as you want, you get an id for every loaded TSD file
2. Generate textures, by specifying the id and texture number

Enumerators:

Error Enumerators:

`TEDT_NO_ERROR`
No error

`TEDT_ERROR_INVALID_VALUE`
Numeric argument out of range

`TEDT_ERROR_INVALID_ENUM`
Enumerate out of range

`TEDT_ERROR_OUT_OF_MEMORY`
Not enough memory left to execute command

`TEDT_ERROR_WRONG_VERSION`
Incorrect file format

`TEDT_ERROR_FILE_NOT_FOUND`
File not found

`TEDT_ERROR_READ`
Can not read file

Output Enumerators:

`TEDT_FORMAT_BYTE_LUMINANCE`
luminance image, 1 byte per pixel

`TEDT_FORMAT_BYTE_RED`
red component, 1 byte per pixel

`TEDT_FORMAT_BYTE_GREEN`
Green component, 1 byte per pixel

`TEDT_FORMAT_BYTE_BLUE`
blue component, 1 byte per pixel

Appendix B
Short API Library Reference

TEDT_FORMAT_BYTE_RGB
Red, green, blue component, 3 bytes per pixel, interleaved order

TEDT_FORMAT_BYTE_RGBA
red, green, blue, alpha component, 4 bytes per pixel, interleaved order

TEDT_FORMAT_BYTE_BGR
blue, green, red, alpha component, 3 bytes per pixel, interleaved order

TEDT_FORMAT_BYTE_BGRA
blue, green, red, alpha component, 4 bytes per pixel, interleaved order

TEDT_FORMAT_FLOAT_LUMINANCE
luminance image, float single precision (4 bytes per pixel)

TEDT_FORMAT_FLOAT_RED
red component, float single precision (4 bytes per pixel)

TEDT_FORMAT_FLOAT_GREEN
green component, float single precision (4 bytes per pixel)

TEDT_FORMAT_FLOAT_BLUE
blue component, float single precision (4 bytes per pixel)

TEDT_FORMAT_FLOAT_RGB
Red, green, blue component, float single precision (12 bytes per pixel), interleaved order

TEDT_FORMAT_FLOAT_RGBA
Red, green, blue, alpha component, float single precision (16 bytes per pixel), interleaved order

TEDT_FORMAT_FLOAT_BGR
blue, green, red component, float single precision (12 bytes per pixel), interleaved order

TEDT_FORMAT_FLOAT_BGRA
blue, green, red, alpha component, float single precision (16 bytes per pixel), interleaved order

Texture / Layer Status Enumerators:
TEDT_STATUS_EMPTY
Texture / Layer is empty

TEDT_STATUS_PRESENT
Texture / Layer is present

TEDT_STATUS_UNDEFINED
Texture / Layer is undefined; this happens when id, texture or layer was specified outside the range

Appendix B
Short API Library Reference

Functions:

```
int   tedtLoadTextureSet(const char *filename);
```
Load TSD file.
Parameters:
filename TSD filename
Return value:
TSD id, on error zero

```
int tedtGetError();
```
Get last error.
Parameters:
none
Return value:
error enumerator

```
int tedtGetTextureSize(int id, int texture);
```
Get texture size.
Parameters:
id id referencing the texture set
texture texture number (1..16)
Return value:
Texture size or zero if texture doesn't exists.

```
int tedtGetTextureStatus(int id, int texture);
```
Get texture status.
Parameters:
id id referencing the texture set
texture texture number (1..16)
Return value:
status enumerator

```
int tedtGetLayerStatus(int id, int texture, int layer);
```
Get layer status.
Parameters:
id id referencing the texture set
texture texture number (1..16)
layer layer number (1..9)
Return value:
status enumerator

Appendix B
Short API Library Reference

```
void tedtRenderTexture(int id, int texture, int format, void *pixels);
```

Render texture. If texture is empty, the memory block is not touched.

Parameters:

id	id referencing the texture set
texture	texture number (1..16)
format	your desired output format, must be one of the Output Enumerators
pixels	pointer to a memory block; make sure *you* have allocated enough memory

Return value:

none

```
void tedtRenderLayer(int id, int texture, int layer, int format, void *pixels);
```

Render layer. If layer is empty, the memory block is not touched.

Parameters:

id	id referencing the texture set
texture	texture number (1..16)
layer	layer number (1..9)
format	your desired output format, must be one of the Output Enumerators
pixels	pointer to a memory block; make sure *you* have allocated enough memory

Return value:

none

Appendix C
Legal Disclaimer and Copyright Information

Copyright (c) 2001-2003 Aick in der Au, All rights reserved.

Permission to use, copy, and distribute this software and its documentation, without any modification, for any purpose (including commercial) and without fee or royalty is hereby granted.

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

DO NOT USE THIS SOFTWARE FOR DIRECT OR INDIRECT CREATION OF WEAPONS OF MASS DESTRUCTION.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.