



.theprodukt

# Procedural Texture Generation

dierk.chaos.ohlerich, .theprodukt GmbH

Vortrag, 20. August 2007



.theprodukt

... will make you happy

## what is Procedural Texture Generation?

cooking recipe  
(list of steps, formula, program)

generator

image



... will make you happy

## in contrast to image compression

original image

packer

binary data

depacker

depacked image



... will make you happy

## how good is it?

Debris, 177kb exe

- texture data
- geometry data
- music score and synthesizer patches and all the code
- procedural texture generator
- procedural geometry generator
- synthesizer and sequencer
- 3d engine
- windows startup code
- ...

## other demo: kkrieger

- fps in 96k
- doom3-style engine





... will make you happy

## why use procedural techniques?

### to save memory

- deployment (download, DVD)
- storage (HD, Flash)
- graphics RAM (when done in pixel shader)
- during development



.theprodukt

... will make you happy

## why use procedural techniques?

### to save memory

- deployment (download, DVD)
- storage (HD, Flash)
- graphics RAM (when done in pixel shader)
- during development

### for the sideeffects

- resolution independend
- no compression artefacts
- nonlinear editing
- tiling
- no photo required
- normal map generation
- mutable content

procedural textures should become a common technique in game studios!



... will make you happy

## how is it done?

### break up the image into simple operations.

- starting points: noise, gradients, simple shapes, circles, rectangles, text, ...
- color correction : contrast, brightness, hue, color balance, ...
- combining: add, mul, screen, alpha blending, ...
- others: rotate, zoom, blur, sharpen, lighting, distortion, ...



... will make you happy

## functional approach

- traditional approach, older than textures themselves
- pixel = function(u,v)
- example: perlin noise



... will make you happy

## full image approach

- every step of the operation works on whole image



... will make you happy

## full image approach

- every step of the operation works on whole image

### simple decision:

- functional approach gives elegant code
- full image approach gives great images



... will make you happy

## drawbacks of functional approach

### almost impossible to make „blur“

- actually integrate / fold with gaussian curve
- spawn many samples
- defer the „blur“ to its input



... will make you happy

## drawbacks of functional approach

### almost impossible to make „blur“

- actually integrate / fold with gaussian curve
- spawn many samples
- defer the „blur“ to its input

### other hard ops:

- hightmap to normal map conversion
- text and other shapes
- everything non-trivial non-linear



.theprodukt

... will make you happy

**user interface**



... will make you happy

## user interface

- it takes many, many operators to make a great image
- you need a tool to quickly organise operators



... will make you happy

## tree and graphs

- it's a directed graph
- most of it looks like a tree
- actually, a small directed graph of large trees



... will make you happy

## tree and graphs

- it's a directed graph
- most of it looks like a tree
- actually, a small directed graph of large trees

### user interfaces:

- graph: boxes and wires
- trees: folders



... will make you happy

## operator stacking

- no wires require
- where you need: connect by „load and store“
- you can generate high quality bumpmaps on the fly
- it is very good at generating „textures“



.theprodukt

... will make you happy

## operator stacking

- no wires require
- where you need: connect by „load and store“
- you can generate high quality bumpmaps on the fly
- it is very good at generating „textures“

### hard to learn?

- uncommon at first
- highly intuitive after one hour
- very little to learn in the end



.theprodukt

... will make you happy

## implementing texture generators



.theprodukt

... will make you happy

## some implementation choices

### color depth of intermediate storage:

- 8 bit integer: inadequate, especially for „hightmap to normal map conversion“
- 32 bit float: if you can afford the bandwidth...
- 16 bit integer: good
- 16 bit float: may be better for HDR



... will make you happy

## some implementation choices

### color depth of intermediate storage:

- 8 bit integer: inadequate, especially for „hightmap to normal map conversion“
- 32 bit float: if you can afford the bandwidth...
- 16 bit integer: good
- 16 bit float: may be better for HDR

### number of channels

- RGB required
- RGBA with premultiplied alpha is a good choice
- Monochrome path advantageous
- 2-channel format for normal maps?



.theprodukt

... will make you happy

## some implementation choices

### reproducability:

- be really carefull about this...



.theprodukkt

... will make you happy

## some implementation choices

### reproducibility:

- be really carefull about this...

### where to do it?

- CPU: a good candidate for perfect reproducibility and portability in integer
- GPU: high bandwidth be aware of 16 bit filtering and blending
- SPU: high bandwidth and very flexible, local store is quite small



.theprodukt

... will make you happy

## parallel processing?

- because of the tree-like nature there is lot's of inherent parallelism
- render siblings independent
- generate multiple textures simultaneously
- schedule with caches in mind



.theprodukt

... will make you happy

## parallel processing?

- because of the tree-like nature there is lot's of inherent parallelism
- render siblings independent
- generate multiple textures simultaneously
- schedule with caches in mind

### on GPU's

- don't use last blit's rendertarget immediatly as texture



.theprodukt

... will make you happy

## DXT (S3TC)

- highest quality texture compression is SLOW
- fast texture compression is not that bad



... will make you happy

## optimising trees

- don't treat multiple textures separate. mix'em all
- find common subtree's
- encourage library use, then there will be a lot to find
- create multiple variants of a texture

can decrease filesize and generation time by a „factor“



.theprodukt

... will make you happy

**standardisation**



.theprodukt

... will make you happy

## standardisation

- technology is ready to use but rarely used
- should be part of common toolset
- standardisation might help
- especially on platforms without proper programming language (mobiles)



... will make you happy

## standardisation

- technology is ready to use but rarely used
- should be part of common toolset
- standardisation might help
- especially on platforms without proper programming language (mobiles)

### **extract and standardise the performance critical part**

- many choices for implementing the generation process
- once an image is in the generator, it needs an optimized storage format

.theprodukt

... will make you happy

## our standard draft

- OGL style C API
- specify complete operator graphs
- then press „execute“
- this allows any choice of implementation within accuracy bounds



.theprodukt

... will make you happy

## our standard draft

- OGL style C API
- specify complete operator graphs
- then press „execute“
- this allows any choice of implementation within accuracy bounds

### **you can insert images into the system before we press „execute“**

- easy to allow input images. (in contrast to modifying something in the middle)
- hard to standardize input operators
- easy to standardize all other operators, THERE IS A common subset
- tool-vendors can differentiate their products without compromising the standard
- input operators rarely critical to performance



... will make you happy

## astonishingly simple interface

- only 15 operators
- can be combined to form anything we ever needed
- more flexible than anything we ever designed before
- reference implementation: 1438 lines of code
- well suited for CPU, GPU and mobile implementation
- exact specification of accuracy and border cases



.theprodukt

... will make you happy

# q&a

.theprodukt GmbH, dierk.chaos.ohlerich



... will make you happy

## SVG? it has a pixel based image generator!

- incomplete feature set
- missuse required even for incomplete feature set
- slow performance
- inadequate quality



.theprodukkt

... will make you happy

## SVG? it has a pixel based image generator!

- incomplete feature set
- misuse required even for incomplete feature set
- slow performance
- inadequate quality

### SVG is highly usefull as input!

- you would not want to include SVG into the generator
- you might want to include the generator into SVG
- you can use SVG as input, since it might be available anyway
- same goes for font rendering



.theprodukkt

... will make you happy

## tiling approach for CPU's/SPU's

tile the images in chunks that easily fit into cache

classify operations in four classes:

- the no problem class: can work completely in tile (color correct)
- requires complete input (rotate, zoom, scroll)
- complete input and output (blur)
- writes complete output (?)



.theprodukt

... will make you happy

## tiling approach for CPU's/SPU's

tile the images in chunks that easily fit into cache

classify operations in four classes:

- the no problem class: can work completely in tile (color correct)
- requires complete input (rotate, zoom, scroll)
- complete input and output (blur)
- writes complete output (?)

so what do you do?

- work in tiles as long as possible
- store complete images only for non tiling ops