CS 563 Advanced Topics in Computer Graphics
Classifying Shaders

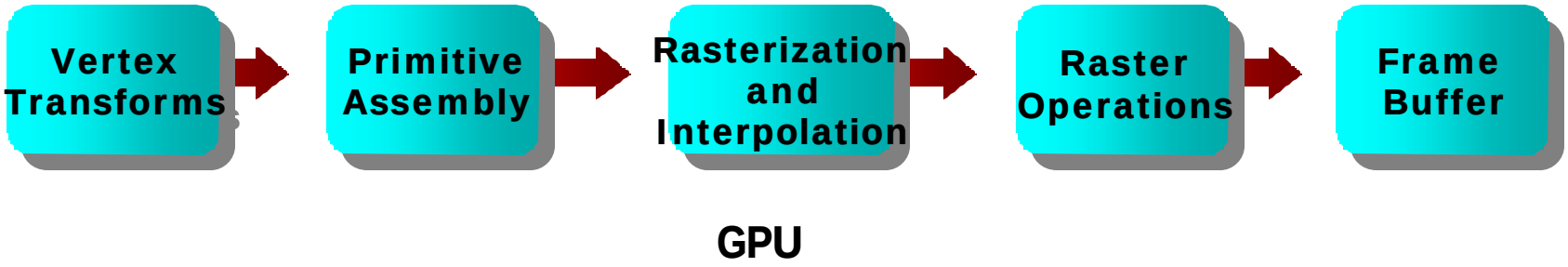
by Dan Adams

Topics

- Fixed function pipeline
- Parameterized shading
- Cook’s Shade Trees
- Programmable shading
- Procedural shading

Fixed-Function Pipeline

- The standard graphics pipeline
- A set number of T&L functions



GPU

http://www.cis.upenn.edu/~suvenkat/700/

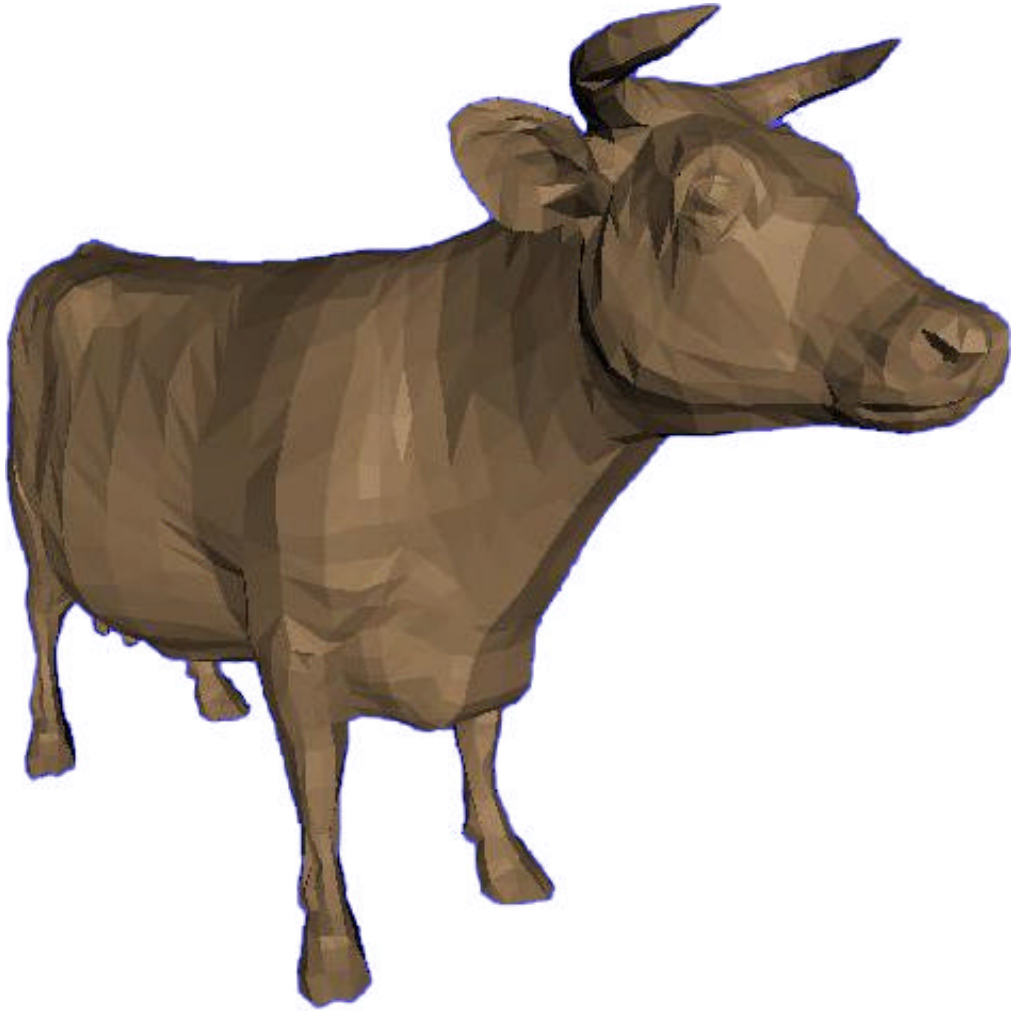
Fixed-Function Pipeline

- Limited to how it can be changed (ie pushing matrices into the pipeline)
- Implemented in hardware to be faster
- If it's not supported in the pipeline, you can't do it.

- Ex: glShadeModel()
  - Want to change the shading model when rendering
  - What can we do with the fixed-function shading?

Fixed-Function Pipeline

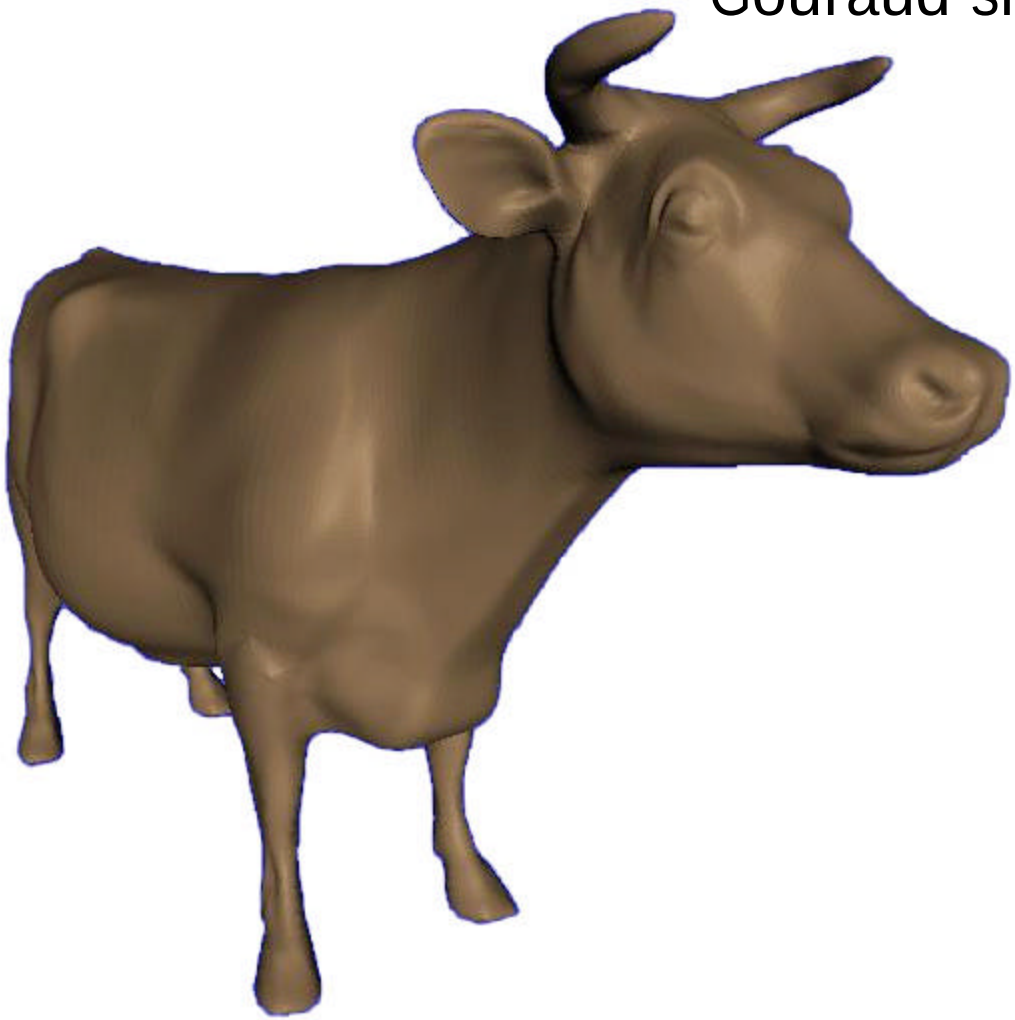
Flat shading (GL_FLAT)



http://www.cc.gatech.edu/classes/AY2003/cs4451_spring/linint.html

Fixed-Function Pipeline

Gouraud shading (GL_SMOOTH)



http://www.cc.gatech.edu/classes/AY2003/cs4451_spring/linint.html

Fixed-Function Pipeline

Phong shading (GL_?)

- Not supported in hardware so you can’t do it

http://www.cc.gatech.edu/classes/AY2003/cs4451_spring/linint.html

Parameterized Shading

- Add more flexibility but keep hardware speeds
- Hardware implementation of noise-based functions (clouds, wood, etc) based on Perlin Noise

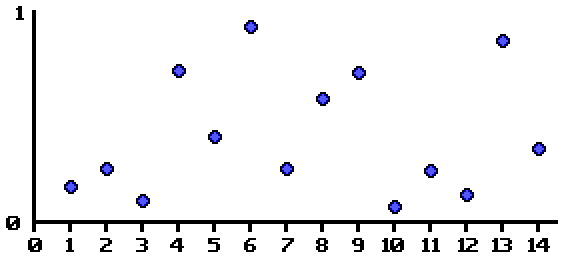
Perlin Noise

- A function which is composed of numerous noise function
- Results in infinitely non-repeating detail
  - Coastlines
  - Mountain ranges
  - Clouds
  - Water
  - Etc
- Has a very “natural” feel to it

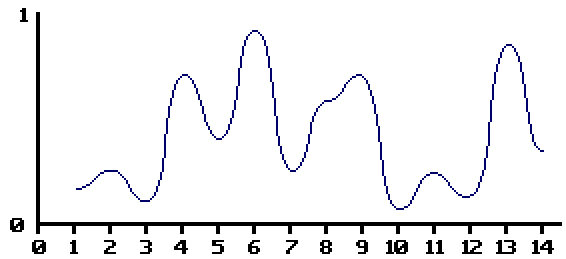
http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

Perlin Noise

- **Noise function** is basically a random number generator with a set seed
  - Get the same numbers each time it's run
- Randomness are what give the textures a "natural" quality



- Generate a number of random values

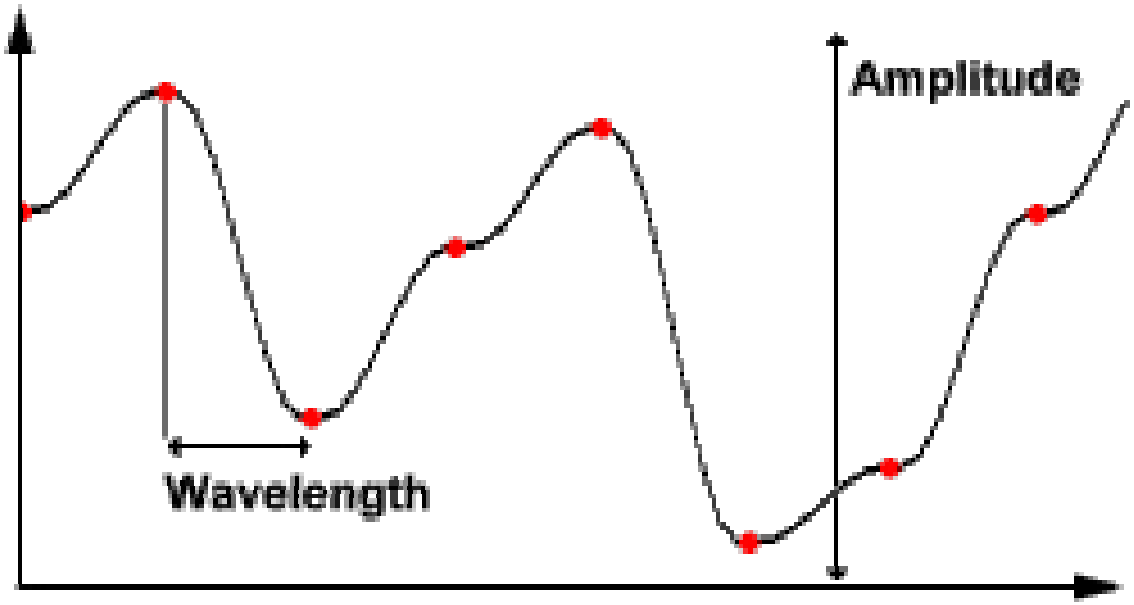


- Interpolate between them
- Can use linear or cubic interpolation or a curve

http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

Perlin Noise

- **Amplitude**, **wavelength**, and **frequency**

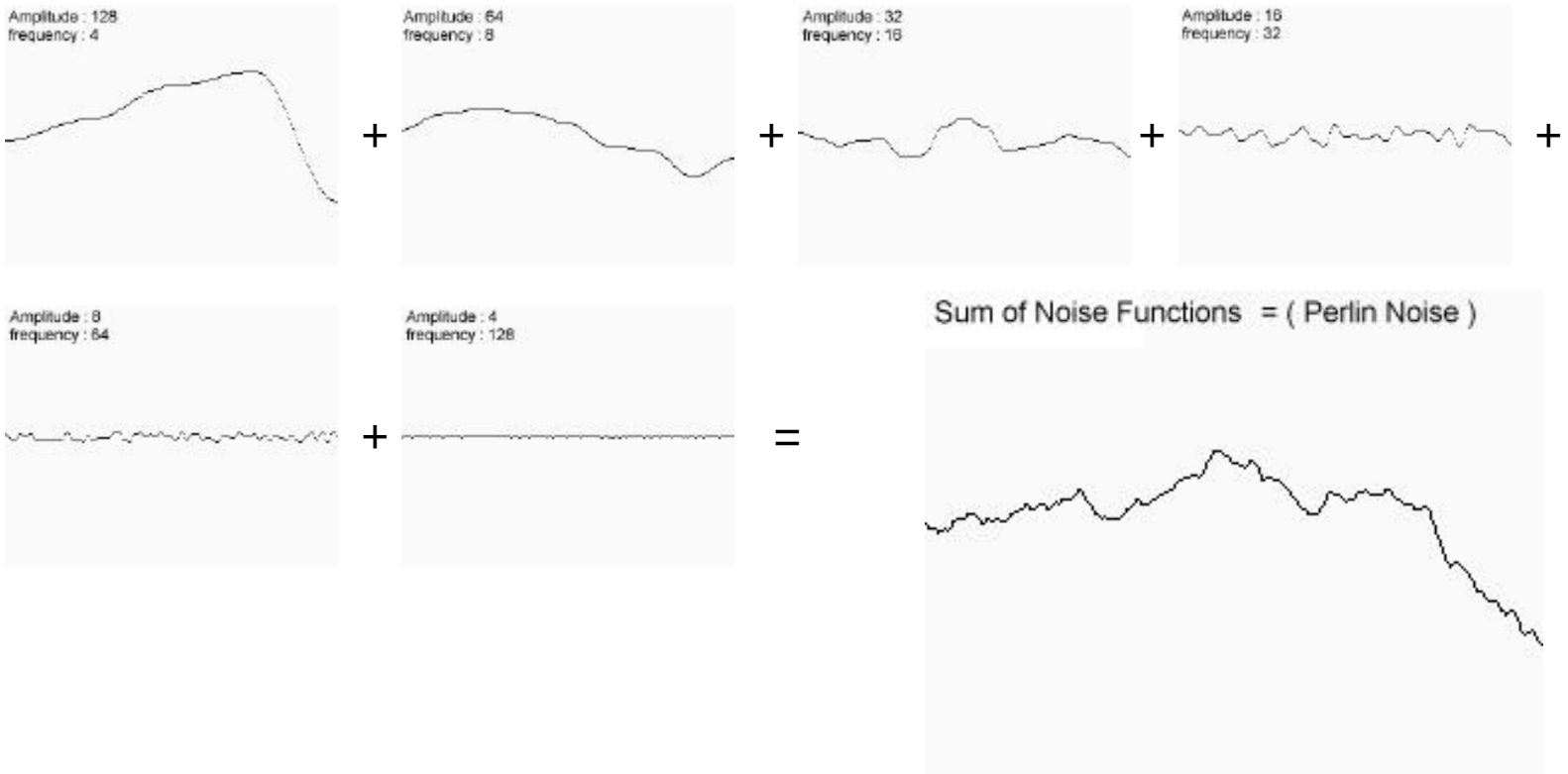


frequency = 1/wavelength

http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

Perlin Noise

- Each **octave** has twice the frequency of the previous
- Create a number of octaves using noise functions and sum together



http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

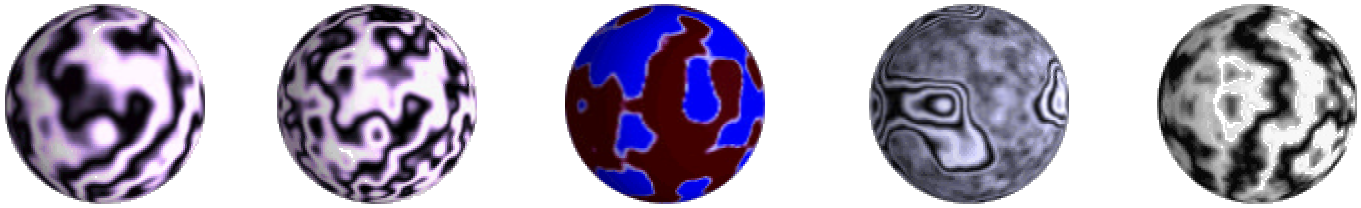
Perlin Noise

- The number of octaves created depends on the level of detail desired
- Too many octaves = wasted processing time
- Too few octaves = boring Perlin function

http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

Perlin Noise

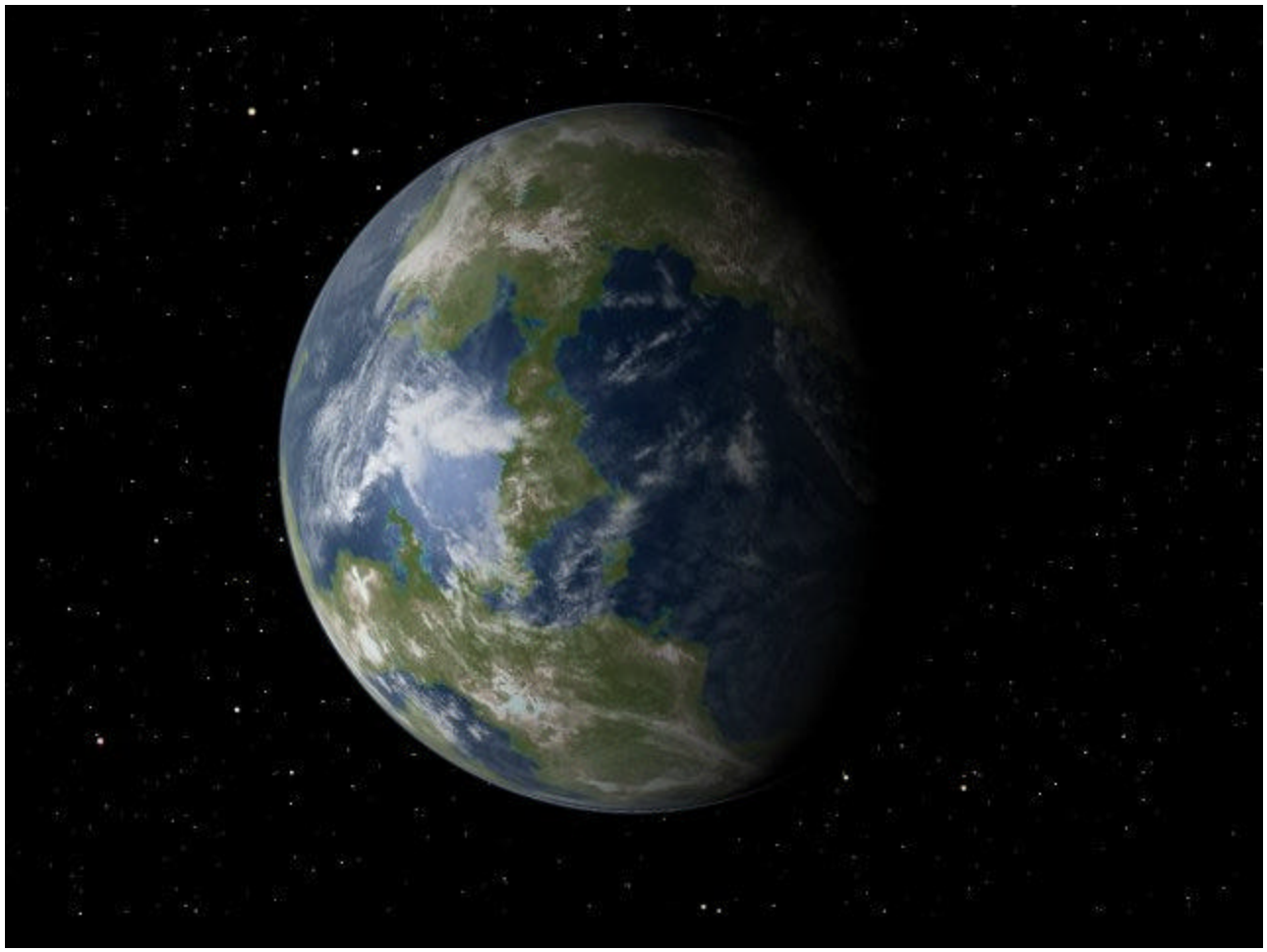
- Examples of noise-based textures:



http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

Perlin Noise

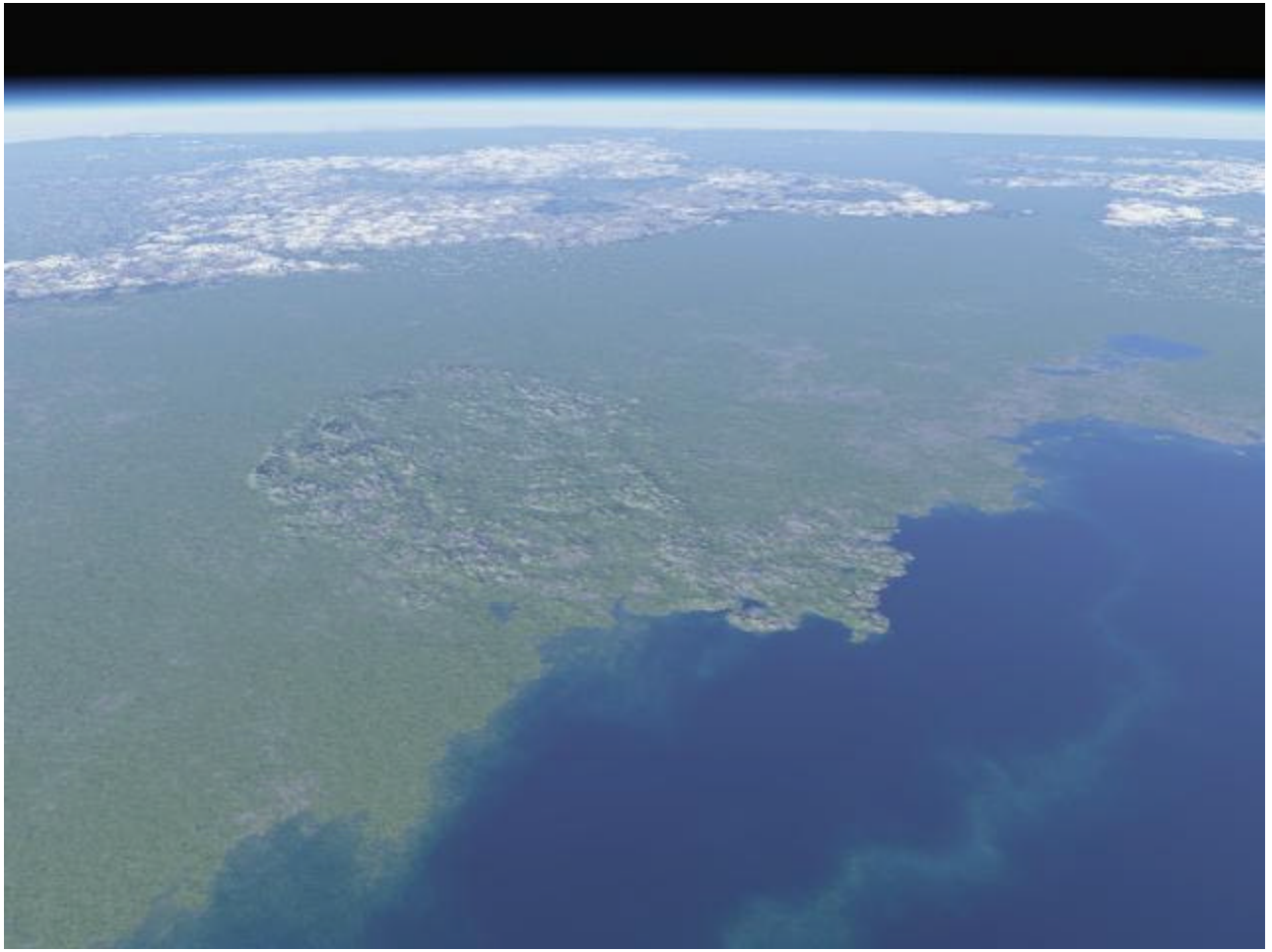
- Some more different examples



http://libnoise.sourceforge.net/examples/complexplanet/index.html

Perlin Noise

- Some more different examples



http://libnoise.sourceforge.net/examples/complexplanet/index.html

Perlin Noise

- Some more different examples

http://libnoise.sourceforge.net/examples/complexplanet/index.html

Perlin Noise

- Some more different examples

http://libnoise.sourceforge.net/examples/complexplanet/index.html

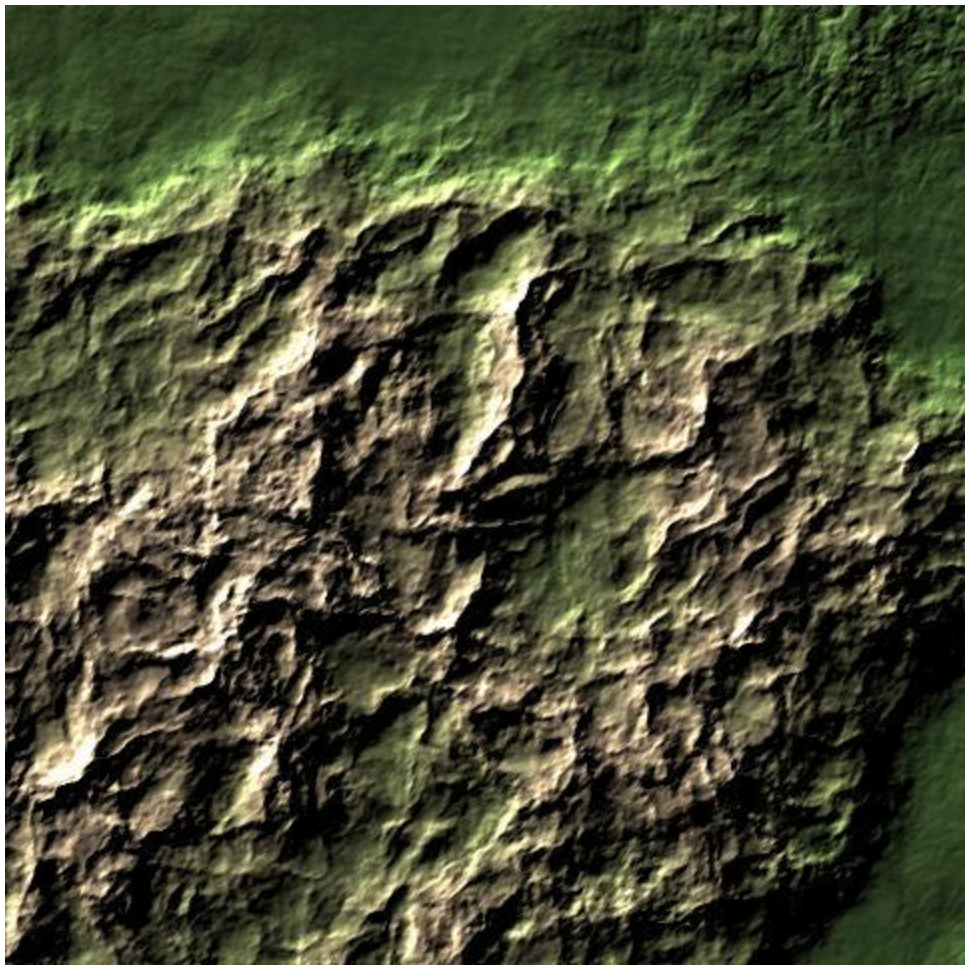
Perlin Noise

- Some more different examples

http://libnoise.sourceforge.net/examples/complexplanet/index.html

Perlin Noise

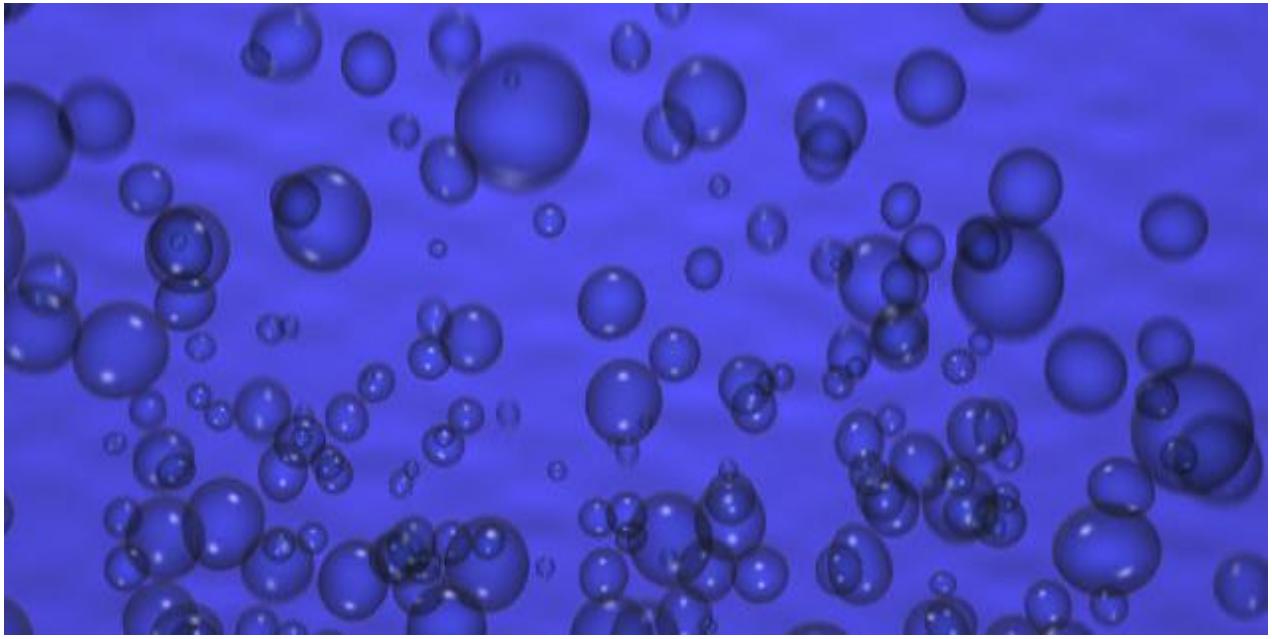
- Some more different examples



http://libnoise.sourceforge.net/examples/complexplanet/index.html

Perlin Noise

- Some more different examples



Noise used for random movement and perturbation of bubbles

http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Students/fruit/final_writeup.html

Parameterized Shading

- Now back to doing this in hardware...
- Implemented in a few different systems
  - Pixel Planes (1992)
  - PixelFlow (1989)
  - Pixel Machine (1989)
- We can use parameters to get different noise-based effects
- Rendering performed in GPU rather than CPU

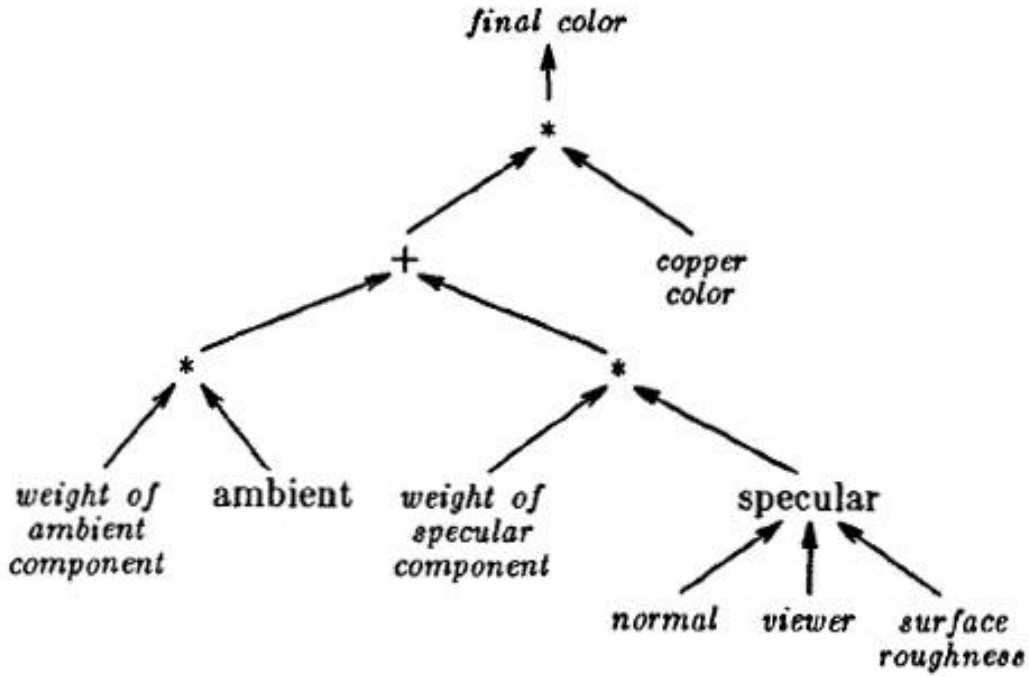
Parameterized Shading

- So now we have
  - Fixed-function pipeline abilities
  - Can supposed noise in hardware

- Can we do more?
- What if... we had some way to arbitrarily compose our own textures/shading?

Cook Shade Trees

- Presented by Cook in 1984 [cook]
- Using operators and operands we can compose our own effects using a tree structure



Shade tree for a copper texture

Cook Shade Trees

- Now take this idea and try to put it into an assembly-type form:
  - operator operand operand operand

Copper texture tree idea in this form:

a = specular normal viewer roughness

a = * a specweight

b = * ambientweight ambient

a = + a b

color = * a coppercolor

*Can think of the shade tree as an Abstract Syntax Tree (AST)

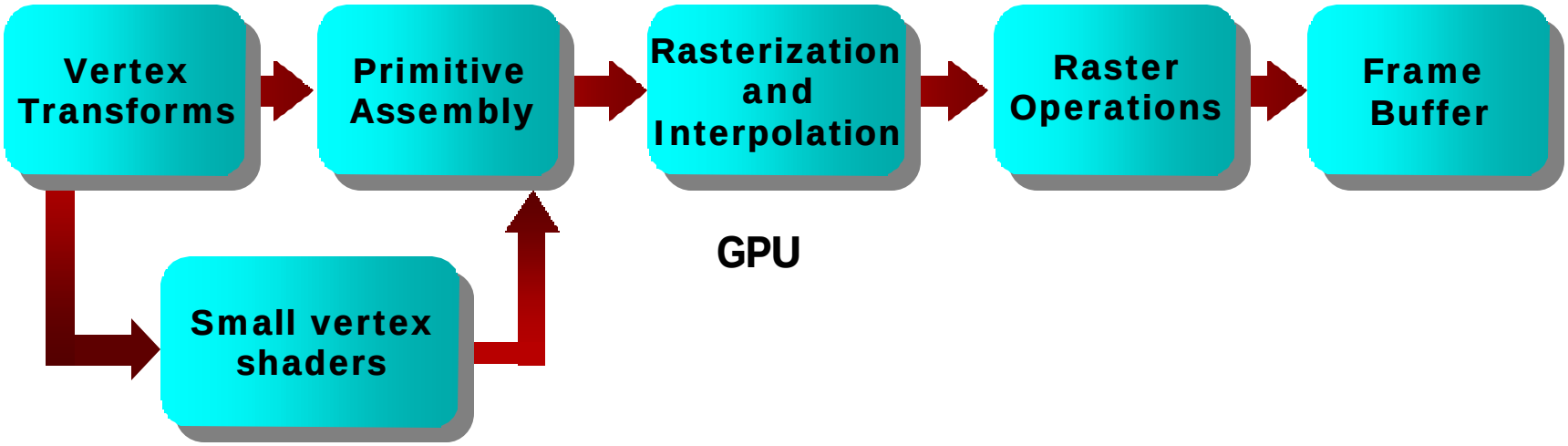
Programmable & Procedural

- The author uses these terms to basically mean the following:

- **programmable** – You can give assembly-type instructions to the GPU
- **procedural** – Can use a higher level language such as C

Programmable Shading

- Modifies the fixed-function pipeline to allow a set of user-provided instructions
- First implemented in the Pixel-Planes 5 in 1992
- First in commodity hardware in NVIDIA GeForce3



http://www.cc.gatech.edu/classes/AY2003/cs4451_spring/linint.html

Programmable Shading

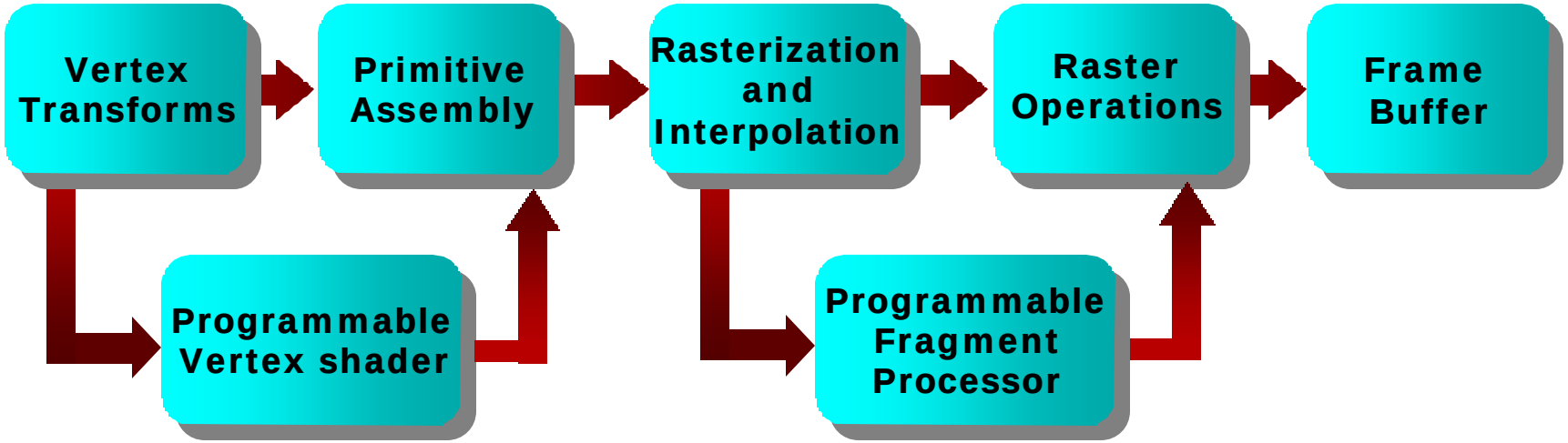
- Pros
  - Can now write programs for the GPU
- Cons
  - Very difficult to write programs
  - No control constructs (if, while, etc)

Procedural Shading

- Cook's Shading Trees provided some capabilities but not all
- Perlin's Image Synthesizer [pis] was the first fully capable
- Now have an instruction set which high-level languages can be compiled down to
  - Easer to write
  - Libraries of common functions
  - Can be optimized by the compiler

Procedural Shading

- Graphics hardware now provides programmable vertex and fragment (pixel) shaders
- First commodity hardware to support it were the Radeon 9700 and GeForceFX in 2002



http://www.cc.gatech.edu/classes/AY2003/cs4451_spring/linint.html

Procedural Shading

Procedural Shading

Melting Paint

From NVIDIA demo

Procedural Shading

Bump Horizon Mapping

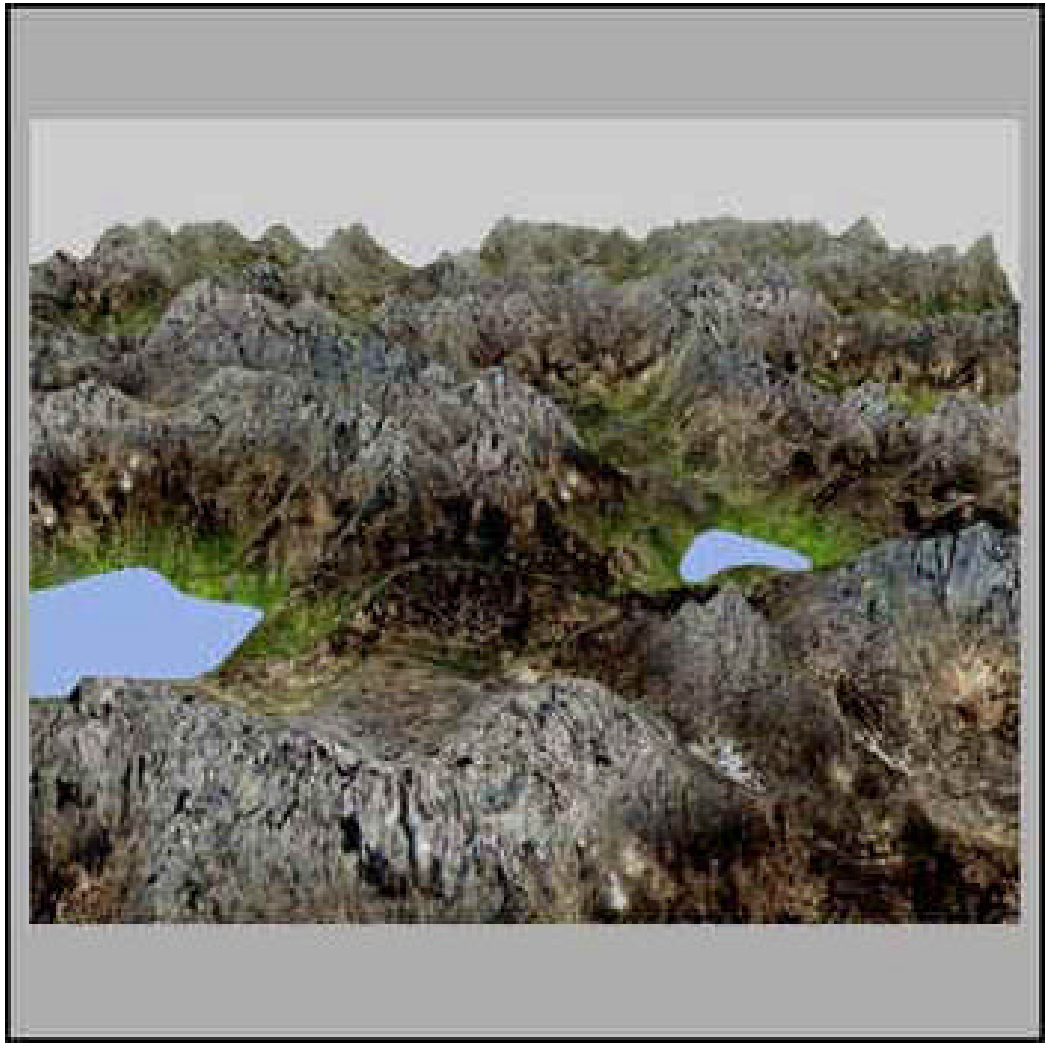
From NVIDIA demo

Procedural Shading

Flare

From NVIDIA demo

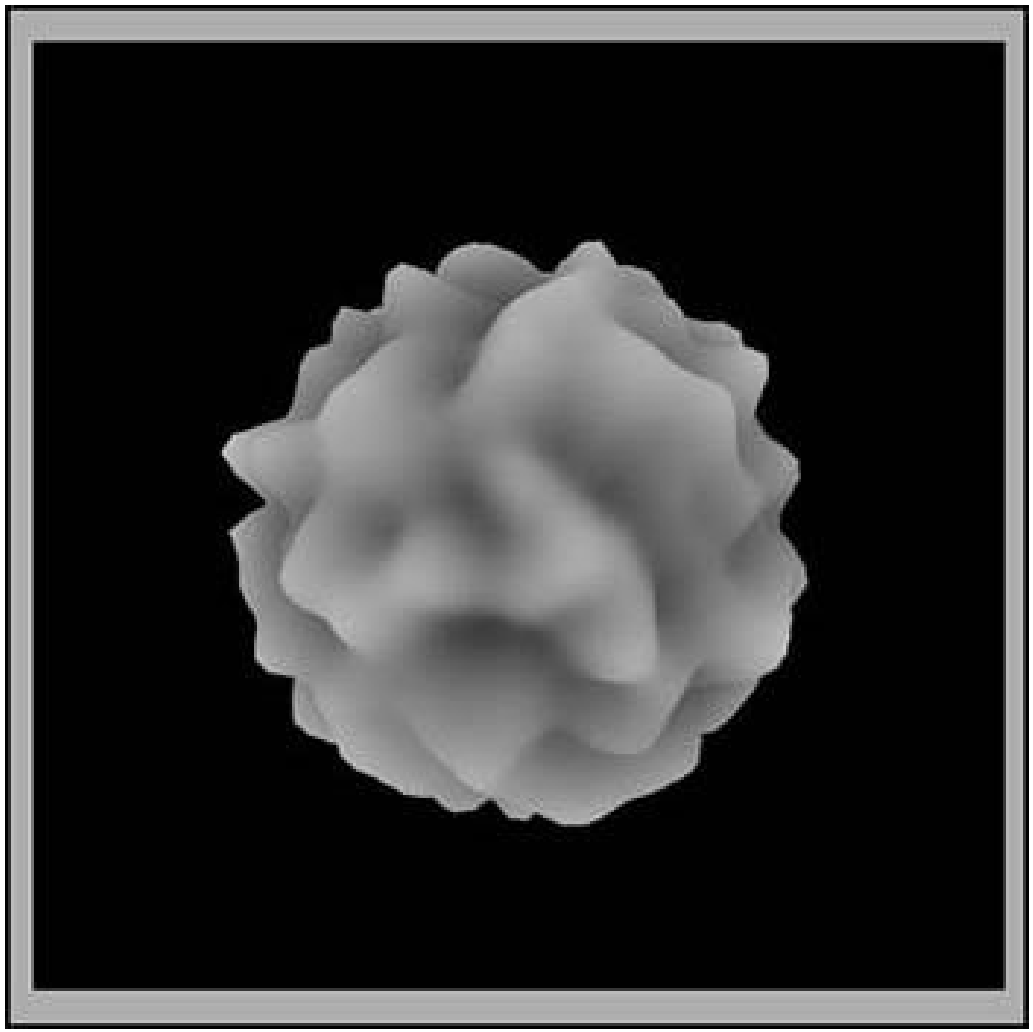
Procedural Terrain



From NVIDIA demo

Procedural Shading

Vertex Noise



From NVIDIA demo

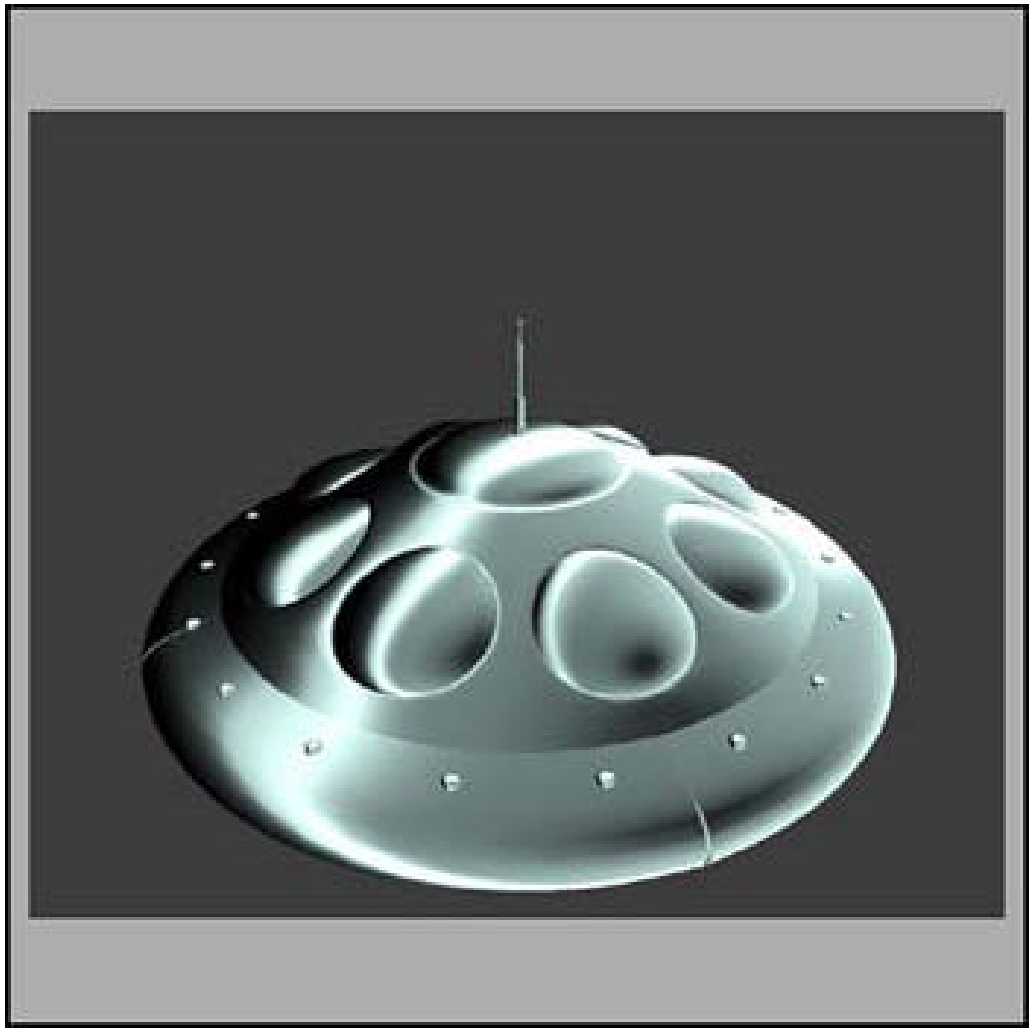
Procedural Shading

Refractive Dispersion

From NVIDIA demo

Procedural Shading

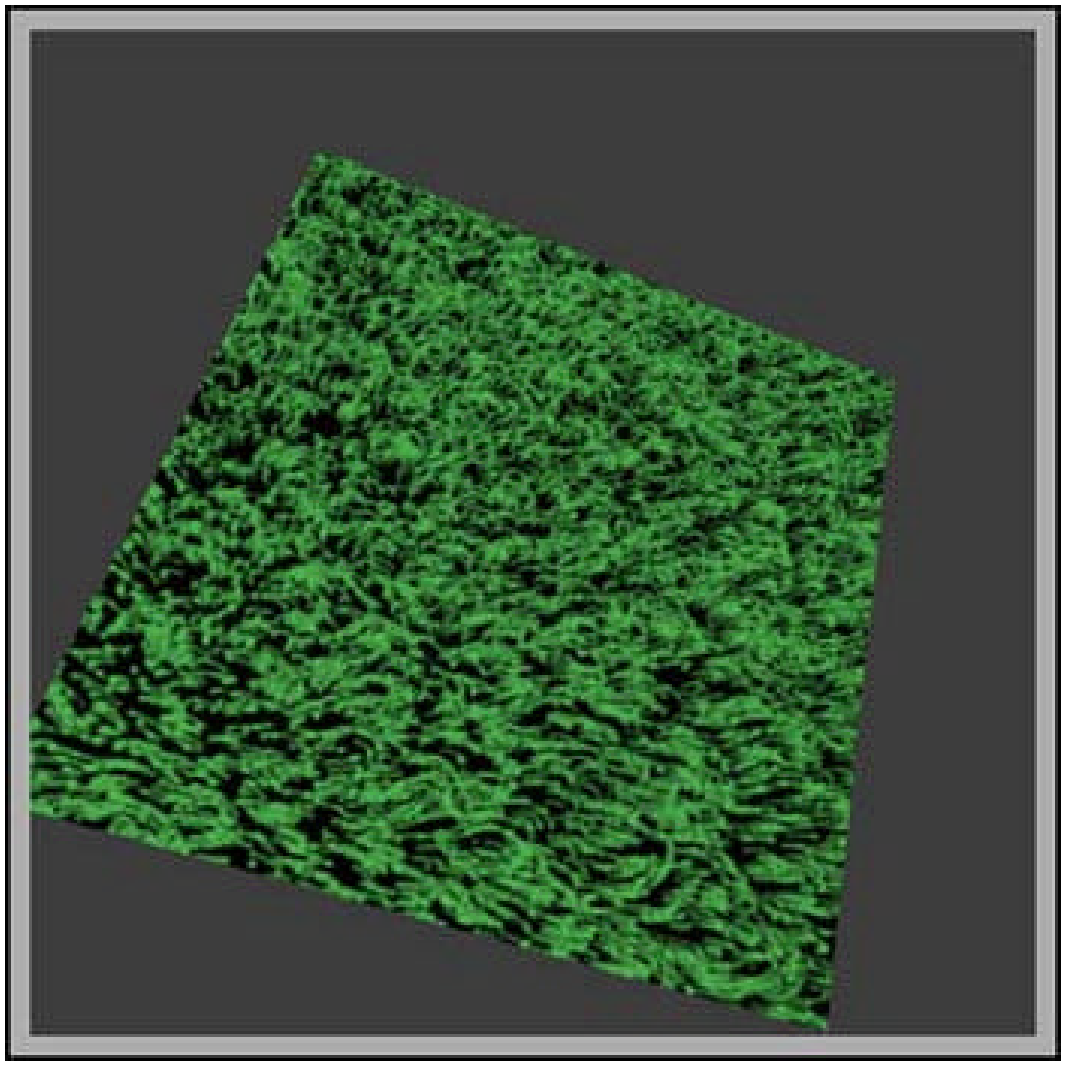
Anisotropic Lighting



From NVIDIA demo

Procedural Shading

Grass



From NVIDIA demo

Procedural Shading

Fur

[fur]

Questions?

References

- [cook] Cook, R. L. “Shade Trees.” In *Computer Graphics (Proc. SIGGRAPH ’84)* 18(3): 223-231(1984)
- [pis] Perlin, K. “An image synthesizer.” In *Computer Graphics (Proc. SIGGRAPH ’85* 19(3): 287296(1985)
- [fur] Lengyel, J. et al. “Real-Time Fur over Arbitrary Surfaces” 2001