

Shaders & Transformation



MIT EECS 6.837, Durand Lots of slides from Ken Perlin 1

C++ compilation

- Compile:
 - Each .cc is compiled independently
 - Only link to other files is their included .h
 - This is why .h tells the compiler
 - available methods & their signature
 - memory needs: fields, including private ones
- Link
 - Put everything together
- Advantage: reduces dependencies

MIT EECS 6.837, Durand

2

Ray tracing class design?

MIT EECS 6.837, Durand

3

Questions?

MIT EECS 6.837, Durand

4

BRDFs in the movie industry

- <http://www.virtualcinematography.org/publications/acrobat/BRDF-s2003.pdf>
- For the Matrix movies
- Clothes of the agent Smith are CG, with measured BRDF

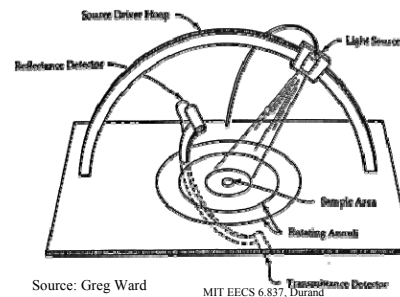


MIT EECS 6.837, Durand

5

How do we obtain BRDFs?

- Gonioreflectometer
 - 4 degrees of freedom



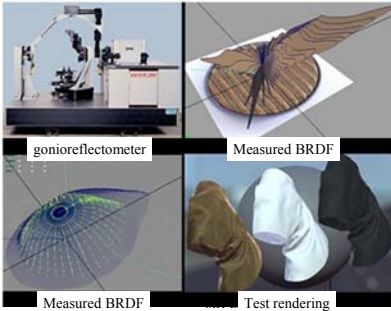
Source: Greg Ward

MIT EECS 6.837, Durand

6

BRDFs in the movie industry

- <http://www.virtualcinematography.org/publications/acrobat/BRDF-s2003.pdf>
- For the Matrix movies



7



Photo CG Photo CG

MIT EECS 6.837, Durand

8

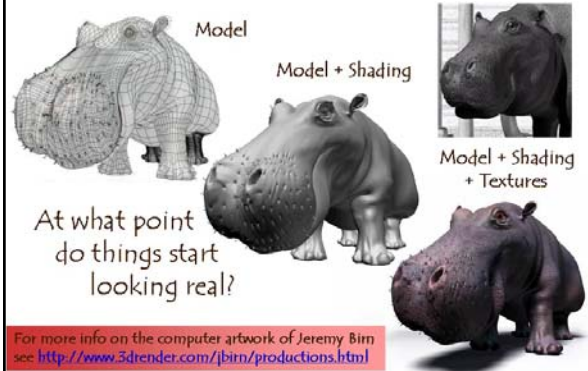
Questions?



MIT EECS 6.837, Durand

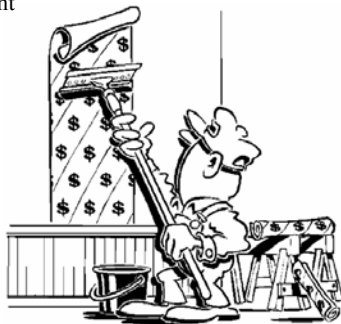
9

The Quest for Visual Realism



Texture Mapping

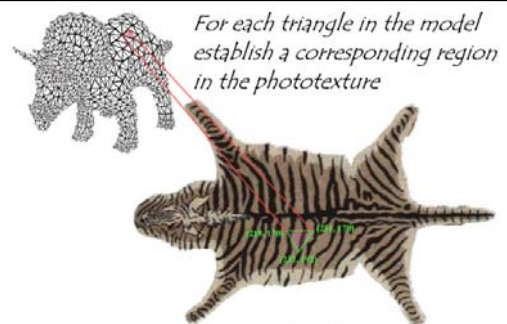
- Increase the apparent complexity of simple geometry
- Like wallpapering or gift-wrapping with stretchy paper
- Curved surfaces require extra stretching or even cutting



MIT EECS 6.837, Durand

11

Photo-textures

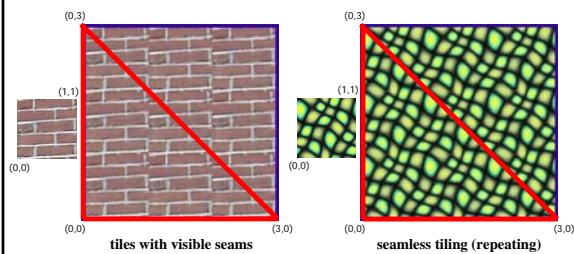


MIT EECS 6.837, Durand

12

Texture Tiling

- Specify a texture coordinate (u,v) at each vertex
- Canonical texture coordinates $(0,0) \rightarrow (1,1)$

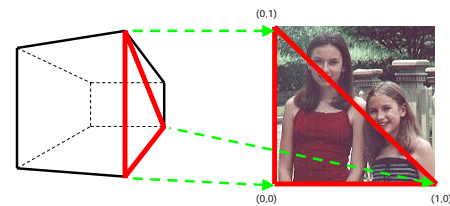


MIT EECS 6.837, Durand

13

Texture Interpolation

- Specify a texture coordinate (u,v) at each vertex
- Interpolate texture coordinates with barycentric coordinates

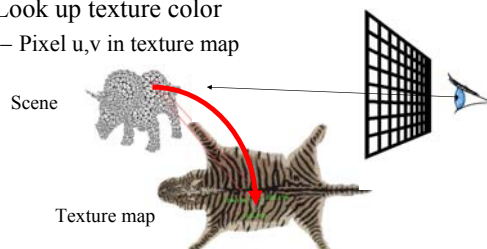


MIT EECS 6.837, Durand

14

Recap

- Ray cast, get visible point
- Get texture coordinates (u, v) at that point
 - e.g. barycentric coordinates
- Look up texture color
 - Pixel u,v in texture map



15

Problem

- For non-triangle geometry, not per vertex uv
- Solution: deduce uv from x,y,z
 - Various mappings are possible

MIT EECS 6.837, Durand

16

Common Texture Coordinate Mappings

- Orthogonal
- Cylindrical
- Spherical
- Perspective Projection
- Texture Chart

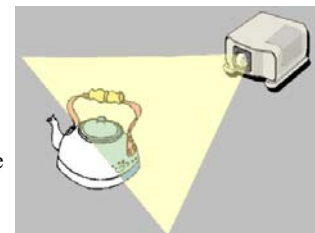


MIT EECS 6.837, Durand

17

Projective Textures

- Use the texture like a slide projector
- No need to specify texture coordinates explicitly
- A good model for shading variations due to illumination
- A fair model for reflectance (can use pictures)



MIT EECS 6.837, Durand

18

Projective Texture Example

- Modeling from photographs
- Using input photos as textures

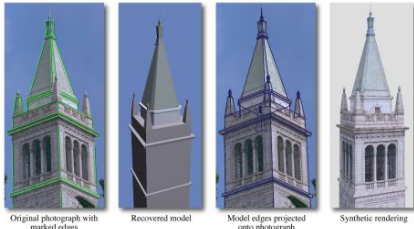


Figure from Debevec, Taylor & Malik
<http://www.debevec.org/Research>

19

Texture Mapping & Illumination

- Texture mapping can be used to alter some or all of the constants in the illumination equation:
 - pixel color, diffuse color, alter the normal,

$$I_{\text{total}} = k_r I_{\text{ambient}} + \sum_{i=1}^{N_{\text{lights}}} I_i (k_d (\hat{N} \cdot \hat{L}) + k_s (\hat{V} \cdot \hat{R})^{\alpha})$$

Phong's Illumination Model



Constant Diffuse Color



Diffuse Texture Color



Texture used as Label



Texture used as Diffuse Color

MIT EECS 6.837, Durand

20

Questions?

MIT EECS 6.837, Durand

21

Shaders (Material class)

- Functions executed when light interacts with a surface
- Constructor:
 - set shader parameters
- Inputs:
 - Incident radiance
 - Incident & reflected light directions
 - surface tangent (anisotropic shaders only)
- Output:
 - Reflected radiance

MIT EECS 6.837, Durand

22

Shader

- Initially for production (slow) rendering
 - Renderman in particular
- Now used for real-time (Games)
 - Evaluated by graphics hardware
 - More later in the course

MIT EECS 6.837, Durand

23

Questions?

MIT EECS 6.837, Durand

24

Procedural Textures

$f(x,y,z) \rightarrow \text{color}$

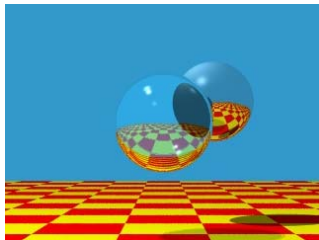


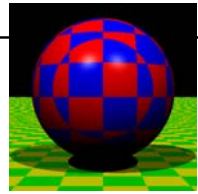
Image by Turner Whitted

MIT EECS 6.837, Durand

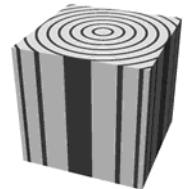
25

Procedural Textures

- Advantages:
 - easy to implement in ray tracer
 - more compact than texture maps (especially for solid textures)
 - infinite resolution



- Disadvantages
 - non-intuitive
 - difficult to match existing texture

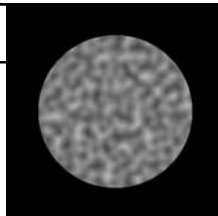


MIT EECS 6.837, Durand

26

Perlin noise

- Pseudo-random function
 - But continuous
 - band pass
- Useful to add lots of visual detail



<http://www.noisemachine.com/talk1/index.html>
<http://mrl.nyu.edu/~perlin/doc/oscar.html>
<http://mrl.nyu.edu/~perlin/noise/>
http://en.wikipedia.org/wiki/Perlin_noise
http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
(not really Perlin noise but very good)
<http://portal.acm.org/citation.cfm?id=325247>

Ken Perlin

MIT EECS 6.837, Durand

27

Requirements

- Pseudo random
- For arbitrary dimension
 - 4D is common for animation
- Smooth
- Band pass
- Little memory usage

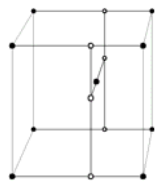
- How would you do it?

MIT EECS 6.837, Durand

28

Perlin noise

- Cubic lattice with pseudo gradient at vertices
- Set vertex values to zero
 - To avoid low frequencies
- Splines to interpolate the values



MIT EECS 6.837, Durand

29

Algorithm

- Given an input point
- For each of its neighboring grid points:
 - Pick a "pseudo-random" gradient vector
 - Compute linear function (dot product)
- Take weighted sum, using ease curves
 - [demo](#)

MIT EECS 6.837, Durand

30

Computing the pseudo-random gradient

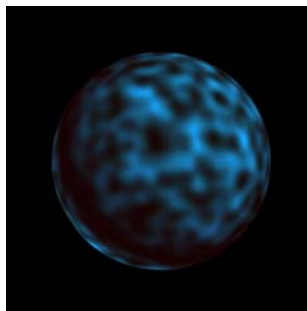
- Precompute table of permutations $P[n]$
- Precompute table of gradients $G[n]$
- $G = G[(i + P[(j + P[k]) \bmod n]) \bmod n]$

- in practice only 256 gradients are stored!
 - But optimized so that they are well distributed

MIT EECS 6.837, Durand

31

Noise



MIT EECS 6.837, Durand

32

Sum $1/f$ noise

- That is, each octave f has weight $1/f$



MIT EECS 6.837, Durand

33

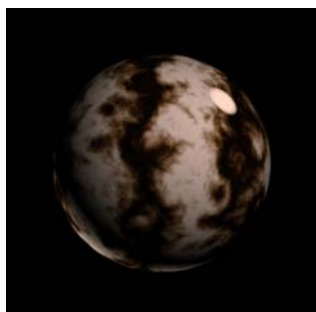
Sum $1/f$ |noise|



MIT EECS 6.837, Durand

34

$\sin(x + \text{sum } 1/f \text{ |noise|})$

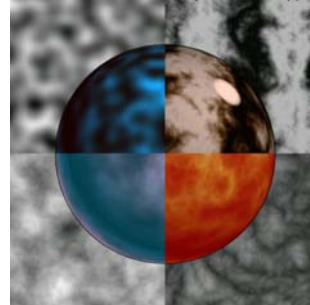


MIT EECS 6.837, Durand

35

Comparison

•noise $\sin(x + \text{sum } 1/f \text{ |noise|})$



sum $1/f(\text{noise})$ sum $1/f(\text{|noise|})$

MIT EECS 6.837, Durand

36

Question?

MIT EECS 6.837, Durand

37

Noise for solid textures

- Wood
 - color = color map (r+noise)
 - <http://www.connectedpixel.com/blog/texture/wood>
- Marble
 - parallel plane + noise
 - Color = color map (x + turbulence)
 - <http://legakis.net/justin/MarbleApplet/>



<http://student.kuleuven.be/~m0216922/CG/perlinnoise.html>



MIT EECS 6.837, Durand

38

Corona

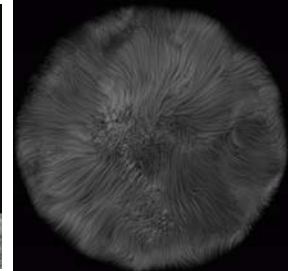
- The corona was made as follows:
 - Create a smooth gradient function the drops off radially from bright yellow to dark red.
 - Phase shift this function by adding a turbulence texture to its domain.
 - Place a black cutout disk over the image.
- Animation
 - Scale up over time
 - Use higher dim noise (for time)
 - <http://www.noisemachine.com/talk1/imgs/flame500.html>



MIT EECS 6.837, Durand

Slides by Ken Perlin

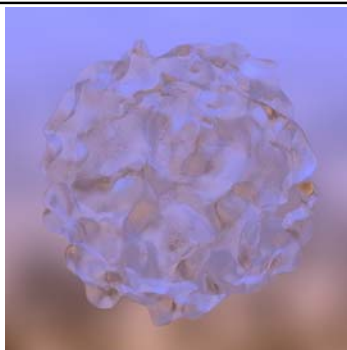
Other cool usage: displacement, fur



MIT EECS 6.837, Durand

40

Question?



MIT EECS 6.837, Durand

41

Shaders

- Noise: one ingredient of shaders
- Shaders control diffuse color, but also specular components, maybe even roughness (exponent), transparency, etc.
- Shaders can be layered (e.g. a layer of dust, peeling paint, mortar between bricks).
- Notion of shade tree
 - Pretty much algebraic tree
- Assignment 5: checkerboard shader based on two shaders

MIT EECS 6.837, Durand

42

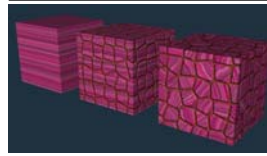
Bottom line

- Programmable shader provide great flexibility
- Shaders can be extremely complex
 - 10,000 lines of code!
- Writing shaders is a black art

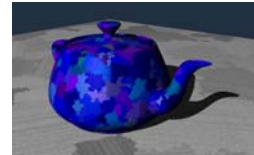
MIT EECS 6.837, Durand

43

Questions?



Justin Legakis



MIT EECS 6.837, Durand

Justin Legakis

44