CS 543 Computer Graphics

Texture Mapping Effects

by Cliff Lindsay

"*Top Ten List*" Courtesy of David Letterman's Late Show and CBS

Talk Format

List of 

Texture Mapping Effects from Good to Spectacular (my biased opinion):

Highlights:

- Define Each Effect
- Describe Each Effect Briefly: Theory and Practice.
- Talk about how each effect extends the idea of general Texture Mapping (previous talk) including Pros and Cons.
- Demos of selected Texture Mapping Effects

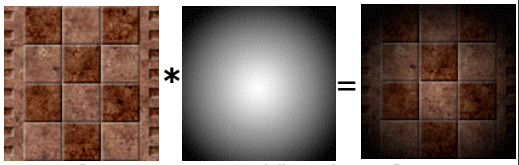
Texture Mapping Effect #10

Light Mapping

Main idea: Static diffuse lighting contribution for a surface can be captured in a texture and blended with another texture representing surface detail.

Highlights:

- Eliminate lighting calculation overhead
- Light maps are low resolution
- Light maps can be applied to multiple textures



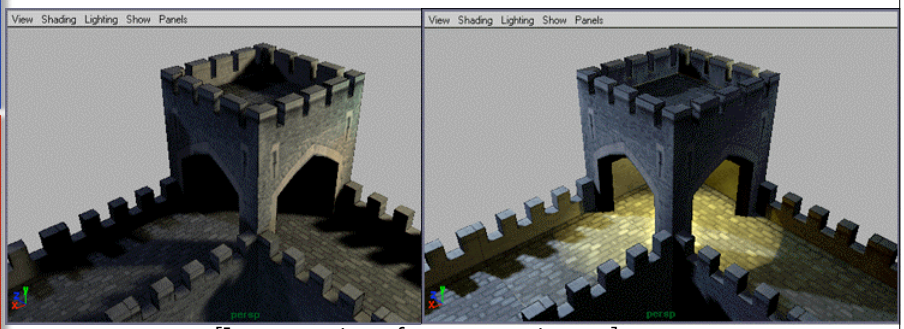
[Images courtesy of flipcode.com]

Light Mapping

Below is a night scene of a castle. No lighting calculation is being performed at all in the scene.

Left: No Light Map applied

Right: Light Map applied



[Images courtesy of www.gamasutra.com]

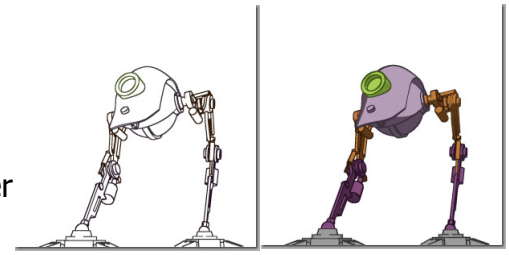
Texture Mapping Effect #9

Non-Photorealistic Rendering

Main idea: Recreating an environment that is focused on depicting a style or communicating a motif as effectively as possible. This is in contrast to Photorealistic which tries to create as real a scene as possible.

High lights of NPR:

- Toon Shading
- Artistic styles (ink, water color, etc.)
- Perceptual rendering

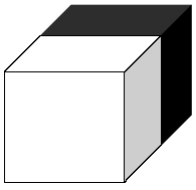


[Robo Model with and without Toon Shading, Image courtesy of Michael Arias]

Non-Photorealistic Rendering

Simple Example: Black Outline*

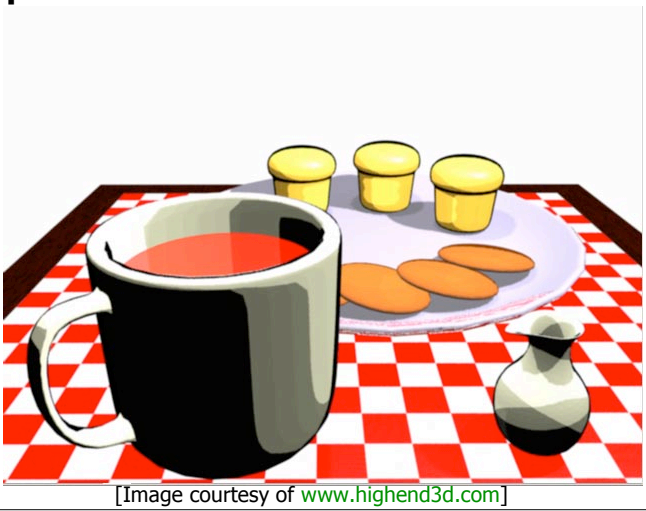
- Store a black stripe in a cube map at the 90 degree angle
- By using the vertex normal in view space to index the Cube map, you get a black outline



*Many other methods exist that don't use texture hardware

Non-Photorealistic Rendering

Visual Example:



[Image courtesy of www.highend3d.com]

Procedural Texturing

Main Idea: "Procedurally or analytically varying the surface properties from point to point in order to give the appearance of surface detail that is not actually present in the geometry of the surface."

[adapted from ptm:apa, pg. 7]

Highlights:

- Noise (fire, smoke)
- Fractals (terrain)
- Analytical (marble, wood)
- Grammars (trees, leaves)

[Blue Marble Procedural Texture, tam:apa]

Procedural Texture Mapping

Noise Functions: Perlin noise (most popular function) produces noise with the desirable property that the transition from one point to another within the function is a smooth one.

<u>Highlights:</u>

- Psuedo-random number generation with repeatability in R^3
- Smooth
- Band-limited (low-pass filter), i.e. rolling hills vs. sharp peaks.

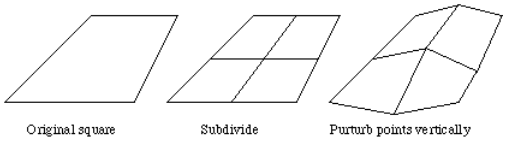
[Vase with Perlin noise, Image courtesy of graphics.lcs.mit.edu]

Procedural Texture Mapping

Fractals (statistical self-similarity): "A complex object, the complexity of which arises from the repetition of a given shape at a variety of scales."
[tam:apa, pg. 571]

Fractal Terrain Generation (Basic idea):

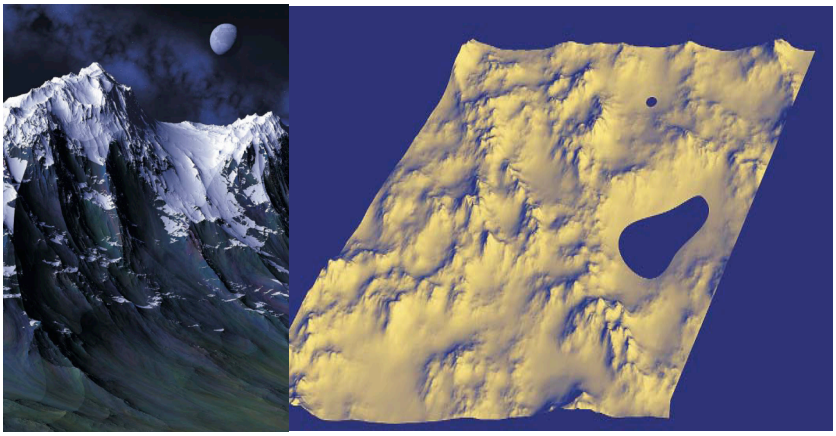
- Start with a course model (square)
- Subdivision of surfaces (2x2)
- Vertically perturb each of the 5 new vertices by a random amount
- Repeat until done



[image courtesy of Paul Bourke @ astronomy.swin.edu.au]

Procedural Texture Mapping

Fractal Examples:



[Right: Ridged multi-fractal, Left: A Fractal Generated Terrain, *Texturing and Modeling: A Procedural Approach*]

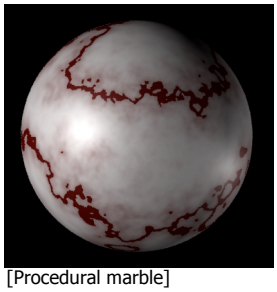
Procedural Texture Mapping

Analytical Examples:

Marble

$$f(s,t,r) = r + \sum_{i=0}^{N} 2^{-i}\, noise(2^i s, 2^i t, 2^i r)$$

*__i__ continually increases the noise amplitude animates the formation of the veins

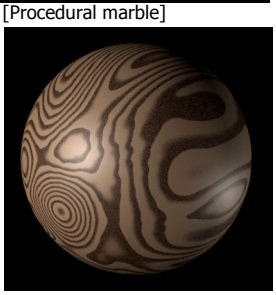


[Procedural marble]

Wood - Vertical cylinders

$$f(s,t,r) = s^2 + t^2 + noise(4s, 4t, r)$$

*__s,t,r__ are solid texture coordinates, these are used to do a color map look up for the texture.



[Procedural wood, both courtesy of http://renderman.ru]

Texture Mapping Effect #8

Pros:

- Memory requirement is minimal, procedural or analytical representation is very compact
- No fixed resolution, infinite zoom-in and zoom-out
- Occupy infinite space or area

Cons:

- Difficult to build and debug
- Slower than texture fetching
- Antialiasing can be more difficult than a regular texture
- Procedures tend to be problem specific (fractal terrain, etc.), not generalized

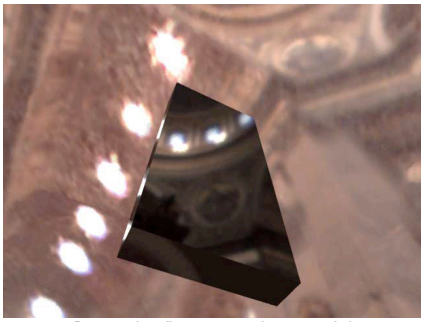
Texture Mapping Effect #7

High Dynamic Range Texture Mapping

Main idea: "Visualizing HDR image-based scenes in graphics hardware without compressing the dynamic range." [HDRTM]

Highlights:

- Using texture hardware for HDR rendering
- HDR Texure comprised of multiple regular textures
- Multi-texturing support to combine textures

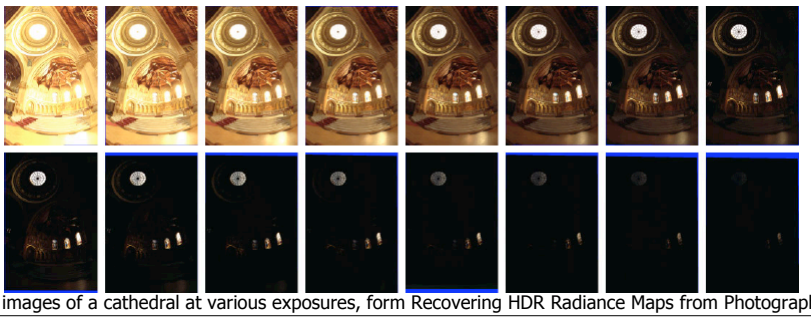


[Fresnel reflection on the monolith in cathedral, image from HDRTM]

High Dynamic Range Texture Mapping

What is dynamic range?:

- Measure of luminance ranges
- Sun on the order 1 Million Lumens
- TV on the order of a 100-200 Lumens



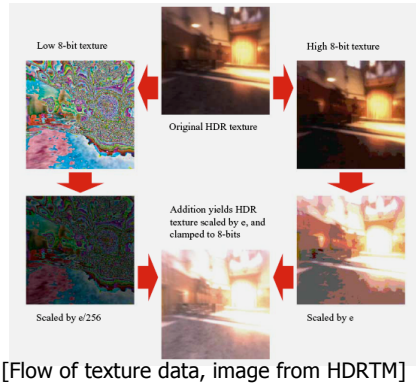
[16 images of a cathedral at various exposures, form Recovering HDR Radiance Maps from Photographs]

High Dynamic Range Texture Mapping

How to encode HDR in 8 bit textures:

$$I(v) = clamp(clamp(e * v_0) + clamp(256 * e * v_1))$$

- The low order bits are stored in texture v0 and the high order are stored in texture v1
- Exposure parameter is e
- Surface appearance is I



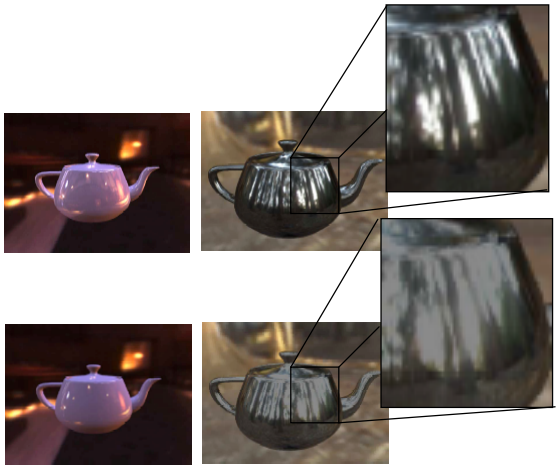
[Flow of texture data, image from HDRTM]

HDR Texture Mapping

Visual Differences Between LDR and HDR

High Dynamic Range

Low Dynamic Range



[Images from HDRTM]

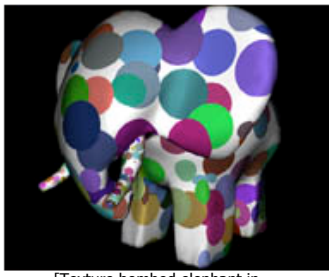
Texture Mapping Effect #6

Texture Bombing

Main idea: Divide the UV texture space into grids or cells then randomly place an image within selected cells giving the texture a collage look.

Highlights:

- Compositing can be with images or done procedurally
- Multiple images per cell (overlapping images)
- Many ways to pick cells

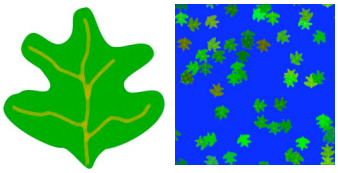


[Texture bombed elephant in Rendermonkey, courtesy of ATI]

Texture Bombing

Technique:

- Find a cell to place image (usually a random process)
- Copy or draw the image in the Cell procedurally
- Consider adjacent cells (overlap?)



[Texture bombing Images, courtesy of www.webnation.com]

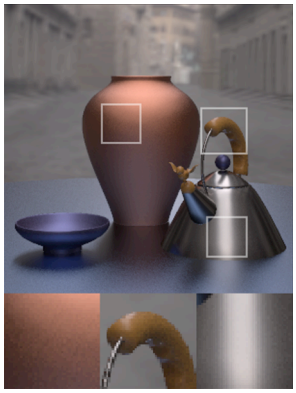
***Placement is usually important depending on effect**

Texture Shading

Main Idea: Texture Shading precomputes complex surface and lighting models, such as BRDFs, into a lookup texture for real-time applications with fixed pipelines.

High lights:

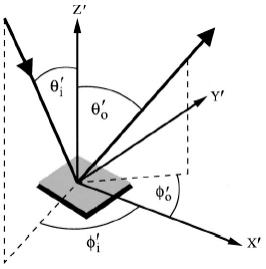
- Complex BRDFs are usually precomputed in this fashion
- Multi-pass rendering or multi-fetch texture lookups
- Factorization

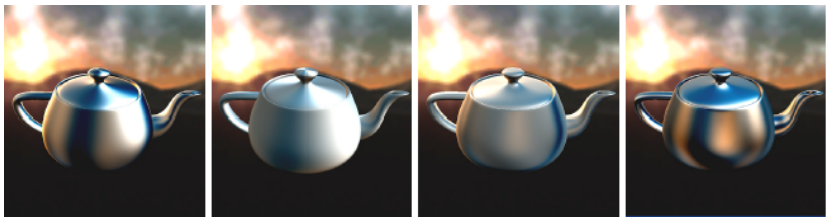


[Factored BRDFs: LaFortune model, Images from Siggraph Paper (see references)]

Precomputing

- Most BRDFs can be factored or broken up with the parts being factorable.
- Factor over 2 variables: θ, ϕ





A) Ashikhmin-Shirley B) Poulin-Fournier C) Vinyl (measured) D) Alum. Foil (measured)

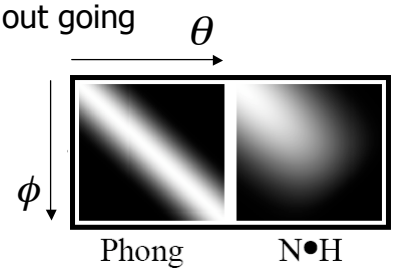
Texture reference (precompute & run time)

Precompute:

- Increment through θ, ϕ storing the evaluated/measured values in the appropriate texture coordinate

Run Time:

- Calculate the incoming and out going vector to get θ, ϕ
- Index into texture per



[Precomputed reflectance textures, Frequency Environment Mapping]

More Examples

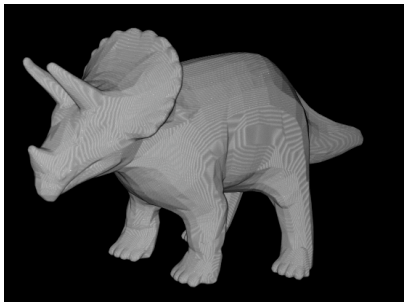
A) Ashikhmin-Shirley B) Poulin-Fournier C) Vinyl (measured) D) Alum. Foil (measured)

Volume Rendering

Main idea: "Volume rendering methods generate images of a 3D volumetric data set without explicitly extracting geometric surfaces from the data" [gpugems]

Highlights:

- Stacks of 2D texture slices
- Voxel (analogus to pixel, texel)
- Reconstruction (interpolation between voxels)



[Screen shot from Texture Visualizer Software]

Volume Rendering Simple Example

Condensed Steps For Rendering:

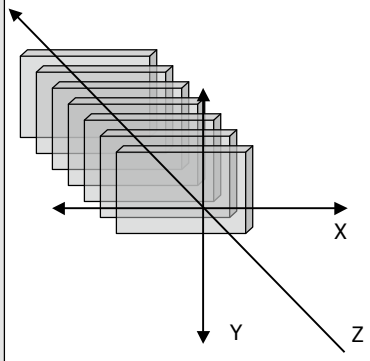
1. Set up texture data, fragment program, Modelview and Projection matrices.
2. Enable Alpha blending.

   glEnable(GL_BLEND);

   glBlendFunc(GL_ONE , GL_ONE_MINUS_SRC_ALPHA);
3. Disable lights and depth test.

   glDisable(GL_LIGHTING);

   glDisable(GL_DEPTH_TEST);
4. Bind texture data and fragment program.
5. Draw textured quads.



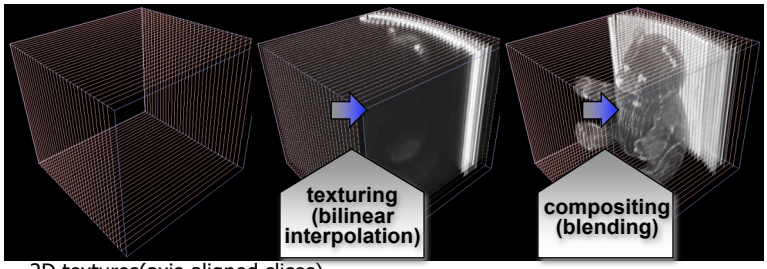
[steps provided in GPU Gems, ch. 39.3.1, pg 673]

Cg Fragment Program For Simple Volume Rendering

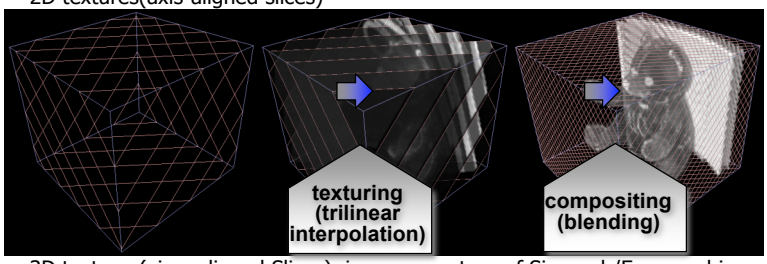
```
void main (
        uniform float3 emissivecolor,
        uniform sampler3d dataTex,
        float3 texCoord : TEXCOORD0,
        float4 color : COLOR)
{
         //read volume data
        float a = tex3d(texCoord, dataTex);

         //multiply color and opacaity
        color = a * emissiveColor;
}
```

[Cg code sample from GPU Gems, ch 39, pg. 673]

Visual Examples



2D textures(axis-aligned slices)



3D textures(view-aligned Slices), images courtesy of Siggraph/Eurographics

Texture Atlas

Main Idea: Store multiple smaller textures into a larger single texture.

Highlights:

- Preserve surface details
- Combine multiple textures
- Can even compute lighting!?

[Image courtesy of Cat Mother]

Texture Atlas

Used Extensively in Games:

- High resolution meshes generate more detail when shaded, we can preserve that detail and apply to lower resolutions meshes. This is great for game characters!
- An internal nVidia survey of four DirectX9 titles reveals that the following render-state changes occur most frequently [nsdk]:

- ***SetTexture() // bad news!!!***
- **SetVertexShaderConstantF()**
- **SetPixelShader()**
- **SetStreamSource()**
- **SetVertexDeclaration()**
- **SetIndices()**

[Images courtesy of Garage Games]

Texture Atlas

Pros:

- Conserve memory - texture memory is scarce even with today's graphics cards
- Reduce computation at run time
- Complexity reduction, model has one texture

Cons:

- Seams and texture pollution
- How do we layout the mesh (non-trivial)?
  - Need tools

Environment Mapping

Main idea: "Environment Maps are textures that describe for all directions the incoming or out going light at a point in space." [rt_shade, pg. 49]"

Three main types:

- Cube Mapping
- Sphere mapping
- Paraboloid Mapping

No Map applied Map Applied

[Images courtesy of Microsoft, msdn.microsoft..com]

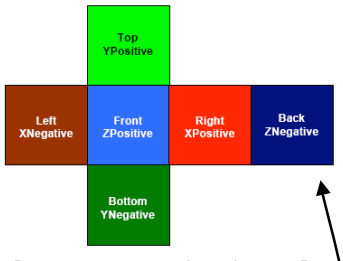
Cubic Mapping

- Camera takes orthographic pictures in six axis (-X,X, Y, -Y, Z, -Z)
- Look up is defined calculating a **reflection** vector

I.E.: R = (3.14, .21, -8.7)

X, Y, Z

Z is largest & negative

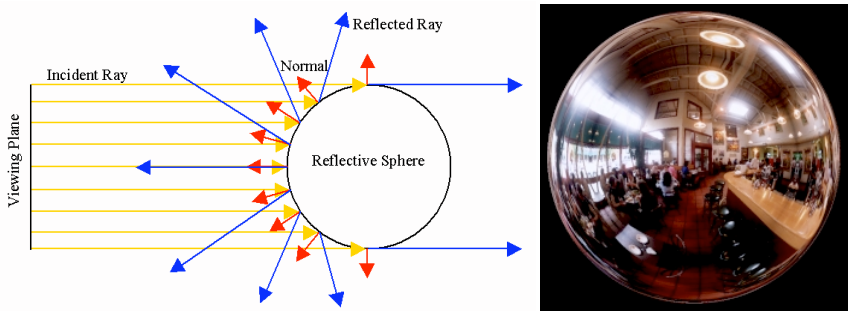


[image courtesy of nVidia.com]

* Index into the Negative Z region (dark blue)

Sphere Mapping

- Generated from photographing a reflective sphere
- Captures whole environment



[Diagram and Sphere Map image of a Cafe in Palo Alto, CA, Heidrich]

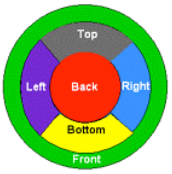
Sphere Mapping

- Obtain the reflection vector: $\vec{R} = \vec{I} - 2.0 \cdot \vec{N} \cdot (\vec{N} \bullet \vec{I})$

Index into the Sphere map:

$$s = \frac{R_x}{m} + \frac{1}{2}, \quad t = \frac{R_y}{m} + \frac{1}{2}$$

$$m = 2\sqrt{\left(R_x^2 + R_y^2 + \left(R_z + 1\right)^2\right)}$$



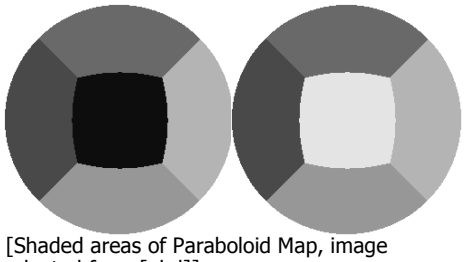
[image courtesy of nVidia.com]

Parabaloid Mapping

$$f(x,y) = \frac{1}{2} - \frac{1}{2}\left(x^2 + y^2\right)$$, where $x^2 + y^2 \leq 1$

High Lights:

- Two textures, one for each hemisphere
- No artifacts at poles
- Requires 2 passes or two texture fetches to render



[Shaded areas of Paraboloid Map, image adapted from [phd]]

Cons :

- Sphere maps have a singularity of the parameterization of this method, we must fix viewing direction, view-dependent (meaning if you want to change the viewers direction you have to regenerate the Sphere map).
- Paraboloid maps requires 2 passes

Pros:

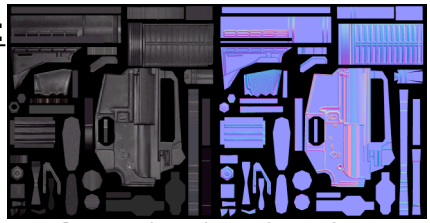
- Better sampling of the texture environment for Paraboloid mapping, view-independent,
- Cube maps can be fast if implemented in hardware (real-time generation), view independent,

Bump Mapping

Main idea: "Combines per-fragment lighting with surface normal perturbations supplied by a texture, in order to simulate light interactions on a bumpy surface." [Cg Tutorial, pg 199]"

Where Can I get these maps?:

- Normal Maps from Height Fields (most common)
- Vector Offset Maps



[texture atlas and normal map of an M16 rifle, images courtesy of cat mother]

Bump Map

$$P'(u,v) = P(u,v) + \vec{N}(u,v)F(u,v)*$$

- P = original Surface location/height
- N = Surface Normal
- F = Displacement Function
- P' = New Surface location/height

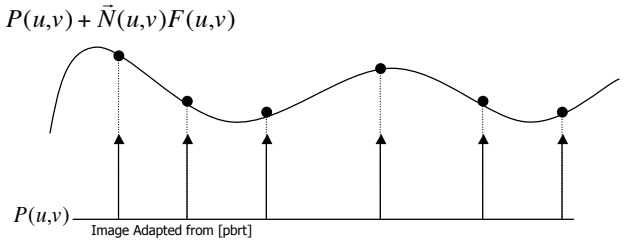


Image Adapted from [pbrt]

* Assumes $\vec{N}$ is normalized.

Bump Map

- The new Normal N' for P' can be calculated from the cross product of it's partial derivatives[Blinn].

***Differential Math*!!!**

$$\vec{N}' = \frac{\partial P'}{\partial u} \times \frac{\partial P'}{\partial v} \approx \vec{N} + \frac{\partial F}{\partial u}\left(\vec{N} \times \frac{\partial P}{\partial u}\right) + \frac{\partial F}{\partial v}\left(\vec{N} \times \frac{\partial P}{\partial v}\right)$$

Bump Maps

Pros:

- Produces the appearance of high detail w/ out the cost
- Can be done in hardware

Cons:

- No self shadowing (natively)
- Artifacts on the silhouettes

Cg Bump Mapping

```
float4 main(
        float2 detailCoords : TEXCOORD0,
        float2 bumpCoords: TEXCOORD1,
        float3 lightVector : COLOR0,
        uniform float3 ambientColor,
        uniform sampler2D detailTexture : TEXUNIT0,
        uniform sampler2D bumpTexture : TEXUNIT1): COLOR
{
        float3 detailColor = tex2D(detailTexture, detailCoords).rgb;
        // Uncompress vectors ([0, 1] -> [-1, 1])
        float3 lightVectorFinal = 2.0 * (lightVector.rgb - 0.5);

        float3 bumpNormalVectorFinal
                = 2.0 * (tex2D(bumpTexture, bumpCoords).rgb - 0.5);

        // Compute diffuse factor
        float diffuse = dot(bumpNormalVectorFinal, lightVectorFinal);
        return float4(diffuse * detailColor + ambientColor, 1.0); }
```


Some Other Texture Related Algorithms:

- Texture Mapping Hardware
- Anti-Aliasing
- 3D Texture Mapping
- Animated Textures
- Alpha Mapping
- Projective Textures
- Texture Animation

Summary

- As you can see from the slides, texture mapping goes beyond the general definition of texture mapping.
- All of these exciting advancements in texture mapping have come about from the hardware's companies desire to make texture mapping fast.
- The effects shown here are probably the most responsible for making real-time graphics look and feel more realistic.

References

#1 Bump Mapping:

- *A Practical and Robust Bump-mapping Technique for Today's GPUs*, Mark J. Kilgard, NVIDIA Corporation, GDC 2000: Advanced OpneGL Game Development.
- *Simulation of Wrinkled Surfaces*, Blinn, Jim, International Conference on Computer Graphics and Interactive Techniques, 1978
- The Cg Tutorial, Fernando, Randima, Kilgard, Mark, Addison-Wesley, 2003
- Cg Bump Mapping, Surdulescu, Razvan, http://www.gamedev.net/reference/articles/article1903.asp, 4/15/2003, Gamedev.net
- Course Slide from "Interactive Shading Course", Brad Grantham, Siggraph 1999, http://www.opengl.org/resources/tutorials/sig99/shading99/course_slides/Shading1/sld002.htm
- Physically Based Rendering, Pharr, Matt, Humphreys, Greg, Elsevier Inc./Morgan-Kaufman, 2004

References

#2 Environment Mapping

- Real-time Shading, Olano, Marc, Hart, C., John, Heidrich, Wolfgang, McCool, Michael,A.K. Peters, 2002.
- cube diagram from: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/directx/graphics/TutorialsAndSamples/Tutorials/HLSLWorkshop/EnvironmentMap2.asp
- Perfect Reflections and Specular Lighting Effects With Cube Environment Mapping, nVidia Technical Brief, www.nvidia.com, Geforce 256 GPU.
- Notes from opengl site http://www.opengl.org/resources/tutorials/sig99/advanced99/notes/node183.html
- High-quality Shading and Lighting for Hardware-accelerated Rendering, heidrich, Wolfgang, PhD thesis, University of Erlangen-Niirnberg, April 1999

References

#3 Texture Atlas

- Improve Batching Using Texture Atlases - Nvidia SDK whitepaper.
- Garage Games, Torque Game Engine.
- Real-time Shading, Olano, Marc, Hart, C., John, Heidrich, Wolfgang, McCool, Michael,A.K. Peters, 2002.

#4 Volume Rendering

- Klaus Engel, Martin Kraus, Thomas Ertl, Siggraph/Eurographics Workshop on Graphics Hardware, 2001, Power Point Slides
- GPU Gems, Chapter 39 Volume Rendering Techniques, Fernando, Randima, Addison-wesley and nVidia, 2004.
- Texture Visualizer Software, Wei Li, Stony Brook University, http://www.cs.sunysb.edu/.../ TextureVisualizer.htm

References

#5 Texture Shading

- Efficient BRDF Importance Sampling Using a Factored Representation, Jason Lawrence, Szymon Rusinkiewicz, Ravi Ramamoorthi, Siggraph, 2004
- Frequency Space Environment Map Rendering, Ravi Ramamoorthi, Pat Hanrahan, Siggraph 2002
- Real-time Shading, Olano, Marc, Hart, C., John, Heidrich, Wolfgang, McCool, Michael,A.K. Peters, 2002

#6 Texture Bombing

- Texture bombing, Chapter 20 of GPU Gems, Fernando, Randima, Addison-wesley and nVidia, 2004, pg. 323

#7 High Dynamic Range Texture Mapping

- Real-time HDR Texture Mapping, Jonathan Cohen, Chris Tchou, Tim Hawkins, and Paul Debevec, Eurographics wrkshop on rendering, 2001
- Recovering High Dynamic Range Radiance Maps from Photographs, Paul E. Debevec Jitendra Malik, Siggraph 1997

References

#8 Procedural Texturing

- Texturing and Modeling: A procedural Approach, third addition, Ebert, David, Musgrave, F, Peachey, Darwyn, Perlin, Ken, Worley, Steven, Morgan Kaufman, 2003
- Real-time Shading, Olano, Marc, Hart, C., John, Heidrich, Wolfgang, McCool, Michael,A.K. Peters, 2002.

#9 Non-Photorealistic Rendering

- The Cg Tutorial, Fernando, Randima, Kilgard, Mark, Addison-Wesley, 2003
- Non-Photorealistic Computer Graphics, Thomas Strothotte, Stefan Schlechtweg, Morgan Kaufman 2002
- GPU Toon Shading, D. Sim Dietrich Jr., nVidia Inc., GDC Talk 2002

#10 Light Mapping

- Notes from opengl.org tutorials/advanced/advanced98/notes/node103.html
- Real-time Rendering 2nd Edition, tomas Akenine-Moller, Eric haines, A.K. Peters, 2002
- Various Articles, http://www.gamasutra.com