Computer Graphics (CS 543)
Lecture 10: Bump Mapping, Parallax, Relief, Alpha, Specular Mapping

Prof Emmanuel Agu

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
Bump Mapping
Bump mapping: examples
Bump mapping

- by Blinn in 1978
- Inexpensive way of simulating wrinkles and bumps on geometry
  - Too expensive to model these geometrically
- Instead let a texture modify the normal at each pixel, and then use this normal to compute lighting

Bump map
Stores heights: can derive normals

Bump mapped geometry
Use normals of bumpy geometry
Bump mapping: Blinn’s method

- **Idea:** Distort the surface normal at point to be rendered.
- **Option a:** Modify normal $\mathbf{n}$ along $u$, $v$ axes to give $\mathbf{n}'$
  - In texture map, store how much to perturb $\mathbf{n}$ ($\mathbf{b}_u$ and $\mathbf{b}_v$)
- **Using bumpmap**
  - Look up $\mathbf{b}_u$ and $\mathbf{b}_v$
  - $\mathbf{n}' = \mathbf{n} + \mathbf{b}_u \mathbf{T} + \mathbf{b}_v \mathbf{B}$
  
    ($\mathbf{T}$ and $\mathbf{B}$ are tangent and bi-tangent vectors)

- **Note:** $\mathbf{N}'$ is not normalized.

- Bump map code similar to normal map code.
- Just compute, use $\mathbf{n}'$ instead of $\mathbf{n}$. 
Bump mapping: Blinn’s method

- **Option b:** Store values of $u$, $v$ as a heightfield
  - Slope of consecutive columns determines how much changes $n$ along $u$
  - Slope of consecutive rows determines how much changes $n$ along $v$

- **Option c (Angel textbook):** Encode using differential equations
Bump Mapping Vs Normal Mapping

- **Bump mapping**
- (Normals $\mathbf{n} = (n_x, n_y, n_z)$ stored as *local distortion of face orientation*. Same bump map can be tiled/repeated and reused for many faces)

- **Normal mapping**
- Coordinates of normal (relative to tangent space) are encoded in color channels
- Normals stored combines face orientation + plus distortion.

---

Vertex normals

Triangle

Disturbed normals

Bump map

![Bump Mapping Vs Normal Mapping Diagram](image)
Displacement Mapping

- Uses a map to displace the surface at each position
- Offsets the position per pixel or per vertex
  - Offsetting per vertex is easy in vertex shader
  - Offsetting per pixel is architecturally hard
Parallax Mapping

- Bump and normal maps increase surface detail, but do not simulate:
  - Parallax effects: Slanting of texture with view angle
  - Blockage of one part of surface by another part

- Parallax mapping
  - simulates parallax effects
  - Looks up a texture location offset depending on view angle
  - Different texture returned after offset
Relief (or Parallax Occlusion) Mapping

- Parallax mapping approximates parallax
- Sometimes doesn’t work well for occlusion effects
- Implement a heightfield raytracer in a shader, detect blockage
- Pretty expensive, but looks amazing
Relief Mapping Example

Cool YouTube Video: https://youtu.be/EkLKh5RzE-g
Light Mapping
Light Maps

- Good shadows are complicated and expensive
- If light and object positions do not change, shadows do not change
- Can “bake” the shadows into a texture map as a preprocess step
- During lighting, lightmap values are multiplied into resulting pixel

[Diagram showing the process of applying lightmap to diffuse texture]
Specular Mapping

- Store specular in a map
- Use greyscale texture to store specular component
Alpha Mapping

- RGBA: A or alpha is how transparent material is
  - 0 transparent, 1 opaque
- Represent the alpha channel with a texture
- Can give complex outlines, used for plants

---

RGB

Alpha

Render Bush on 1 polygon  

+  

=  

Render Bush on polygon rotated 90 degrees
Alpha Mapping

- Rotation trick works at eye level (left image)
- Breaks down from above (right image)
Mesh Parametrization
Mesh Parametrization

- The concept is very simple: define a mapping from the surface to the plane

For each triangle in the model establish a corresponding region in the phototexture
Parametrization in Practice

- Texture creation and parametrization is an art form
- Option: Unfold the surface
Parametrization in Practice

- Option: Create a Texture Atlas
- Break large mesh into smaller pieces
References

- Real Time Rendering by Akenine-Moller, Haines and Hoffman