



*n*VIDIA™

## **Per-Pixel Lighting Intro**

**Sim Dietrich**

**[sim.dietrich@nvidia.com](mailto:sim.dietrich@nvidia.com)**

# Per-Pixel Lighting

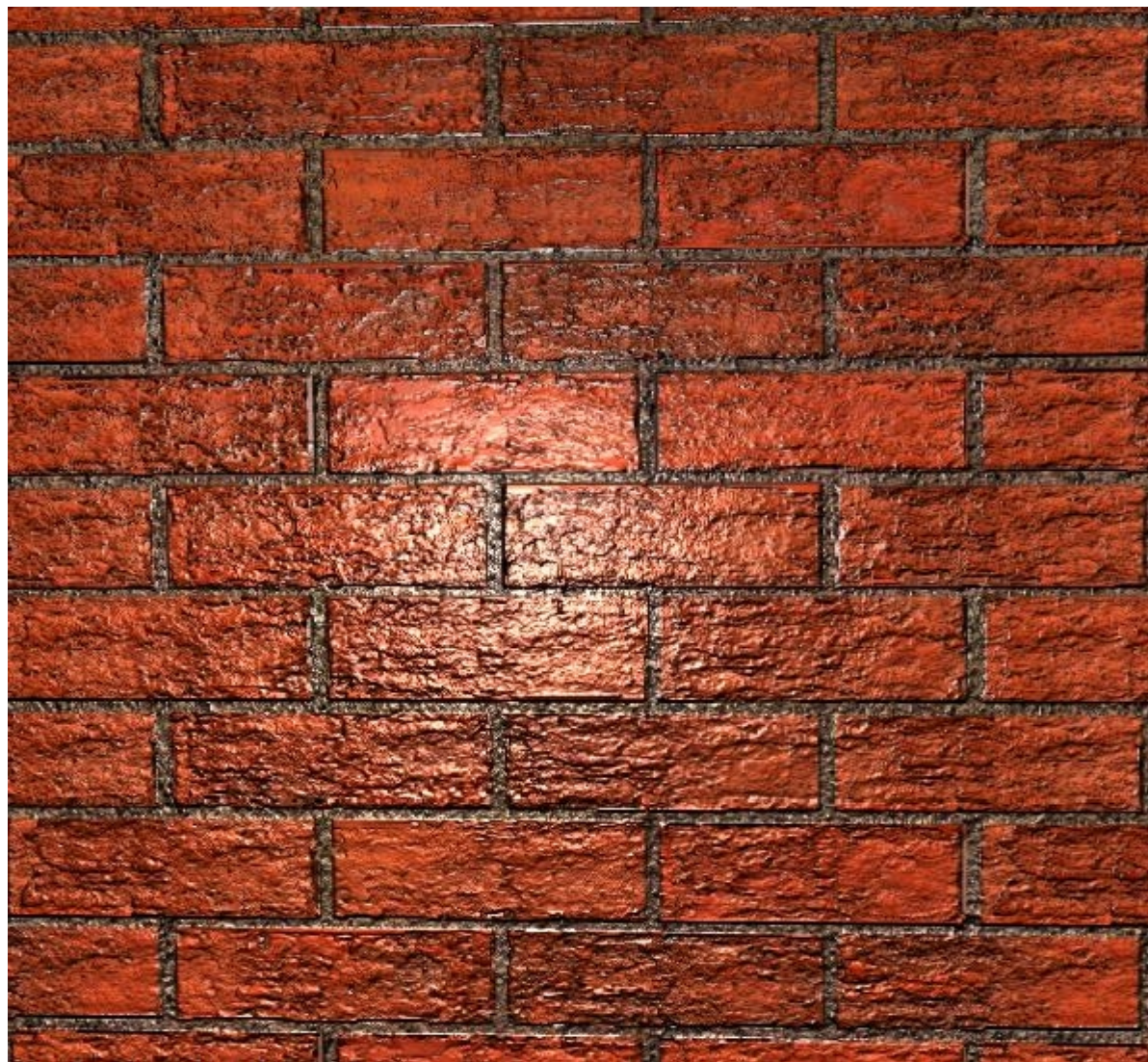
---

- **Per-Pixel lighting is the next leap in visual quality after simple multi-texturing**
- **It allows more apparent surface detail than would be possible with triangles alone**
- **Dx7 HW with DOT3 was a huge leap in per-pixel capability**
- **Dx8 HW increases performance again, and adds completely new capabilities**



*n*VIDIA™

# A Single Quad Lit Per Pixel



nVIDIA™

# Per Pixel Lighting / Bump Mapping

---

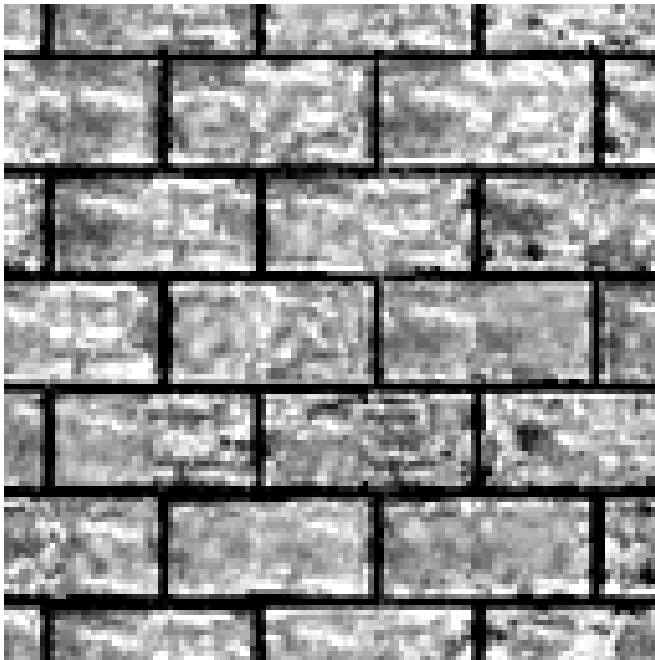
- **Bump Mapping is a subset of Per-Pixel Lighting**
- **These slides will discuss them interchangeably**
- **Most older Bump Mapping examples were simply performing diffuse directional lighting**
- **However, Bump Mapping / Per Pixel Lighting can be used to achieve diffuse and/or specular point lights, spotlights and volumetric lights as well**



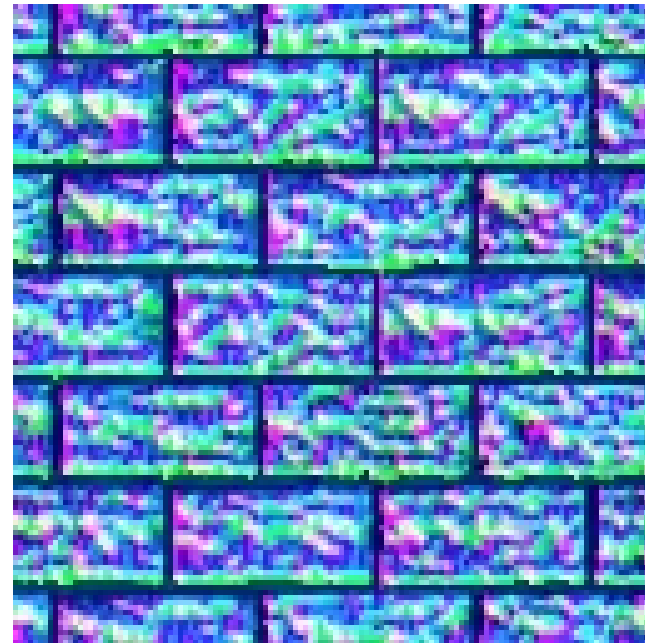
*n*VIDIA™

# Bump Maps == Normal Maps

- The type of Bump Map employed by our examples is also known as a Normal Map



Height Map



Normal Map



nVIDIA™



# Normal Maps

---

- Normal maps are simply RGBA textures where the R G and B *colors* in the range [0,1] are interpreted as  $\langle X, Y, Z \rangle$  *unit vectors* in the range [-1, 1 ]
- Each texel in the Normal Map texture is interpreted as the local unit surface normal



NVIDIA™

# Height Maps To Normal Maps

---

- To take advantage of either per-pixel lighting, artists should generate Height Maps
- A Height Map is a grayscale image, either stored in the RGB or Alpha channel
  - White indicates 'High'
  - Black indicates 'Low'
  - Gray values represent values in between
- For DOT3 techniques, including bump mapping and per-pixel lighting, this height map must be converted into a Normal Map



NVIDIA™

# Converting Height Maps To Normal Maps

---

- Use 3 adjacent height samples to form a triangle in model space at each texel
- Turn this triangle's normal into an RGB vector and store in a Normal Map Texture
- There is an arbitrary scale factor that determines the relative scale of the height dimension compared to the s & t texel dimensions
- Higher scale values create taller triangles, thus more horizontal normals



NVIDIA



# Normal Map Tools

---

- **NVIDIA has developed three tools to assist in Normal Map conversion and generation**
- **Adobe Photoshop Plugin**
  - **Outputs DDS files**
  - **Also supports compression**
  - **A variety of conversion types supported**
- **Bump Maker Tool**
  - **Imports .3ds files to view bump maps on**
  - **Imports textures to turn into normal maps**
  - **Exports Normal Maps as TGA files**



*n*VIDIA™

# Normal Maps

---

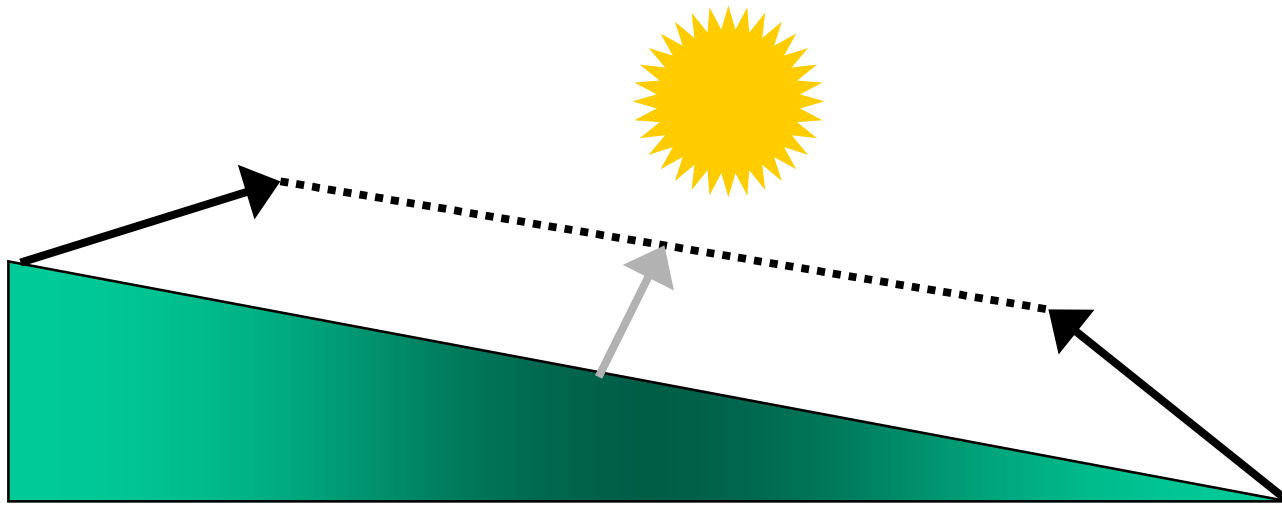
- **Normal Maps give us a per-pixel description of the surface normal vector of a surface**
- **We typically must convert a unit light vector into a color so that we can perform a per-pixel dot product in the register combiners or texture stages**



*n*VIDIA™

# Light Vector to Color

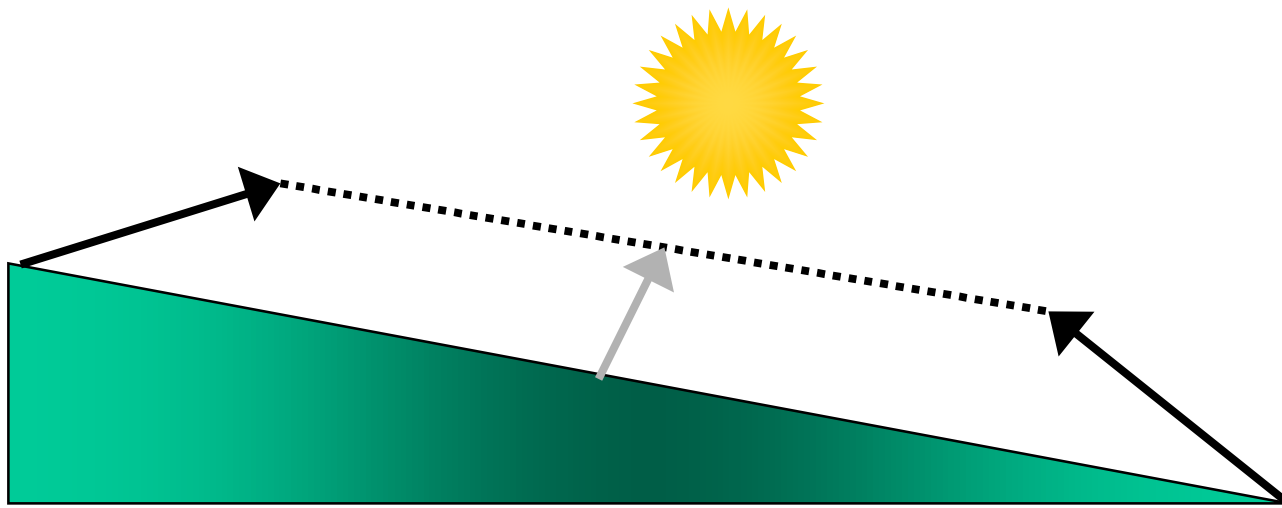
- We can directly convert a light vector into a RGBA color and pass in as the diffuse ( primary color ) or specular ( secondary color )



nVIDIA™

# Iterated Light Vectors

- This has the disadvantage that the light vector will get shortened across large polygons



nVIDIA™

# Normalization Cube Maps

---

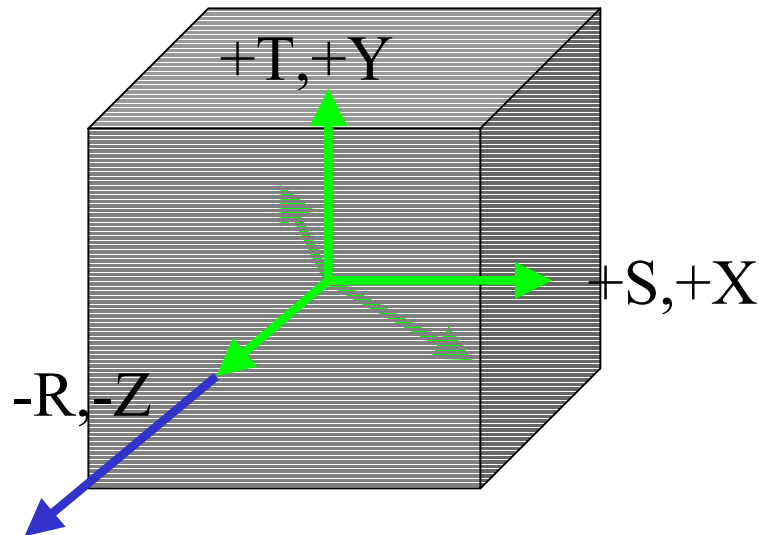
- You can use a Cube Map to renormalize your vectors on a per-pixel basis
- Pass in the vectors as S,T,R 3D Texture coordinates
- The Cube Map is set up to turn S,T,R floating point numbers into the correct RGBA Normals



NVIDIA™

# Visualizing the Vector Normalization Cube Map

The Cube map is accessed via vectors expressed as 3D texture coordinates ( S, T, R ).



The Blue Vector  $\langle 0, 0, -8 \rangle$  is passed in.

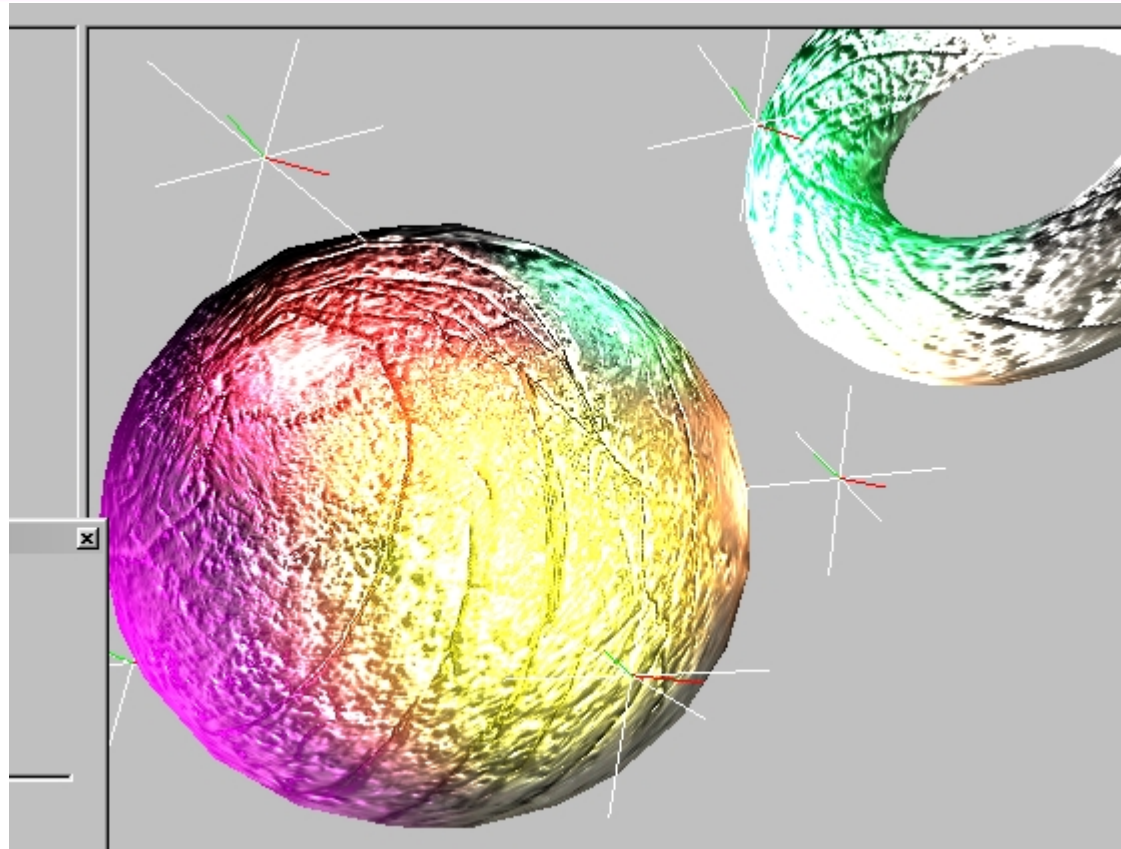
The Green Vector  $\langle 0, 0, -1 \rangle$  is returned in RGBA form as  $\langle 0x80, 0x80, 0x00 \rangle$  or  $0xff808000$



nVIDIA



# Per Pixel Lighting / Bump Mapping



nVIDIA™