NVIDIA™

Direct3D 7 Vertex Lighting

D. Sim Dietrich Jr.

NVIDIA Corporation

sim.dietrich@nvidia.com

Direct3D 7 Vertex Lighting

- How does Vertex Lighting Work?
- How do I Enable Vertex Lighting in my app?
- Optimizing Vertex Lighting

How Does Lighting Work?

- Light Types
  - Ambient Light – Approximation of Global Illumination
    - The color of shadows
  - Directional Light – Infinite, Parallel rays
    - Like the Sun or Moon
  - Point Light – Omni-directional, Local lights
    - Fireball, torch, explosion
  - Spotlight – Conical volume of light
    - Flashlight, or ummm… a Spotlight

How Does Lighting Work?

- Vertex Lighting uses several factors to determine the light intensity and color
  - The View Position
  - The Light Direction
  - The Vertex Normal
  - The Reflection Vector
  - The Light Attenuation Function
  - The Light Color and Intensity
  - The Material Color
  - The Light Type (Directional, Point, Spot)

How Does Lighting Work?

- Vertex Lighting uses several factors to determine the light intensity and color
  - The View Position
  - The Light Direction
  - The Vertex Normal
  - The Reflection Vector
  - The Light Attenuation Function
  - The Light Color and Intensity
  - The Material Color
  - The Light Type (Directional, Point, Spot)

View Position

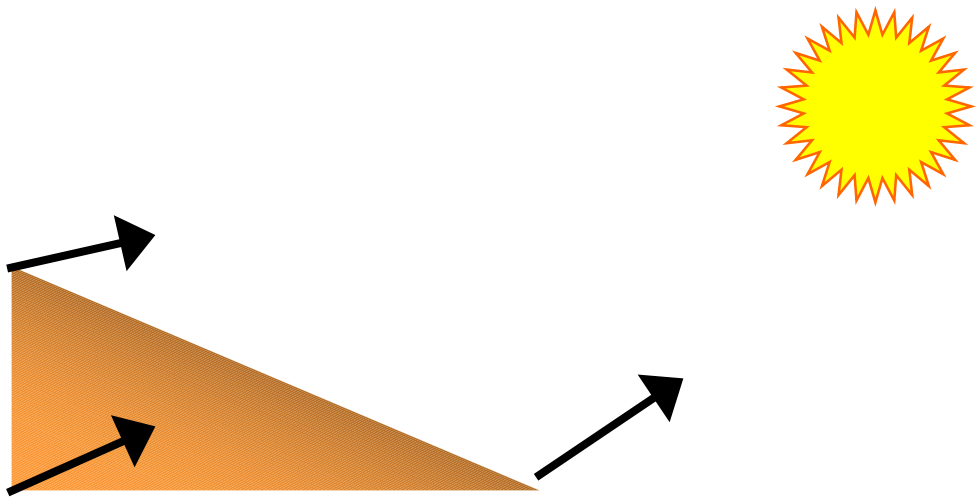
- **The Viewer position is <0,0,0> in view space**
- **It is irrelevant if using a Diffuse Directional Light**
- **Also if D3DRENDERSTATE_LOCALVIEWER is false – more on this later**

How Does Lighting Work?

- **Vertex Lighting uses several factors to determine the light intensity and color**
  - **The View Position**
  - **The Light Direction**
  - **The Vertex Normal**
  - **The Reflection Vector**
  - **The Light Attenuation Function**
  - **The Light Color and Intensity**
  - **The Material Color**
  - **The Light Type (Directional, Point, Spot)**

Light Direction

- **Normalized Vector Pointing towards the light :** L
- **For Directional, this is constant**
- **For Point and Spotlights, this is calculated per-vertex**

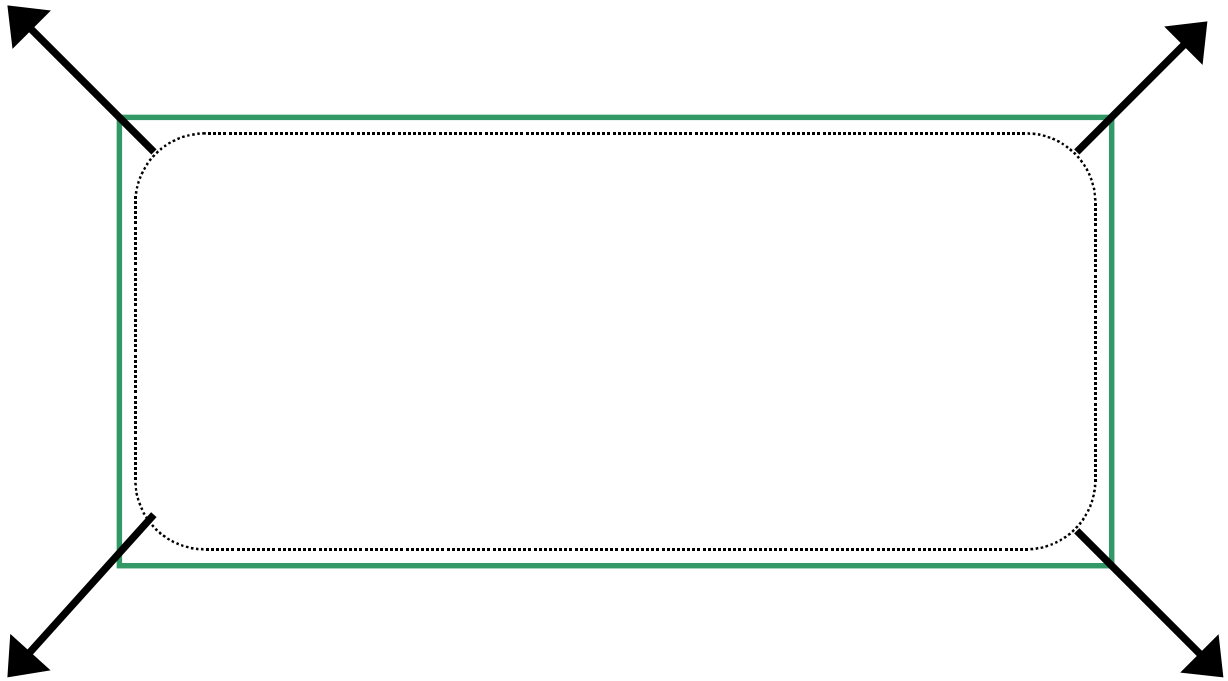


How Does Lighting Work?

- **Vertex Lighting uses several factors to determine the light intensity and color**
  - **The View Position**
  - **The Light Direction**
  - **The Vertex Normal**
  - **The Reflection Vector**
  - **The Light Attenuation Function**
  - **The Light Color and Intensity**
  - **The Material Color**
  - **The Light Type (Directional, Point, Spot)**

Vertex Normals

- **Vertex Normals define the true surface that the polygonal boundary representation is approximating.**
- **This is the basis for Gouraud Shading**

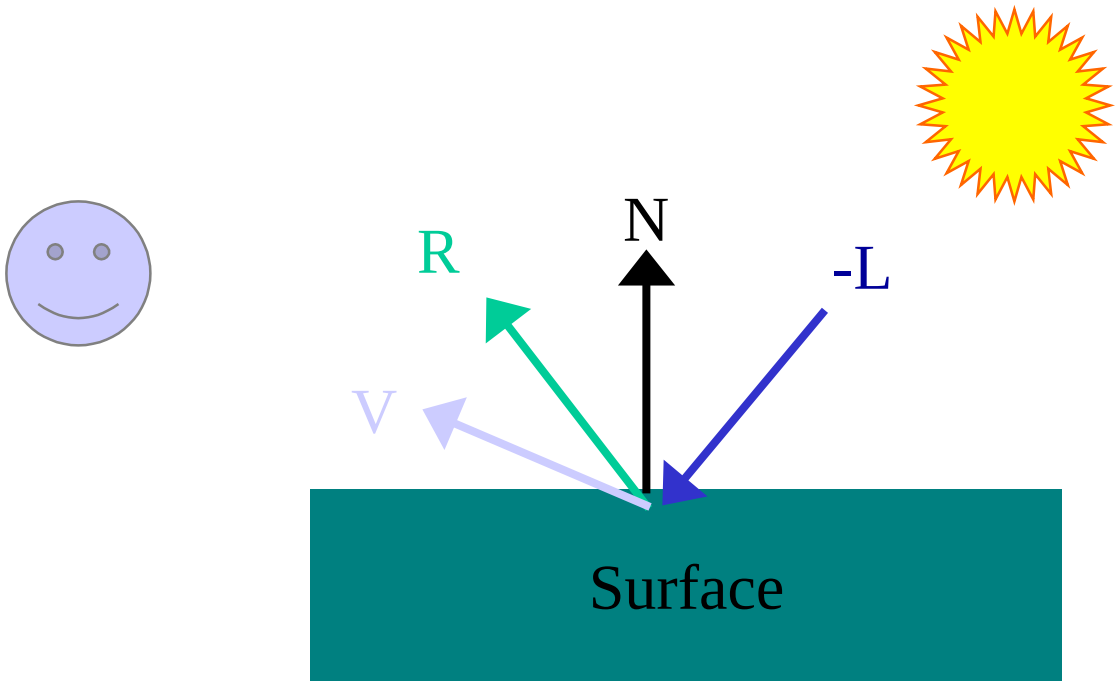


How Does Lighting Work?

- Vertex Lighting uses several factors to determine the light intensity and color
  - The View Position
  - The Light Direction
  - The Vertex Normal
  - The Reflection Vector
  - The Light Attenuation Function
  - The Light Color and Intensity
  - The Material Color
  - The Light Type (Directional, Point, Spot)

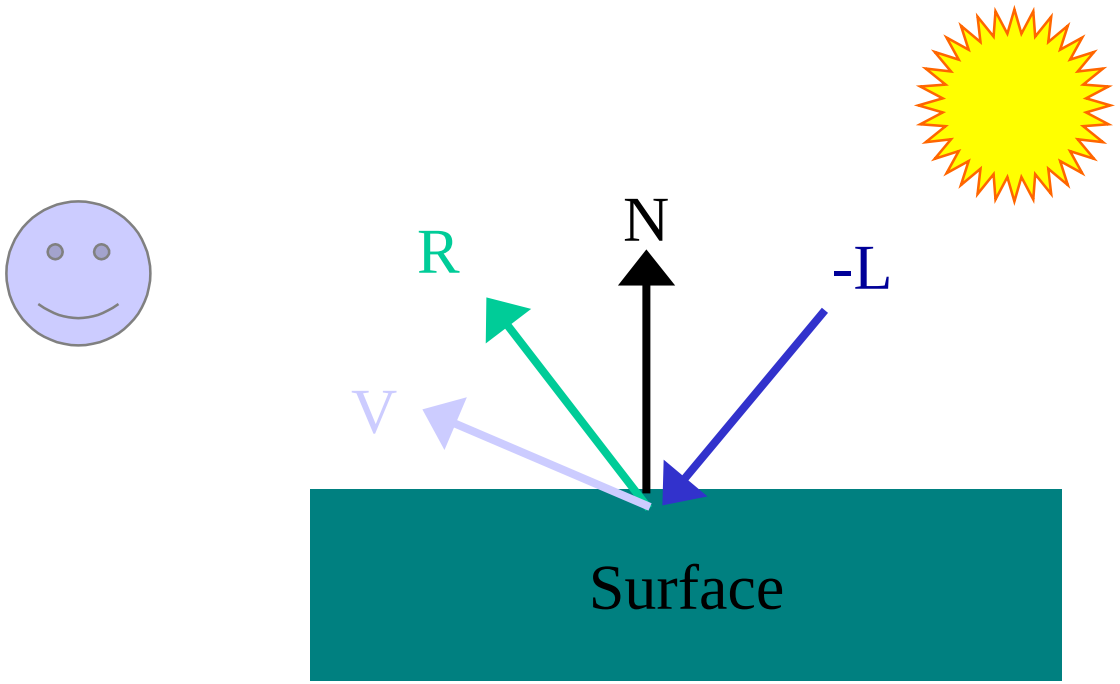
Reflection Vector

- **Angle of Reflection = Angle of Incidence**
- **Used for Specular lighting to determine how much light is reflected off the surface and towards the eye**



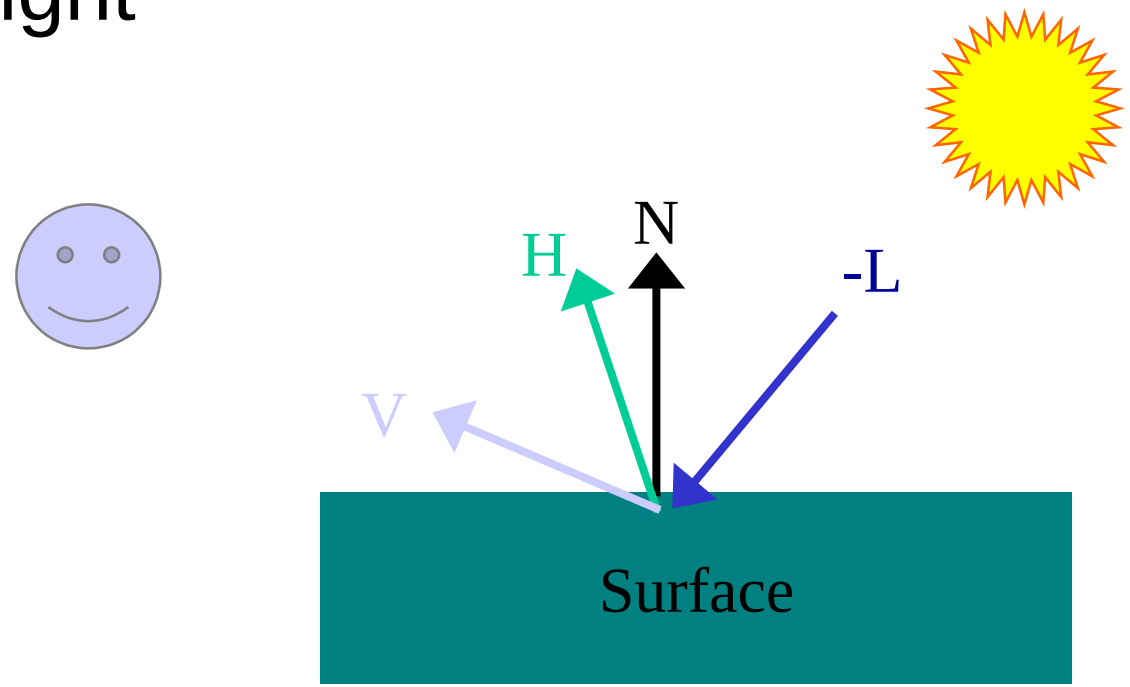
Reflection Vector

- **With Specular, instead of L DOT N, R DOT V is used.**
- **Actually, N DOT H is used instead**



Half Angle Vector H

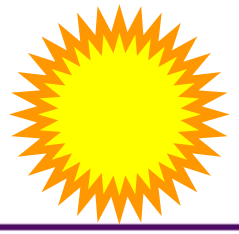
- H is the Half-Angle vector, the vector between V – the vector from the vertex to the eye, and L, the vector towards the light



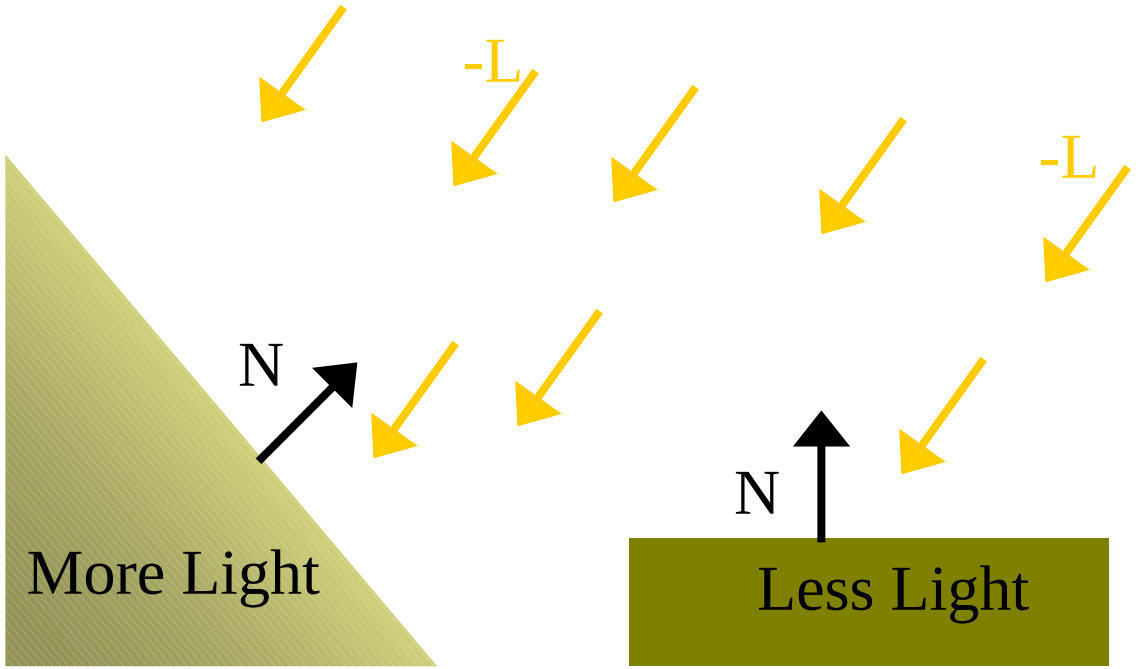
How Does Lighting Work?

- Vertex Lighting uses several factors to determine the light intensity and color
  - The View Position
  - The Light Direction
  - The Vertex Normal
  - The Reflection Vector
  - The Light Attenuation Function
  - The Light Color and Intensity
  - The Material Color
  - The Light Type (Directional, Point, Spot)

Attenuation Function



- **Ambient light – Not attenuated at all**
- **Directional – Attenuates based on surface orientation relative to light vector**
- **Diffuse - L DOT N**



The Attenuation Function

- **Point Lights have position, so distance matters**
- **The maximum amount of diffuse light visible from a surface is still L DOT N**
- **But intensity is further reduced based on distance from light source**
- **If** d **represents distance from a light,**

Attenuation =

$$\frac{1}{(c0 + c1 * d + c2 * d * d)}$$

The Attenuation Function

$$\frac{1}{(c0 + c1 * d + c2 * d * d)}$$

- This function is different from DX6
  - The model is now the same as OpenGL
- As d goes to infinity, Attenuation becomes very small, but not zero
- In practice, the color has no effect if the light is too far
- But the calculations are still performed

The Attenuation Function

$$\frac{1}{(c0 + c1 * d + c2 * d * d)}$$

- By Setting the constants c0, c1 & c2, you can adjust the falloff ramp of the light
- Setting c1 = 0, c2 > 0 will give you a radial distance squared falloff
- Setting c1 > 0, c2 = 0 gives a linear falloff
- Attenuation denominators less than one will give very large factors, thus making the light too bright close up
- To avoid this, make sure c0 is set to some positive number, 1 or greater

Spotlight Attenuation

- Spotlights attenuate based on cone angle
- Also an exponent to effect the angular falloff amount
- L DOT D
- L points from the vertex towards the spotlight
- D is the spotlight direction
- If L DOT D is less than the Spotlight's cutoff angle, the attenuation factor is set to 0
- There is also a power, which allows a quicker angular falloff, similar to the specular power of a material

How Does Lighting Work?

- **Vertex Lighting uses several factors to determine the light intensity and color**
  - **The View Position**
  - **The Light Direction**
  - **The Vertex Normal**
  - **The Reflection Vector**
  - **The Light Attenuation Function**
  - **The Light Color and Intensity**
  - **The Material Color**
  - **The Light Type (Directional, Point, Spot)**

Light Color And Intensity

- **Each Light has an RGBA color, specified as 4 floats from 0.0f to 1.0f**
- **A color of (0.0f, 1.0f, 0.0f, 0.0f) would specify a full intensity green light**
- **A color of (0.2f, 0.2f, 0.0f, 0.0f) would specify a dim purple light**

How Does Lighting Work?

- **Vertex Lighting uses several factors to determine the light intensity and color**
  - **The View Position**
  - **The Light Direction**
  - **The Vertex Normal**
  - **The Reflection Vector**
  - **The Light Attenuation Function**
  - **The Light Color and Intensity**
  - **The Material Color**
  - **The Light Type (Directional, Point, Spot)**

Material Properties

The D3DMATERIAL7 structure represents how the D3D lighting should be applied to the surface being lit

```
diffuse;        /* Diffuse color RGB */
ambient;         /* Ambient color RGB */
specular;        /* Specular Reflectivity */
emissive;        /* Emissive color RGB */
power;          /* Sharpness for specular highlight */ }
```

Material Properties

The Diffuse material property is used as a factor to modulate with the sum of all diffuse lights

The Specular material property is used similarly

The Ambient material parameter is used to determine how much the material reacts to the ambient light in the scene

The Emissive property describes how much light is coming directly from an object – like an LCD, or the surface of a light bulb

All Diffuse and Specular Material Properties can be pulled from the global currently selected material or from the vertex colors

Material Properties

- The Power is used as the exponent to increase directional falloff of the specular term
  - If H dot N is very close to 1, it will not be dimmed much by raising it to a power
  - If H dot N is small, it will quickly fall off to zero
- Thus, Power is used to get tighter specular highlights

Enabling Vertex Lighting

- In DX7, Vertex lighting is on by default
  - Enable and disable via D3DRENDERSTATE_LIGHTINGENABLE
- Add lights to your scene
  - Fill out the D3DLIGHT7 structure and pass it to AddLight()
  - Enable up to 8 lights at a time
- Set up your global material, via the D3DMATERIAL7 structure
  - Set it via SetMaterial()

Enabling Vertex Lighting

- **Apps should specify**

  D3DFVF_XYZ | D3DFVF_NORMAL | D3DFVF_DIFFUSE | D3DFVF_SPECULAR in the vertex format

- **Can’t light if D3DFVF_XYZRHW is specified,**
  - **Can’t light D3DTLVERTICES**

Optimizing Vertex Lighting

- Don’t use all 8 lights all the time
  - 8 lights can be applied per triangle, per pass
  - That being said, if you are using all 8 lights, there is something wrong
  - 2 or 3 is usually enough
- No need to ‘sort by light’
  - Simply track for each object or terrain chunk drawn which lights could effect it
    - If Spot or Directional lights, ensure they aren’t facing away from the object
    - If Point light, ensure object is within falloff range
    - It is fast to enable/disable lights

Relative Light Cost

- **Light cost from most to least**
  - **Spot Lights**
  - **Local Lights**
  - **Infinite Lights**
  - **Ambient Light**

Turn Off Local Viewer

- **The D3DRENDERSTATE_LOCALVIEWERENABLE state forces the API or GPU to take the viewer position into account for specular lighting**
  - **In other words, if Local Viewer is enabled the H in the H dot N equation needs to be recalculated per-vertex using the viewer position and the vertex position**
  - **If Local Viewer calculation is disabled, the view vector is used instead, effectively making the specular reflection view position independent**

Questions