Visual Computing Exercise 10:
Lighting and Shading

Philine Witzig
philine.witzig@inf.ethz.ch

Today

- Lighting, Shading, Textures
  - Phong Reflection Model
  - 2D Textures
  - Cubemaps
  - Tangent-Space Normal Mapping
- Exercise 10 (Lighting and Shading)
- Solution

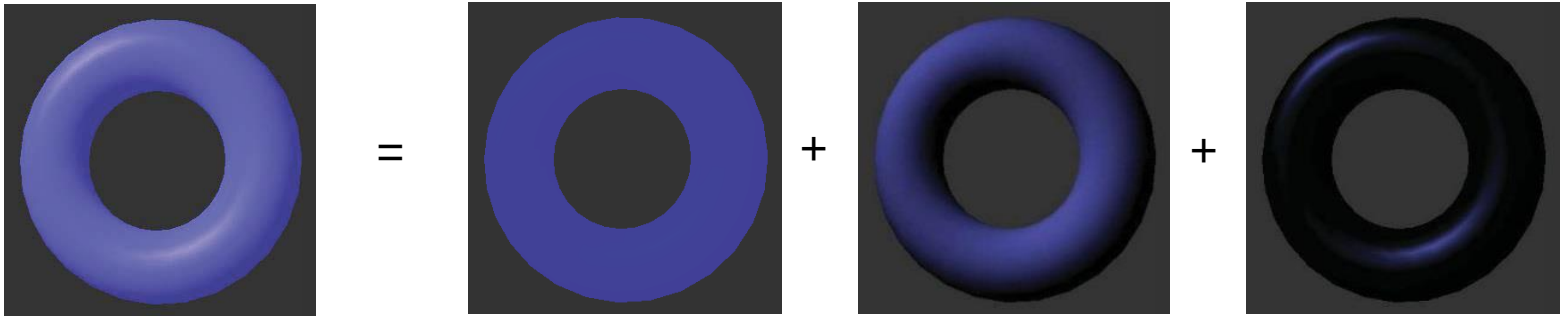
Lighting vs. Shading

- **Lighting**: modeling physical interaction between materials and light sources

- **Shading**: process of determining the color of a pixel (shader)

- Note: color information is a per vertex *attribute*

Phong Reflection Model



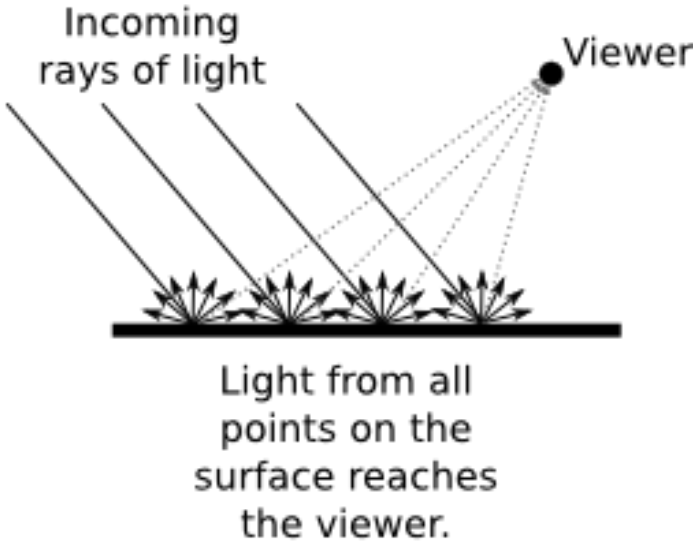
Ambient

- Light inherent in scene
- Approximate global light exchange of objects with eachother

$$I = I_a k_a$$

Diffuse

- Reflection on matt, dull surfaces

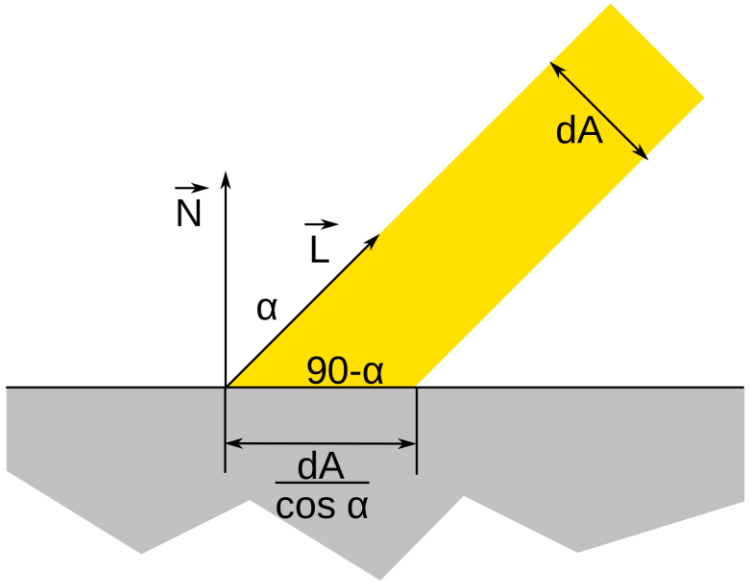


https://math.hws.edu/graphicsbook/c4/s1.html

Diffuse

$$I = I_p k_d \cos\theta$$

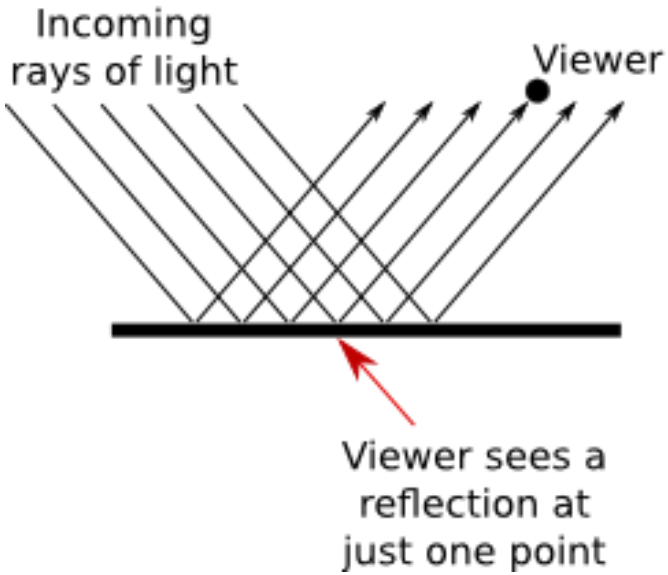
$$I = I_p k_d (\mathbf{N} \cdot \mathbf{L})$$



https://en.wikipedia.org/wiki/Lambertian_reflectance#/media/File:Oswietlenie_lamberta.svg

Specular

- Reflection on shiny surfaces



https://math.hws.edu/graphicsbook/c4/s1.html

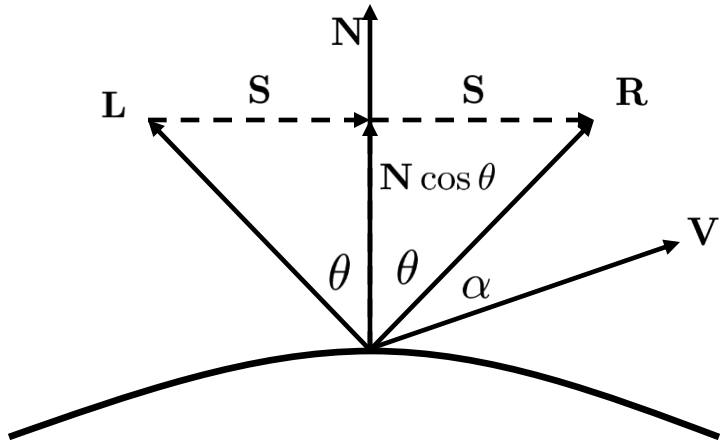
Specular

$$I = I_p k_s (R \cdot V)^n$$

$$\mathbf{R} = \mathbf{N}\cos\theta + \mathbf{S}$$

$$\mathbf{R} = 2\mathbf{N}\cos\theta - \mathbf{L} = 2\mathbf{N}(\mathbf{N}\cdot\mathbf{L}) - \mathbf{L}$$

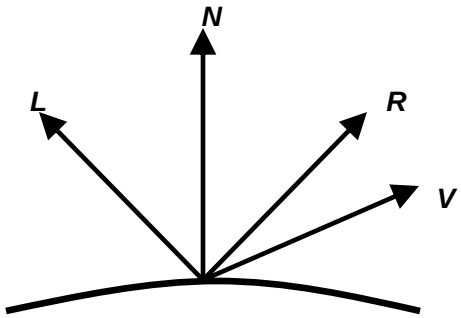
$$\cos\alpha = \mathbf{R}\cdot\mathbf{V} = (2\mathbf{N}(\mathbf{N}\cdot\mathbf{L}) - \mathbf{L})\cdot\mathbf{V}$$



Phong Reflection Model

$$I = \underbrace{I_a k_a}_{\text{Ambient}} + I_p \left(\underbrace{k_d (\mathbf{N} \cdot \mathbf{L})}_{\text{Diffuse}} + \underbrace{k_s (\mathbf{R} \cdot \mathbf{V})^n}_{\text{Specular}} \right)$$

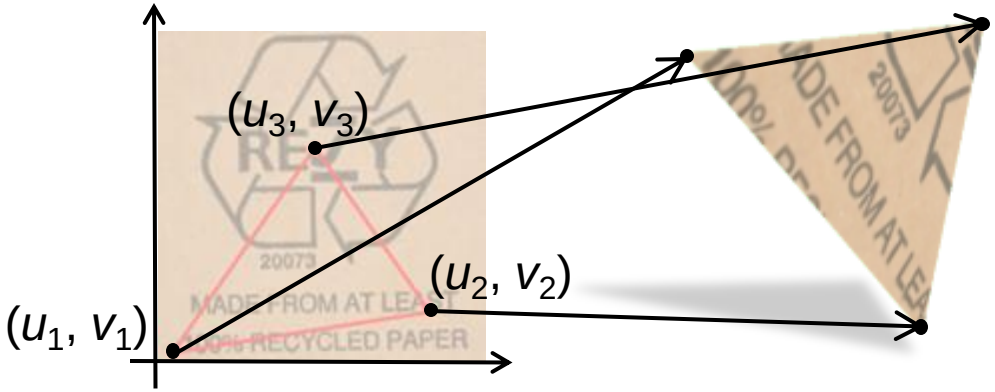
- Material parameters
  - k_a, k_d, k_s, n
- Light parameters
  - I_a, I_p
- Geometry parameters
  - N, L, V



Texture Mapping

- **Idea**: “Map an image to a triangle”

- Per vertex texture coordinates (uv-coordinates)



Textures in WebGL

- Creating a texture in WebGL
  - Load image from file: create an Image object and set source
  - `gl.texImage2D` command

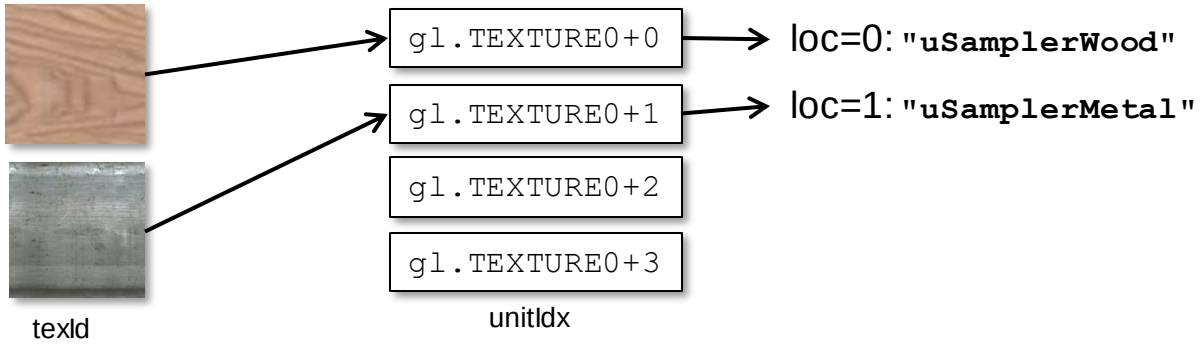
```
const texture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, texture);

gl.texImage2D(gl.TEXTURE_2D, level,
              internalFormat, width, height,
              border, srcFormat, srcType, pixel);
```

Textures in WebGL

- Pass texture handle to fragment shader
  - Associate texture with a *texture unit*
  - Pass index of texture unit to fragment shader

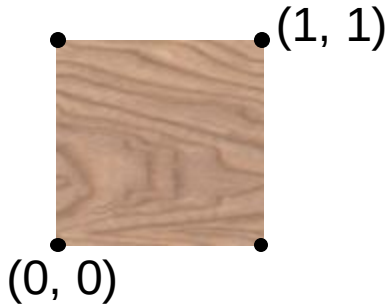
```
gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, textureMap);
var textureLocation =
        gl.getUniformLocation(shaderProgram, "uSamplerTexture")
gl.uniform1i(textureLocation, 0);
```

Textures in WebGL



Textures in WebGL

- Texture coordinates are per-vertex *attributes*
- Can process them in vertex shader
- Then perform texture lookup in fragment shader



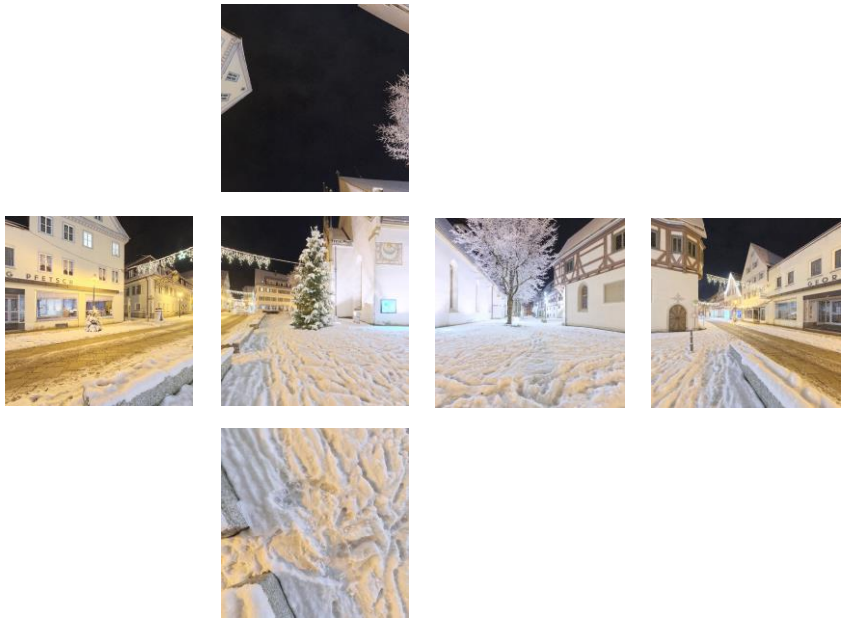

```
uniform sampler2D uSamplerTexture;
varying vec2 texCoords

void main(void) {
 vec4 txtColor = texture2D(uSamplerTexture, texCoords);
}
```

Sphere Maps

- Spherical texture
- Orthographic projection of the surface of a mirroring sphere

Cube Maps

- “6-sided texture”
- Images of six views into a 360° environment



Skybox

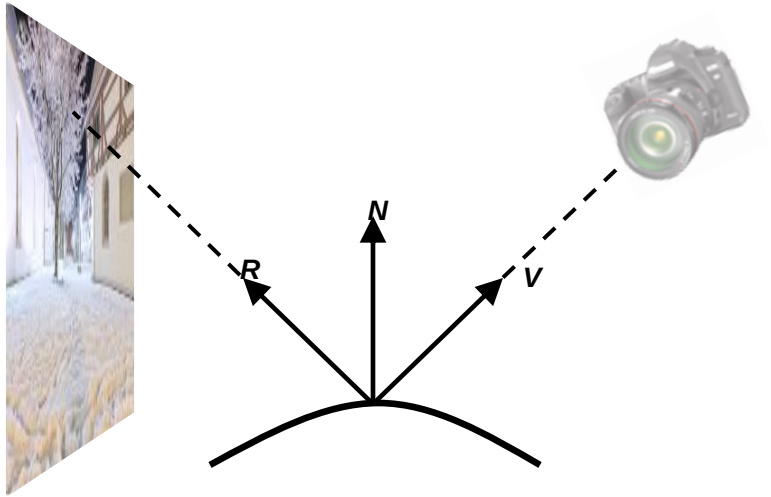
- Box of 6 textured quads (6 2D textures)
- If viewer in center: correct distortion (assumption: sky is infinitely far way)

Reflection Mapping

- Simulate mirror surface
- Reflect viewing direction, look up color in cube map

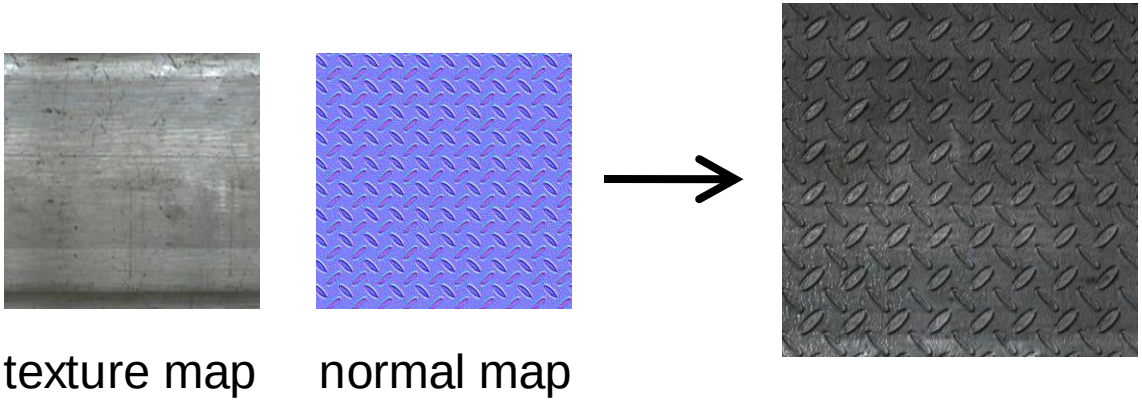
Reflection Mapping

- Reflect V at N → R
- Lookup: `textureCube(uSamplerEnvMap, R)`



Tangent-Space Normal Mapping

- **Idea**: each pixel has its own normal
- Fake high-resolution geometry



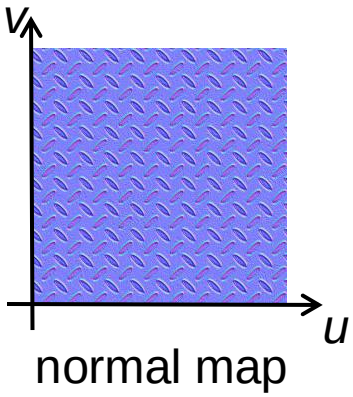
texture map normal map

Normal Map

- Stores perturbed normal vector for each pixel as (*r*, *g*, *b*) color:

$$n' = (2r - 1, 2g - 1, 2b - 1)^T$$

- Map from $[0, 1]$ to $[-1, 1]$
- Usually normalized so $|n'| = 1$

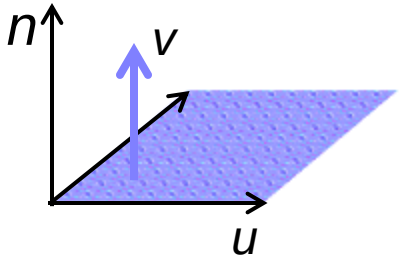


normal map

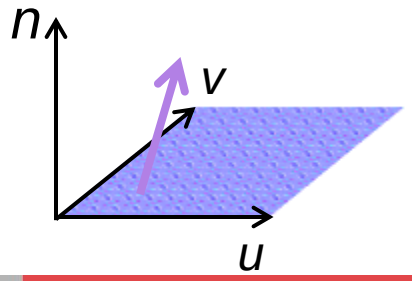
Normal Map

- Examples

$$n' = (2r - 1, 2g - 1, 2b - 1)^T$$



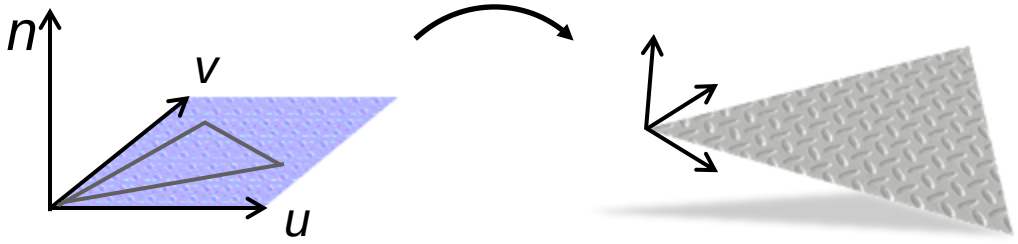
Unperturbed normal
color = (0.5, 0.5, 1)



Tilt normal in u direction
color = (0.7, 0.5, 0.958)

Normal Map

- Normal map is relative to texture coordinates
- How to find corresponding coordinates in space?

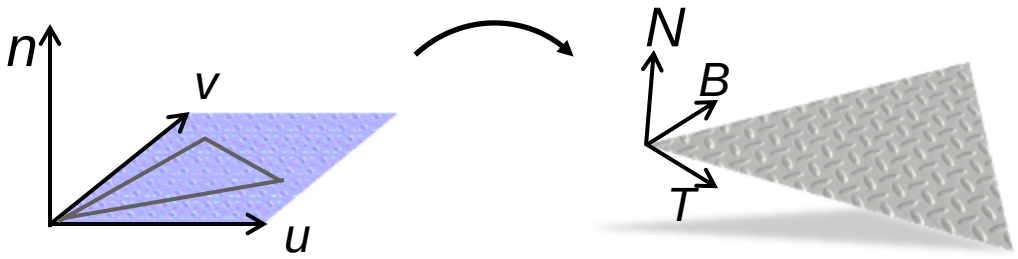


Texture space

?

Tangent Space

- Vectors corresponding to (u, v, n) in space are (N, T, B)
- If we know them, we can compute the perturbed normal in space



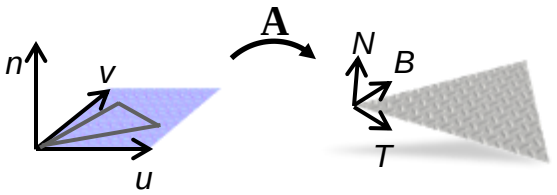
Texture space

Tangent space

Tangent Space

- Per-triangle: Linear relationship between 3D positions (x,y,z) and texture coordinates (u,v)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{A} \begin{bmatrix} u \\ v \end{bmatrix}$$



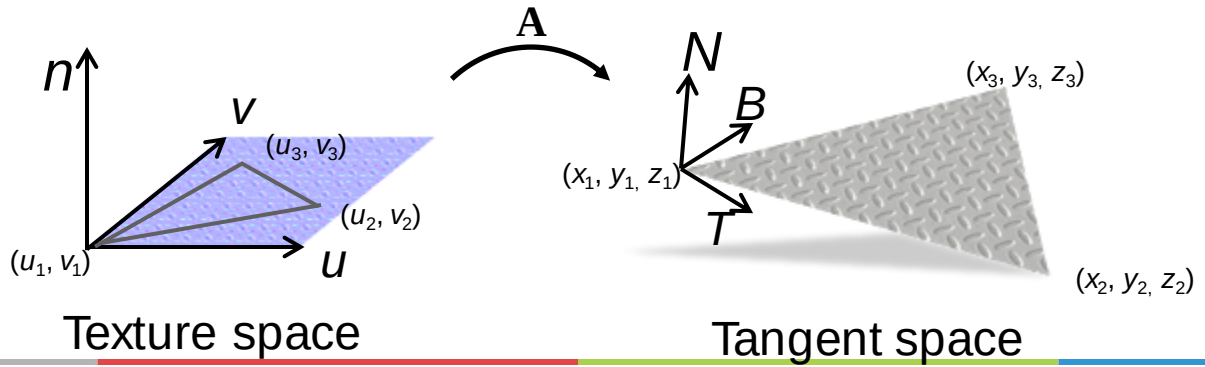
- T and B correspond to (1,0) and (0,1) in texture space

$$\begin{bmatrix} | \\ \vec{\mathbf{T}} \\ | \end{bmatrix} = \mathbf{A} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} | \\ \vec{\mathbf{B}} \\ | \end{bmatrix} = \mathbf{A} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} | & | \\ \vec{\mathbf{T}} & \vec{\mathbf{B}} \\ | & | \end{bmatrix} = \mathbf{A} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{A}$$

Tangent Space

- Got two conditions for A

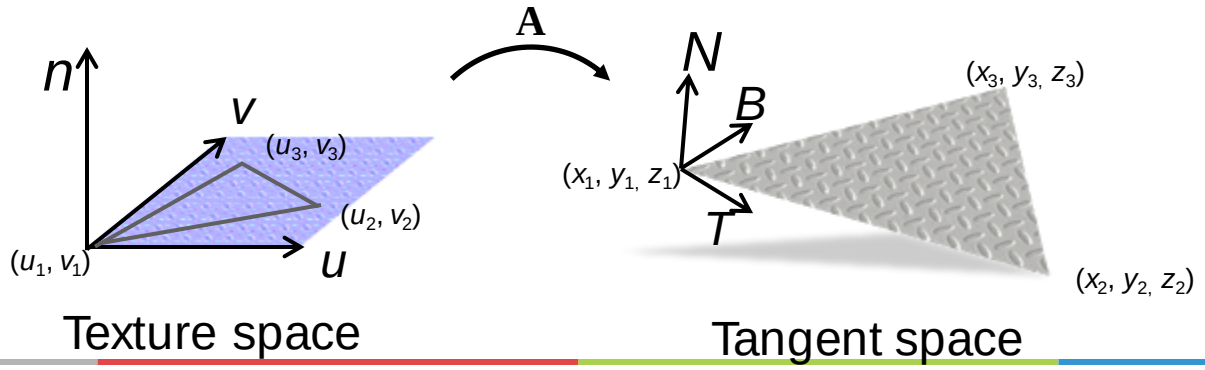
$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \mathbf{A}\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} \qquad \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \mathbf{A}\begin{bmatrix} u_3 \\ v_3 \end{bmatrix} \longrightarrow \begin{bmatrix} x_2 & x_3 \\ y_2 & y_3 \\ z_2 & z_3 \end{bmatrix} = \mathbf{A}\begin{bmatrix} u_2 & u_3 \\ v_2 & v_3 \end{bmatrix}$$



Tangent Space

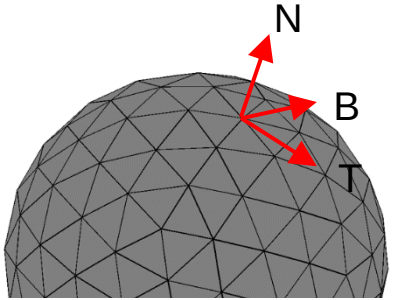
- Can compute Tangent and Bitangent per triangle

$$\begin{bmatrix} | & | \\ \vec{\mathbf{T}} & \vec{\mathbf{B}} \\ | & | \end{bmatrix} = \mathbf{A} = \begin{bmatrix} x_2 & x_3 \\ y_2 & y_3 \\ z_2 & z_3 \end{bmatrix} \begin{bmatrix} u_2 & u_3 \\ v_2 & v_3 \end{bmatrix}^{-1}$$



Tangent Space

- For per-vertex T/B:
  Average over adjacent triangles
  (similar to computing per-vertex normal)



- Need orthogonalization to make T and B orthogonal to the vertex normal

$$\mathbf{T}' = \mathbf{T} - \mathbf{N}(\mathbf{N} \cdot \mathbf{T}) \qquad \mathbf{B}' = \mathbf{B} - \mathbf{N}(\mathbf{N} \cdot \mathbf{B})$$

Exercise 10

Exercise 10

- Goals
  - Understand and implement the Phong reflection model
  - Use textures in the fragment shader
  - Understand reflection mapping and normal mapping

Exercise 10

- Initial output

Task 1)

- Phong Model
  - Implement Phong reflection model in the fragment shader (`fragment.js`)
  - Useful GLSL commands: `dot, clamp, reflect, pow`

Task 2)

- Texture Mapping
  - Perform a **texture lookup** in the fragment shader and assign the colors to the bauble's

Task 3)

- Normal Mapping
  - Understand the computation of **tangents** and **bitangents**
  - Implement **normal mapping** in the fragment shader

Task 4)

- Reflection Mapping
  - Perform a **texture lookup** in the cube map
  - Take a look around the environment by **rotating** the **view matrix** of the skybox

Additional Resources

- Mozilla tutorial on textures: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial/Using_textures_in_WebGL
- glmatrix API: https://glmatrix.net/docs/
- WebGL Skybox Example: https://webglfundamentals.org/webgl/lessons/webgl-skybox.html

Questions?

philine.witzig@inf.ethz.ch