UNIVERSITY OF DERBY

5CM507 Graphics

Lecture 08 More Texture Maps

Dr Youbing Zhao

2025 年 11 月 17 日

Last Week

Texture Mapping

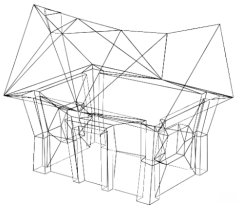
- Texture coordinates
- Sampling
- Mip-mapping

Contents

Texture : Image-based details imposed on polygon models

The most common texture map: colour/diffuse/albedo

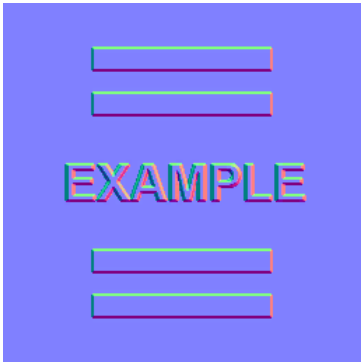
- Normal Pertubations
  - Normal mapping
  - Bump Mapping
- Cube map
  - Skybox
  - Environment Mapping
- Physically-based Rendering Maps



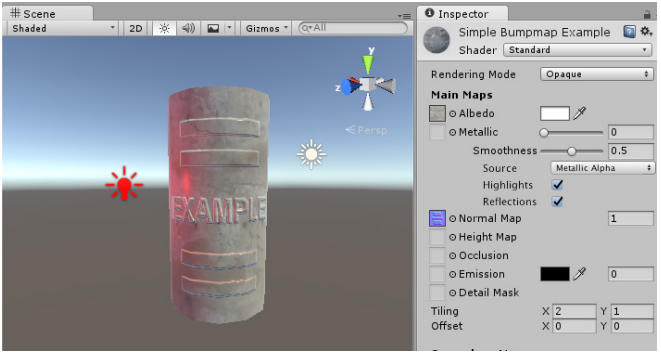
Normal Pertubations

Normal Mapping

Uses an RGB texture to describe detailed surface normals. Each pixel in the texture is a normal vector, $\vec{n} = (r_x, g_y, b_z)$ relative to its low-res target mesh surface.



An example normal map

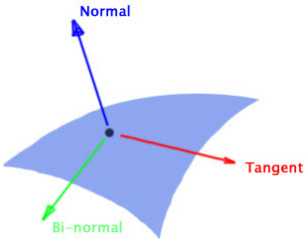


An Unity Example

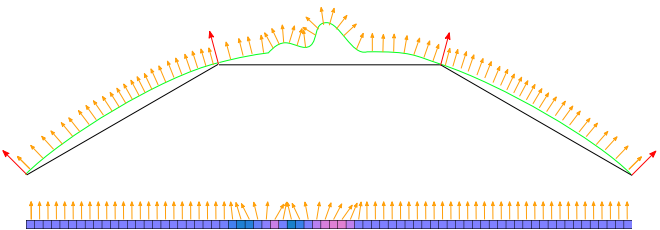
Normal Mapping

We need to find a reference frame for the normals specified in the normal map.

- The tangent space
  - Z: Normal, X: Tangent (texture u), Y: Bitangent(binormal) (texture v)
  - Axes defined in body space, then to world space via modelling transform



The tangent space



Applying normals to a mesh

Normal Mapping Calculation

- Blinn-Phong Lighting $\boldsymbol{I_a} k_a + k_d \left(\boldsymbol{l_p} \cdot \boldsymbol{n}\right) + k_s \left(\boldsymbol{h} \cdot \boldsymbol{n}\right)^{\beta}$
  - Normal
  - Light Direction: LightPos - FragPos
  - Half-vector: View Direction: View Pos - FragPos
- Method 1: Convert the normal to world space for each fragment
- Method 2: Convert the LightPos and ViewPos to tangent space for each vertex
- Tangent $\Longleftrightarrow$ World Frame
  - TBN matrix: Tangent space $\rightarrow$ World space
  - Inverse (transpose) of TBN : World space $\rightarrow$ Tangent space

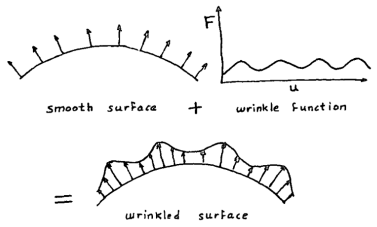
$$\mathbf{TBN} = \begin{pmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{pmatrix} \quad \mathbf{TBN}^{-1} = \mathbf{TBN}^{T} = \begin{pmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{pmatrix}$$

Bump Mapping

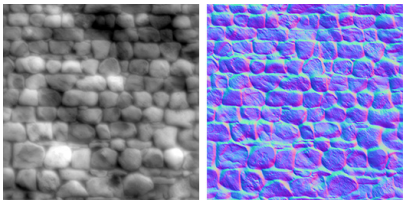
Older technique, proposed by James Blinn in 1978

- Give offset height only
- Less flexible
- Use texture gradient(slope) to indirectly change normals
- Artists may use terms interchangeably
- Can be converted to normal map

Math: `https://www.pbr-book.org/3ed-2018/Materials/Bump_Mapping`



Bump function

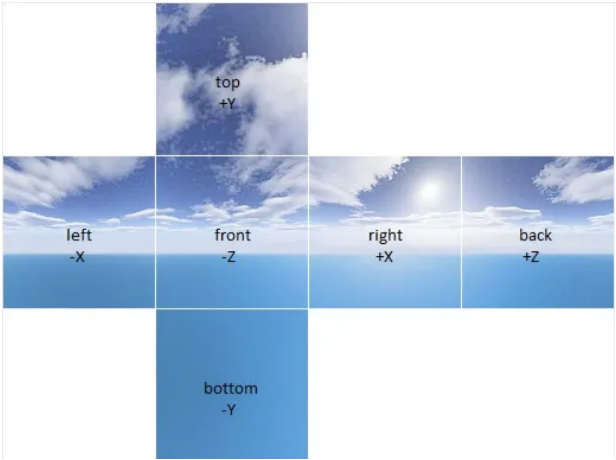


Bump map vs Normal map

Cube Maps

Cube Maps

- Six 2D texture images for a box
- Supported by OpenGL and D3D
- Texture coordinates are 3D : usually given by the vertex location
- Used for games and virtual environments



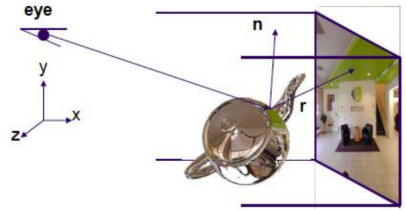
Skybox with OpenGL Cube Map

- OpenGL
  - Draw a unit cube
  - Use the vertex position as texture coordinates
- Shader
  - Directly use texture colour
  - No lighting or normals needed

```
unsigned int textureID;
glGenTextures(1, &textureID);
glBindTexture(GL_TEXTURE_CUBE_MAP, textureID);

glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, images[0]);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, images[1]);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, images[2]);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, images[3]);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, images[4]);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, images[5]);
```

Environment Mapping with Cube Maps

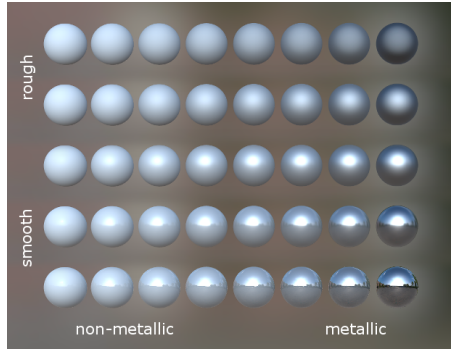


- Compute the reflection vector
  $r = I - 2.0 * N * dot(N, I)$
- Assume the object at origin
- Use r as the cube map texture coordinate

Physically-based Rendering Texture Maps

PBR Maps

- Parameters in Physically Based Rendering
- Examples
  - Albedo (Colour), Normal
  - Roughness
  - Metalness
  - Specular
  - Height
  - Opacity
  - Ambient occlusion (AO)
  - Refraction
  - Self-illumination (Emission)
  - Subsurface scattering
  - ...



Roughness and Metalness

`https://conceptartempire.com/texture-maps/`

Ambient Occlusion (AO) Map

- Ambient occlusion : self shadows independent of light sources
- Achieve realistic high resolution shadow detail on lower resolution models

Original model

With ambient occlusion

Extracted ambient occlusion map

Maps Used to Create Realistic Skins

Maps used to create realistic skins : an example

Summary

- Normal Mapping
- Bump Mapping
- Cube Maps
- PBR Maps

Questions?