CS418
GLSL Shader Programming Tutorial
(II)
Textures in Shader

Presented by : Wei-Wen Feng
4/1/2009

MP3

- MP3 is now available
  - Mesh Display based on GLSL
  - Animation by vertex transformation
  - Per-pixel Phong Shading
  - Tangent-Space Normal Mapping

Today's Agenda

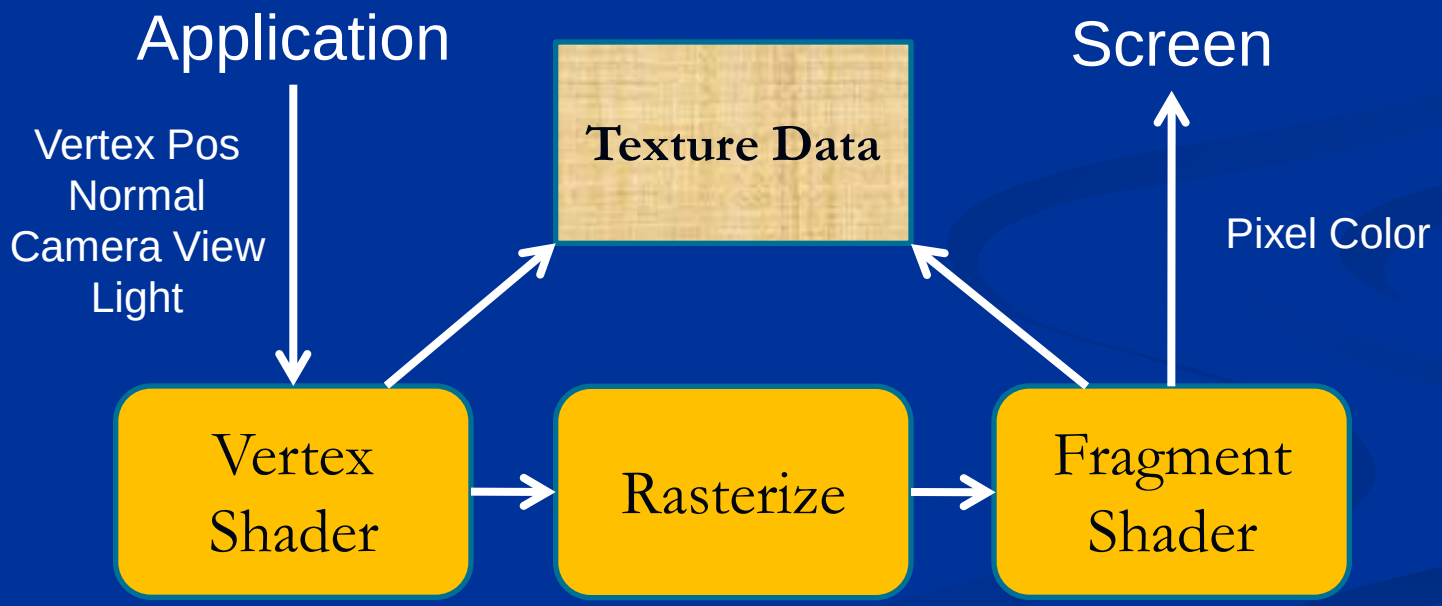
- Texture in Shader
- Normal Mapping
- Displacement / Parallax Mapping
- Q & A

Texture

- Texture can be :
  - Image → Texture mapping
  - Normal Map → Detail shading
  - Matrix Table → GPGPU
  - In general → a table storing data for your shader.

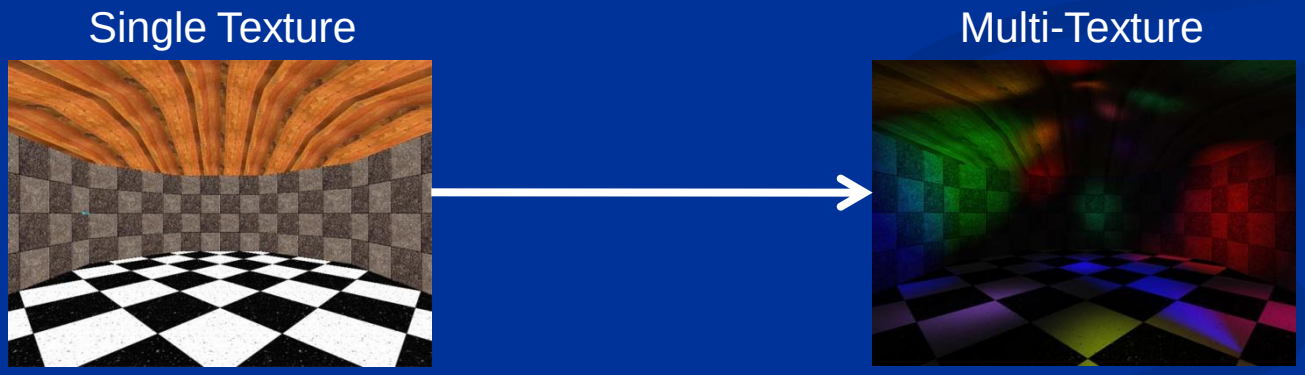
Texture in Shader

- Accessible in both Vertex & Fragment Shader



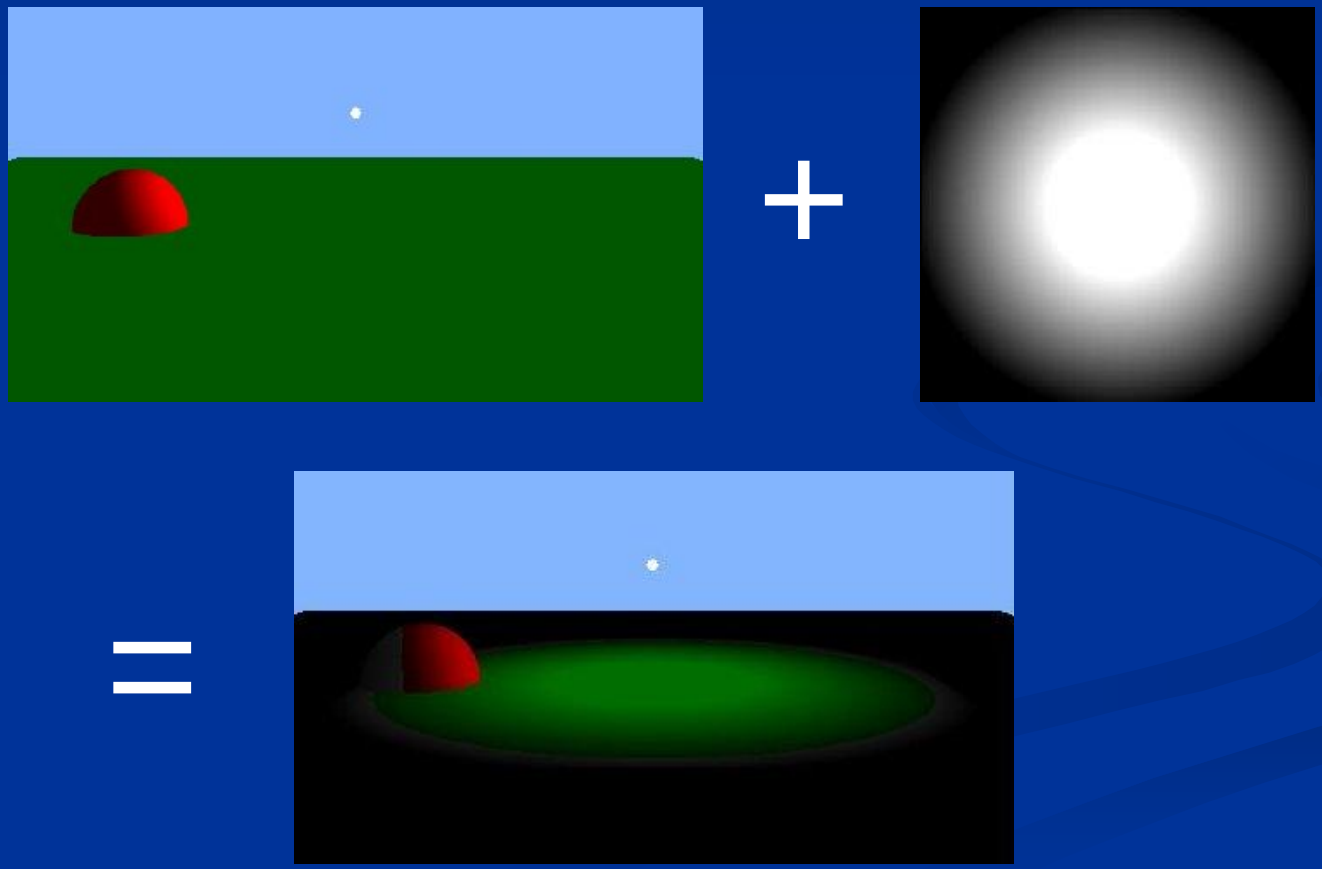
Multi-Texture

- OpenGL supports multi-texturing
  - Blend multiple texture together
  - Useful application : light map, normal map



Light map

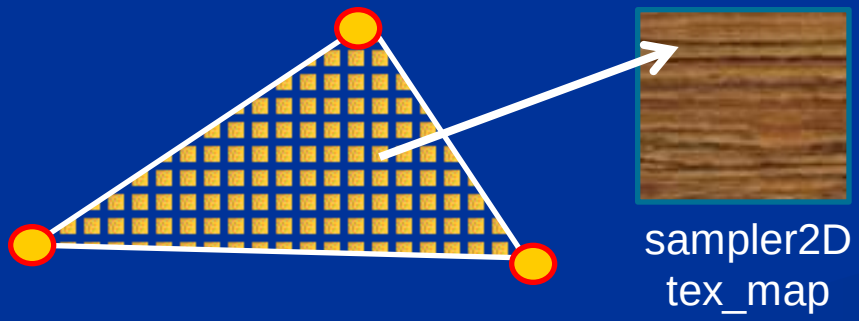
- Use texture to enhancing lighting effect



Using Texture in Shader

- Sampler2D (1D, etc)
  - Uniform variable
  - Represent a texture image
- texture2D(sampler, texcoord)
  - Access a 2D texture (sampler)

```
uniform sampler2D tex_map;
varying vec2 texcoord;

void main (void)
{
   gl_FragColor =
   texture2D(tex_map,texcoord);
}
```



Setup Texture for GLSL

- Create texture and obtain **TexID**.
  - Identical to creating texture in OpenGL

- Get the sampler uniform location.
- Bind the texture **TexID** to texture unit **N**.
- Pass **N** as an integer by glUniform.

- Still use glTexCoord to pass texture coordinates into shader
- Access texture coordinate in shader : gl_MultiTexCoord[0-7]

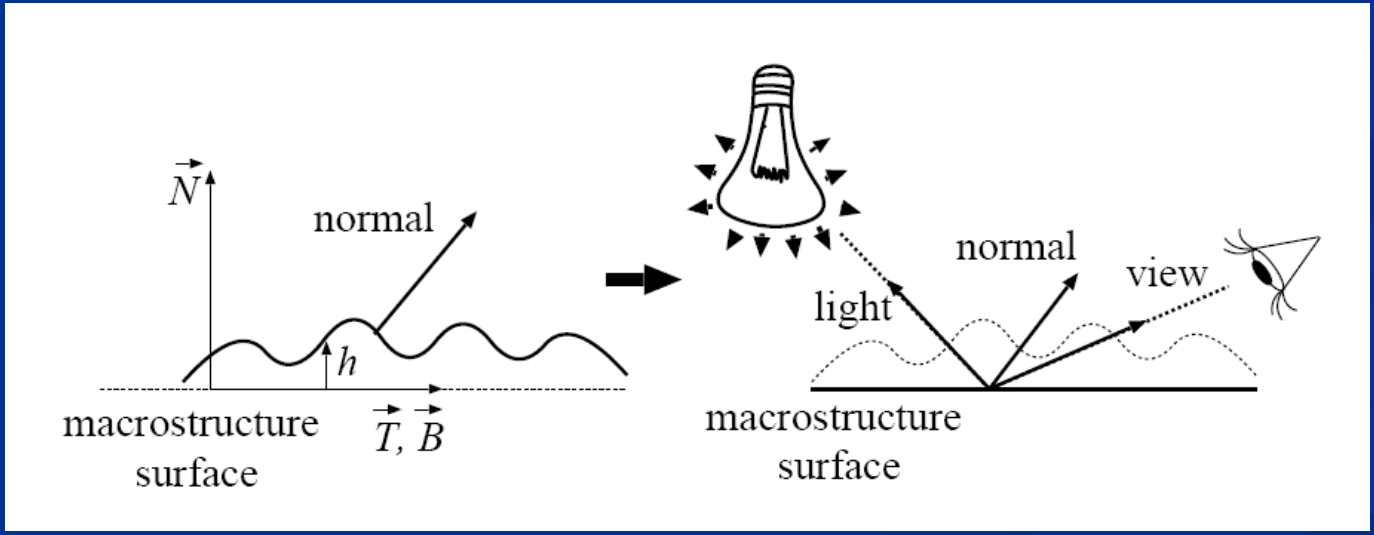
Code

```
glActiveTexture(GL_TEXTURE0 + 0);  // set unit-0 to current texture unit

int loc = glGetUniformLocation(shader_program,"tex_map");
glBindTexture(GL_TEXTURE_2D, iTexID);

glUniform1i(loc,  0);
```

```
uniform sampler2D tex_map;
varying vec2 texcoord;

void main (void)
{
   gl_FragColor =
   texture2D(tex_map,texcoord);
}
```

Normal Map

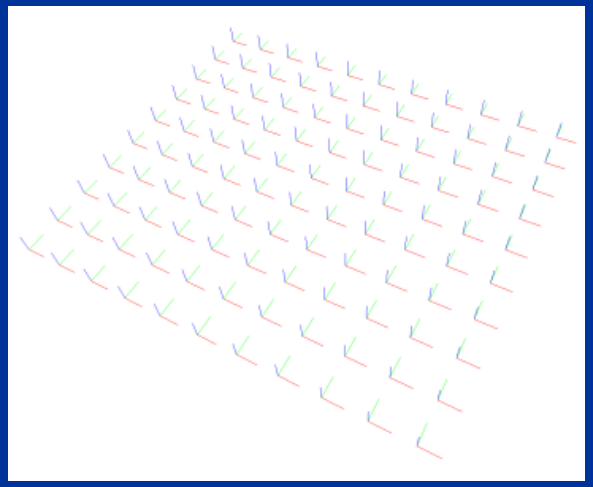
- Encoding small surface bump as texture color
- Object space VS Tangent Space



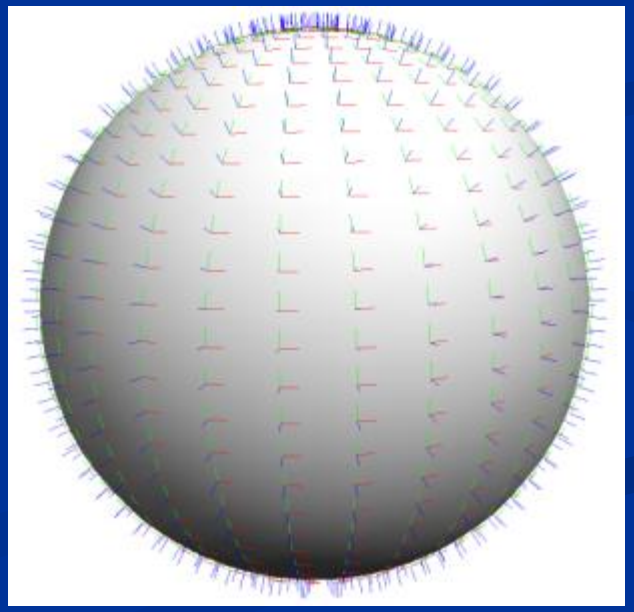
Compute TBN

- We need normal (N), tangent (T), bi-normal (B) vectors at each vertex.

Simple for a plane

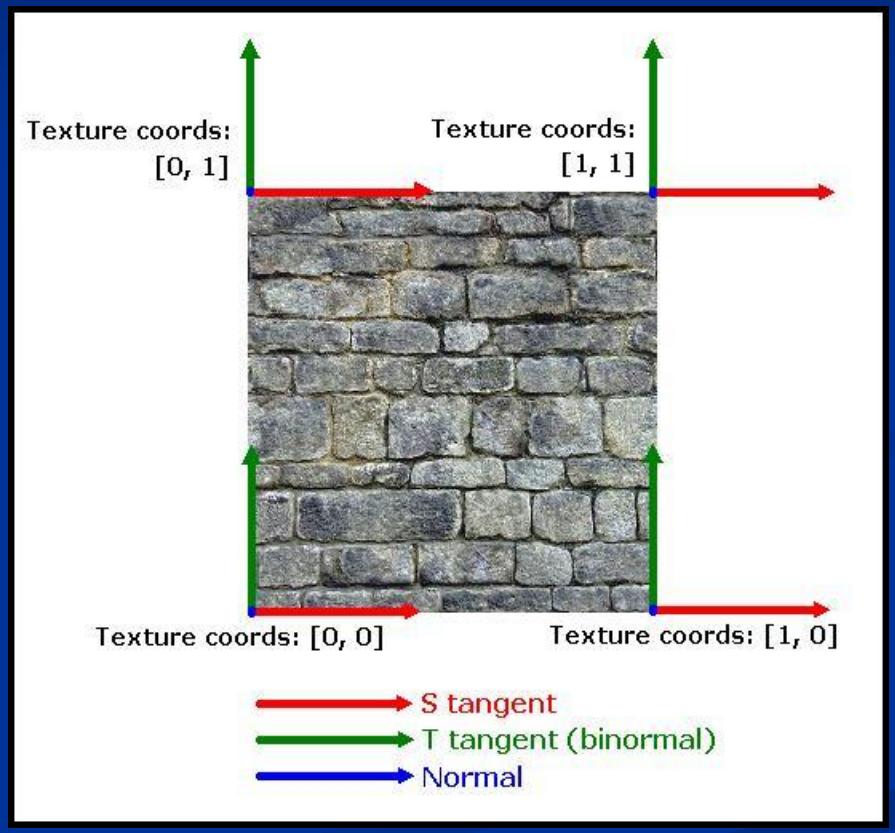


What about a sphere ?



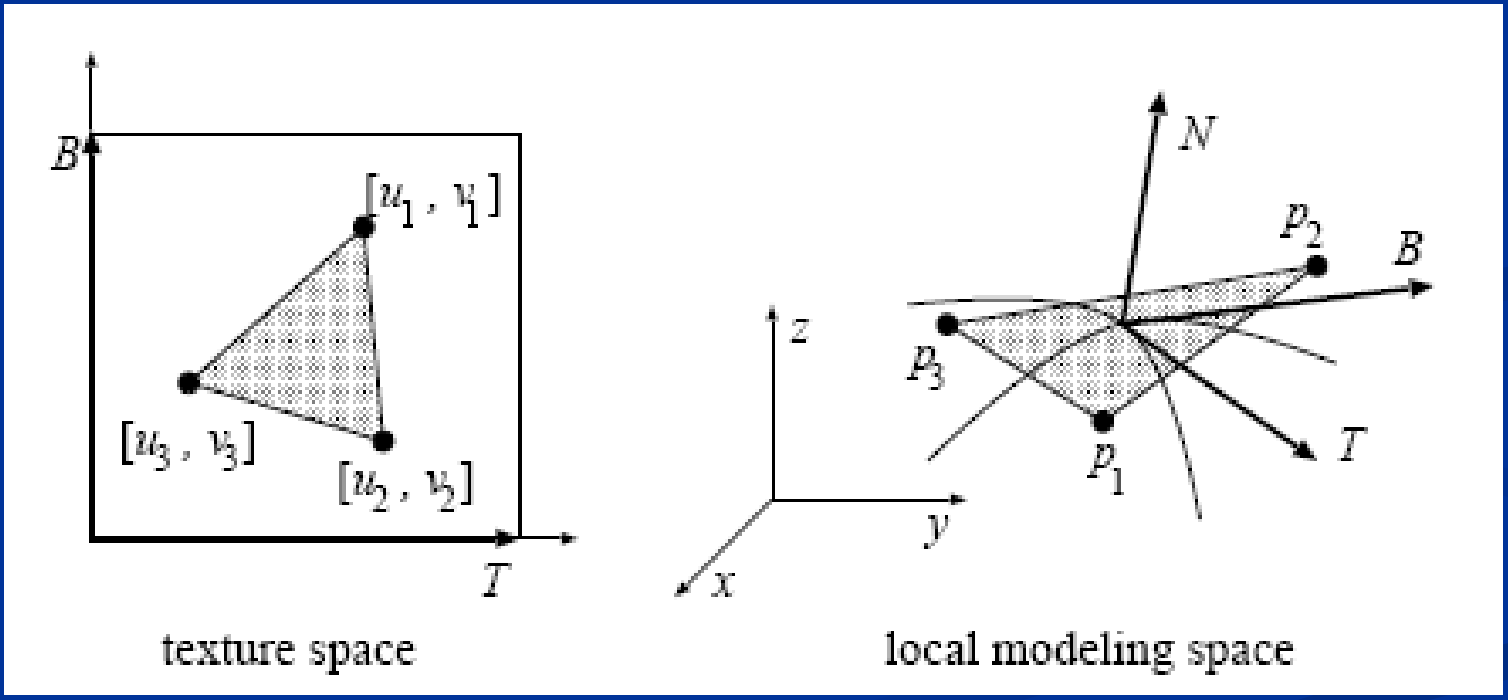
Compute TBN

- Use texture coordinates (s,t) to determine T/B vectors
- T should map to s-direction in texture coord. So is B.

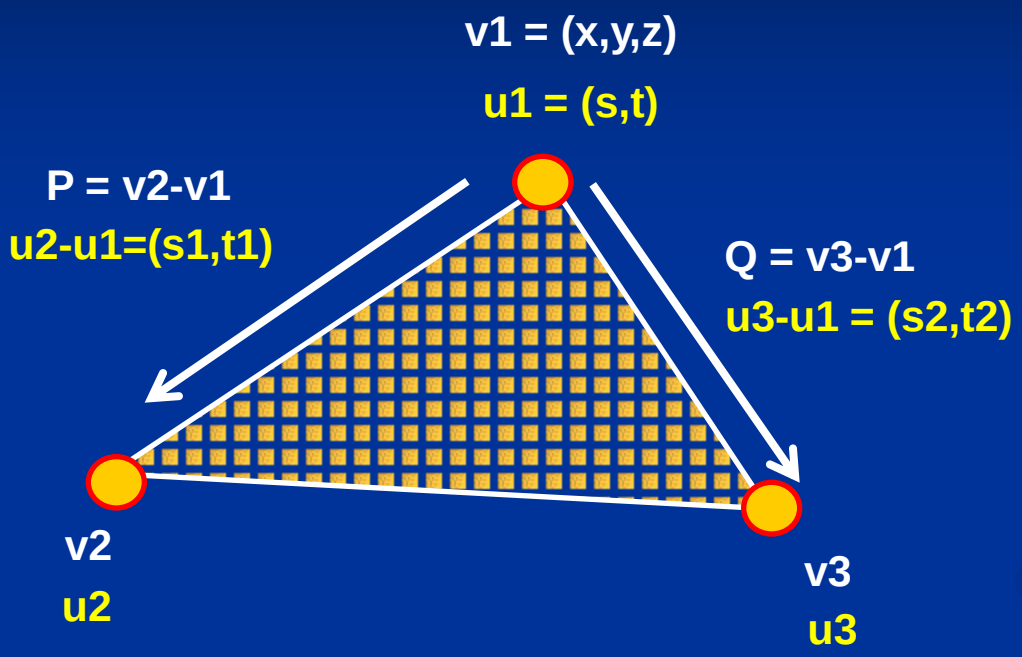


Compute TBN

- How ?



Compute TBN

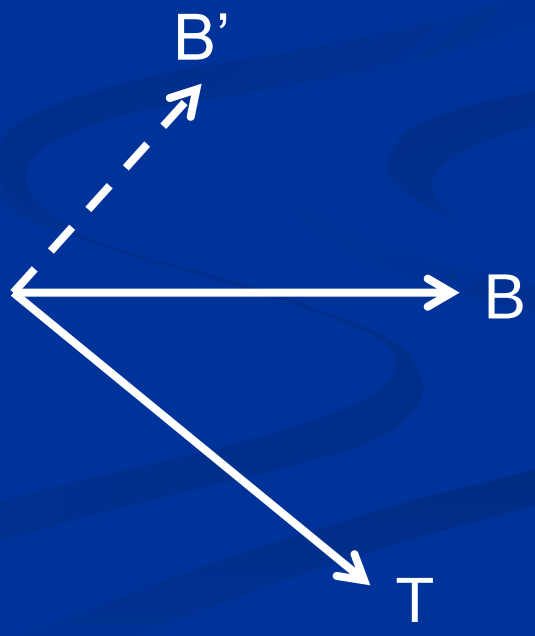


P = v2-v1 = Tangent*s1 + Binormal*t1

Q = v3-v1 = Tangent*s2 + Binormal*t2

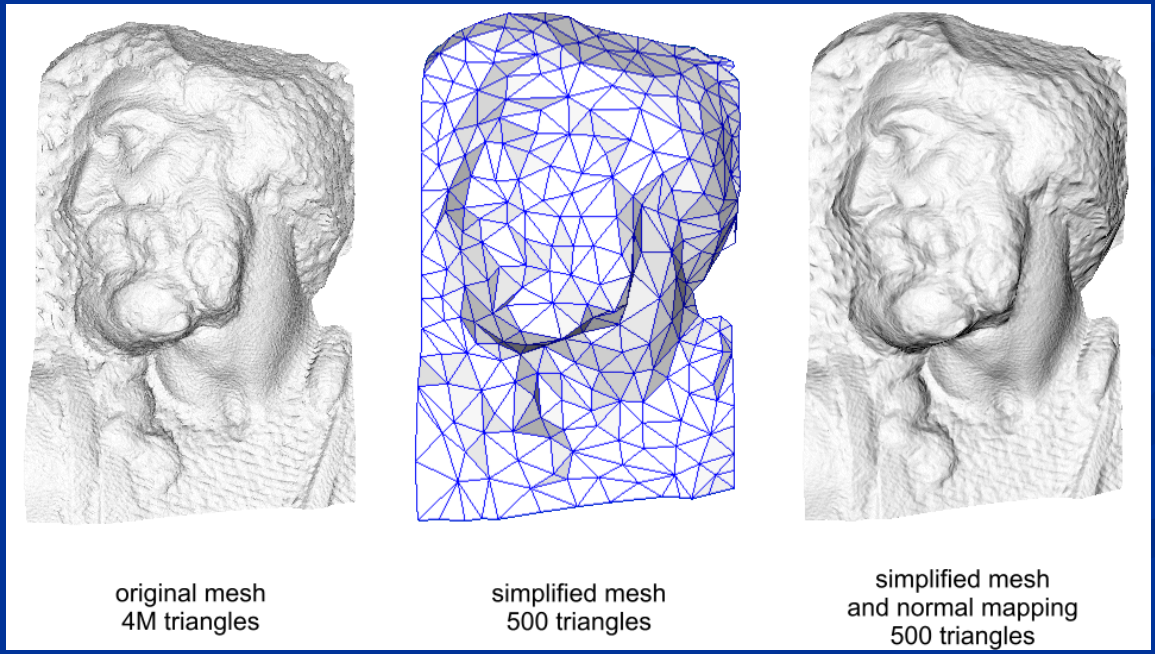
Compute TBN

- Compute Tangent, Bi-Normal for each face.
- Average T,B from adjacent faces to find vertex T/B.
- Make sure they are orthogonal and unit length:
  - B' = B – (B dot T)*T
  - unitize B' and T
  - N = B' cross T



Normal Map Results

- At fragment shader :
  - Take normal vector (nx,ny,nz) from normal map.
  - Map to correct range (Image : 0~1, Normal : -1~1).
  - New normal = nx*T + ny*B + nz*N



More than Normal Mapping

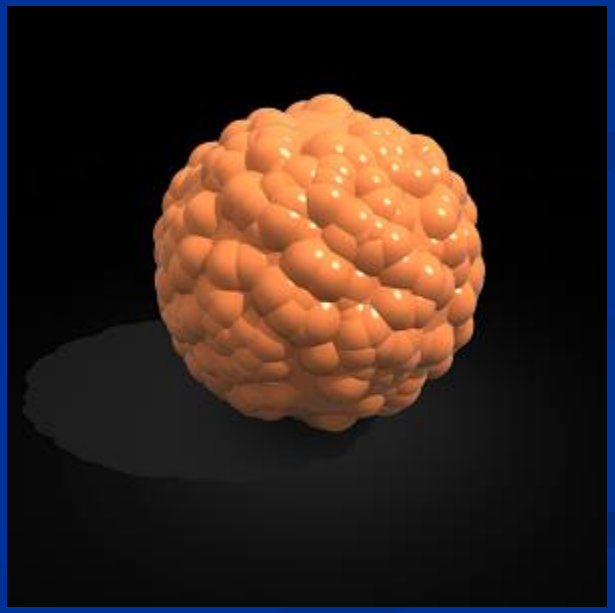
- Normal map can fake shading details, but geometry is the same.
  - No occlusion
  - Smooth silhouette at object.
  - Recent improvement :
    - Displacement mapping
    - Parallax Mapping

Displacement Map

- Change vertex positions in addition to normal.

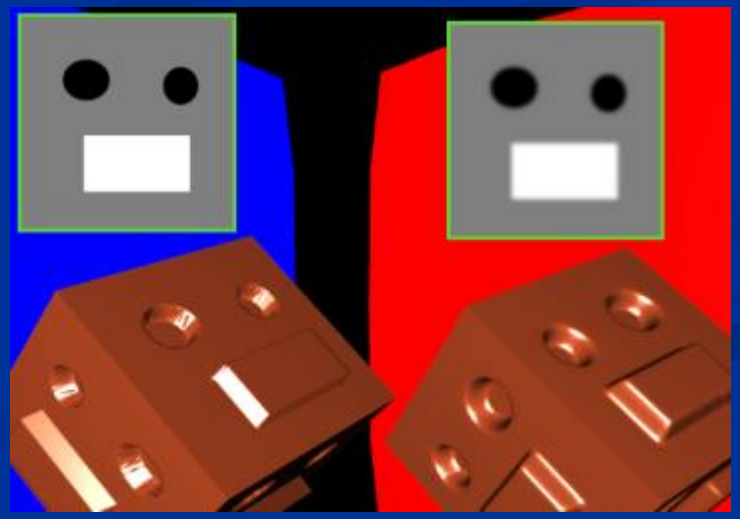
Normal Map

Displacement Map



Displacement Map

- How displacement map works :
  - Subdivide input mesh into suitable detail
  - Offset vertex position in normal direction.
  - Store the corresponding offset in texture map.
  - Require texture access at Vertex Shader.



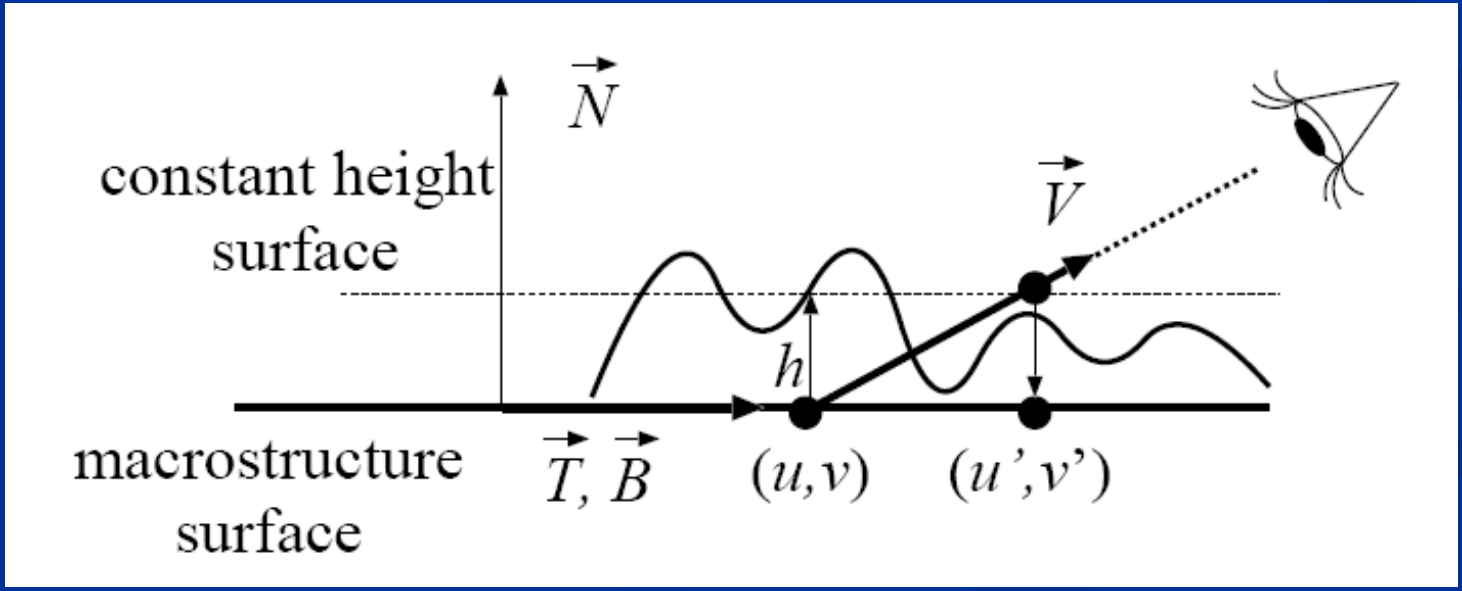
Parallax Map

- Problem with displacement map
  - Can only offset "vertex position" in shader.
  - Quality directly depends on number of vertices
  - Can we do per-pixel displacement ?

Parallax Map in Crysis

Parallax Map

- Not only fake the shading, fake the occlusion.
- Shift texture coordinate based on height map at fragment shader.



Parallax Map

- Problems :
  - Doesn't work as well at grazing angle.
  - No correct silhouette.
  - Artifact for large height scale.

Without Parallax

With Parallax

Q&A

- Summary
  - GLSL Quick Reference
    - http://www.opengl.org/sdk/libs/OpenSceneGraph/glsl_quickref.pdf
  - Normal Map
    - Compute TBN on CPU and pass them as attributes to GPU.
    - Remember to re-normalize T,B,N in fragment shader.