COSE436

Lecture 16:
Texture Mapping II

Nov 07, 2024

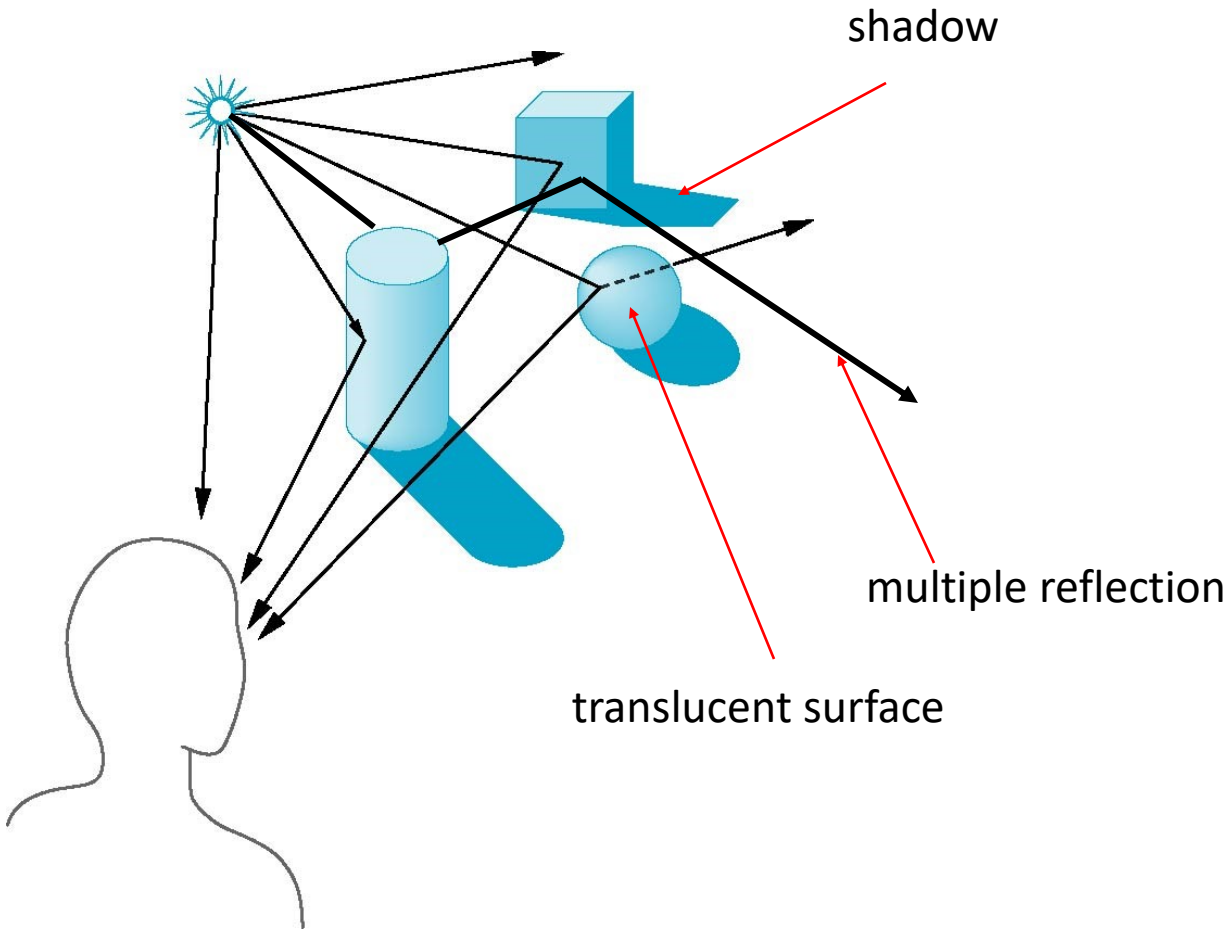
Won-Ki Jeong
(wkjeong@korea.ac.kr)

KOREA UNIVERSITY

Outline

- Environment mapping
- Bump Mapping
- OpenGL texture mapping

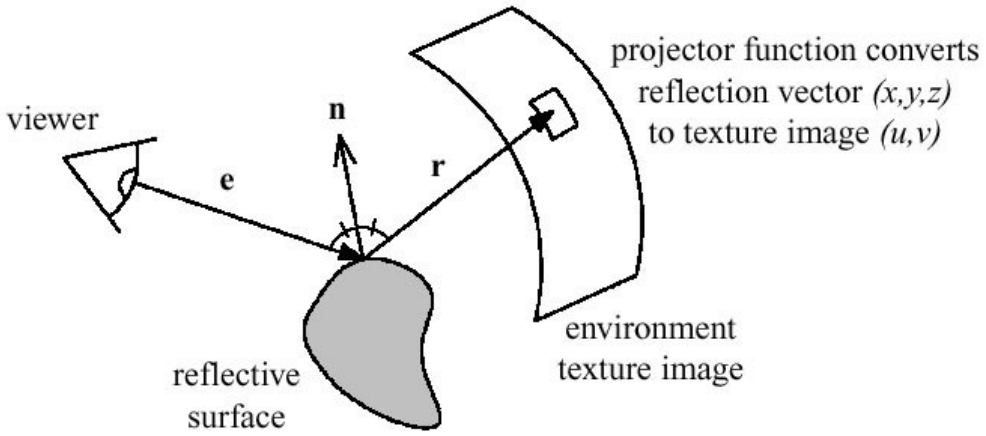
Local vs. Global



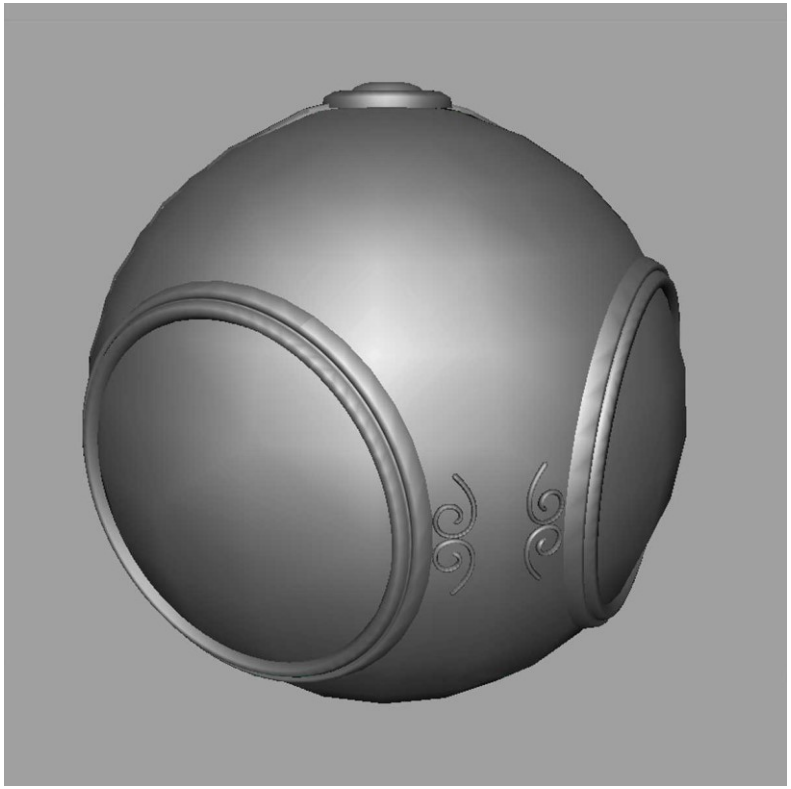
Mirror/Reflective Surface Rendering

Environment Mapping

- How to represent shiny surface reflecting its surroundings in local rendering?
  - Use texture that represents surroundings



Environment Mapping Example



Sphere Mapping

- 360 degree view reflected on a mirror ball

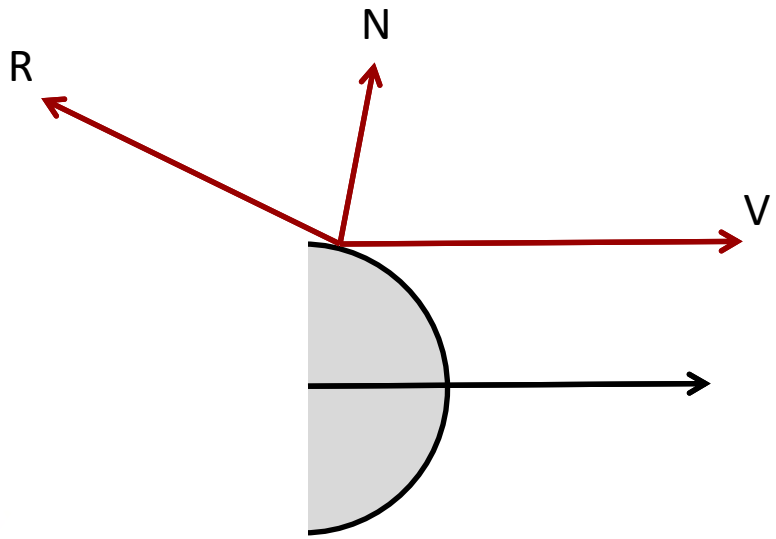


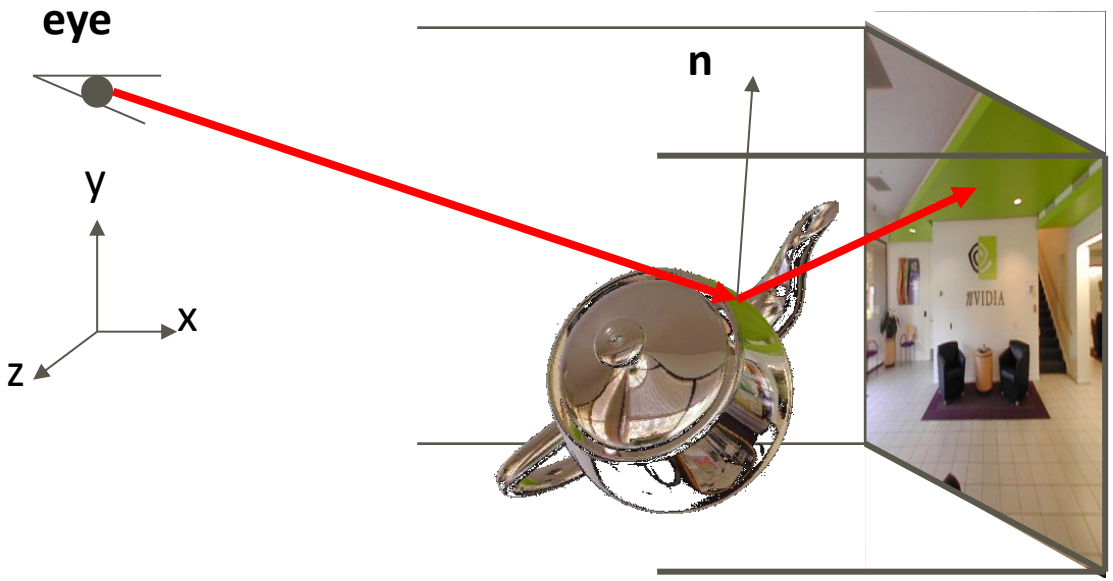
Image Courtesy Bixorama

Sphere Mapping

- Single texture – efficient
- Poor texel distribution per angle
- Distortion near edge
  - Linear interpolation does not work
- View dependent

Cube Mapping

- Sampling from 2D texture per cube side



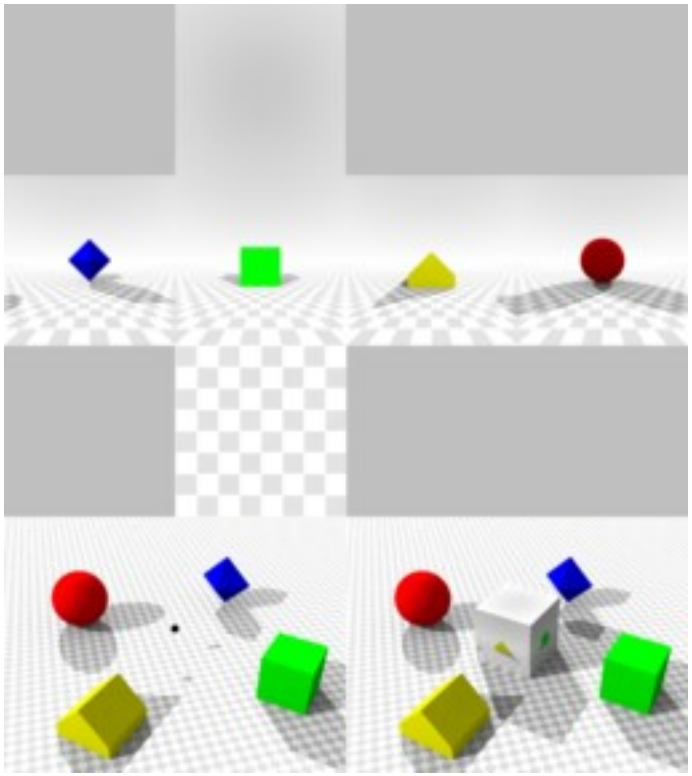
Cube Map Textures

- Create surrounding texture at the center of a cube

Image Courtesy of Akenine-Möller

Generate Cube Map

- Take 6 pictures with 90 degree field of view



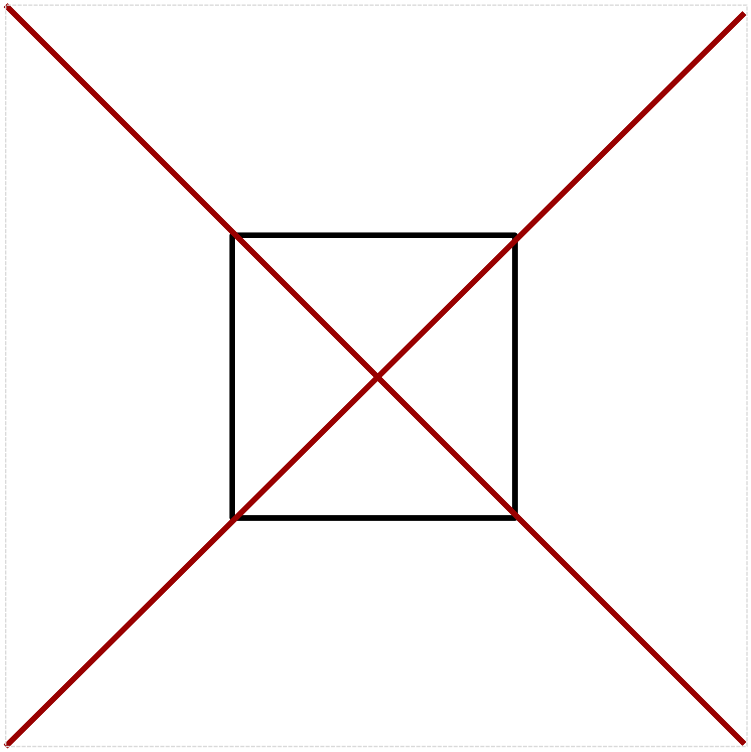
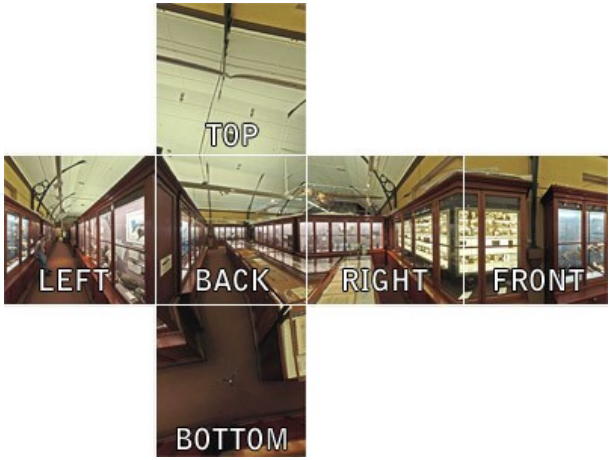


Image Courtesy of Wikipedia

Cube Mapping Example



Cube Map Texture Coordinate

- Compute reflection vector r
- Take largest absolution value of component
  - r=(5,-1,2) then use positive X face
- Divide r by 5
  - (-0.2, 0.4)
- Rescale to [0,1]
  - Add 1 and /2
  - (0.4,0.7)

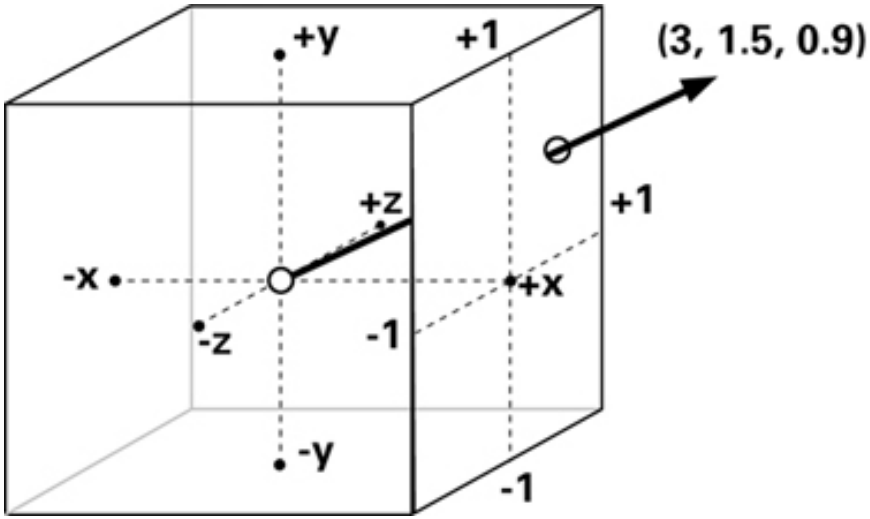


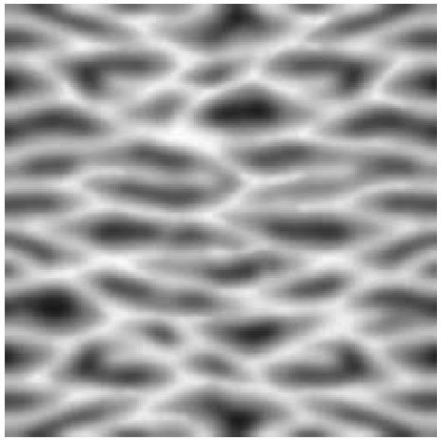
Image Courtesy of Adrian-Florin

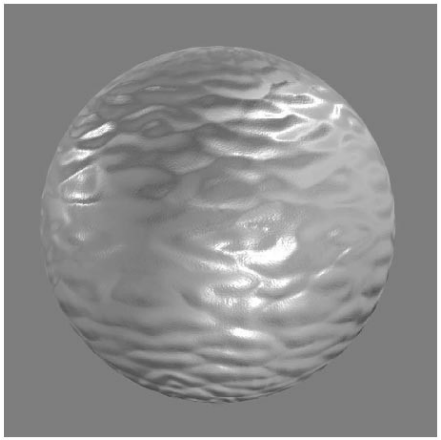
Cube Mapping

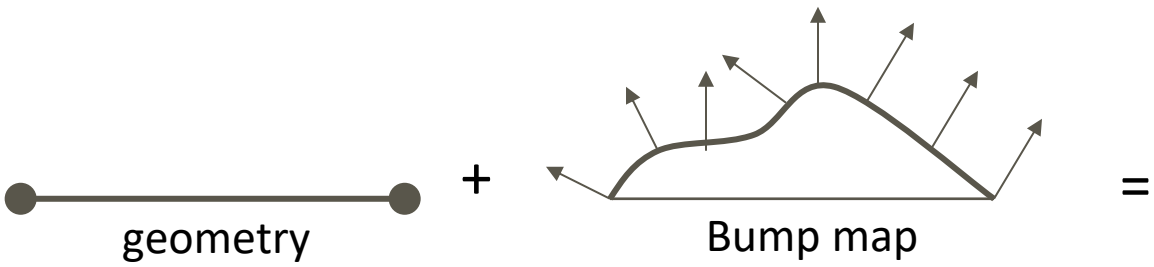
- More memory usage
  - 6 2D textures
- Slow
- Less distortion
- No view dependency
- Default environment mapping for current hardware

Bump Mapping

- Developed by Jim Blinn in 1978
- Simple method to simulate wrinkles and bumps on surfaces
- Use normal texture







Stores heights: can derive normals

Derive Normal from Height

- Finite difference
  - H: height at (i, j)
  - Assume grid spacing is 1 (h = k = 1)

$$f_x \approx \frac{f(x+h,y)-f(x-h,y)}{2h}$$

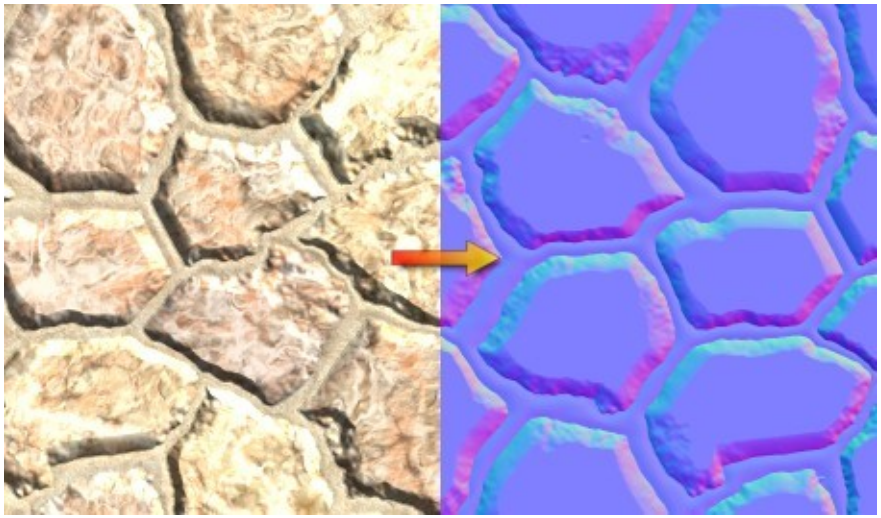
$$f_y \approx \frac{f(x,y+k)-f(x,y-k)}{2k}$$

$$f(i,j)=(i,j,H(i,j))$$

$$n = f_x \times f_y$$

How to Store Normal Map?

- $\mathbf{n}=(n_x, n_y, n_z)$ are in [-1,1]
- Add 1, mult 0.5: in [0,1]
- Multiply by 255
  - 8 bit per color component
- Can store in a RGB texture

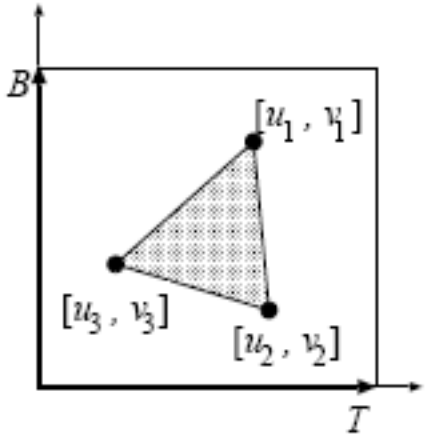


Applying Normal Map

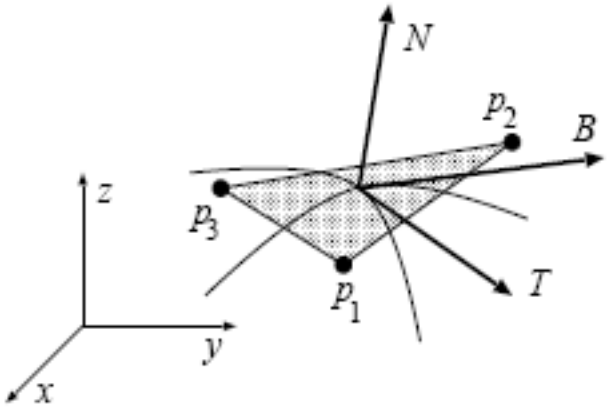
- Normals are defined in local (tangent space) coordinate
  - Relative to local geometry
- We need to do either
  - Transform normal to object(world) coordinate, or
  - Transform light to tangent space coordinate

Applying Normal Map

- Tangent space coordinate
  - Normals are defined in tangent reference frame
  - Tangent frame has to be calculated or stored per vertex and interpolated



tangent space



object (world) space

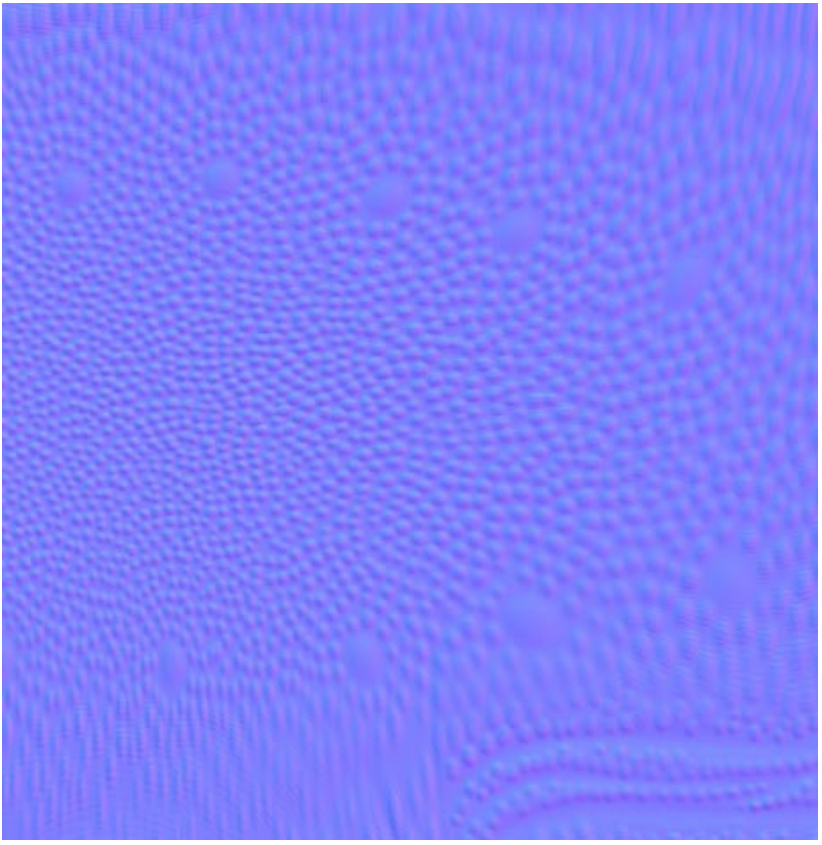
Applying Normal Map

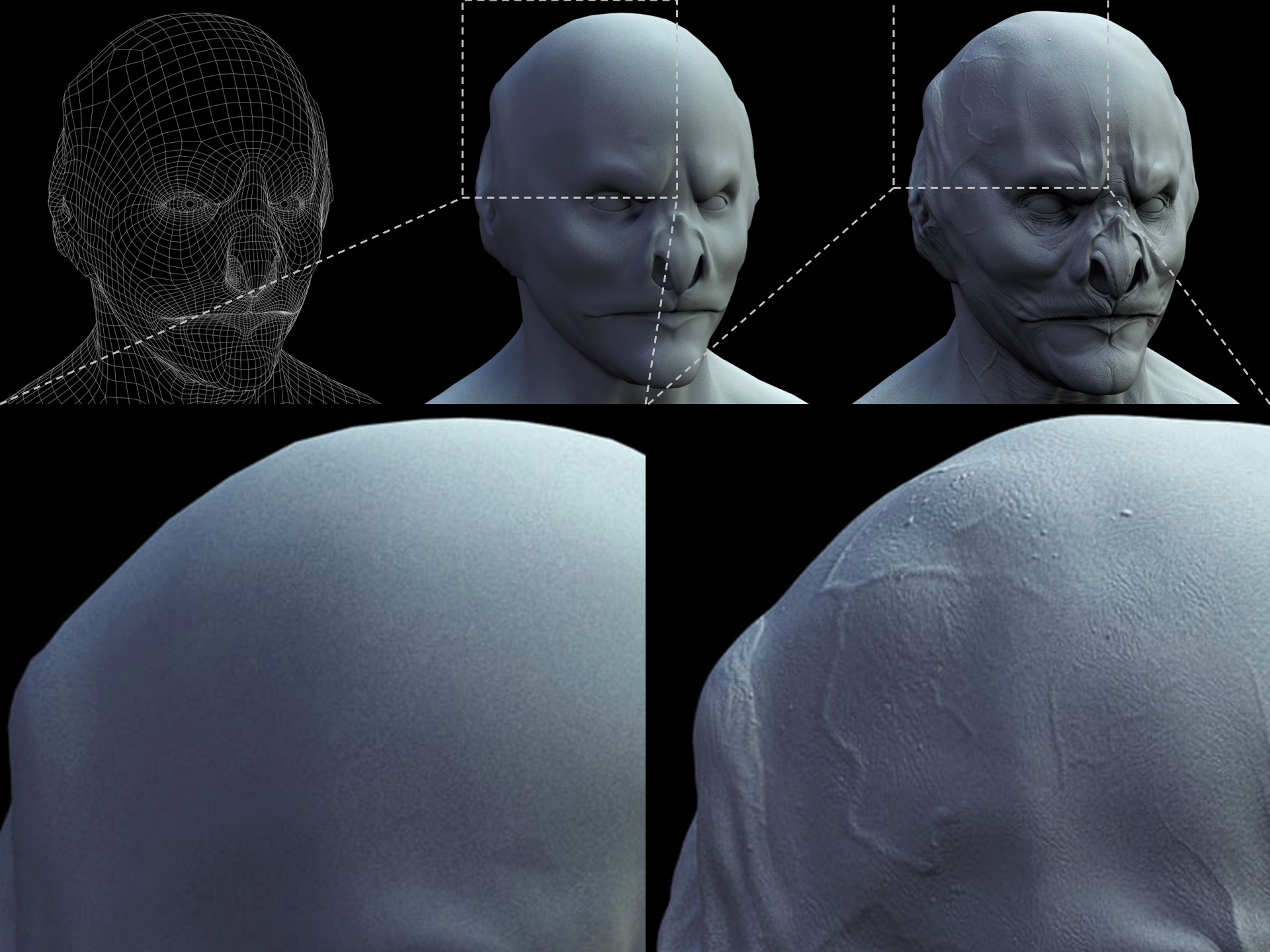
- Transform matrix
  - Transform all normal vectors to world coordinate, or vice versa

$$V_{\text{tangent}} = \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{bmatrix} V_{\text{world}}$$

$$V_{\text{world}} = \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{bmatrix}^{-1} V_{\text{tangent}}$$

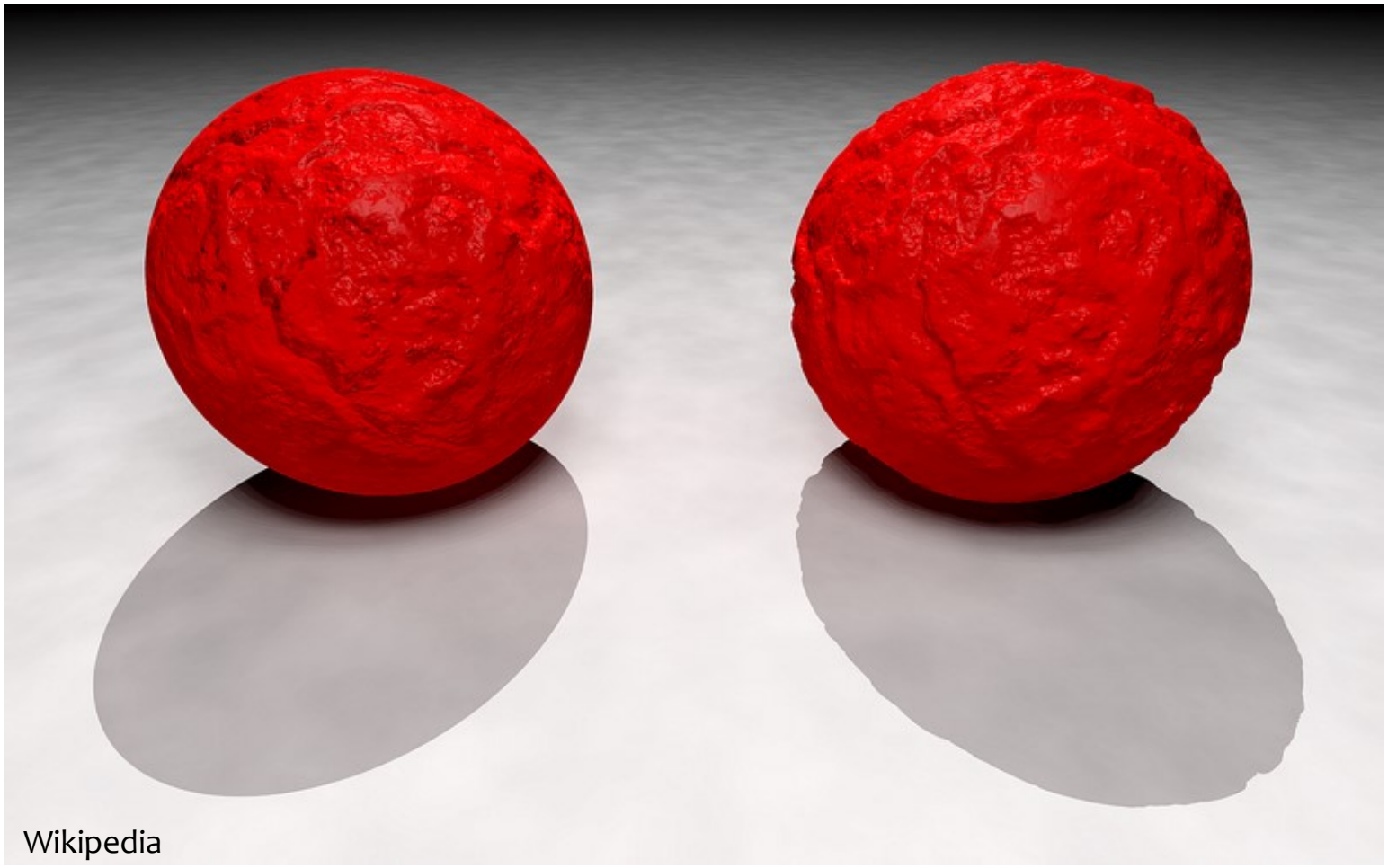
Normal Mapping Example





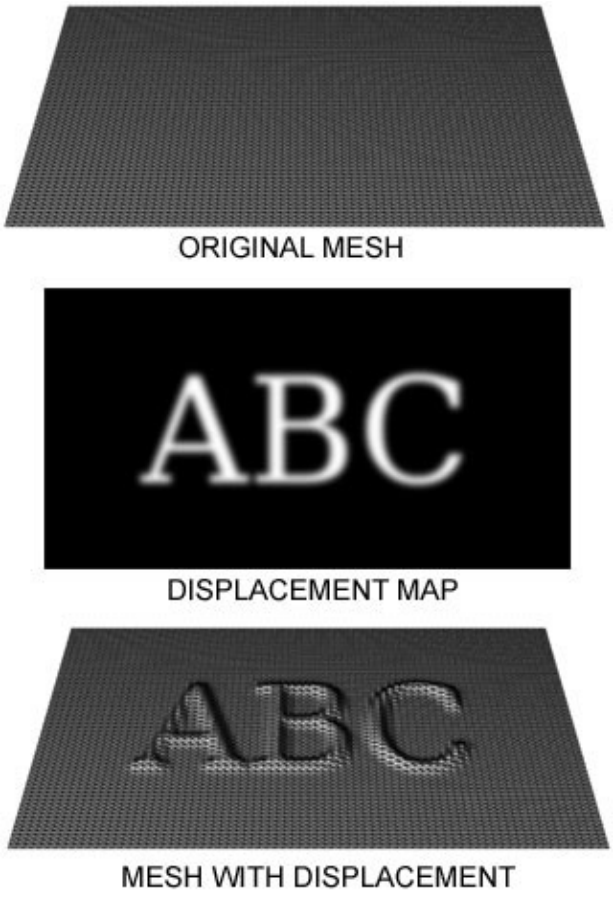
Normal Mapping

- Limitation?

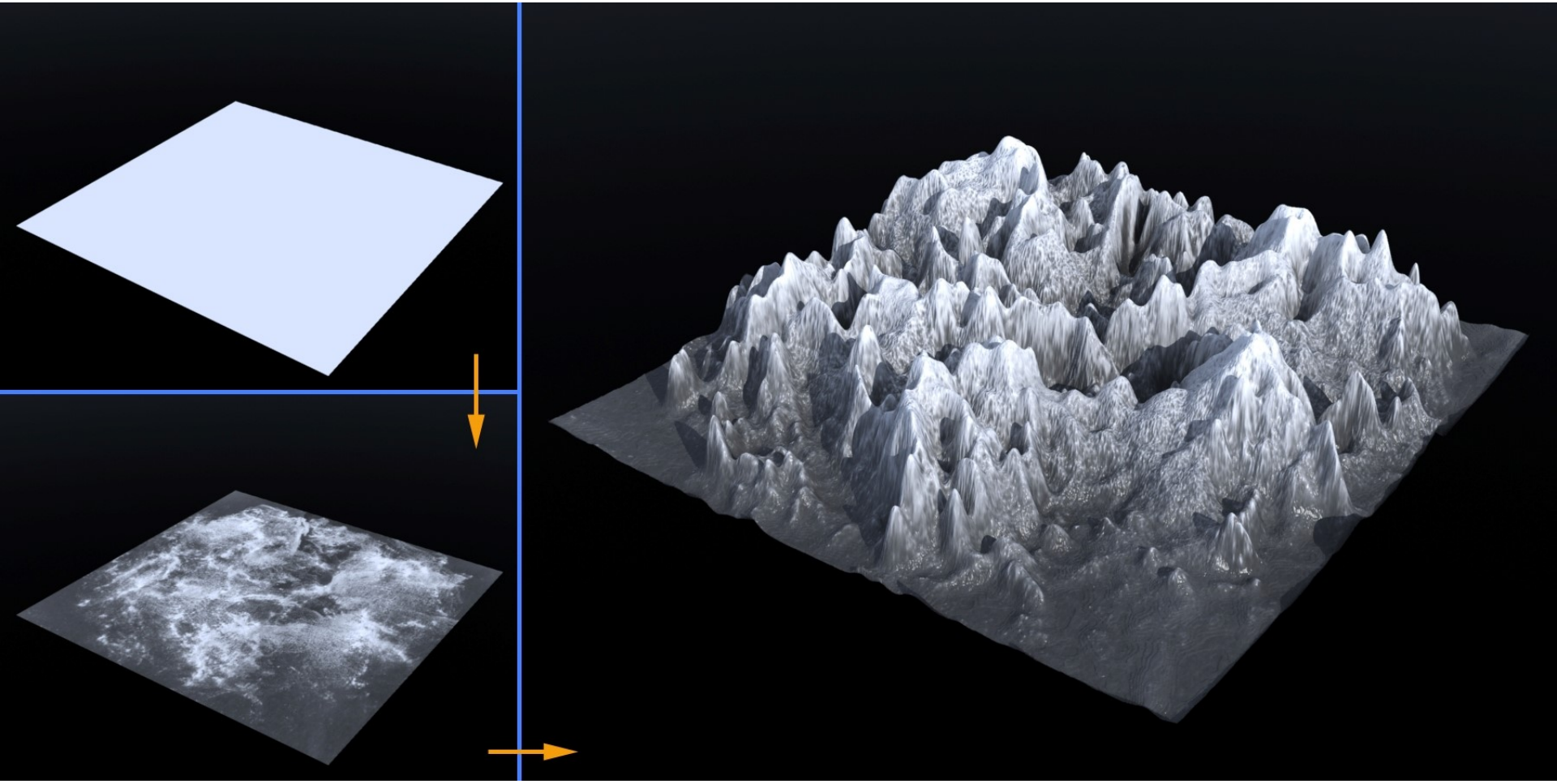


Displacement Mapping

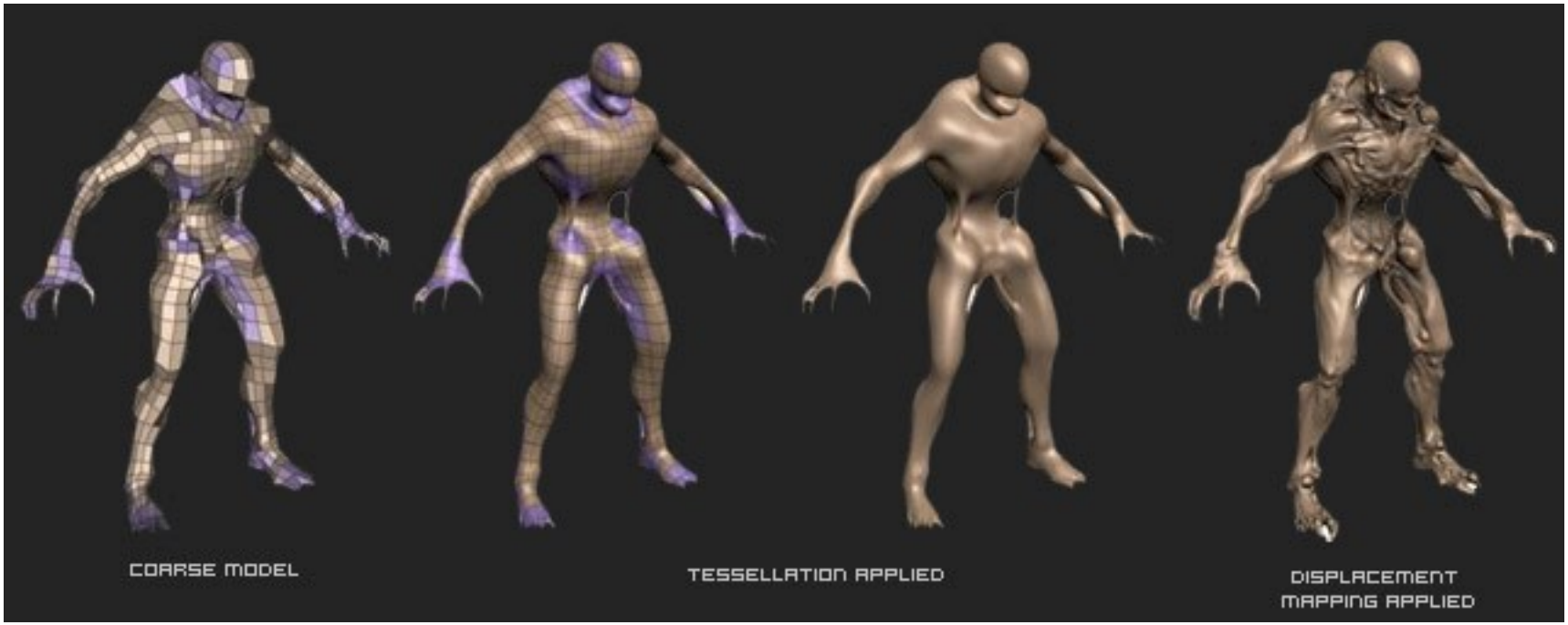
- Altering geometry



Wikipedia



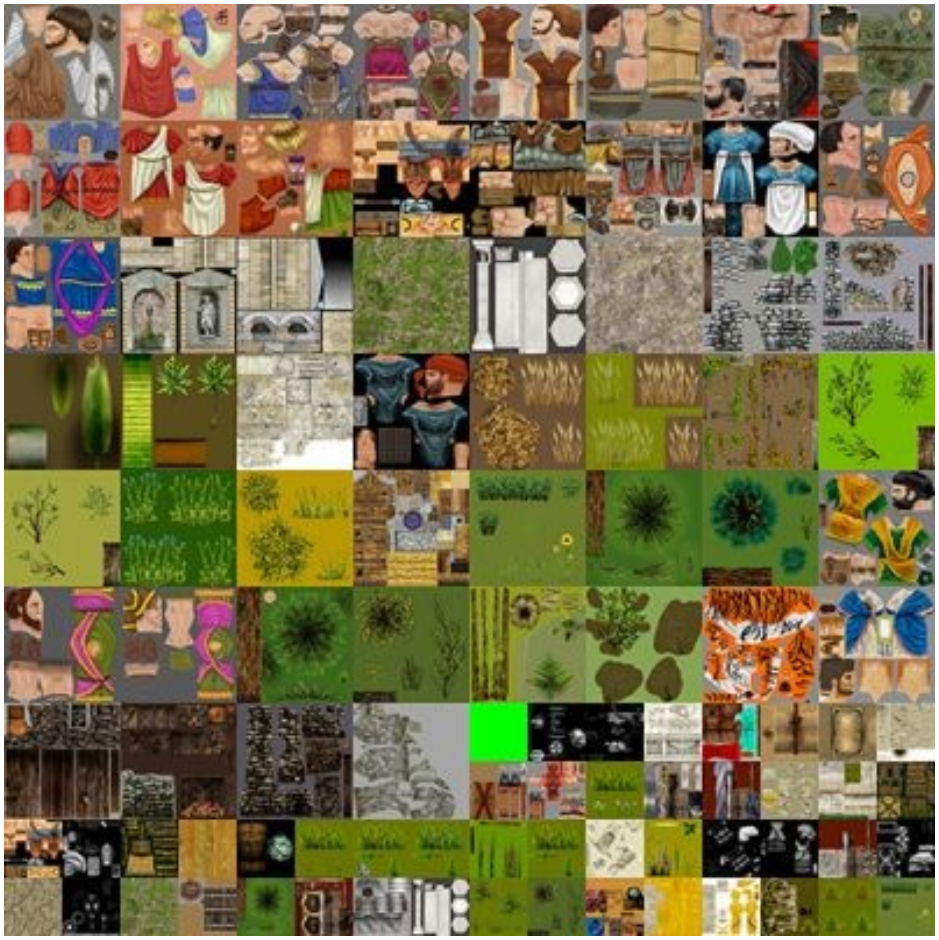
KOREA UNIVERSITY



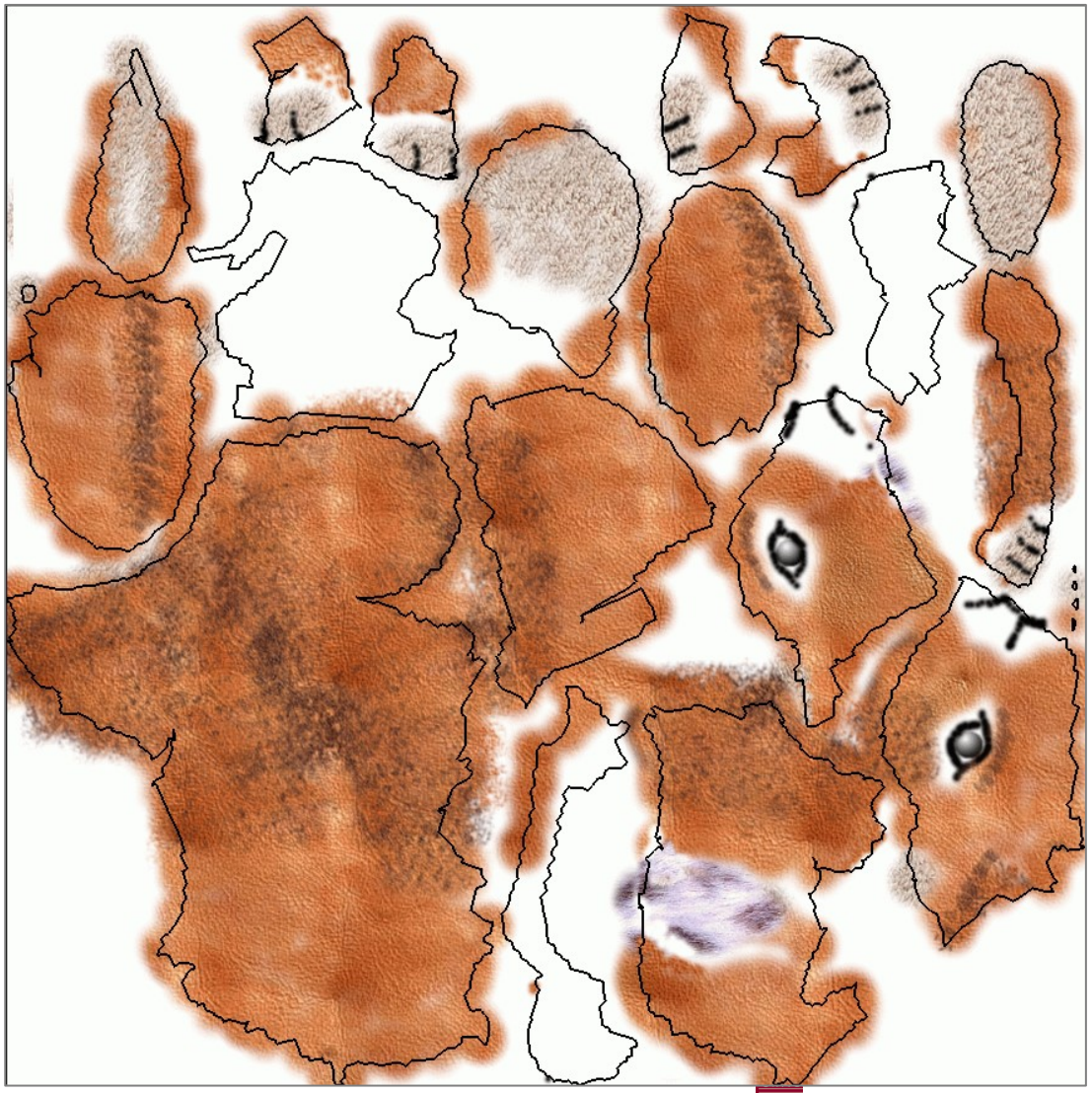
KOREA UNIVERSITY

Texture Atlas

- Combine multiple textures into large one

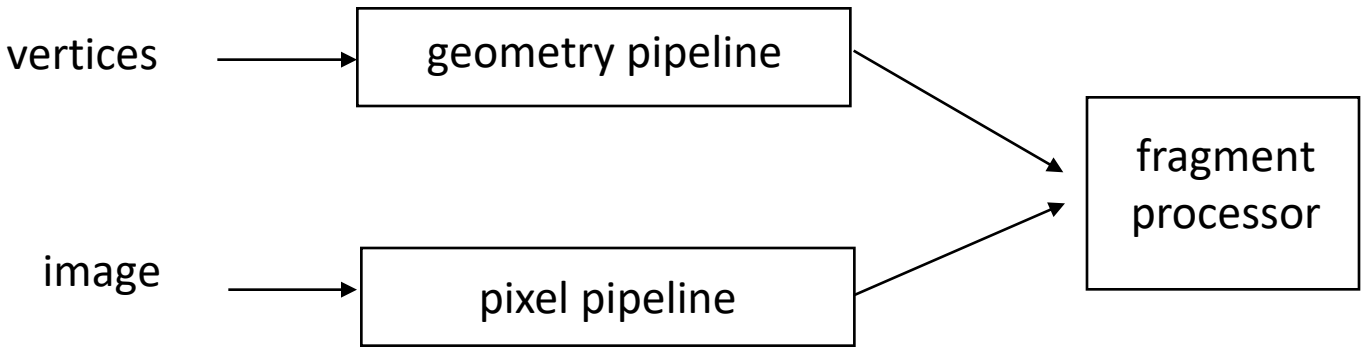


Gamasutra

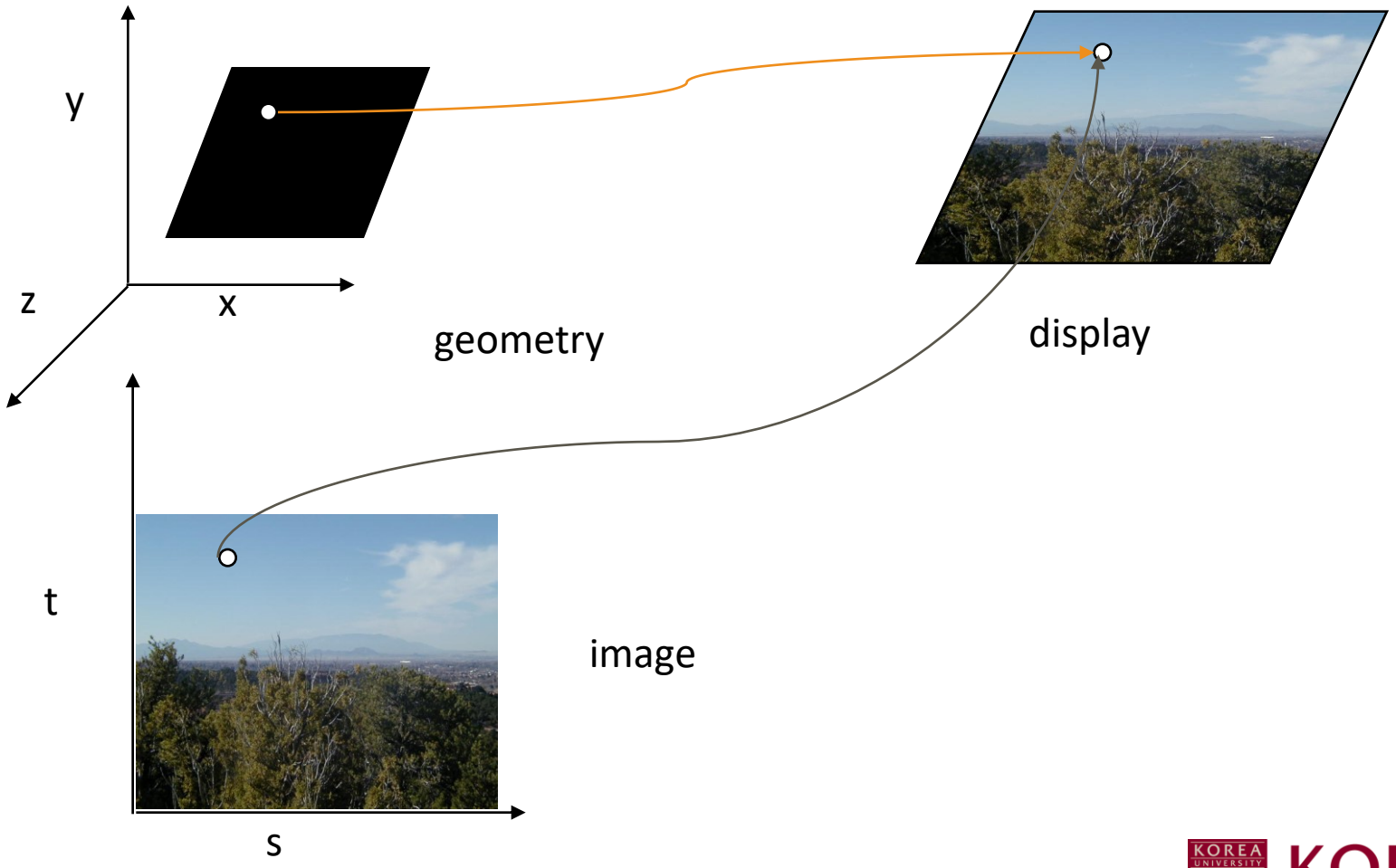


Texture Mapping and GL Pipeline

- Images and geometry flow through separate pipelines that join during fragment processing
  - "complex" textures do not affect geometric complexity

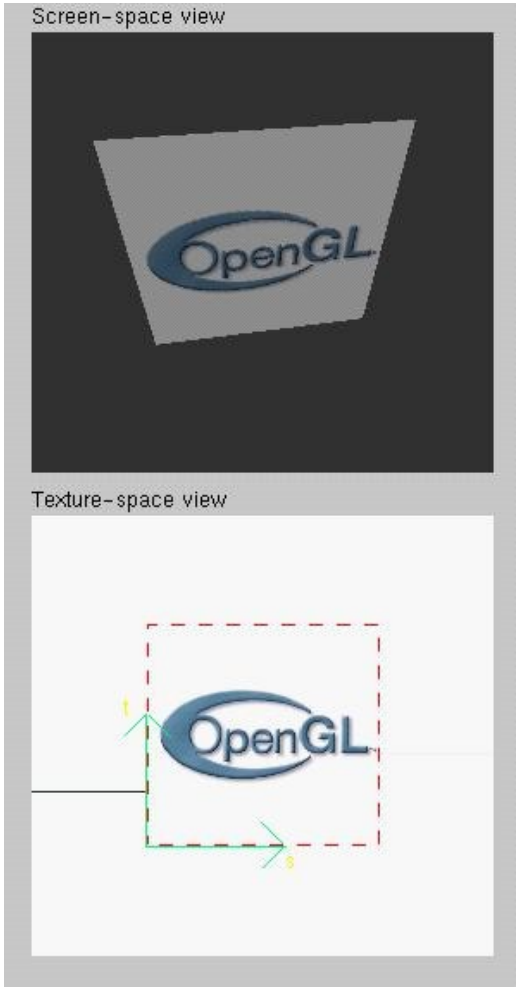


OpenGL Texture Mapping



OpenGL Texture Example

- The texture is a 256 x 256 image that has been mapped to a rectangular polygon which is viewed in perspective



OpenGL Texture Mapping Process

- Create a texture object
  - read or generate raw image
  - assign raw image to texture object
  - Specify texture parameters
- Indicate how texture will be applied to pixel
- Enable texture mapping
- Draw scene with texture and geometric coordinate

Read or Generate Image

- Image needs to be defined as "raw" data for OpenGL
  - 8 bit per pixel

```
Glubyte my_texels[512][512];
```

  - 24 bit per pixel

```
Glubyte my_texels[512][512][3];
```

- Ex) Red color at texel location (2,4)

```
my_texels[2][4][0] = 255;  // red channel
my_texels[2][4][1] = 0;    // blue channel
my_texels[2][4][2] = 0;    // green channel
```

Read Image

- Use image I/O library
  - Get the raw image data from image format
  - BMP, JPG, TIF readers
- Use simple image format
  - PPM, PGM (header + raw data)

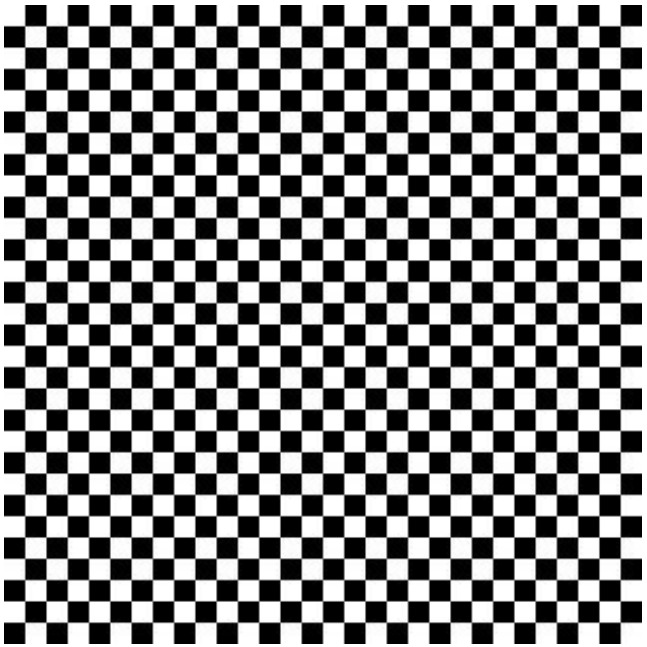
max intensity →

```
P3
# example from the man page
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

Generate Image

- Your program generate image on-the-fly

```
void makeCheckImage(void)
{
   int i, j, c;

   for (i = 0; i < checkImageHeight; i++) {
      for (j = 0; j < checkImageWidth; j++) {
         c = ((((i&0x8)==0)^((j&0x8))==0))*255;
         checkImage[i][j][0] = (GLubyte) c;
         checkImage[i][j][1] = (GLubyte) c;
         checkImage[i][j][2] = (GLubyte) c;
         checkImage[i][j][3] = (GLubyte) 255;
      }
   }
}
```



Generate Texture Name

- glGenTextures(GLsizei n, Gluint* texName)
  - Generate an identifier (name) for a texture object
  - n : the number of the texture
  - texName: array to store texture names

Generate and Bind Texture Object

- glBindTexture(GLsizei target, Gluint texName)
  - Create texture object if not existing
  - Bind the texture to use for texture mapping
  - Unbind if texName is 0
  - target
    - GL_TEXTURE_1/2/3D, GL_TEXTURE_CUBE_MAP, ...
  - textName
    - Texture name (returned from glGenTextures)

Example

```
static GLuint texName[2];
glGenTextures(2, texName); // get two texture names

glBindTexture(GL_TEXTURE_2D, texName[0]);

...  // set up texture 1

glBindTexture(GL_TEXTURE_2D, texName[1]);

...  // set up texture 2
```

Define Texture

```
void glTexImage2D(GLenum target, GLint level, GLint internalFormat, Glsizei width,
GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid* texels);
```

- Defines a texture image on 2D
  - *target*
    - GL_TEXTURE_2D Only
  - *level*
    - Level of detail. 0 if there are no Mipmaps
  - *internalFormat*
    - format of texel data
  - *width, height*
    - Width and height of the texture image
  - *Border*
    - The width of the border
  - *format*
    - Format of pixel data
  - *type*
    - Data type of the pixel data
  - *texels*
    - A pointer to the image data in memory

Texture Format and Type

- Internal Format constants

The name of constants	meaning
GL_RGBA	R, G, B, A
GL_RGB	R, G, B
GL_ALPHA	A
GL_LUMINANCE	Intensity
GL_LUMINANCE_ALPHA	Intensity, A

- Type constants

The name of constants	meaning
GL_UNSIGNED_BYTE	unsigned 8 bit integer
GL_BYTE	signed 8 bit integer
GL_UNSIGNED_SHORT	unsigned 16 bit integer
GL_SHORT	signed 16 bit integer
GL_INT	signed 32 bit integer
GL_FLOAT	single precision float

- Format constants

The name of constants	meaning
GL_COLOR_INDEX	Color index
GL_RGB	R, G, B value
GL_RGBA	R, G, B, value
GL_RED	Red value
GL_GREEN	Green value
GL_BLUE	Blue value
GL_ALPHA	Alpha value

Example

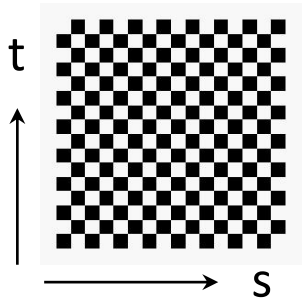
```
GLubyte myimage[64][64][3];

... // filling myimage with raw image data

static GLuint texName;
glGenTextures(1, &texName);
glBindTexture(GL_TEXTURE_2D, texName); // create & bind

glEnable(GL_TEXTURE_2D);
glTexImage2D(GL_TEXTURE_2D,
             0,
             GL_RGB,
             64, 64, 0,
             GL_RGB, GL_UNSIGNED_BYTE, myimage);
```

void glTexImage2D(GLenum target, GLint level, GLint internalFormat, Glsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid* texels);

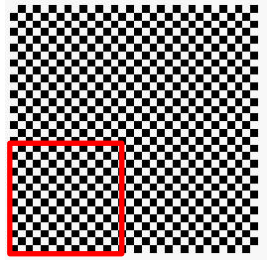
Wrapping Mode

- Clamping: if s,t > 1 use 1, if s,t <0 use 0
- Wrapping: use s,t modulo 1

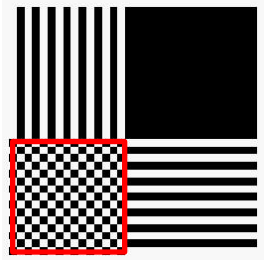
```
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_S, GL_CLAMP )
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_T, GL_REPEAT )
```



texture



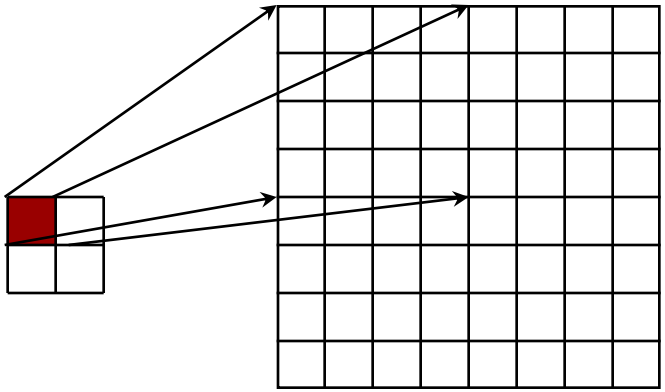
GL_REPEAT
wrapping



GL_CLAMP
wrapping

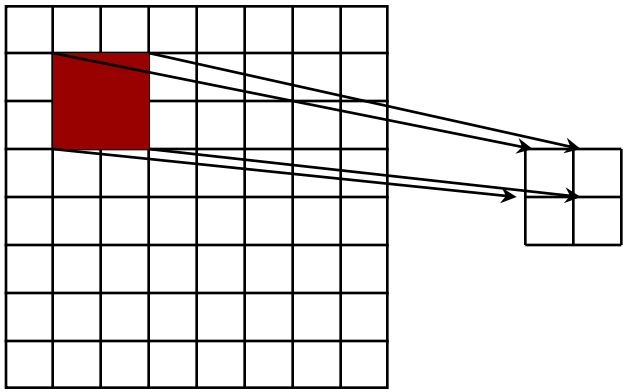
Magnification and Minification

- More than one texel can cover a pixel (*minification*) or more than one pixel can cover a texel (*magnification*)
- Can use point sampling (nearest texel) or linear filtering(2 x 2 filter) to obtain texture values



Texture Polygon

Magnification



Texture Polygon

Minification

Texture Filtering

- glTexParameterf(GLenum target, GLenum pname, GLfloat param)
  - Target
    - GL_TEXTURE_1/2/3D
  - pname
    - GL_TEXATURE_MAG/MIN_FILTER
  - param
    - Parameter how to sample texture

Filtering Parameters

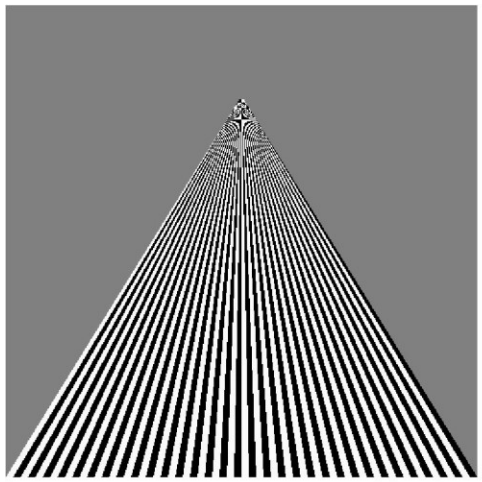
- GL_NEAREST
- GL_LINEAR
- GL_NEAEREST_MIPMAP_NEAREST
- GL_NEAEREST_MIPMAP_LINEAR
- GL_LINEAR_MIPMAP_NEAREST
- GL_LINEAR_MIPMAP_LINEAR

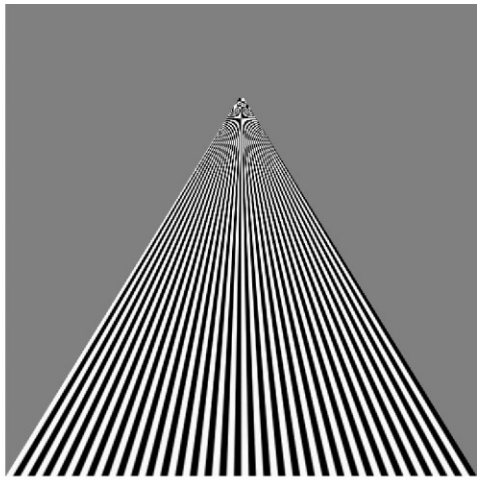
filtering within level

filtering across level

Example

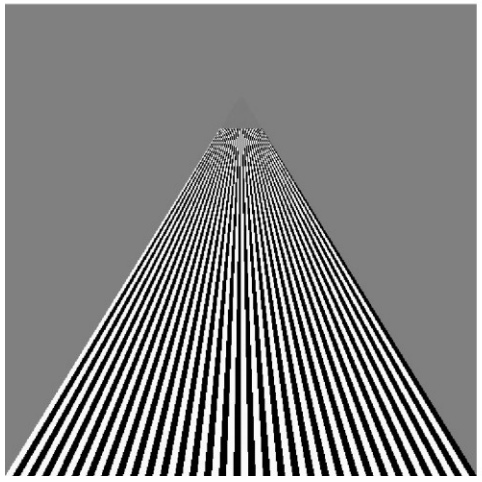
nearest sampling

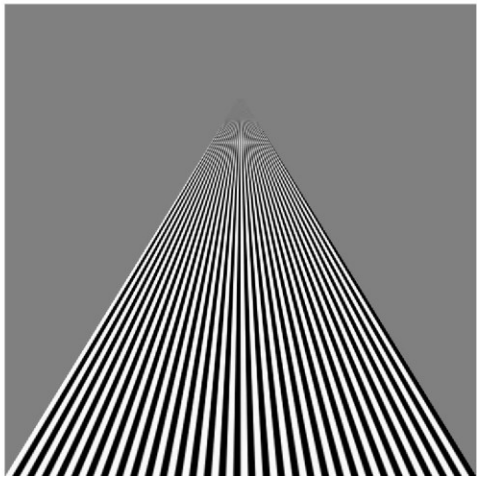




linear filtering

mipmapped nearest sampling





mipmapped linear filtering

Cleaning Up Texture Objects

- Free texture memory
- glDeleteTextures(n, *texNames)
  - Delete n texture objects given by texNames
  - Freed texture names can be reused

Access Texture in Shader

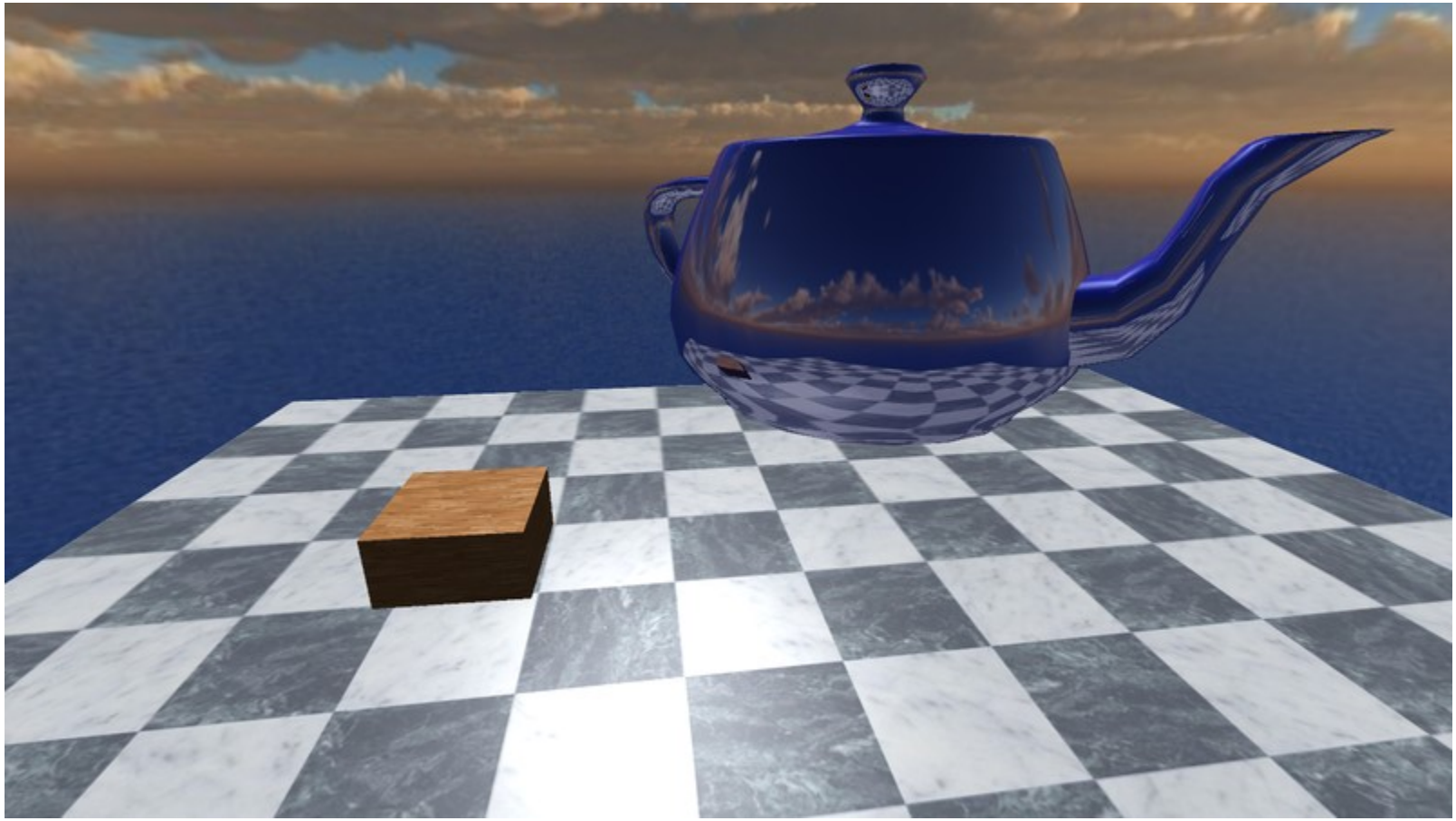
- Textures are applied during fragments shading by a **sampler**
- Samplers return a texture color from a texture object

```
in vec4 color;    //color from rasterizer
in vec2 texCoord; //texure coordinate from rasterizer
uniform sampler2D texture; //texture object from application

void main()  {
    gl_FragColor = color * texture2D( texture, texCoord );
}
```

Other Techniques

- 3D textures
  - Supported on modern GPUs
  - Texture filtering is quadlinear (trilinear + mipmap)
- Multitexturing
  - More than one set of textures per vertex
- Float texture format

Questions?



Wikipedia