Texture Mapping – Part 2

Frederick Li

Third Year Project

- My research interests include:

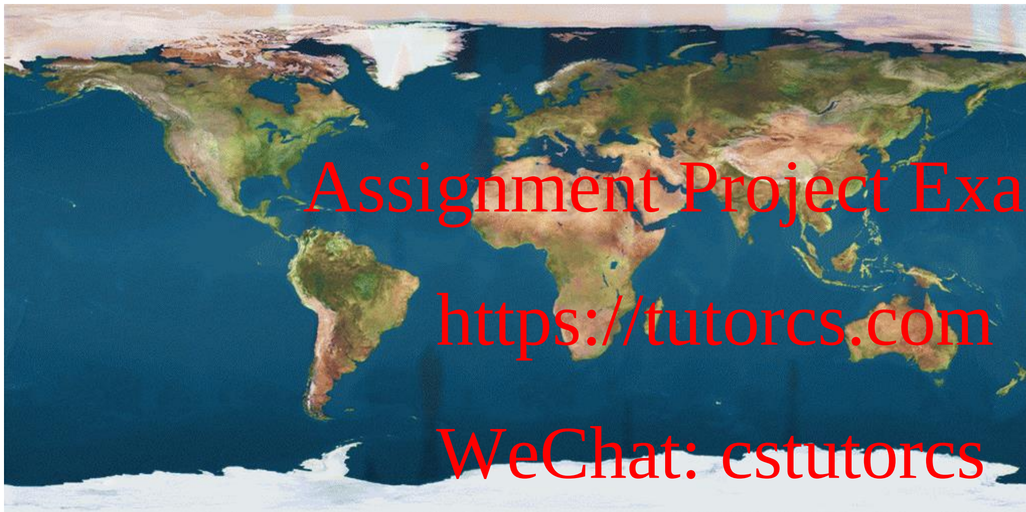
  - Computer graphics
  - Virtual reality
  - E-learning systems
  - Computer gaming, and
  - 3D modelling with machine learning

This Lesson

- Introduction to texture mapping
- Mapping Methods
  - Forward and backward mapping
  - Two-part mapping
- WebGL Implementation
- Optimization: Mip-mapping
- Applications of texture mapping
  - Bump mapping, Normal mapping, Displacement mapping, Environment mapping, Light mapping, Fog mapping

Texture Image Size

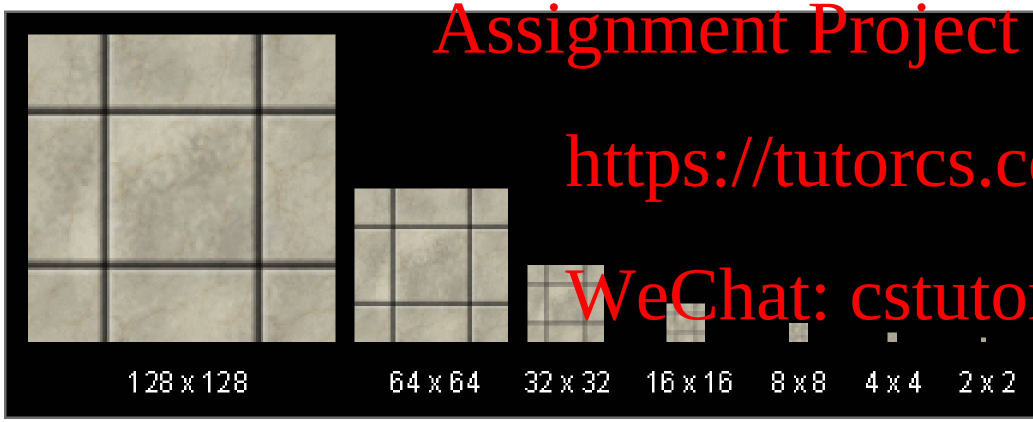
- What is the size of a texture map?



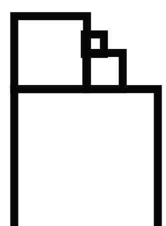
- Resolution: 1024 x 512 pixels
- Memory size occupied = 16 MB

Optimization: MIP-Mapping

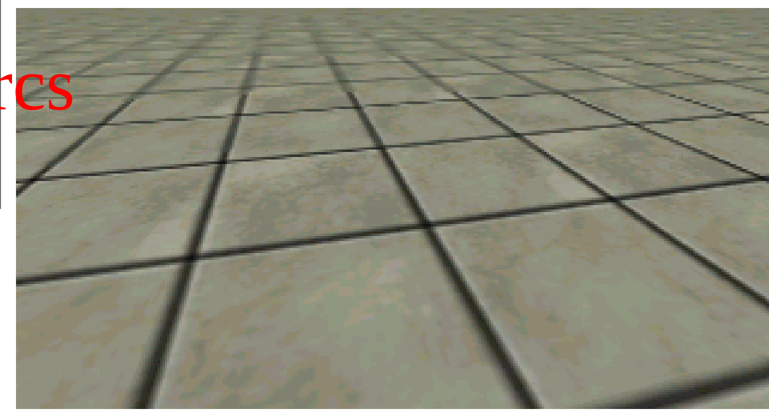
Use "image pyramid" to precompute coarse versions of a texture



store whole pyramid in single block of memory



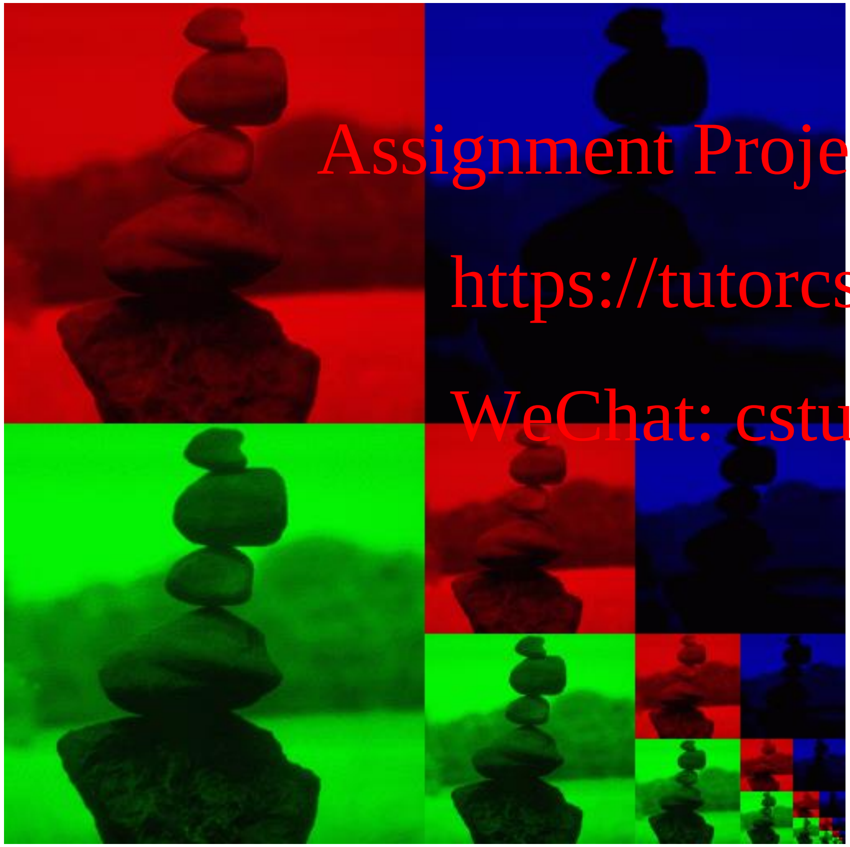
Aliasing problem: Without MIP-mapping



Problem solved: MIP-mapping allows properly sampled images to be used, avoiding over-sampling problem

MIP-Mapping

- Storage: Only 1/3 more space required



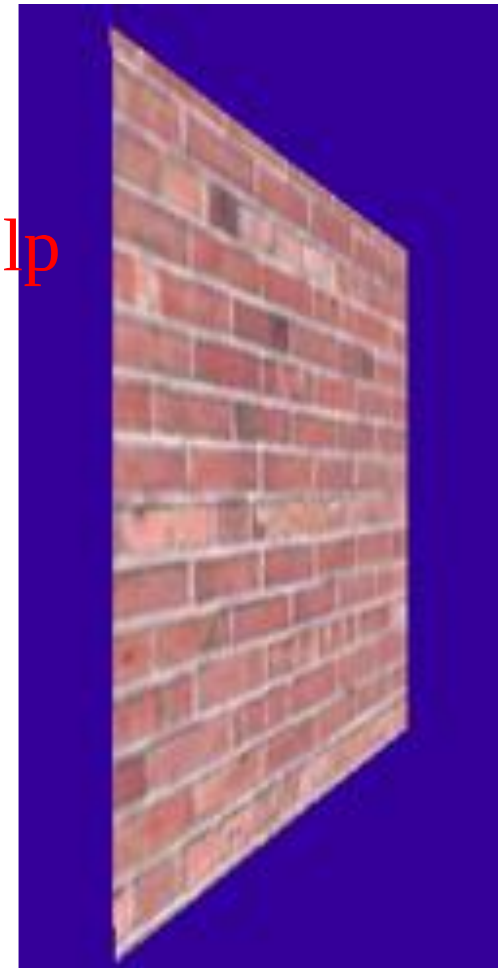
Advantages of MIP-mapping:

- Reduce memory consumption of running applications
- Support anti-aliasing, offering better output quality of a CG application

What's Missing with Texture Mapping?

- What's the difference between a real brick wall and a photograph of the wall texture-mapped onto a plane?

- What happens if we change the lighting or the camera position?



Normal mapping

- Normal vectors encoded as an image
  - Generate visually 3D effect by applying lighting to perturbed normal vectors on the object surface

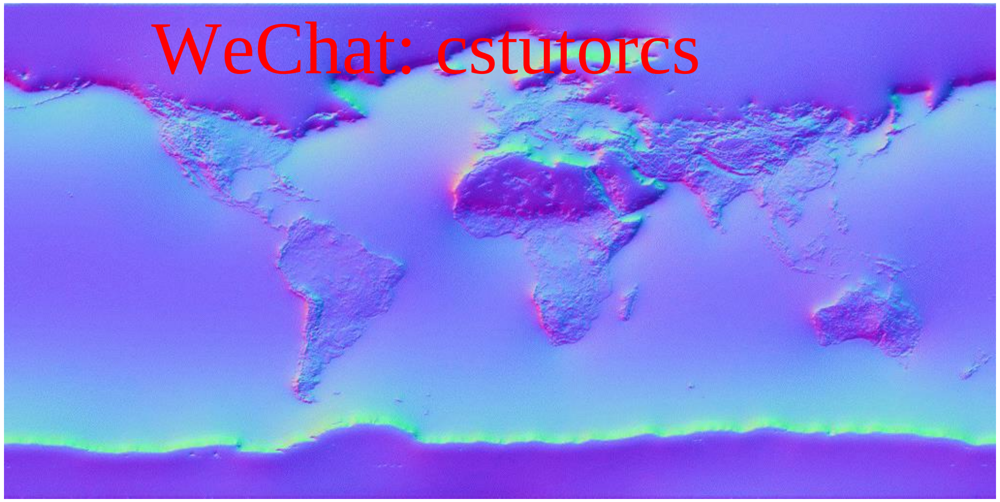
normal map

with normal mapping actual geometry

Advantage

- Use textures to alter the surface normal
  - Does not change the actual shape of the surface, particularly does not increase geometry complexity, i.e. do not impose significant performance overhead
  - Just shaded as if it were a different shape, producing visually pleasing results

Texture and Normal Maps



Normal Mapping Implementation

- Load color and normal maps:

```
earthTexture.image.onload = function () {earthNormalMap
   handleLoadedTexture(earthTexture, sp.samplerUniform, 0);}
earthNormalMap.image.onload = function () {
   handleLoadedTexture(earthNormalMap, sp.samplerUniform_normal, 1);}

earthTexture.image.src = "earthmap1k.gif";
earthNormalMap.image.src = "earth.normalmap.gif";
```

import color and normal maps

```
// Load color map and normal map into the program texture image buffers
function handleLoadedTexture(texture, uSampler, texUnit) {
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);

    if (texUnit == 0) {
        gl.activeTexture(gl.TEXTURE0);
        g_texUnit0 = true;
    } else {
        gl.activeTexture(gl.TEXTURE1);
        g_texUnit1 = true;
    }

    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, texture.image);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_NEAREST);
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.uniform1i(uSampler, texUnit);
}
```

generate color & normal maps with mip-mapping enabled.

Vertex Shader

- Mainly take the inputs and hand over them to rasterization for interpolation

```
//Transformed vertex position
vec4 vertex = uMVMatrix * vec4(aVertexPosition, 1.0);

//Transformed normal position
vec3 normal = vec3(uNMatrix * vec4(aVertexNormal, 1.0));
//light direction, from light position to vertex
vec3 lightDirection = uPointLightingLocation - vertex.xyz;

//eye direction, from camera position to vertex
vec3 eyeDirection = -vertex.xyz;

//Final vertex position
gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
vTextureCoord = aTextureCoord;
vLightDir = lightDirection;
vEyeDir = eyeDirection;
```

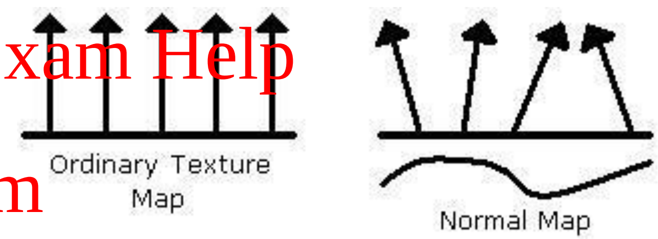
Fragment Shader

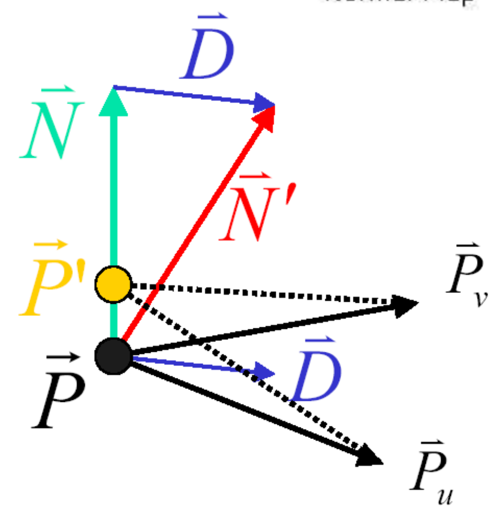
- Unpack normal vectors from the normal map and apply lighting accordingly

```
// Unpack normal from texture
vec3 normal = normalize(2.0 * (texture2D(uSampler_normal, vTextureCoord).rgb - 0.5));

// Normalize the light direction and determine how much light is hitting this point
vec3 lightDirection = normalize(vLightDir);
float lambertTerm = max(dot(normal, lightDirection), 0.20);

// Calculate Specular level
vec3 eyeDirection = normalize(vEyeDir);
vec3 reflectDir = reflect(-lightDirection, normal);
float Is = pow(clamp(dot(reflectDir, eyeDirection), 0.0, 1.0), 12.0);

// Combine lighting and material colors
vec4 Ia = vec4(uAmbientColor, 1.0);
vec4 Id = vec4(uPointLightingColor, 1.0) * texture2D(uSampler, vTextureCoord) * lambertTerm * 1.8;

gl_FragColor = Ia + Id + Is * 0.5;
```

Bump Mapping

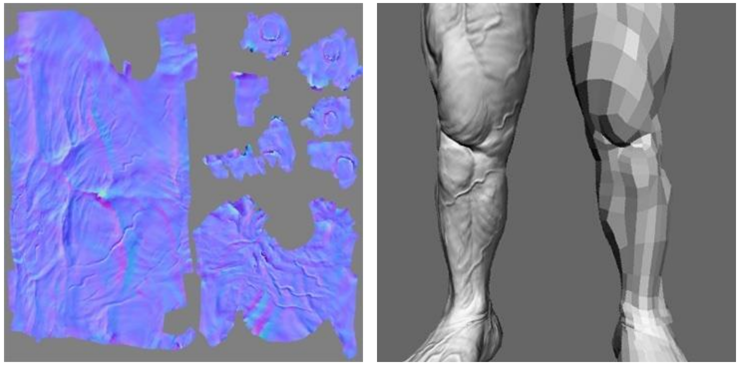
- Treat the texture as a single-valued height function
- Compute the normal from the partial derivatives in the texture
- The heights encode the amount by which to perturb **N** in the (u,v) directions of the parametric space describing the object surface





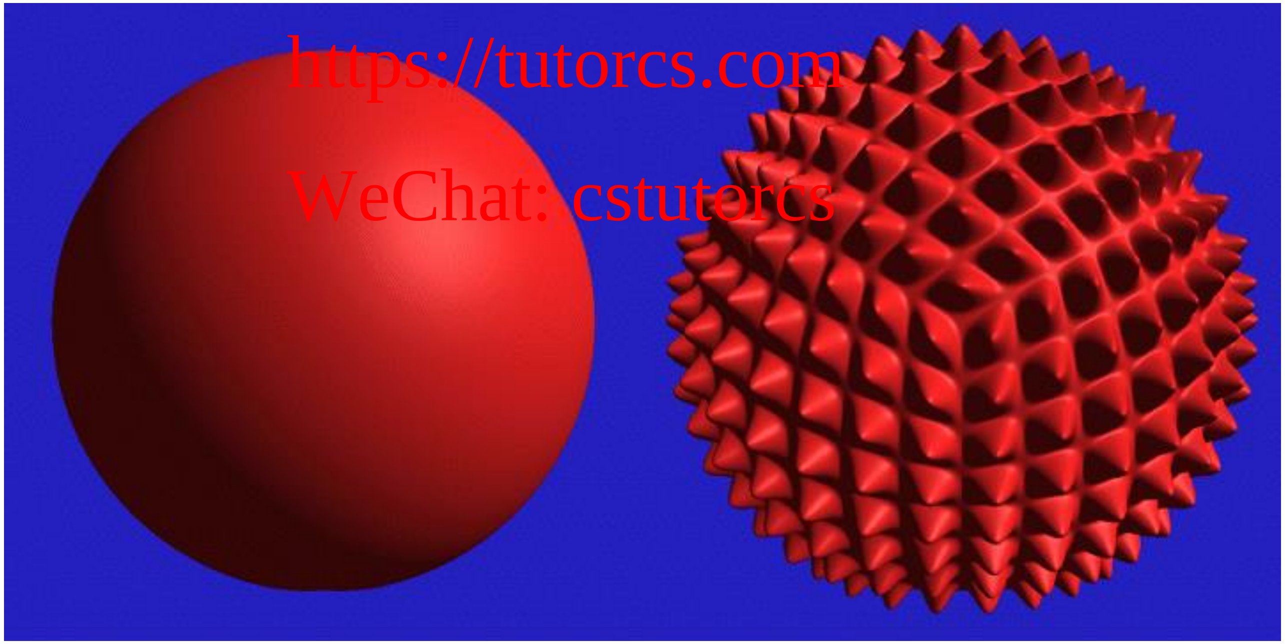
Normal map vs. Bump map

- Bump map
  - texture (greyscale) encodes height
  - Modifies the geometric normal
  - Harder to implement
  - Easier to specify
- Normal map
  - texture (RGB) encodes normal directly
  - Replaces the normal
    - but local coordinates
  - Easier to implement
  - Harder to specify



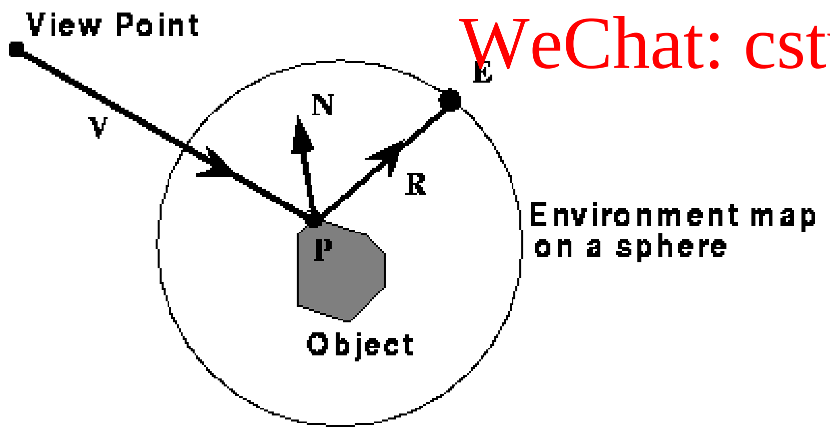
Displacement Mapping

- Use texture map to actually move surface points
- Geometry must be displaced before visibility is determined
- Done as a preprocess or with complicated vertex/fragment shader implementation



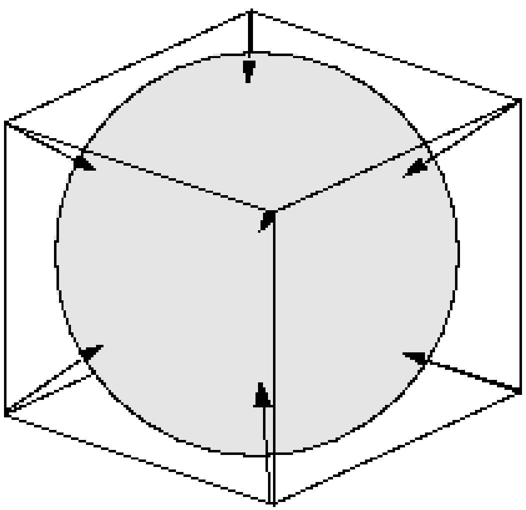
Environment Maps

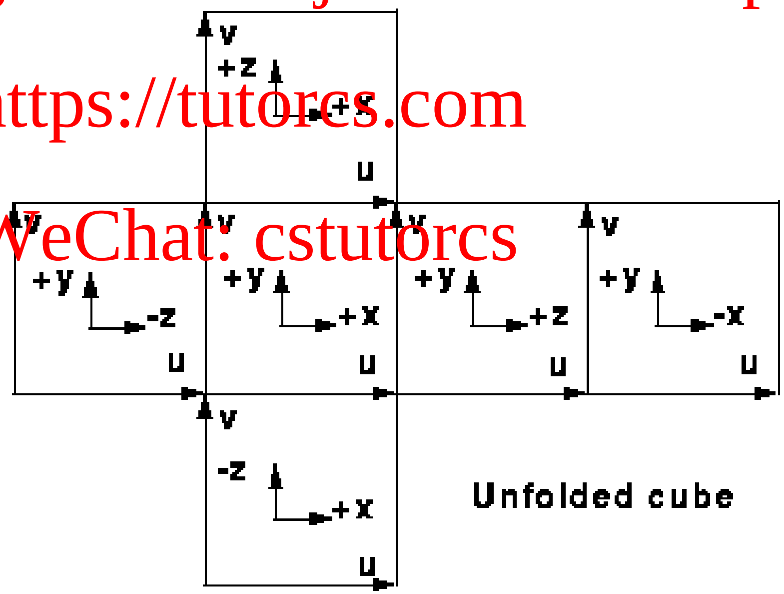
- We can simulate reflections by using the direction of the reflected ray to index a spherical texture map at "infinity".
- Assumes that all reflected rays begin from the same point.

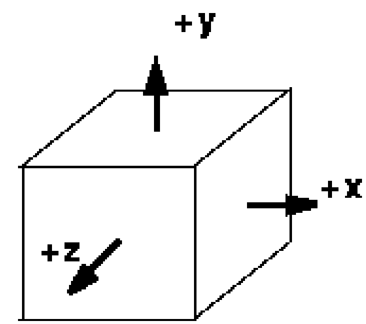


Cube Environment Mapping

➢ use surface normal as an index for each texel on the cube surface







Example

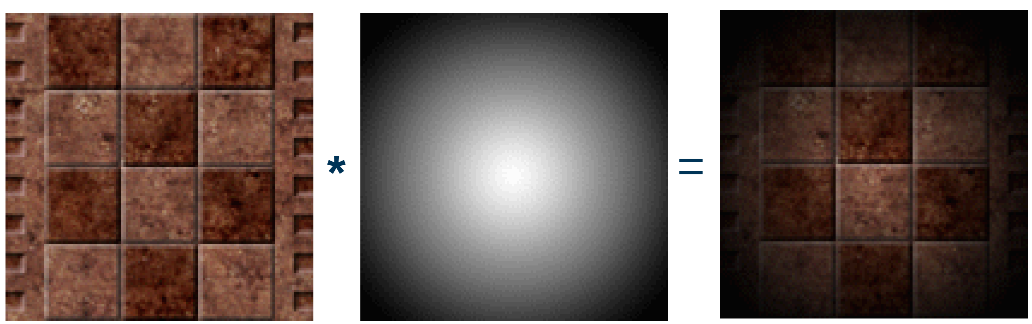
Texture Maps for Illumination

- Also called "Light Maps"
- often different resolution than other textures

Quake

Light Mapping

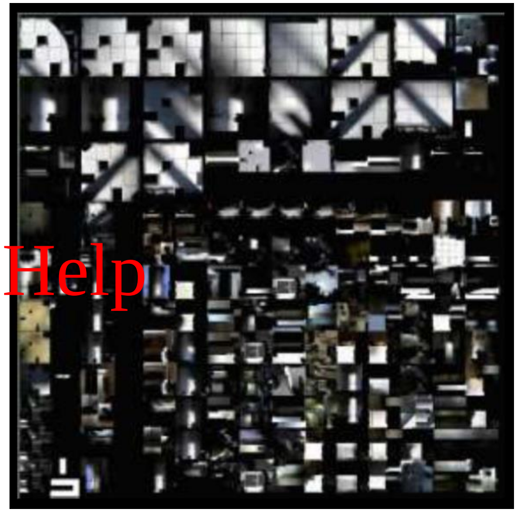
- Realistic lighting can be achieved
- Every single bit of expensive lighting calculation is done during pre-process time. Hence, avoid runtime overhead
- At run-time, all calculations (color arithmetic) are done by hardware. Hence, it is very fast
- Visual quality of the lighting is directly dependent on the size of the light map texture(s)
- For every triangle, a diffuse texture map is applied first and then, a light map is usually modulated (multiplied) with it



[Images courtesy of flipcode.com]

About Lightmap

- **Lightmap Texture** – the lightmaps for different parts of an object are "packed" into a large texture



Fog Maps

- Dynamic modification of light-maps
- Put fog objects into the scene
- Compute where they intersect with geometry and paint the fog density into a dynamic light map
  - Use same mapping as static light map uses
- Apply the *fog map* as with a light map
  - Extra texture stage

Fog Map Example

Summary

- Texture mapping optimization
- Various applications of texture mapping

References:

- **Computer Graphics with Open GL [Chapter 16]**

- **WebGL Programming Guide [Ch. 8]**