centre adscrit a:

UNIVERSITAT DE BARCELONA

Computer Graphics

7. Texturing

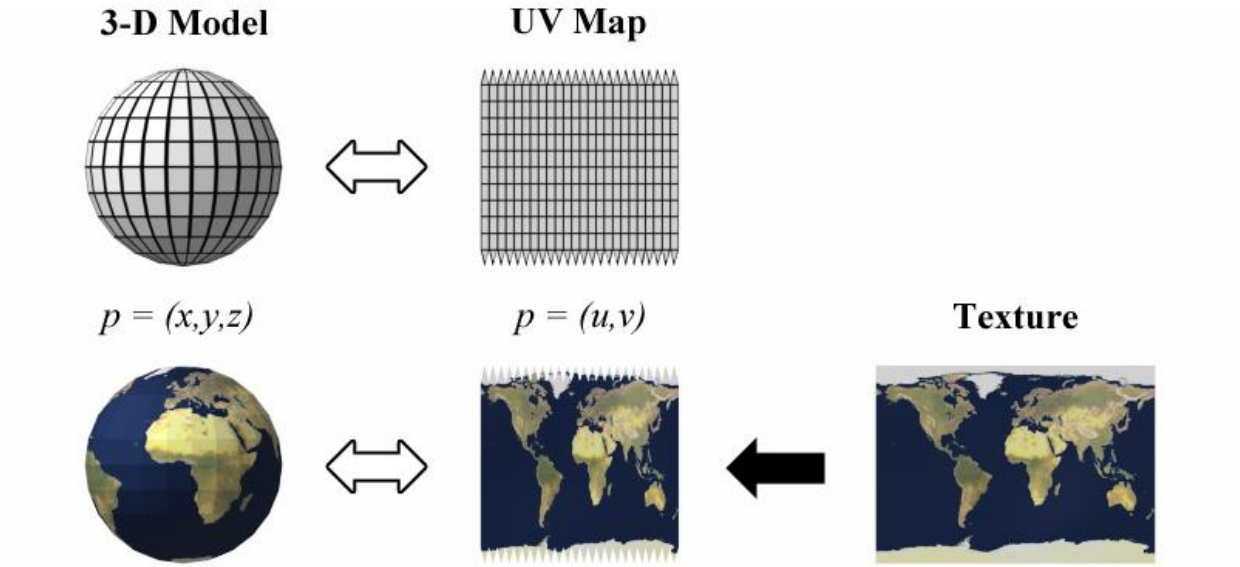
Dr Jesus Ojeda – jesusojeda@enti.cat

Contents

- What is texturing
- Texture Coordinates
- Aliasing
- OpenGL use and exercises
- Advanced techniques

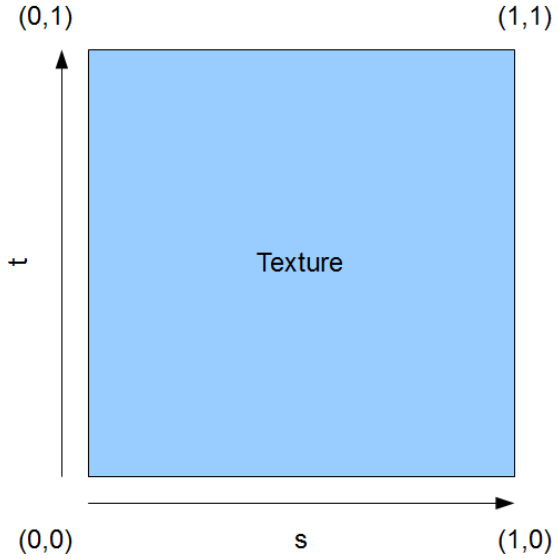
Texturing

- Store properties, commonly a reflectance as an image
- Map that image to the geometry
- Each model primitive will have a corresponding region in the image



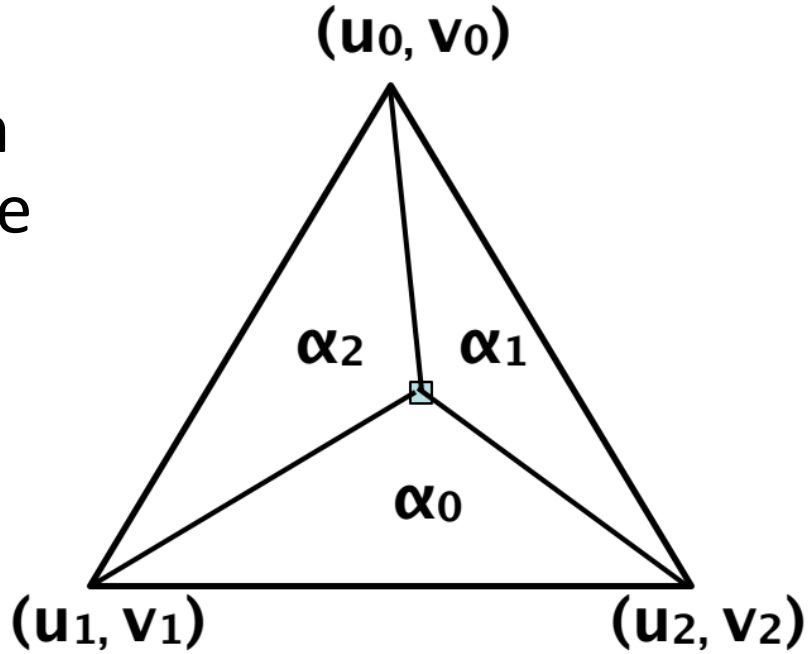
Texture Coordinates

- A texture image is (usually) defined in a 2D coordinate system: (u,v) or (s,t)
- Each triangle vertex is assigned a (u,v) coordinate, so that one knows which part of the texture to stick on the triangle



Texture Coordinates

- Then, for each (rasterized) pixel within the triangle, its texture coordinates are computed from barycentric coordinates.



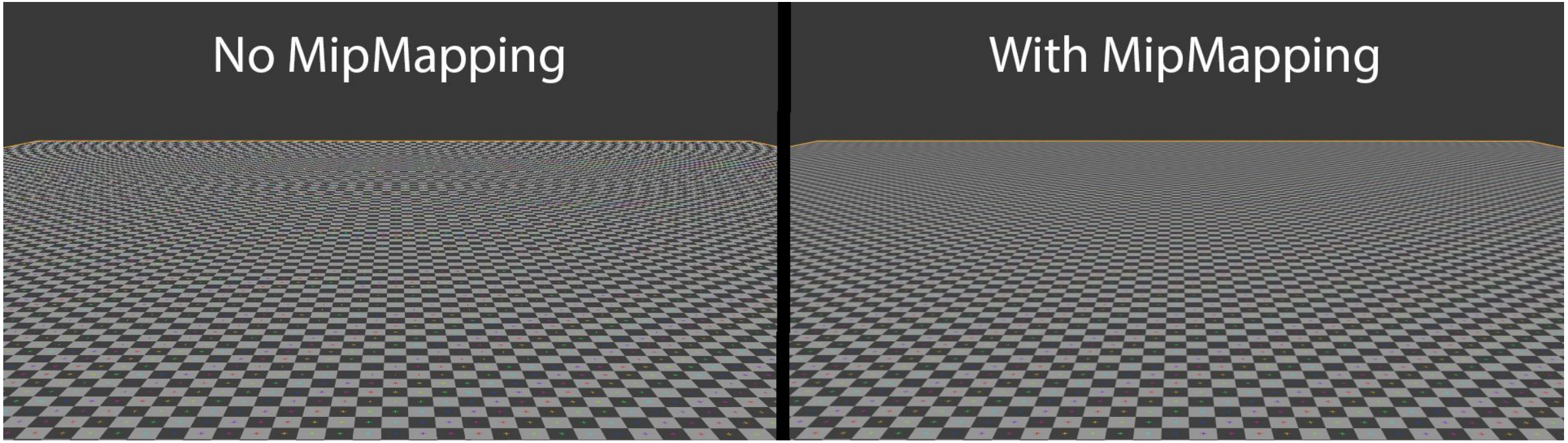
$$(u,v) = \alpha_0(u_0,v_0) + \alpha_1(u_1,v_1) + \alpha_2(u_2,v_2)$$

Texture interpolation

GL_NEAREST

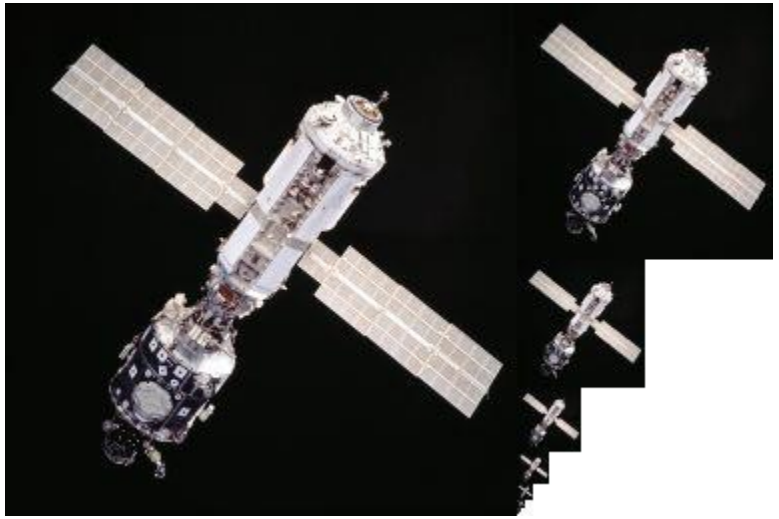
GL_LINEAR

Aliasing



Aliasing

- Using mip mapping, we will generate the same texture at multiple scales.
- Depending on distance to camera, we should access one or other level of the mip map.
- Even interpolation between levels.



- Also used as a Level of Detail technique.

OpenGL – Typical texture setup and use case

- Load image into CPU memory space
- Create texture in GPU & Bind it
- Upload Data
- Use it
- Clean afterwards

Very similar to working with buffers.

After all, its all memory, but with some access parameters.

OpenGL – Hands-On

- Google for **stb_image.h** and add it to the framework

```
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h“

int x,y,n;
unsigned char *data = stbi_load(filename, &x, &y, &n, 0);
// ... process data if not NULL …
// ... x = width, y = height, n = # 8-bit components per pixel ...
// ... replace '0' with '1'..'4' to force that many components per pixel
// ... but 'n' will always be the number that it would have been if you said 0
stbi_image_free(data);
```

OpenGL – Create and Load

- glGenTextures(1, &textureID); // Create the handle of the texture
- glBindTexture(GL_TEXTURE_2D, textureID); //Bind it
- glTexImage2D(/*…*/); //Load the data
- glTexParameteri(/*…*/); //Configure some parameters
  - GL_TEXTURE_MIN_FILTER
  - GL_TEXTURE_MAG_FILTER
  - GL_TEXTURE_WRAP_S
  - GL_TEXTURE_WRAP_T

//Then, to clean at the end…

- glDeleteTextures(1, &textureID);

OpenGL – Use the texture

- In C++:

```
glActiveTexture(GL_TEXTUREi); //
glBindTexture(GL_TEXTURE_2D, textureId);
glUniform1i("diffuseTexture”, i);
```

- In GLSL (fragment shader for example):

```
in vec2 fragmentUV;
uniform sampler2D diffuseTexture;
void main() {
//…
    vec4 textureColor = texture(diffuseTexture,fragmentUV);
//…
}
```

Let’s try it live!

From a loaded model with texture coordinates (the cube OBJ for examples)

1. Load a texture image.
2. Draw the model with the texture.
3. Animate the texture coordinates through time. What is the result?

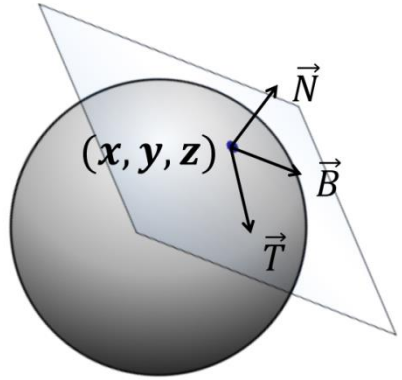
Advanced techniques

- Bump Mapping
- Normal Mapping
- Displacement Mapping
- Environment Mapping
- Procedural Textures
- 3D Textures

A local coordinate system

- We can define a local coordinate system using the mapping between each vertex (x, y, z) and the texture coordinates u, v to calculate a Tangent, Binormal, and Normal
- Assume that x, y, and z are each linear functions of u and v in each triangle:

$$x = a_0u + b_0v + c_0$$
$$y = a_1u + b_1v + c_1$$
$$z = a_2u + b_2v + c_2$$



- For each vertex in the triangle plug them into the equations, with total of 9 equations
- After solving them:

$$\vec{T} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right) = (a_0, a_1, a_2) \quad \vec{B} = \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right) = (b_0, b_1, b_2) \quad \vec{N} = \vec{T} \times \vec{B}$$

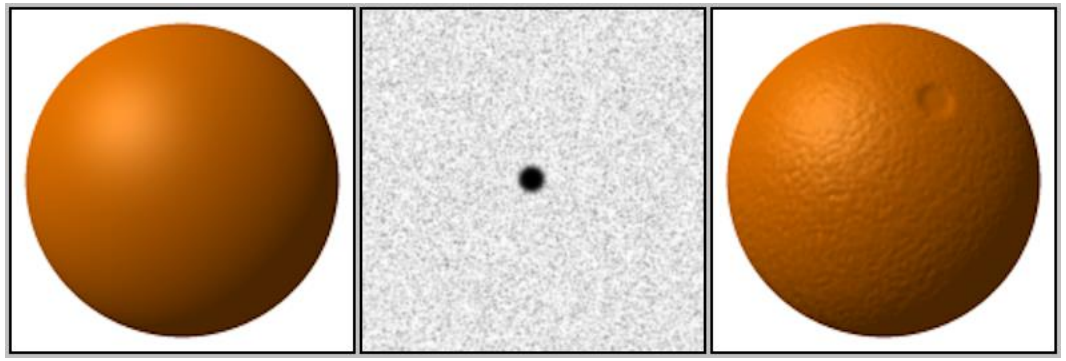
Perturbing Normals – Bump Maps

- Instead of using the normal from the geometry, use (or modify the previous with) one normal stored or computed from a texture.
- A bump map is a single-channel (grey-scale) height map h(u, v)
- Points on the three dimensional surface are given by (u, v, h(u, v))

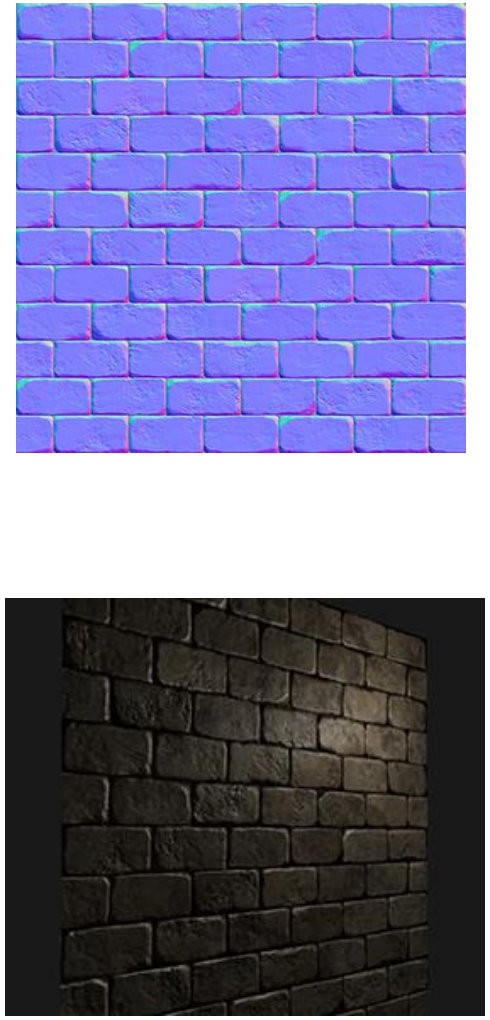
$$\frac{\partial h(u,v)}{\partial u} = \frac{h(u_{right}, v_{center}) - h(u_{left}, v_{center})}{u_{right} - u_{left}} \quad \frac{\partial h(u,v)}{\partial v} = \frac{h(u_{center}, v_{top}) - h(u_{center}, v_{bottom})}{v_{top} - v_{bottom}}$$

- The new normal will be then the normalized version of

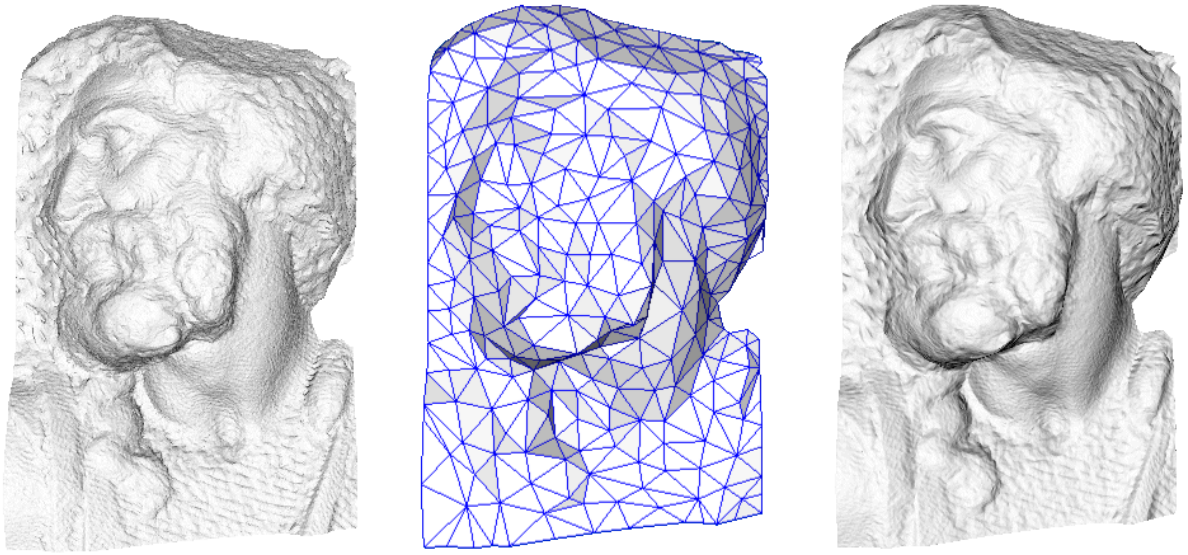
$$\left(-\frac{\partial h(u,v)}{\partial u}, -\frac{\partial h(u,v)}{\partial v}, 1\right)$$



Normal Mapping



From defining normals per pixel, it can also be used to reduce polycount without losing apparent detail.



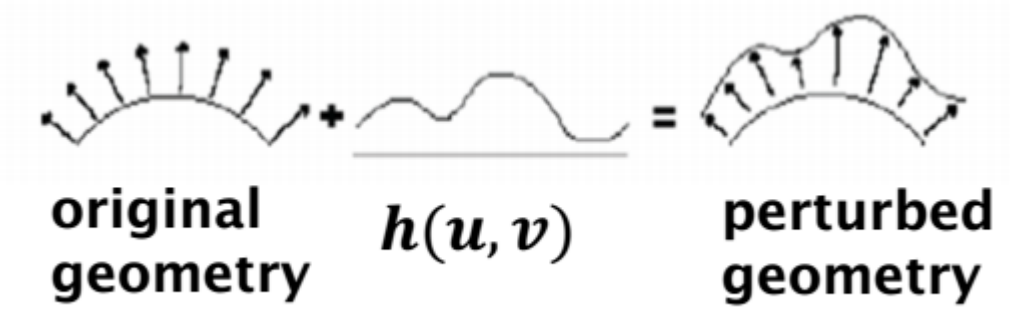
original mesh
4M triangles

simplified mesh
500 triangles

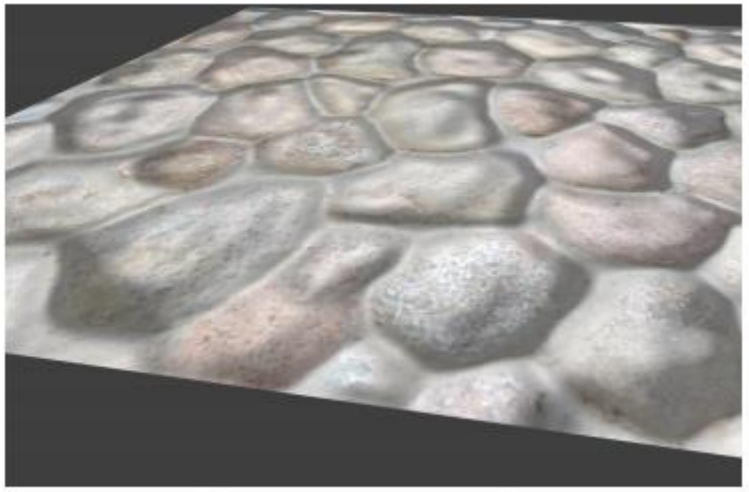
simplified mesh
and normal mapping
500 triangles

Displacement Mapping

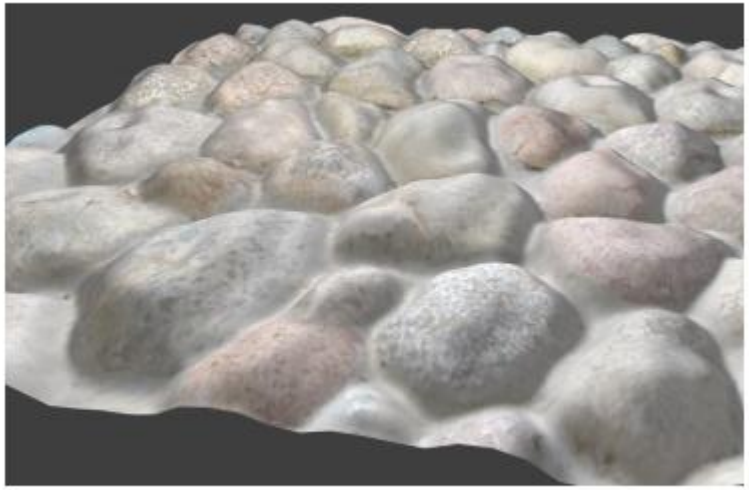
- What if we perturb the surface geometry with a heightfield stored in texture?



- Pros: self-occlusion, self-shadowing
- Cons: expensive (if generating new geometry)



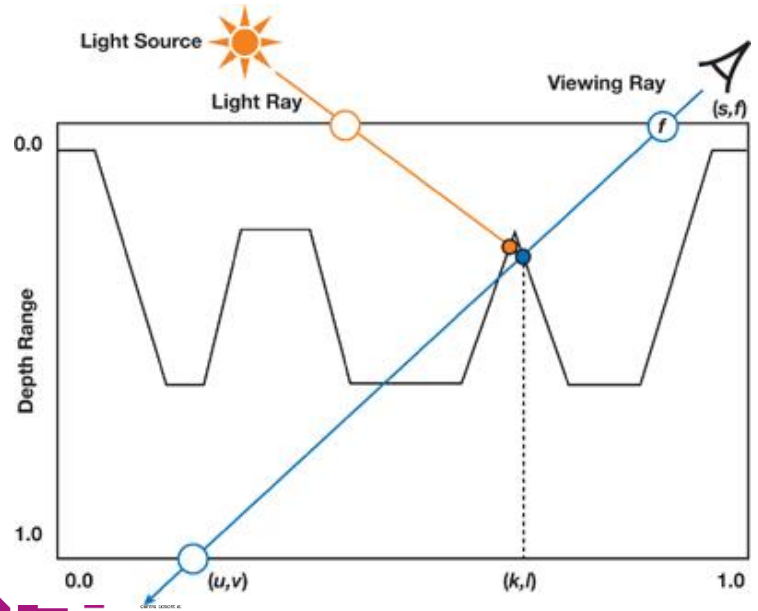
Bump mapping

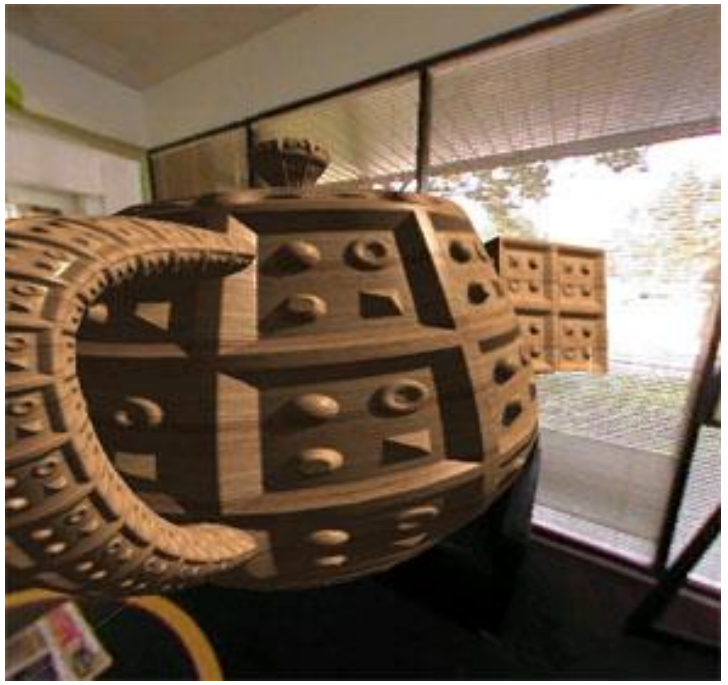


Displacement mapping

Relief Mapping

- Relief mapping simulates the appearance of geometric surface details by shading individual fragments in accordance to some depth and surface normal information that is mapped onto polygonal models.
- Uses a raycast implementation in the fragment shader to travel through a ray until a textured heigthfield is intersected.

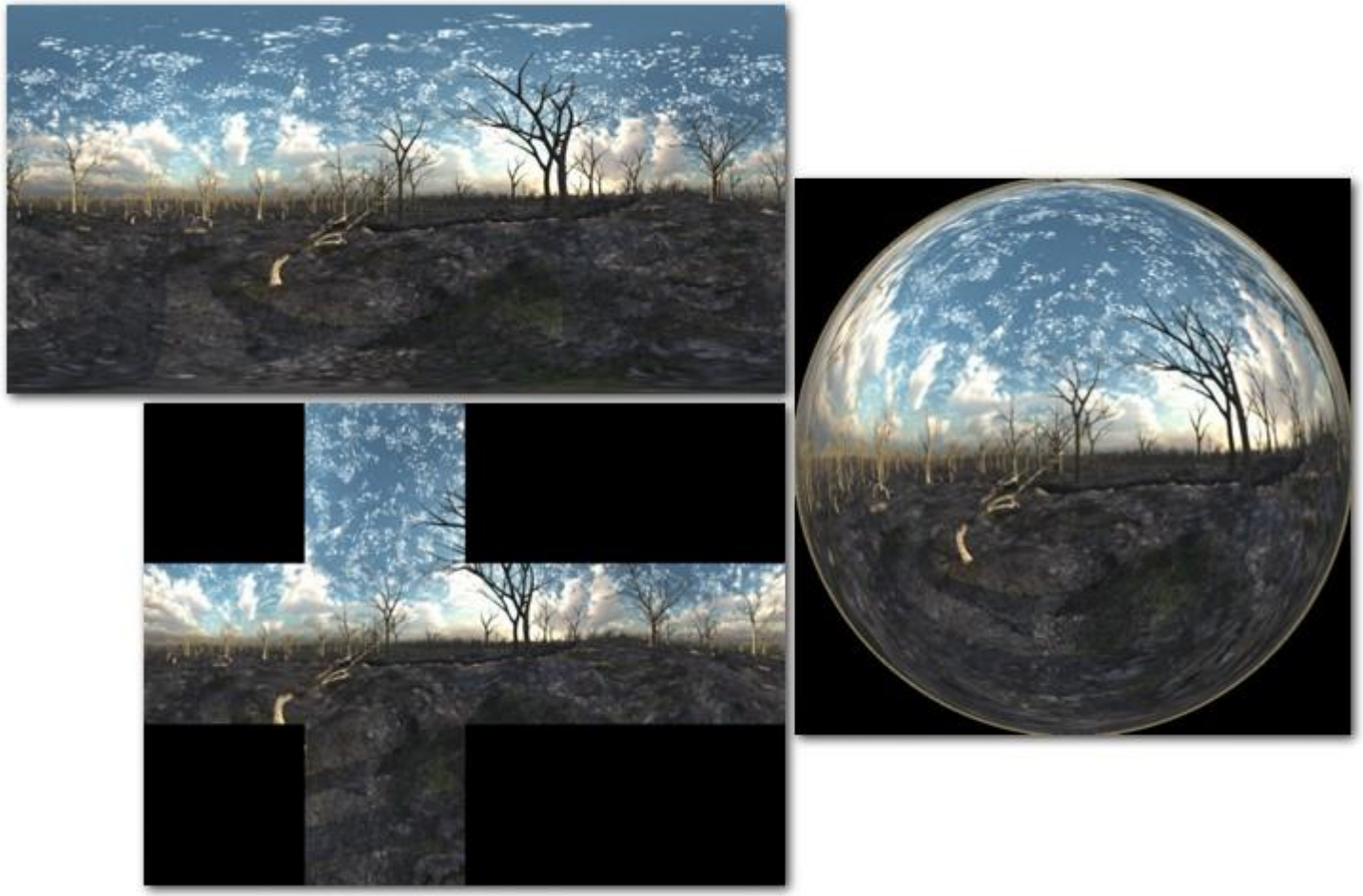


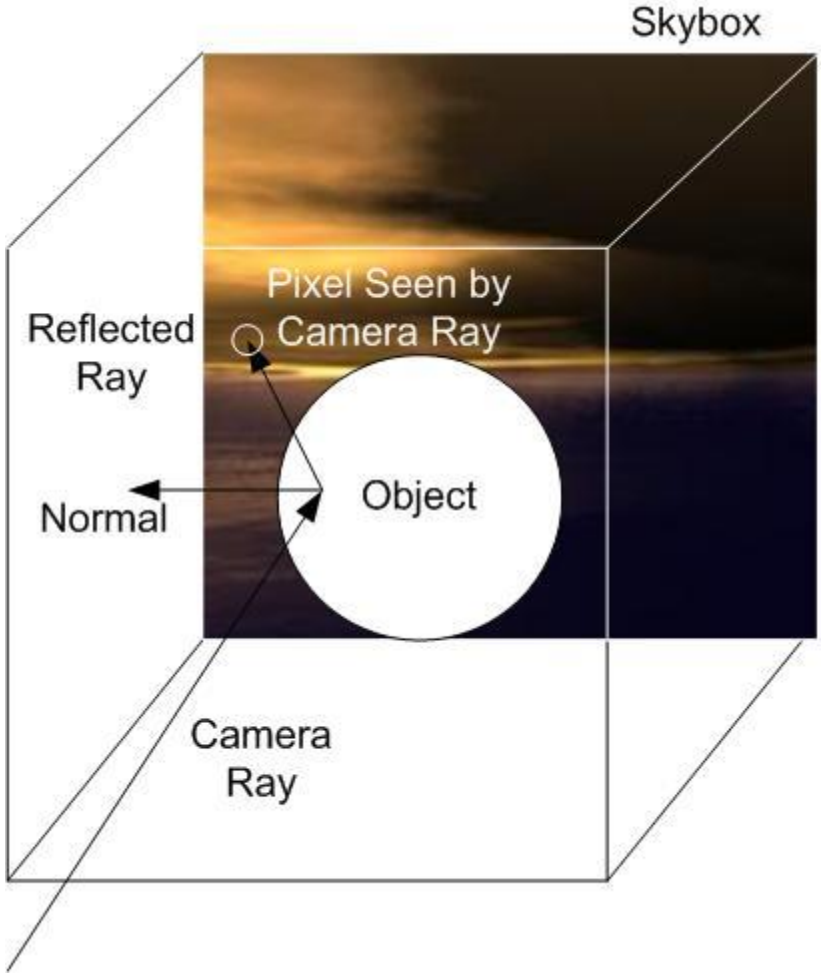


Environment Mapping

- Objects reflect light transmitted to them from other objects
  - e.g. the refection of a tree on a car
- An environment texture map stores the reflected image of the environment as a texture image
- Used for image-based lighting

Environment Mapping



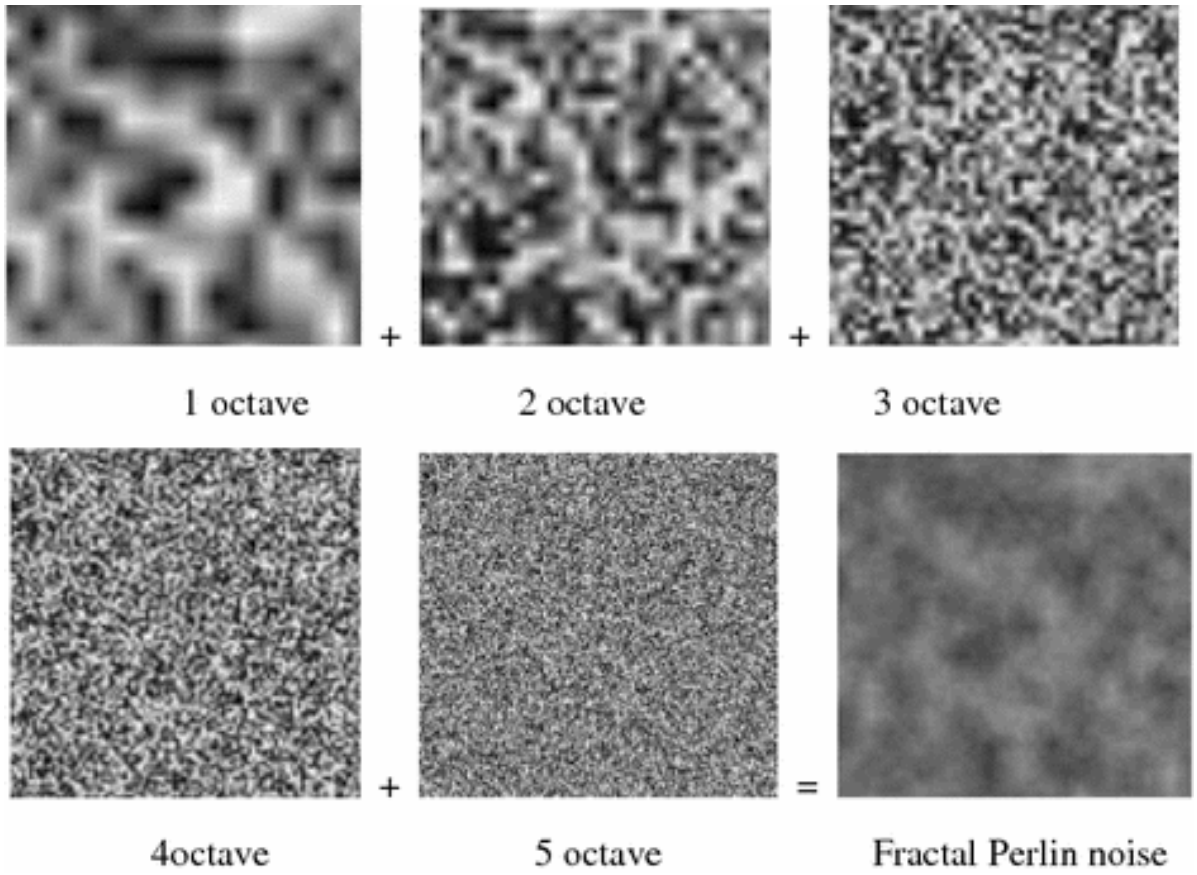


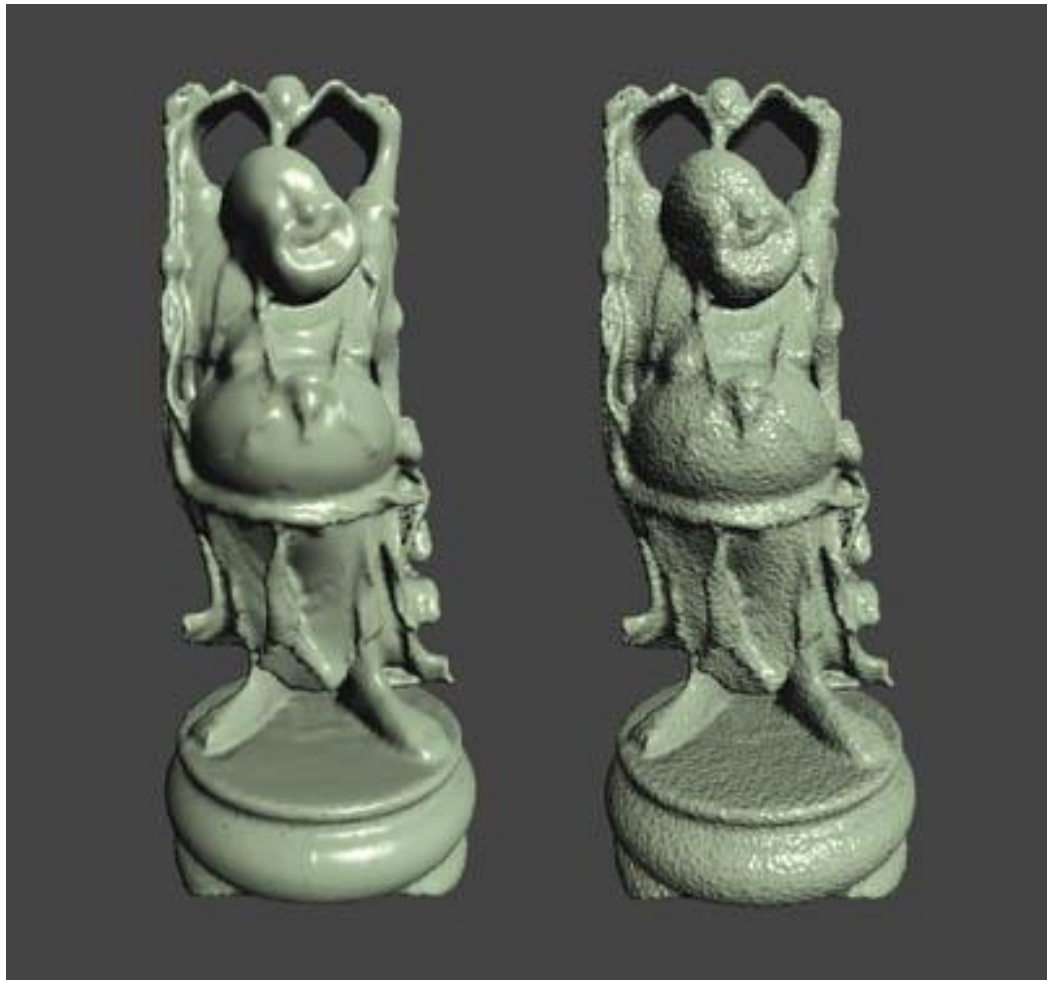
Procedural textures

- Created/generated using a mathematical/computational algorithm
- Good for generating natural elements
  - e.g. wood, marble, granite, stone, etc.
- The natural look is achieved using noise or turbulence functions
- These turbulence functions are used as a numerical representation of the “randomness” found in nature

- Search for noise functions (Perlin Noise, for example)

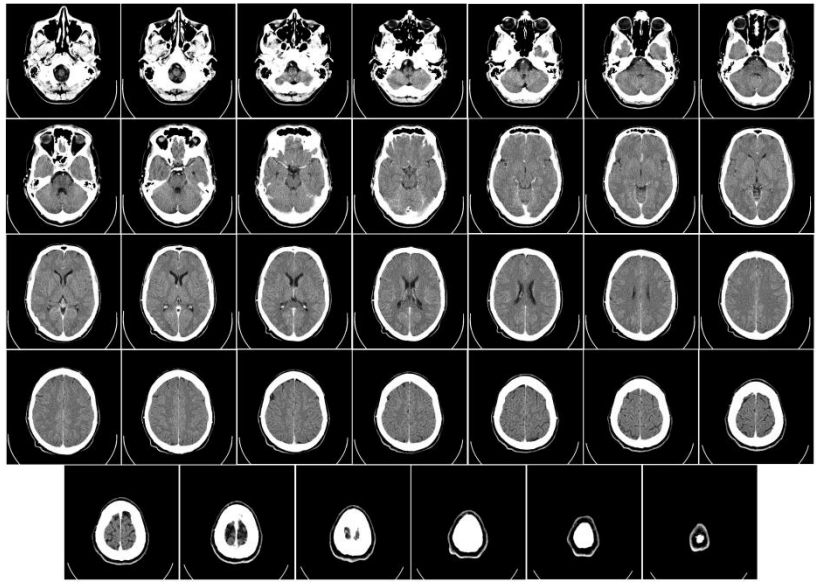
Procedural textures

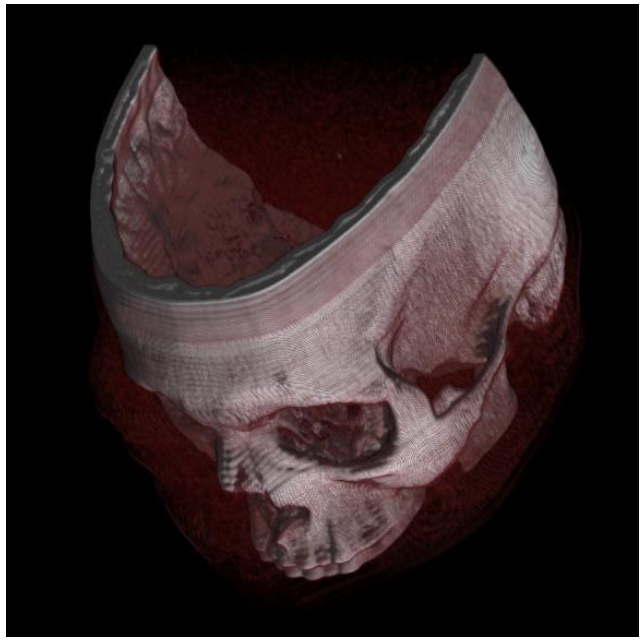




3D textures

- Procedurally generated (clouds in games, for example) or from some 3D imaging technology (MRI, CT, etc in medicine)
- Search for Volume Rendering -> typically some raycast solution





Resources

- David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley. **TEXTURING & MODELING A Procedural Approach**, 3rd edition. Morgan Kaufmann.
- Graham Sellers, Richard S. Writght, Jr. Nicholas Haemel. **OpenGL SuperBible**, 6th Edition. Pearson education.
- John Kessenich, Graham Sellers, Dave Shreiner. **OpenGL Programming guide**. Ninth Edition. Pearson Education.