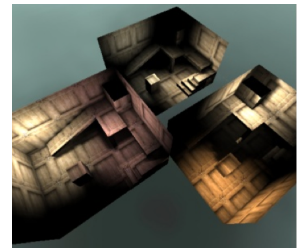
Applications of Texture Mapping

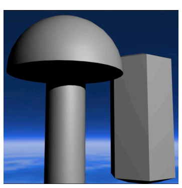
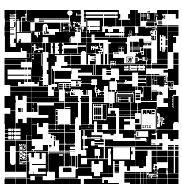
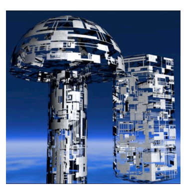
Prof. Vladlen Koltun
Computer Science Department
Stanford University

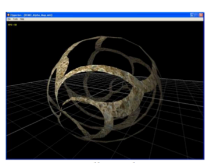
Light maps

http://www.irrlicht3d.org

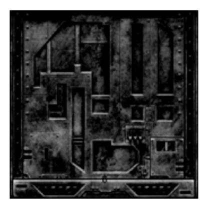
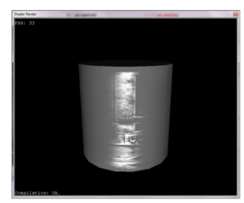
Opacity mapping

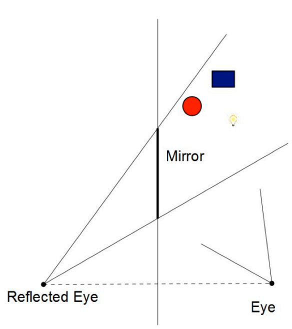


http://lectrablog.lectralizard.com/page/2

Specular mapping

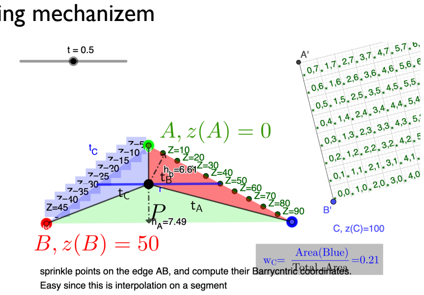
Mirrors



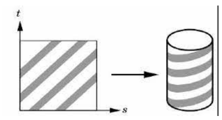
Lets understand how texture mapping works

https://www.geogebra.org/m/jamjwsuk

- Mapping a single triangle - use the b-coordinates and restoring mechanizem

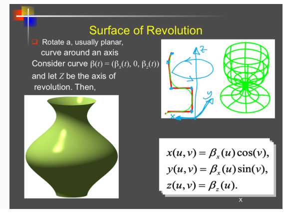


What About cylinders?



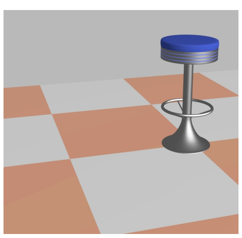
- https://www.geogebra.org/m/vckhvb6s
- Map $(i, j) \leftrightarrow (cos(i), sin(i), j)$

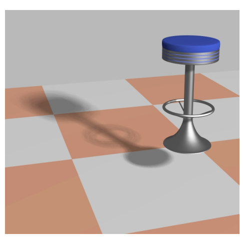
Surface of revolution

-  

- (later in the syllabus (modeling)
- Idea - calculate the length of the curve

Shadows

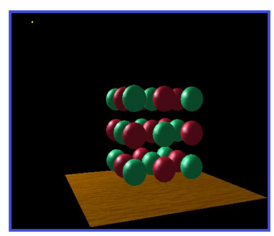


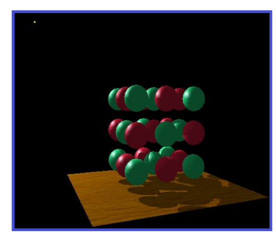


- Valuable cue of spatial relationships
- Increases realism

Akenine-Moeller and Haines

Shadows

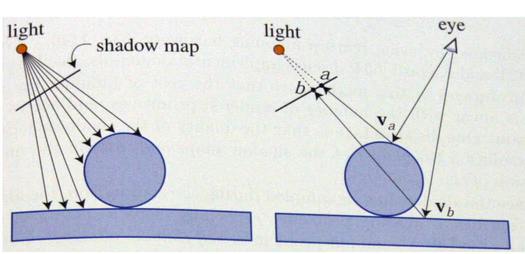




- Valuable cue of spatial relationships
- Increases realism

Mark J. Kilgard

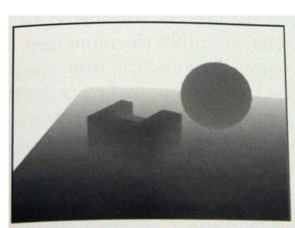
Shadow mapping

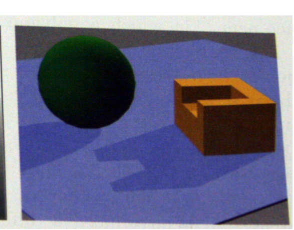


- First pass: render the scene from the viewpoint of the light, store depth buffer as texture (shadow map)
- Second pass: project vertices into shadow map and compare depth values

Akenine-Moeller et al., Real-Time Rendering

Shadow mapping

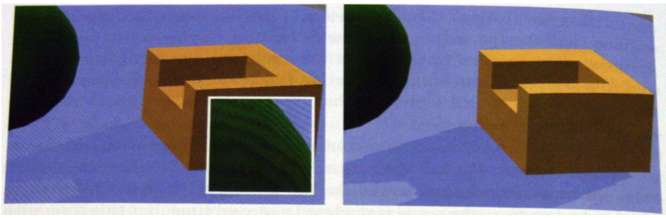




- First pass details: can disable all rendering features that do not affect depth map.
- Second pass details: For each fragment, use the light's modelview and projection transforms to obtains (u,v) coordinates in the shadow map and the depth w of the vertex.
- Compare w with value w' stored in (u,v) in the shadow map. If w ≤ w', perform lighting calculations with this light. Otherwise, do not.

Akenine-Moeller et al., Real-Time Rendering

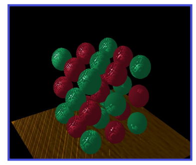
Bias



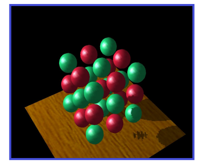
- Numerical imprecision leads to self-shadowing
- Solution: add a bias ε. Change comparison from $w \le w'$ to $w \le w' + \varepsilon$
- Can use glPolygonOffset

Akenine-Moeller et al., Real-Time Rendering

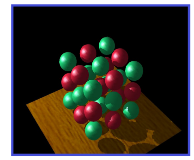
Setting the bias



Too little



Too much

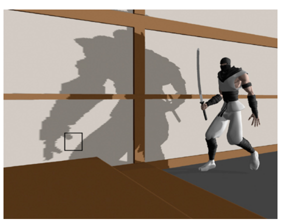


Just right

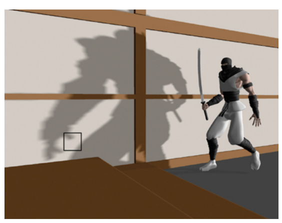
- Numerical imprecision leads to self-shadowing
- Solution: add a bias ε. Change comparison from $w \le w'$ to $w \le w' + \varepsilon$
- Can use glPolygonOffset

Mark J. Kilgard

Shadow map aliasing



Unfiltered



Filtered

- Insufficient shadow map resolution leads to blocky shadows
- No easy solution. Should not filter depth values: leads to errors at object boundaries
- Percentage-closer filtering: filter comparison results

Bunnell and Pellacini

Other issues

- Additional rendering pass for each shadow-casting light
- Setting the "field of view" of the light. Can use spotlights, or a cube map (six shadow maps) for a point light.
- For directional lights, use orthographic projection

Reflection mapping

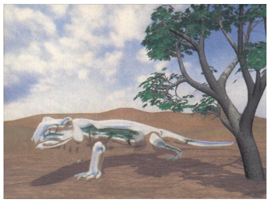
Terminator 2

Reflection mapping

- Render the scene from a single point inside the reflective object. Store rendered images as textures.
- Map textures onto object. Determine texture coordinates by reflecting view ray about the normal.

Terminator 2

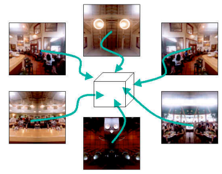
Cube mapping

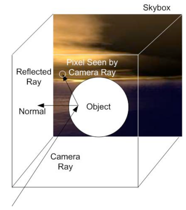


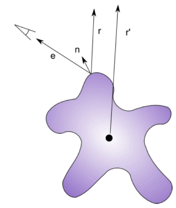
- Render the scene six times, through six faces of a cube, with 90-degree field-of-view for each image.
- Store images in six textures, which represent an omni-directional view of the environment

Greene, 1986

Cube mapping







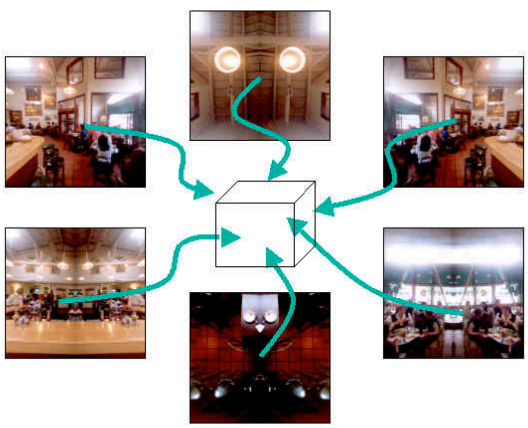
- To compute texture coordinates, reflect the view vector **v** about the normal **n**:

$$\mathbf{r} = 2(\mathbf{v} \cdot \mathbf{n})\mathbf{n} - \mathbf{v}$$

- The highest (in absolute value) coordinate of r identifies which of the six maps we need. The texture coordinates in this map are obtained by normalizing the other two coordinates of r.

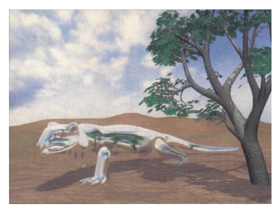
http://developer.nvidia.com/object/cube_map_ogl_tutorial.html; TopherTG (Wikipedia)

Cube mapping



http://developer.nvidia.com/object/cube_map_ogl_tutorial.html

Sphere mapping



- Cube maps require maintaining six texture in memory
- Sphere mapping uses a single viewpoint-specific environment map, updated every frame
- Map depicts a perfectly reflective sphere viewed orthographically

Greene, 1986

Rough Surfaces

- Relief mapping, Bump Mapping, Parallax Mapping
- Parallax Mapping - far way objects looks smaller, and appear to move slower, comparing to same-size nearby objects

Relief mapping

normal mapping

relief mapping

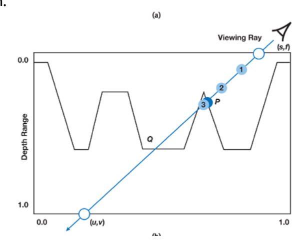
Image from Natalya Tatarchuk

Relief mapping

Image from Policarpo et al. (2005)

Relief mapping

Could be injected into ray-tracing algorithm, or could be use to inject ray tracing into z-buffer algorithm.

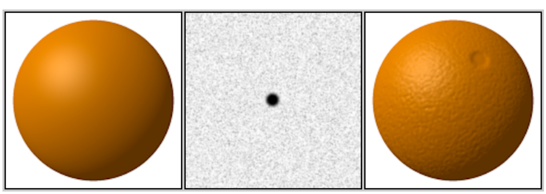


Note - in a scene with multiple triangles, we do not apter the decision which triangle is hit

- Trace the eye ray into the bump map. A simple implementation can rasterize the projection of the ray onto the tangent plane, stepping along (e_t, e_b) and adjusting the height by a factor proportional to e_h.

Image from Policarpo and Oliveira (2008)

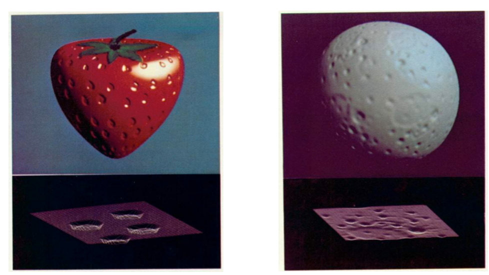
Bump mapping



- Simulates roughness ("bumpiness") of a surface without adding geometry
- Uses a two-dimensional height field (bump map) to perturb the normal during per-fragment shading calculations
- Limitation: silhouette is unaffected
- The surface is still smooth - just the normals are modified, so diffused and specular shading are effected.

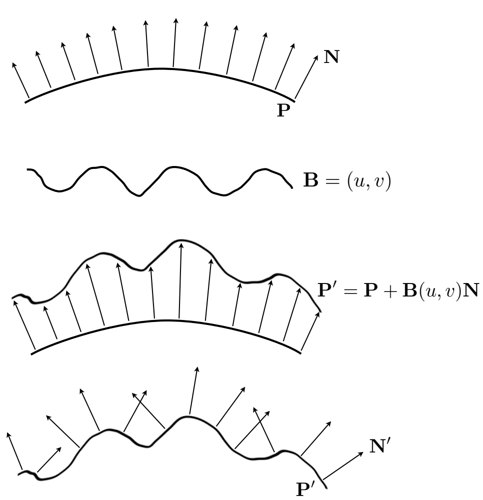
GDallimore (Wikimedia Commons)

Bump mapping

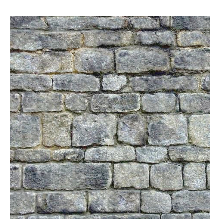


- Simulates roughness ("bumpiness") of a surface without adding geometry
- Uses a two-dimensional height field (bump map) to perturb the **normal** during per-fragment shading calculations (but does not effect the geometry)
- Limitation: silhouette is unaffected

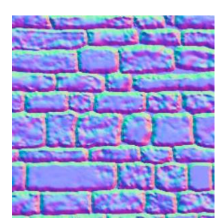
Blinn, SIGGRAPH 1978



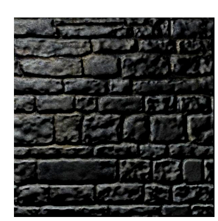
Normal mapping



original wall



normal map



shaded wall

- Store the displaced normals directly. Reduces runtime overhead, at the expense of memory requirements
- (x,y,z) values in the tangent space are stored in the RGB channels. To compute the normal at a fragment, we simply multiply the (interpolated) tangent space basis by (x,y,z)

http://www.blacksmith-studios.dk/projects/downloads/bumpmapping_using_cg.php

Parallax mapping

Image from Terry Welsh

Can be combined into
Parallax occlusion mapping

Relief mapping

normal mapping

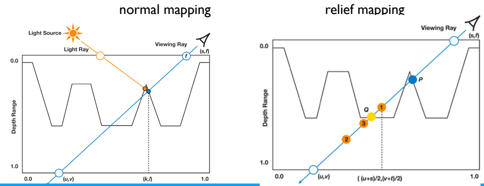
relief mapping

Similar results, different math. Impacts both shading and outcome image.

Relief mapping

normal mapping

relief mapping



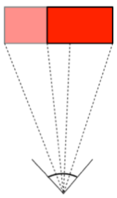
1. Similar results to parallel occlusion map.
2. Different math/algorithm
3. Impacts both shading and outcome image.
4. Credit https://developer.nvidia.com/gpugems

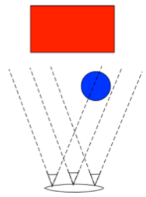
Other Buffers

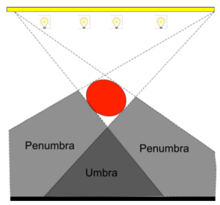
Accumulation buffer

- High-precision image buffer. Can integrate images that are rendered into the framebuffer. Supports anti-aliasing, motion blur, depth of field, soft shadows, etc.
- 16 bits for each red, green, blue, and alpha component: total of 64 bits per pixel.
- Supports the following operations:
  - Clear: set all values to zero.
  - Add with weight: Each pixel in the drawing buffer is added to the accumulation buffer after being multiplied by a floating-point weight that can be positive or negative.
  - Return with scale: The contents of the accumulation buffer are returned to the drawing buffer after being scaled by a positive floating-point constant
- Can integrate up to 256 images without loss of prevision, and even more using weight less than 1.0

Accumulation buffer

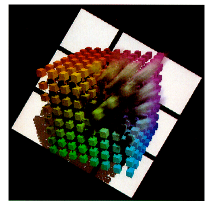

motion blur

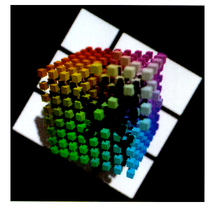

depth of field

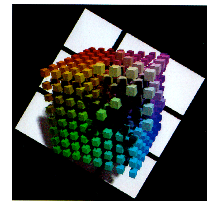


soft shadows

Accumulation buffer


motion blur

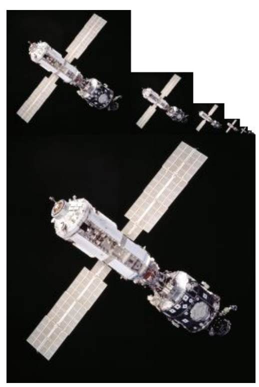

depth of field


soft shadows

Haeberli and Akeley, SIGGRAPH 1990

Mipmap

- How accurate should be the images used as a texture?
- Wasteful if too detailed (depending on viewer position)
- If multiple copies of image are placed next to each other, sensitive to aliasing (next slide)
- idea **"level of details"**
- Antialiasing is only one of the applications of mipmaps
- To quickly compute averages, store the texture at multiple resolutions
- For each lookup, estimate the size of the footprint and index into the mipmap accordingly



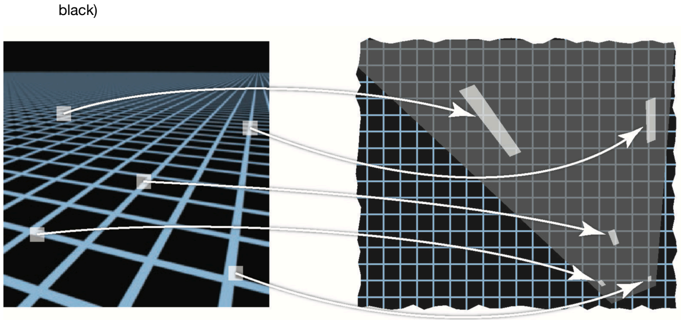
https://en.wikipedia.org/wiki/Mipmap

Problem: Sampling Textures Can Lead to Aliasing

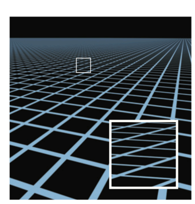
- Just as we've seen with image processing and raytracing applications, if details are not captured with sufficient samples we can see noticeable artifacts
- Solution: use a better sampling/reconstruction

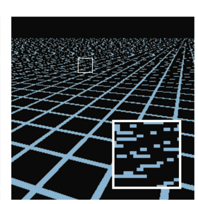
Pixel Footprints

- Can vary in size, shape, and orientation relative to the texture
- Problem: Which of the texture pixels show we pick for each image pixel ? (blue or black)



Answer: neither blue nor black is correct. We need to average them.


High-resolution image


Point sampling

To resolve the aliasing problem: For each rendered image pixel, we need to average multiple texture pixels.
Their number might be large.

Sampling and Reconstruction

- If footprint is small, need better reconstruction (e.g. bilinear instead of nearest neighbor)
- If the footprint is large, need to average many samples