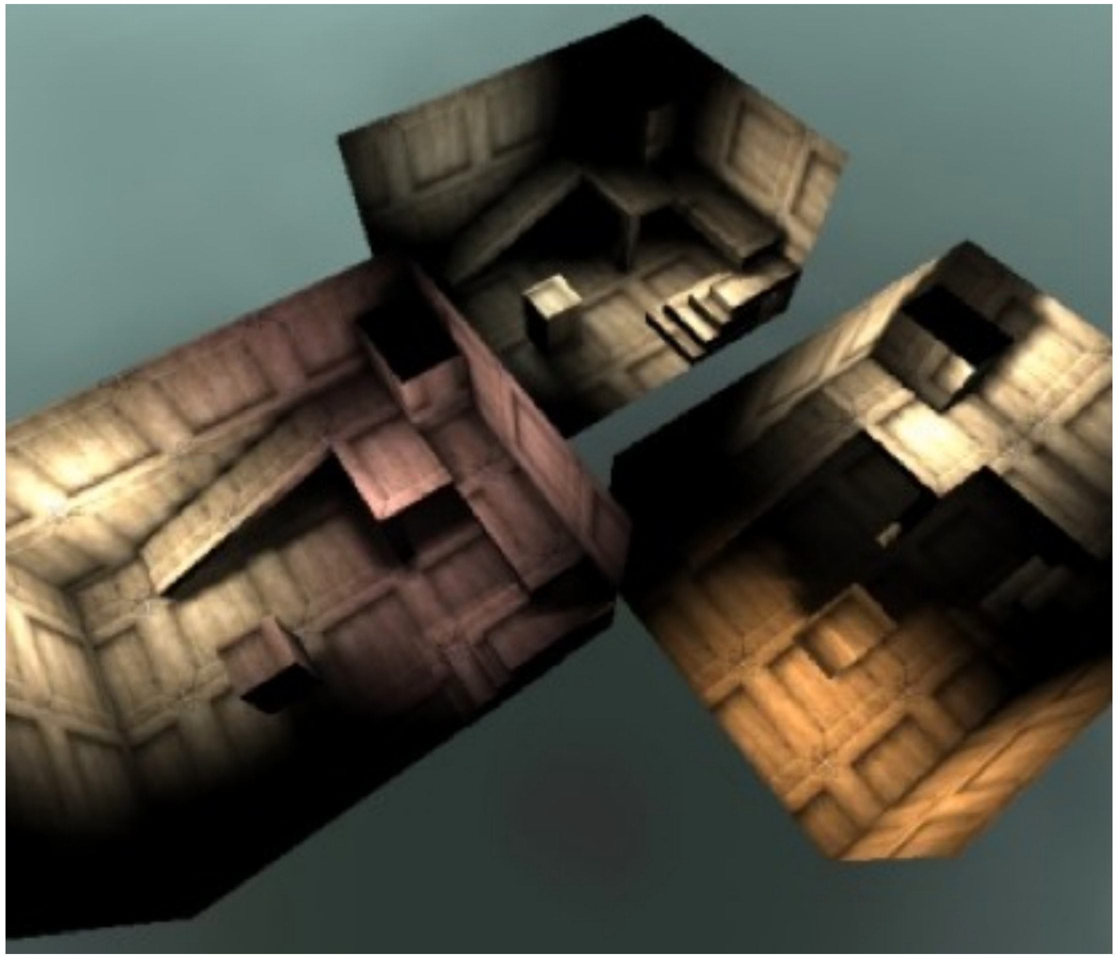
Applications of Texture Mapping

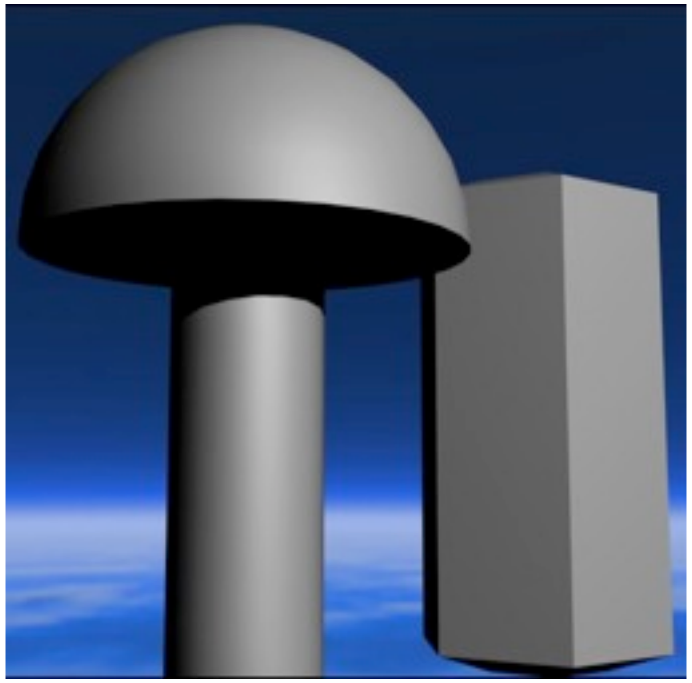
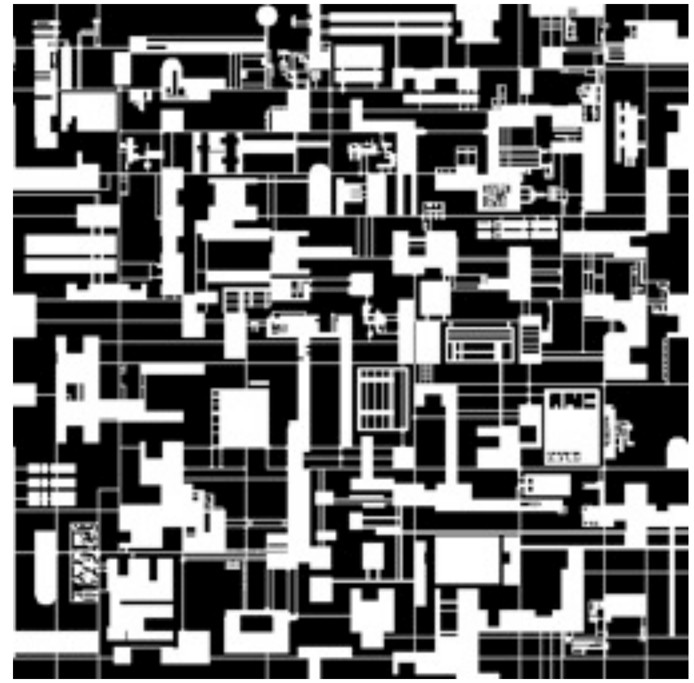
Prof. Vladlen Koltun
Computer Science Department
Stanford University

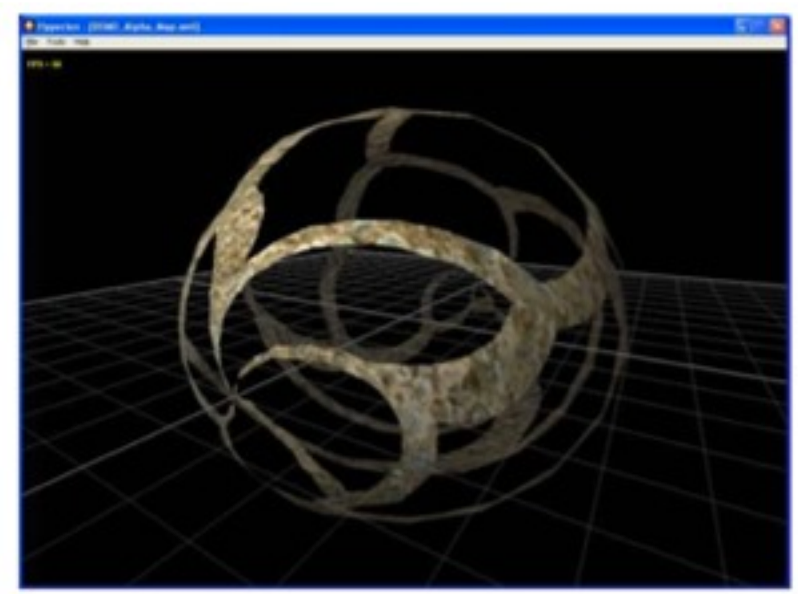
Light maps



http://www.irrlicht3d.org

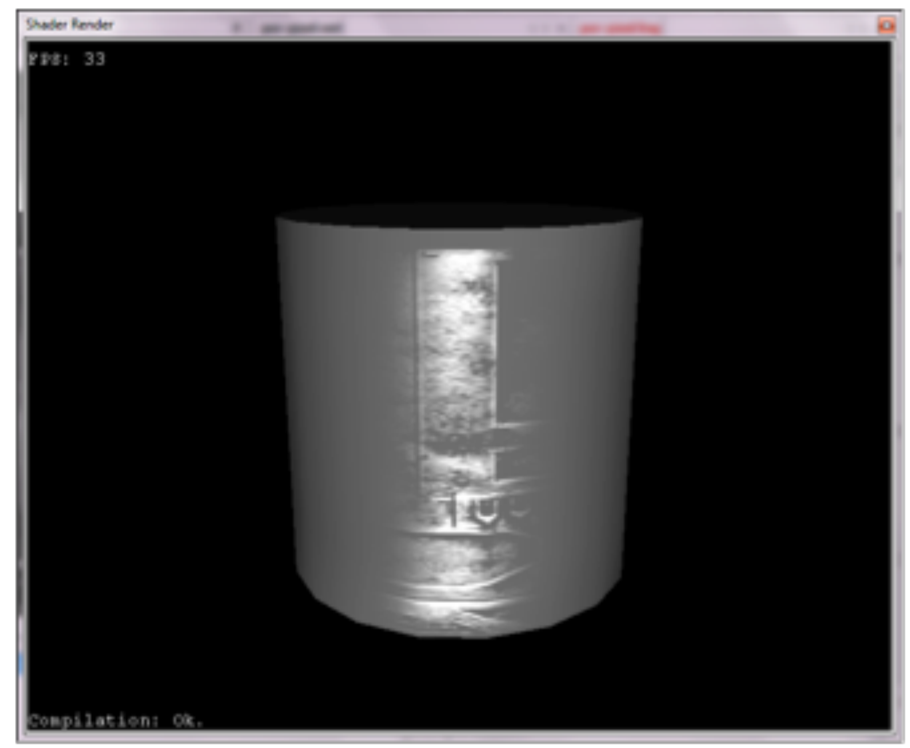
Opacity mapping

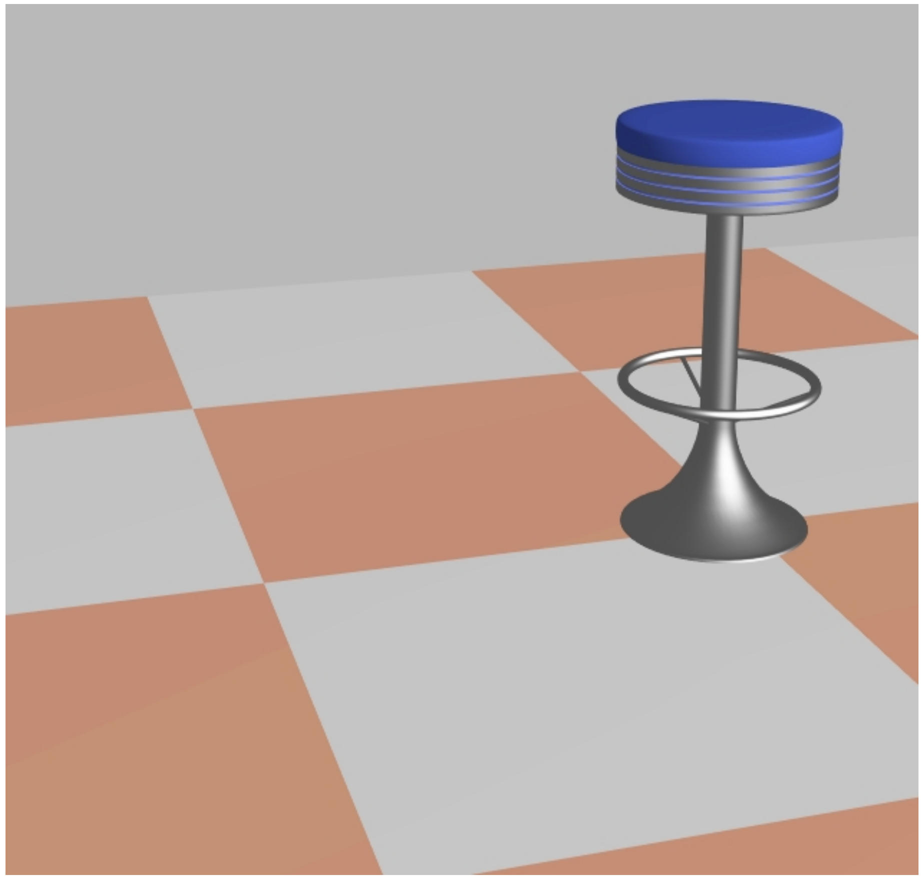



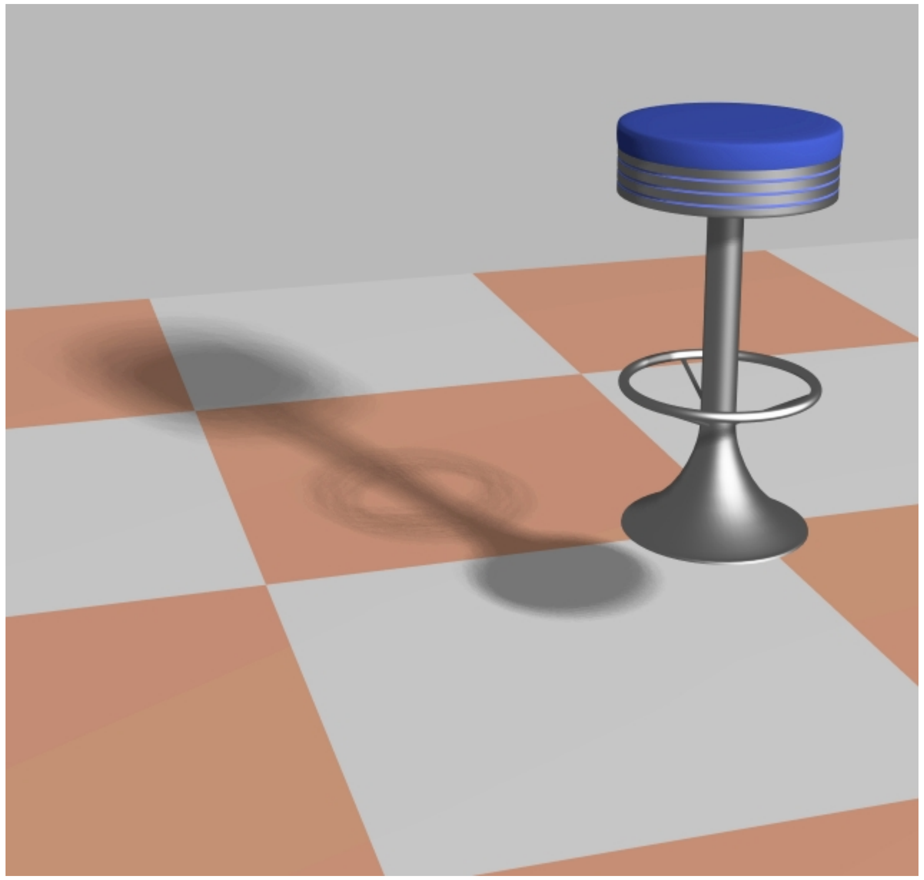
http://lectrablog.lectralizard.com/page/2

Specular mapping



Shadows

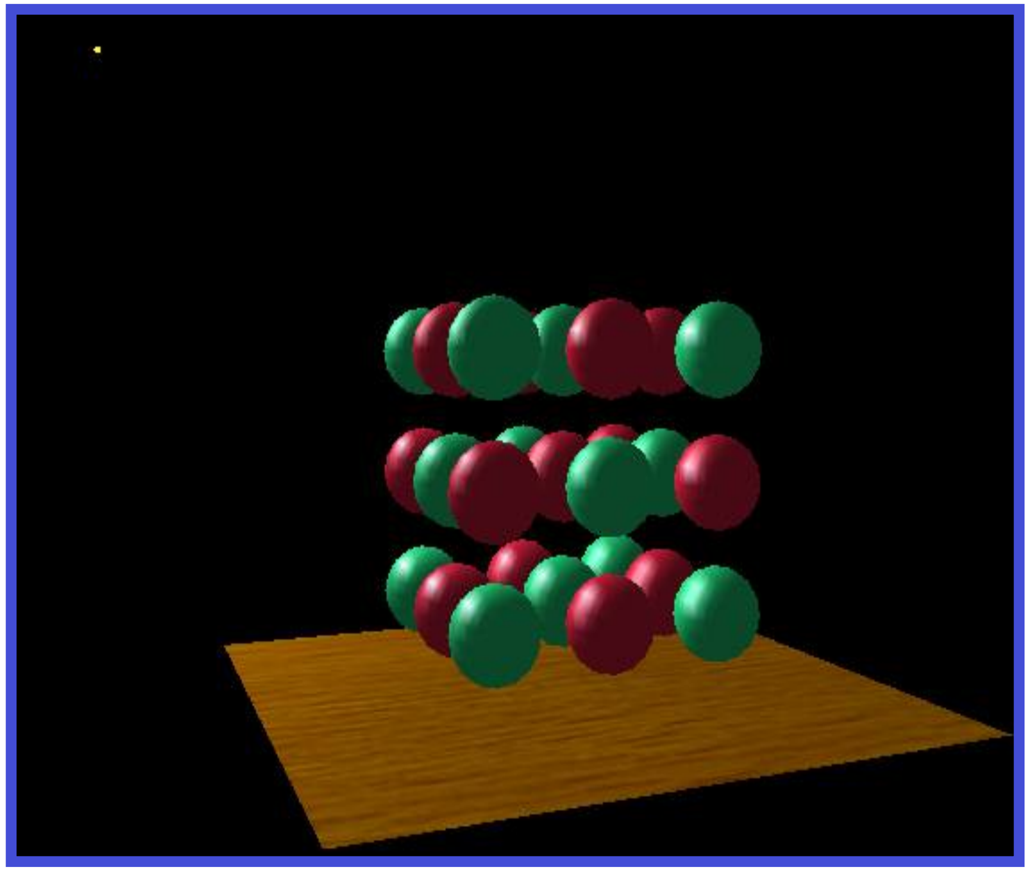


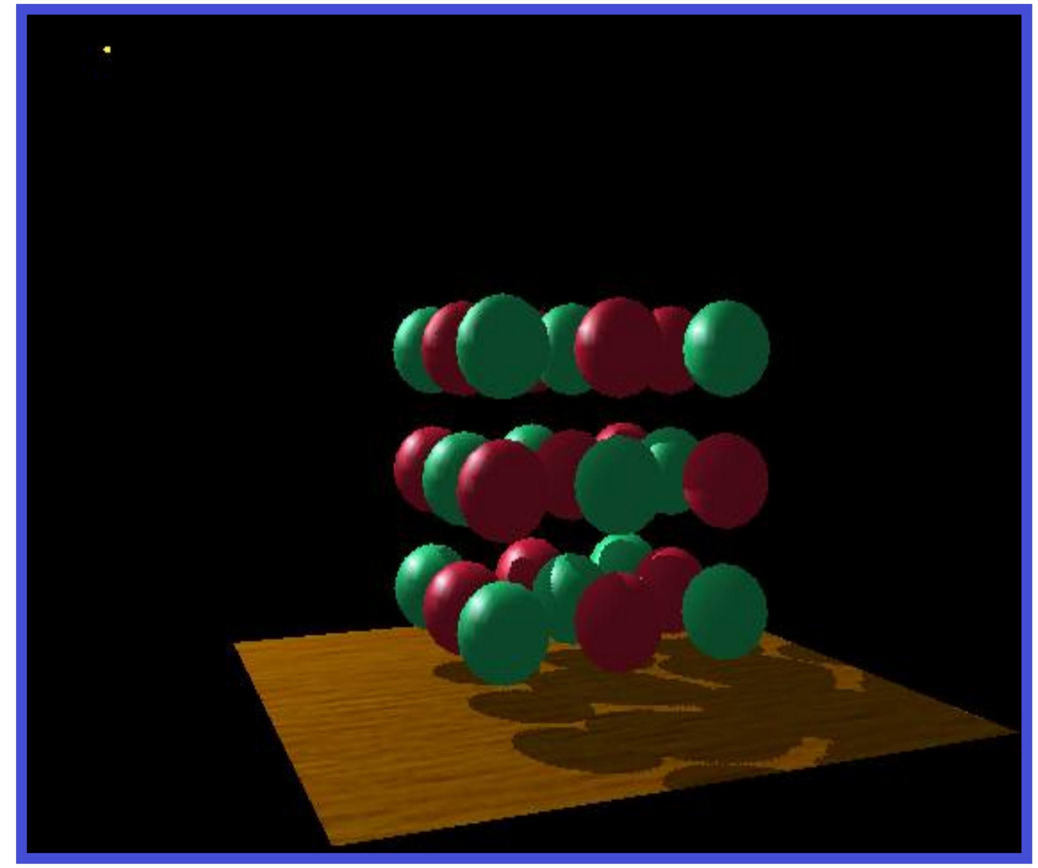


- Valuable cue of spatial relationships
- Increases realism

Akenine-Moeller and Haines

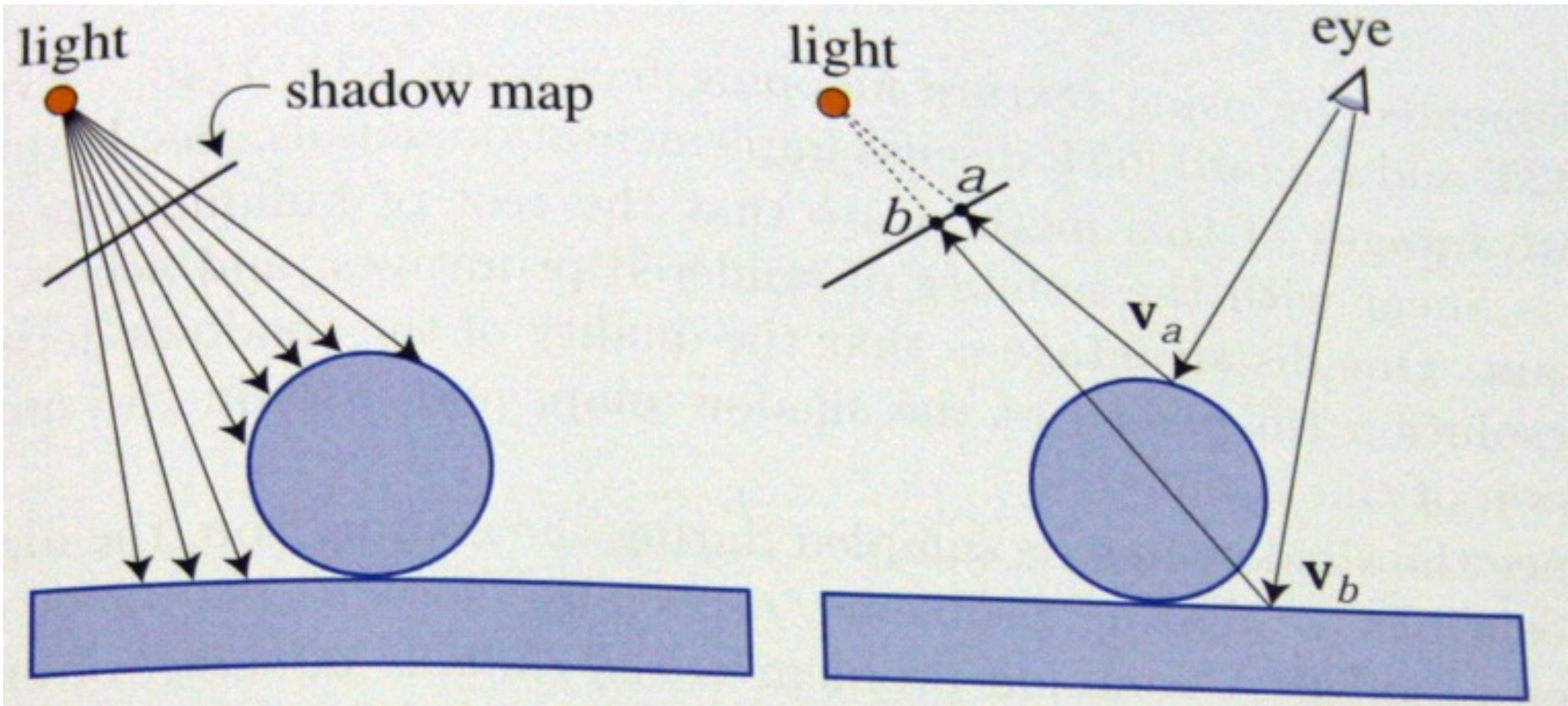
Shadows





- Valuable cue of spatial relationships
- Increases realism

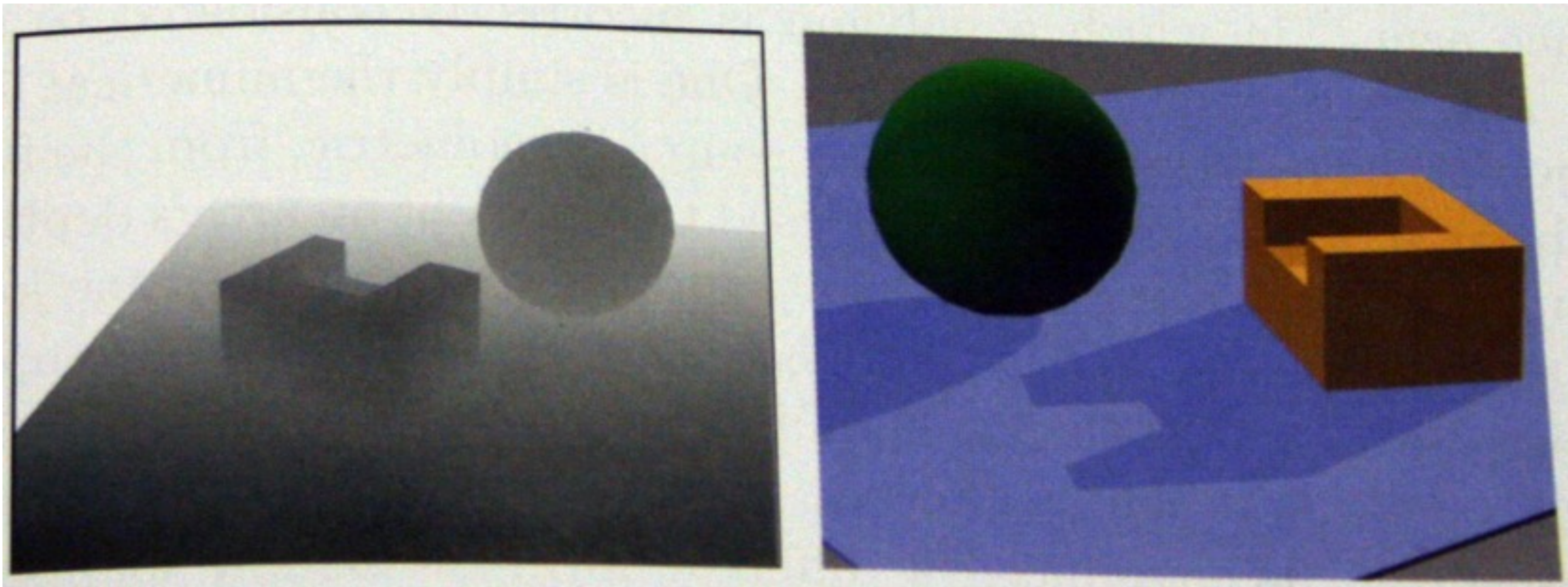
Shadow mapping



- 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

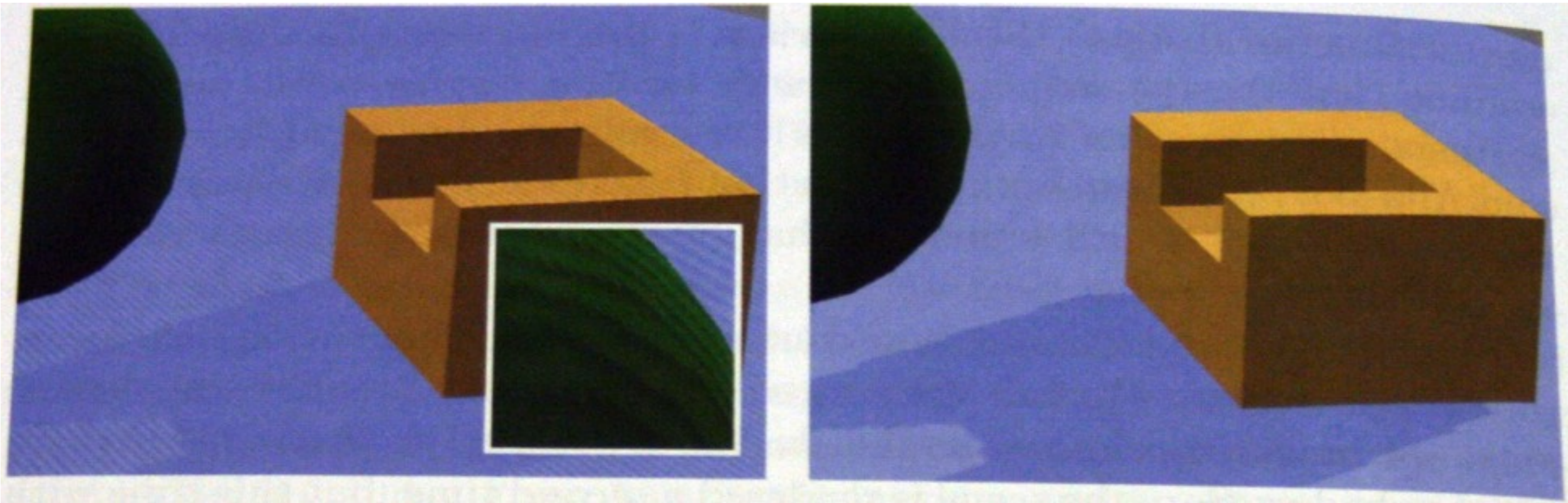
Shadow mapping



- 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

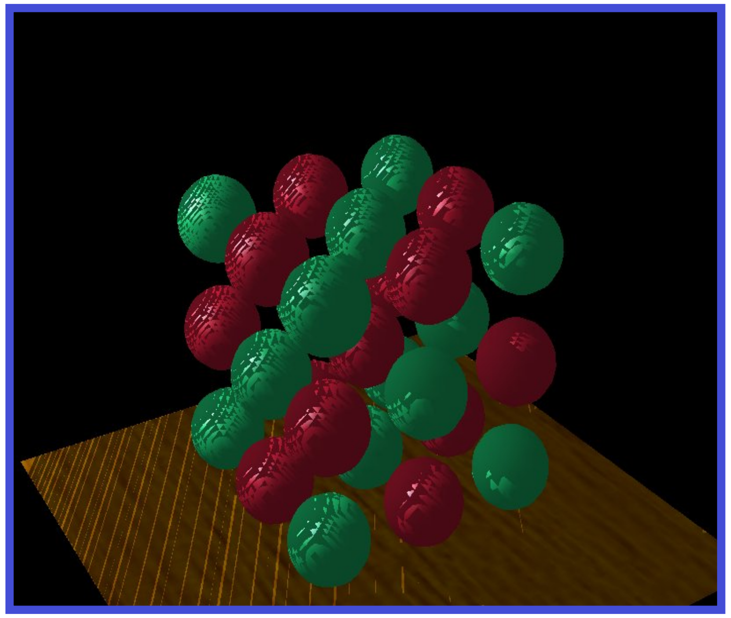
Bias



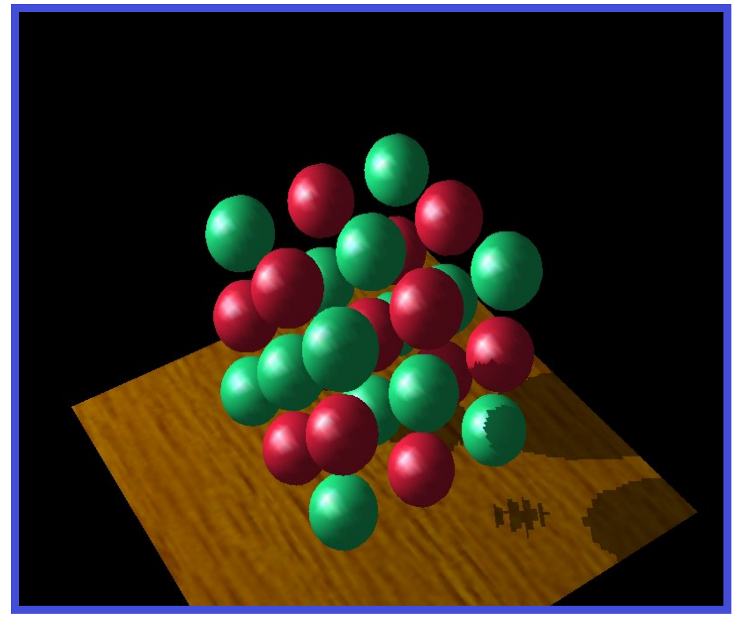
- Numerical imprecision leads to self-shadowing
- Solution: add a bias ε. Change comparison from $w \leq w'$ to $w \leq w' + \varepsilon$
- Can use glPolygonOffset

Akenine-Moeller et al., Real-Time Rendering

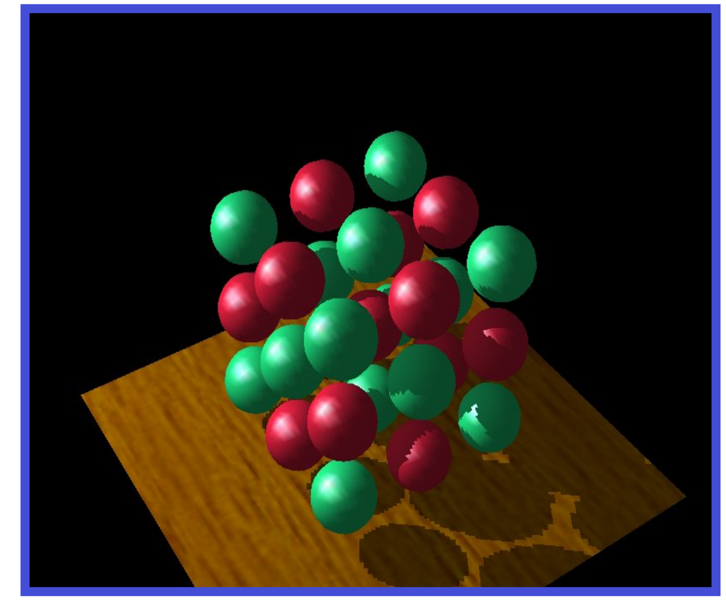
Setting the bias



Too little



Too much

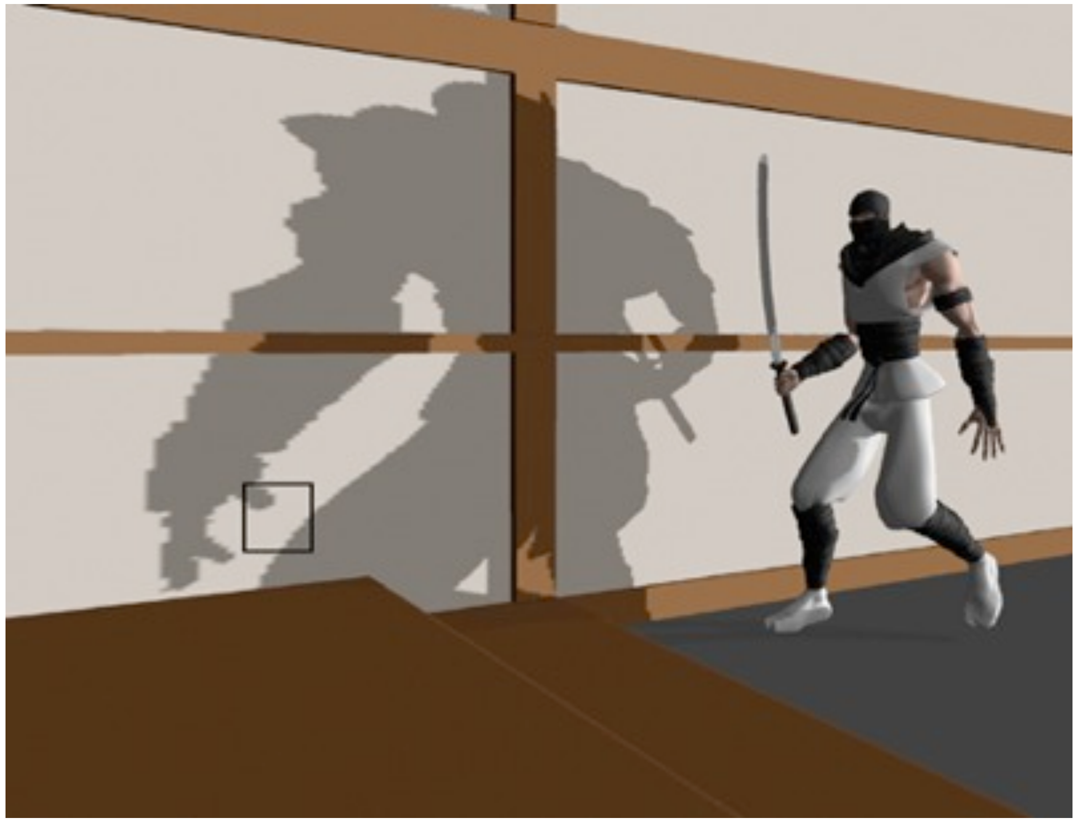


Just right

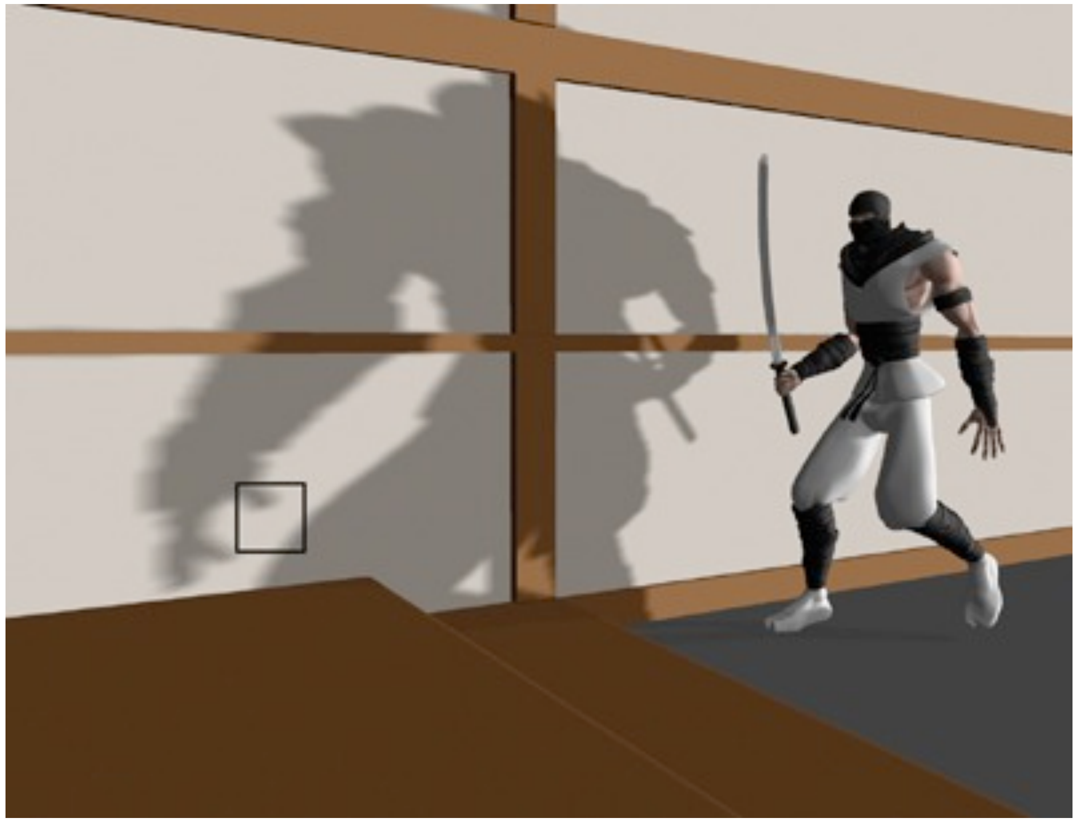
- Numerical imprecision leads to self-shadowing
- Solution: add a bias ε. Change comparison from $w \leq w'$ to $w \leq w' + \varepsilon$
- Can use glPolygonOffset

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

- 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.

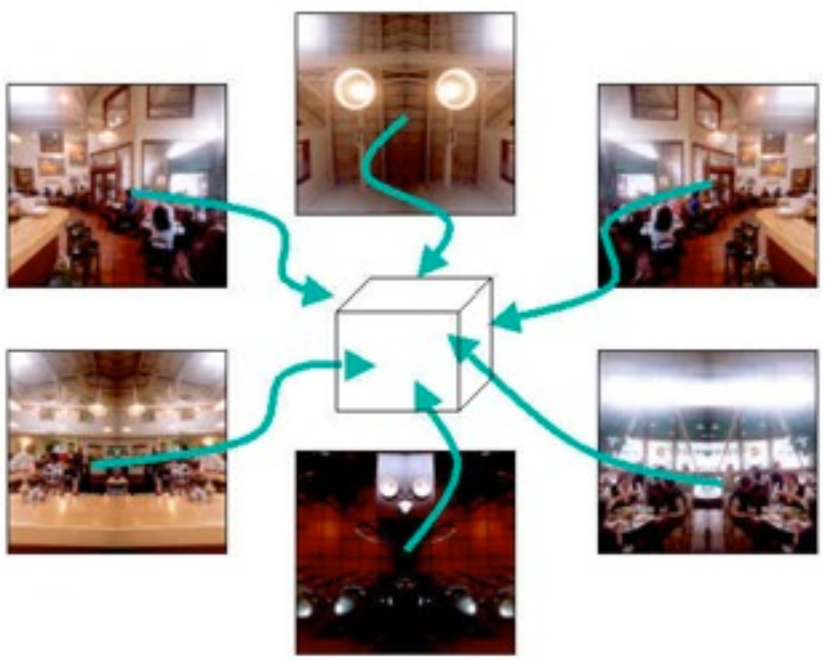
Terminator 2

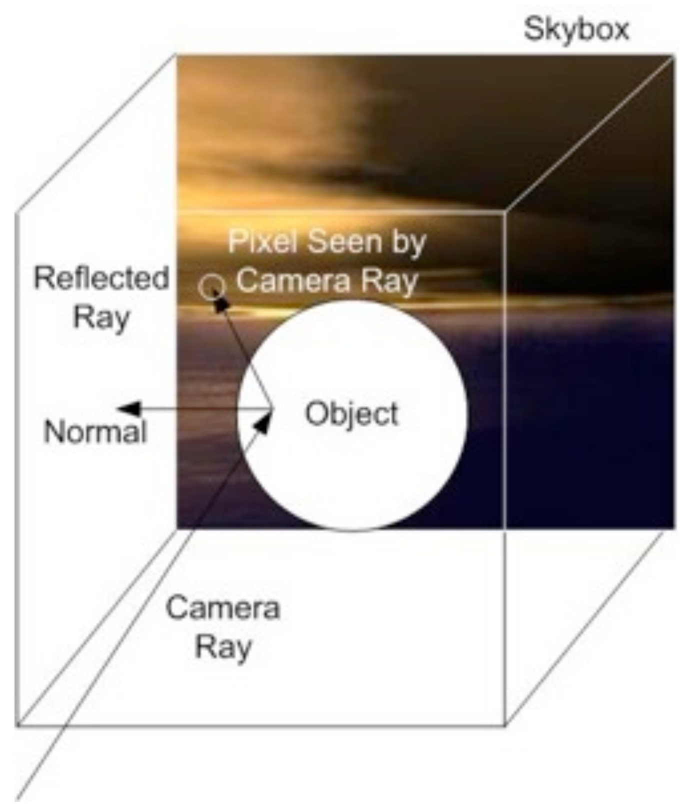
Cube mapping

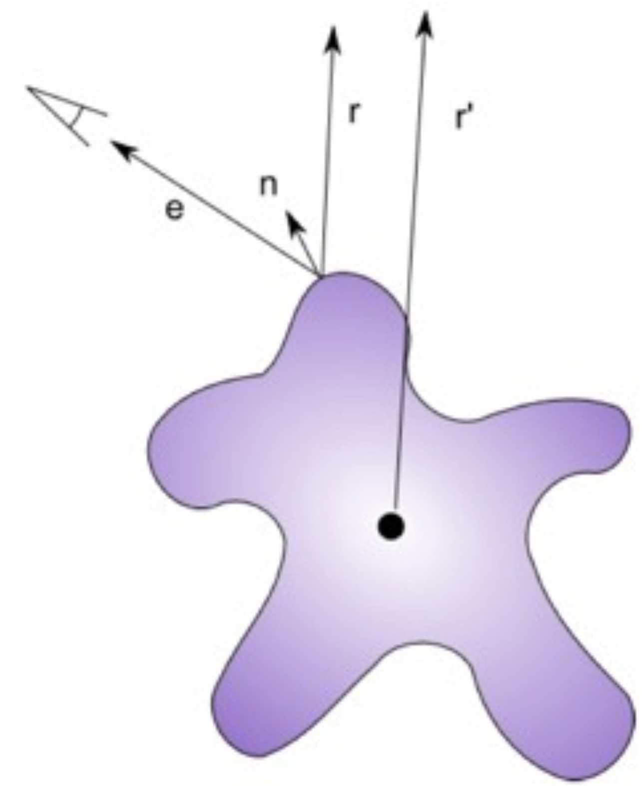
- Render the scene six times, through six faces of a cube, with 90-degree field-of-view for each image.
- Store images in six textures, which represent an omni-directional view of the environment

Greene, 1986

Cube mapping







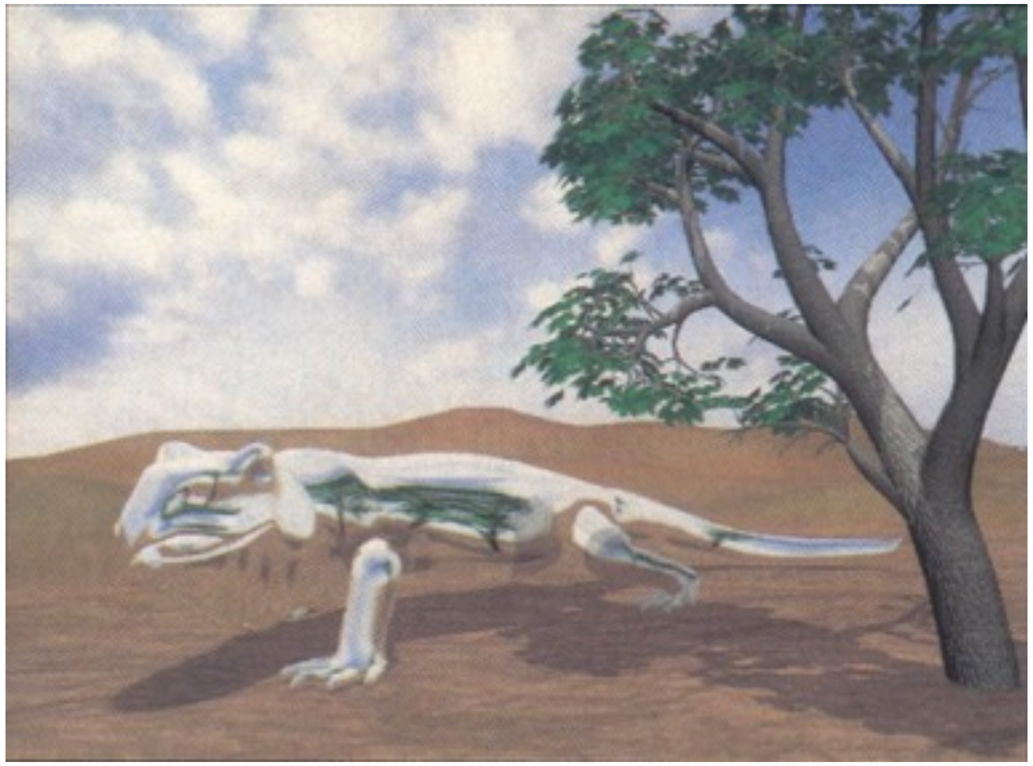
- To compute texture coordinates, reflect the view vector **v** about the normal **n**:

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

- 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)

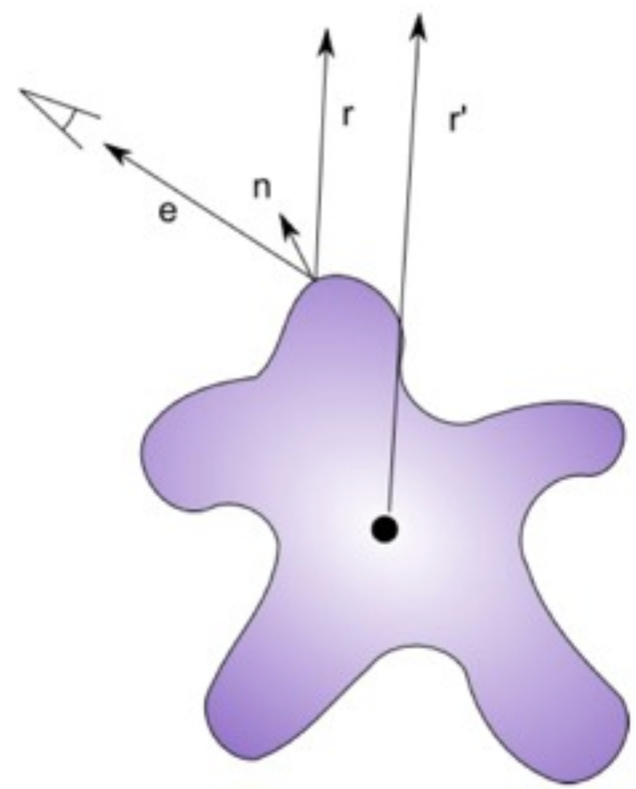
Sphere mapping



- 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

Reflection mapping limitations



- Self-reflections not supported. A concave object will not reflect parts of itself.
- Environment map only correct for the point from which it was rendered. Good approximation for distant reflected objects, but can lead to substantial artifacts in general.