2016 상반기 인턴 및 취업 설명회

- **5월25일(수) 오후 1:00~6:00**
- **N1: 117호**

- SW중심대학 사업의 일환으로 산업협력 콘소시움을 조직하고 학부학생 대상 인턴 및 취업 설명회 개최
- 대상:
  3~4학년, 학석박 졸업예정자
-
- 목적:
  - 학생들에게 방학동안 인턴을 장려하고 인턴을 수행할 회사의 바람직한 인턴 주제를 소개
  - 졸업 예정인 학생들의 취업 설명회

jinah@cs.kaist.ac.kr CS38

			Readings	Homework
Tue	3	Materials	Chap 14	
Wed	4	Lighting setup exercise		HW #3
Thur	5	<Children's Day>		Due (5/6)
	10	Shaders (Review+)	Chap 1~14	HW #4
	11	Open Lab		
	12	Color / Shading	Chap 19/Ext	
	17	Raytracing	Chap 20	
	18	Open Lab		
	19	Light	Chap 21	Due: May 24 11:59PM
	24	Texture Mapping 1	Chap 15	
< E11: #307 >	25	Texture mapping exercise		HW #5
	26	Quiz / HW#5 / Q&A < N1: #112 >		
	31	Texture Mapping 2	Chap 15	
7-10PM	1	CUDA Special Lab (by NVIDIA)		
< N1: #102 >	2	Sampling	Chap 16	
	7	Samplling/Reconstruction	Chap 16/17	
	8	Open Lab		
	9	Geometirc modeling	Chap 22	
	14	Animation	Chap 23	
	21	Final Exam		

Announcement

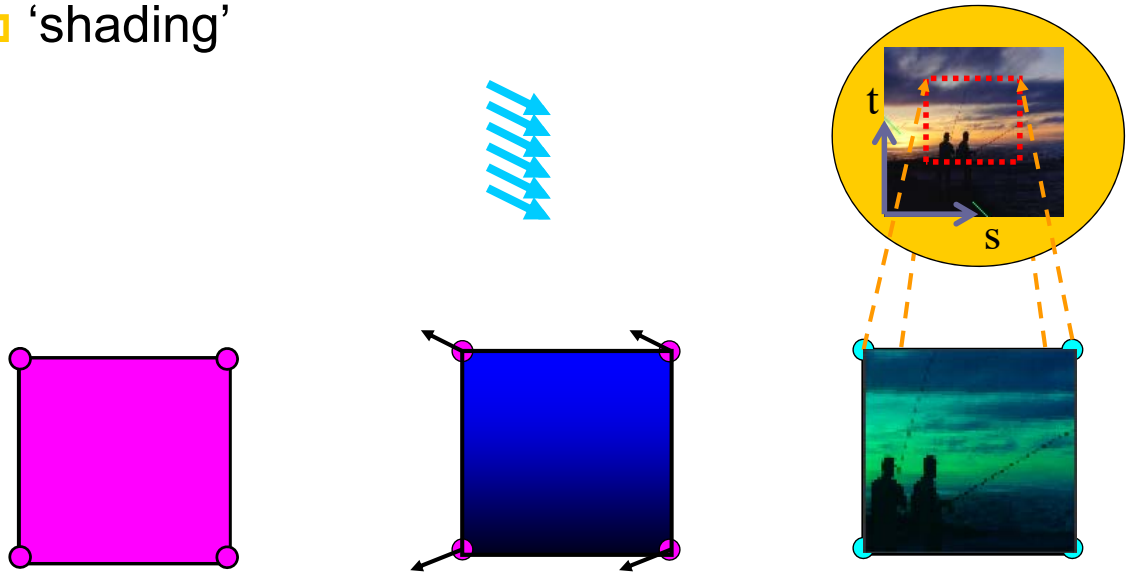
- Mandatory Lab tomorrow
  - Texture mapping exercise
- This Thursday
  - Quiz on 'Texture mapping'
    - Related to the lab materials
  - HW #5
  - Q&A for programming to TA
- Grading for Homework #3 will be posted next week.

Texture Mapping

Chapter 15

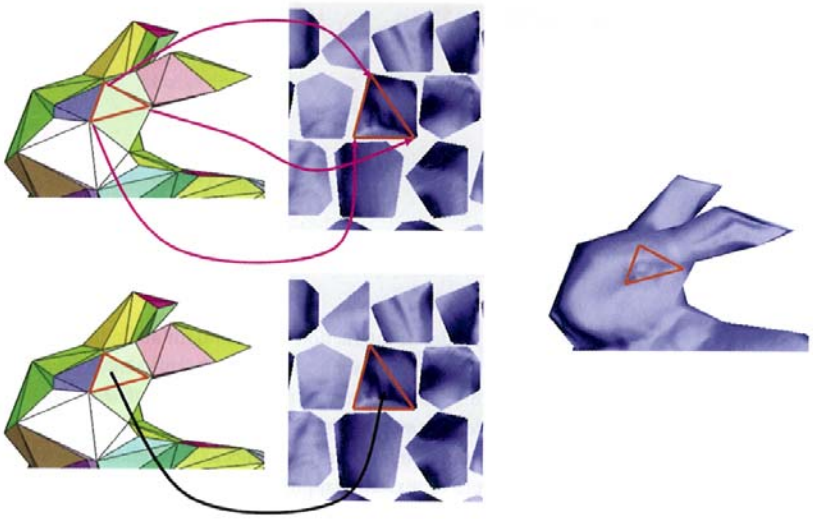
‘coloring’

- ‘shading’



Texture mapping

- In basic texturing, we simply ‘glue’ part of an image onto a triangle by specifying *texture coordinates* at the three vertices.

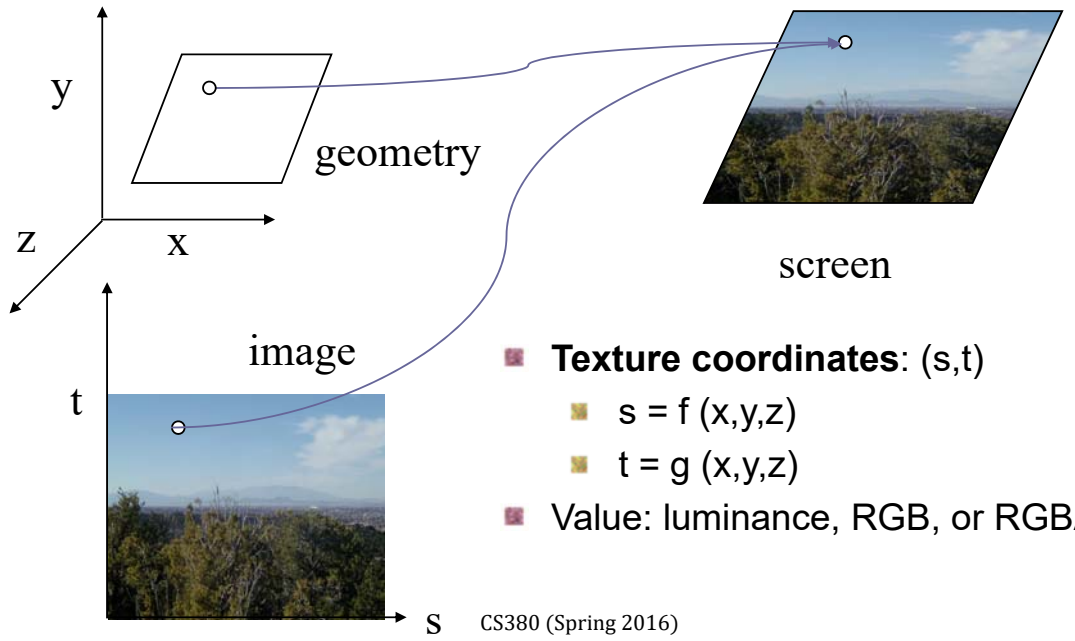


Texture mapping

- Bunch of OpenGL codes to load a texture and set various parameters (mipmap, wrapping rules, etc.).
- A uniform variable is used to point to the desired texture unit.
- Varying variables are used to store texture coordinates.
- In this simplest incarnation, we just fetch r,g,b values from the texture and send them directly to the frame buffer.
- Alternatively, the texture data could be interpreted as, say, the diffuse material color of the surface point, which would then be followed by the diffuse material computation described earlier.

Texture mapping

- 2D texture: n x m continuous array of ***texels***



- **Texture coordinates**: (s,t)
  - s = f (x,y,z)
  - t = g (x,y,z)
- Value: luminance, RGB, or RGBA

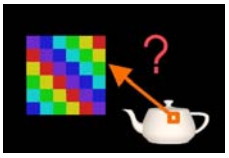
http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_1.htm

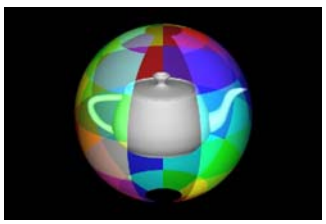
Some key points from slides by Rosalee Wolfe (DePaul University)

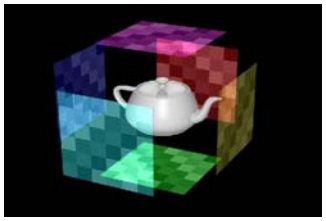
- Texture mapping creates the appearance of grass without the cost of rendering thousands of polygons.
- Most mapping techniques use object coordinates. (vs. world coordinates)
- It's often useful to transform the bounding geometry so its coordinates range between zero and one.
- For each pixel in an object, we encounter the question, "Where do I have to look in the texture map to find the color?" To answer this question, we consider two things: map *shape* and map *entity.*
  - *Planar, cylindrical, square, sphere*
  - *Position, surface normal, reflection, ..*
  - *Parametric patch, non-linear function*

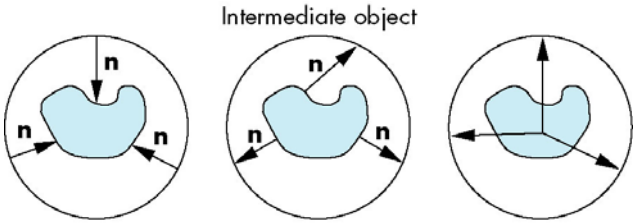
Texture Mapping

- 2D vs. 3D
- Map shape and map entity
  - Planar, sphere, cylinder, cube, parametric surface
    - *Intermediate surface (2-step mapping process)*









We'll comeback to the issues (e.g., parameters) for texture mapping after the lab session tomorrow

Normal Mapping

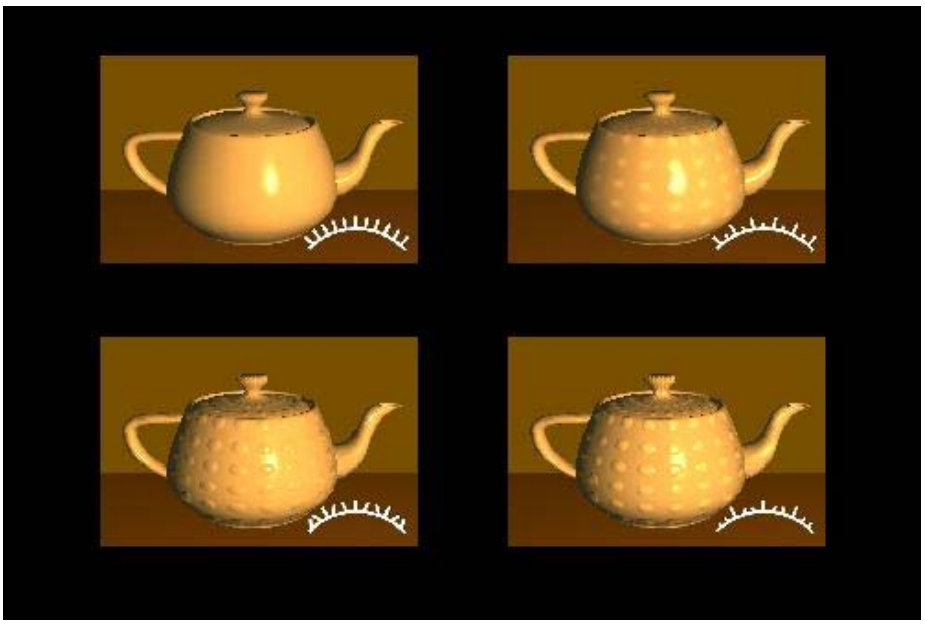
Bump Mapping

- The data from a texture can also be interpreted in more interesting ways.
- In normal mapping, the r,g,b values from a texture are interpreted as the three coordinates of the normal at the point.
- This normal data can then be used as part of some material simulation
- Normal data has three coordinate values, each in the range [-1…1], while RGB textures store three values, each in the range [0…1] (0…255)
  - So need some conversions

Bump Mapping

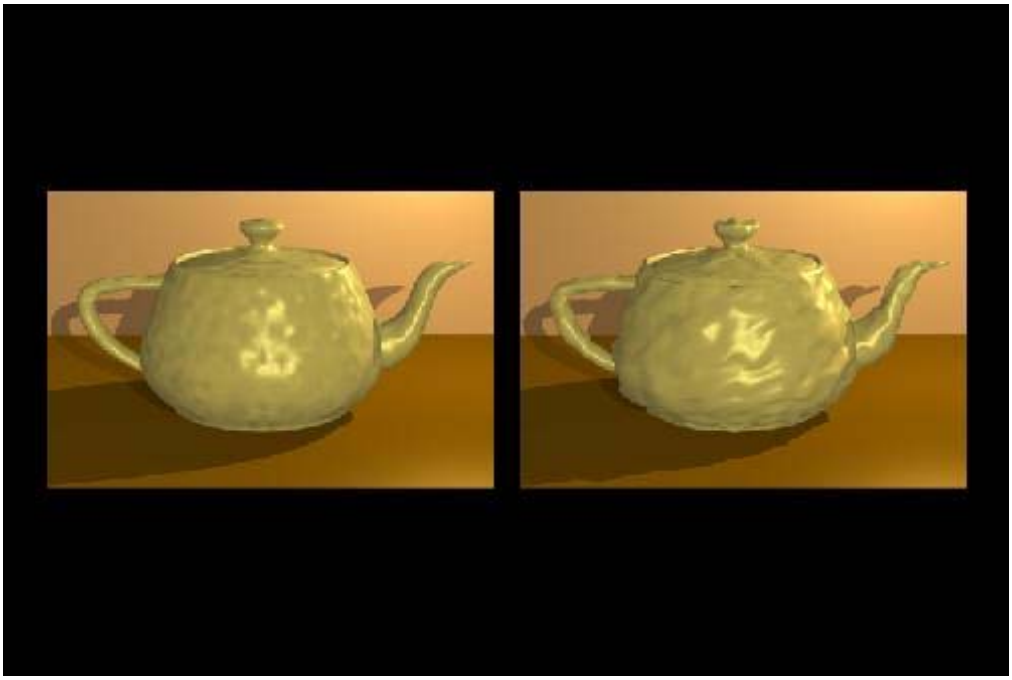
- Create a realistic non-smooth surface (e.g., orange)
  - surface modeling
  - texture mapping
    - What if change the light direction?
  - bump mapping
    - Perturb the normal vector.
    - Do not perturb the surface geometry itself.
    - Bump mapping affects object surfaces, making them appear rough, wrinkled, or dented by altering the surface normal before the shading calculation takes place. It's possible to change a surface normal magnitude or direction.

Bump mapping affects object surfaces, making them appear rough, wrinkled, or dented. Bump mapping alters the surface normals before the shading calculation takes place. It's possible to change a surface normals magnitude or direction



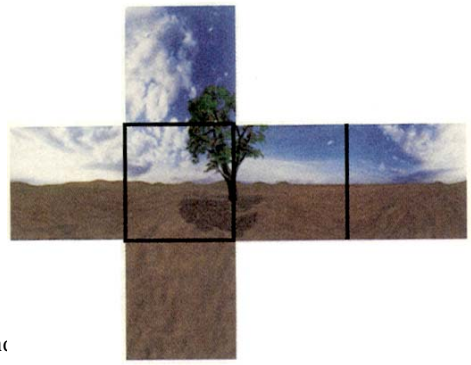
Bump mapping does not change the underlying geometry of the model, but fools the shading algorithm to produce an interesting surface.

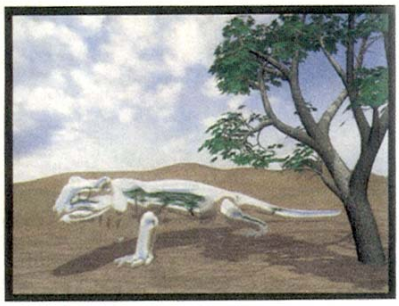
In contrast, *displacement mapping* alters an object's geometry. Compare the profiles and the shadows cast by these two objects. Which is bump mapped?



Environment Cube Maps

- Textures can also be used to model the environment in the distance around the object being rendered.
- In this case, we typically use 6 square textures representing the faces of a large cube surrounding the scene.
- Each texture pixel represents the color as seen along one direction in the environment.
- This is called a *cube map*. GLSL provides a cube-texture data type, samplerCube specifically for this purpose.





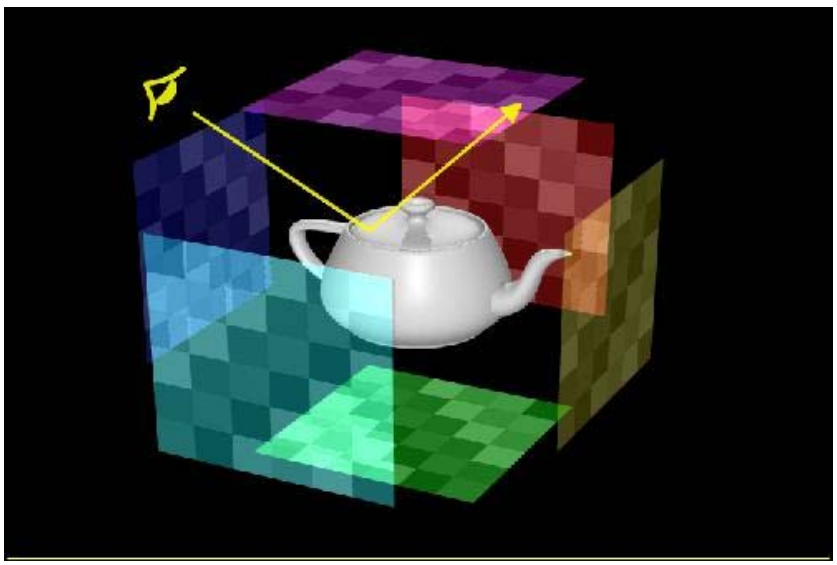
Environment Cube Maps

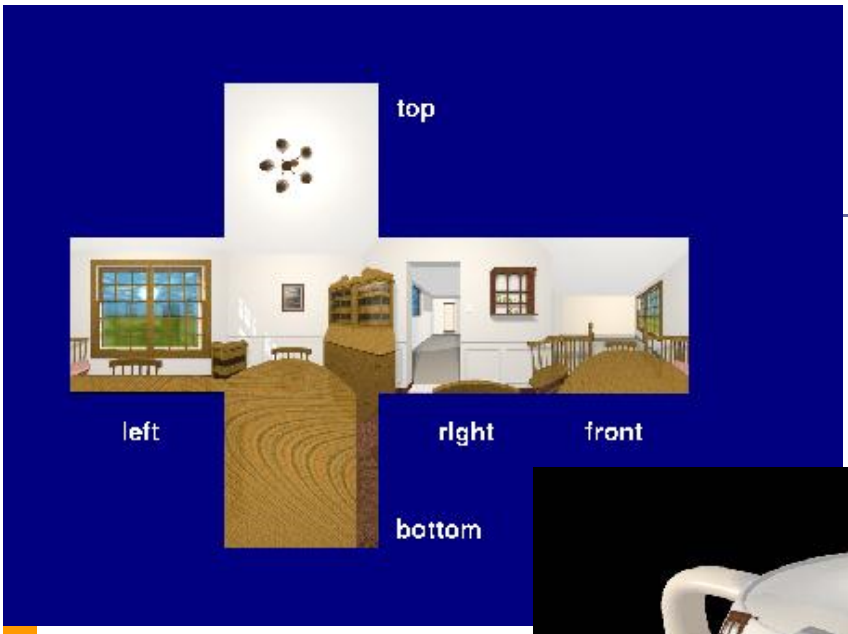
- During the shading of a point, we can treat the material at that point as a perfect mirror and fetch the environment data from the appropriate incoming direction.
- We calculate $B(\vec{v})$ in the previous lecture.
- This bounced vector will point points towards the environment direction, which would be observed in a mirrored surface.
- By looking up the cube map, using this direction, we give the surface the appearance of a mirror.

Environment Mapping

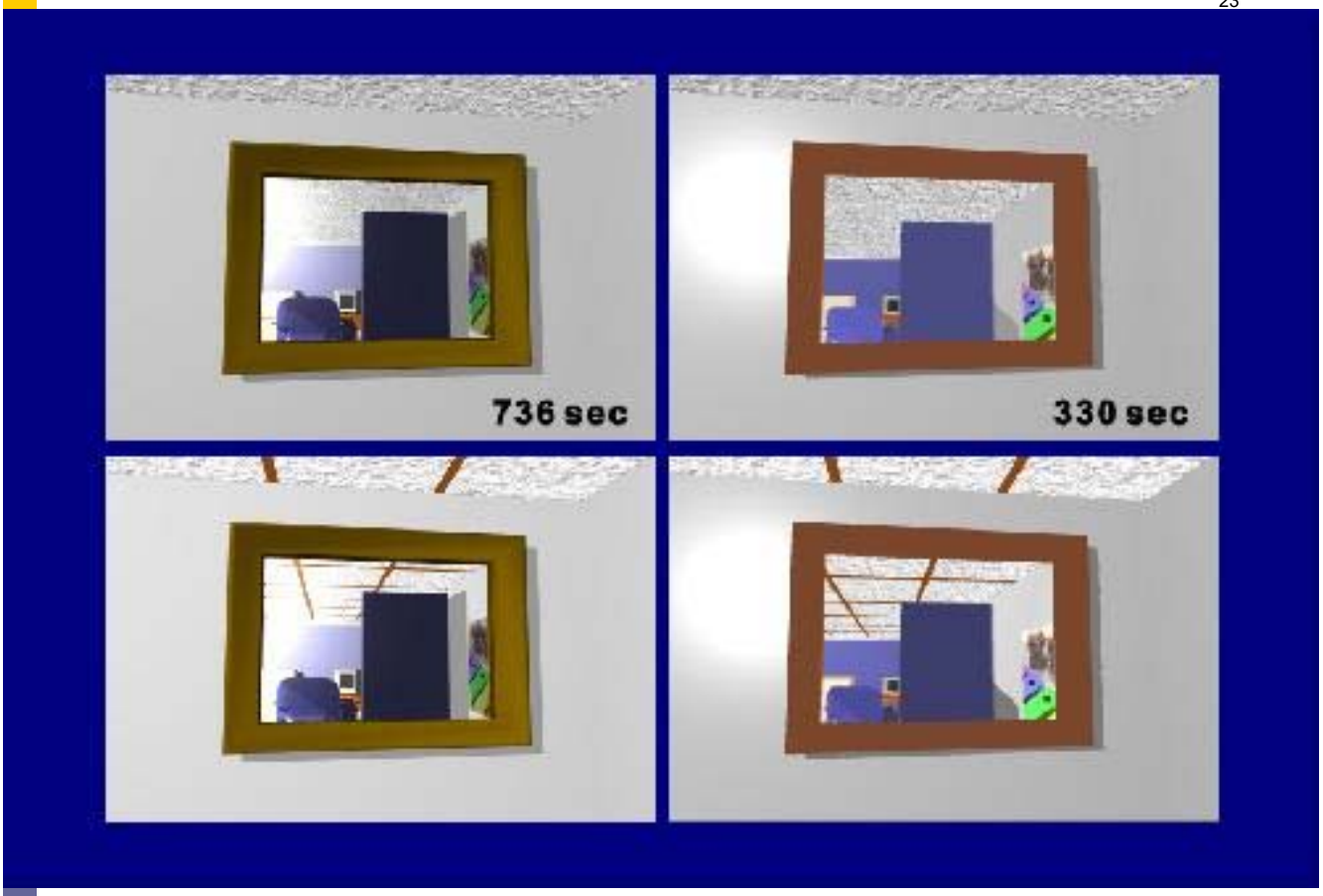
- Method
  1. Set up an imaginary camera.
  2. Remove the object.
  3. Render the image for the imaginary camera.
     (This will create an environmental map.)
  4. Treat the environmental map as a texture.
  5. Perform texture mapping.

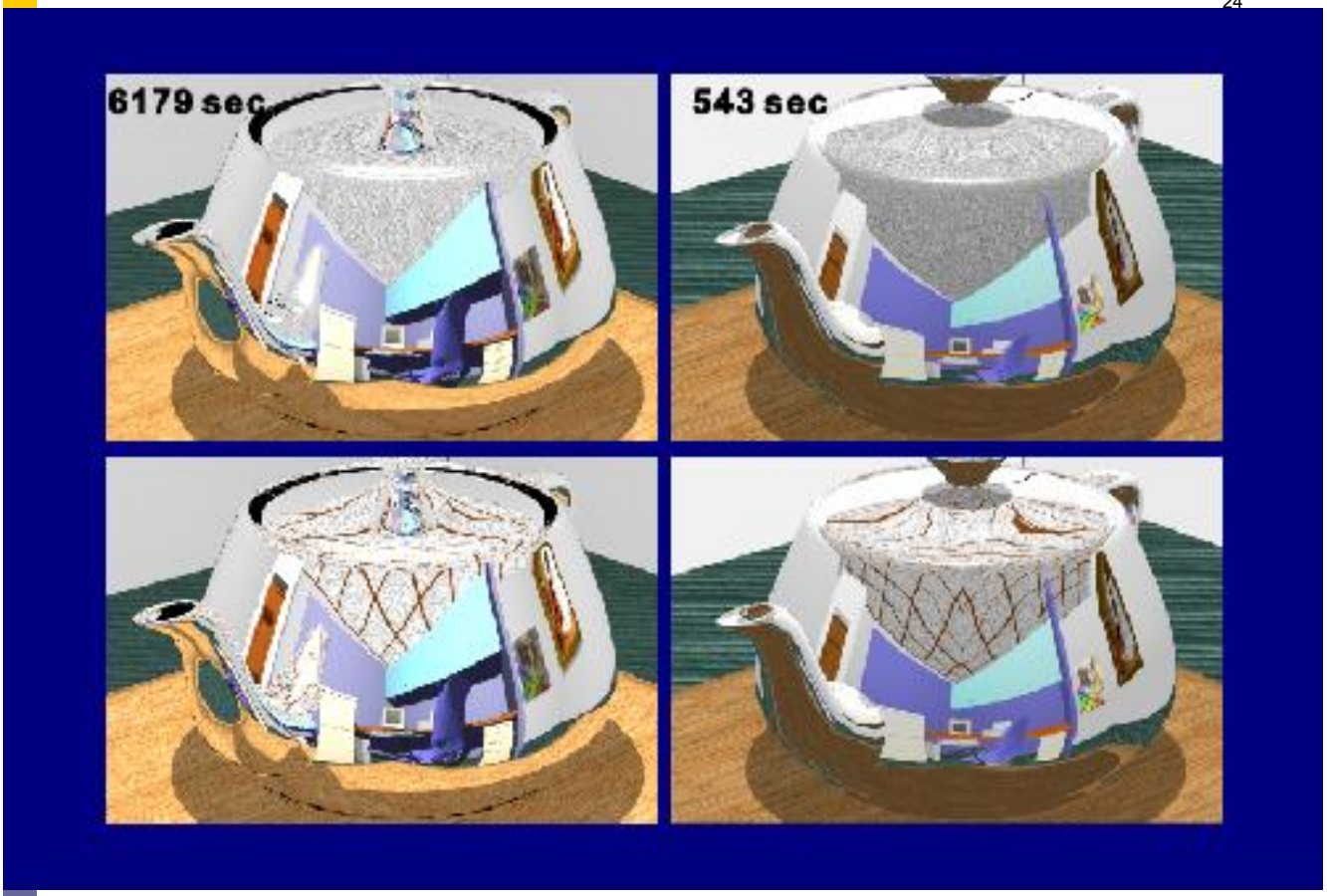
- Environment mapping is a cheap way to create reflections. Environment mapping is a two-dimensional texture mapping technique that uses a map shape of a box and a map parameter of a reflection ray.





Here are side-by-side comparisons of raytracing and environment mapping. What differences can you see?


736 sec
330 sec

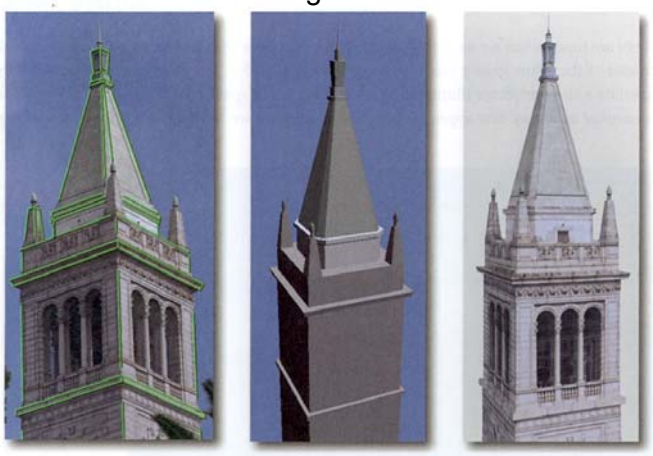

6179 sec
543 sec

Projector Texture Mapping

- There are times when we wish to glue our texture onto our triangles using a *projector* model, instead of the affine gluing model.
- For example, we may wish to simulate a slide projector illuminating some triangles in space.

Rendered geometric model

Actual photograph

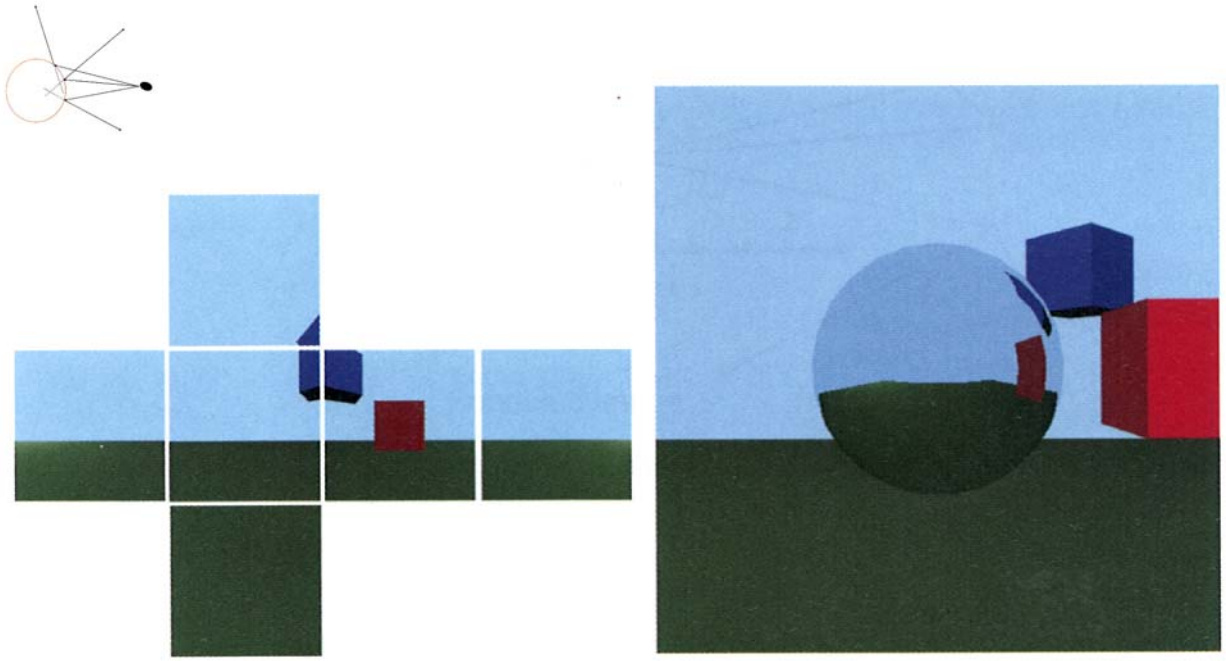


Rendered with projector texture mapping

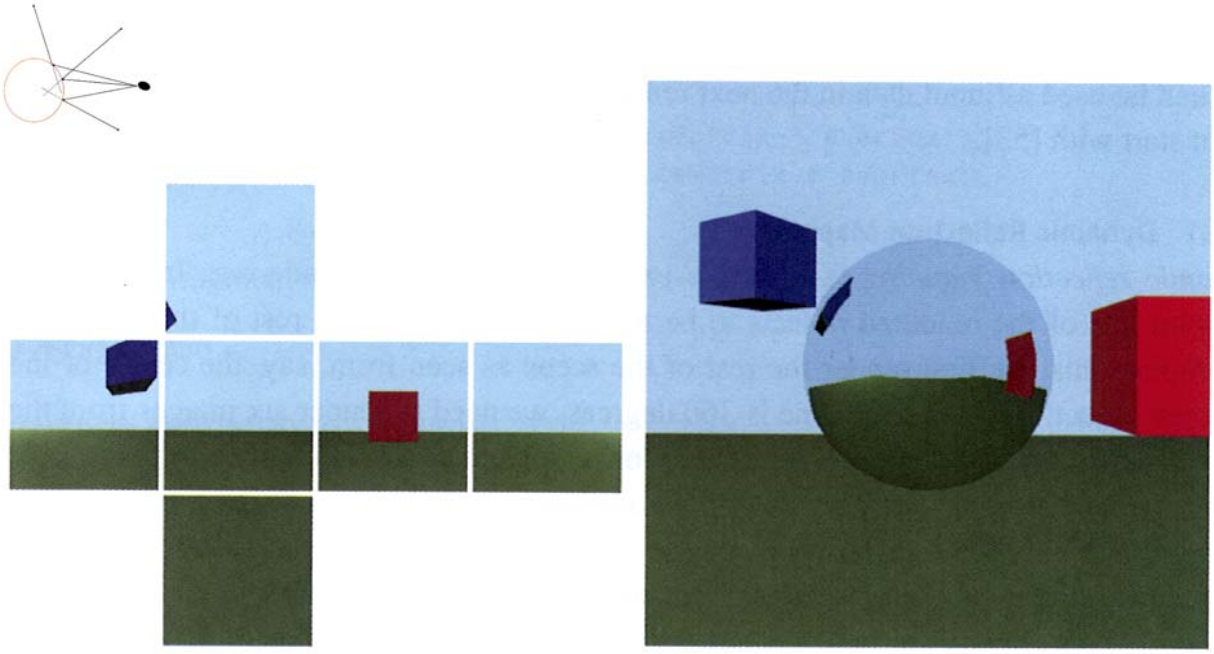
Multipass

- More interesting rendering effects can be obtained using multiple rendering passes over the geometry in the scene.
- In this approach, the results of all but the final pass are stored offline and not drawn to the screen.
- To do this, the data is rendered into something called, a FrameBufferObject, or FBO.
- After rendering, the FBO data is then loaded as a texture, and thus can be used as input data in the next rendering pass.

Dynamic Reflection Mapping

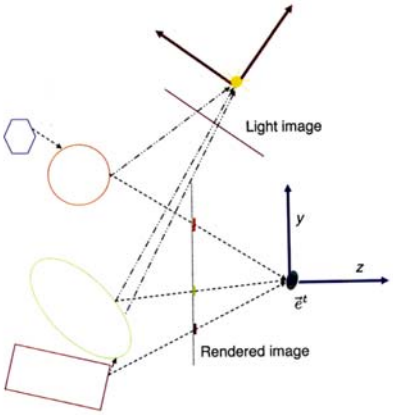


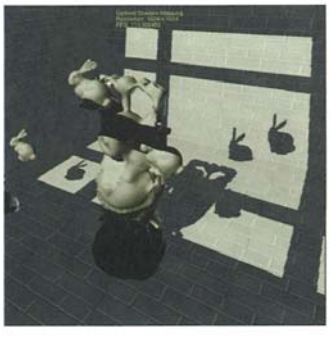
Dynamic Reflection Mapping



Shadow Mapping

- The idea is to first create and store a z-buffered image from the point of view of the light, and then compare what we see in our view to what the light saw it its view.





Shadow Mapping

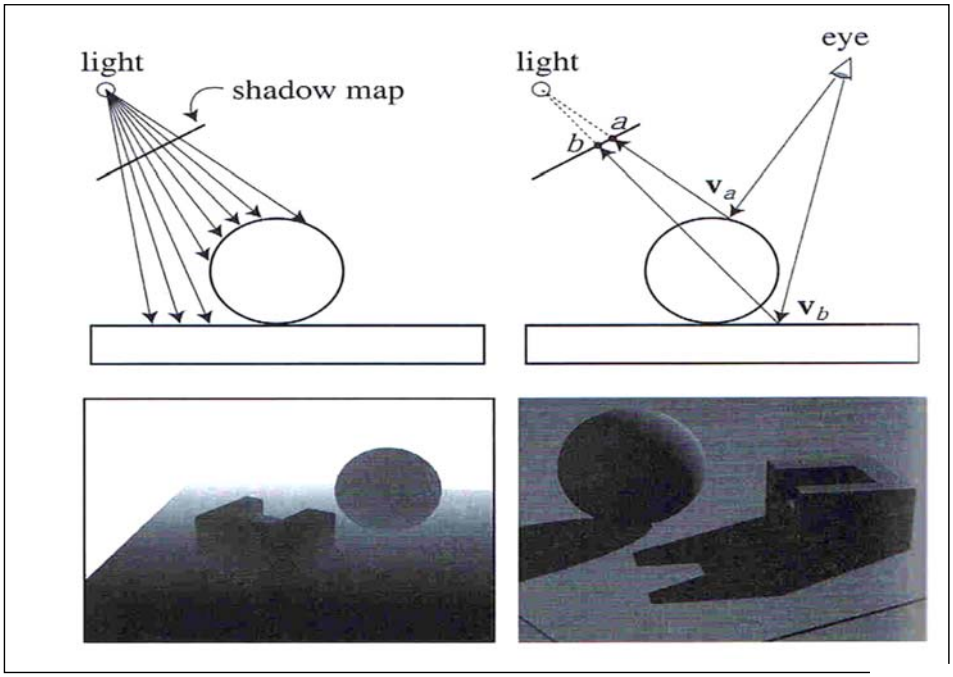
- If a point observed by the eye is not observed by the light, then there must be some occluding object in between, and we should draw that point as if it were in shadow.

Shadow Map

- Basic idea:
  objects that are not visible to the light are in shadow
- How to determine whether an object are visible to the eye?
  - Use z-buffer algorithm, but now the “eye” is light, i.e., the scene is rendered from light’s point of view
  - This particular z-buffer for the eye is called *shadow map*

Shadow Map Algorithm

- illustration



Shadow Map Algorithm

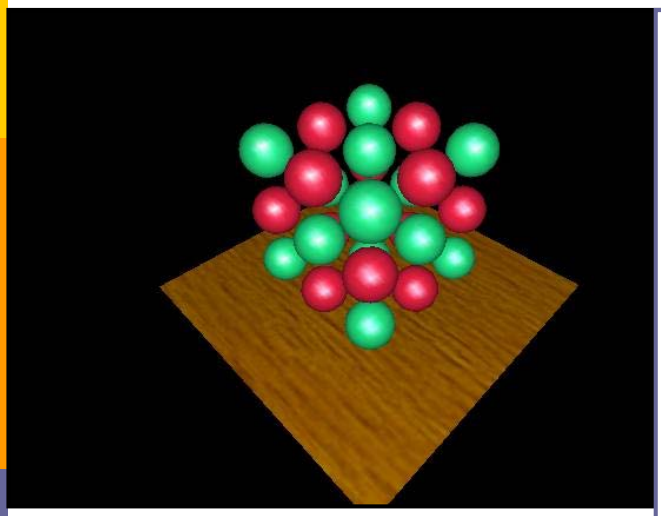
1. Render the scene using the light as the camera and perform z-buffering
2. Generate a light z buffer (called shadow map)
3. Render the scene using the regular camera, perform z-buffering, and run the following steps:

   3.1 For each visible pixel with [x,y,z] in world space, perform a transformation to the light space (light as the eye) [x1,y1,z1]

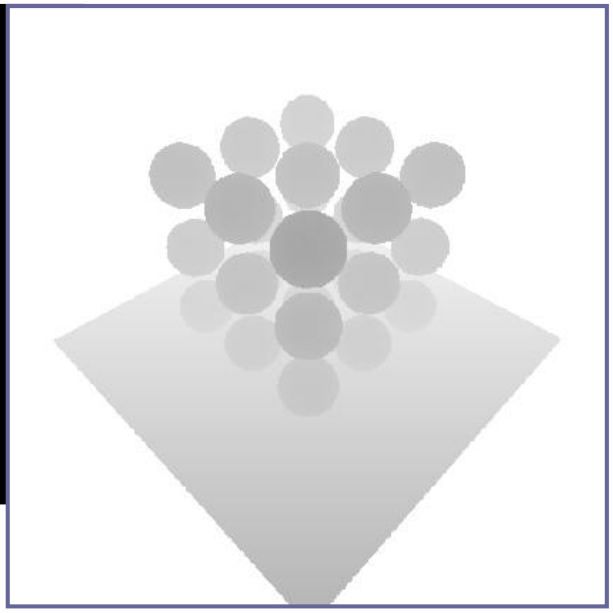
   3.2 Compare z1 with shadow_map[x1,y1]

   - If z1 is closer to light, then the pixel in question is not in shadow; otherwise the pixel is shadowed

1st Pass

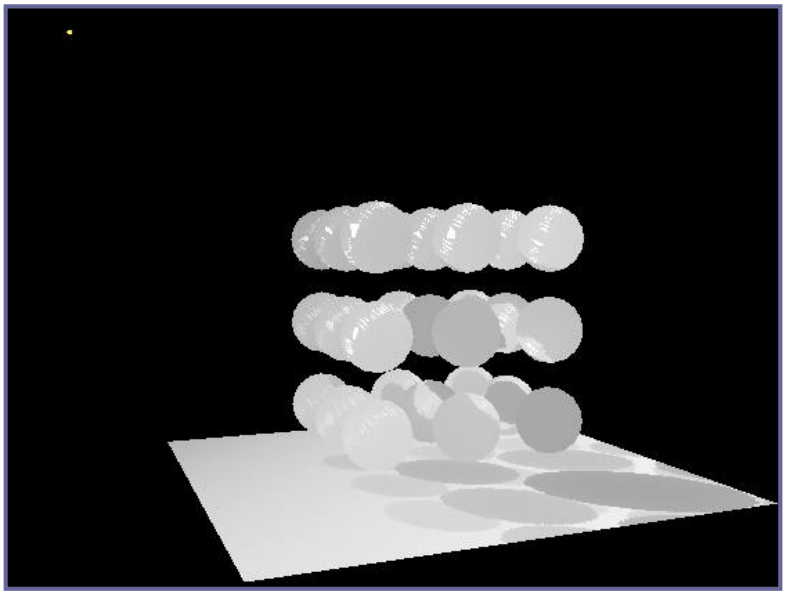


View from light



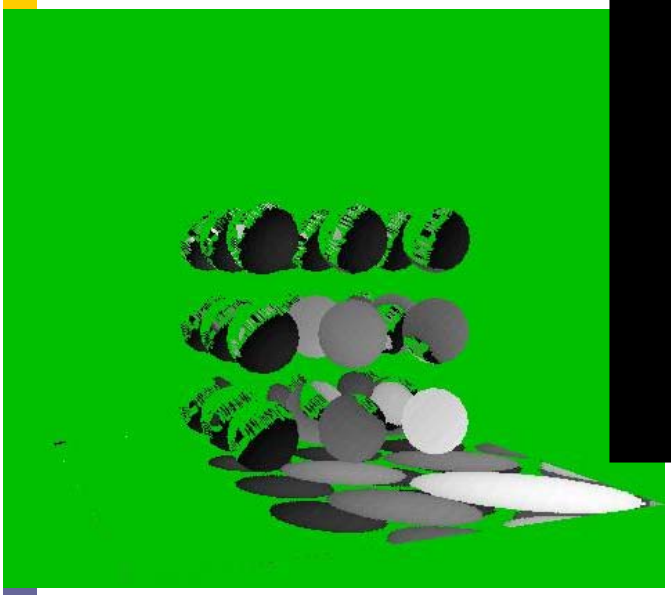
Depth Buffer (shadow map)

2nd Pass

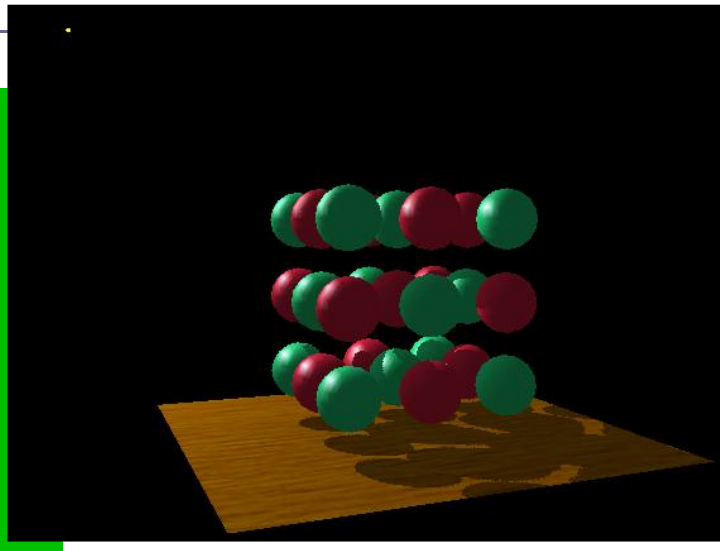


Visible surface depth

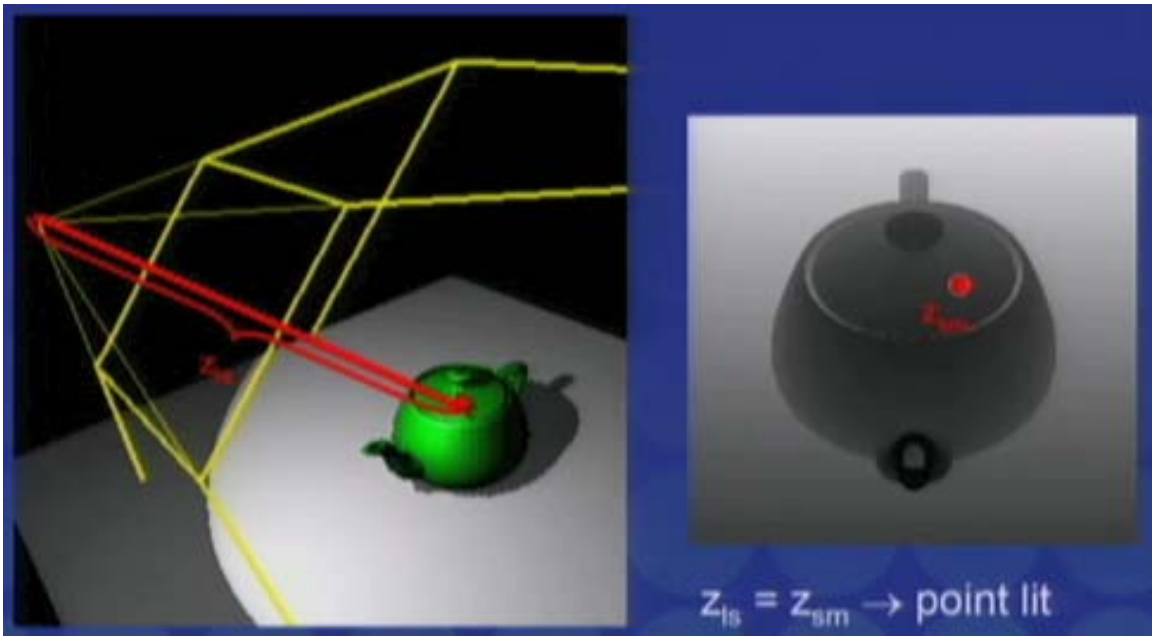
2nd Pass

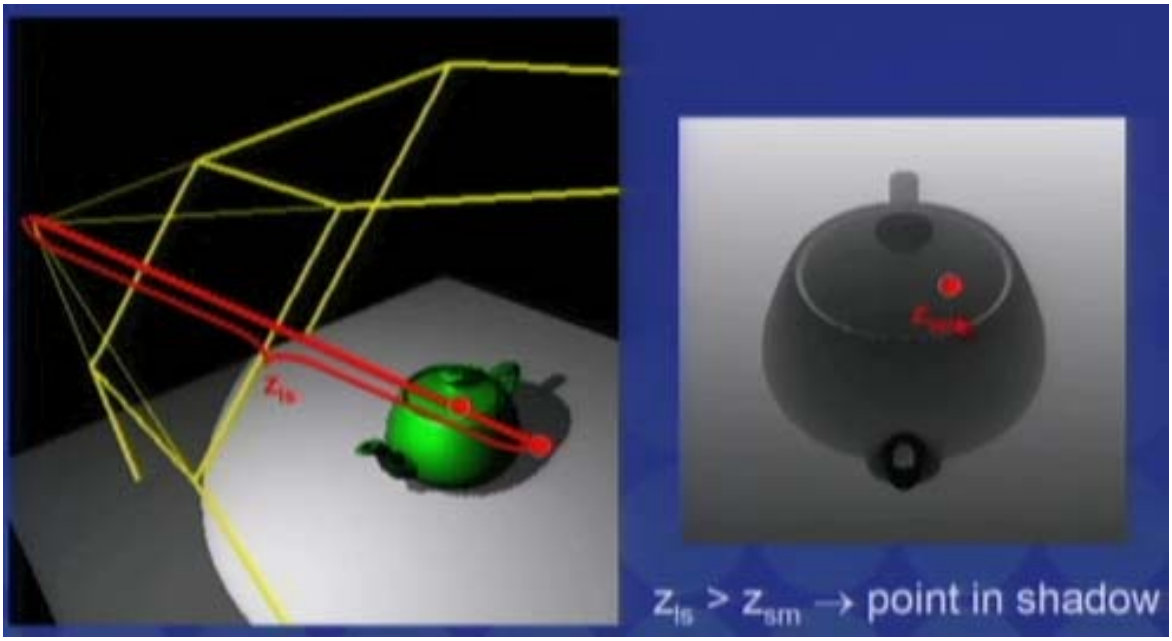


Non-green in shadow



Final Image





Advantages

- Only Z-buffering is required
- The shadow map can be used in several frames as long as neither the light source nor the objects move
- General purpose graphics hardware can be used
- The cost of building the shadow map is linear

Shadow map issues

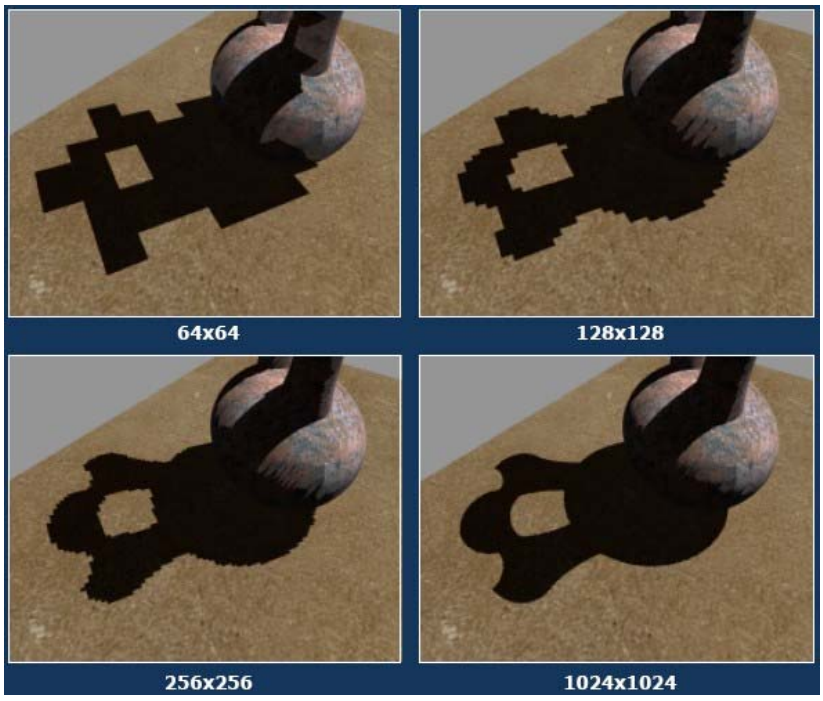
- Shadow quality depends on
  - Shadow map *resolution* – aliasing problem
    - Z resolution – the shadow map is often stored in one channel of texture, which typically has only 8 bits
    - Some hardware has dedicated shadow map support, such as Xbox and GeForce3
  - Self-shadow aliasing – caused by different sample positions in the shadow map and the screen <caused by *precision*>

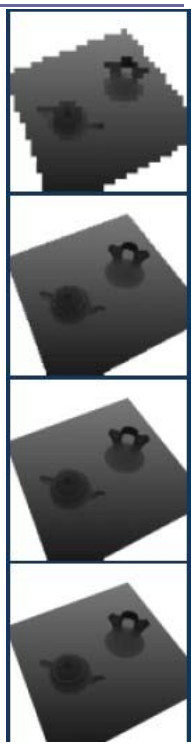
Shadow map aliasing problem

- The shadow looks blocky – when one single shadow map pixel covers several screen pixels
- This is a similar problem to texture magnification

<We will learn about this next lecture>

Increase the shadow map resolution

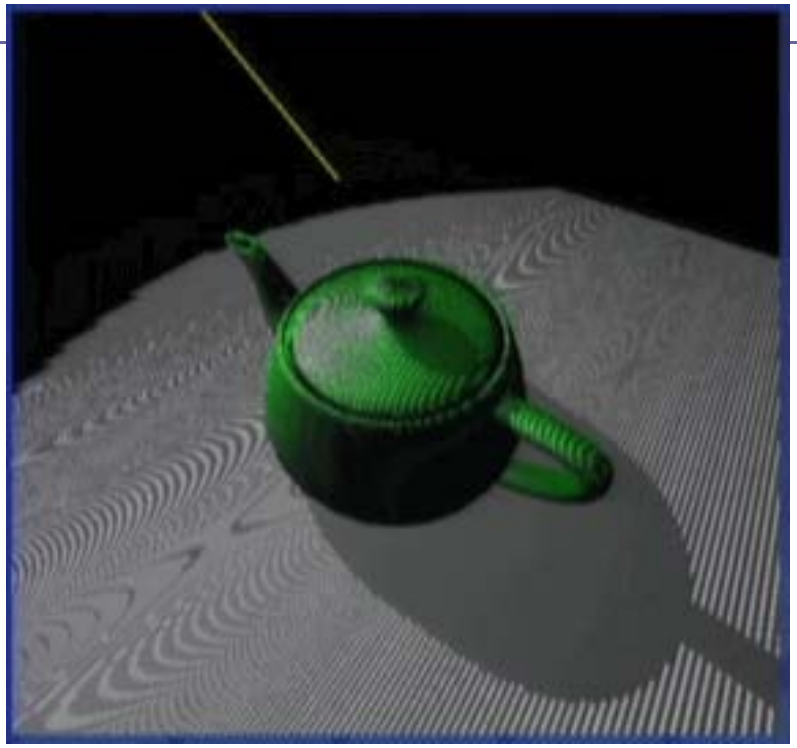




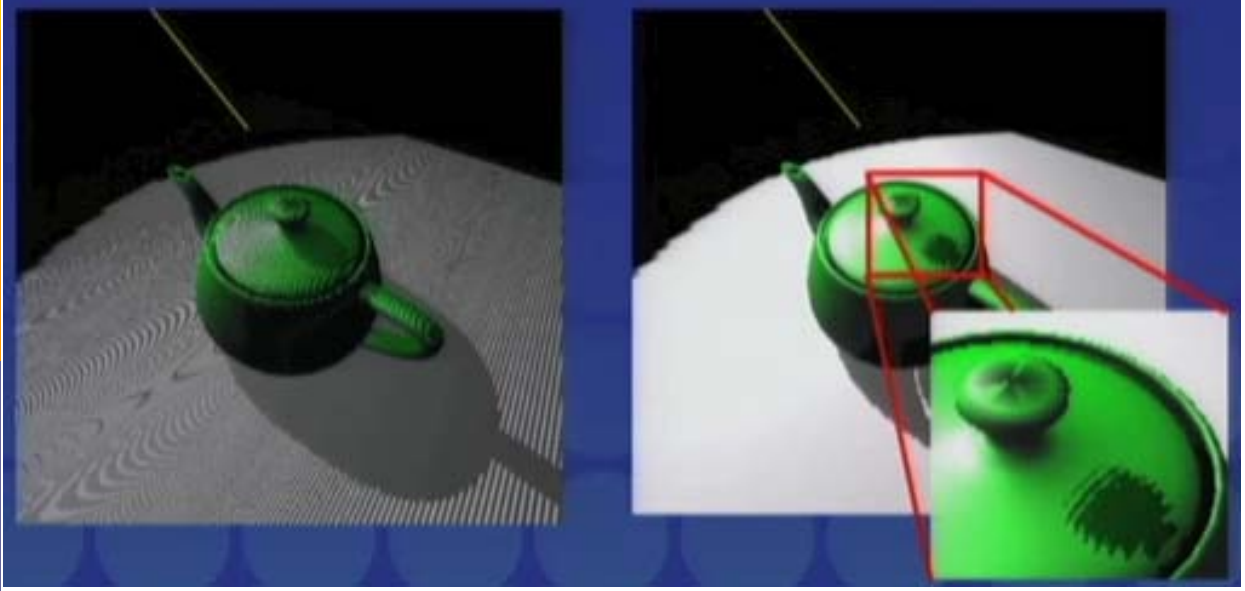
Self-shadow aliasing

- a polygon is incorrectly considered to shadow itself because of the imprecision
  - Samples generated for the light are generally not exactly at the same location as the screen samples.
  - When the light's stored depth value is compared to the viewed surface's depth
  - The light's value may be slightly lower than the surface's
- Solution
  - ***Bias factor***
  - Increase ***precision*** of Z-buffer

    Make sure the light frustum's near plane is as far away from the light as possible and the far plane is as close as possible

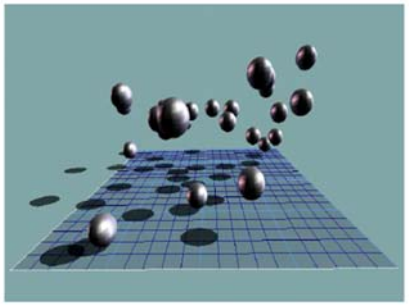


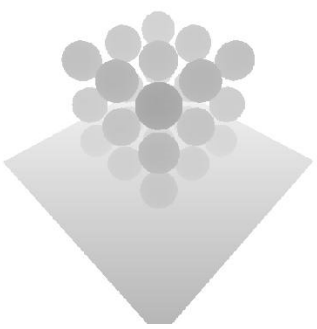
- bias too small → surface acne
- bias too large → shadow leaks

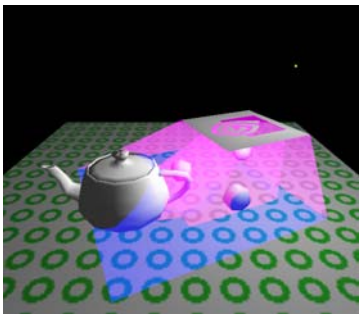


Shadow Algorithms

- hard shadows
  - Planar Shadows
  - **Shadow Maps**
  - Shadow Volume

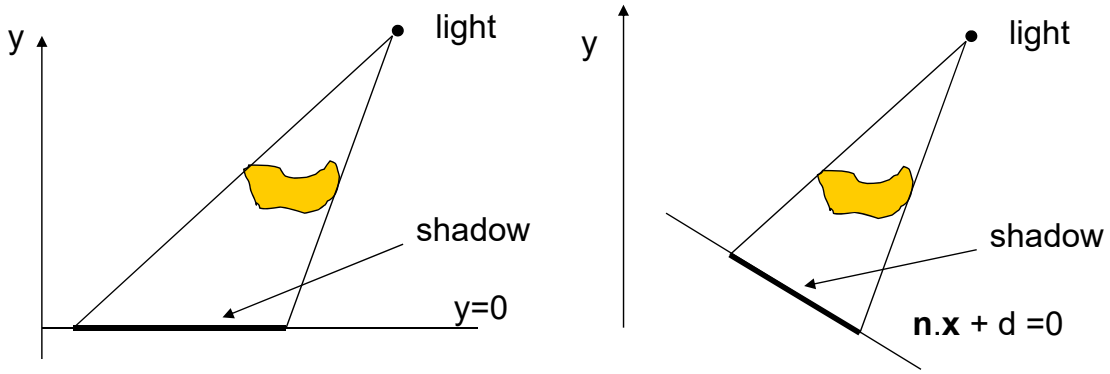






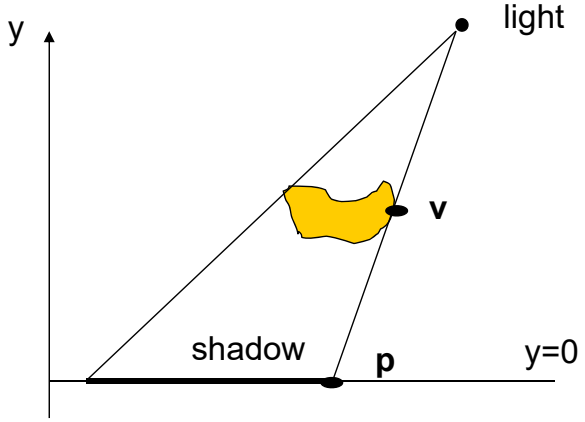
Planar Shadows

- The simplest algorithm – shadowing occurs when objects cast shadows on planar surfaces (projection shadows)



Planar Shadows

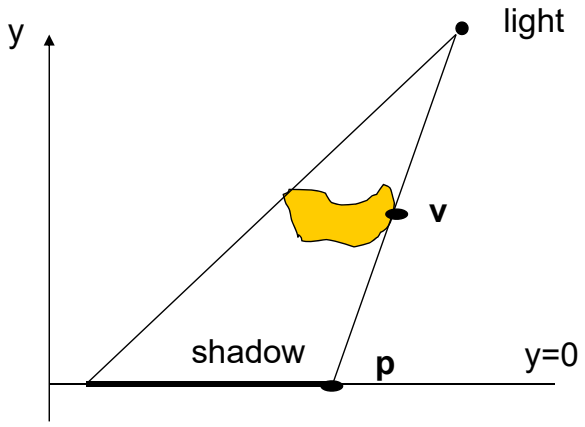
- Shadow receiver is an axis plane
  - Just project all the polygon vertices to that plane and form shadow polygons



Given:
- Light position **l**
- Plane position y = 0
- Vertex position v

Calculate: **p**

Planar Shadows



$$\frac{p_x - l_x}{v_x - l_x} = \frac{l_y}{l_y - v_y}$$

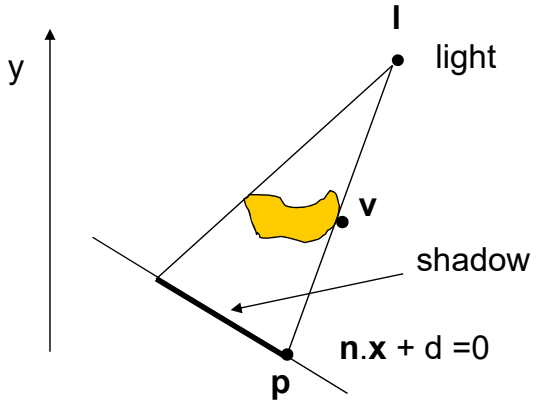
$$p_x = \frac{l_y v_x - l_x v_y}{l_y - v_y}$$

$$\mathbf{M} = \begin{pmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{pmatrix}$$

Han-Wei Shen (Ohio Univ)

Planar Shadows

- How about arbitrary plane as the shadow receiver?



Plane equation: **n**.**x** + d = 0 or

ax + by + cz + d = 0

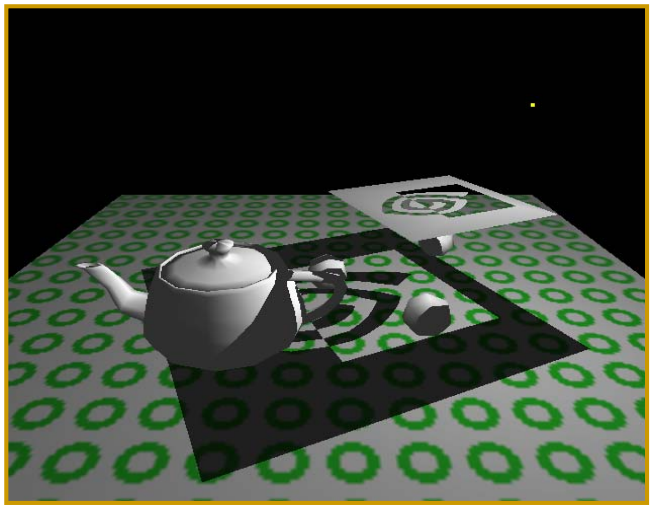
where **n** = (a,b,c) - plane normal

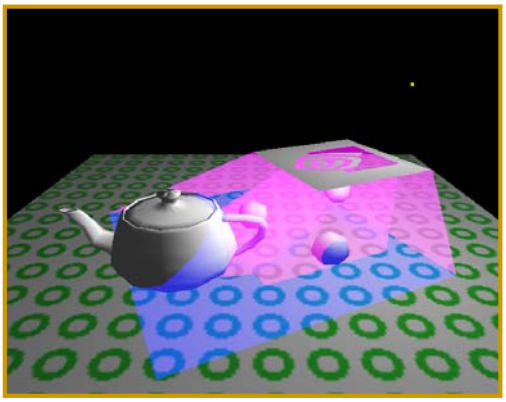
Given light position **l** , **v**

Find **p**

Shadow Volumes

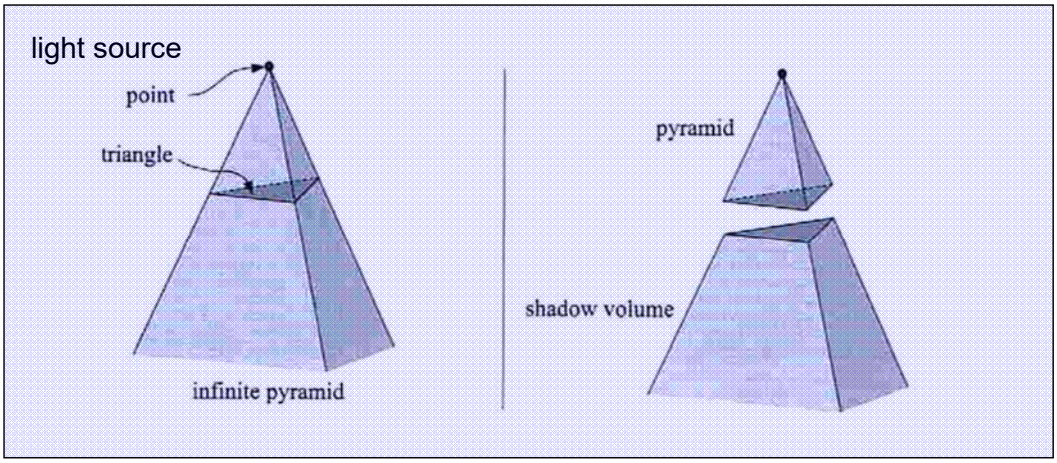
- A more general approach for receivers that have arbitrary shapes



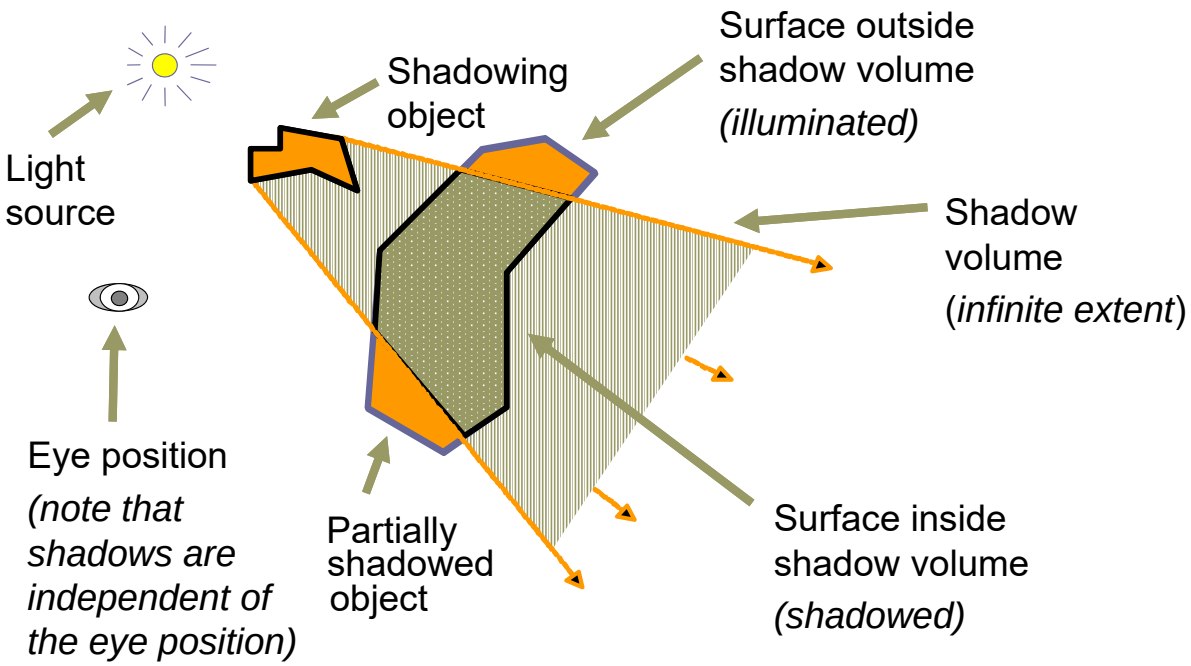


What is shadow volume?

- A volume of space formed by an occluder
- Bounded by the edges of the occluder
- Any object inside the shadow volume is in shadow



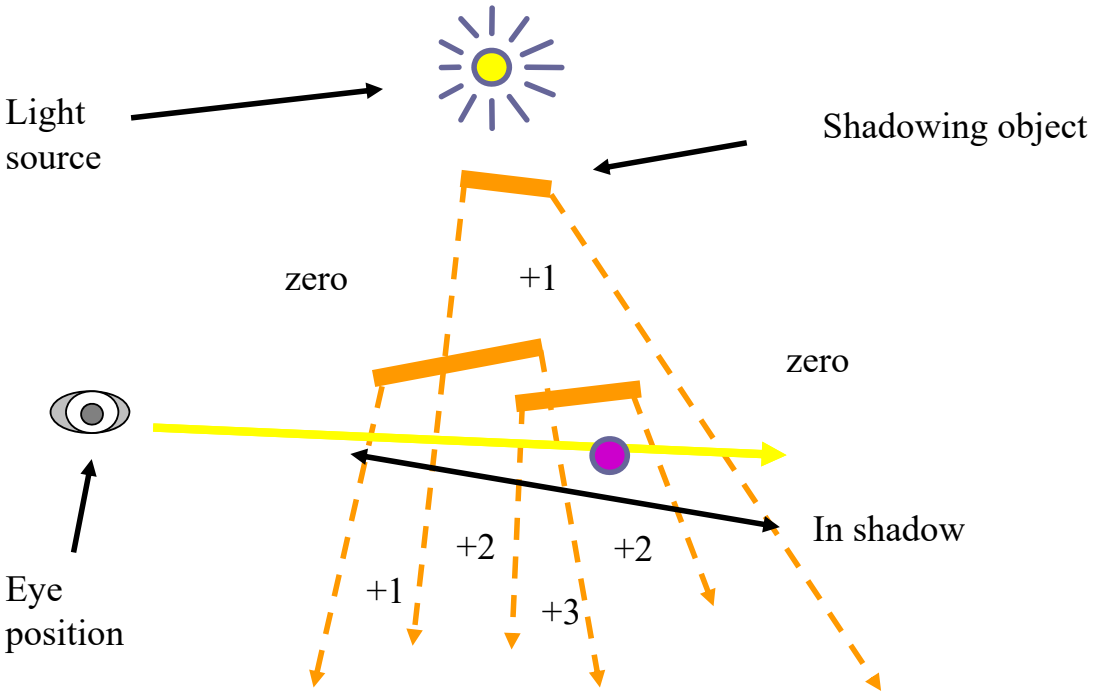
2D Cutaway of a Shadow Volume



In Shadow or not?

- Use shadow volume to perform such a test
- How do we know an object is inside the shadow volume?
  1. Allocate a counter
  2. Cast a ray into the scene
  3. Increment the counter when the ray enter a front-facing polygon of the shadow volume (enter the shadow volume)
  4. Decrement the counter when the ray crosses a back-f acing polygon of the shadow volume (leave the shadow volume)
  5. When we hit the object, check the counter.
     - If counter >0 ; in shadow
     - Otherwise - not in shadow

Counter for Shadow Volume



Real time shadow volume

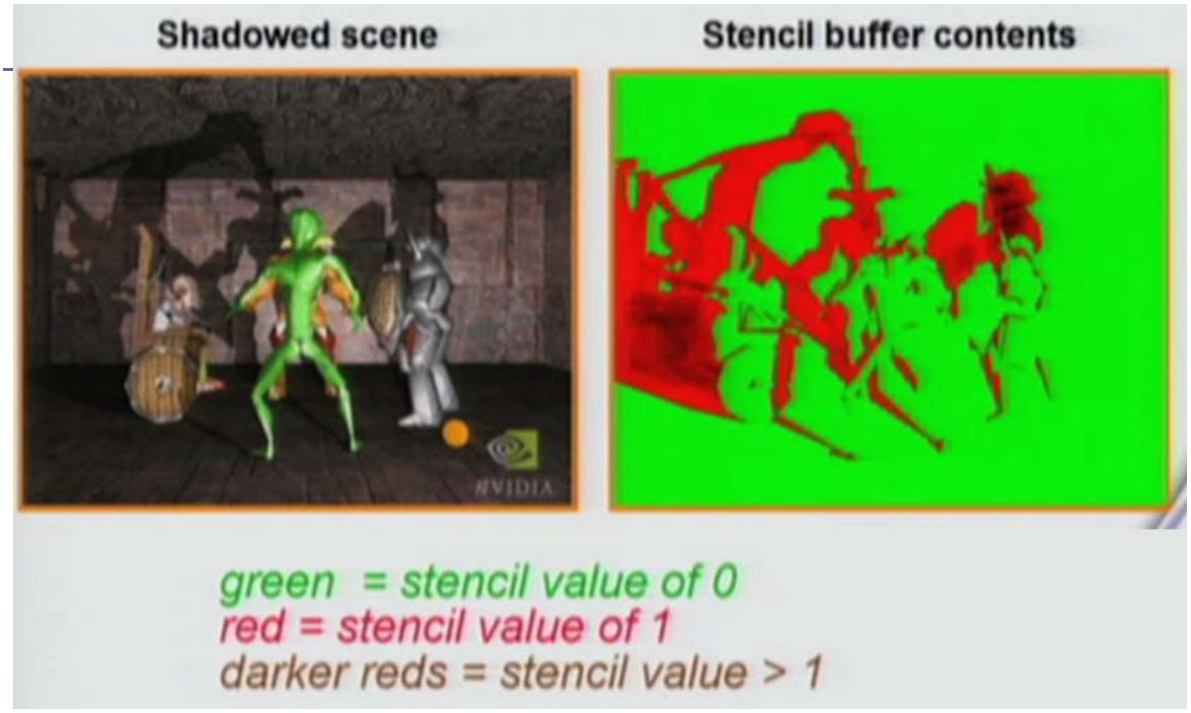
- How can we render the idea of shadow volume in real time?
  - Use OpenGL Stencil buffer as the counter
- Stencil buffer?
  - Similar to color or depth buffer, except it's meaning is controlled by application (and not visible)
  - Part of OpenGL fragment operation – after alpha test before depth test
  - Control whether a fragment is discarded or not
    - Stencil function (Stencil test) - used to decide whether to discard a fragment
    - Stencil operation – decide how the stencil buffer is updated as the result of the test

Stencil Function

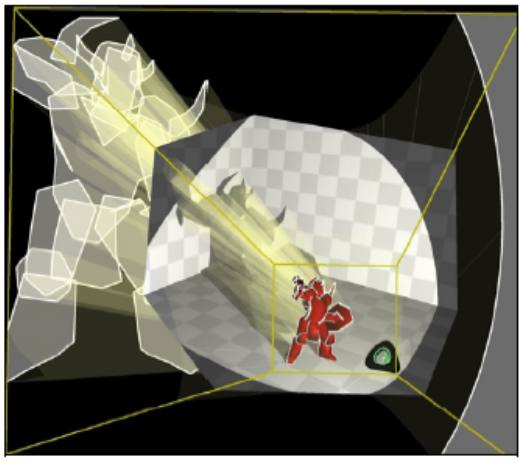
- Comparison test between reference and stencil value
  - GL_NEVER always fails
  - GL_ALWAYS always passes
  - GL_LESS passes if reference value is less than stencil buffer
  - GL_LEQUAL passes if reference value is less than or equal to stencil buffer
  - GL_EQUAL passes if reference value is equal to stencil buffer
  - GL_GEQUAL passes if reference value is greater than or equal to stencil buffer
  - GL_GREATER passes if reference value is greater than stencil buffer
  - GL_NOTEQUAL passes if reference value is not equal to stencil buffer
- **If the stencil test fails**, the fragment is discarded and the stencil operations associated with the stencil test failing is applied to the stencil value
- **If the stencil test passes**, the depth test is applied
  - **If the depth test passes**, the fragment continue through the graphics pipeline, and the stencil operation for stencil and depth test passing is applied
  - **If the depth test fails**, the stencil operation for stencil passing but depth failing is applied

Stencil Operation

- Stencil Operation: Results of Operation on Stencil Values
  - GL_KEEP stencil value unchanged
  - GL_ZERO stencil value set to zero
  - GL_REPLACE stencil value replaced by stencil reference value
  - GL_INCR stencil value incremented
  - GL_DECR stencil value decremented GL_INVERT stencil value bitwise inverted
- Remember you can set different operations for
  - Stencil fails
  - Stencil passes, depth fails
  - Stencil passes, depth passes

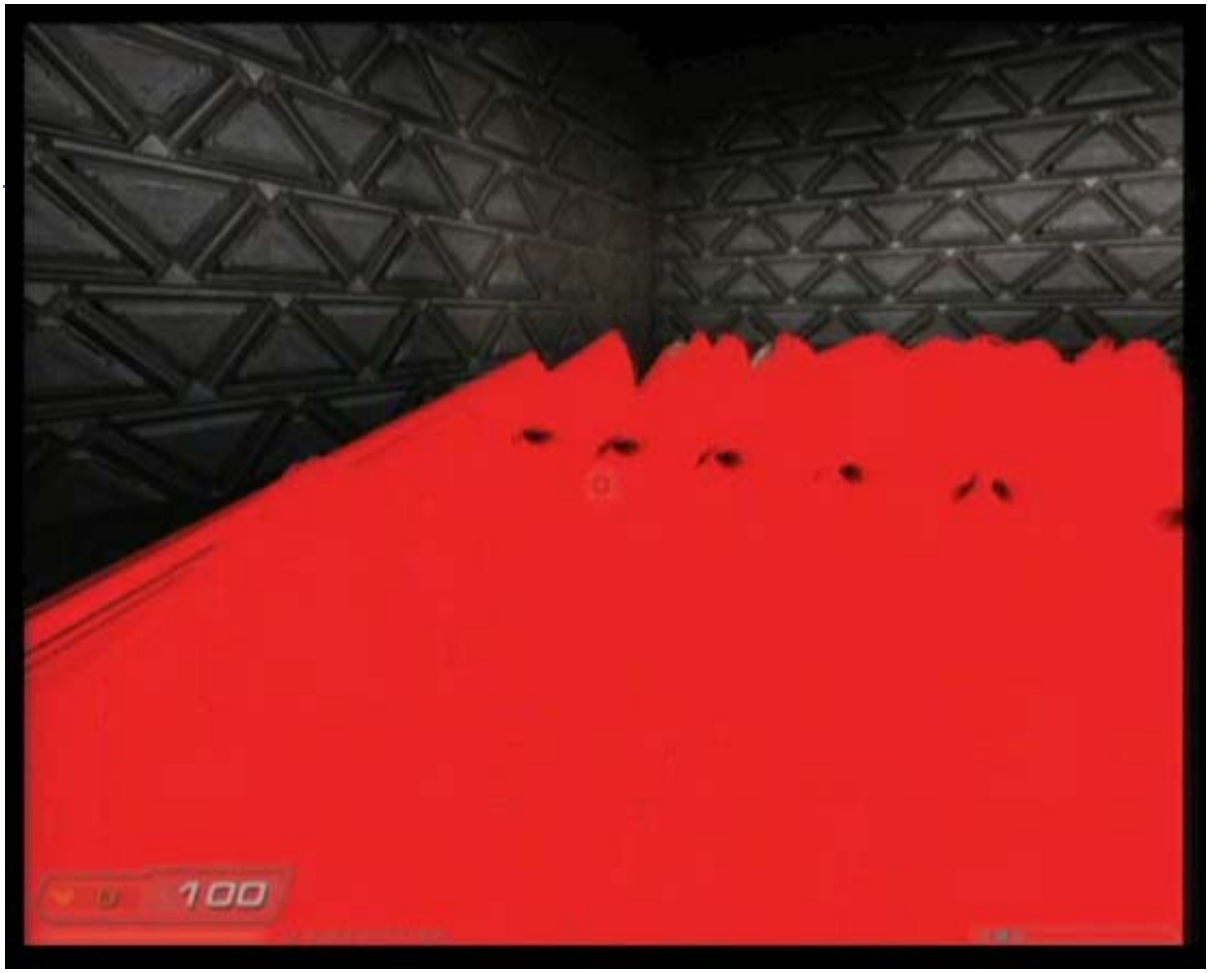


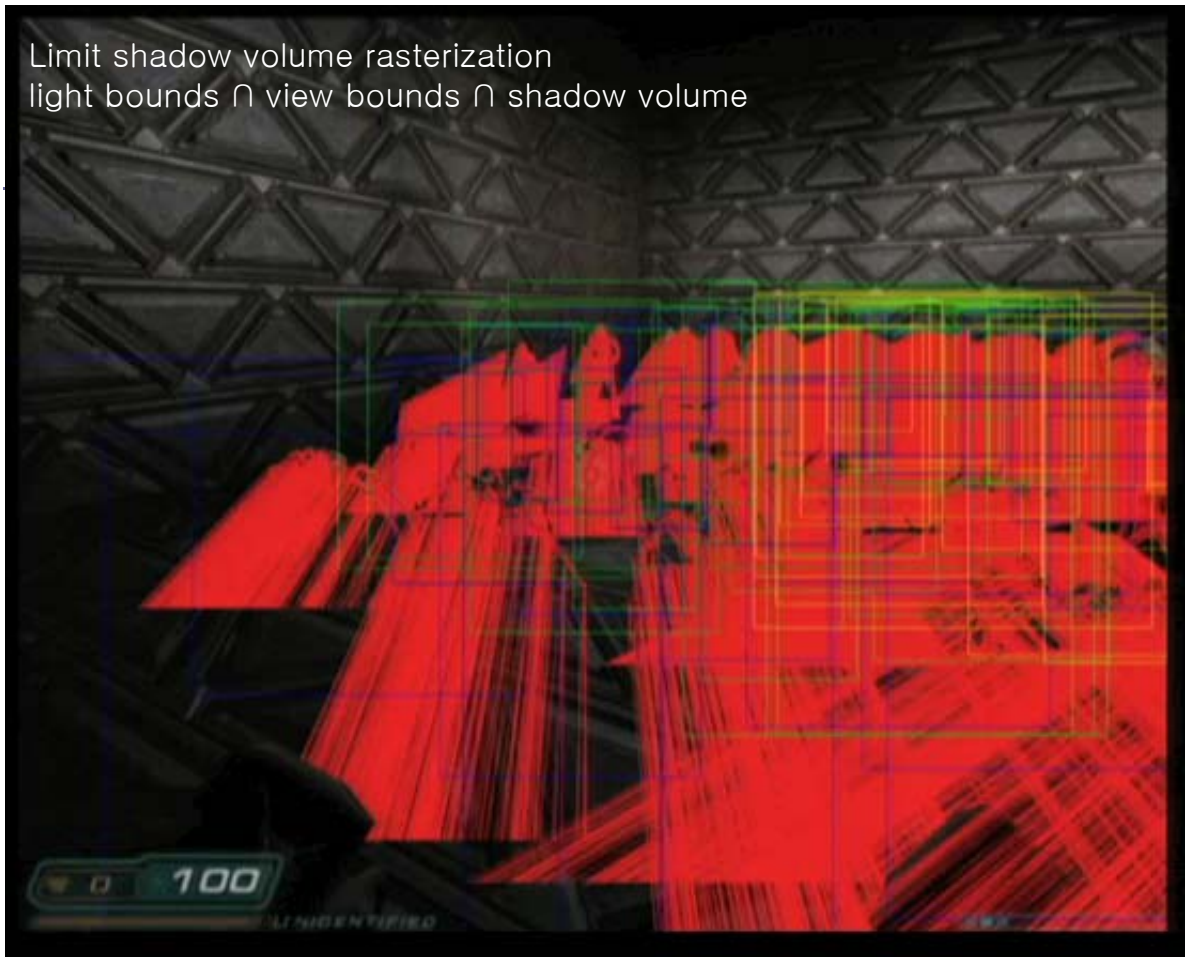
Shadowed scene with the light near the eye and surrounded by a complex surface.



An alternate view of the scene including shadow volumes and silhouette edges with everything outside the eye's infinite frustum clipped away.

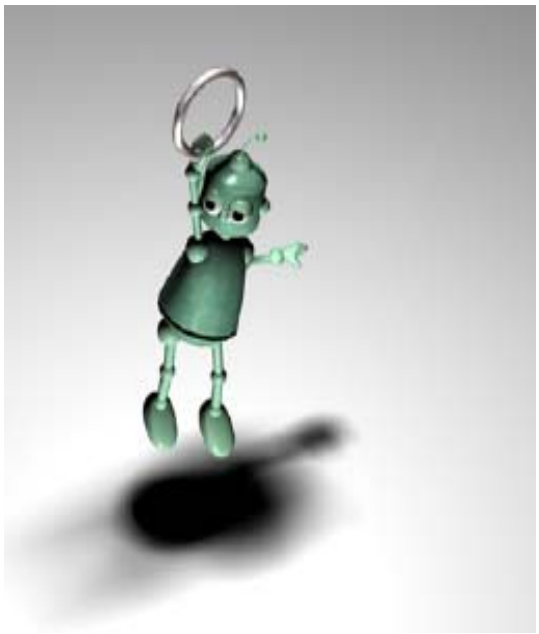

100

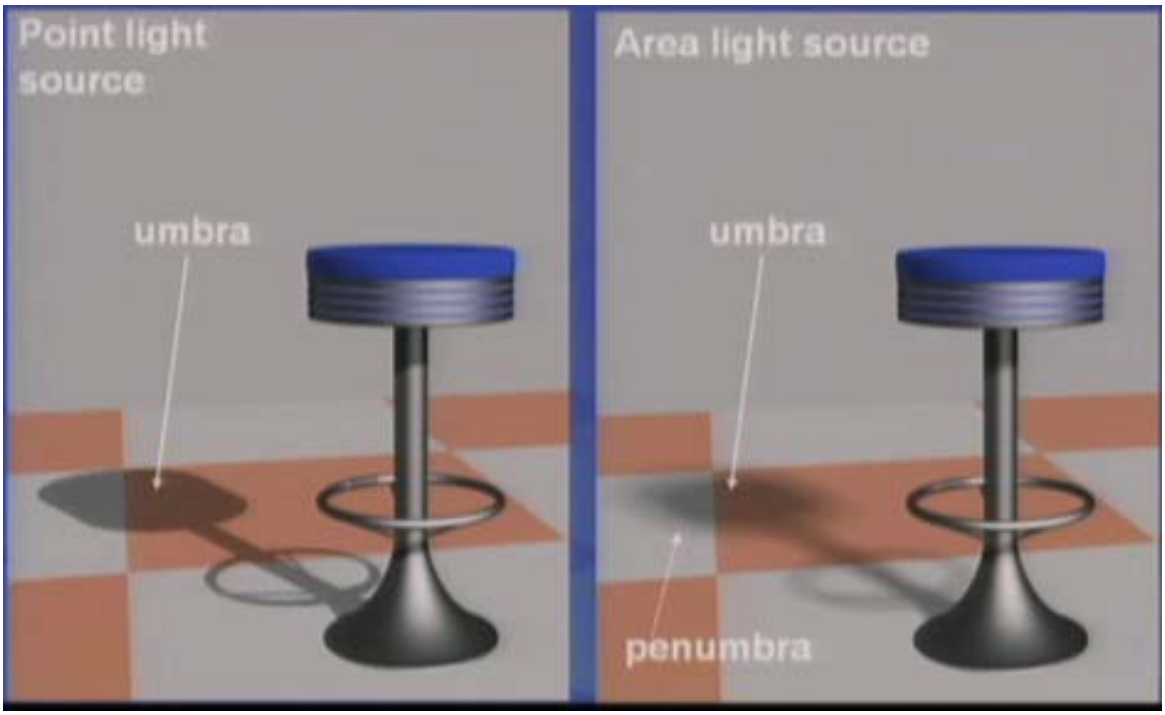

100

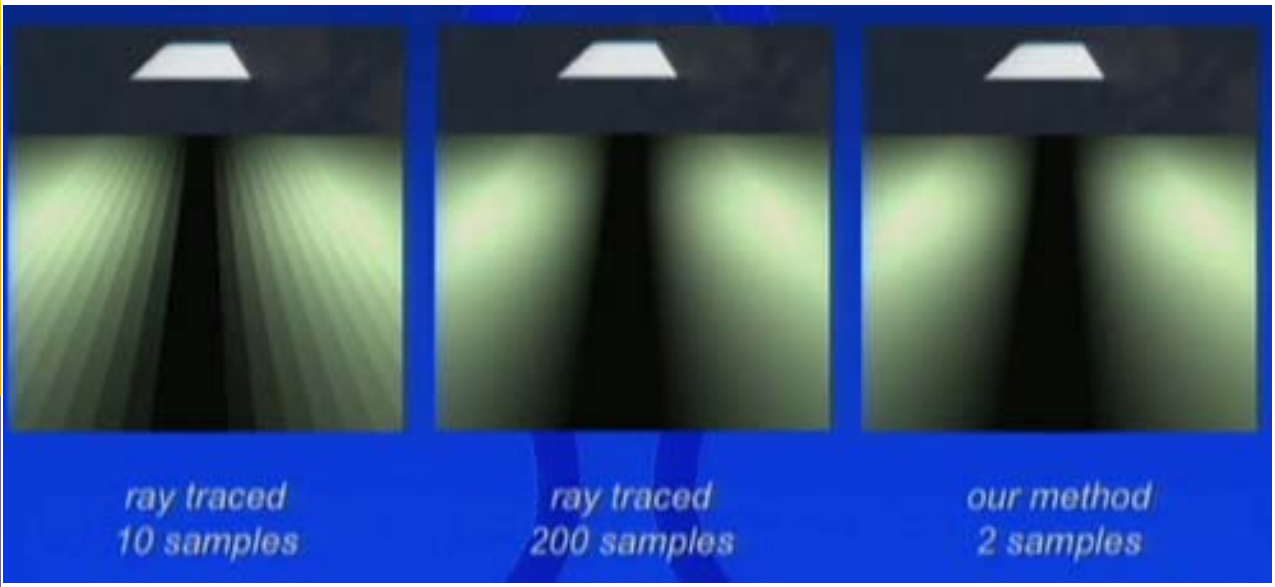


Real time soft shadowing

- Only area light source can generate soft shadows
- We can sample the area light source at different points and then average the results
- Active research area




Point light source
umbra
Area light source
umbra
penumbra


ray traced
10 samples
ray traced
200 samples
our method
2 samples

Siggraph 2009
Silhouettes Of Jazz