# Computer Graphics
## 8. Shadow

I-Chen Lin

Institute of Multimedia Engineering,
National Yang Ming Chiao Tung University

# Intended Learning Outcomes

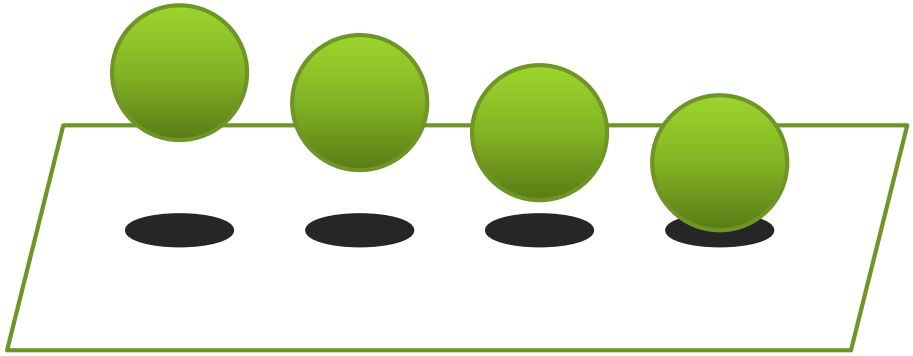
- On completion of this chapter, a student will be able to:
  - Explain how and when the shadows occur.
  - Describe the typical algorithms for real-time shadow rendering.
  - Compare the advantages and limitations of typical shadow algorithms.
  - Apply the shadow mapping method with GLSL.

# Objective

- This chapter focuses on shadows in real-time graphics.
  - Global illumination methods can generate realistic shadows but require heavy computation.
  - Local illumination does not consider the situation where the light can be blocked by other objects.

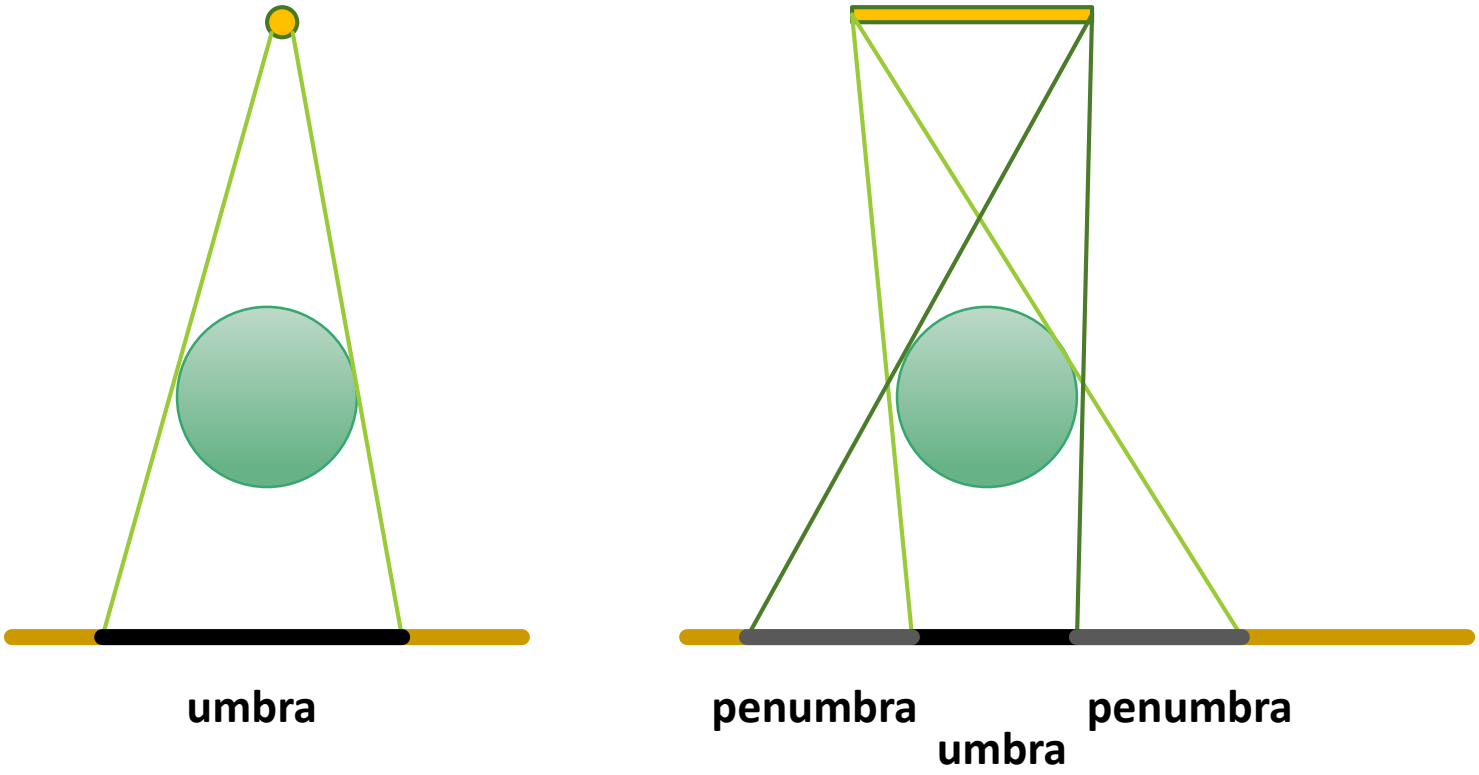
# Why are shadows important?

- Depth cue
- Contact point

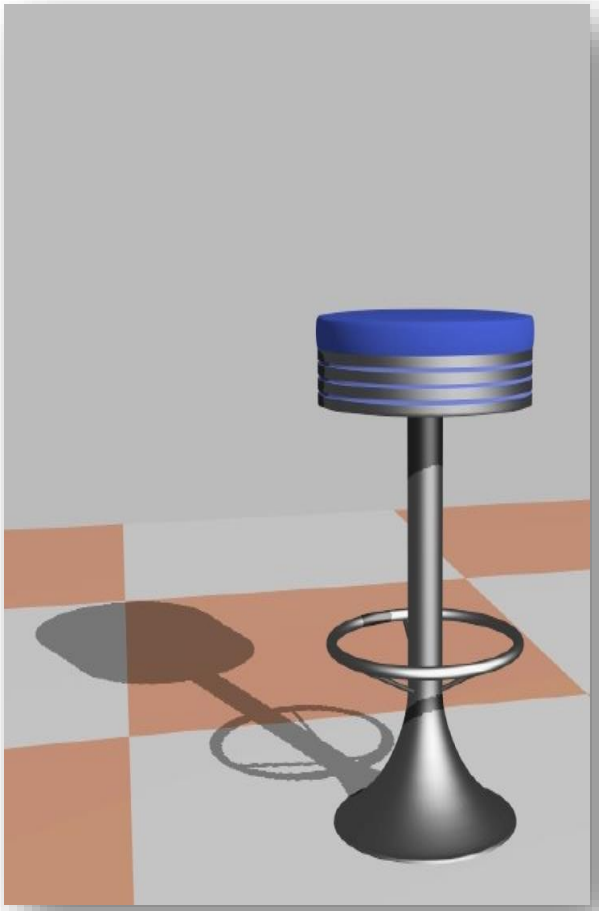


- Realism of illuminated scenes

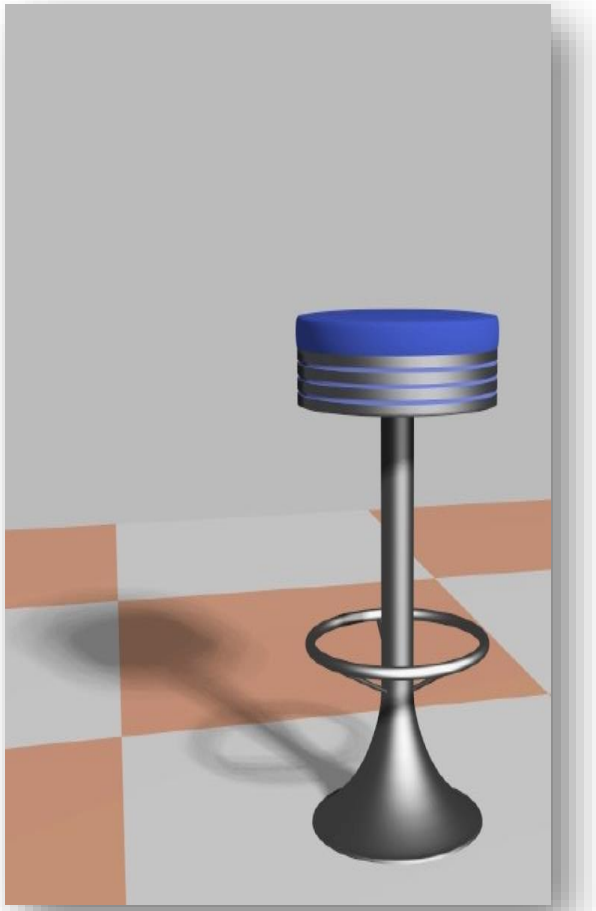
# Why do shadows occur?



# Umbra and penumbra



Point light source



Area light source

Figures from T. Akenine-Möller, Lecture note of “Advanced Shading and Rendering”, Lund Univ.

# Real-time shadow techniques

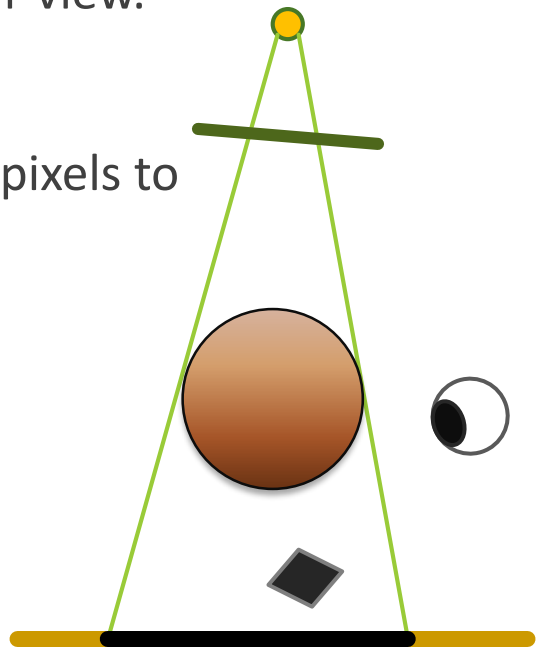
- Projected planar shadows
  - works only on flat surfaces
- Light maps or texture shadows
  - unsuited for dynamic shadows
- Popular methods
  - Shadow map
  - Shadow volume

# Shadow mapping

- Image-space shadow
  - L. Williams published the idea in 1978.
- Completely image-space algorithm
  - No knowledge of scene's geometry is required
  - Aliasing artifacts may occur
- Nearly "standard" for shadow rendering
  - Pixar's RenderMan also include the algorithm
  - Applied in several Pixar's movies, e.g. Luxo Jr., Toy Story, etc.

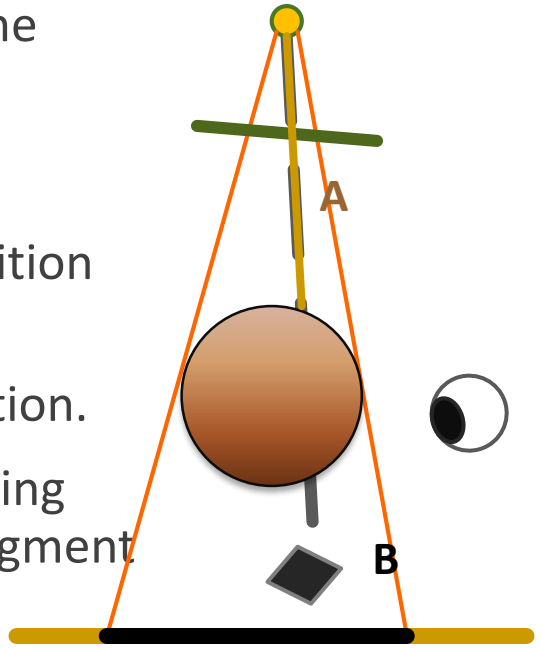
# The concept of shadow mapping

- Depth testing from the light's point-of-view
  - Two-pass algorithm
  - First, render depth buffer from the light's point-of-view.
  - The result is a "shadow map".
  - A 2D function indicating the depth of the closest pixels to the light
- This depth map is used in the second pass.



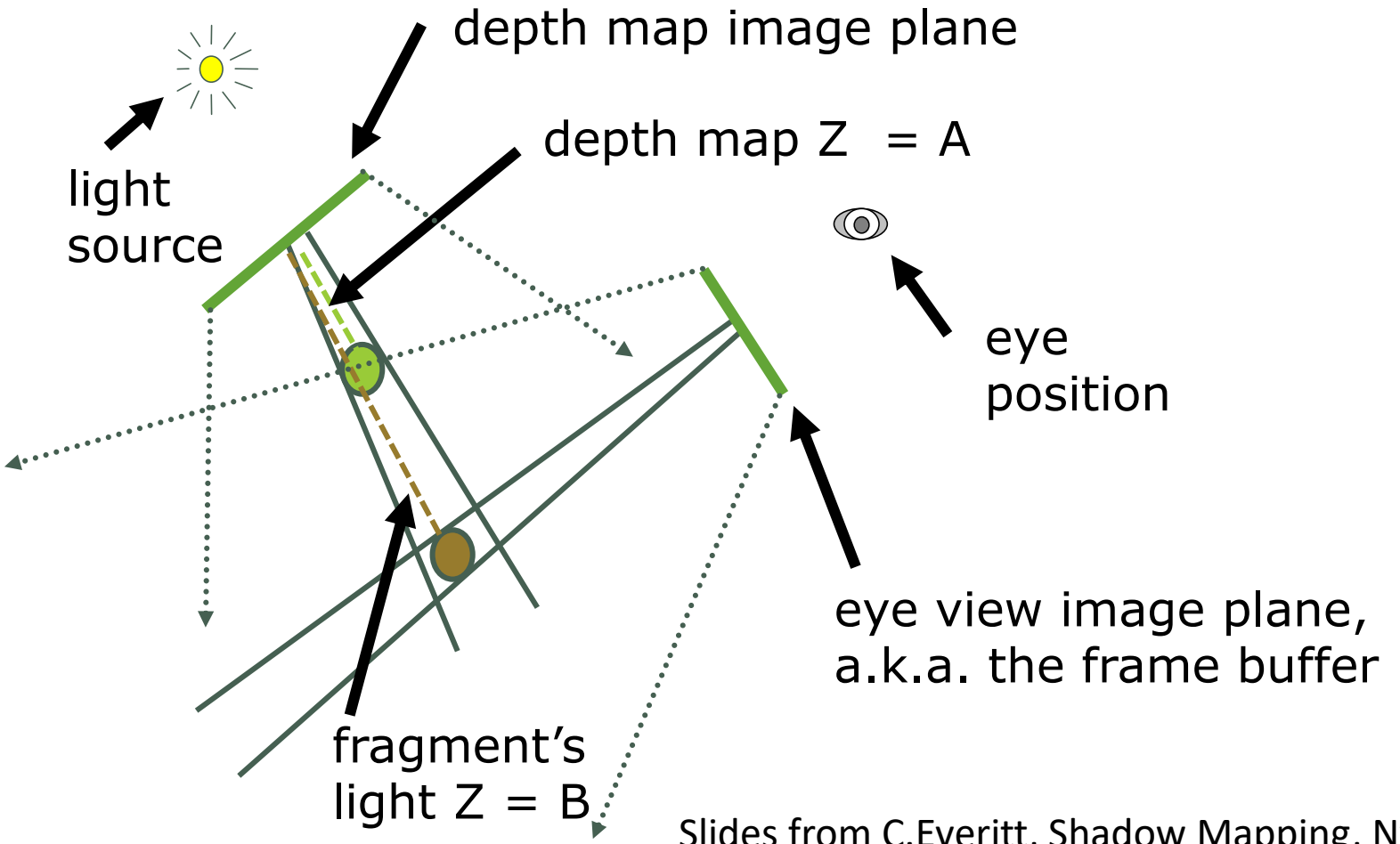
# The concept of shadow mapping (cont.)

- Second, render scene from the eye's point-of-view
- For each rasterized fragment
  - determine fragment's XYZ position *relative to the light*.
  - compare the depth value at light position XY in the depth map to fragment's light position Z.
- Comparing two values:
  - A = Z value from depth map at fragment's XY position from the view of light.
  - B = Z value of fragment with respect to light position.
  - If B is greater than A, then there must be something closer to the light than the fragment then the fragment is shadowed.
  - If A and B are approximately equal, the fragment is lit.



# Shadow mapping with a picture in 2D (1)

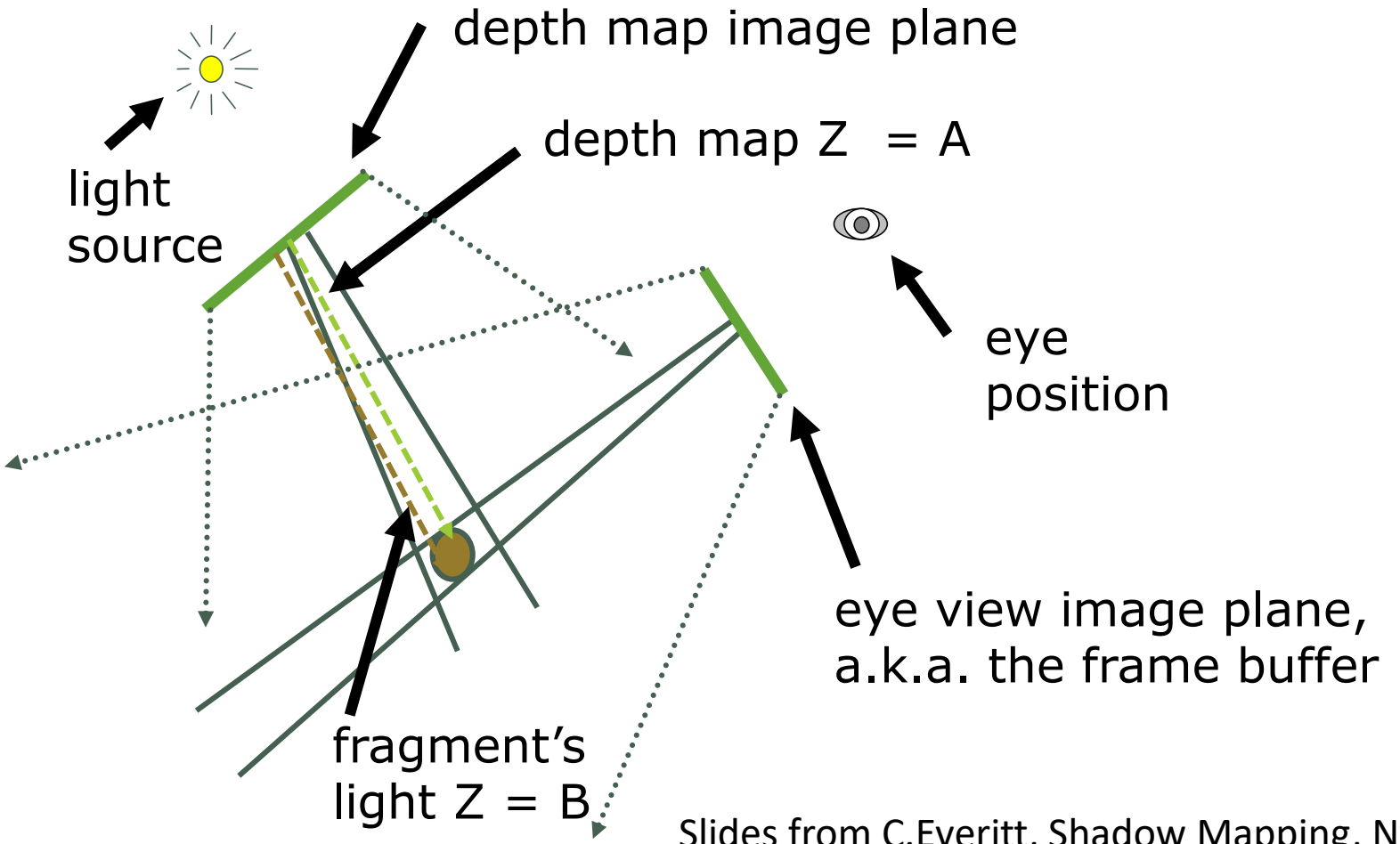
The A < B shadowed fragment case




Slides from C.Everitt, Shadow Mapping, Nvidia

# Shadow mapping with a picture in 2D (2)

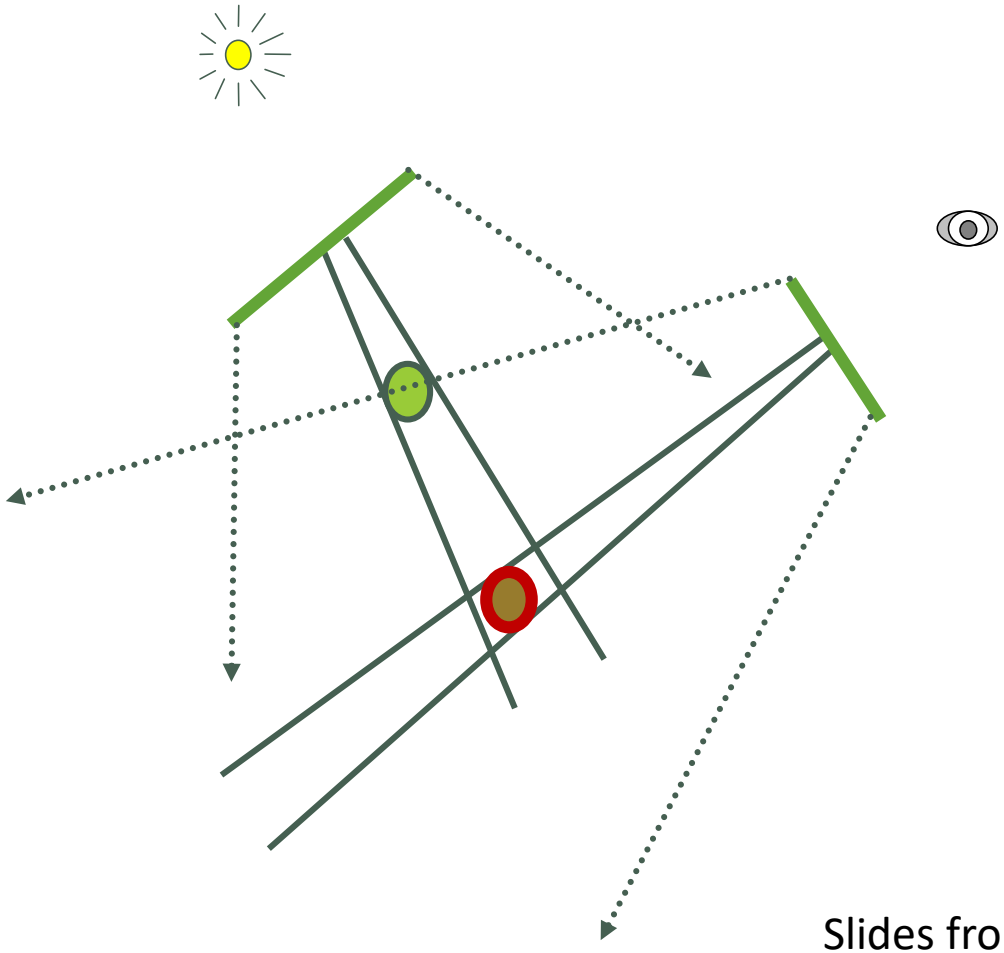
The A ≅ B unshadowed fragment case




Slides from C.Everitt, Shadow Mapping, Nvidia

# Shadow mapping with a picture in 2D (3)

Note image precision mismatch!



The depth map could be at a different resolution from the framebuffer

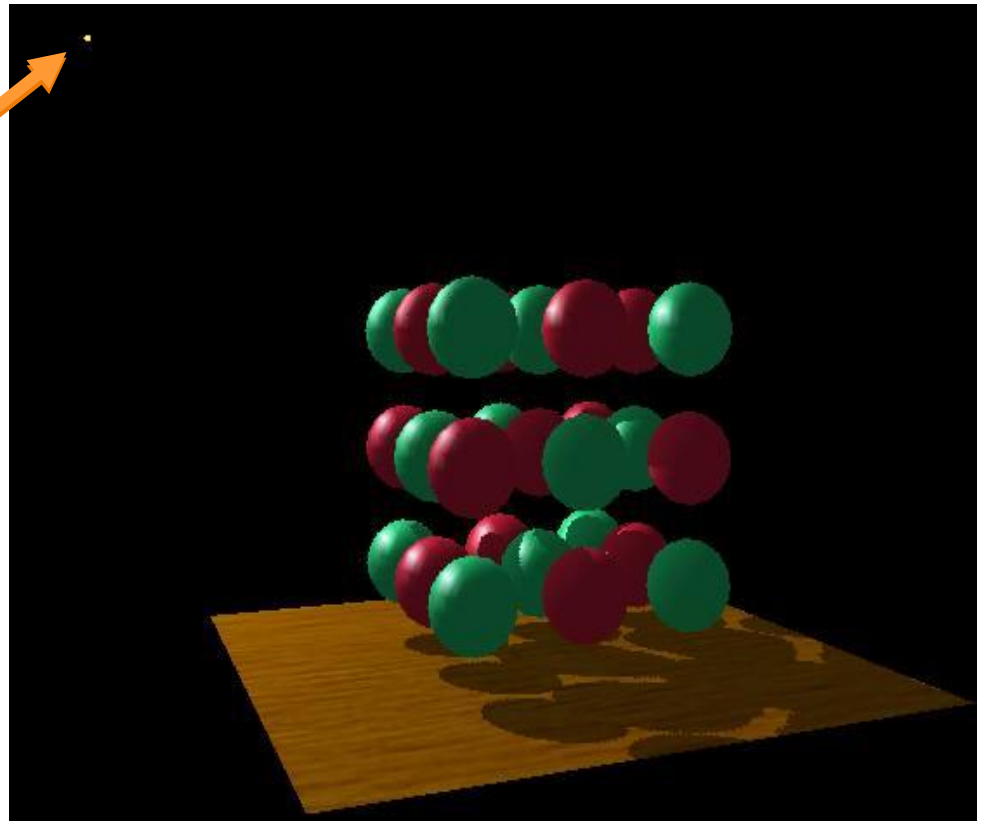
This mismatch can lead to artifacts

Slides from C.Everitt, Shadow Mapping, Nvidia

# Visualizing the shadow mapping (1)

- A fairly complex scene with shadows

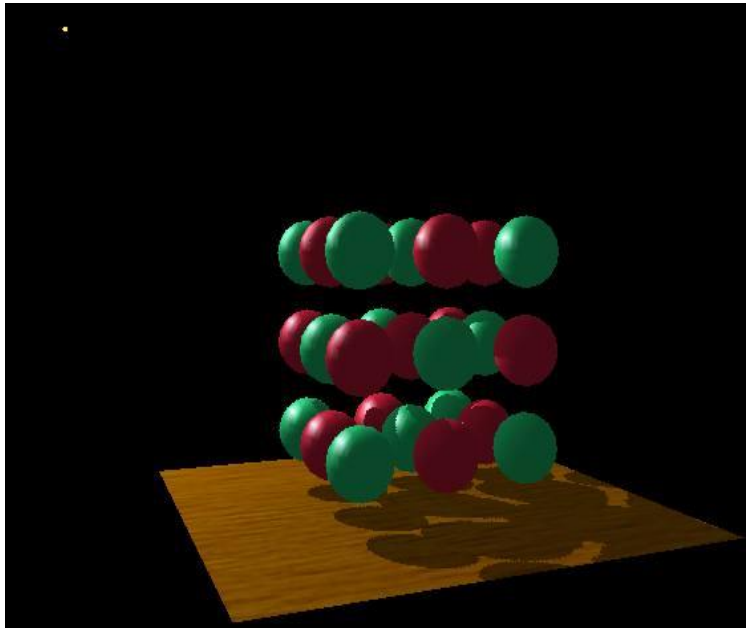
the point light source



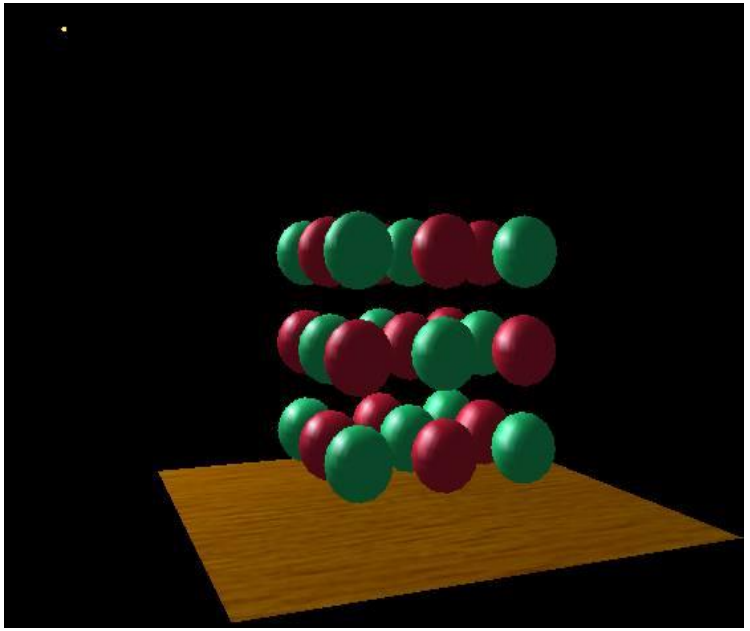
Slides from C.Everitt, Shadow Mapping, Nvidia

# Visualizing the shadow mapping (2)

- Compare with and without shadows



with shadows

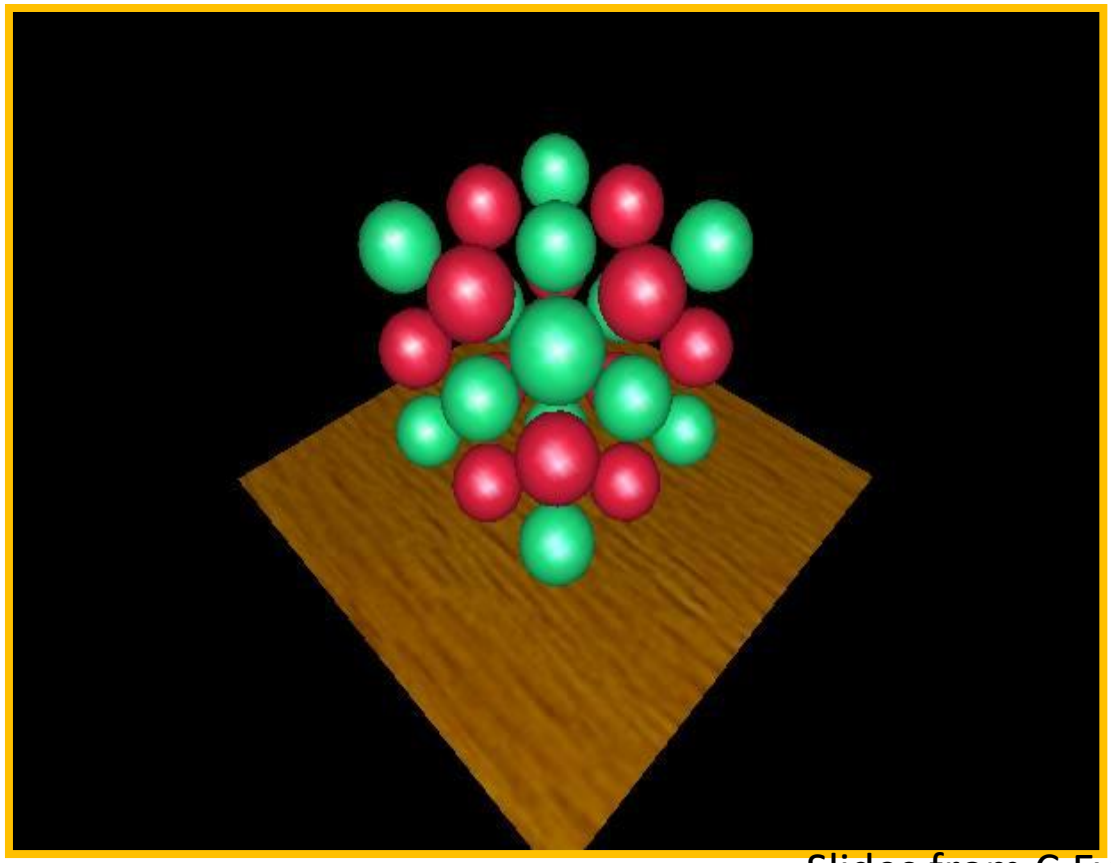


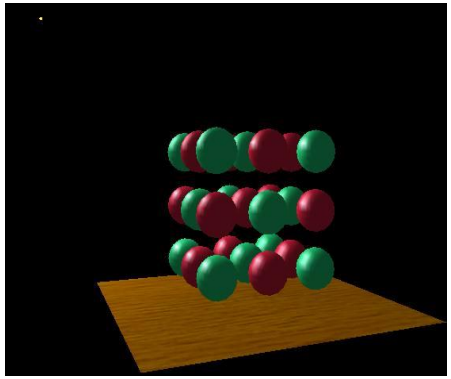
without shadows

Slides from C.Everitt, Shadow Mapping, Nvidia

# Visualizing the shadow mapping (3)

- The scene from the light's point-of-view



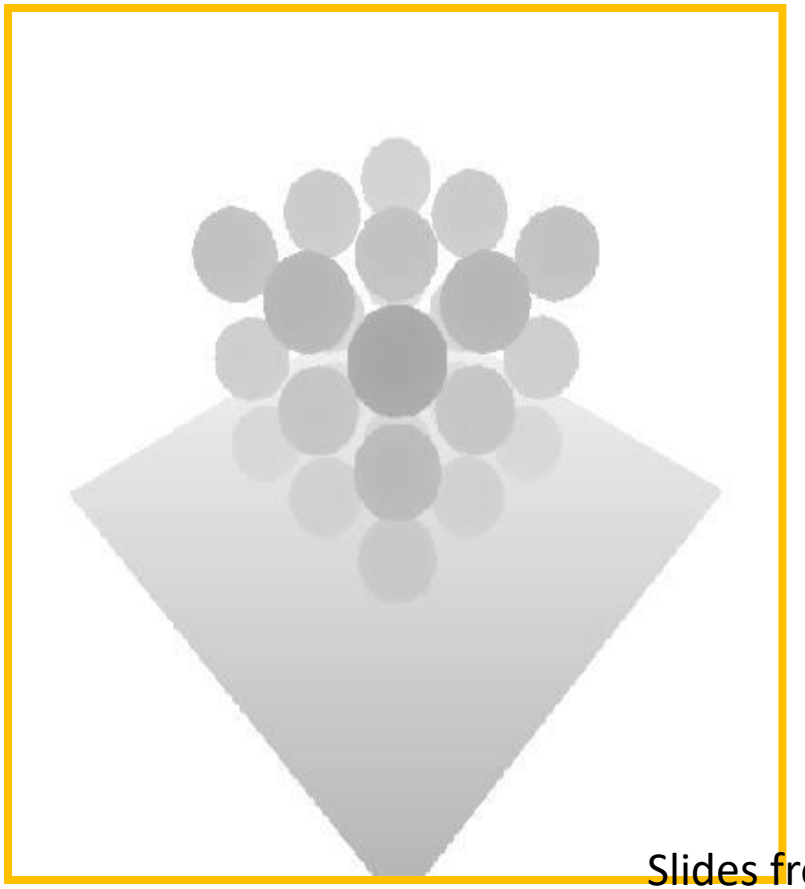


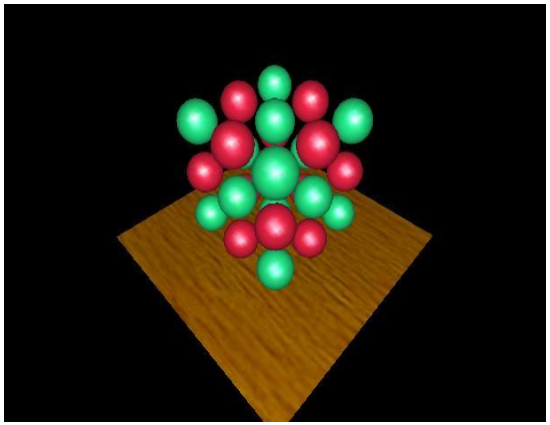
FYI: from the eye's point-of-view again

Slides from C.Everitt, Shadow Mapping, Nvidia

# Visualizing the shadow mapping (4)

- The depth buffer from the light's point-of-view



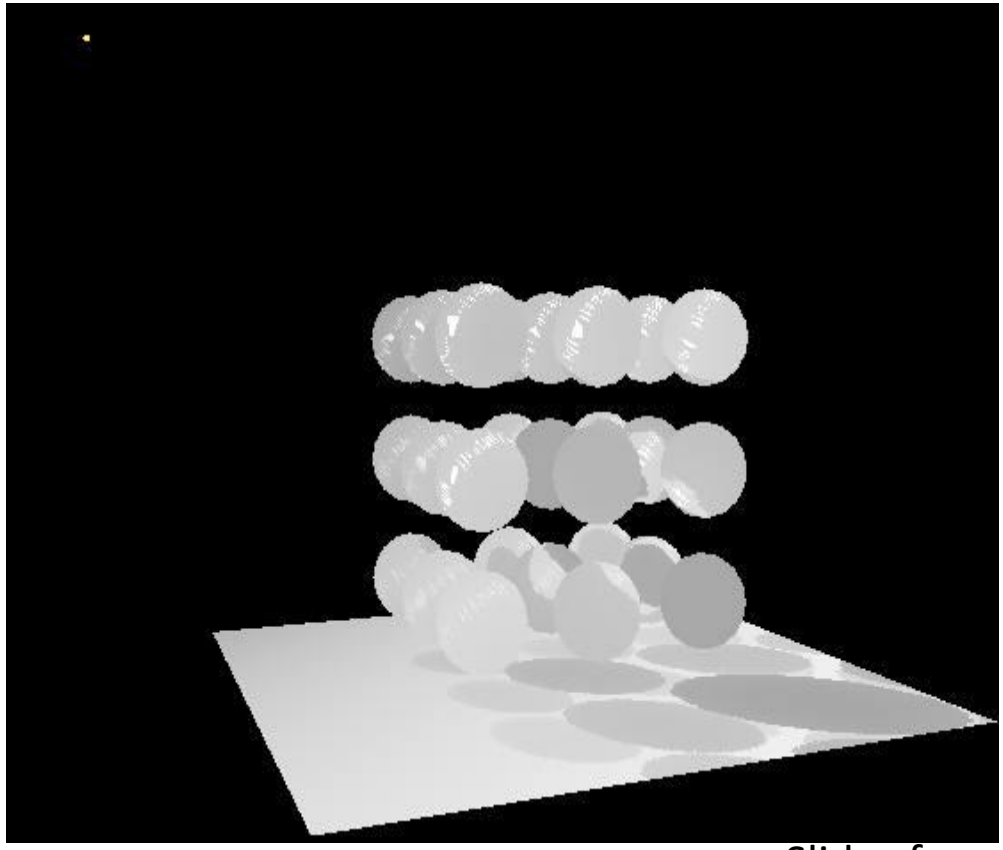


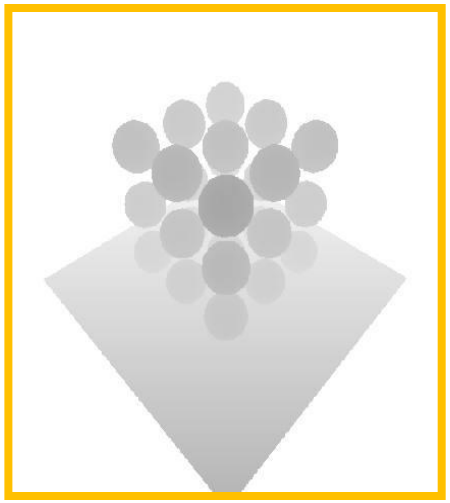
FYI: from the light's point-of-view again

Slides from C.Everitt, Shadow Mapping, Nvidia

# Visualizing the shadow mapping (5)

- Projecting the depth map onto the eye's view



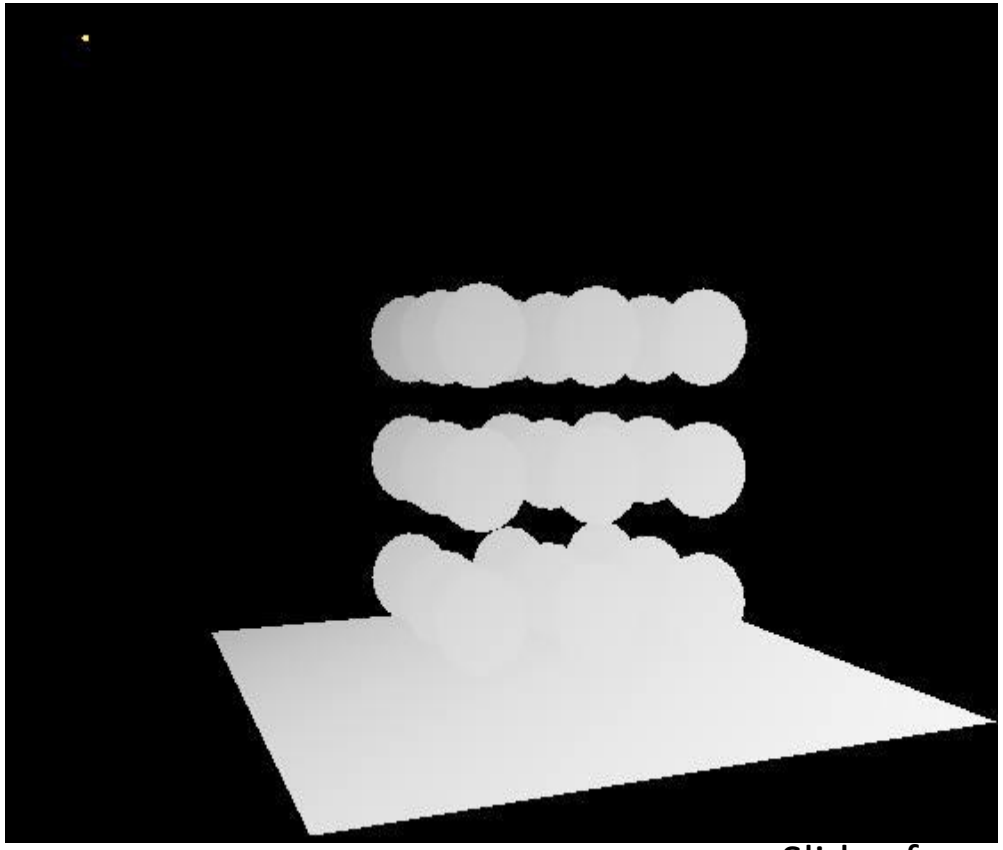


FYI: depth map for light's point-of-view again

Slides from C.Everitt, Shadow Mapping, Nvidia

# Visualizing the shadow mapping (6)

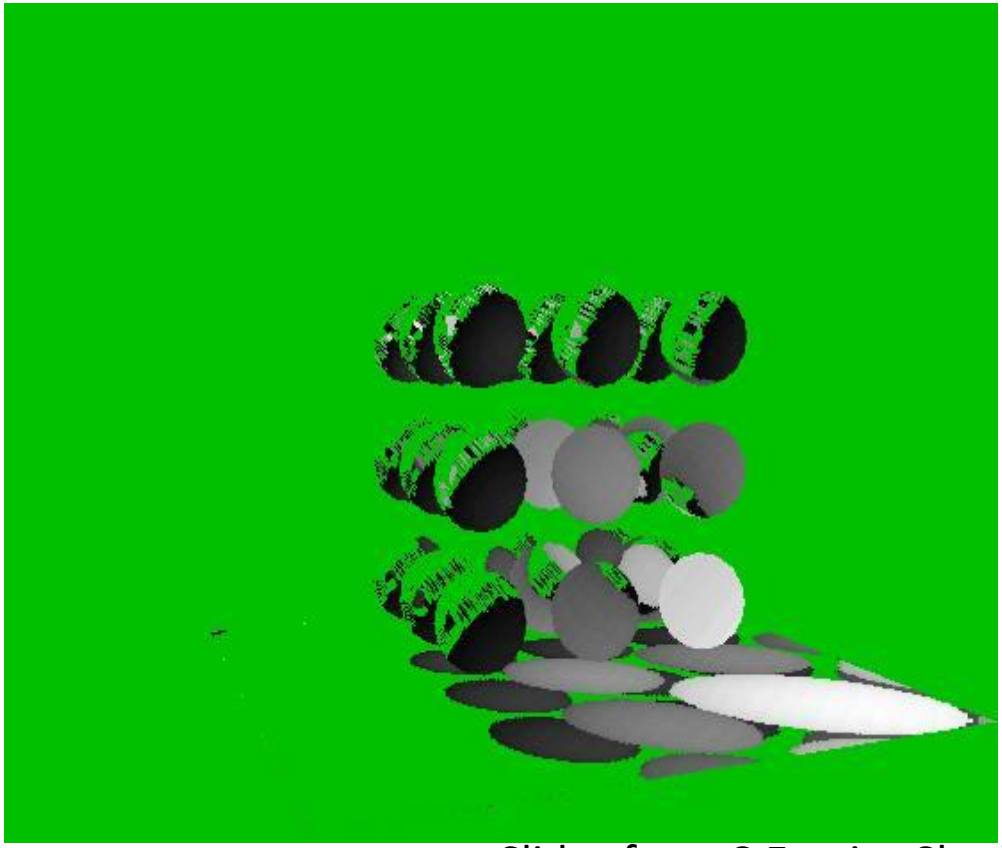
- Projecting light-to-fragment distance onto eye's view



Slides from C.Everitt, Shadow Mapping, Nvidia

# Visualizing the shadow mapping (6)

- Comparing light distance to light depth map



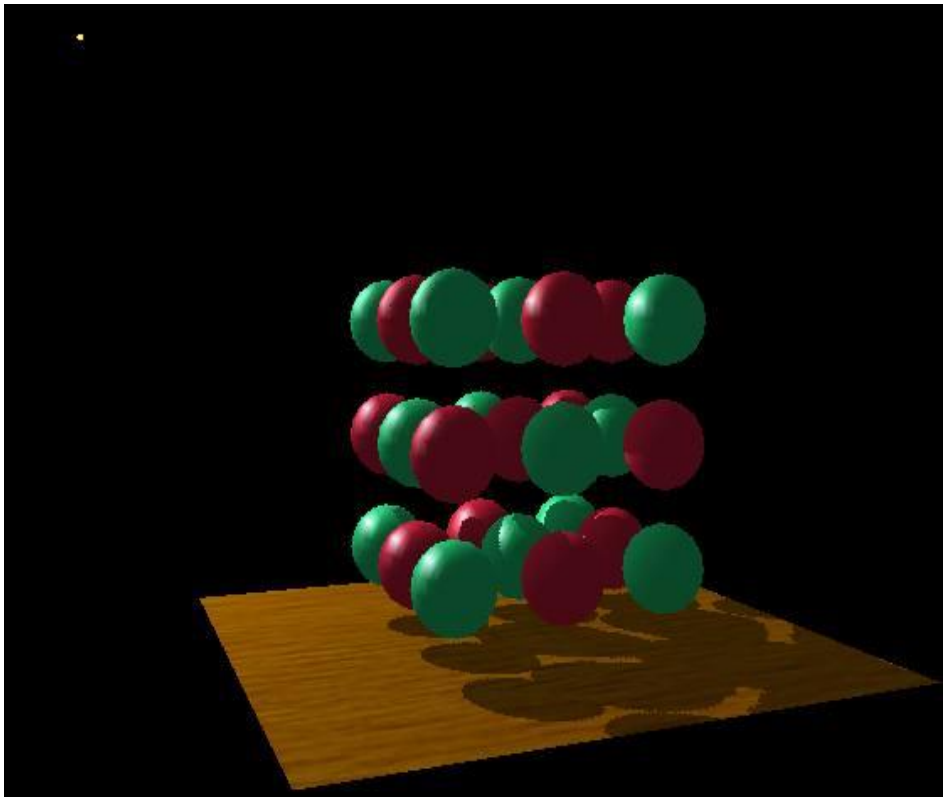
Non-green is where shadows should be

Slides from C.Everitt, Shadow Mapping, Nvidia

# Visualizing the shadow mapping (7)

- Scene with shadows

*Notice how specular highlights never appear in shadows*

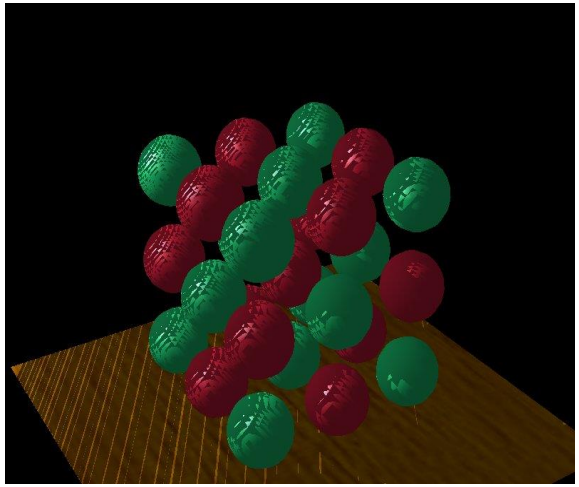


*Notice how curved surfaces cast shadows on each other*

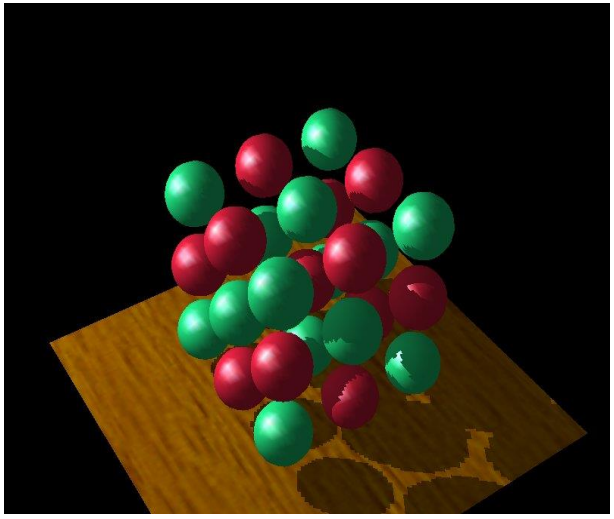
Slides from C.Everitt, Shadow Mapping, Nvidia

# Shadow mapping problem (1)

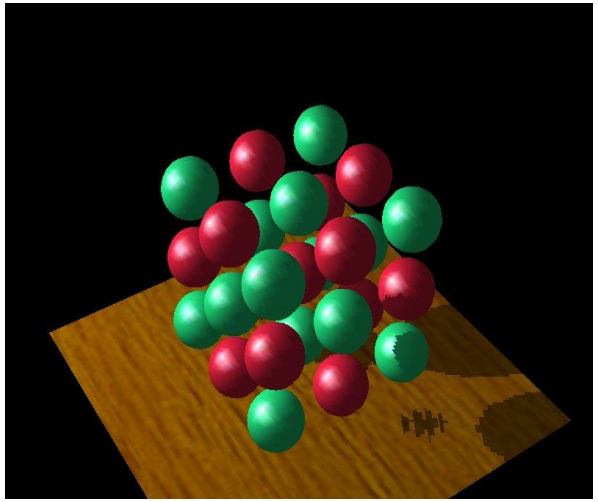
- Accuracy of depth values (quantization and sampling)
  - $D_{sm}$ and $D_{eye}$ may have different values even if they represent the same surface.
- Solution: $D_{sm}$ + bias < $D_{eye}$ → shadow



Too little bias, everything begins to shadow



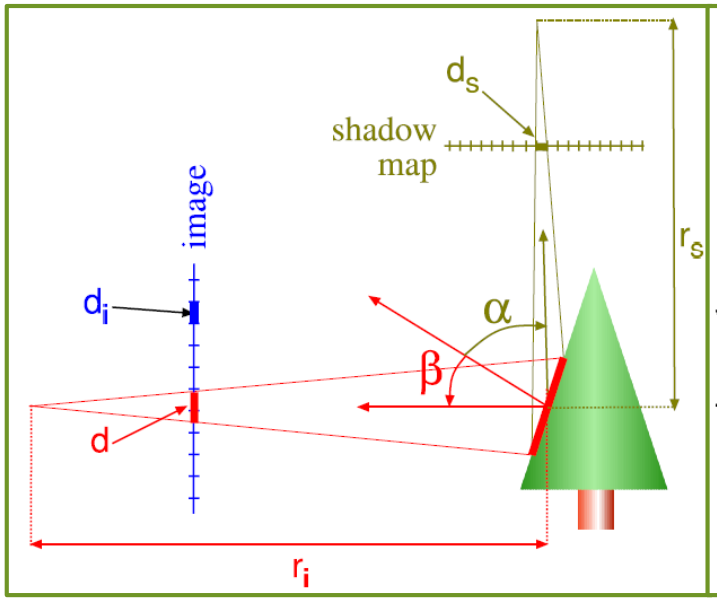
Just right

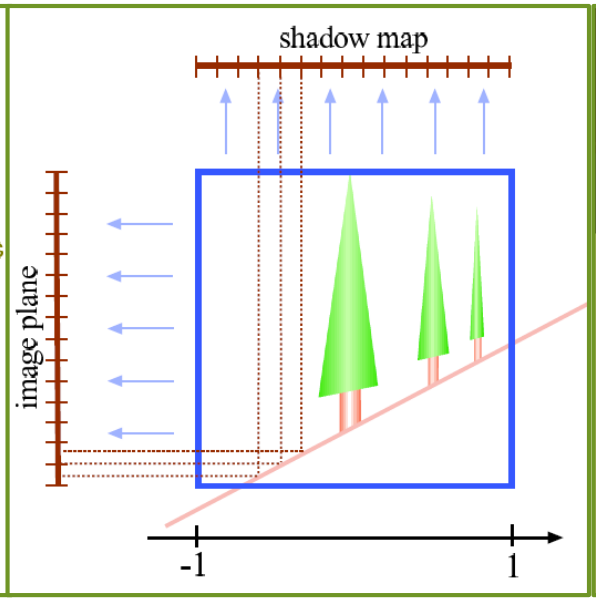


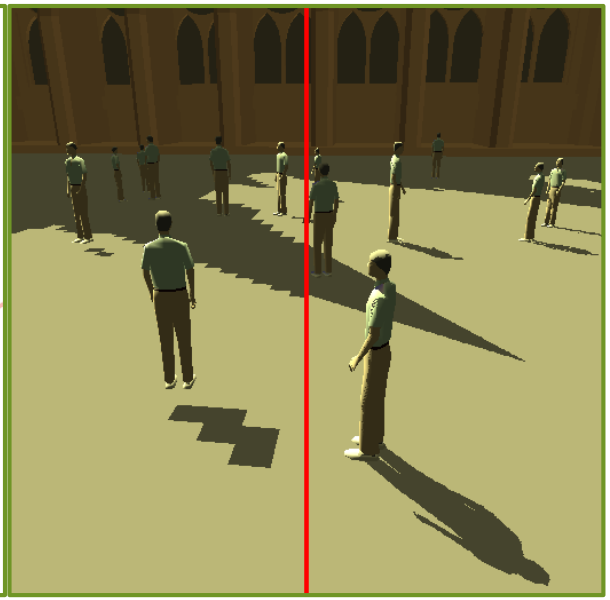
Too much bias, shadow starts too far back

# Shadow mapping problem (2)

- Insufficient resolution of a shadow map
  - Gives jagged shadow edges
  - Resolution needed depends on camera position
- Lots of research on improving quality



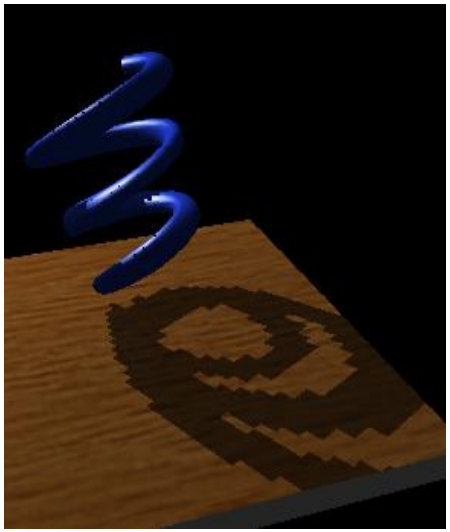



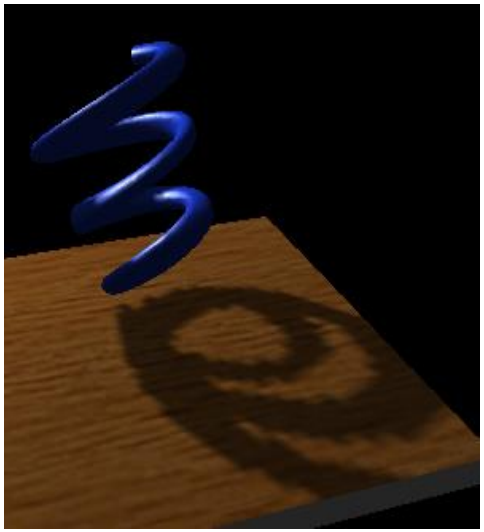



M. Stamminger, “Perspective Shadow Map”, SIGGRAPH’02.

# “Percentage Closer” filtering

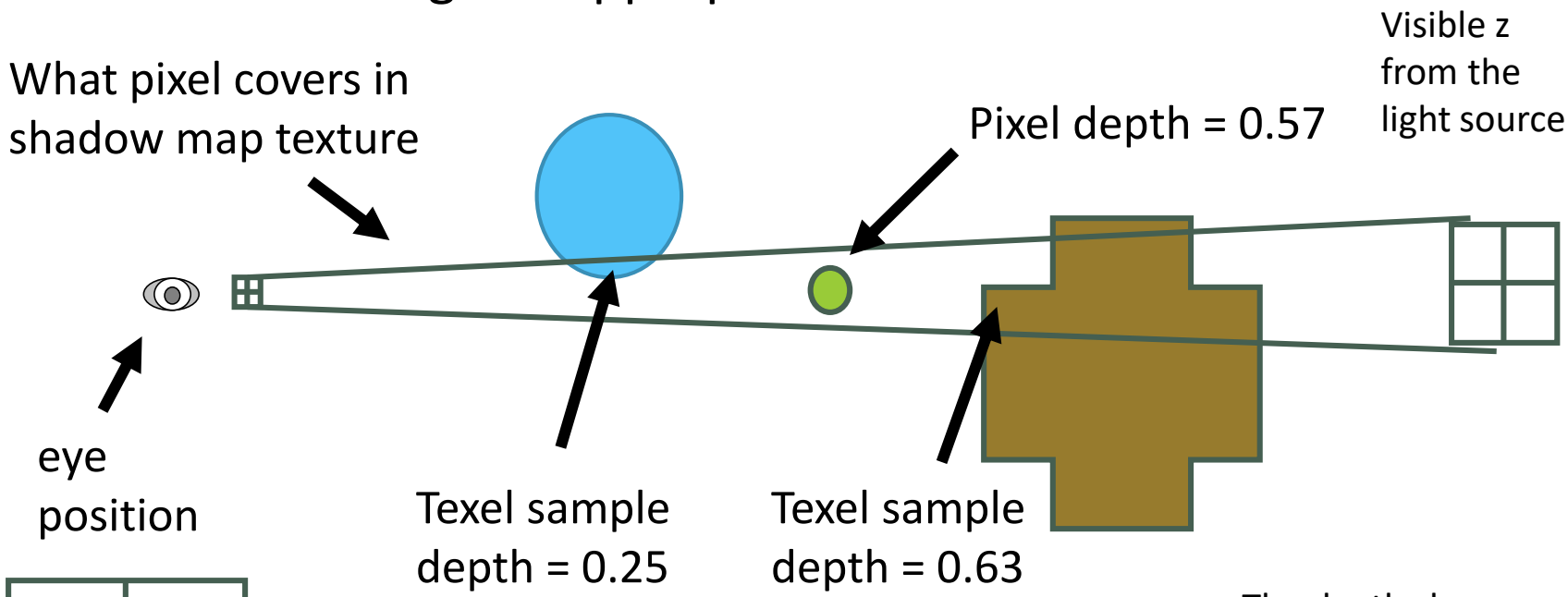
- Normal texture filtering just averages color components.
- Averaging depth values does NOT work.
- Provides anti-aliasing at shadow map edges.
  - Not the real soft shadows in the umbra/penumbra sense.
- Reeves et al. (pixar), Rendering antialiased shadows with depth maps, SIGGRAPH’87.





# Depth values are not blendable

- Traditional filtering is inappropriate




| | |
|---|---|
| 0.25 | 0.25 |
| 0.63 | 0.63 |

Average(0.25, 0.25, 0.63, 0.63) = 0.44

0.57 > 0.44 so pixel is wrongly “in shadow”

Truth: nothing is at 0.44, just 0.25 and 0.57

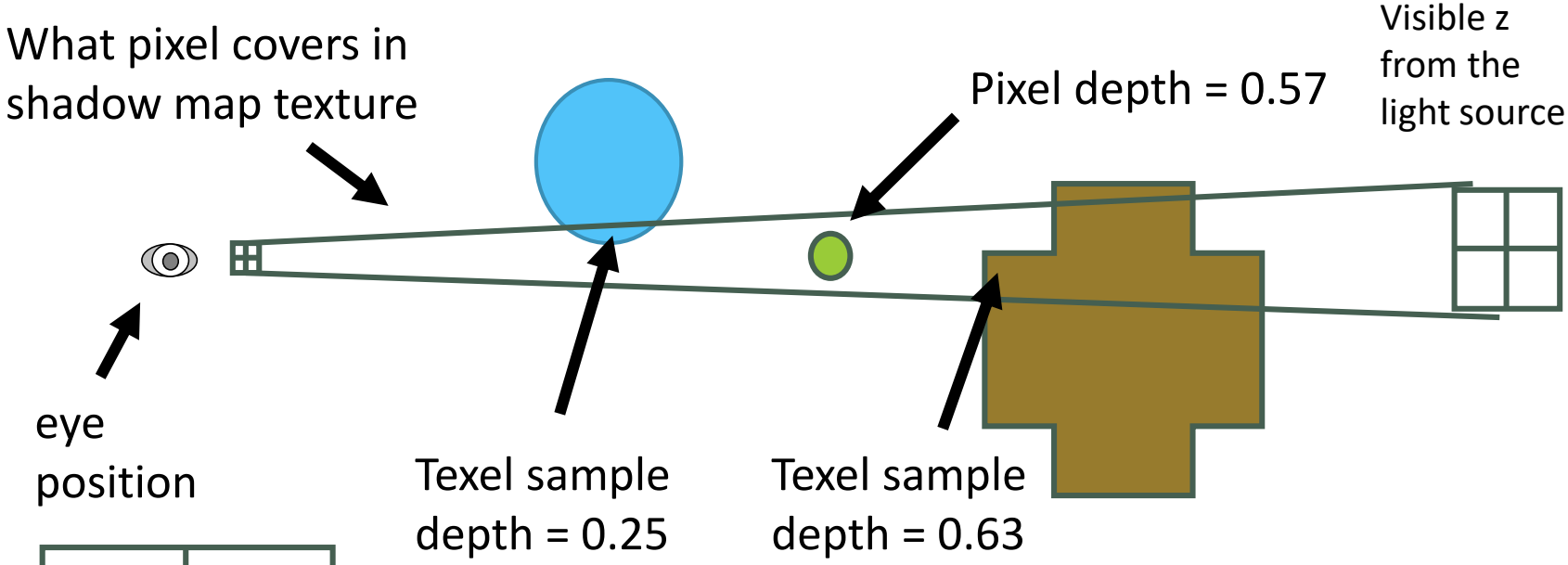
The depths here are the distance from the light source

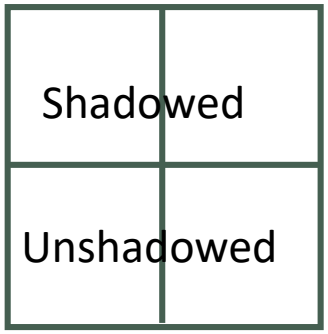
Slides from C.Everitt, Shadow Mapping, Nvidia

# Percentage Closer Filtering

*Note: Alternative ways: Sampling surrounding texels of the depth map of light.*

- Average comparison *results*, not depth values




Average(0.57>0.25, 0.57>0.25, 0.57<0.63, 0.57<0.63) = 50%
so pixel is reasonably 50% shadowed

Slides from C.Everitt, Shadow Mapping, Nvidia

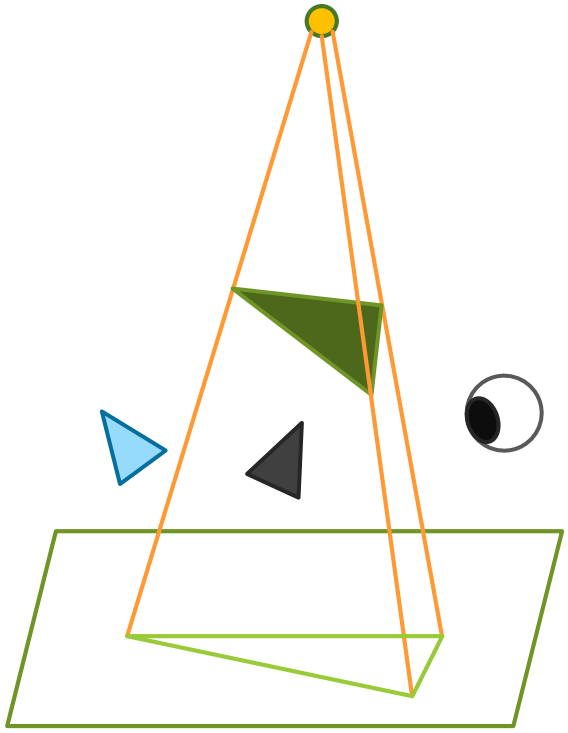
# Percentage Closer Filtering

- Using a bigger filter produces fake soft shadows.

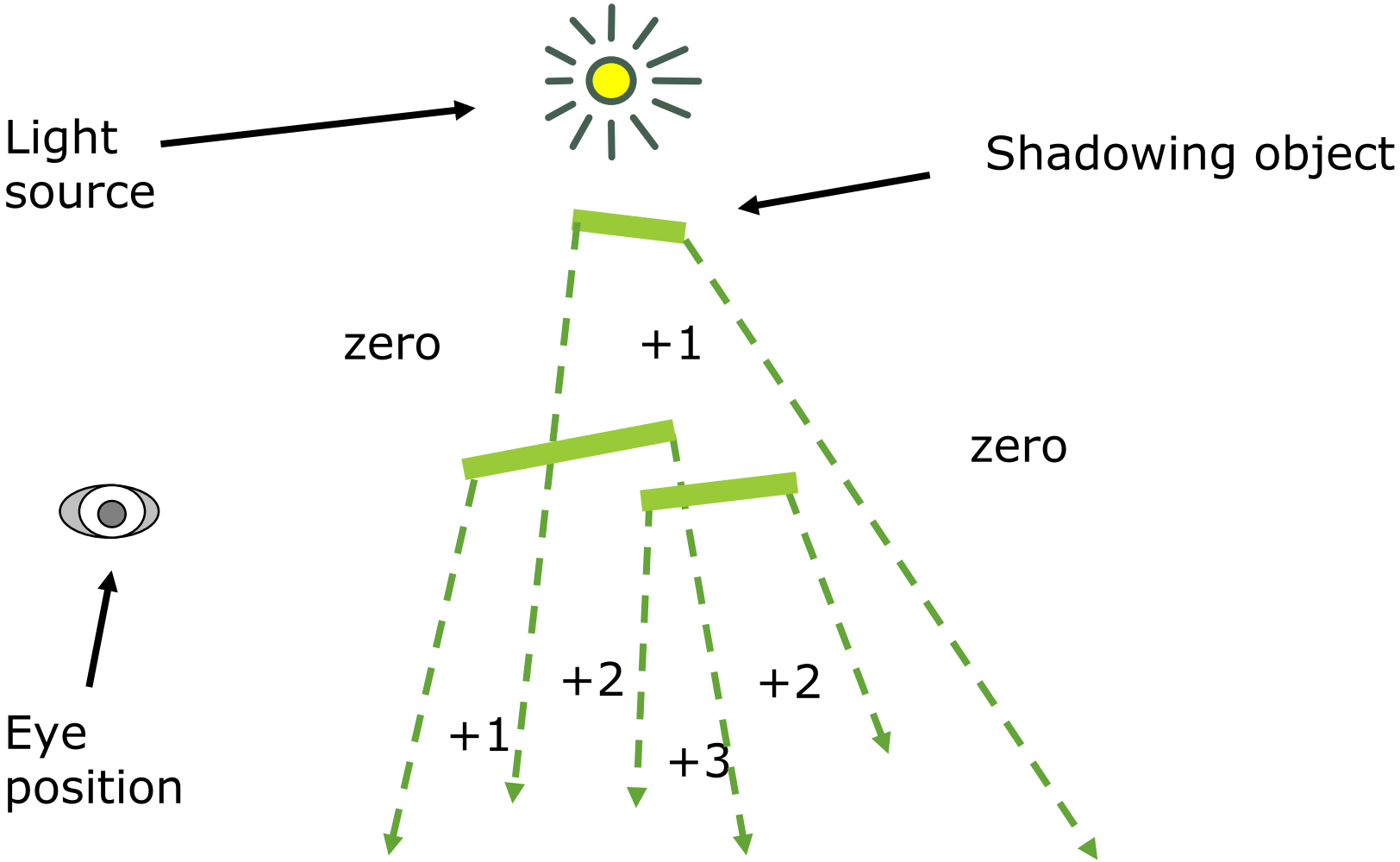
Figures from T. Akenine-Möller, Lecture note of “Advanced Shading and Rendering”, Lund Univ.

# Shadow volume

- Instead of image-space shadow mapping, shadow volume takes scene geometry into account.
- Check whether a polygon is within the shadow “volume”.
- Counts the number of intersections of “shadow volume polygons”.

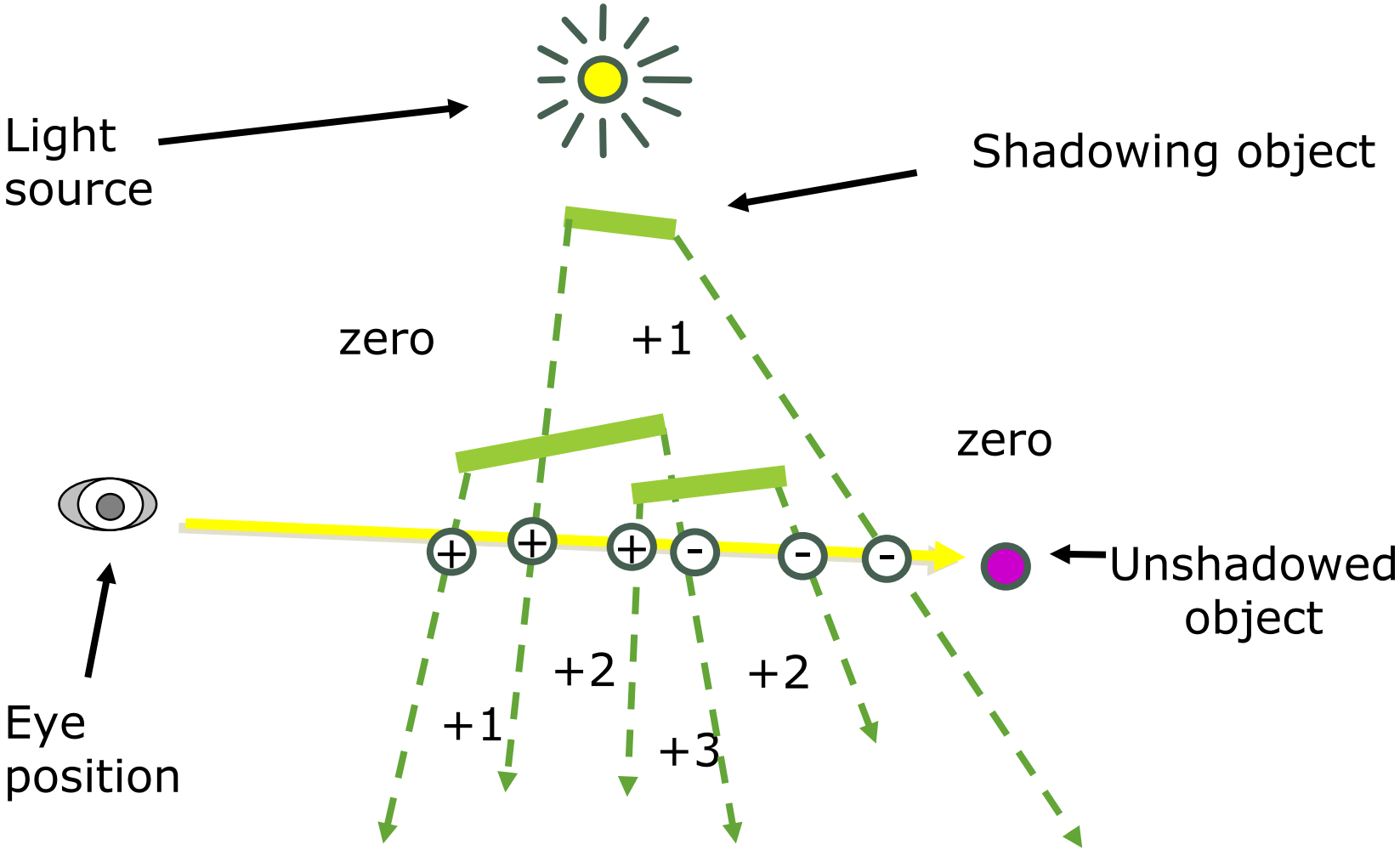


# Stencil Enter/Leave Counting Approach




From C.Everitt and M.J.Kilgard, Practical & Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering, Nvidia

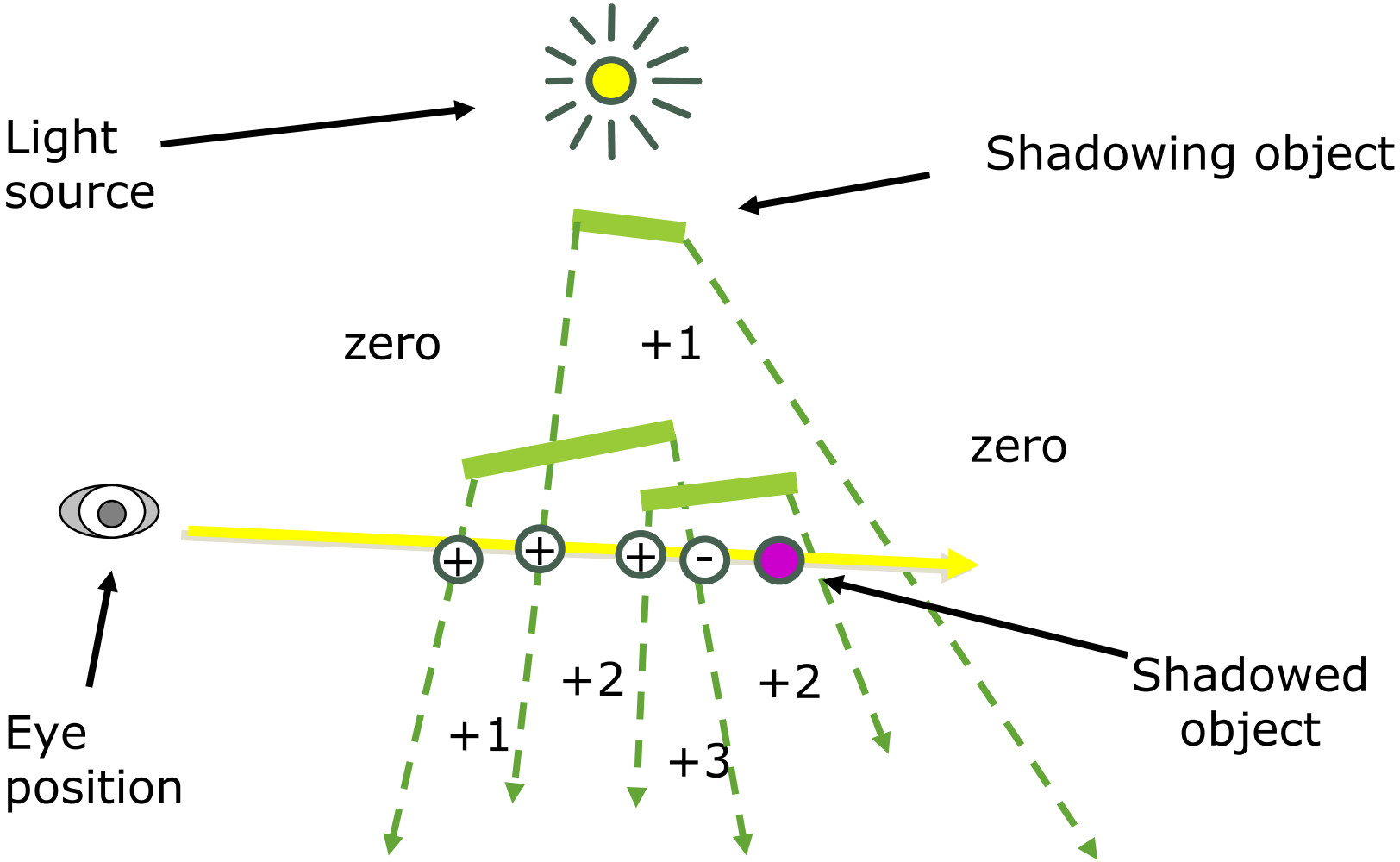
# Stencil Enter/Leave Counting Approach




**Shadow Volume Count = +1+1+1-1-1-1 = 0**

From C.Everitt and M.J.Kilgard, Practical & Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering, Nvidia

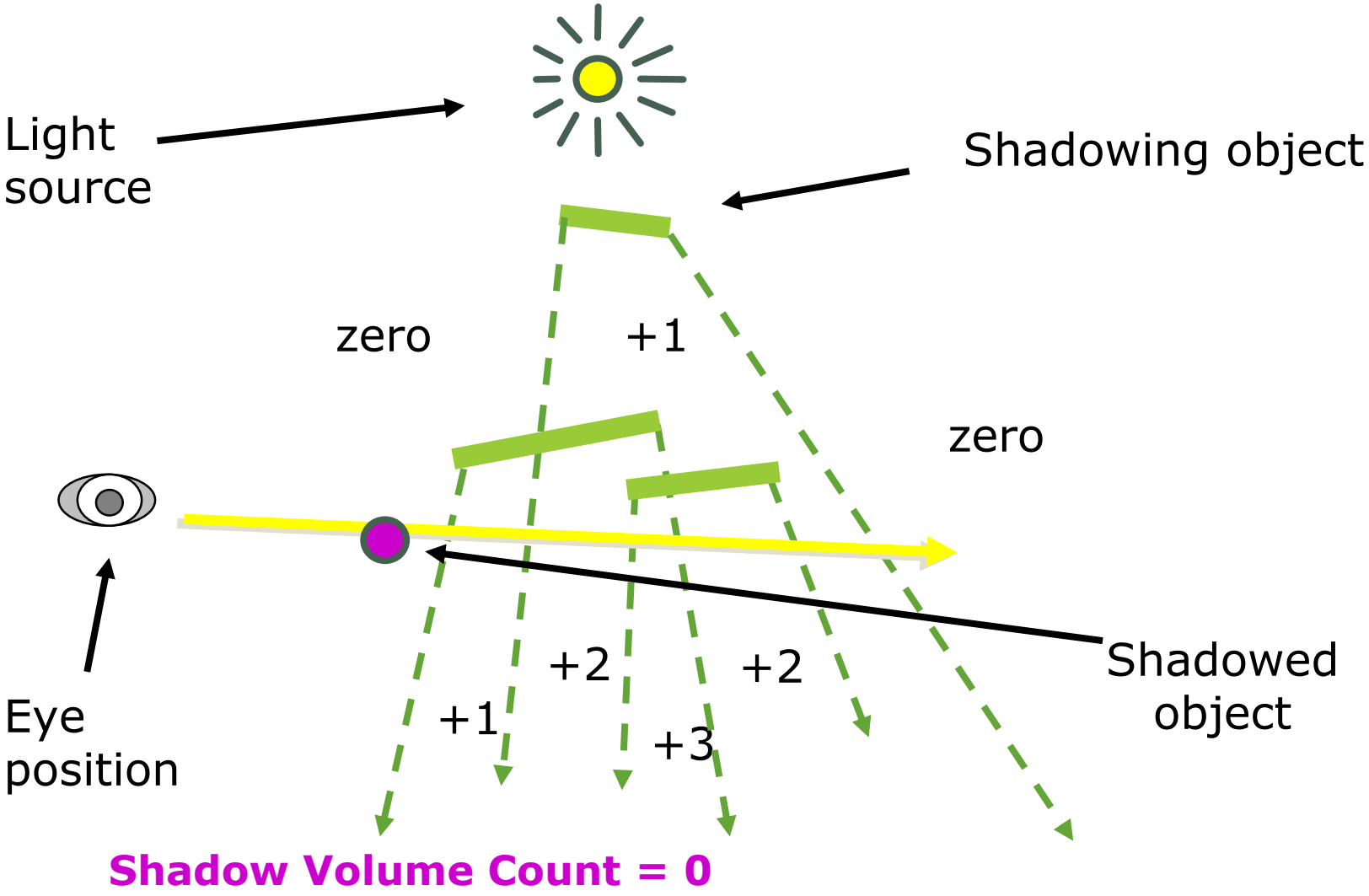
# Stencil Enter/Leave Counting Approach




**Shadow Volume Count = +1+1+1-1 = 2**

From C.Everitt and M.J.Kilgard, Practical & Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering, Nvidia

# Stencil Enter/Leave Counting Approach




From C.Everitt and M.J.Kilgard, Practical & Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering, Nvidia

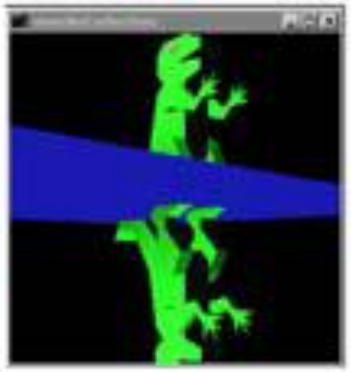
# The stencil buffer

- Used by the shadow volume algorithm
  - often 8 bits per pixel.
- When rendering to it, we can add, subtract, etc.
- Then, the resulting image can be used to mask off subsequent rendering.
  - Can specify different rendering operations for each case:
  - stencil test fails
  - stencil test passes & depth test fails
  - stencil test passes & depth test passes

# Stencil buffer – real-time mirror

- Clear frame, depth & stencil buffers
- Draw all non-mirror geometry to frame & depth buffers
- Draw mirror to stencil buffer, where depth buffer passes
- Set depth to infinity, where stencil buffer passes
- Draw reflected geometry to frame & depth buffer, where stencil buffer passes

without stencil buffer



Reflected objects

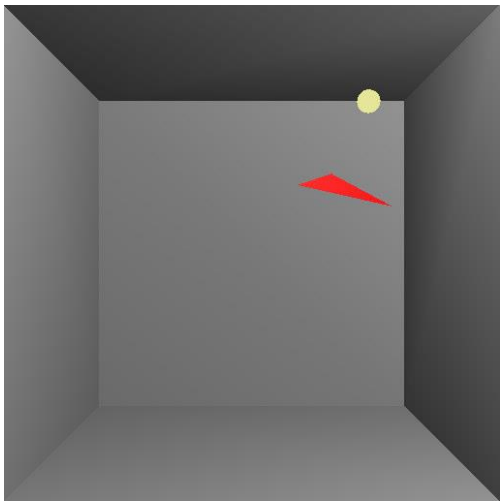
with stencil buffer

Figures from NVIDIA's stencil buffer tutorial http://developer.nvidia.com

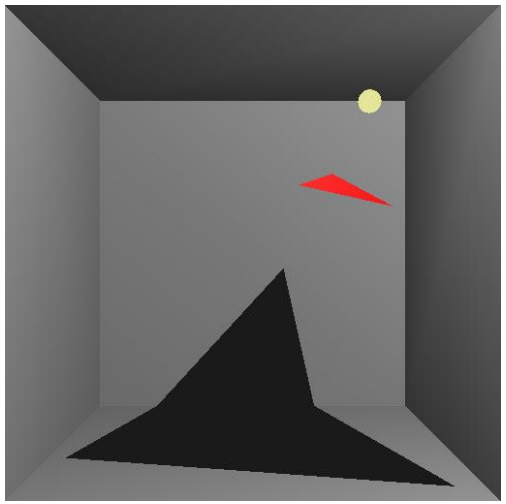
# The shadow volume algorithm (Z pass)

- Rendered in four passes:
  - Pass 1: Draw the scene using ambient lighting only.
    - This is to setup the Z-buffer.
    - Turn off z-buffer and color buffer updates
    - (In later operations, only write to **stencil buffer**)
  - Pass 2: Draw front-facing shadow volume polygons.
    - Set stencil buffer to increment if depth test passes
  - Pass 3: Draw back-facing shadow volume polygons.
    - Set stencil buffer to decrement if depth test passes
  - Pass 4: Draw diffuse and specular lighting.
    - Set stencil buffer to mask all pixels where stencil buffer value is not 0.

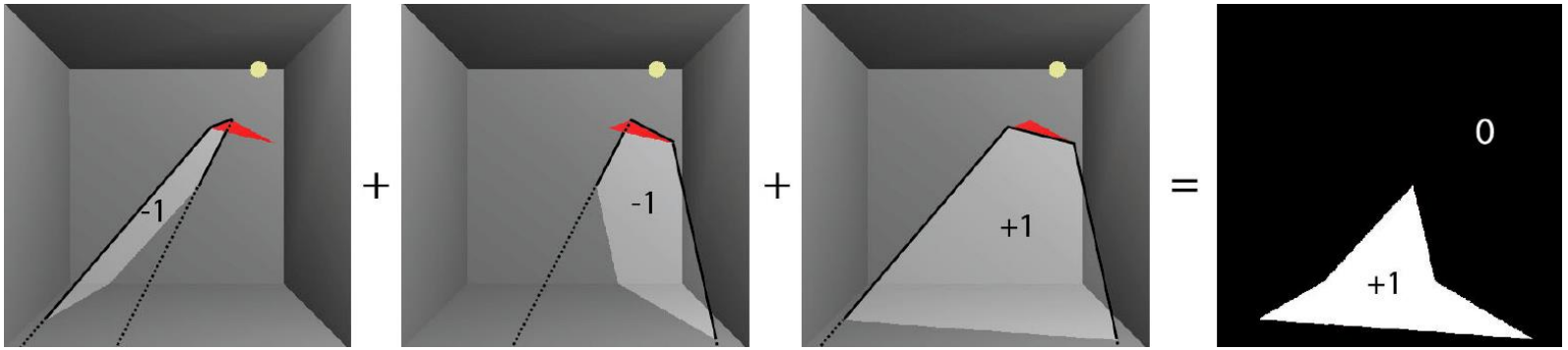
# Stencil buffer for Z-pass algorithm



without shadow

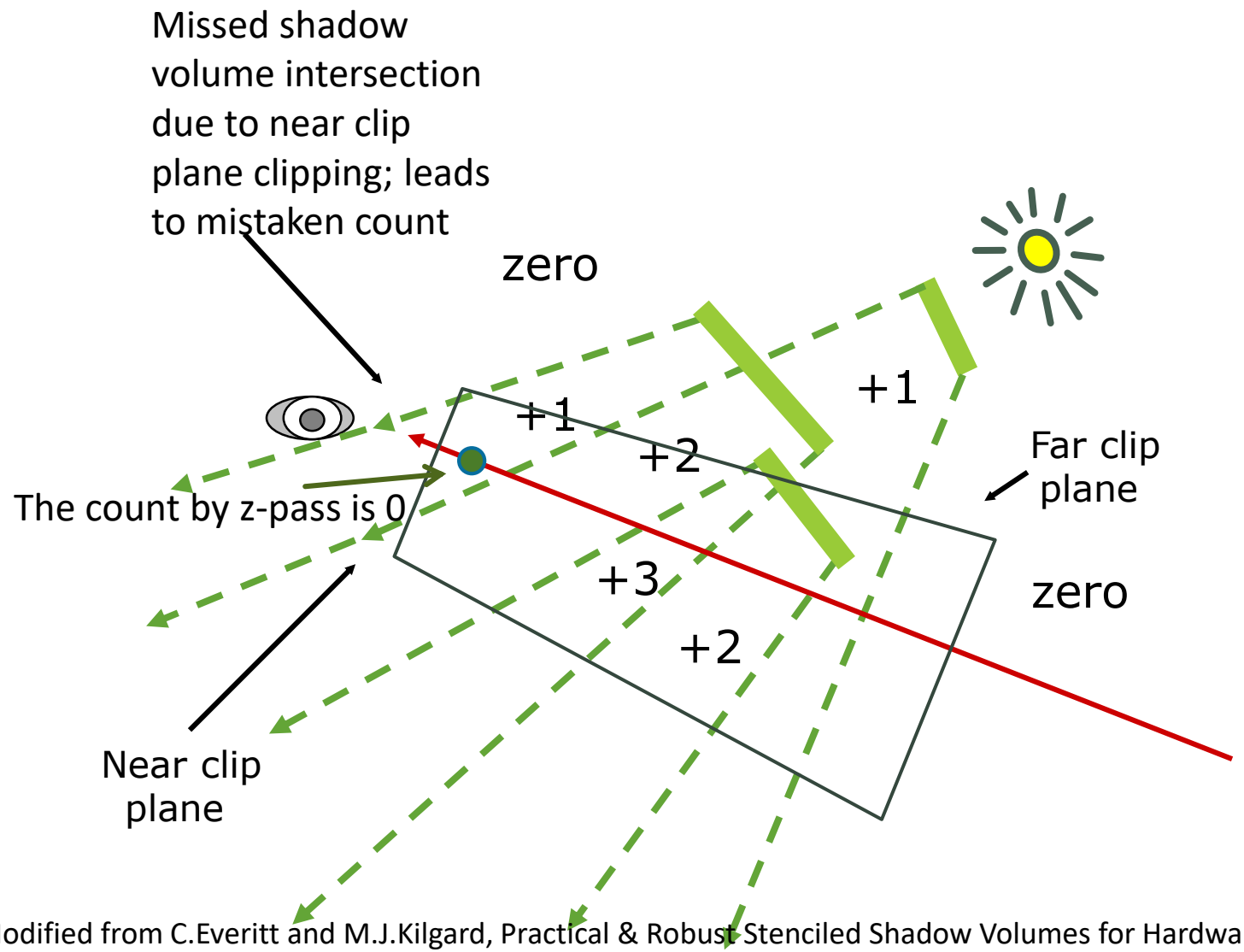


with shadow




Figures from T. Akenine-Möller, Lecture note of “Advanced Shading and Rendering”, Lund Univ.

# Problems created by near plane clipping (*Zpass approach*)




Modified from C.Everitt and M.J.Kilgard, Practical & Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering

# Shadow map vs. shadow volume

- **Shadow Volumes**
  - Good: Anything can shadow anything, including self- shadowing, and the shadows are delicate.
  - Bad: 3 or 4 passes, shadow volume polygons must be generated and rendered (lots of polygons & fill), intensive computation.
  - Ugly: frustum problems for z-pass.

- **Shadow Maps**
  - Good: Anything to anything, constant cost regardless of scene complexity, map can sometimes be reused.
  - Bad: Frustum limited.
  - Ugly: Jagged shadows if resolution is too low, biasing headaches.

# The End of Chapter