Image from: https://api.unrealengine.com/udk/Three/ShadowingReference.html

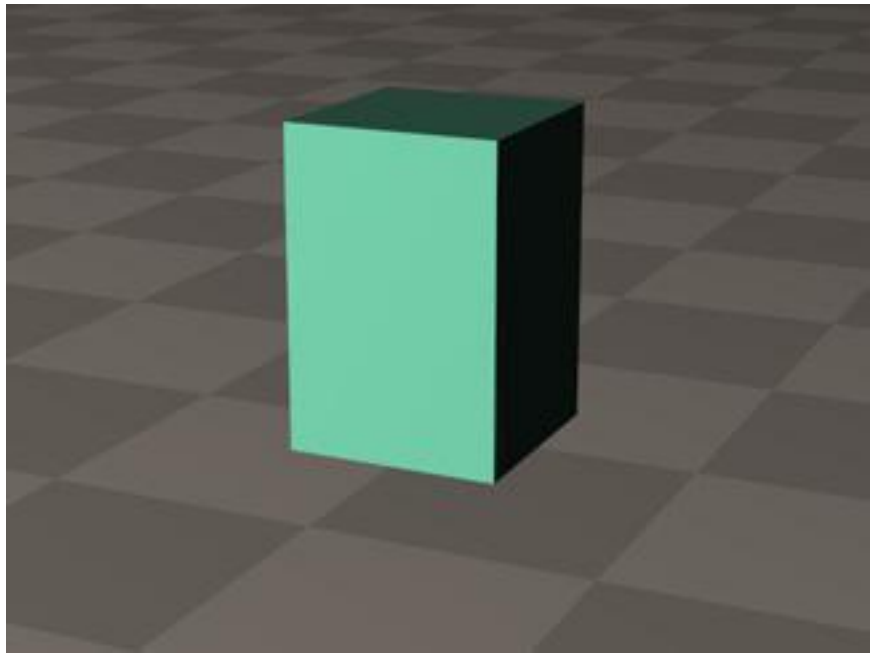
Γραφικά Υπολογιστών

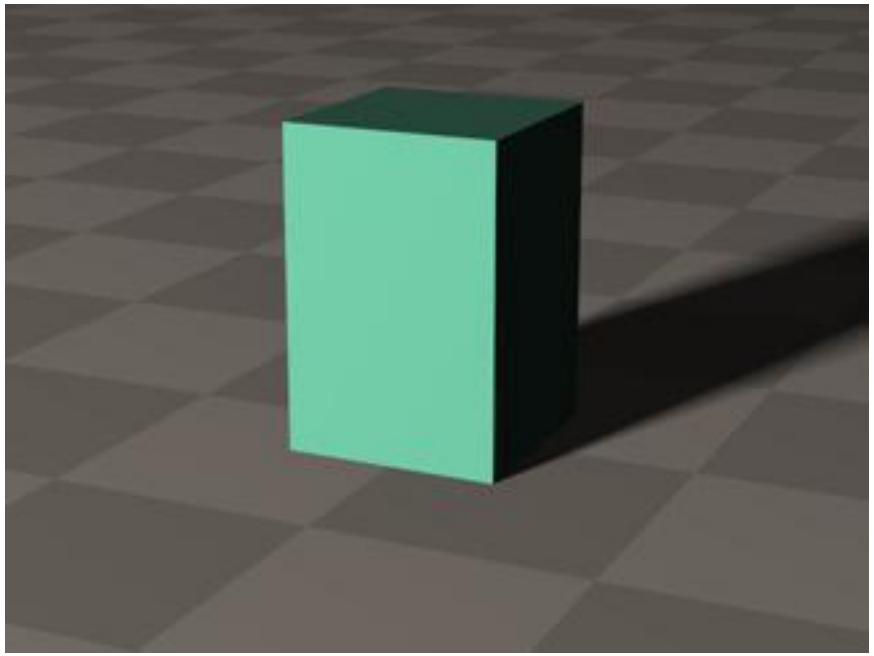
Σκιές και Διαφανείς Επιφάνειες

Andreas Aristidou
andarist@ucy.ac.cy
http://www.andreasaristidou.com

Σκιές

Shadows tell us about the relative locations and motions of objects

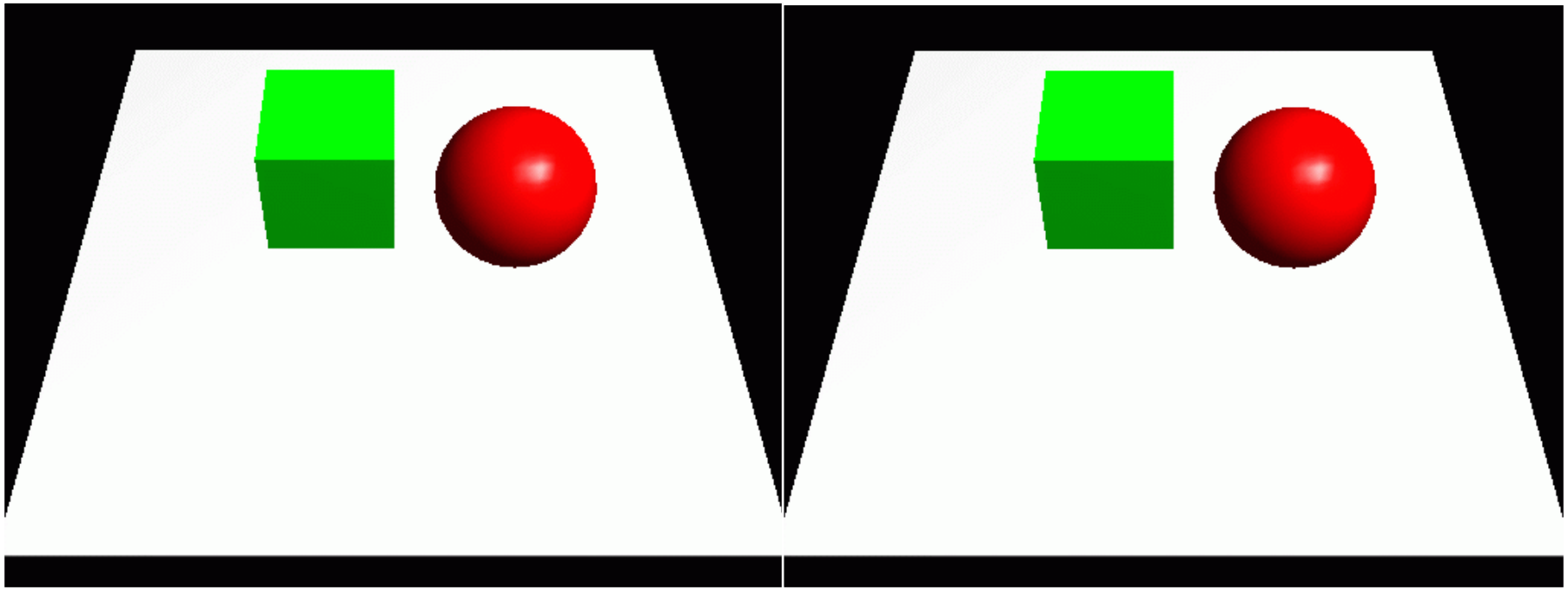




http://gandalf.psych.umn.edu/users/kersten/kersten-lab/images/ball-in-a-box.mov

Κίνητρο

- Οι σφαίρες και οι κύβοι κάθονται πάνω στο επίπεδο;
- Ποια η σχέση μεταξύ τους;
- Είναι οι ίδιες οι δύο εικόνες;

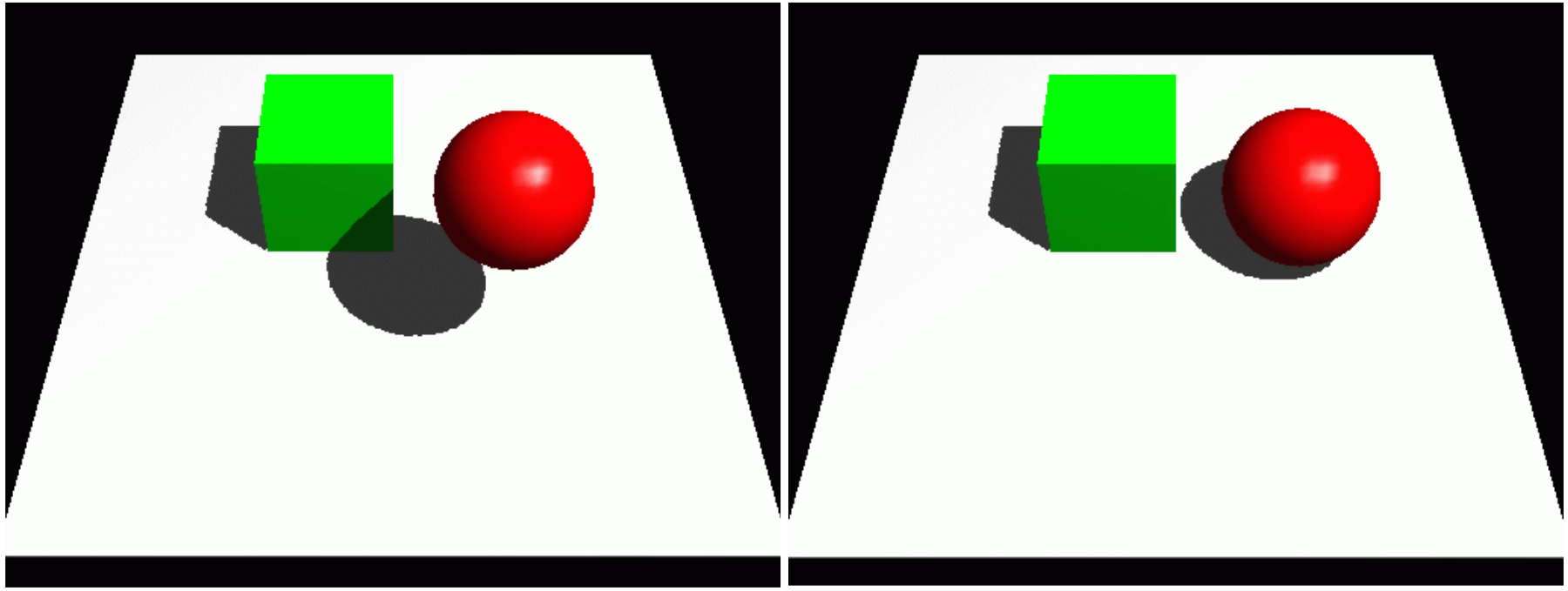


Το αποτέλεσμα της προσθήκης σκιών

- **Αριστερά**
  - Ο κύβος είναι πάνω στο επίπεδο, η σφαίρα όχι
  - Η σφαίρα είναι πιο μπροστά από τον κύβο

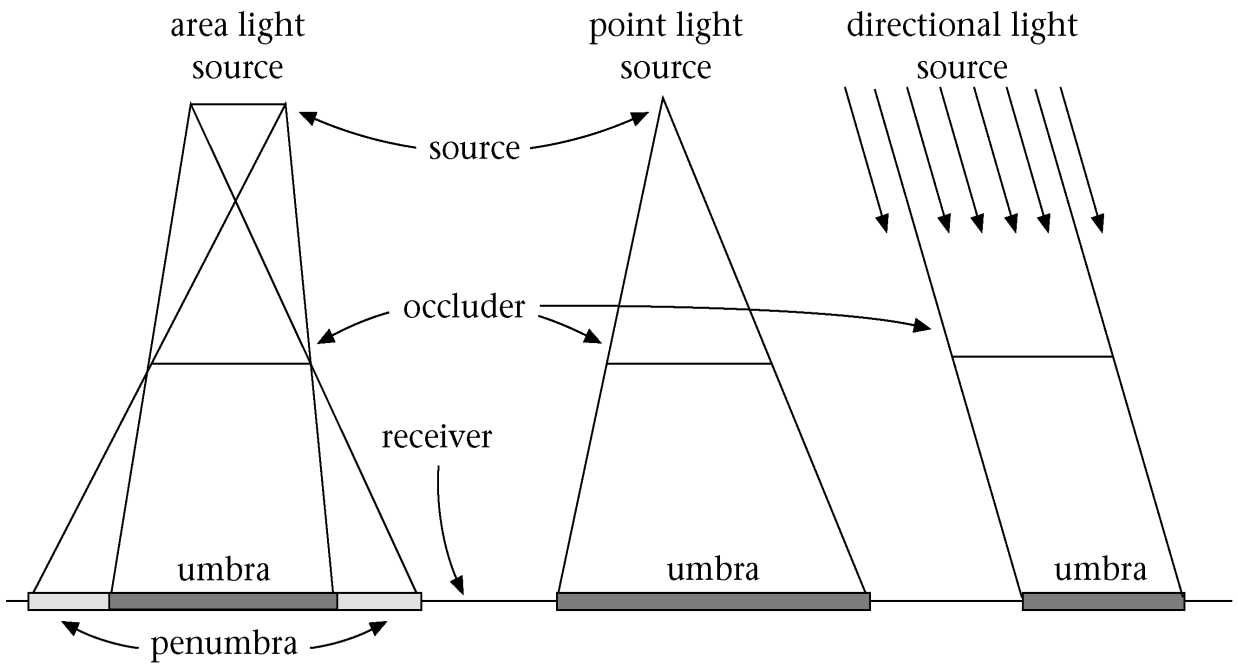
Δεξιά

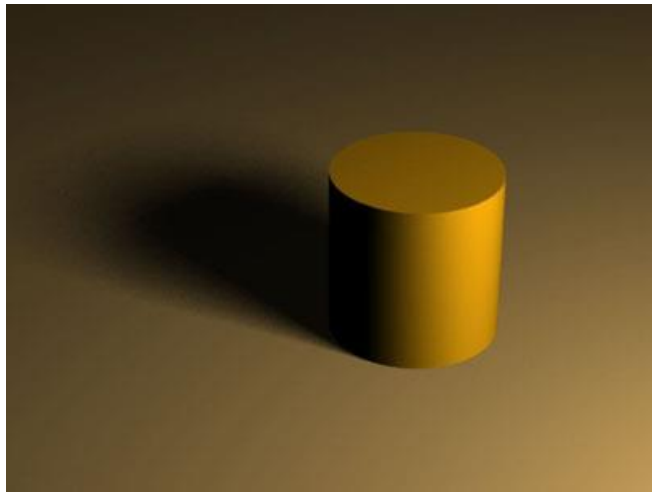
- Είναι και τα δύο στο επίπεδο
- Η σφαίρα είναι δίπλα από τον κύβο



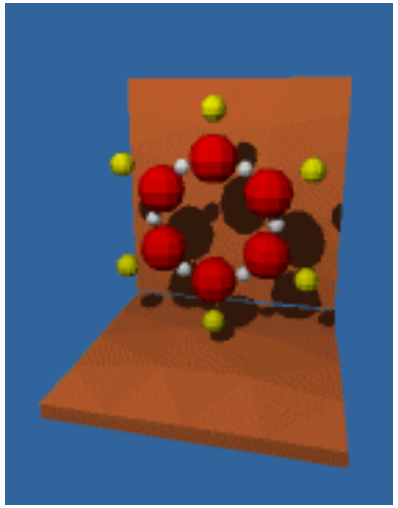

George Drettakis

Σκιές: *Μαλακές και Σκληρές σκιές*





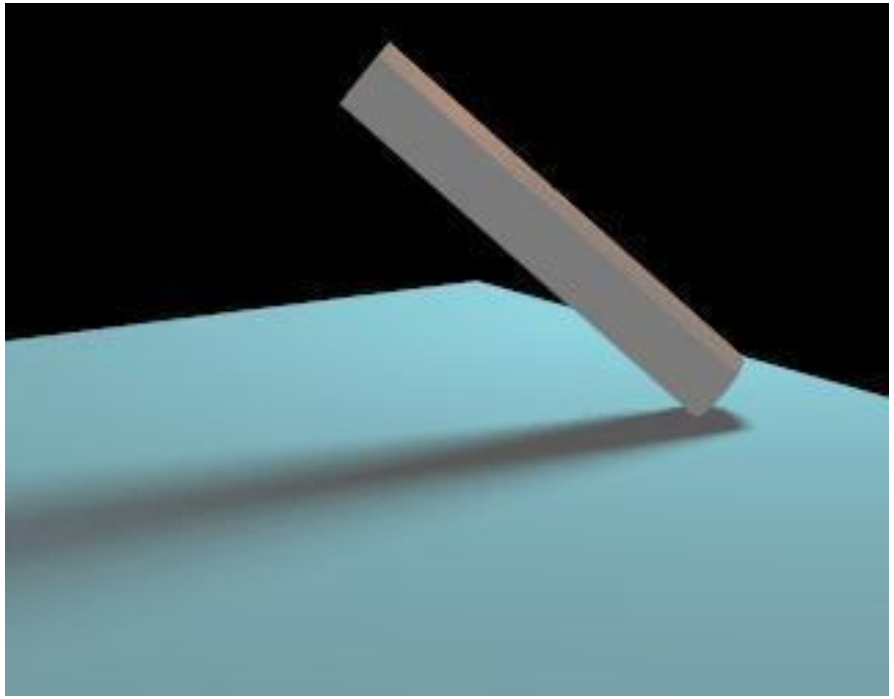
Soft shadows



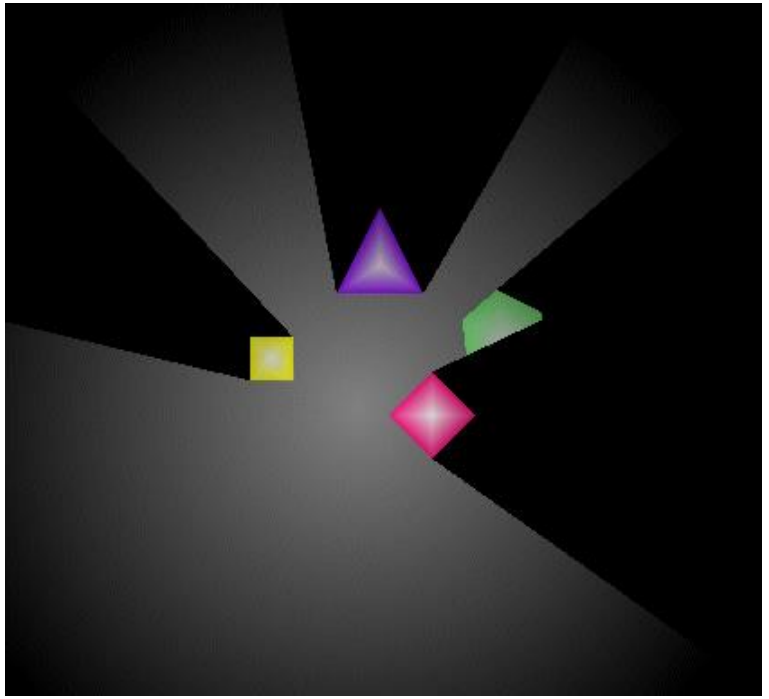
Hard shadows

Σκιές: *Μαλακές και Σκληρές σκιές*

- Point lights have hard edges, and area lights have soft edges

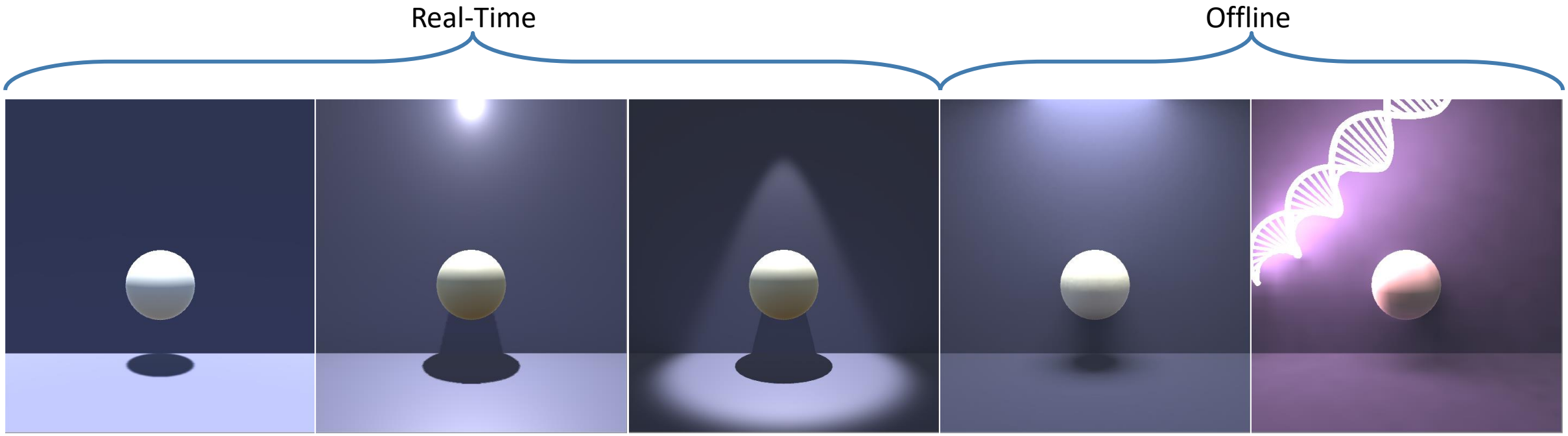


Soft shadows



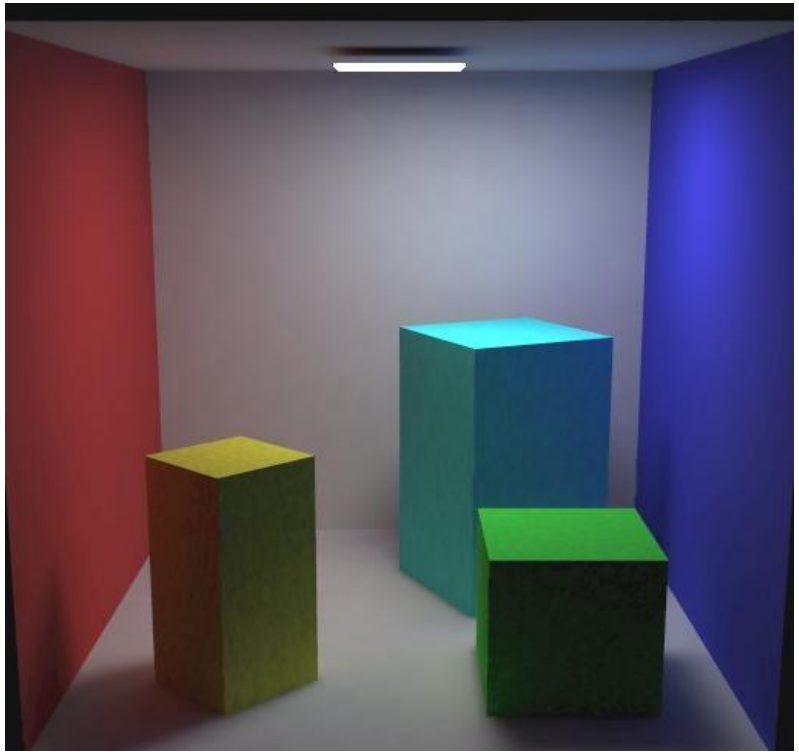
Hard shadows

Σκιές


Real-Time
Offline
Directional
Point Light
Spotlight
Area Light
Mesh Light

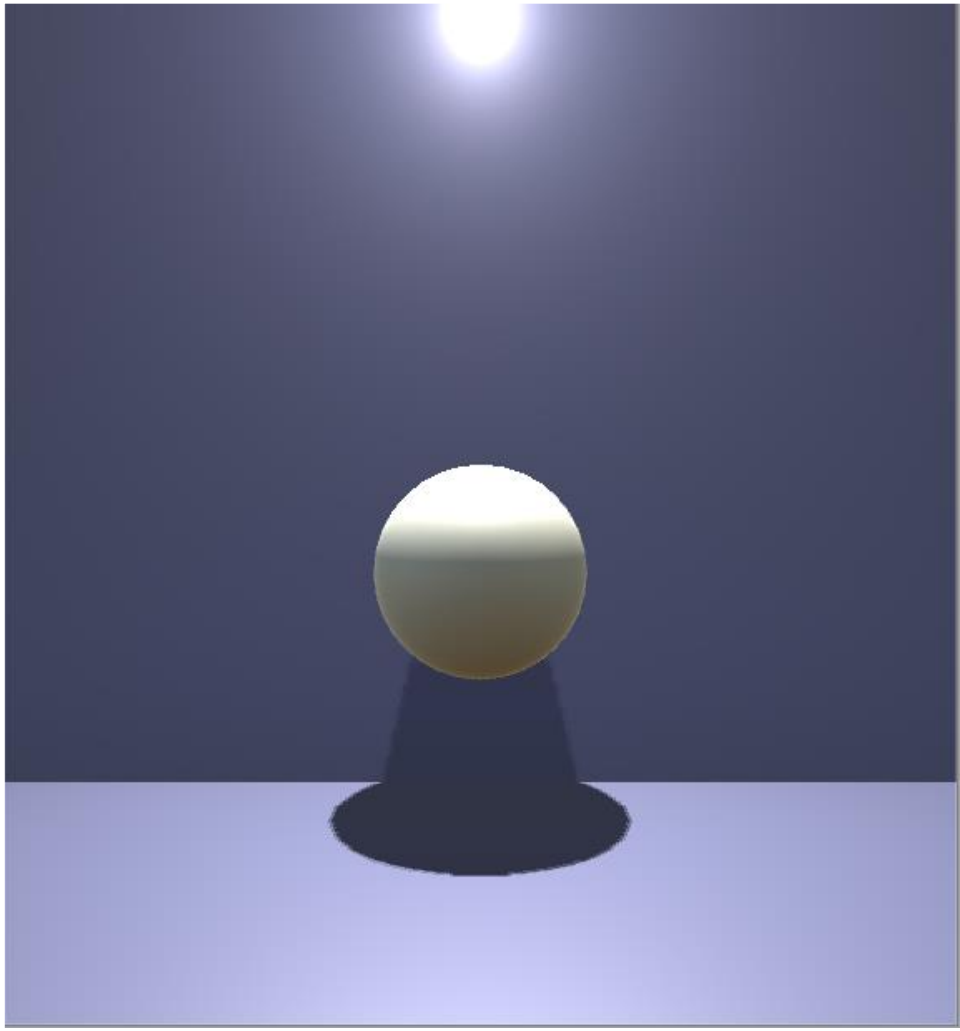
Rendering Shadows

- Despite its importance, rendering shadows is not very straight forward
- Precise rendering of shadows require **ray tracing or global illumination techniques**, which are very computationally costly
- In order to avoid such intense computation, many techniques have been proposed for the graphics pipeline.
  - Projective shadows
  - Shadow texture
  - Shadow volume
  - Shadow map
  - Soft shadows



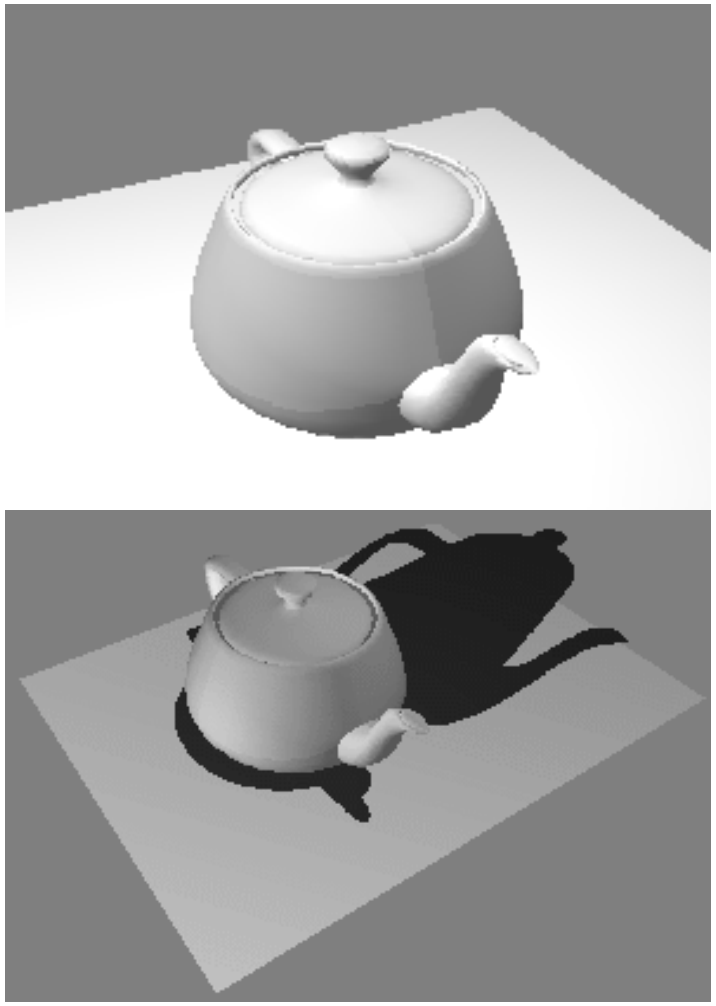
Σκιές από σημειακές πηγές

- Υπολογίζονται πολύ πιο εύκολα και γρήγορα
- Για εφαρμογές πραγματικού χρόνου είναι συνήθως η μόνη επιλογή
- Τύποι αλγόριθμων
  - Υπάρχουν δεκάδες διαφορετικές προσεγγίσεις
  - Λόγω των GPU, έχει επικρατήσει το shadow mapping
  - **Υπολογίζονται σε κάθε frame**



Shadows – Simplest Hack

- Render each object twice
  - First pass: render normally
  - Second pass: use transformations to project object onto ground plane, render completely black
- Pros: Easy, can be convincing. Eye more sensitive to presence of shadow than shadow's exact shape
- Cons: Becomes complex computational geometry problem in anything but simplest case
  - Easy: projecting onto flat, infinite ground plane
  - Hard(er): how to implement for projection on stairs? Rolling hills?



http://web.cs.wpi.edu/~matt/courses/cs563/talks/shadow/shadow.html

Ψευδοσκιές (Fake Shadows)

- Ακόμη τις βλέπουμε σε κάποια παιχνίδια

Images from TombRaider. ©Eidos Interactive.

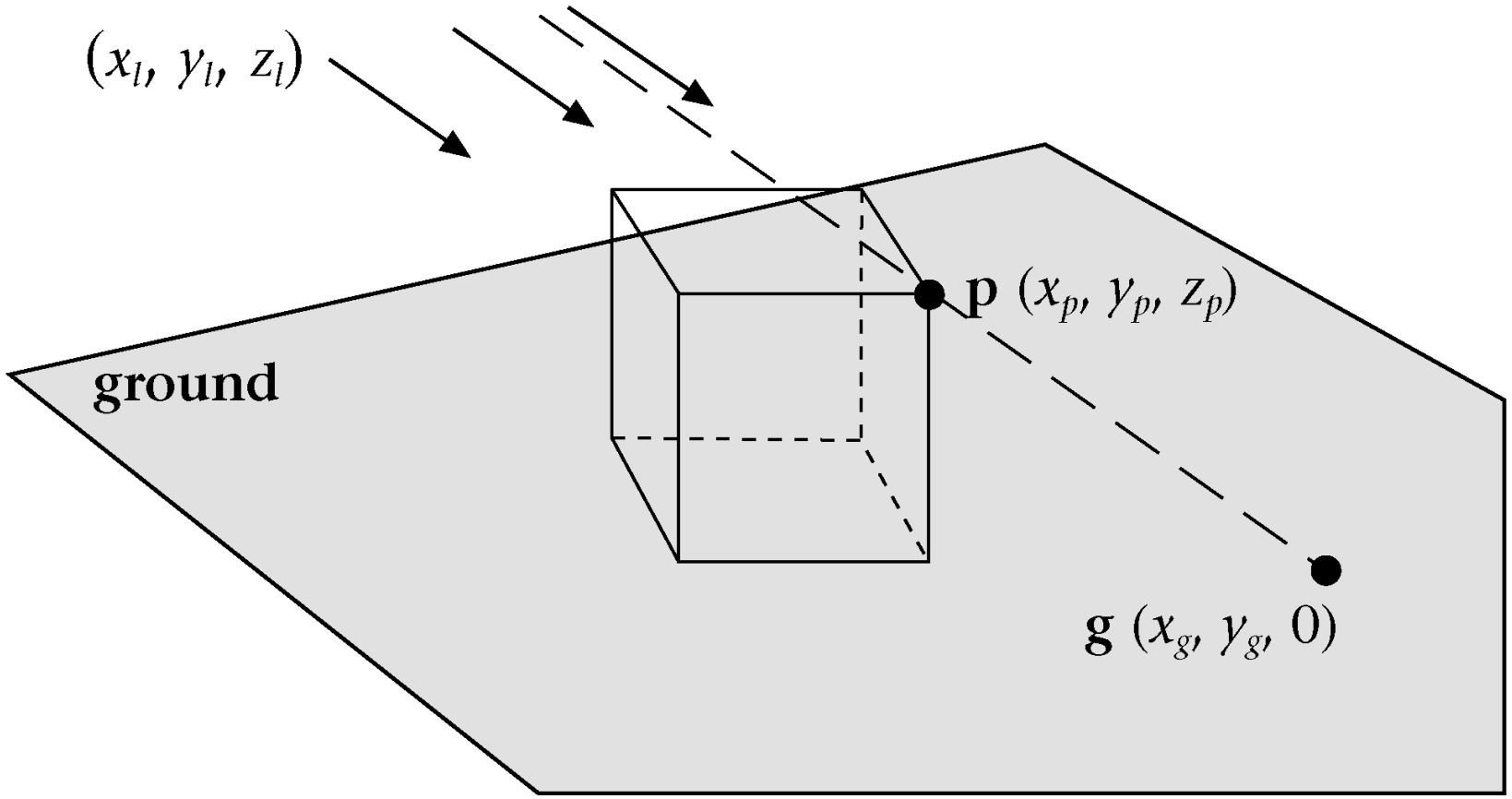
- Στην πιο απλή τους μορφή:
  - Χρήση ενός σταθερού συνόλου από sprites ή ψεύτικων προβολών (fake projections) σαν σκιές
- No global effect

…but better than no shadow at all ☺

Ψευδοσκιές

- Προβολή των πολυγώνων στο πάτωμα
- Να επισημάνουμε ότι:
  - Χρειάζεται ο ανασχεδιασμός ολόκληρης της σκηνής για κάθε receiver (ή τουλάχιστον όλων των αντικειμένων που εμφανίζουν σκιά)
  - Οι σκιές δεν αποκόπτονται στην ακμή του receiver
  - Δεν υπάρχουν σκιές μεταξύ αντικειμένων (inter-object)
  - Αντικείμενα κάτω από τον receiver ακόμα προκαλούν σκιές

Ψευδοσκιές για κατευθυνόμενη πηγή



Ψευδοσκιές για κατευθυνόμενη πηγή

- Κάθε σημείο $p = (x_p, y_p, z_p)$ προβάλλεται στο $g = (x_g, y_g, 0)$ βάση της κατεύθυνσης του φωτός $L = (x_l, y_l, z_l)$
- Αυτό γίνεται προσθέτοντας ένα πίνακα στην γραφική σωλήνωση που «ισοπεδώνει» τα αντικείμενα
- Έχουμε

  $g = p + t \cdot L$ αφού $z_g = 0 \Rightarrow 0 = z_p + t \cdot z_l \Rightarrow t = -z_p/z_l$

  και

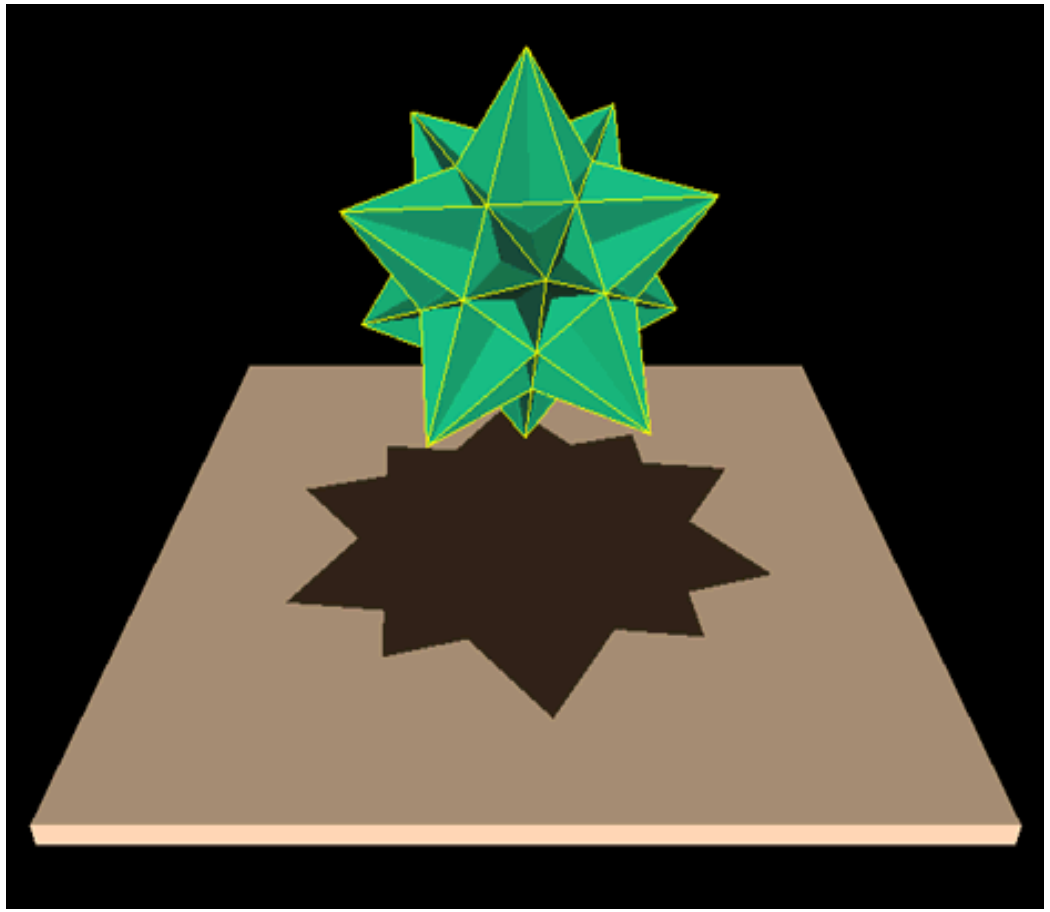
  $x_g = x_p - (z_p/z_l).x_l$

  $y_g = y_p - (z_p/z_l).y_l$

- Τα πιο πάνω μας δίνουν τον πίνακα:

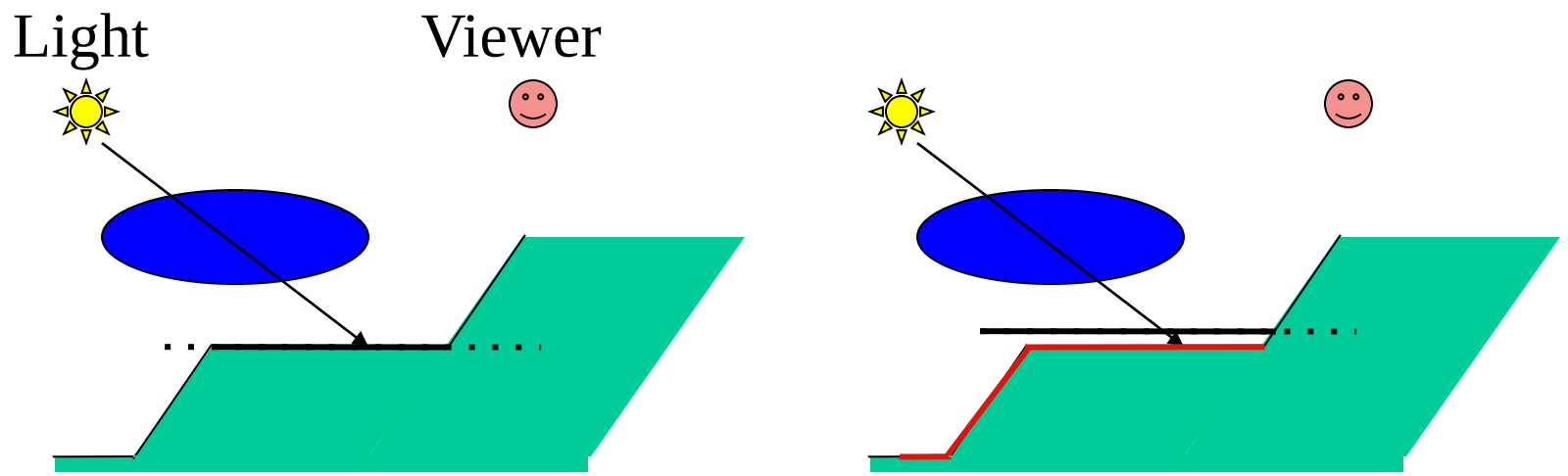
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -x_l/z_l & -y_l/z_l & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Παράδειγμα με ψευδοσκιές



Bert Schoenwaelder

Lifting the shadow above the surface



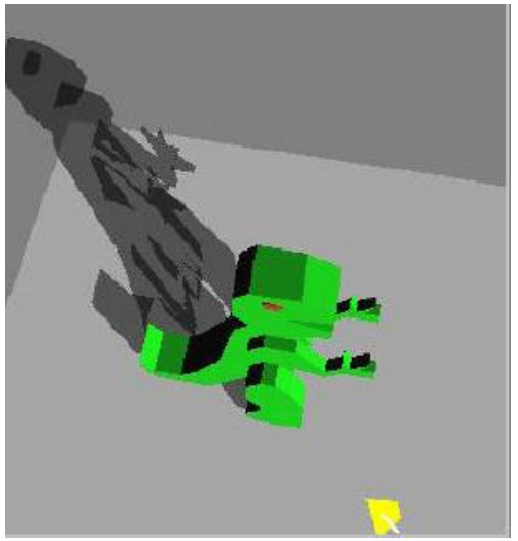
Shadow height equal with hit polygon

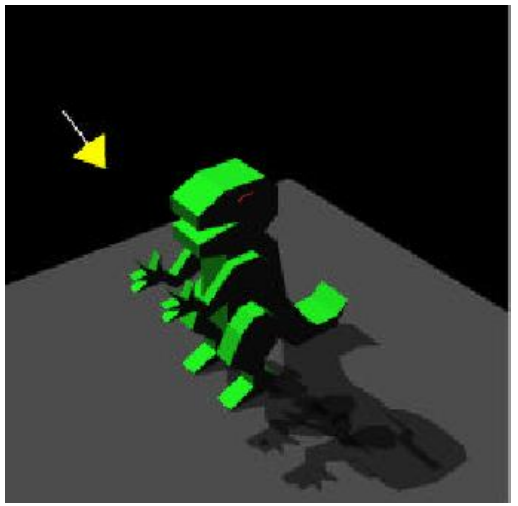
Shadow height above hit polygon

- Tricks:
  - Lift the shadow a little off the plane to avoid z-buffer quantization errors

Point Light Shadows : problem

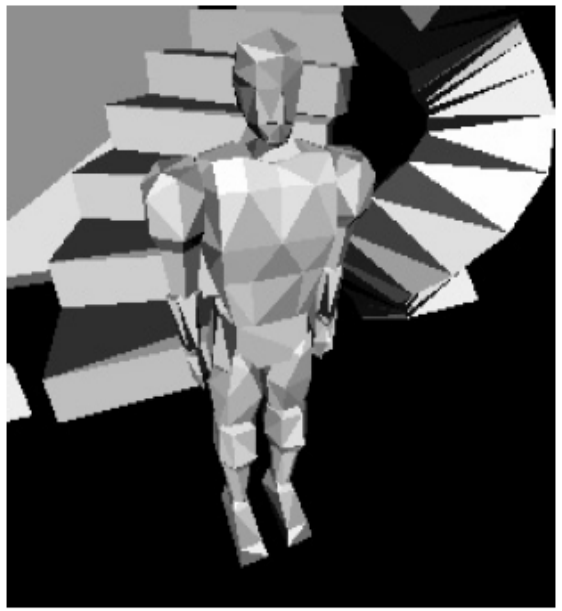
- shadows can only be cast onto planes and not on arbitrary objects.
- the resulting shadows generated have hard edges
- If there is a texture on the floor, grey shadows look bad
  - If we use blending, the shadow part with multiple polygons overlapping will look darker
- The shadows need to be re-rendered at every frame although its shape is view independent





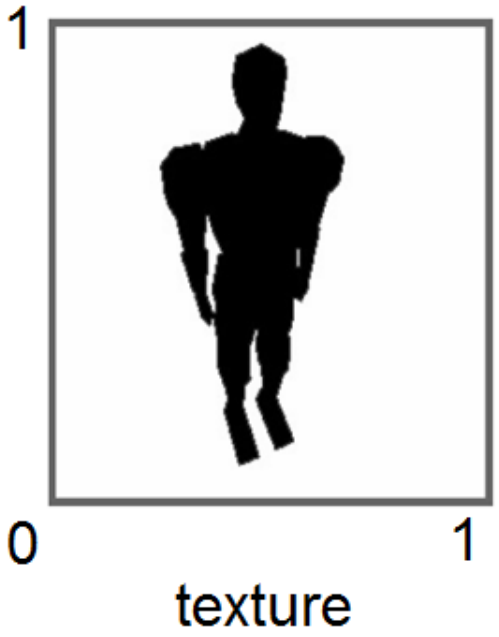
Shadow Texture

- Use a shadow image as a projective texture
- Generate an image of the occluder from the light's view and color it grey
- Produce a shadow by texture mapping this image onto the background object

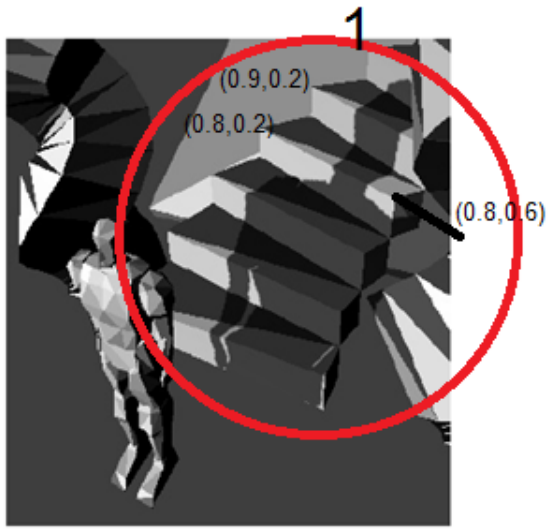


Q: How do we compute the uv coordinates?

- For mapping the shadow texture, we need to know the uv coordinates of the texture at every vertex of the background mesh



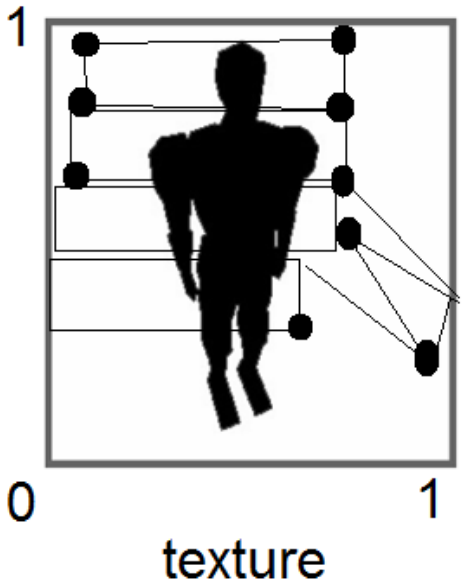
texture



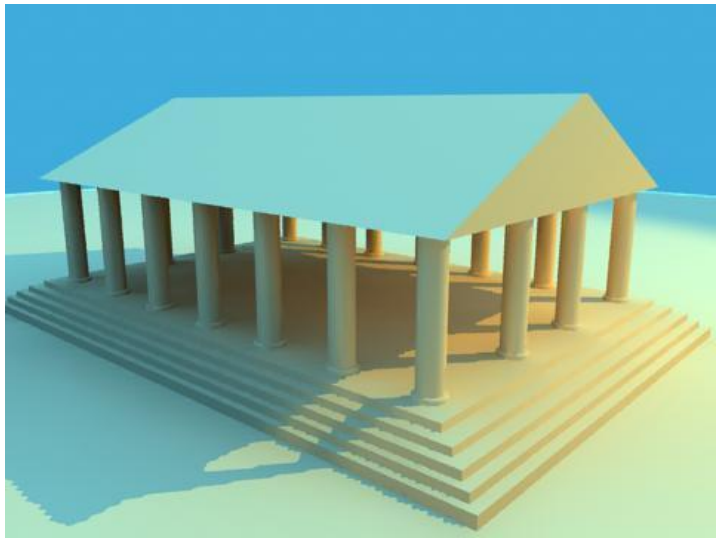
mapping onto the background

Q: How do we compute the uv coordinates?

- View the object from the light source
- Project the background object onto the projection plane used to produce the shadow texture
- Obtain the (x,y) coordinates and normalize
  - This becomes the uv coordinates



Shadow Texture : Cons and Pros

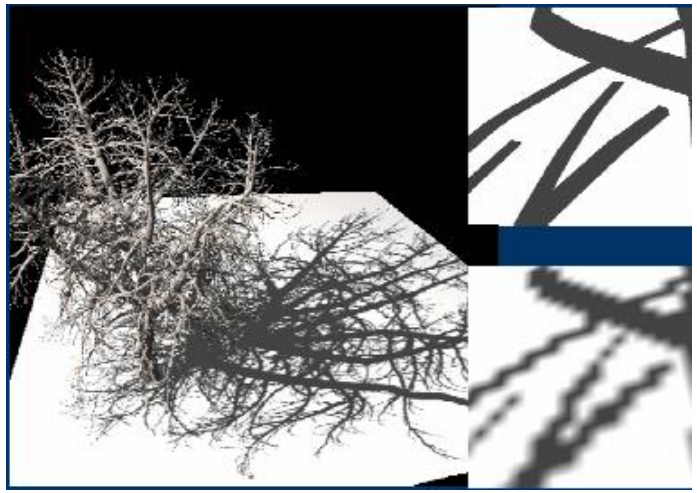


Pros

- How is it good compared to shadow texture?
- The shadow does not need to be recomputed if the occluder does not move.

Cons

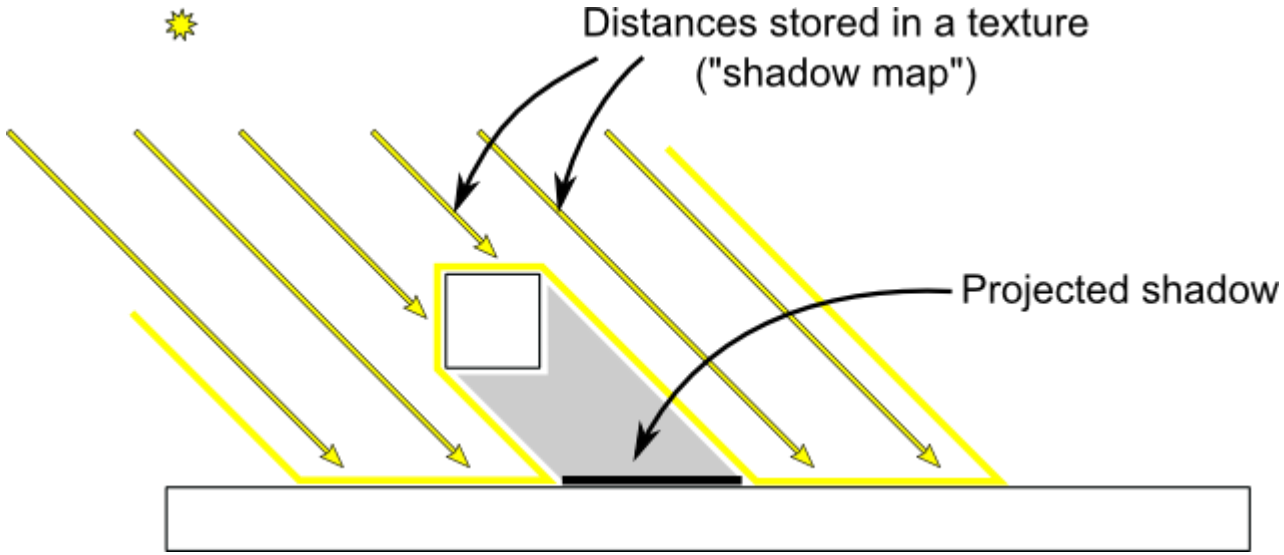
- The object that produces the shadow and the subject that it is projected onto must be specified.
- The quality of the shadow is affected by the resolution of the shadow texture.
- Cannot produce self-shadows.



Shadow Mapping

Shadow Mapping: *Βασική αρχή*

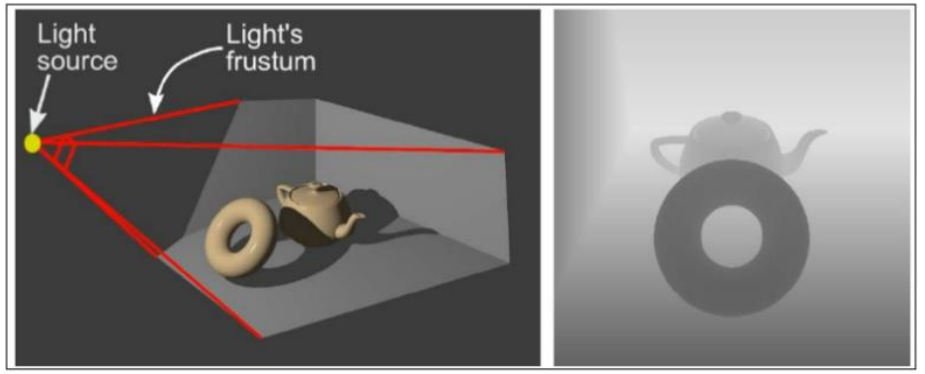
- The basic shadowmap algorithm consists in two passes.
  - The scene is rendered from the point of view of the light. Only the depth of each fragment is computed.
  - The scene is rendered as usual, but with an extra test to see if the current fragment is in the shadow.
- The “being in the shadow” test is actually quite simple. If the current sample is further from the light than the shadowmap at the same point, this means that the scene contains an object that is closer to the light. In other words, the current fragment is in the shadow.



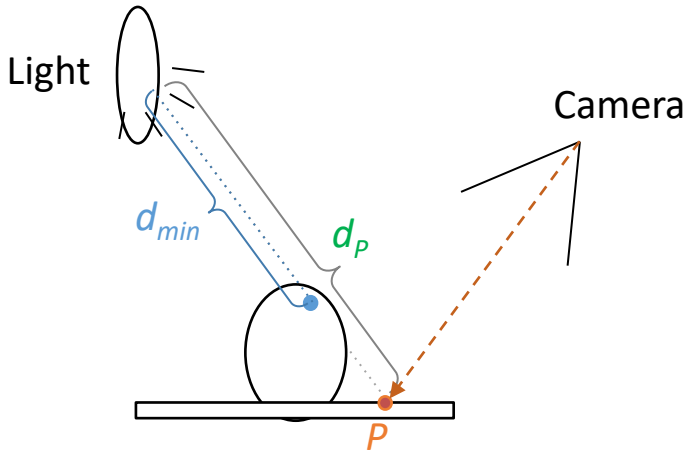
OpenGL Tutorial: http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/

Shadow Mapping: *Βασική αρχή*

- Render scene using each light as center of projection, saving only its z-buffer
  - Resultant 2D images are “shadow maps”, one per light
- Next, render scene from camera’s POV
  - To determine if point *P* on object is in shadow:
    - compute distance d_P from *P* to light source
    - convert *P* from world coordinates to shadow map coordinates using the viewing and projection matrices used to create shadow map
    - look up min distance d_{min} in shadow map
    - *P* is in shadow if $d_P > d_{min}$, i.e., it lies behind a closer object



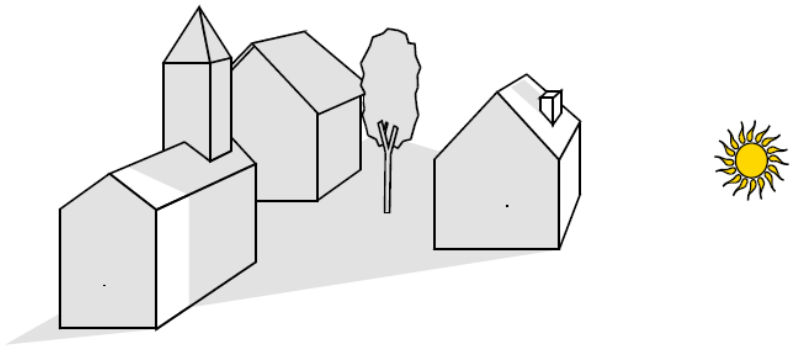
Shadow map (on right) obtained by rendering from light’s point of view (darker is closer)



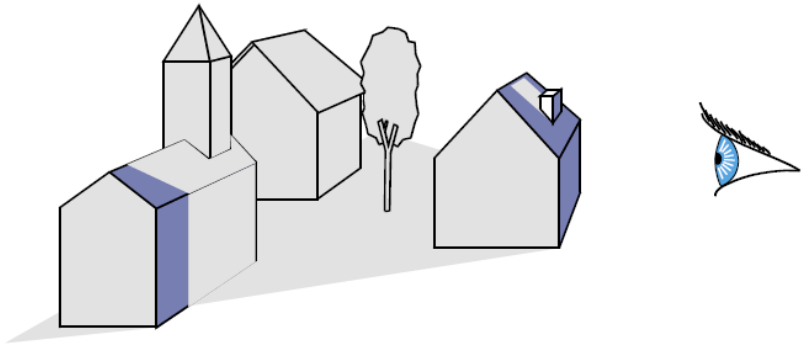
Shadow Mapping: *Βασική αρχή*

Shadow/View Duality

- A point is lit if it is visible from the light source



- Shadow computation similar to view computation



Introduced by **Lance Williams**. 1978. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '78). ACM, New York, NY, USA, 270-274. DOI=http://dx.doi.org/10.1145/800248.807402

Shadow mapping

- ≥ 2 περάσματα μέσω του pipeline
  - Υπολογισμός shadow map (**βάθος** από φωτεινή πηγή)
  - Σχεδιασμός τελικής εικόνας (έλεγχος shadow map για να δούμε αν τα σημεία σκιάζονται)

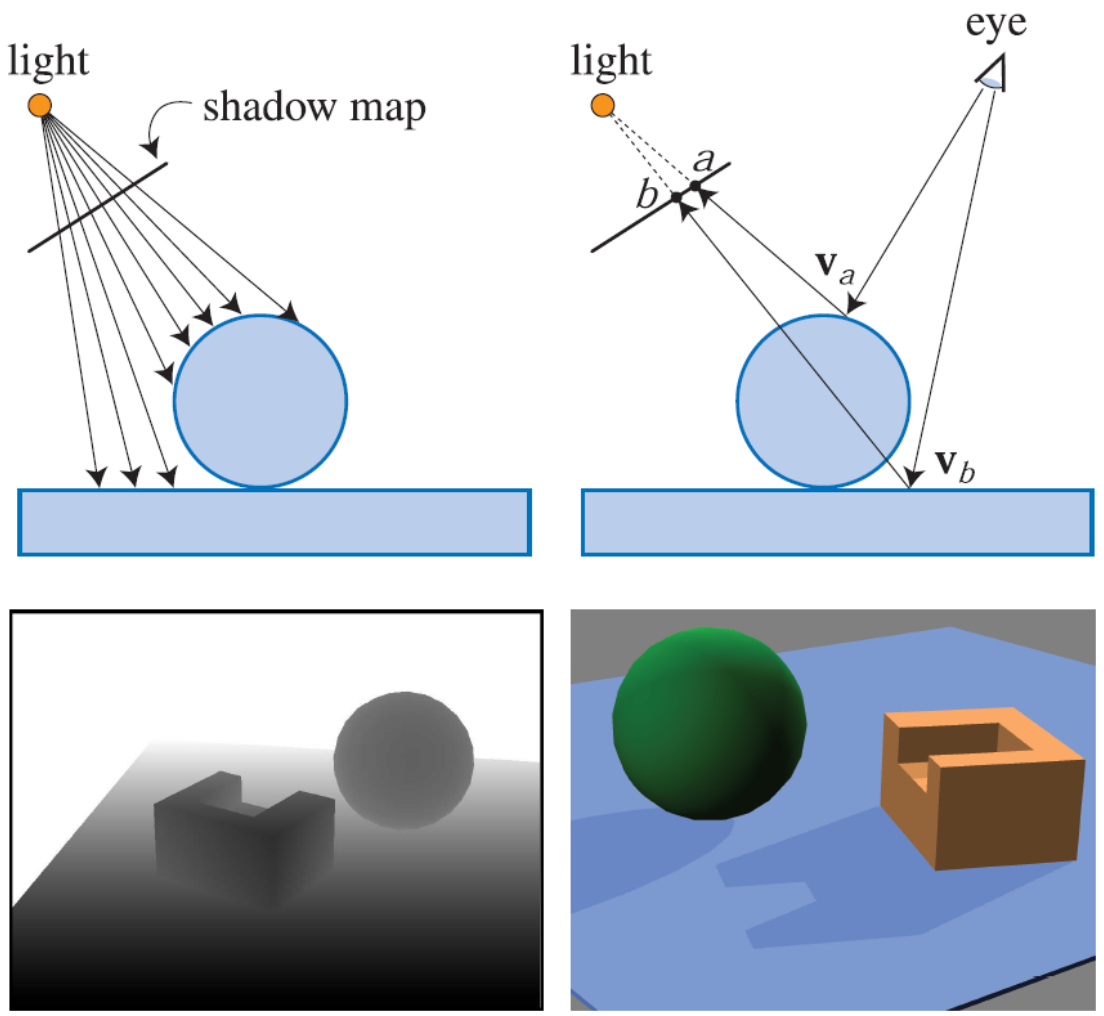
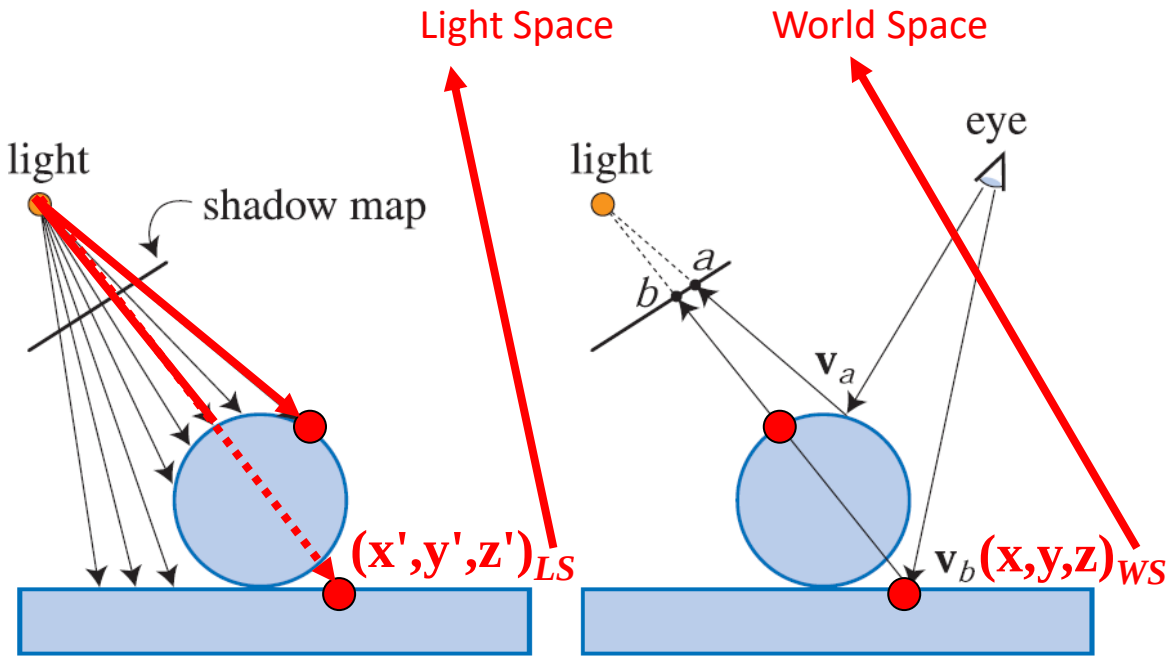


Figure from: Akenine-Moller et al. “Real time Rendering”

Shadow Map Look Up

- Έχουμε ένα 3D σημείο $(x,y,z)_{WS}$
- Πώς βρίσκουμε (look up) το βάθος από το shadow map;
- Χρήση 4x4 πίνακα προοπτικής προβολής από φωτεινή πηγή για να πάρουμε $(x',y',z')_{LS}$
- **ShadowMap(x',y') < z'?**



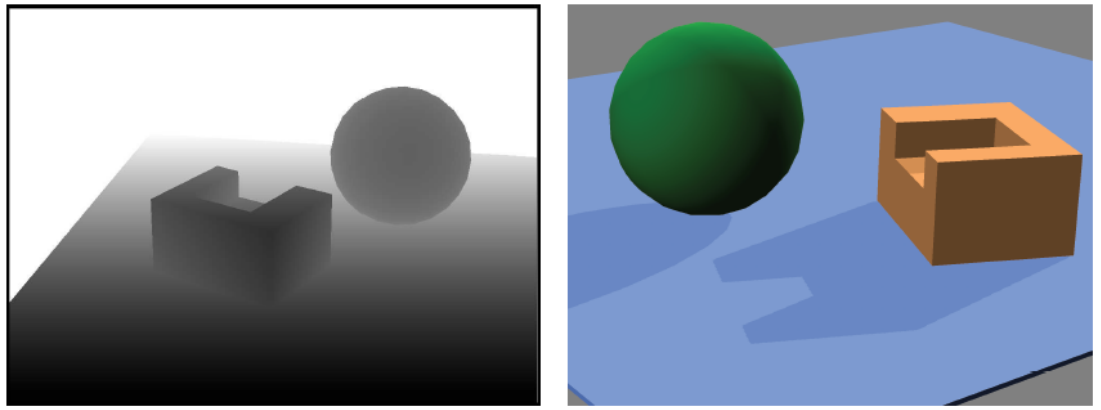
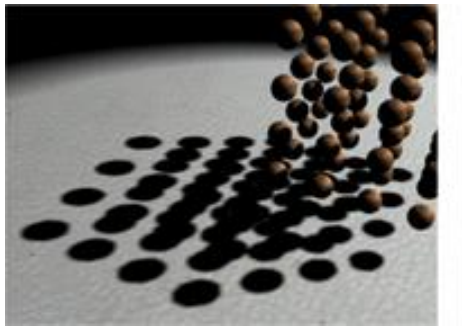


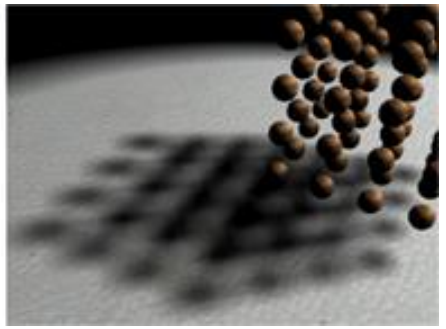
Figure from: Akenine-Moller et al. “Real time Rendering”

Shadow Mapping: *Βασική αρχή*

- Pro: Can extend to support soft shadows
  - Soft shadows: shadows with “soft” edges (e.g., via blurring)
  - Stencil shadow volumes only useful for hard-edged shadows
- Con: Naïve implementation has impressively bad aliasing problems
  - When the camera is closer to the scene than the light, **many screen pixels** may be covered by only **one shadow map pixel** (e.g., sun shining on Eiffel tower—note: z-compression again!)
- Many workarounds for aliasing issues
  - Percentage-Closer Filtering: For each fragment, sample shadow map in multiple places to see how many are in and out of shadow, then average
  - Cascaded Shadow Maps: Multiple shadow maps, higher resolution closer to viewer (like mip mapping!)



Hard shadows



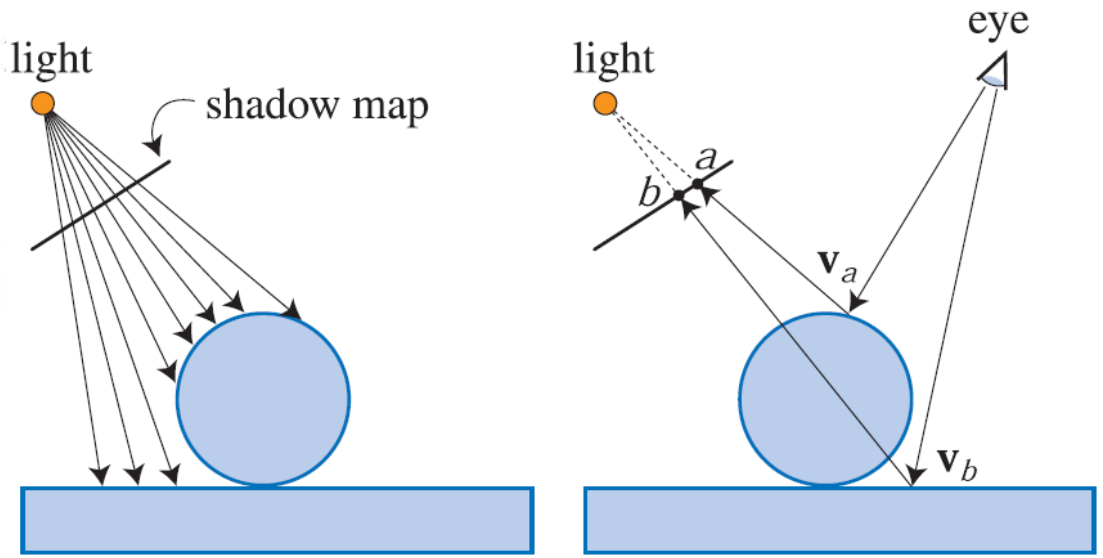
Soft shadows

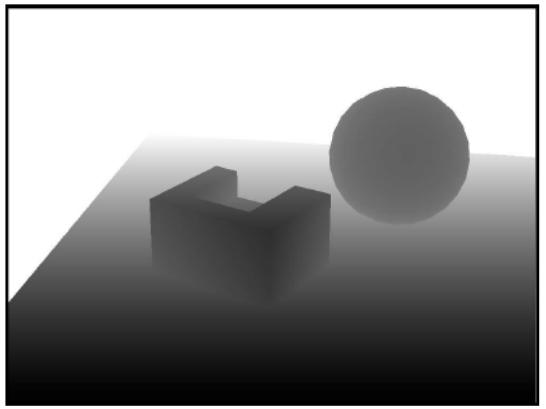
Aliasing produced by naïve shadow mapping

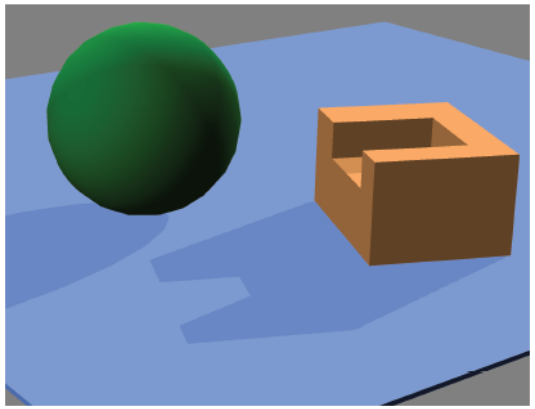
http://developer.download.nvidia.com/books/HTML/gpugems/gpugems_ch08.html
http://www-sop.inria.fr/ reves/Marc.Stamminger/psm/

Περιορισμοί των Shadow Maps

1. Οπτικό πεδίο
2. Bias (Epsilon)
3. Ταύτιση



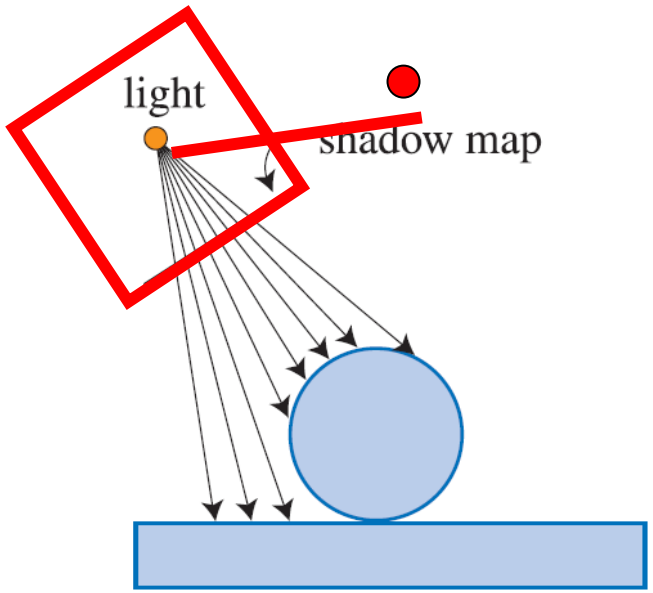


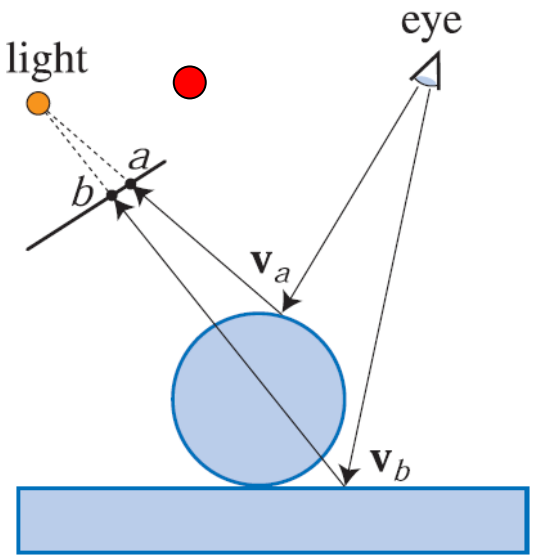


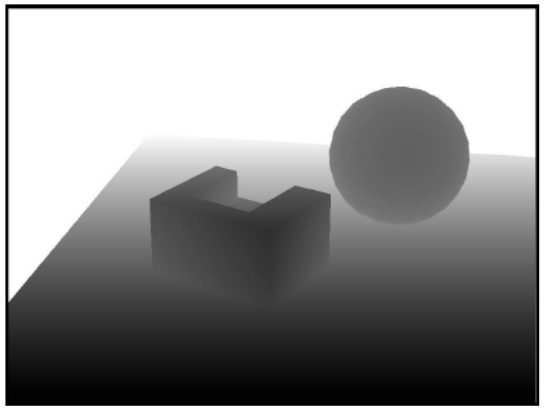
1. Το πρόβλημα του οπτικού πεδίου

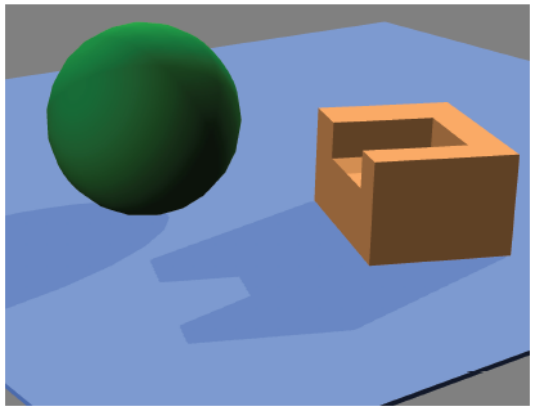
- Τι γίνεται αν το σημείο βρίσκεται έξω από το οπτικό πεδίο του shadow map;
  - **είτε** χρησιμοποιούμε «κυβικό» shadow map (δηλαδή 6 shadow map)
  - **ή** χρησιμοποιούμε μόνο spot lights ή directional lights!

https://docs.microsoft.com/en-us/windows/win32/dxtecharts/common-techniques-to-improve-shadow-depth-maps







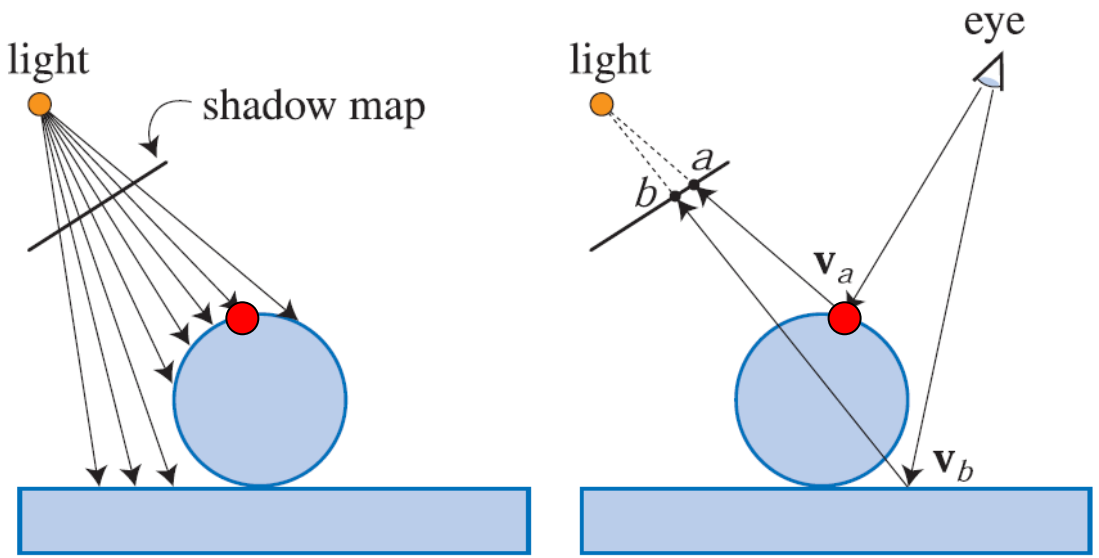


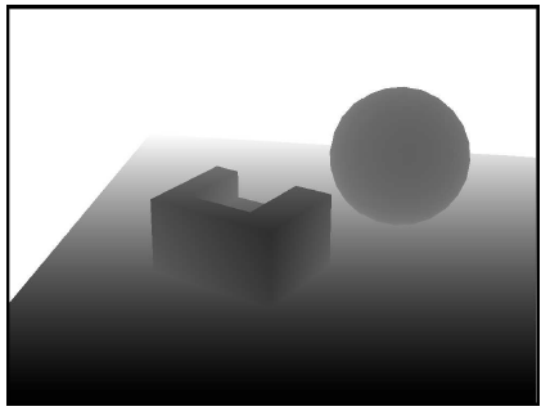
2. The Bias (Epsilon) Nightmare

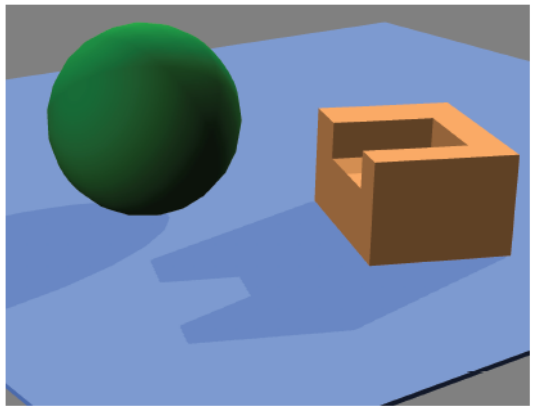
- Για ένα σημείο ορατό από τη φωτεινή πηγή

  ShadowMap(x',y') ≈ z'

- Πώς μπορούμε να αποφύγουμε λανθασμένα αυτό-σκίαση (self-shadowing)?
  - Προσθέτουμε **bias** (**epsilon**)





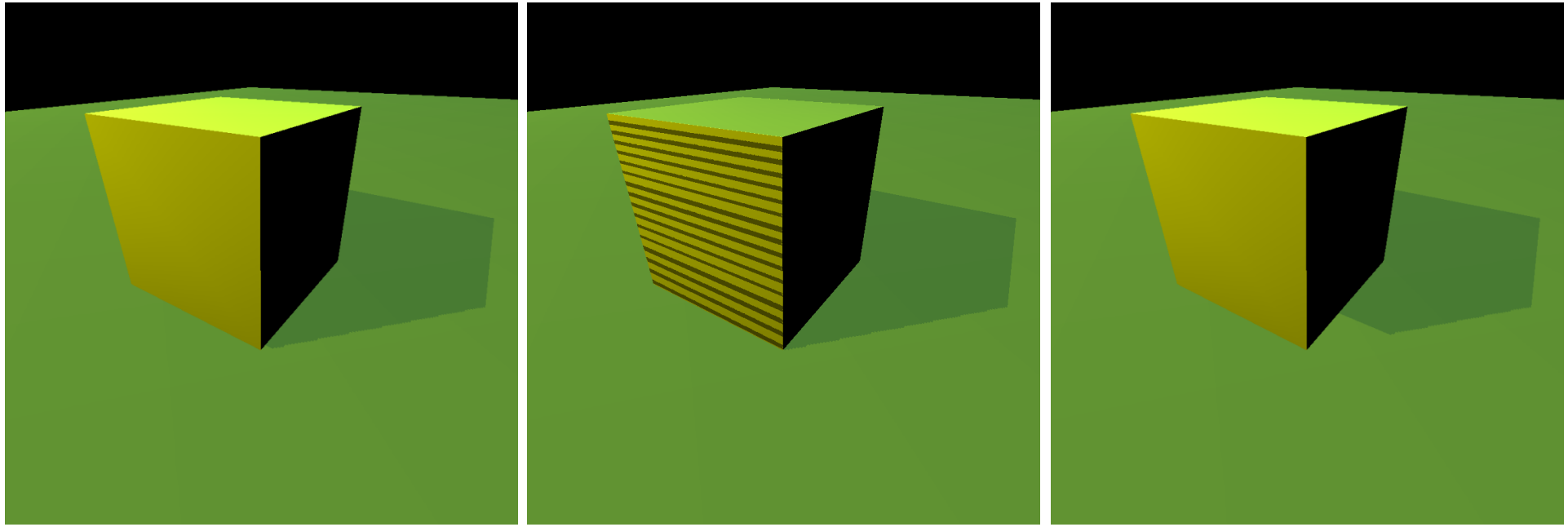


- http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/

2. Bias (Epsilon) for Shadow Maps

ShadowMap(x',y') + bias < z'

- Η επιλογή καλής τιμής για bias μπορεί να είναι πολύ tricky

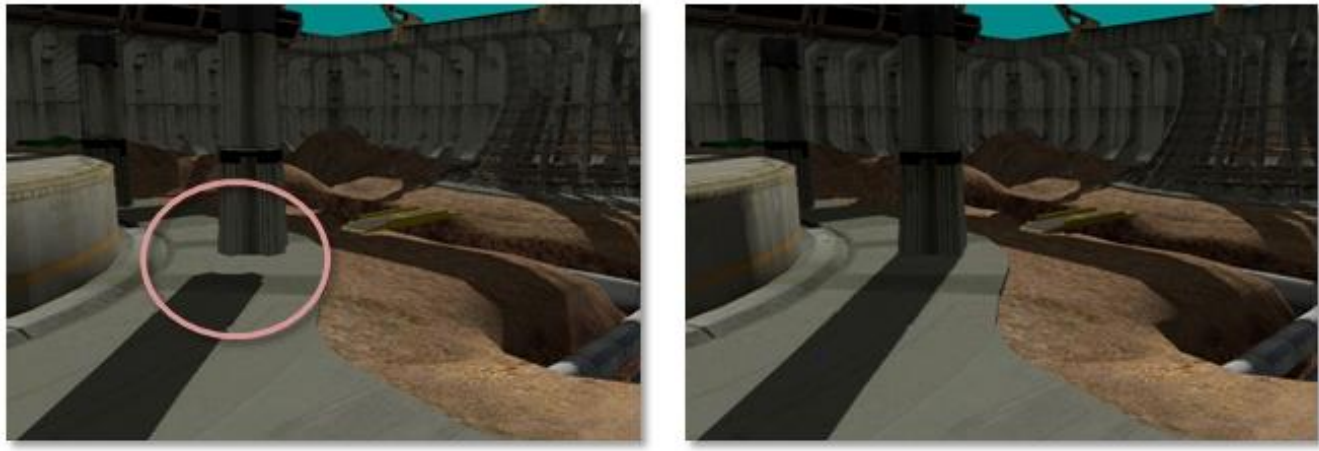


Σωστή εικόνα | Όχι αρκετό bias | Υπερβολικό bias

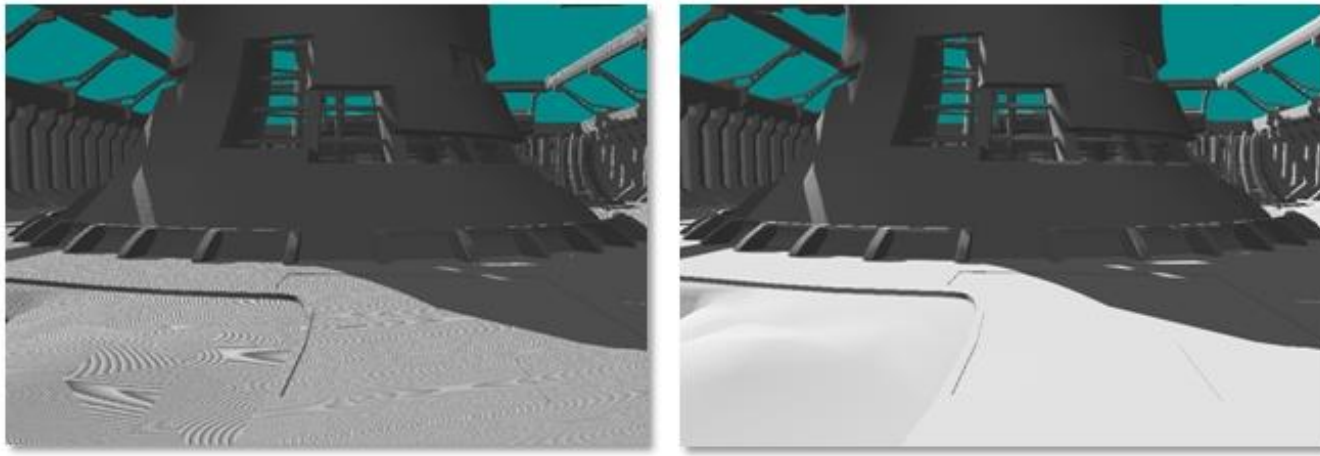
2. Bias (Epsilon) for Shadow Maps

2. Bias (Epsilon) for Shadow Maps

- Peter-panning: shadows incorrectly offset from objects (yes this term comes from the character Peter Pan)
- Shadow acne: precision aliasing
- They are slow. Why?
  - We are looking at a number of texels around a sample and taking the average.
  - You may already be thinking that we could do two 1D convolutions to speed up the naïve approach but we still have the issue of sampling overdraw
  - If only there was a better way…



Peter-panning

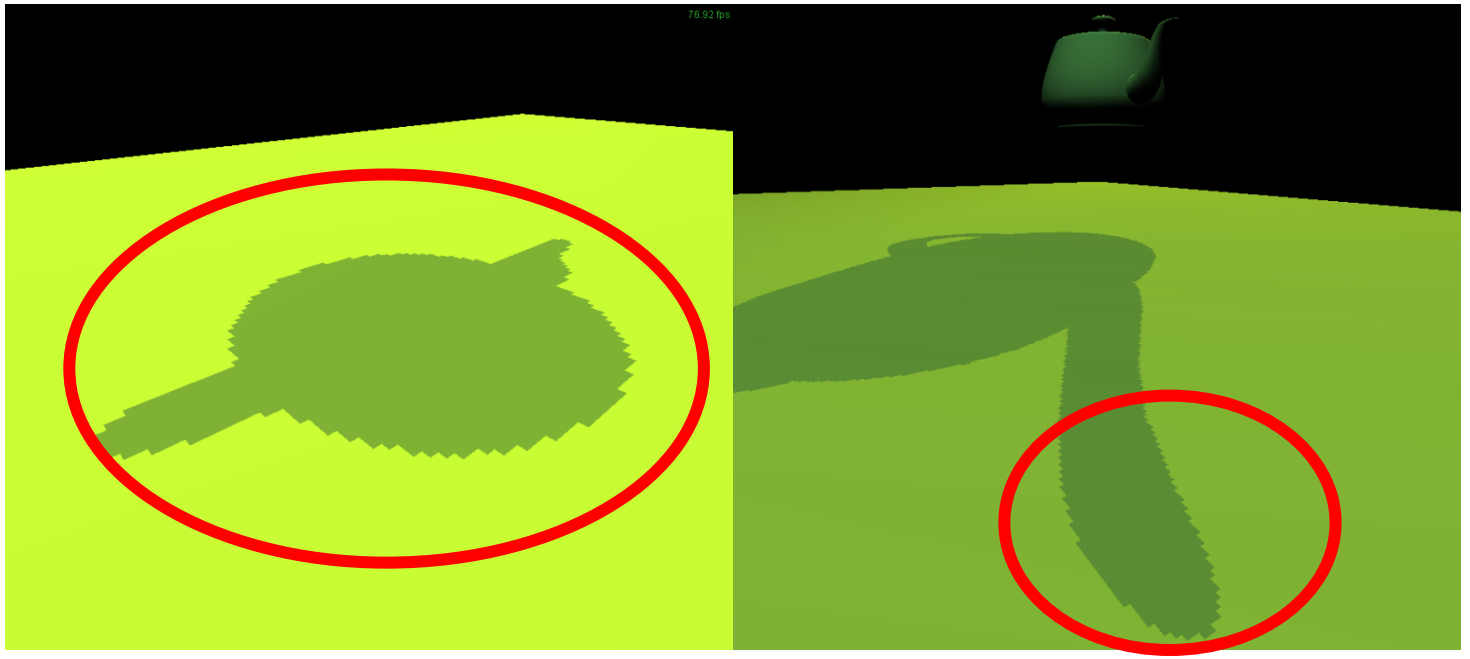


Shadow acne

https://docs.microsoft.com/en-us/windows/desktop/dxtecharts/common-techniques-to-improve-shadow-depth-maps

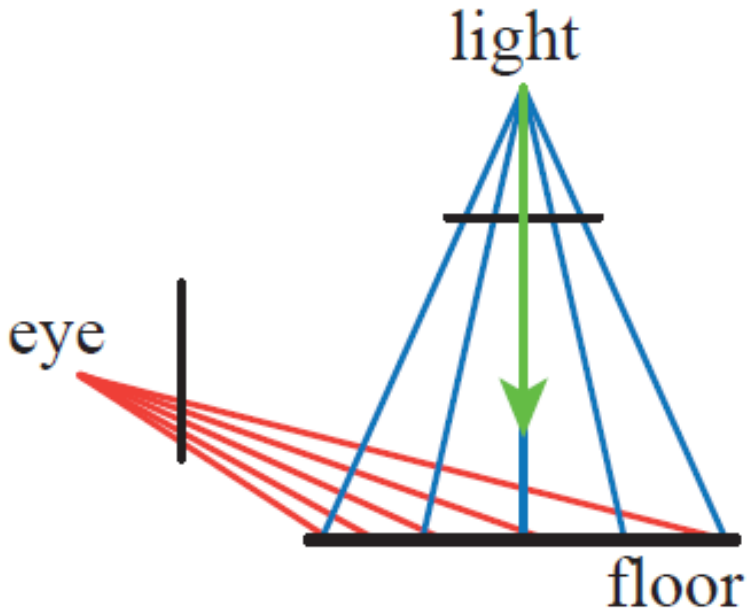
3. Ταύτιση Shadow Map

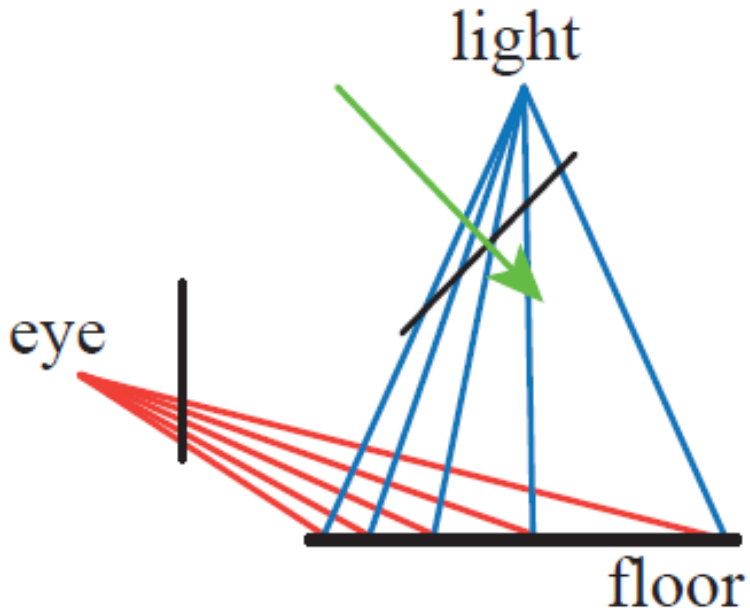
- Ανεπαρκής δειγματοληψία (Under-sampling) του shadow map
- Επαναπροβολή ταύτισης – πολύ κακό όταν η κάμερα και το φως βλέπουν το ένα προς το άλλο



Perpsective aliasing

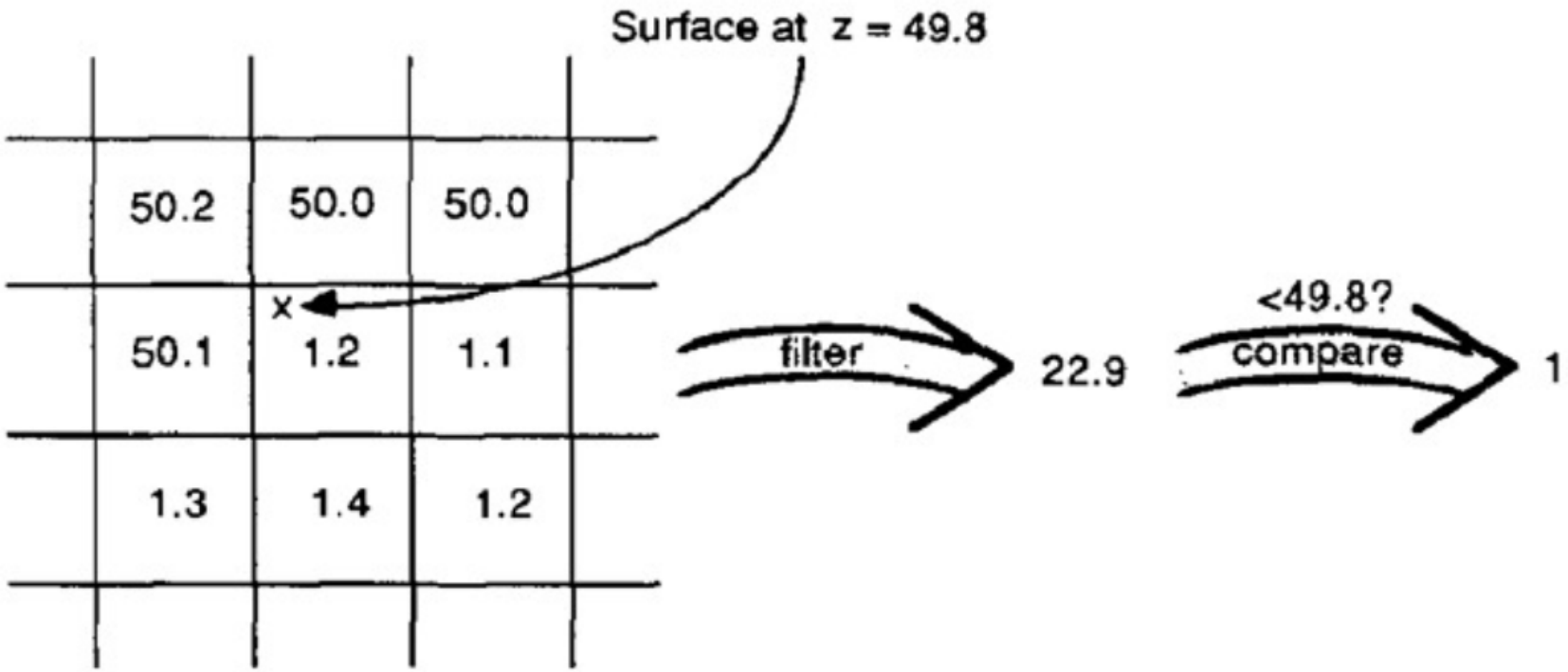
3. Ταύτιση Shadow Map





3. Shadow Map Filtering

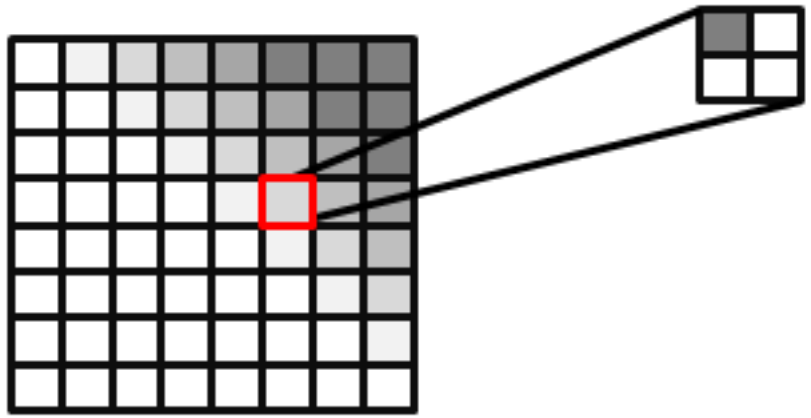
- Should we filter the depth?
  (weighted average of neighboring depth values)
- No... filtering depth is not meaningful

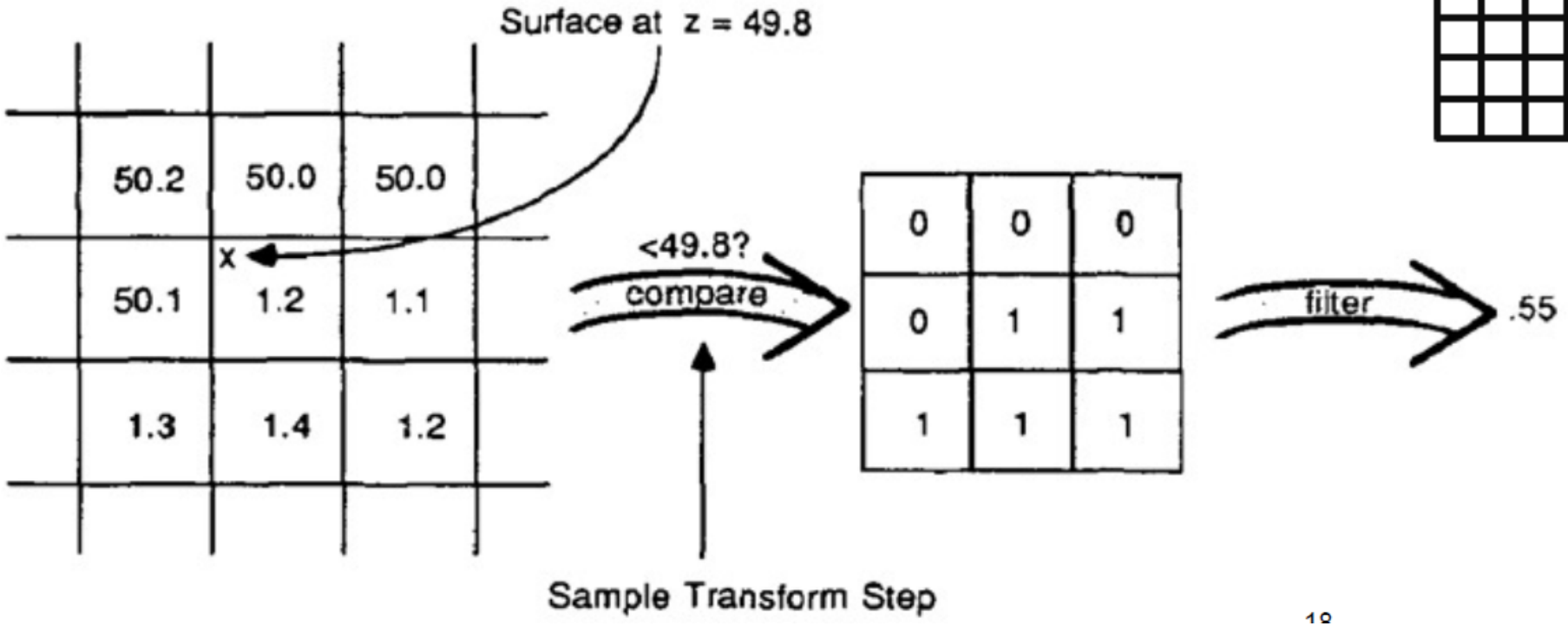


a) Ordinary texture map filtering. Does not work for depth maps. 17

3. Percentage Closer Filtering

- Instead we need to filter the *result* of the shadow test (weighted average of comparison results)



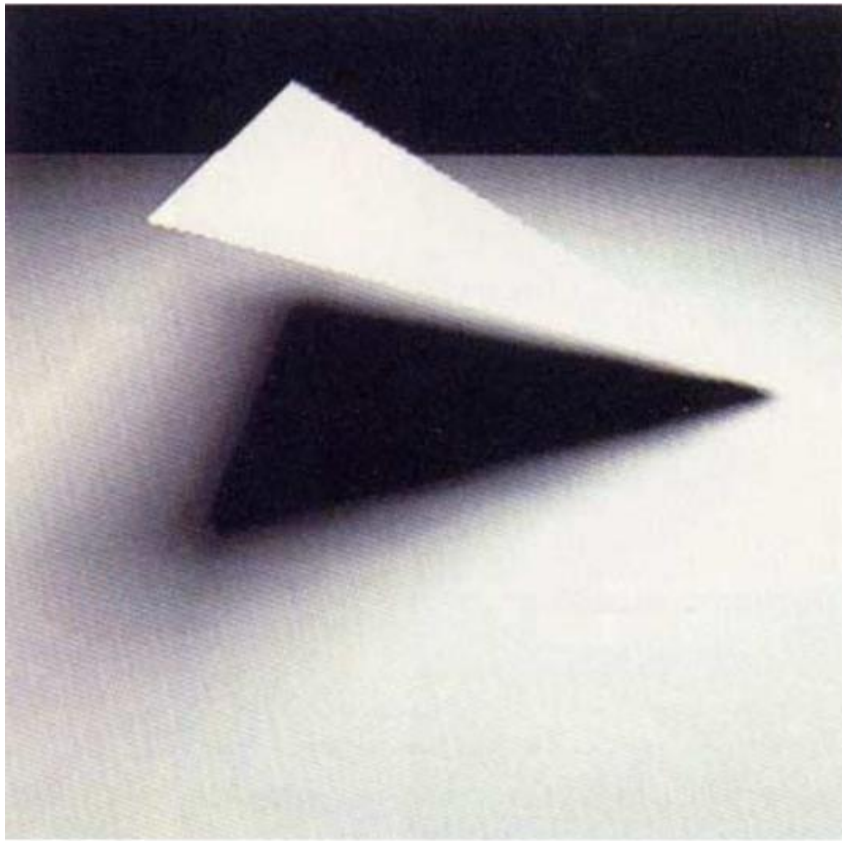


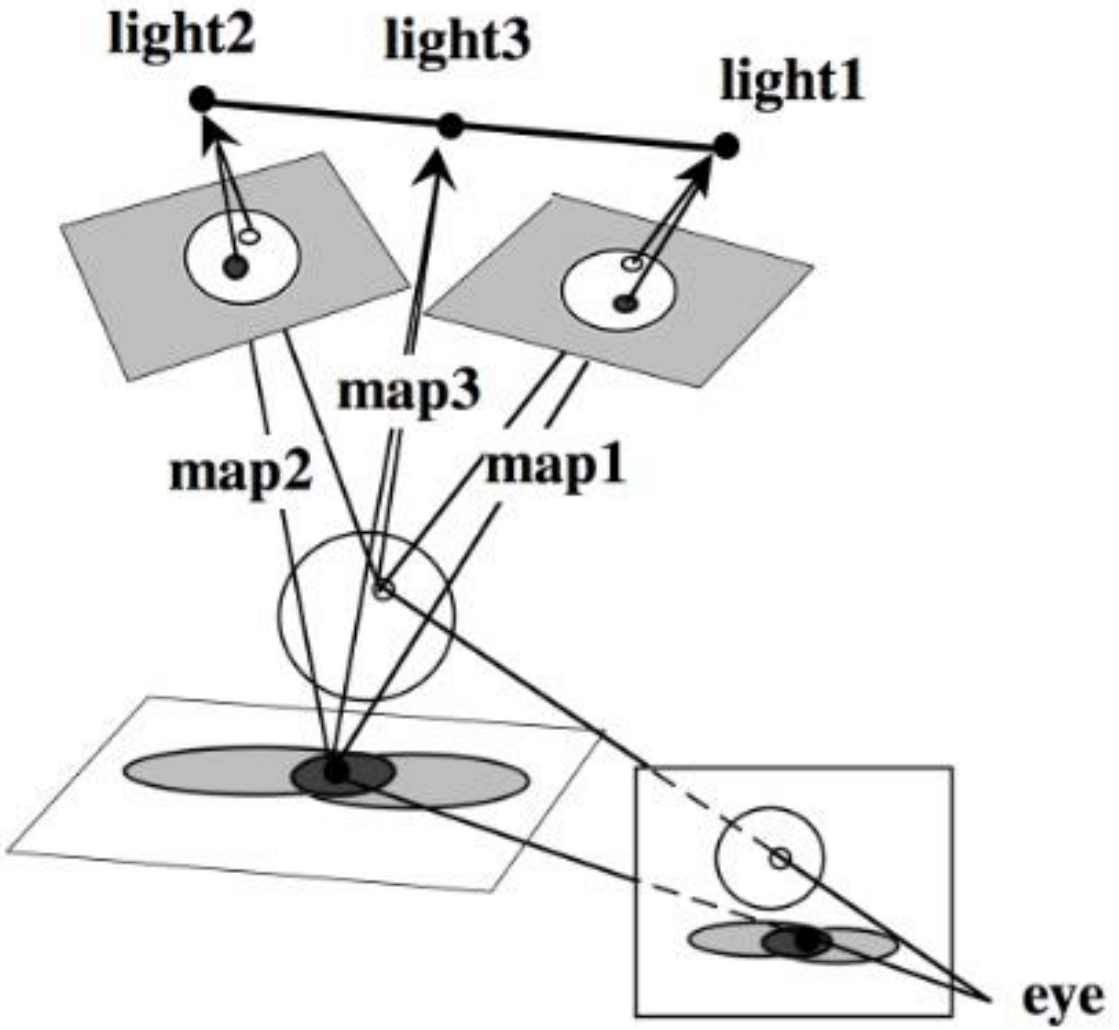
3. Percentage Closer Filtering

- 5x5 samples
- Nice antialiased shadow
- Using a bigger filter produces fake soft shadows
- Setting bias is tricky

Shadow Mapping

- **Soft shadows**
  - One shadow map per light sample





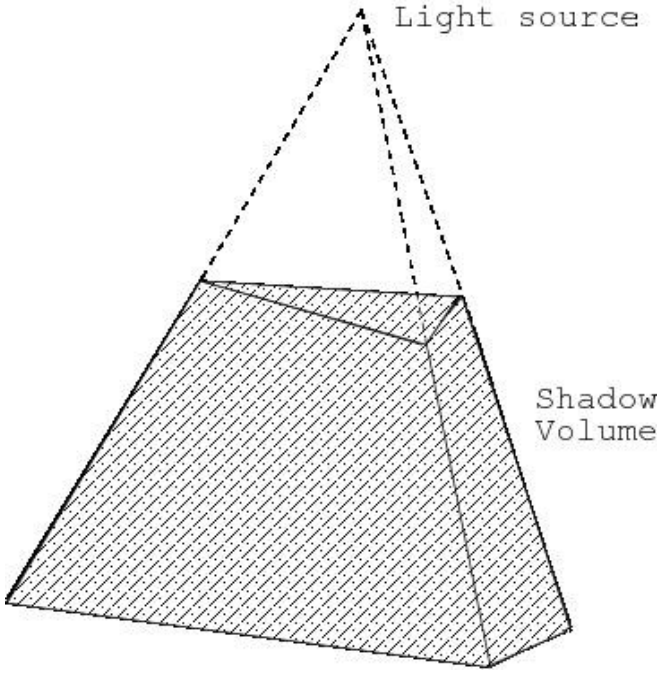
Chen, Shenchang Eric, and Lance Williams. "View interpolation for image synthesis." In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 279-288. ACM, 1993.

Shadow Volumes

Some slides by Mark Kilgard

Shadow Volumes: *Βασική αρχή*

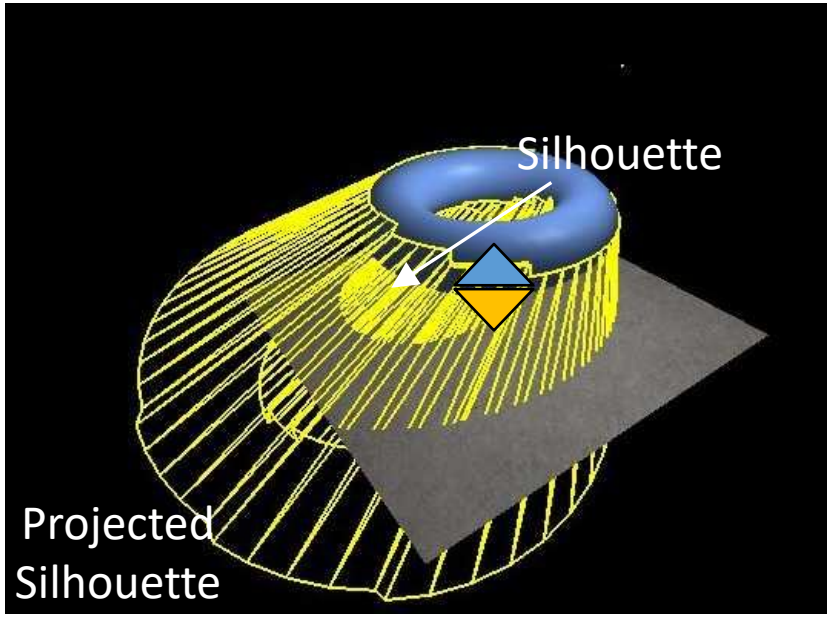
- **Shadow volume** is a technique used in 3D computer graphics to add shadows to a rendered scene.
  - It is the geometry describing the 3D shape of the region occluded from a light source.
  - A shadow volume divides the virtual world in two: areas that are in shadow and areas that are not.
- The stencil buffer implementation of shadow volumes is generally considered among the most practical general purpose real-time shadowing techniques for use on modern 3D graphics hardware (see later notes).



Shadow Volumes: *Βασική αρχή*

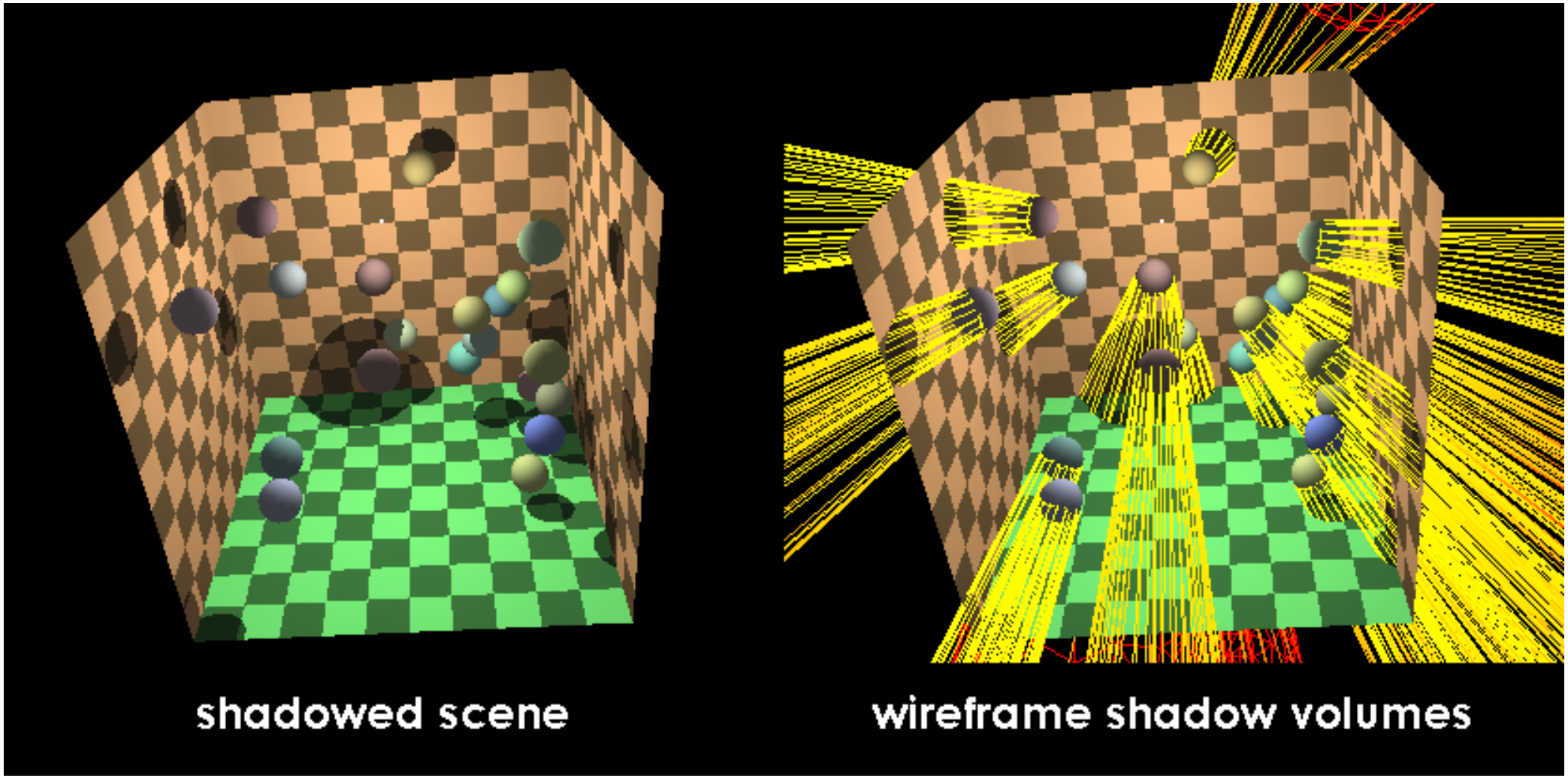
- For each light + object pair, compute mesh enclosing area where the object occludes the light
  - Find silhouette from light's perspective
    - Includes every edge shared by two triangles, such that one triangle faces light source and other faces away – marks the transition from visible to not visible by the light
    - On torus, where angle between normal vector and vector to light becomes >90°
  - Project silhouette along light rays
  - Generate triangles bridging silhouette and its projection to obtain the **shadow volume**
- A point *P* is in shadow from light *L* if any shadow volume *V* computed for *L* contains *P*
  - Can determine this quickly using multiple passes and a **"stencil buffer"**

Stencil Shadows in Doom 3



http://www.ozone3d.net/tutorials/stencil_shadow_volumes.php

Shadow Volumes: *Βασική αρχή*



Using Shadow Volumes to Render Shadows

Two stages

- Compute the shadow volume formed by a light source and a set of shadowing objects.
- Every triangle produces a shadow volume
- Check whether the point is inside / outside the shadow volume
  - inside → shadowed
  - Outside → illuminated by light source

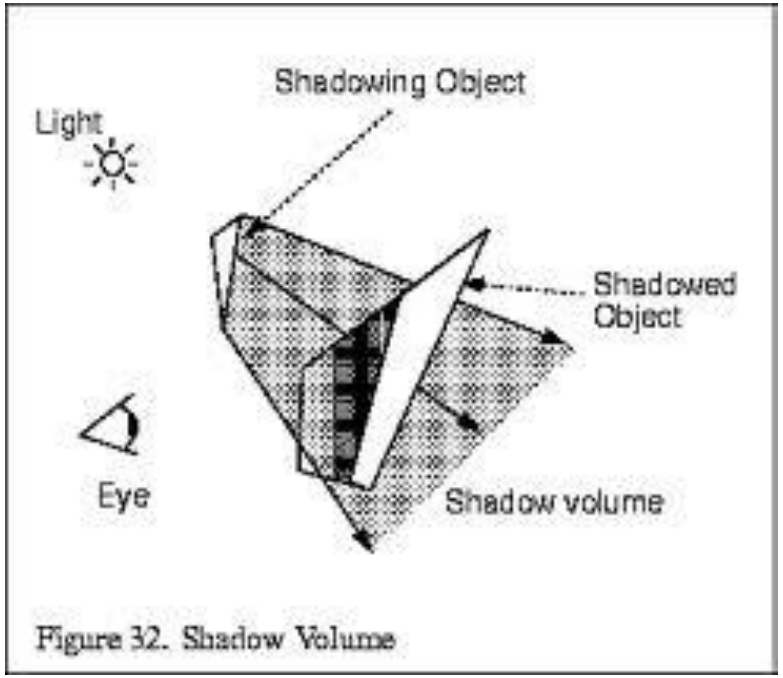


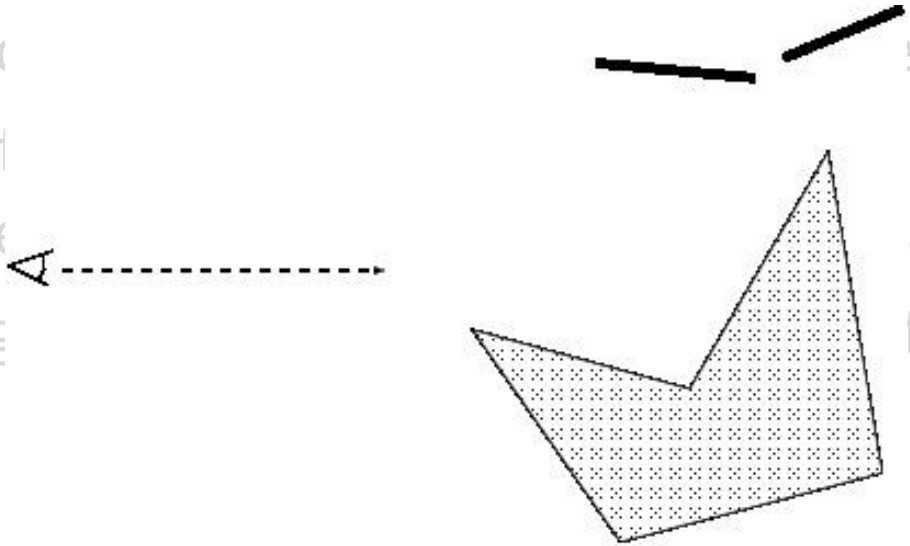
Figure 32. Shadow Volume

Shadow Volumes: *Stencil Buffer*

- The Stencil Buffer is another frame buffer, like the Color Buffer, Depth Buffer and Accumulation Buffer.
- Stencil Buffer can be used to specify a pattern so that only fragments that pass the stencil test are rendered to the buffer.

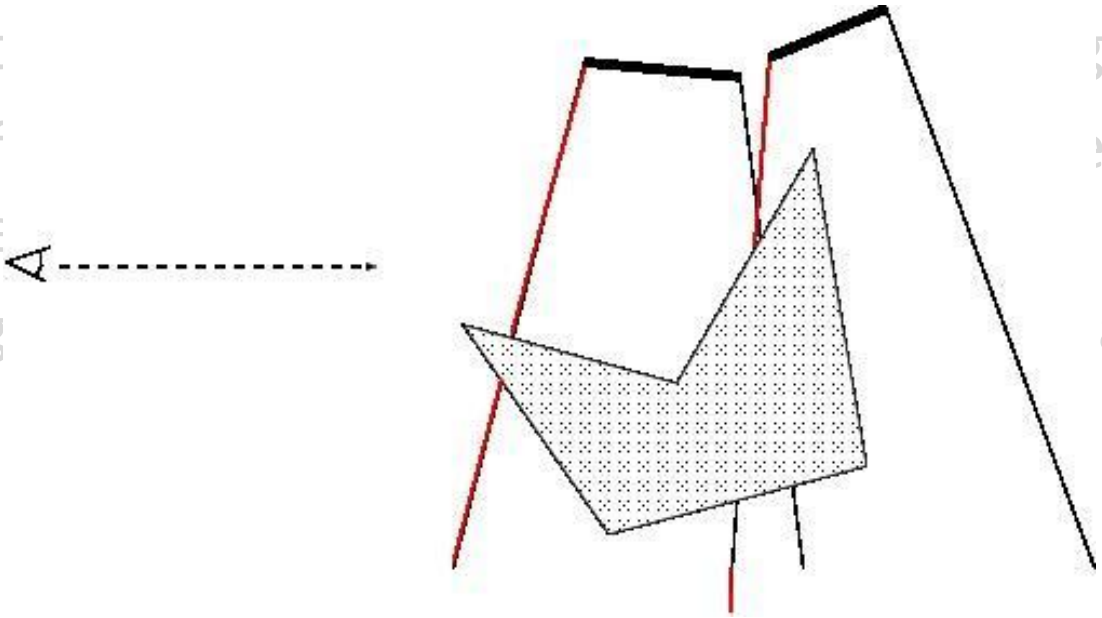
Shadow volume: *Doing this using a stencil buffer*

- Render the scene with ambient light
- Clear the stencil buffer, and render the shadow volume with the colour buffer off and back face culling on
- Whenever a rendered fragment of the shadow volume is closer than the depth of the other objects, increment the stencil value for that pixel
- Turn on the front face culling
- Whenever a rendered is closer than the depth of the other objects, at pixel
- Render the scene using for the area that the stencil value is 0

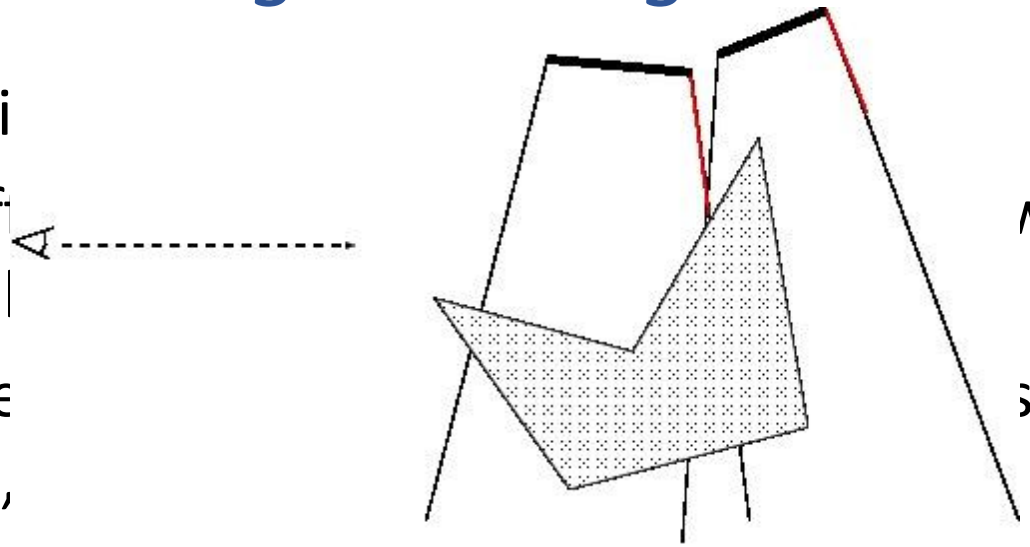


Shadow volume: *Doing this using a stencil buffer*

- Render the scene with ambient light
- Clear the stencil buffer, and render the shadow volume with the colour buffer off and back face culling on
- Whenever a rendered fragment of the shadow volume is closer than the depth of the other objects, increment the stencil value for that pixel
- Turn on the front face c
- Whenever a rendered f
  of the other objects, de
- Render the scene using
  stencil value is 0

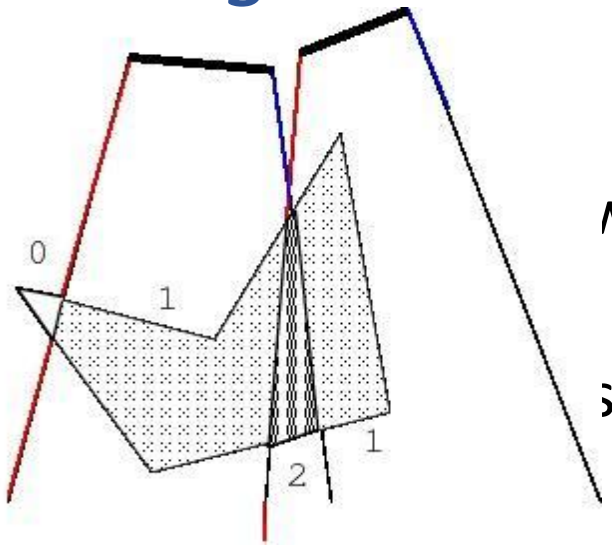


Shadow volume: *Doing this using a stencil buffer*



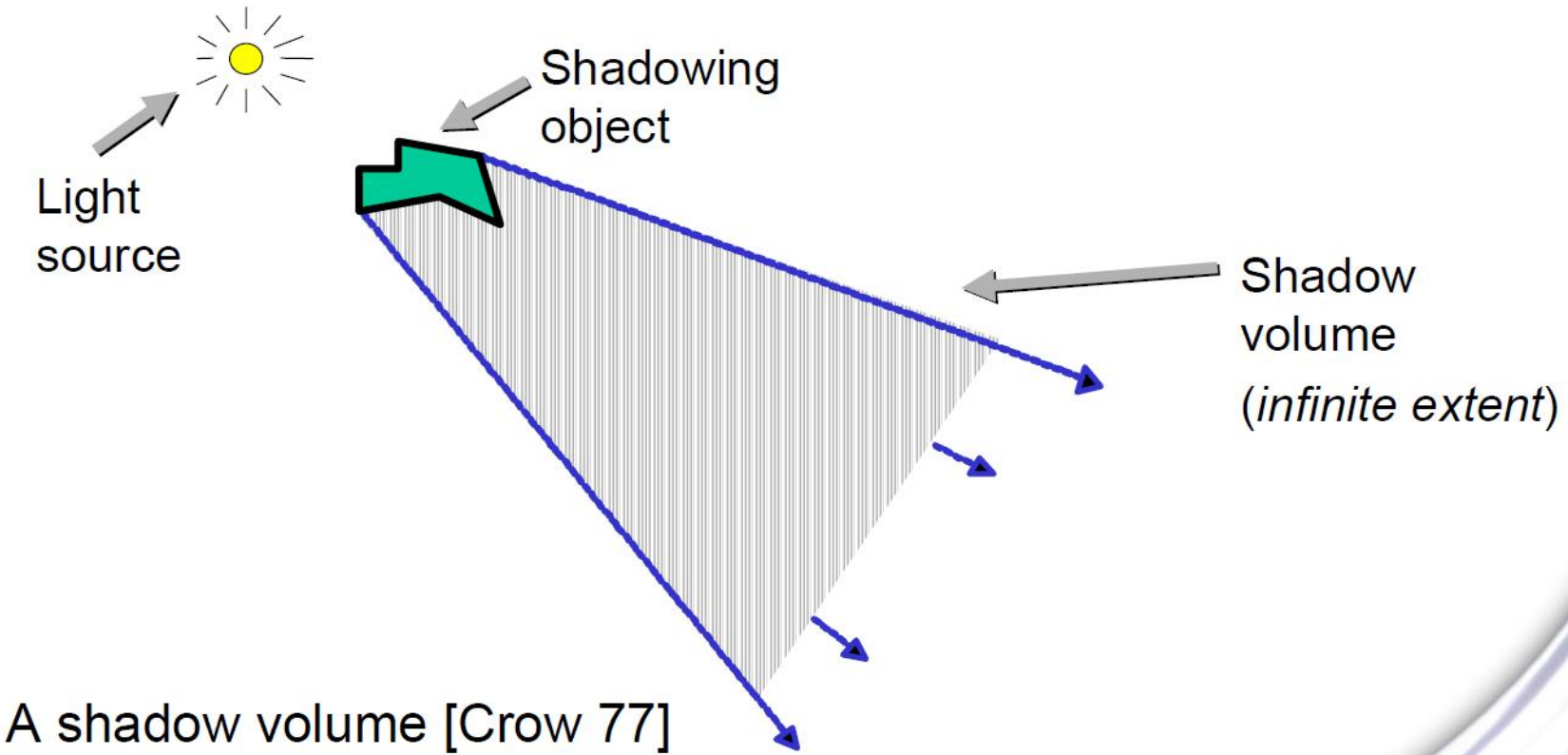
- Render the scene wi
- Clear the stencil buf ith the colour buffer off and back face cul
- Whenever a rendere s closer than the depth of the other objects, pixel
- Turn on the front face culling and turn off the back face culling
- Whenever a rendered fragment of the shadow volume is closer than the depth of the other objects, decrement the stencil value for that pixel
- Render the scene using diffuse and specular reflection for the area that the stencil value is 0

Shadow volume: *Doing this using a stencil buffer*



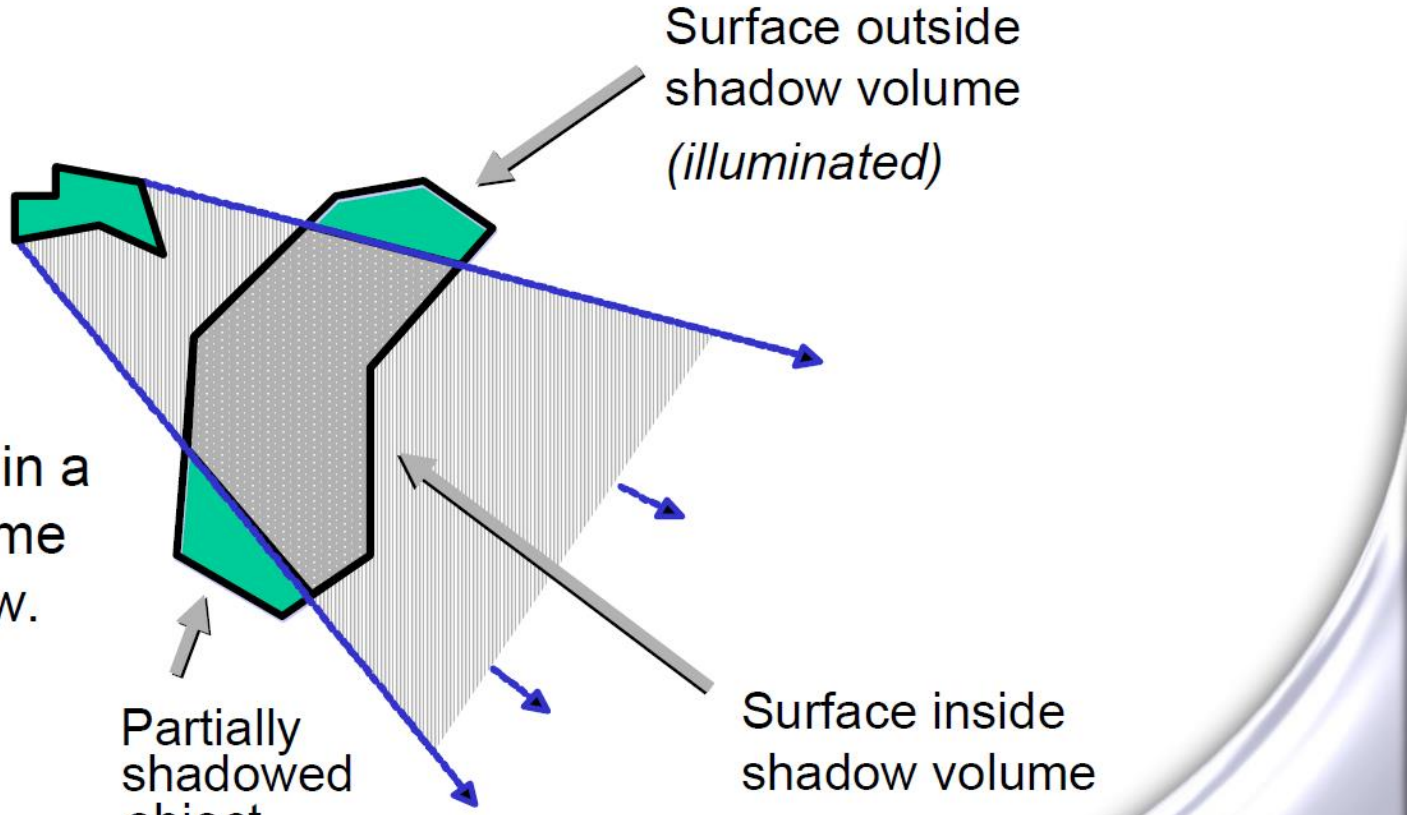
- Render the scene wi
- Clear the stencil buf ... with the colour buffer off and back face cul
- Whenever a rendere ... s closer than the depth of the other objects, ... pixel
- Turn on the front face culling and turn off the back face culling
- Whenever a rendered fragment of the shadow volume is closer than the depth of the other objects, decrement the stencil value for that pixel
- Render the scene using diffuse and specular reflection for the area that the stencil value is 0

Shadow Volume Basics

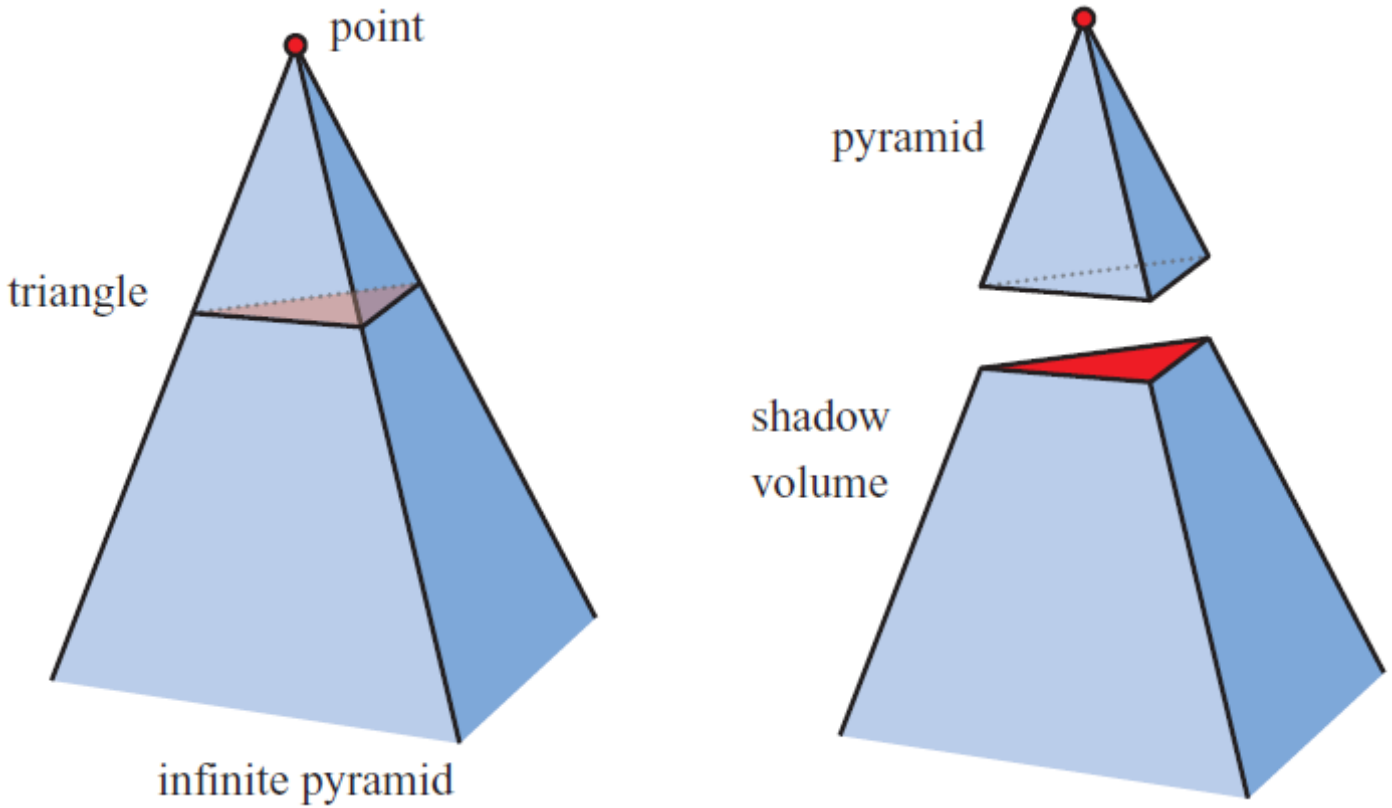


A shadow volume [Crow 77]
is simply the half-space defined
by a light source and a shadowing object

Shadow Volume Basics (2)



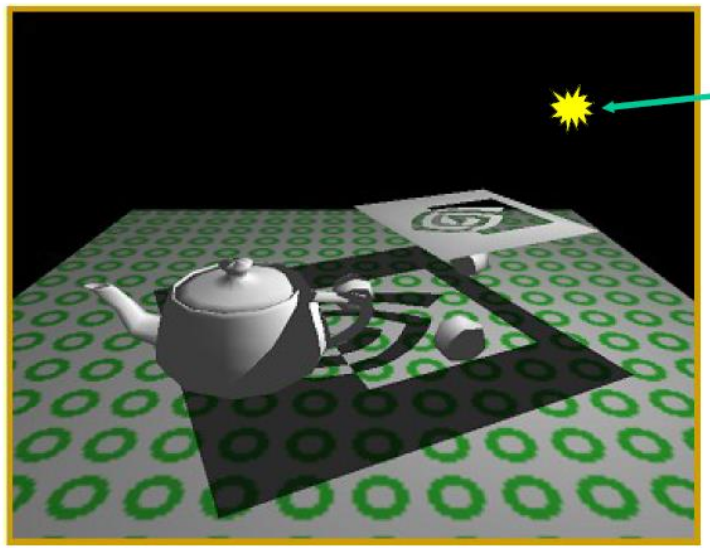
Shadow Volume of a triangle

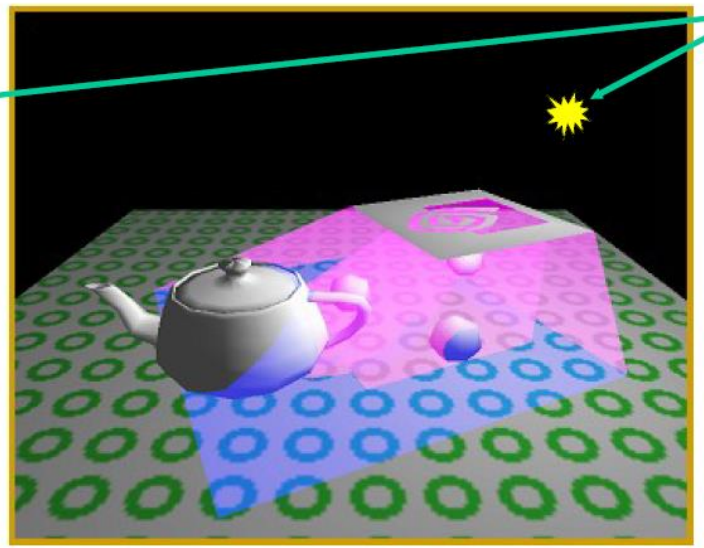


Visualizing Shadow Volumes in 3D

SIGGRAPH 2004

- Occluders and light source cast out a shadow volume
  - Objects within the volume should be shadowed





Scene with shadows from an NVIDIA logo casting a shadow volume

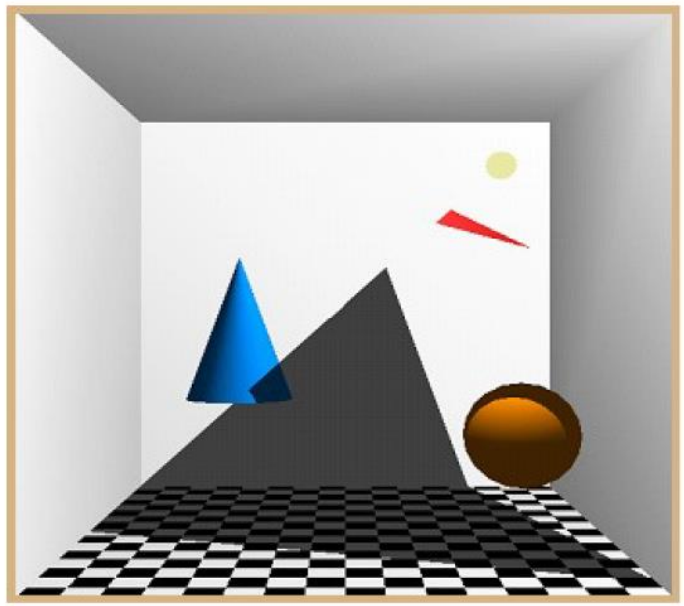
Visualization of the shadow volume

NVIDIA

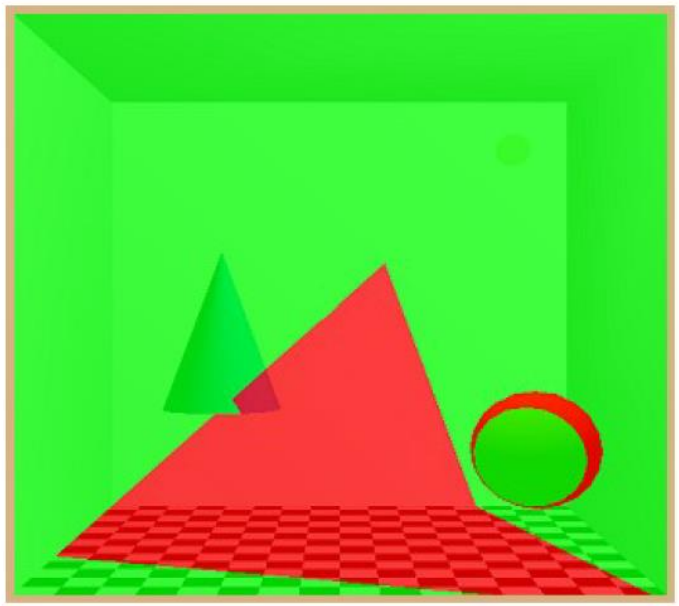
Visualizing the Stencil Buffer Counts

SIGGRAPH 2004

Shadowed scene



Stencil buffer contents



Stencil counts beyond 1 are possible for multiple or complex occluders.

red = stencil value of 1
green = stencil value of 0

GLUT *shadowvol* example credit: Tom McReynolds

NVIDIA

Nested Shadow Volumes: Stencil Counts Beyond One

SIGGRAPH 2004

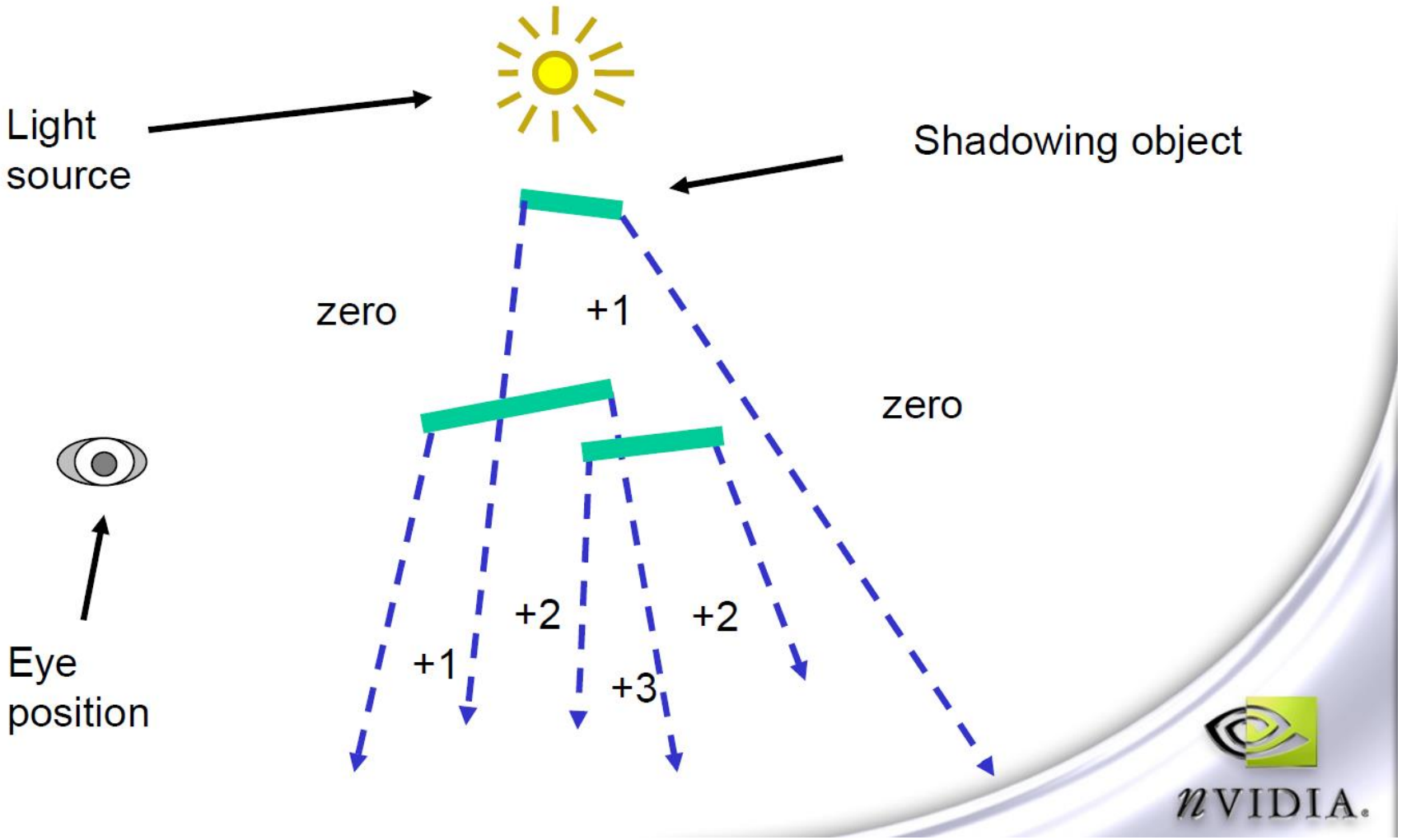
Shadowed scene

Stencil buffer contents

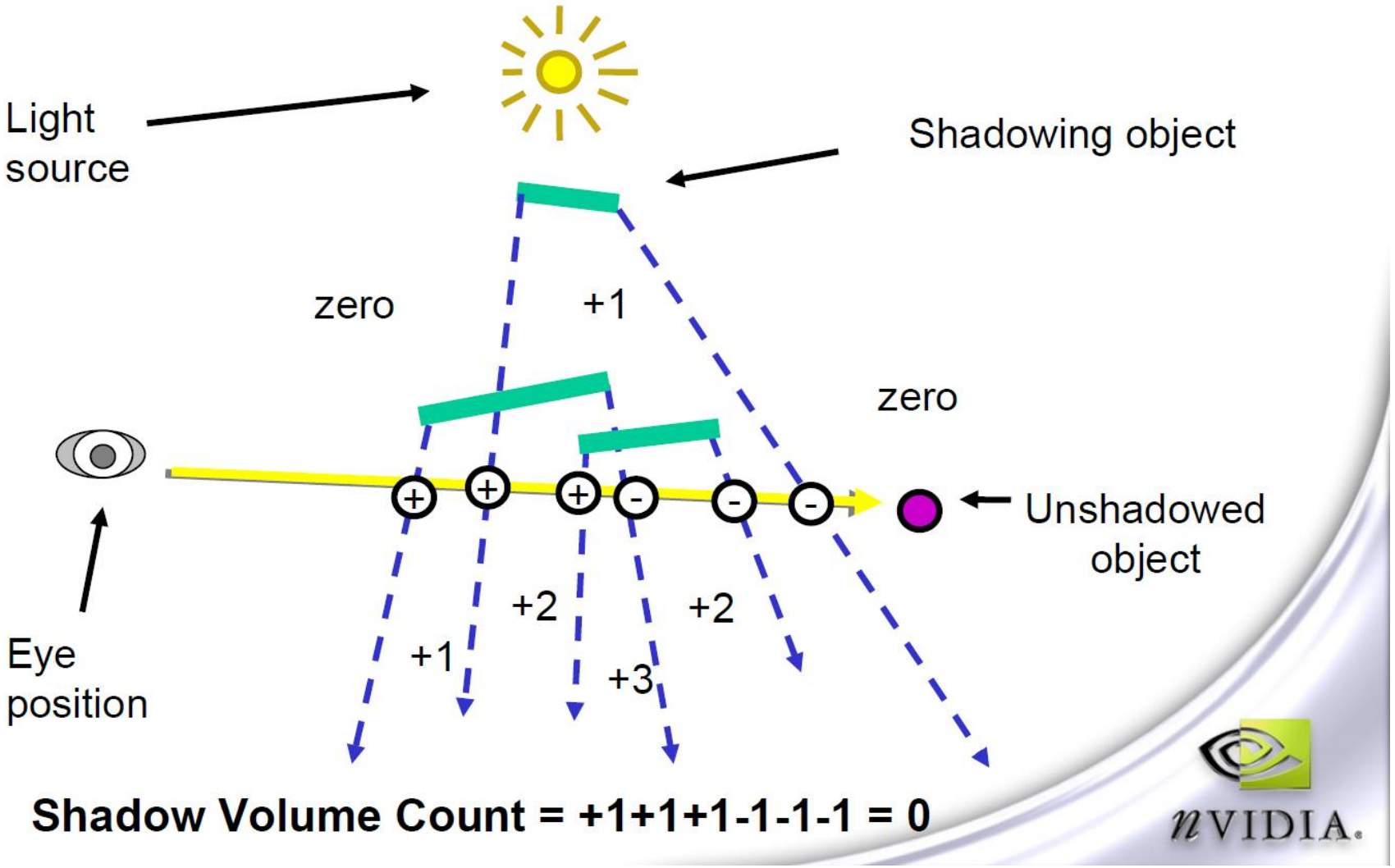
green = stencil value of 0
red = stencil value of 1
darker reds = stencil value > 1

NVIDIA.

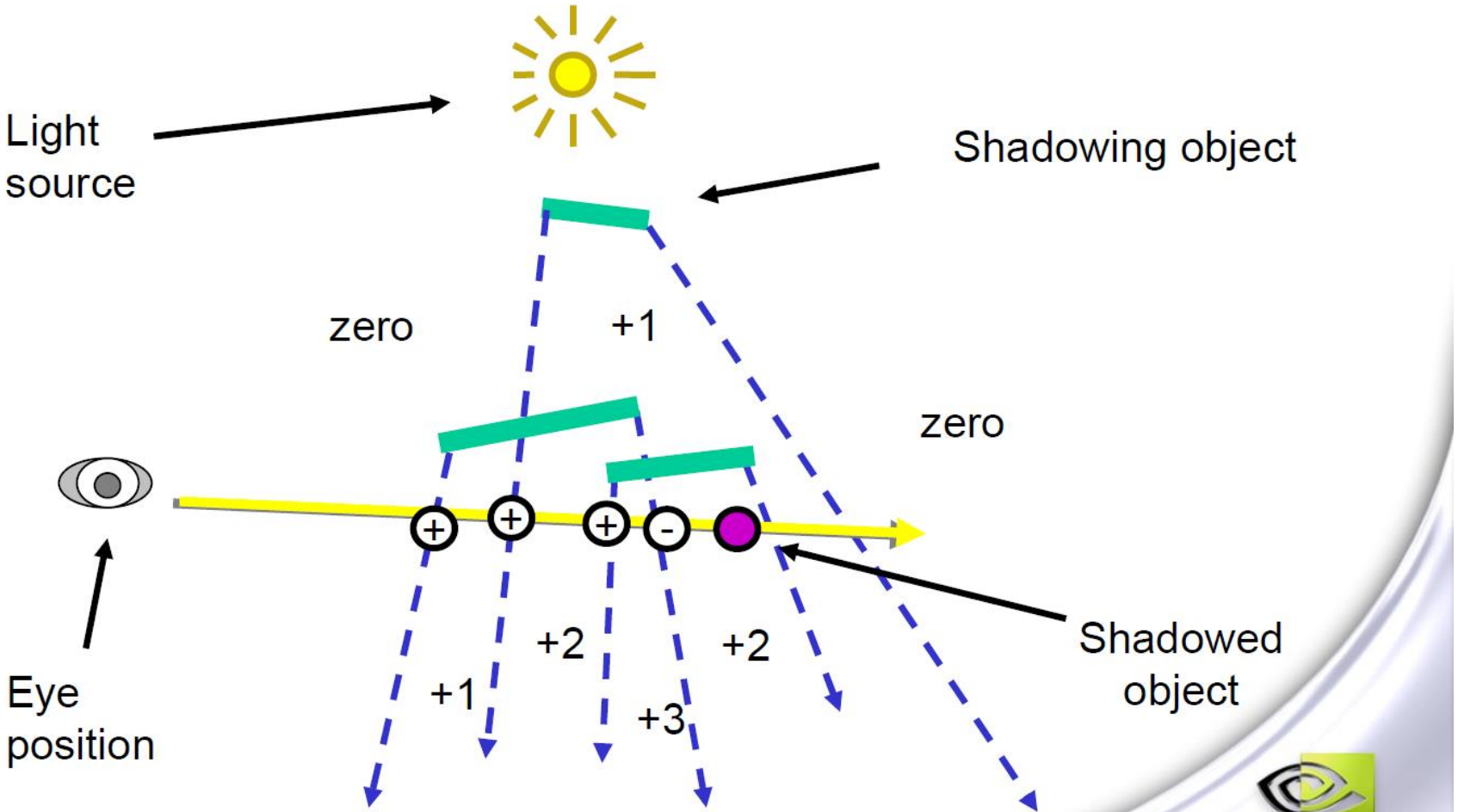
Why Eye-to-Object Stencil Counting Approach Works



Illuminated, Behind Shadow Volumes (*Two-pass Zpass*)

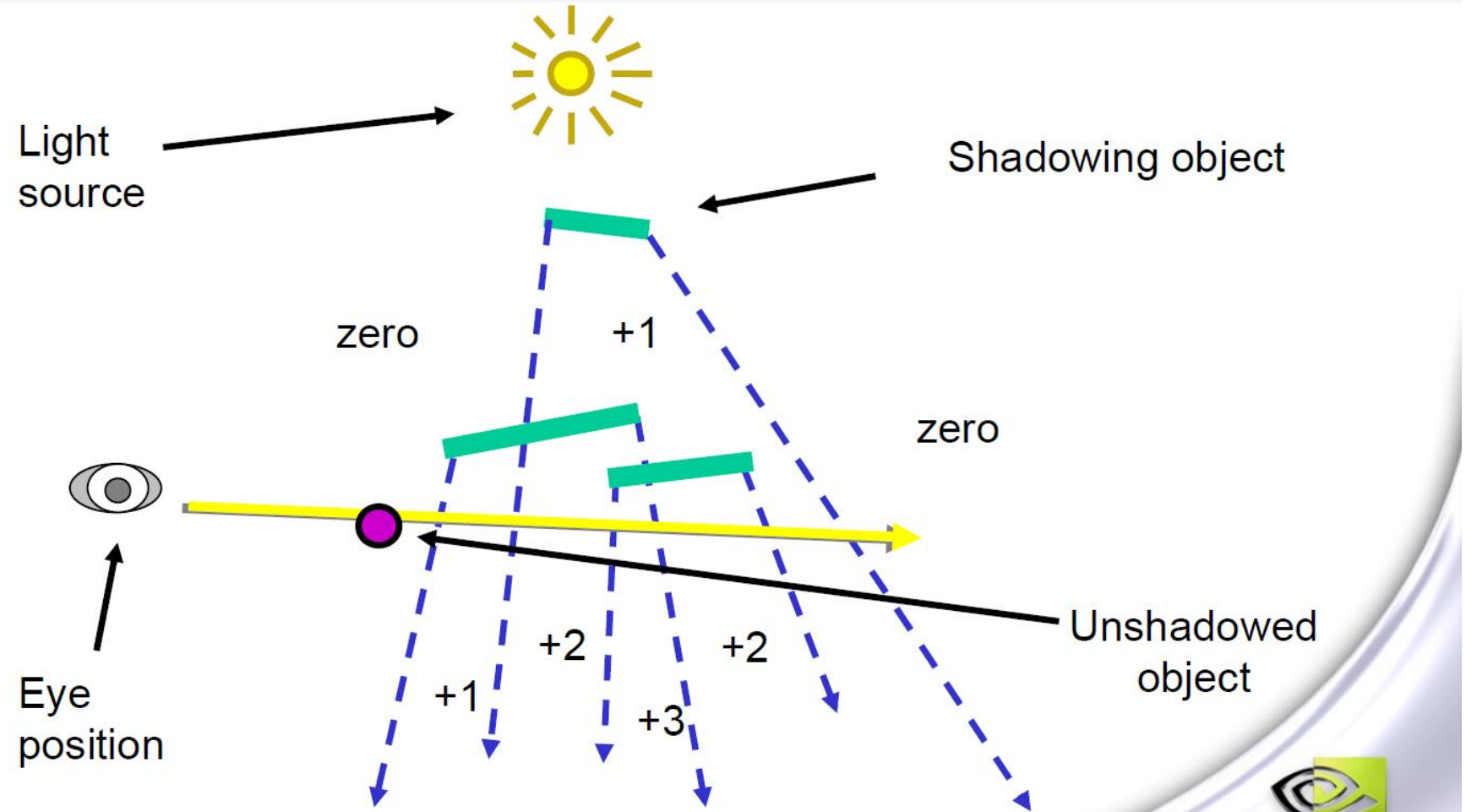


Shadowed, Nested in Shadow Volumes (*Two-pass Zpass*)



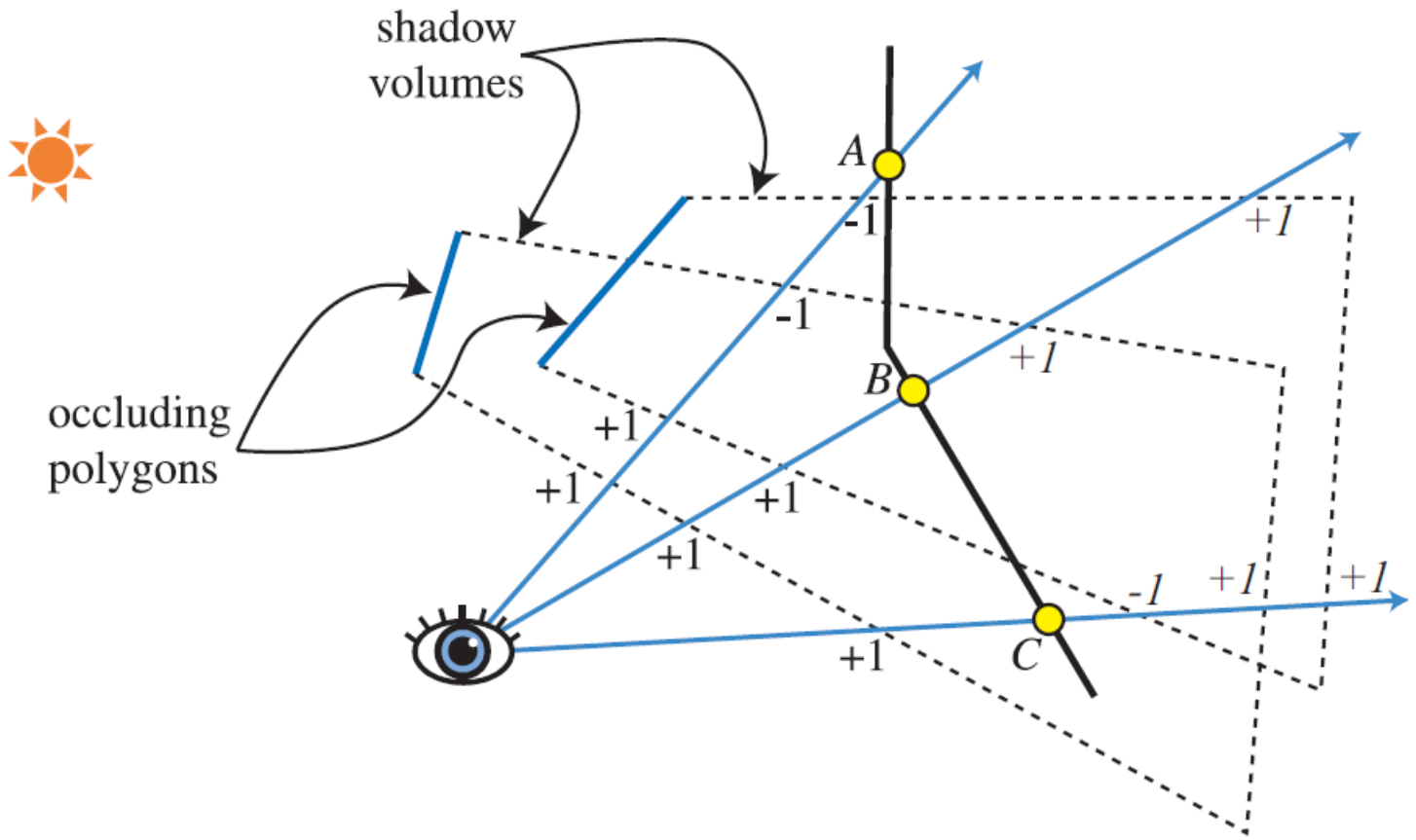
Shadow Volume Count = +1+1+1-1 = 2

Illuminated, In Front of Shadow Volumes (*Two-pass Zpass*)

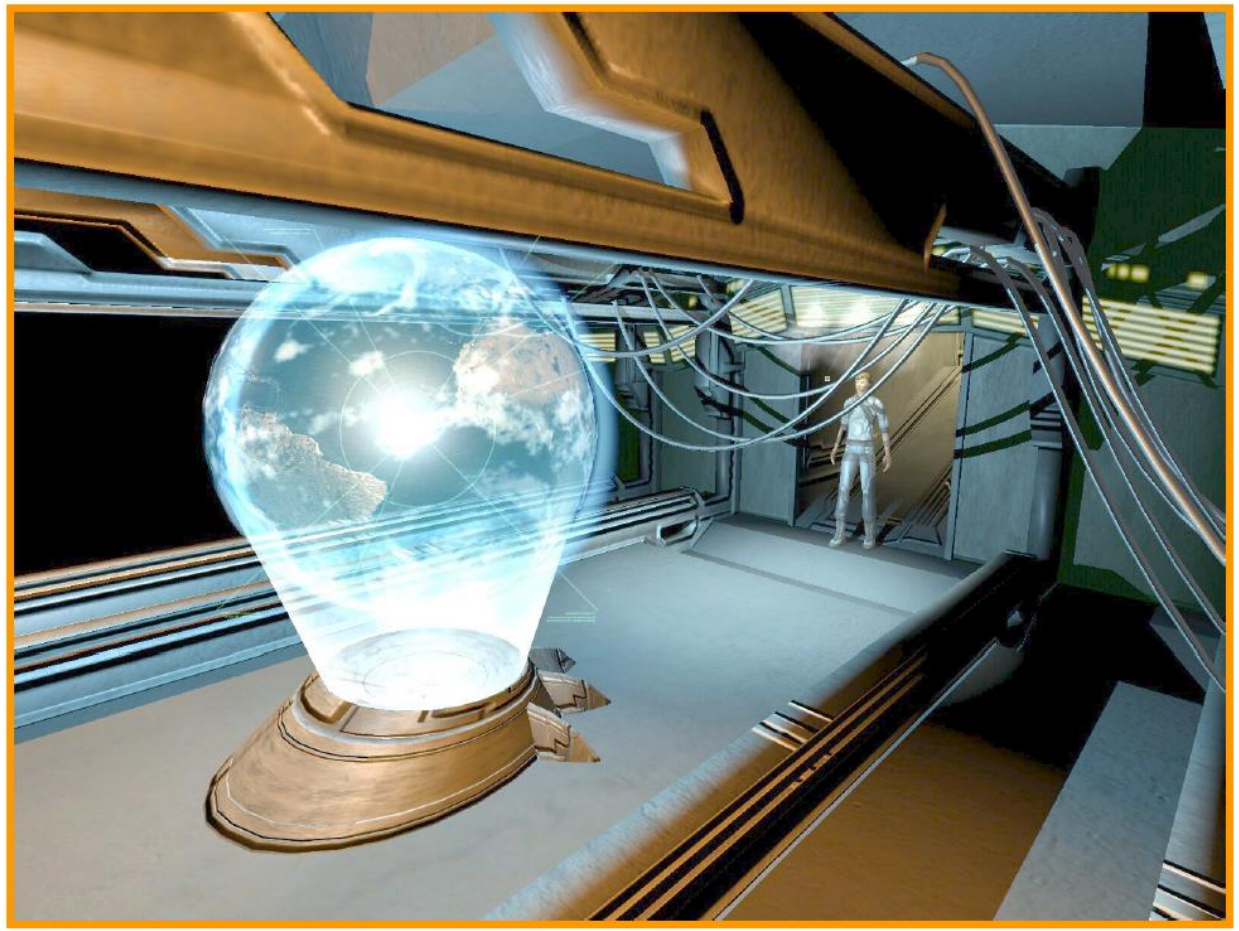


Shadow Volume Count = 0 (no depth tests pass)

Shadow Volume



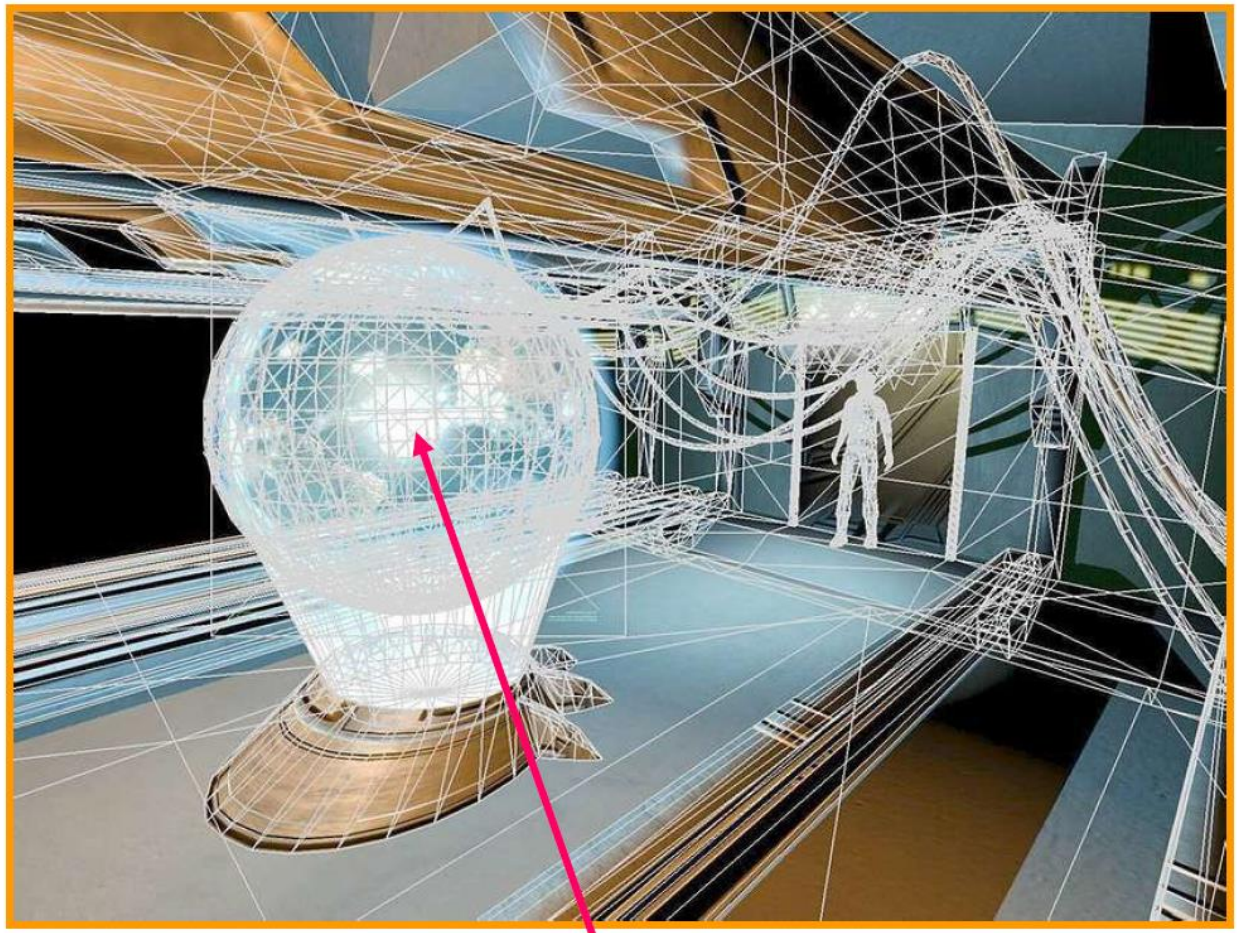
Shadows in a Real Game Scene



Abducted game images courtesy Joe Riedel of Contraband Entertainment

Scene's *Visible* Geometric Complexity

SIGGRAPH 2004

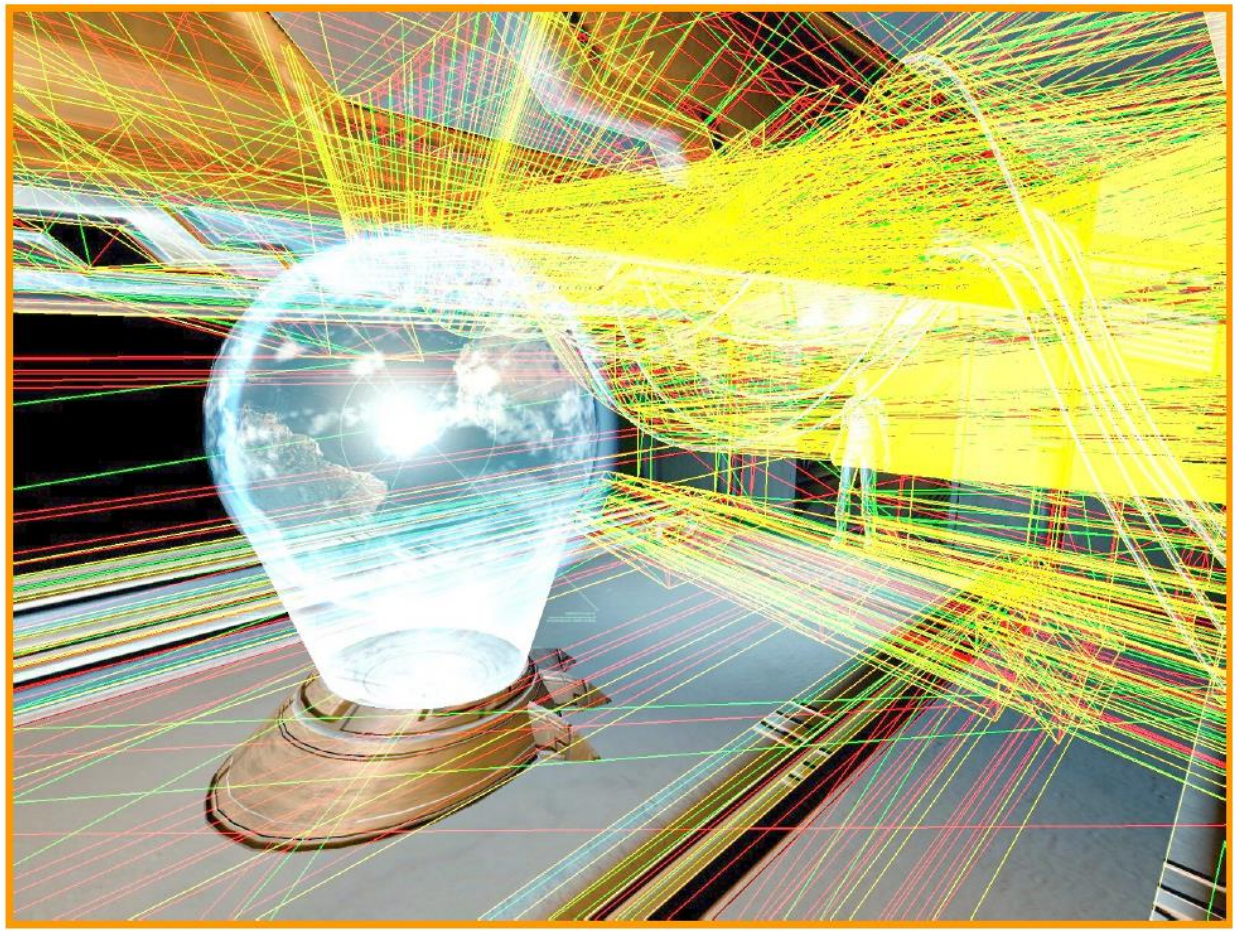


Wireframe shows geometric complexity of visible geometry

NVIDIA.

Scene's *Shadow Volume* Geometric Complexity

SIGGRAPH 2004



Wireframe shows geometric complexity of shadow volume geometry

Shadow volume geometry projects away from the light source

NVIDIA.

Visible Geometry versus Shadow Volume Geometry

SIGGRAPH 2004

<

Visible geometry

Shadow volume geometry

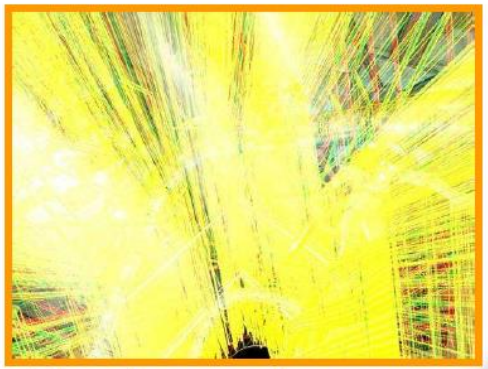
Typically, shadow volumes generate considerably more pixel updates than visible geometry

NVIDIA.

Other Example Scenes (1 of 2)

Dramatic chase scene with shadows

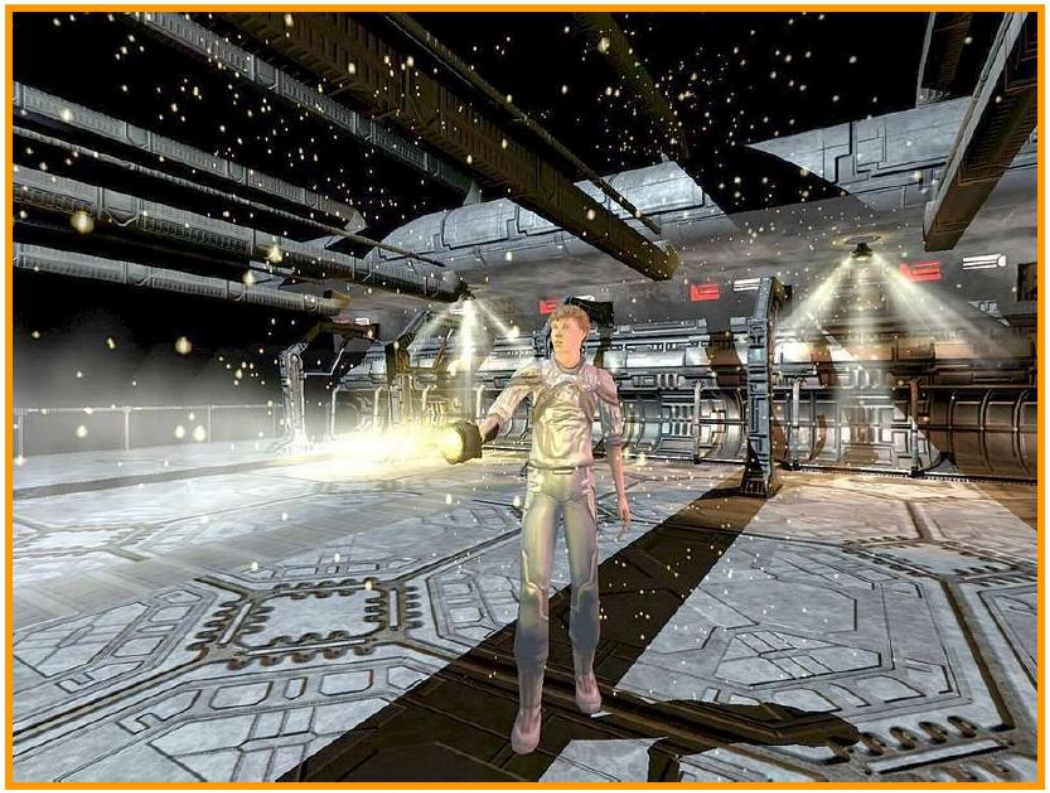
Visible geometry



Shadow volume geometry

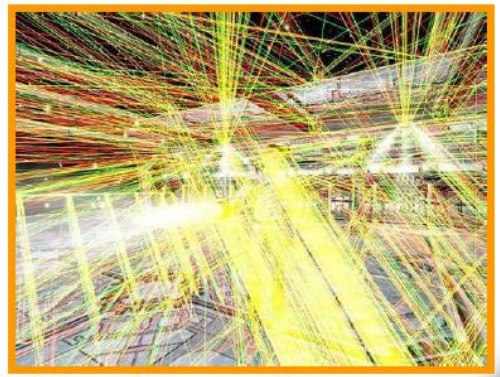
Abducted game images courtesy
Joe Riedel at Contraband Entertainment

Other Example Scenes (2 of 2)



Scene with multiple light sources

Visible geometry

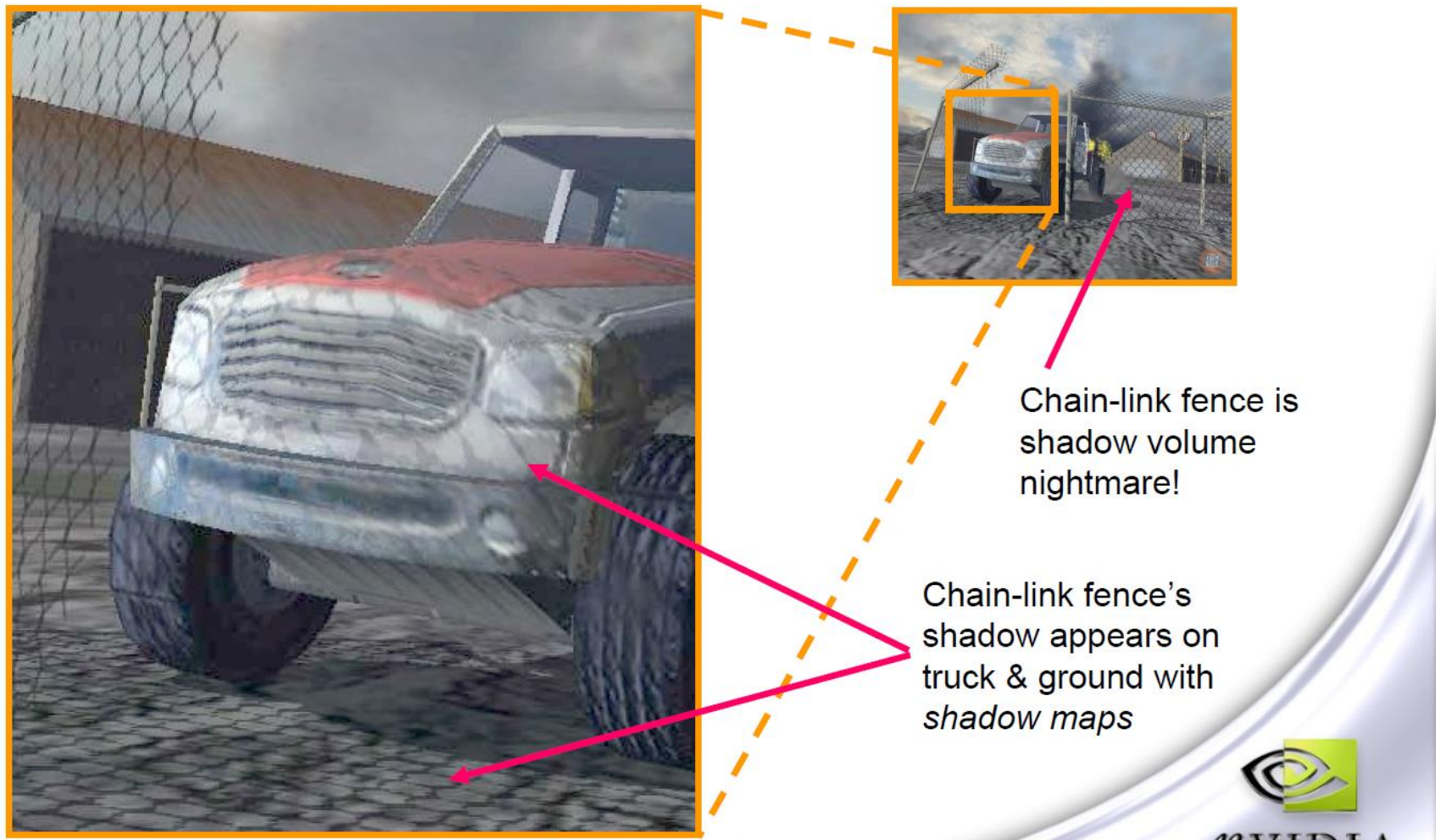


Shadow volume geometry

Abducted game images courtesy
Joe Riedel at Contraband Entertainment

When Shadow Volumes Are Too Expensive

SIGGRAPH 2004



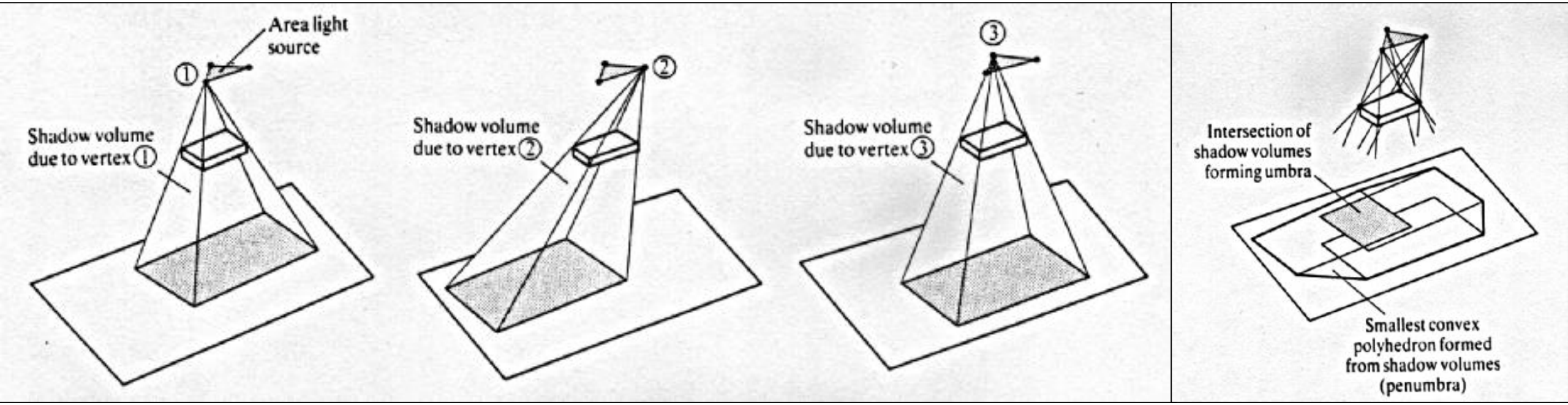
Chain-link fence is shadow volume nightmare!

Chain-link fence's shadow appears on truck & ground with *shadow maps*

NVIDIA.

Fuel game image courtesy Nathan d'Obrenan at Firetoad Software

Shadow Volumes and Area Lights



Advantage / Disadvantages of Shadow Volume

- Advantage
  - Do not need to manually specify the shadowed objects
  - The occluder can shadow itself
  - High precision

- Disadvantage
  - Bottleneck at the rasterizer -> quite slow
  - Many shadow volumes covering many pixels
  - Only hard shadows

Shadow Volumes vs Shadow Maps

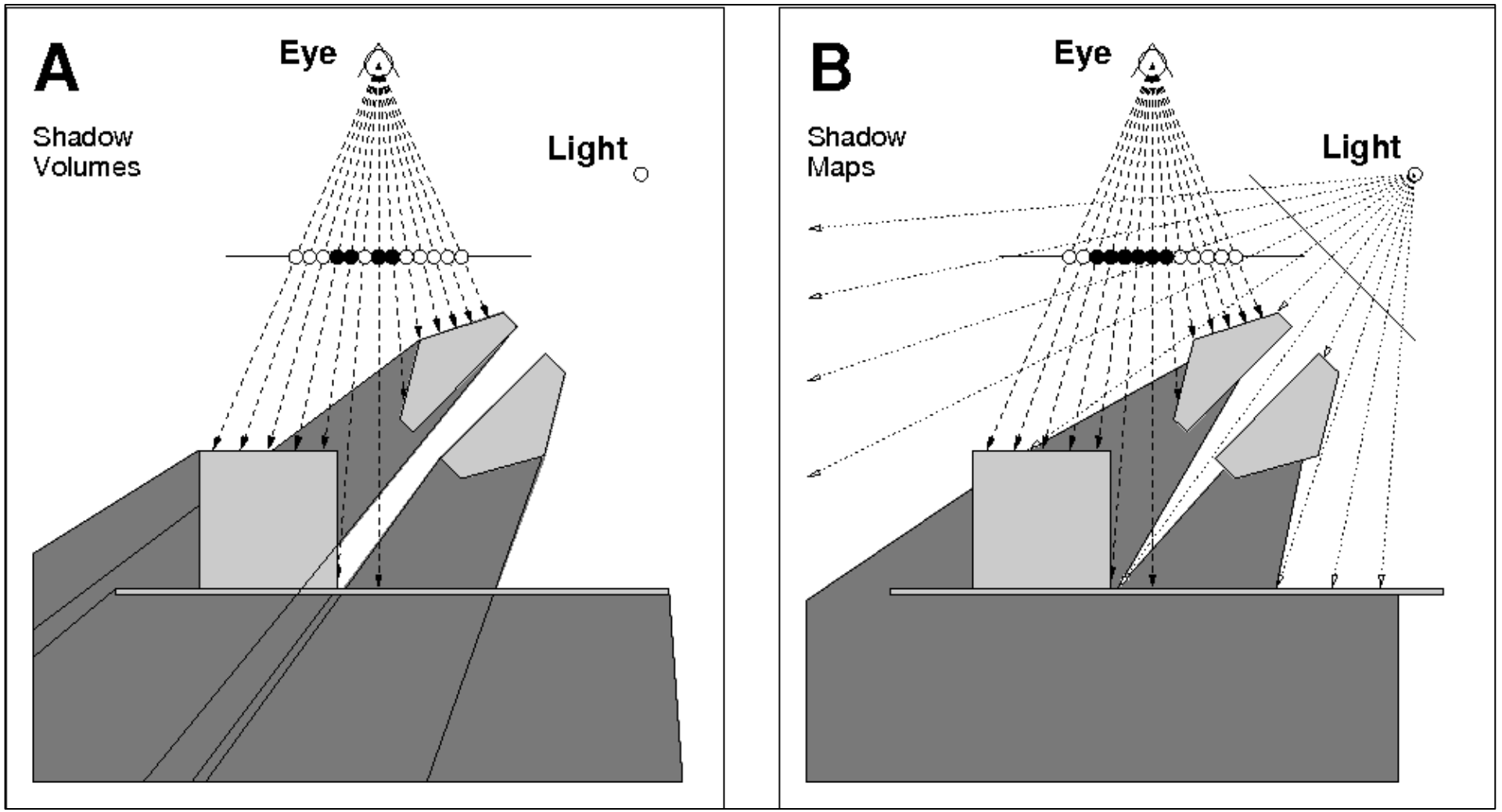


Image from: Chin, Norman, and Steven Feiner. "Near real-time shadow generation using BSP trees." *ACM SIGGRAPH Computer Graphics* 23, no. 3 (1989): 99-106.

Shadow Volumes vs Shadow Maps

Advantages of shadow volumes

- Doesn't suffer from aliasing effects and rounding errors as shadow mapping does
- Can achieve omni-directional shadow casting, shadow mapping requires at least 6 shadow maps to do this

Advantages of shadow mapping

- Faster than shadow volumes (most of the time)
- You don't have to add extra vertices, calculate extra polygons or calculate an object's silhouette. You hardly need to know anything about the objects to be able to use shadow mapping
- More optimization options: different resolutions, depth precision, filtering etc.

Transparency

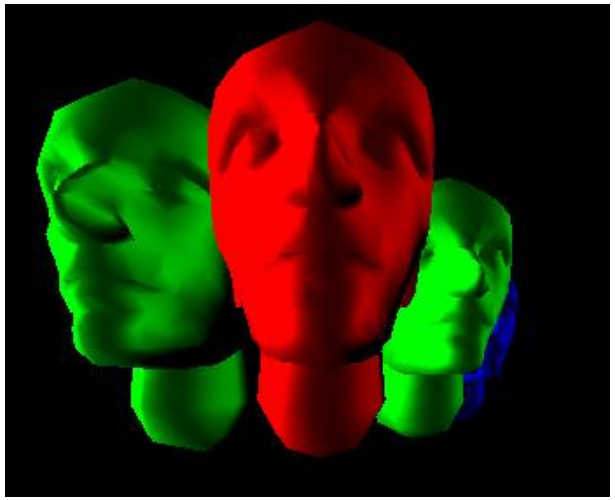
Transparency

- Sometimes we want to render transparent objects
- We blend the colour of the objects along the same ray
  - Alpha blending
  - Screen door transparency

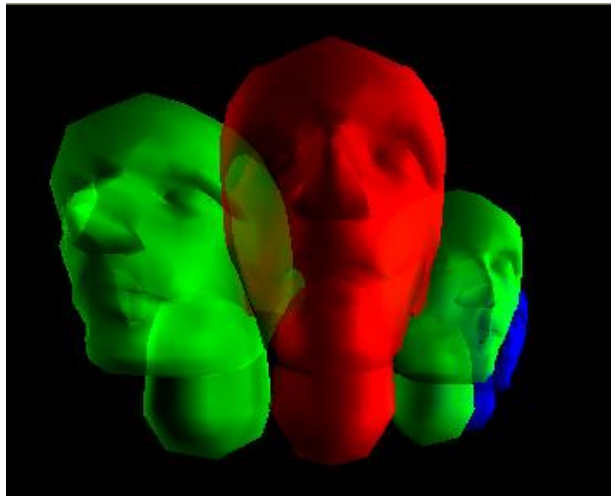
Alpha Blending

Another variable called alpha is defined here
This describes the opacity
Alpha = 1.0 means fully opaque
Alpha = 0.0 means fully transparent

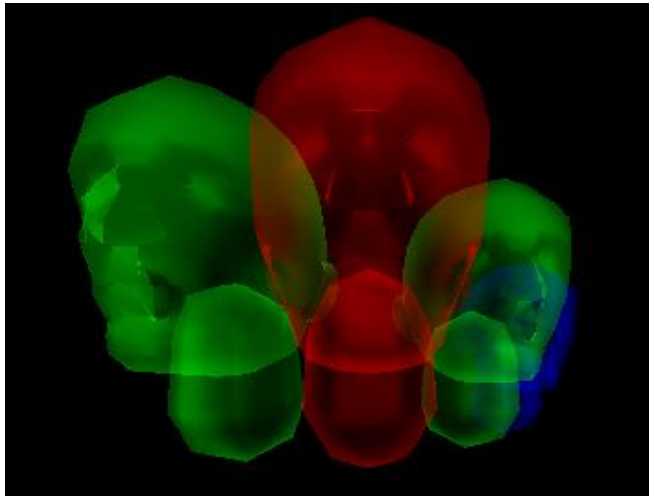
α = 1



α = 0.5

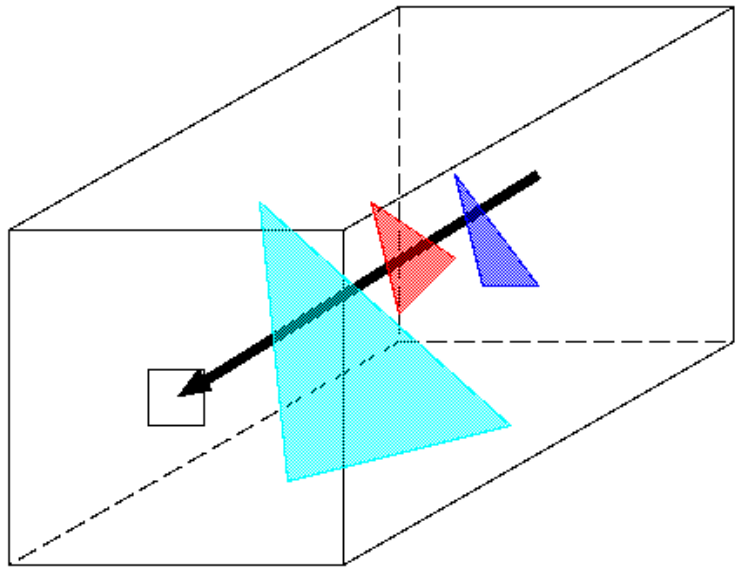


α = 0.2



Sorting by the depth

- First, you need to save the depth and colour of all the fragments that will be projected onto the same pixel in a list
- Then blend the colour from back towards the front



Colour Blending

- The colours of overlapping fragments are blended as follows:

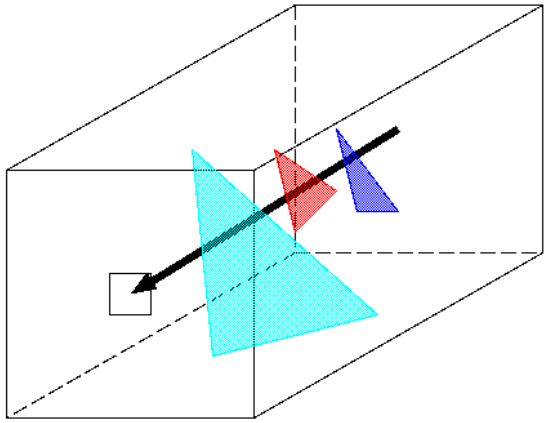
Co = α Cs + (1-α) Cd

Cs : colour of the transparent object,

Cd is the pixel colour before blending,

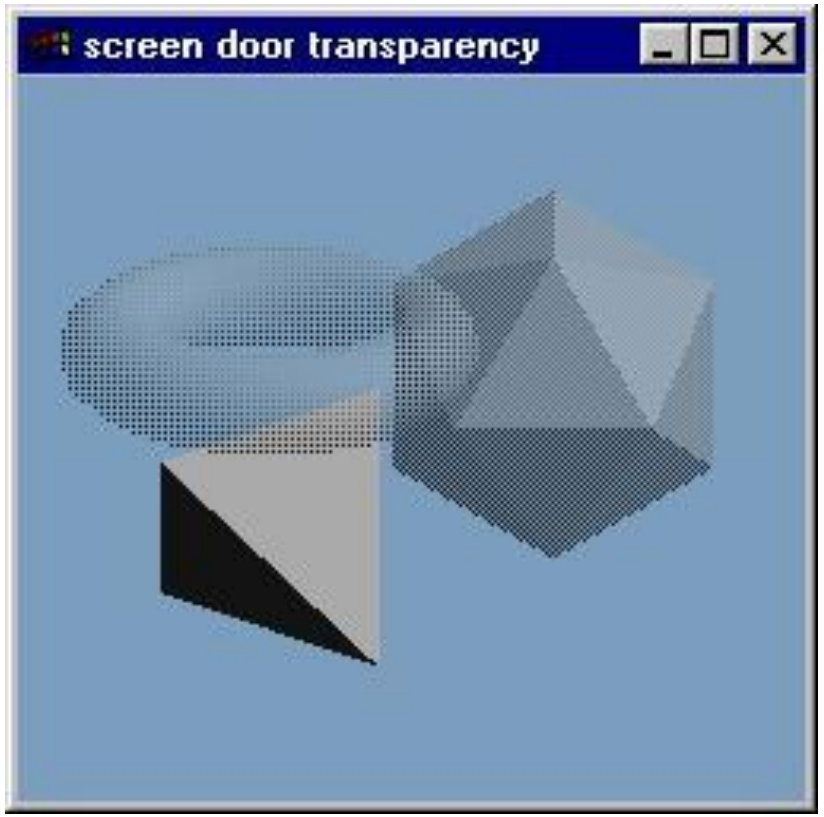
Co is the new colour as a result of blending

Co becomes Cd for the next round



Sorting is expensive

- Need to use BSP Tree
  - Sorting per-pixel is very expensive
- Any faster solution?
  - Screen-door transparency

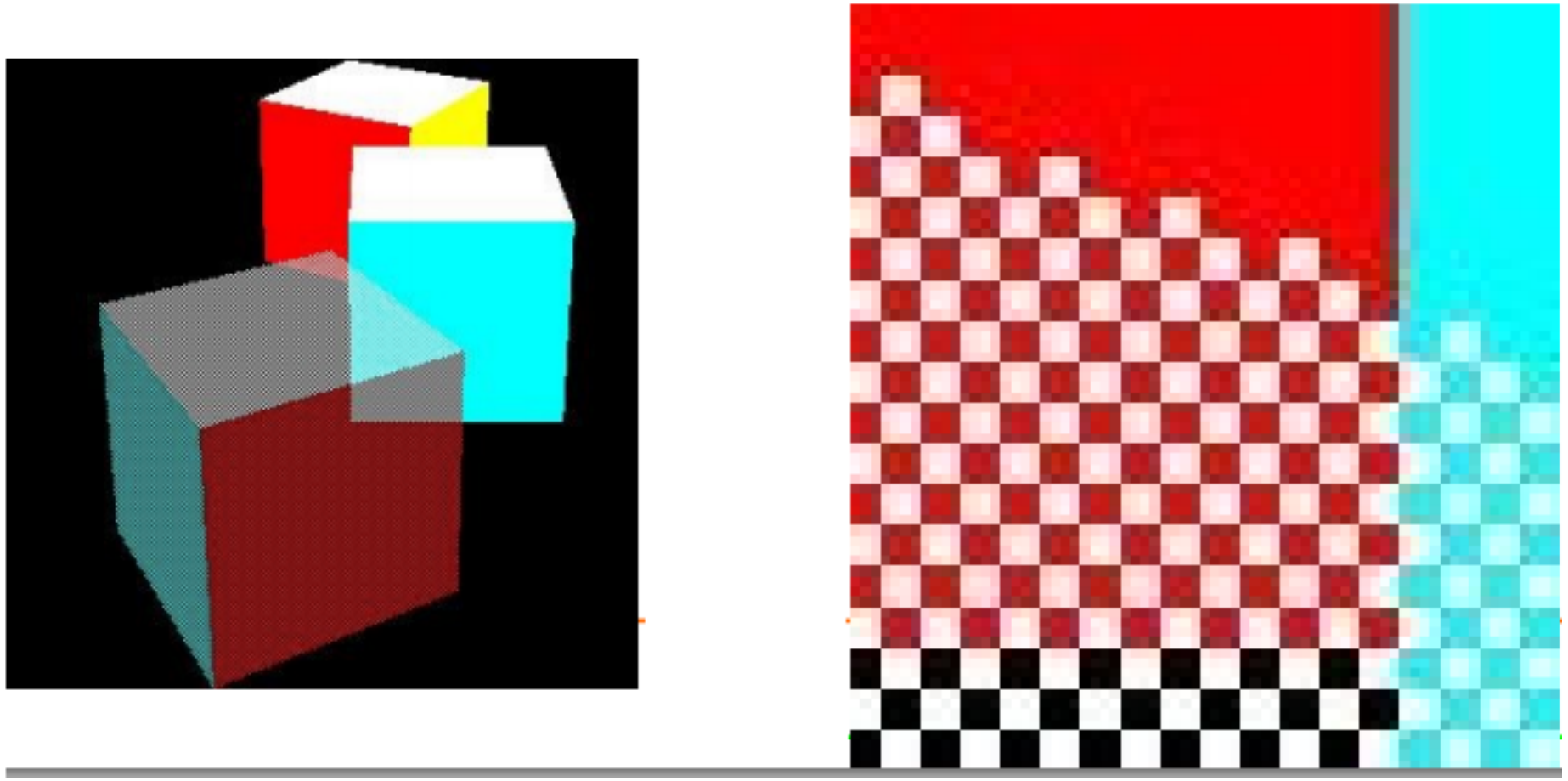


Screen-door Transparency

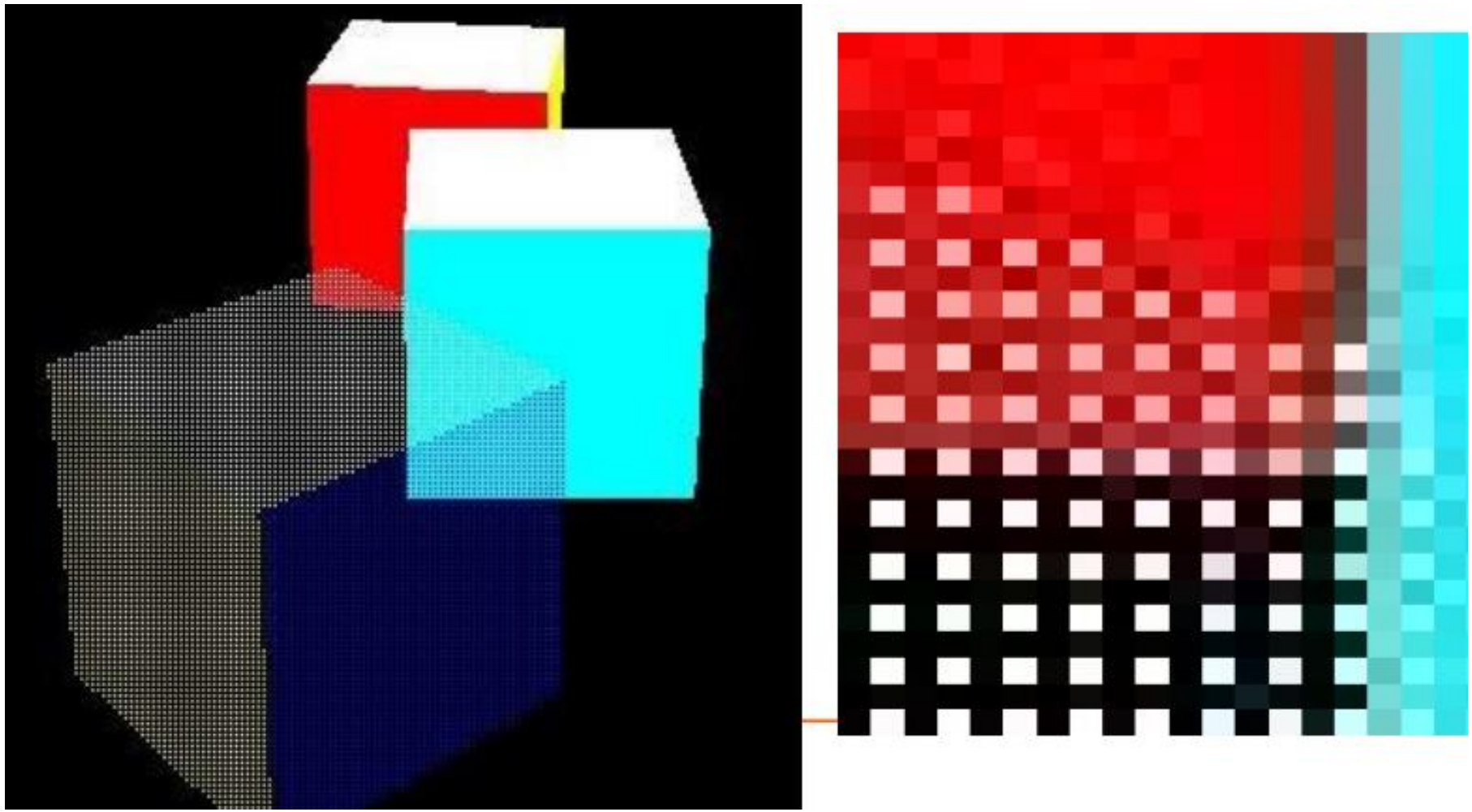
- The object is solid but holes in it
  - like a screen door
- Using a stipple pattern (like a checkboard pattern)

- The ratio of blocked pixels equal to alpha
- No need of sorting : objects can be drawn in any order
- Z-buffer can handle the overlaps of translucent surfaces

alpha = 0.5

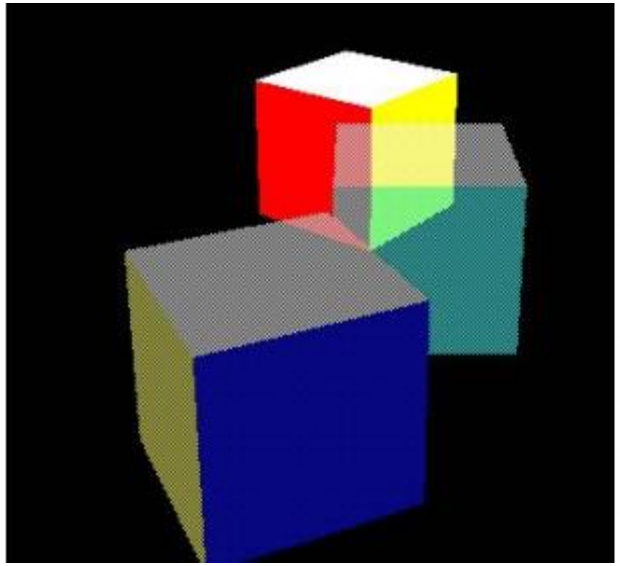


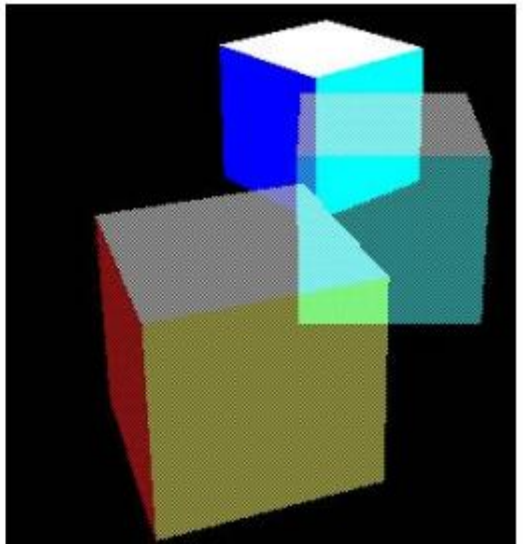
alpha = 0.25



Screen-door Transparency: Issues

- Transparent object over another transparent object can block everything behind
  - When fixed patterns are used





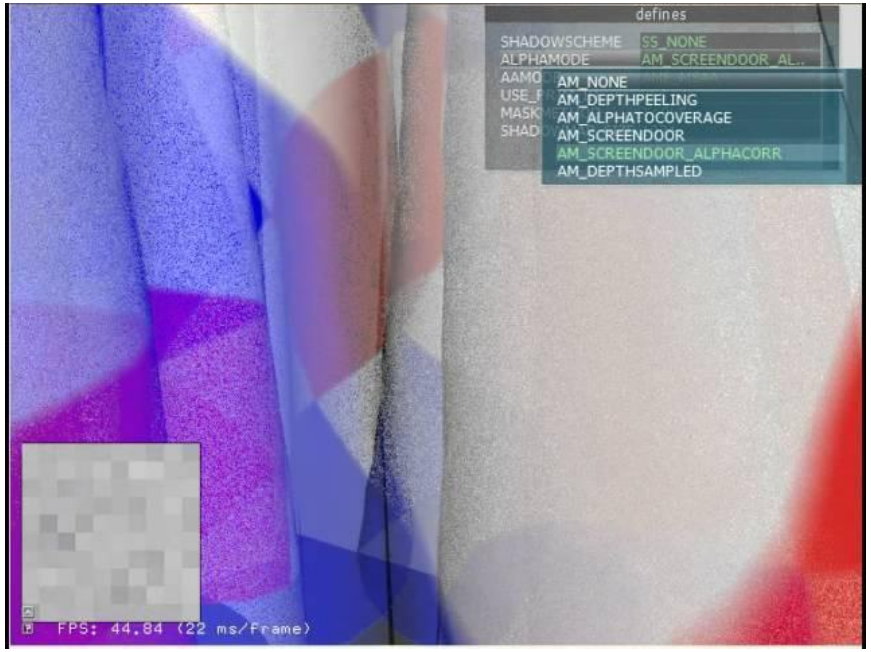
Screen-door Transparency: Issues 2

- Stipple patterns must be set in the screen space – otherwise suffer from aliasing

http://www.youtube.com/watch?v=gMsmJfiApCs

Stochastic Transparency

- Multi-sampling : subpixels are produced and the pixel colour is computed by averaging their colour
- Random sub-pixel stipple pattern



Stochastic Transparency

- No sorting needed
- The final color of the pixel is computed by averaging those of the subpixels