TIEA311
Tietokonegrafiikan perusteet
kevät 2018

(“Principles of Computer Graphics” – Spring 2018)

Copyright and Fair Use Notice:

The lecture videos of this course are made available for registered students only. Please, do not redistribute them for other purposes. Use of auxiliary copyrighted material (academic papers, industrial standards, web pages, videos, and other materials) as a part of this lecture is intended to happen under academic ”fair use” to illustrate key points of the subject matter. The lecturer may be contacted for take-down requests or other copyright concerns (email: paavo.j.nieminen@jyu.fi).

TIEA311 Tietokonegrafiikan perusteet – kevät 2018 (“Principles of Computer Graphics” – Spring 2018)

Adapted from: *Wojciech Matusik*, and *Frédo Durand*: 6.837 Computer Graphics. Fall 2012. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu/.

License: Creative Commons BY-NC-SA

Original license terms apply. Re-arrangement and new content copyright 2017-2018 by *Paavo Nieminen* and *Jarno Kansanaho*

Frontpage of the local course version, held during Spring 2018 at the Faculty of Information technology, University of Jyväskylä:
`http://users.jyu.fi/~nieminen/tgp18/`

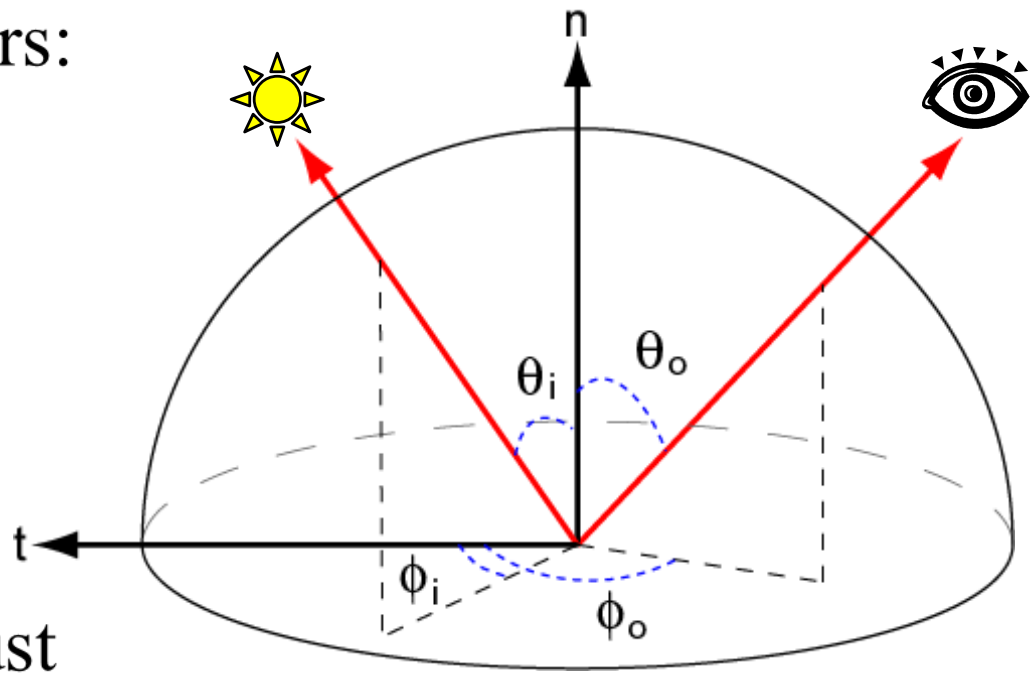
TIEA311 - Today in Jyväskylä

Last lecture plan:

- Shading, texture mapping: Cover the principles up to Phong model and texture coordinates.
- Cherry-pick title slides from advanced stuff that we mostly defer to the follow-up course (starts next week) and/or future self-study.
- When **learning**, don't think about deadlines. **Think about learning!**
- My deadlines are flexible, since this course is about graphics and programming and **not about meeting time-to-market**. Leave that to project courses and traineeship periods!

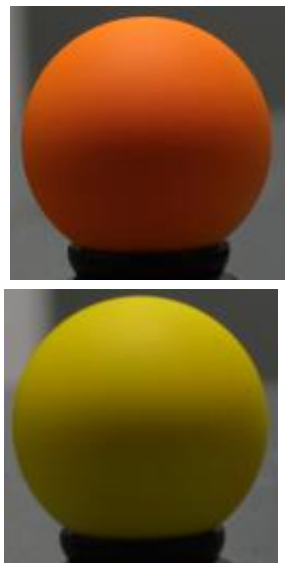
BRDF f_r

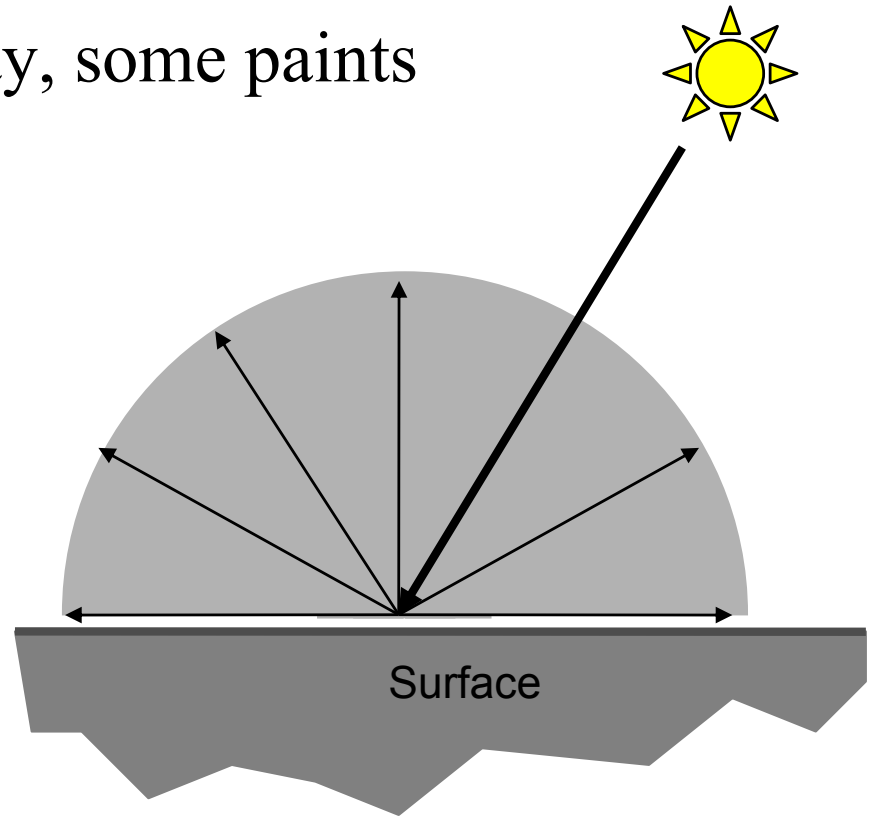
- Bidirectional Reflectance Distribution Function
  - 4D: 2 angles for each direction
  - BRDF = $f_r(\theta_i, \phi_i; \theta_o, \phi_o)$
  - Or just two unit vectors: BRDF = $f_r(\mathbf{l}, \mathbf{v})$
    - **l** = light direction
    - **v** = view direction
  - The BRDF is aligned with the surface; the vectors **l** and **v** must be in a local coordinate system



Ideal Diffuse Reflectance

- Assume surface reflects equally in all directions.
- An ideal diffuse surface is, at the microscopic level, a very rough surface.
  - Example: chalk, clay, some paints

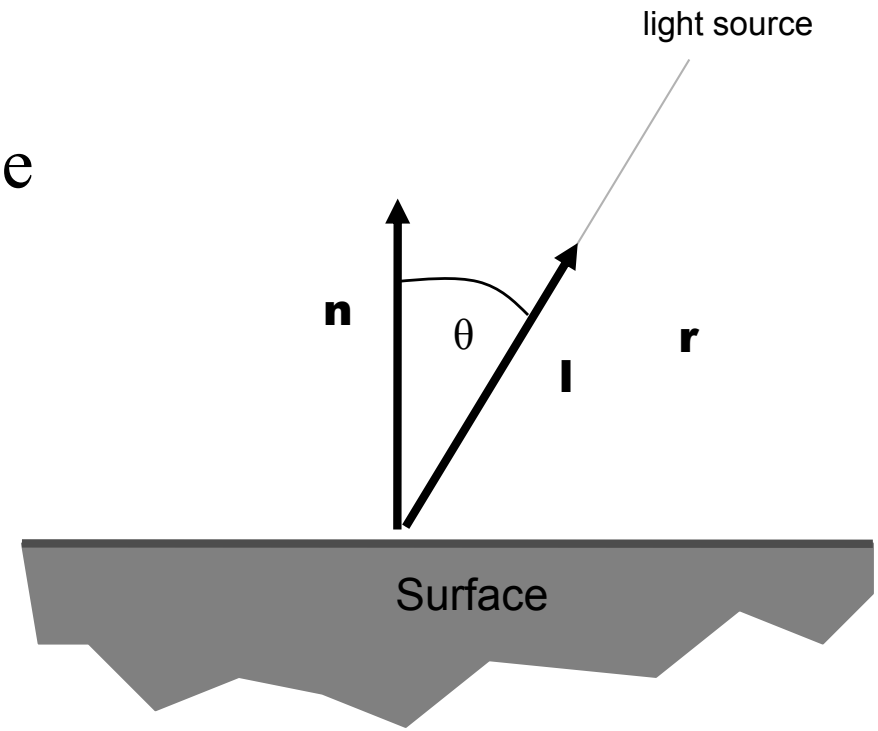




Ideal Diffuse Reflectance Math

- Single Point Light Source
  - k_d: diffuse coefficient (color)
  - **n**: Surface normal.
  - **l**: Light direction.
  - L_i: Light intensity
  - r: Distance to source
  - L_o: Shaded color

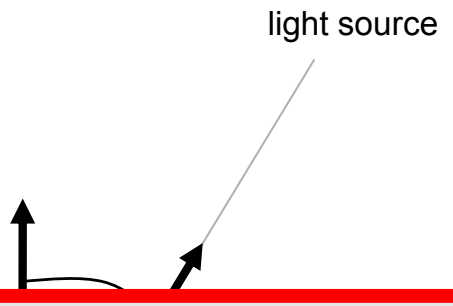
$$L_o = k_d \max(0, \boldsymbol{n} \cdot \boldsymbol{l}) \frac{L_i}{r^2}$$



Ideal Diffuse Reflectance Math

- Single Point Light Source
  - k_d: diffuse coefficient (color)
  - **n**: Surface normal.
  - **l**: Light direction.
  - L_i: Light intensity
  - r: Distance to source
  - L_o: Shaded color

$$L_o = k_d \boxed{\max(0, \boldsymbol{n} \cdot \boldsymbol{l})} \frac{L_i}{r^2}$$

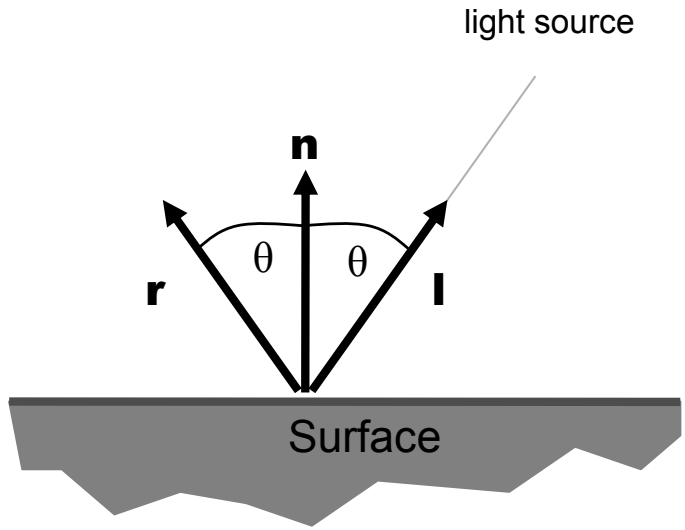


We do not want light from below the surface! From now on we always assume (on this lecture) that **dot products are clamped to zero** and skip writing out the max().

Do not forget to normalize your **n** and **l**!

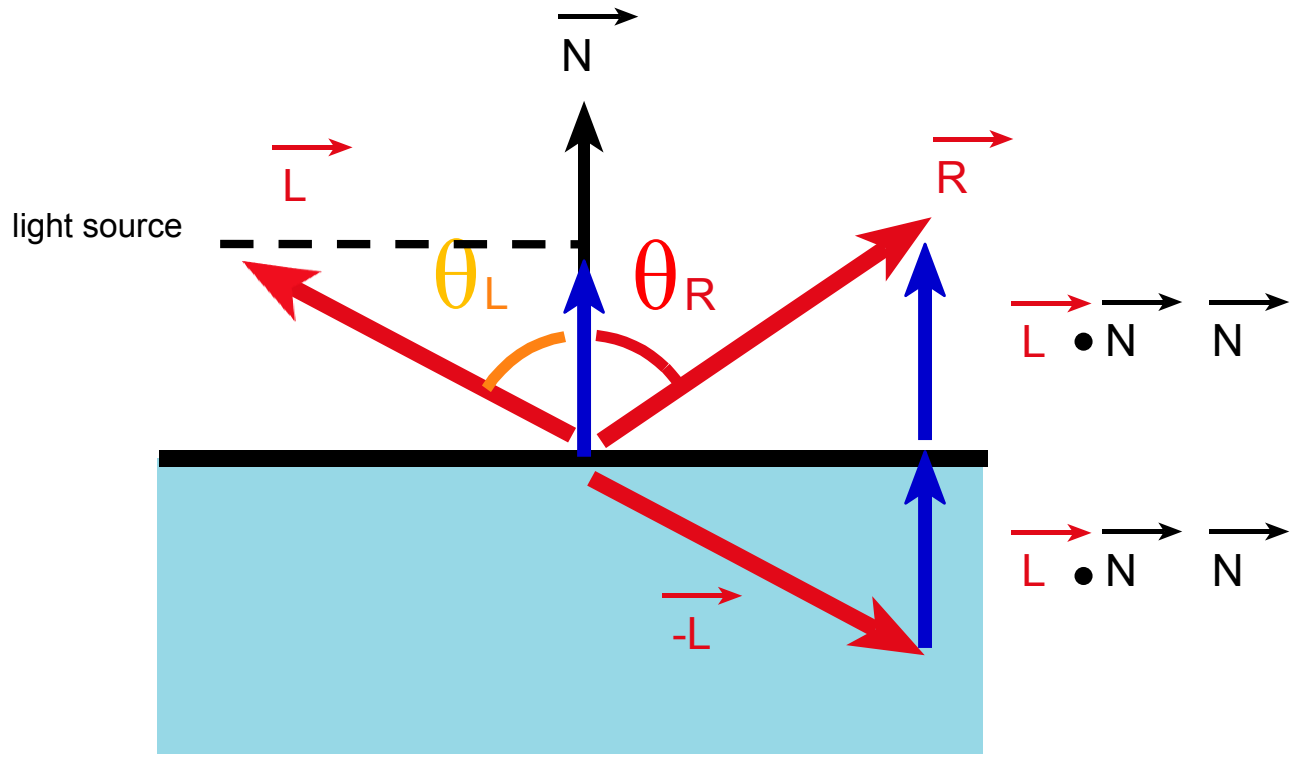
Ideal Specular Reflectance

- Reflection is only at mirror angle
- View dependent
  - Microscopic surface elements are usually oriented in the same direction as the surface itself.
  - Examples: mirrors, highly polished metals.



Recap: How to Get Mirror Direction

- Reflection angle = light angle
  - Both **R** & **L** have to lie on one plane
- $\mathbf{R} = -\mathbf{L} + 2\,(\mathbf{L} \cdot \mathbf{N})\,\mathbf{N}$



Ideal Specular BRDF

- Light **only** reflects to the mirror direction
- A Dirac delta multiplied by a specular coefficient k_s

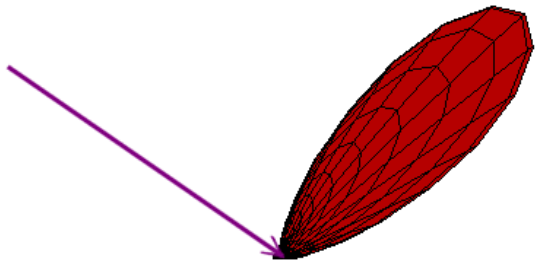
- Not very useful for point lights, only for reflections of other surfaces
  - Why? You cannot really see a mirror reflection of an infinitely small light!

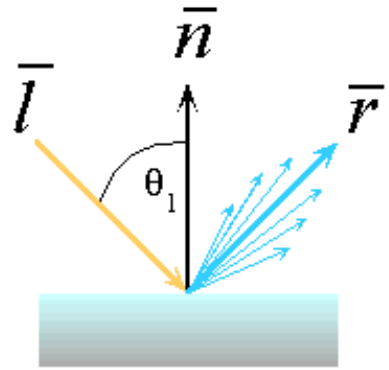
Non-ideal Reflectors

- Real glossy materials usually deviate significantly from ideal mirror reflectors
  - Highlight is blurry
- They are not ideal diffuse surfaces either …

Non-ideal Reflectors

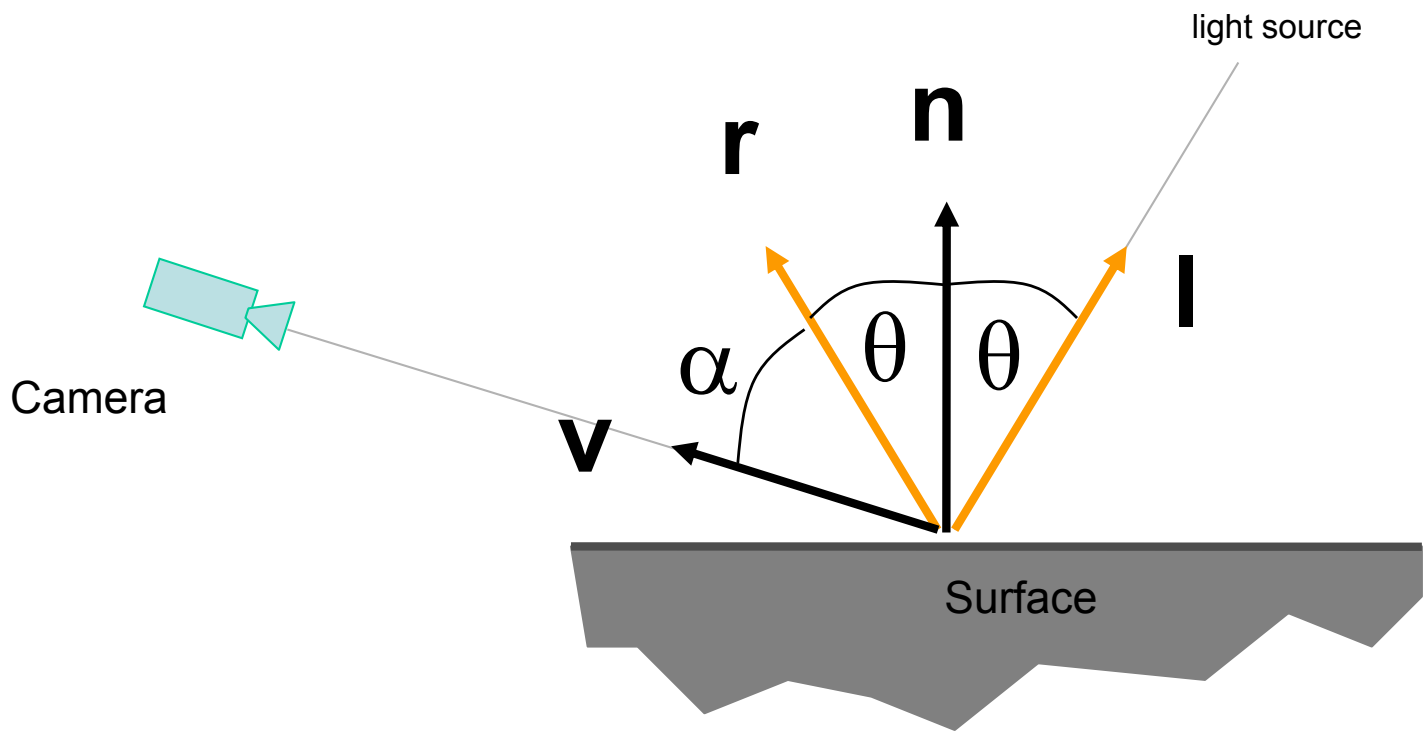
- Simple Empirical Reasoning for Glossy Materials
  - We expect most of the reflected light to travel in the direction of the ideal mirror ray.
  - However, because of microscopic surface variations we might expect some of the light to be reflected just slightly offset from the ideal reflected ray.
  - As we move farther and farther, in the angular sense, from the reflected ray, we expect to see less light reflected.





The Phong Specular Model

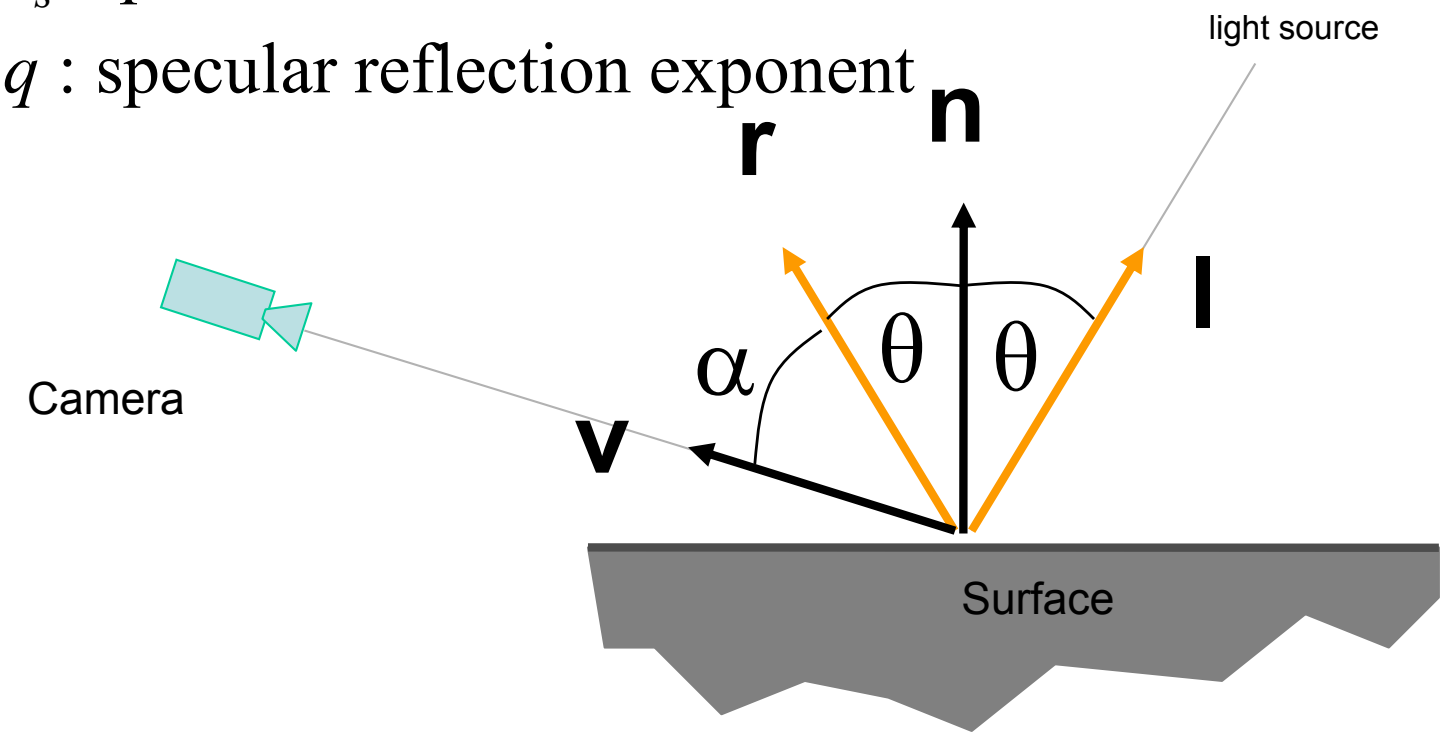
- How much light is reflected?
  - Depends on the angle α between the ideal reflection direction **r** and the viewer direction **v**.



The Phong Specular Model

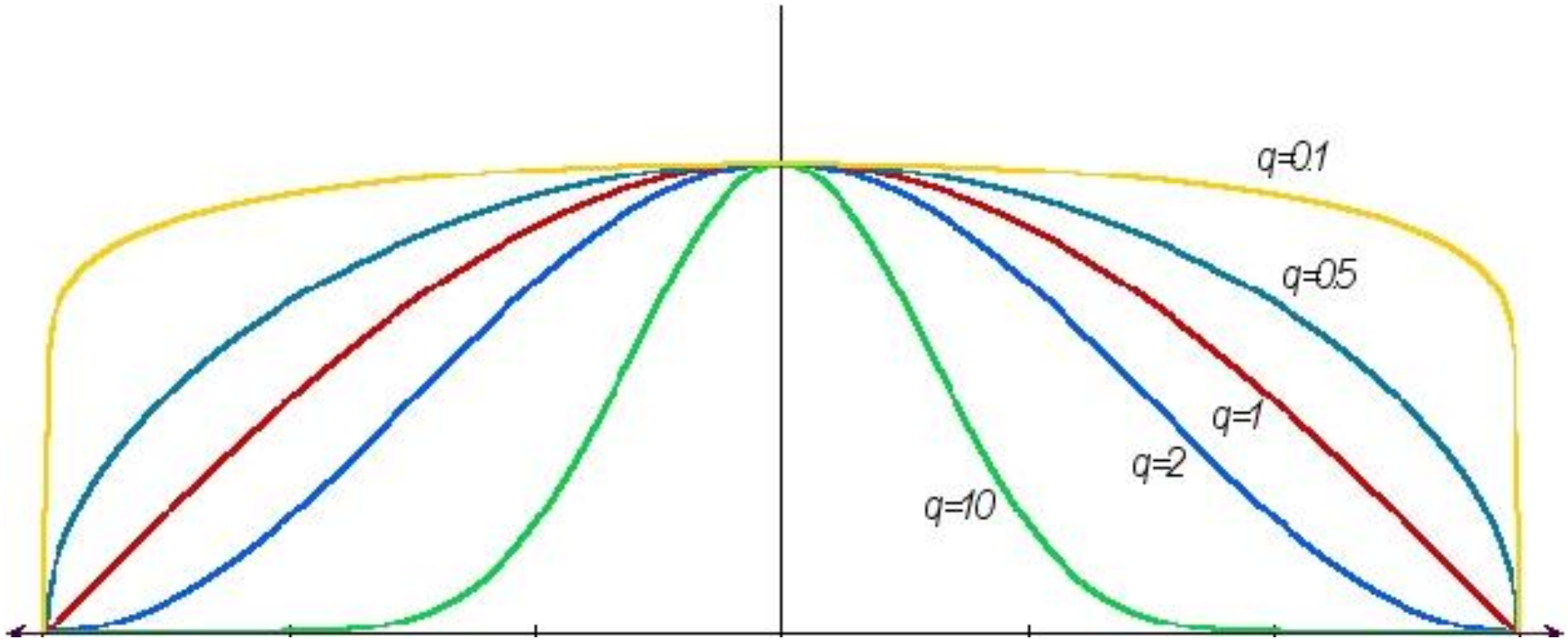
$$L_o = k_s (\cos\alpha)^q \frac{L_i}{r^2} = k_s (\boldsymbol{v} \cdot \boldsymbol{r})^q \frac{L_i}{r^2}$$

- Parameters
  - k_s: specular reflection coefficient
  - q : specular reflection exponent



The Phong Model

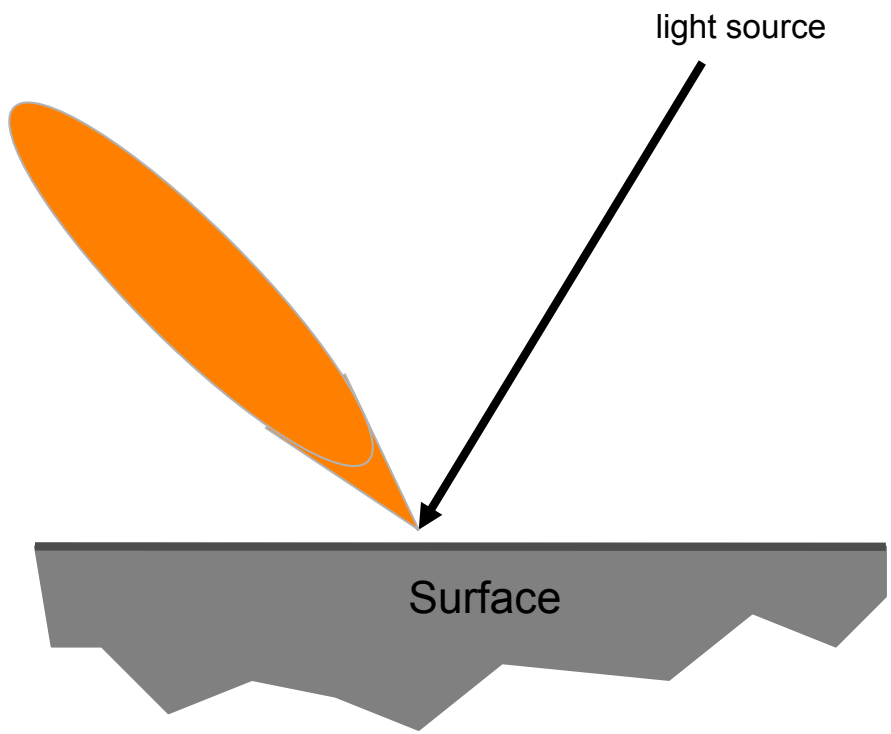
- Effect of q – the specular reflection exponent



$\alpha = -\pi/2$ $\alpha = 0$ $\alpha = \pi/2$

Terminology: Specular Lobe

- The specular reflection distribution is usually called a “lobe”
  - For Phong, its shape is $(\boldsymbol{r} \cdot \boldsymbol{v})^q$



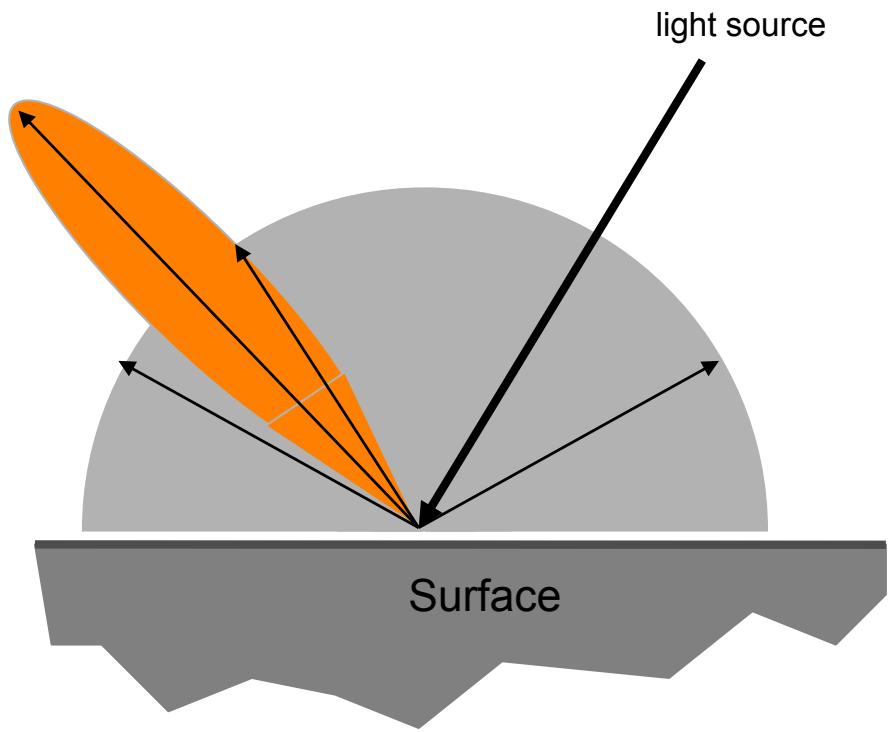
The Complete Phong Model

- Sum of three components:

  ideal diffuse reflection +

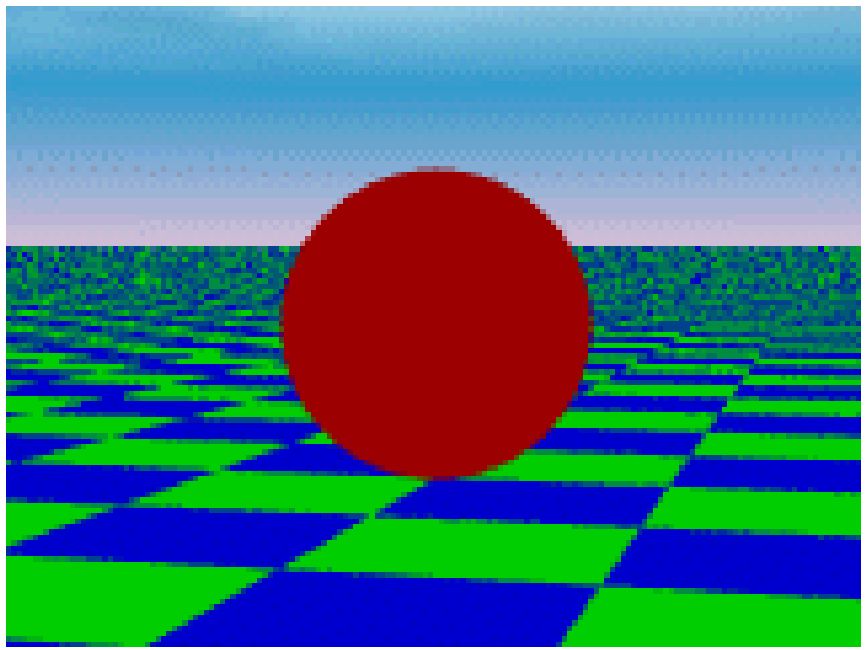
  specular reflection +

  “ambient”.



Ambient Illumination

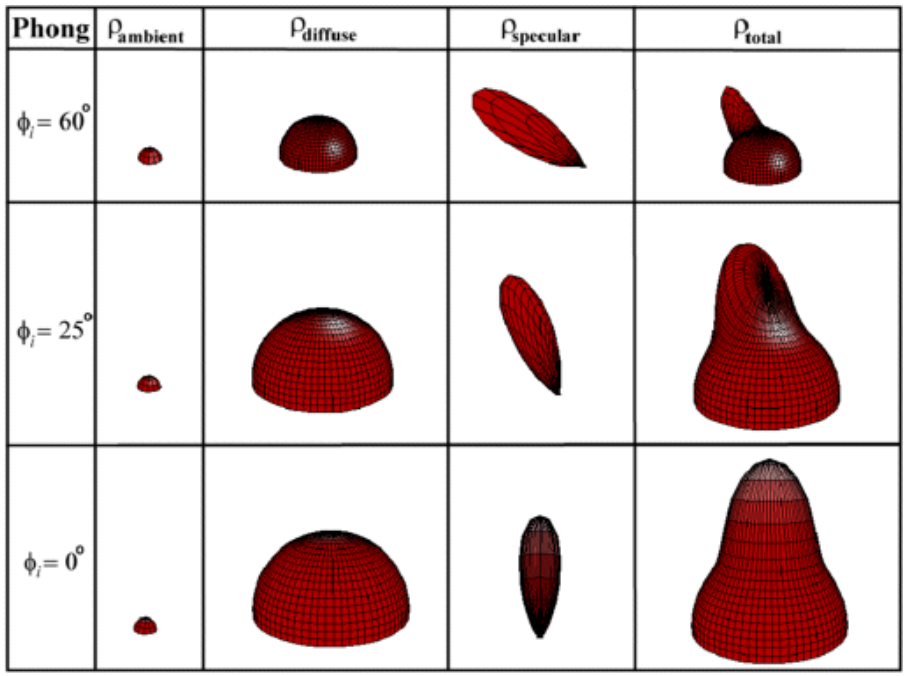
- Represents the reflection of all indirect illumination.
- This is a total hack!
- Avoids the complexity of indirect ("global") illumination



Putting It All Together

- Phong Illumination Model

$$L_o = \left[k_a + k_d \left(\boldsymbol{n} \cdot \boldsymbol{l} \right) + k_s \left(\boldsymbol{v} \cdot \boldsymbol{r} \right)^q \right] \frac{L_i}{r^2}$$



© Leonard McMillan. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Putting It All Together

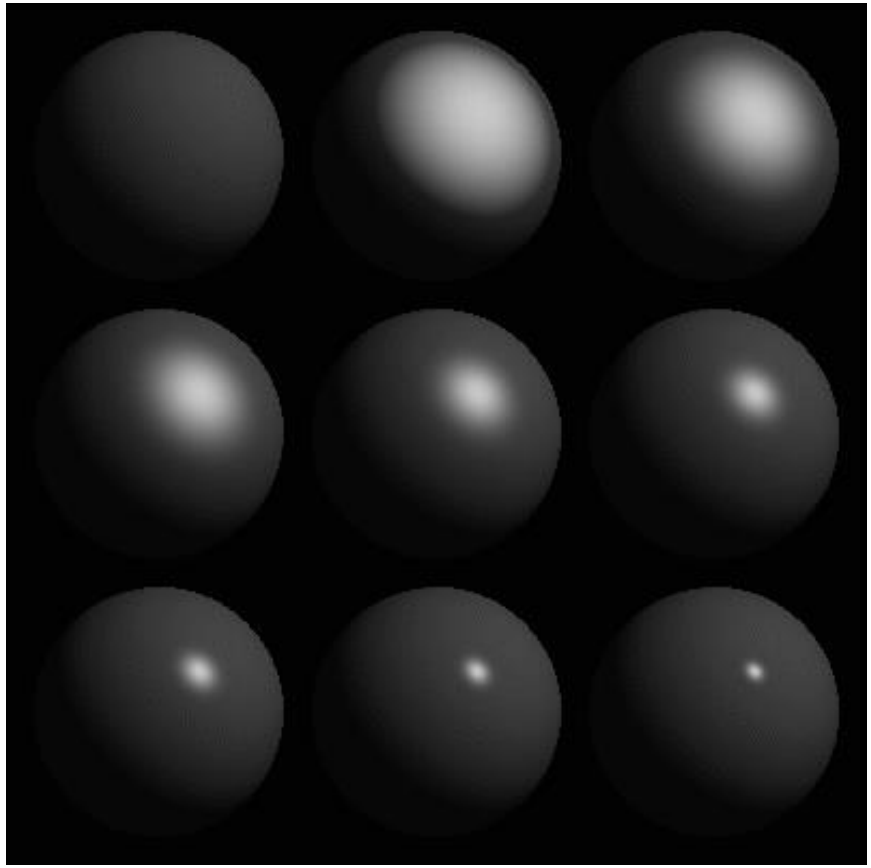
- Phong Illumination Model

$$L_o = \left[k_a + k_d\left(\boldsymbol{n}\cdot\boldsymbol{l}\right) + k_s\left(\boldsymbol{v}\cdot\boldsymbol{r}\right)^q\right]\frac{L_i}{r^2}$$

- Is it physically based?
  - No, does not even conserve energy, may well reflect more energy than what goes in
  - Furthermore, it does not even conform to the BRDF model directly (we are taking the proper cosine for diffuse, but not for specular)
  - And ambient was a total hack

Phong Examples

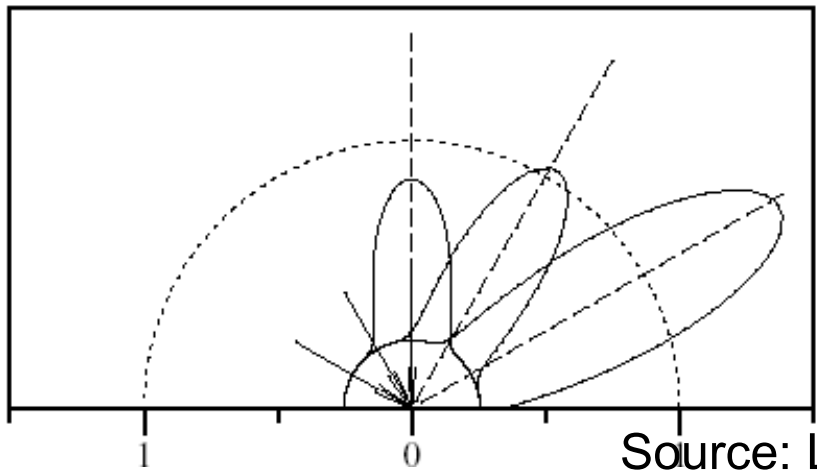
- The spheres illustrate specular reflections as the direction of the light source and the exponent q (amount of shininess) is varied.



$$L_o = \left[k_a + k_d\left(\boldsymbol{n}\cdot\boldsymbol{l}\right) + k_s\left(\boldsymbol{v}\cdot\boldsymbol{r}\right)^q\right]\frac{L_i}{r^2}$$

Fresnel Reflection

- Increasing specularity near grazing angles.
  - Most BRDF models account for this.



Source: Lafortune et al. 97

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Questions?

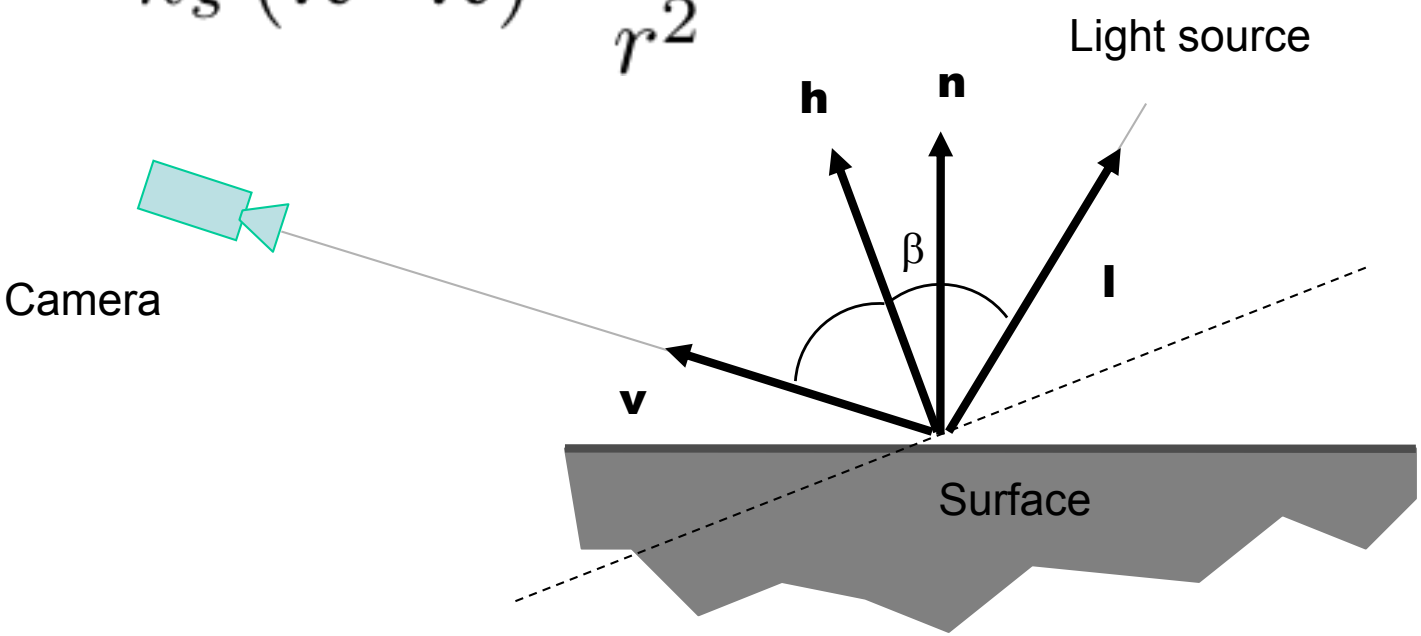
Blinn-Torrance Variation of Phong

- Uses the "halfway vector" **h** between **l** and **v**.

$$L_o = k_s \cos(\beta)^q \frac{L_i}{r^2}$$

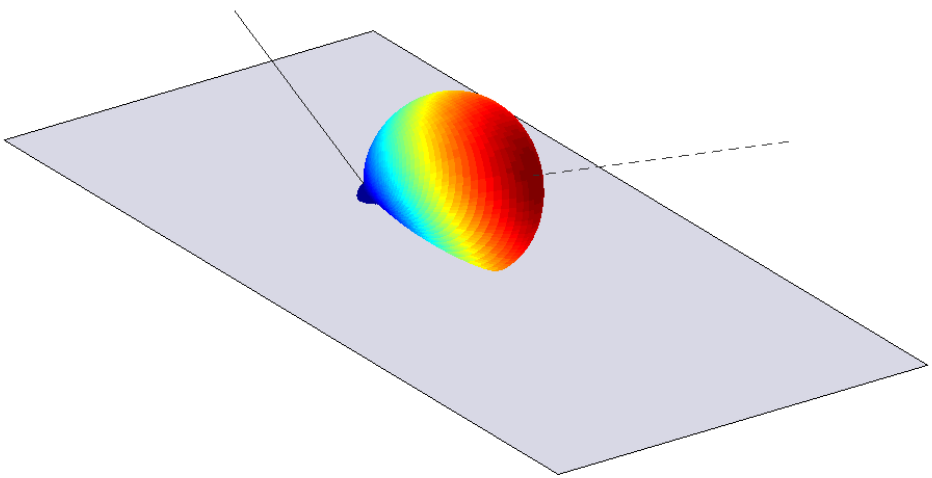
$$= k_s (\boldsymbol{n} \cdot \boldsymbol{h})^q \frac{L_i}{r^2}$$

$$\boldsymbol{h} = \frac{\boldsymbol{l} + \boldsymbol{v}}{\|\boldsymbol{l} + \boldsymbol{v}\|}$$

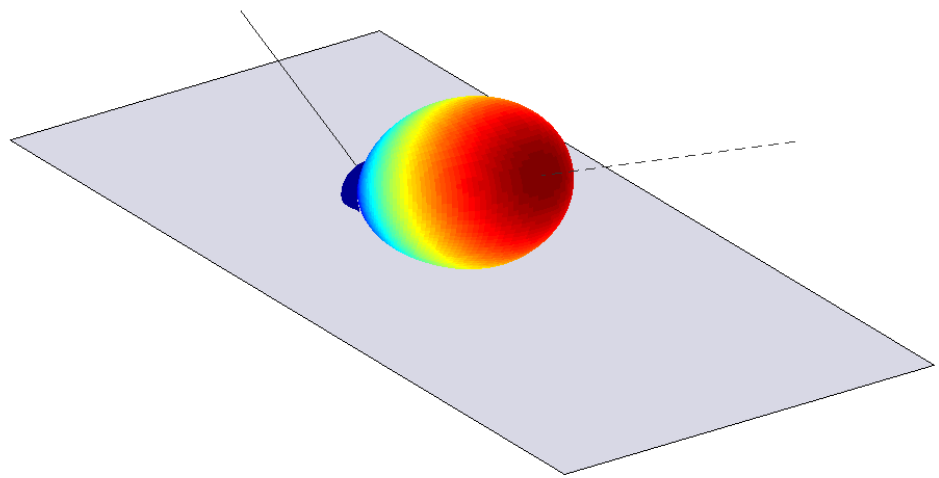


Lobe Comparison

- Half vector lobe
  - Gradually narrower when approaching grazing
- Mirror lobe
  - Always circular



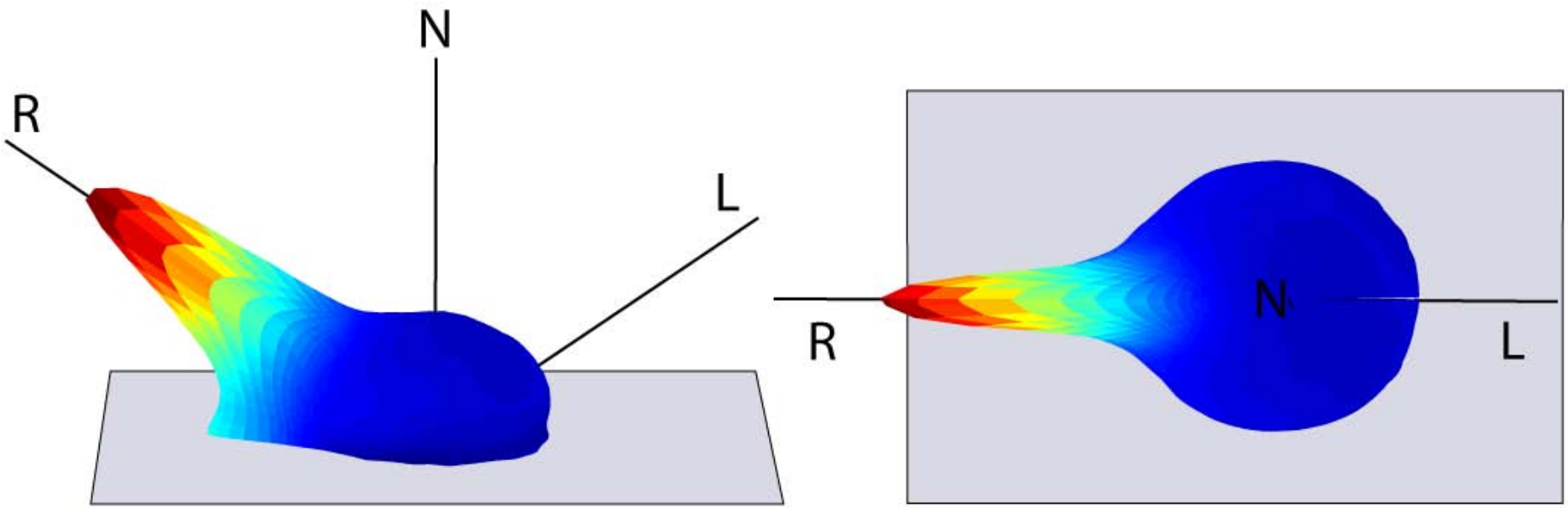
Half vector lobe



Mirror lobe

Half Vector Lobe is Better

- More consistent with what is observed in measurements (Ngan, Matusik, Durand 2005)



Courtesy of Mitsubishi Electric Researh Laboratories, Inc. Used with permission.

Example: Plot of "PVC" BRDF at 55° incidence

Questions?

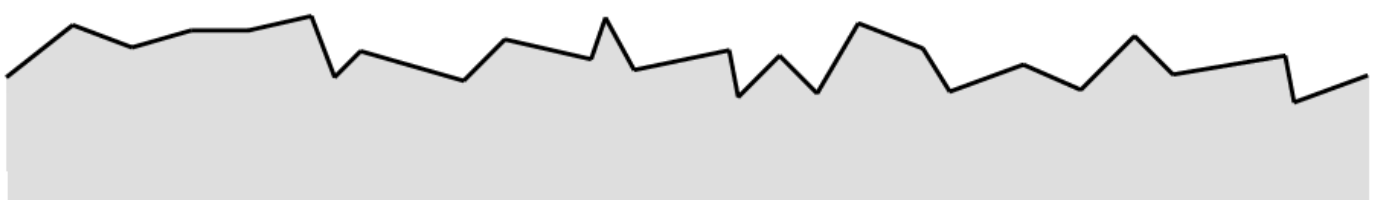
Microfacet Theory

- Example
  - Think of water surface as lots of tiny mirrors (microfacets)
  - “Bright” pixels are
    - Microfacets aligned with the vector between sun and eye
    - But not the ones in shadow
    - And not the ones that are occluded

Image of sunset removed due to copyright restrictions.

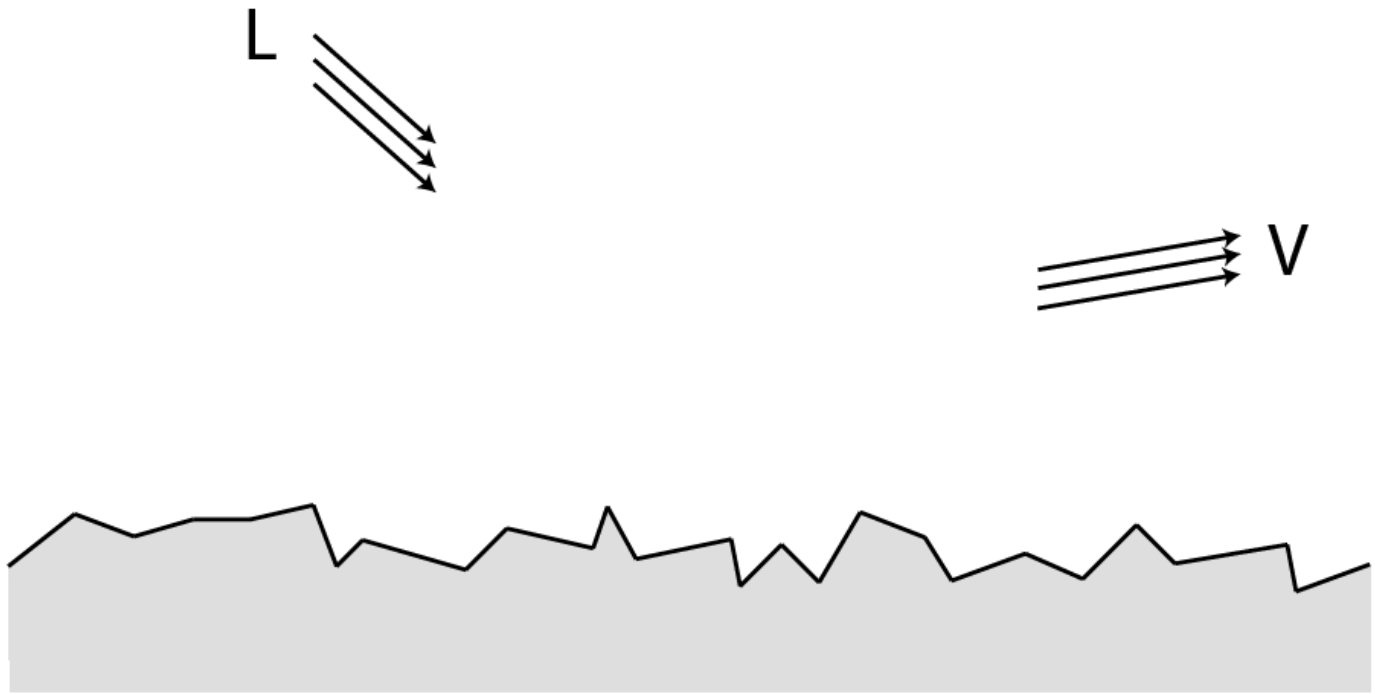
Microfacet Theory

- Model surface by tiny mirrors
  [Torrance & Sparrow 1967]



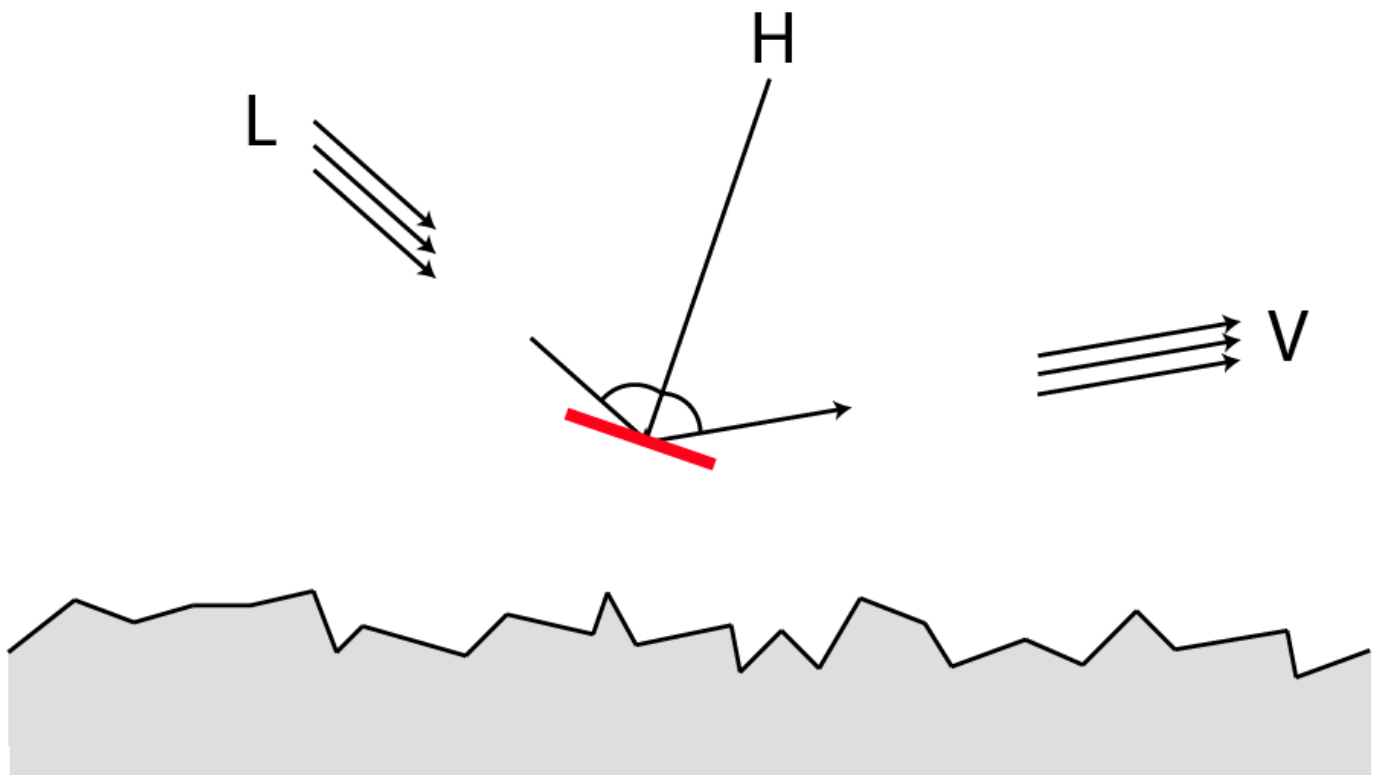
Microfacet Theory

- Value of BRDF at (L,V) is a product of
  - number of mirrors oriented halfway between L and V



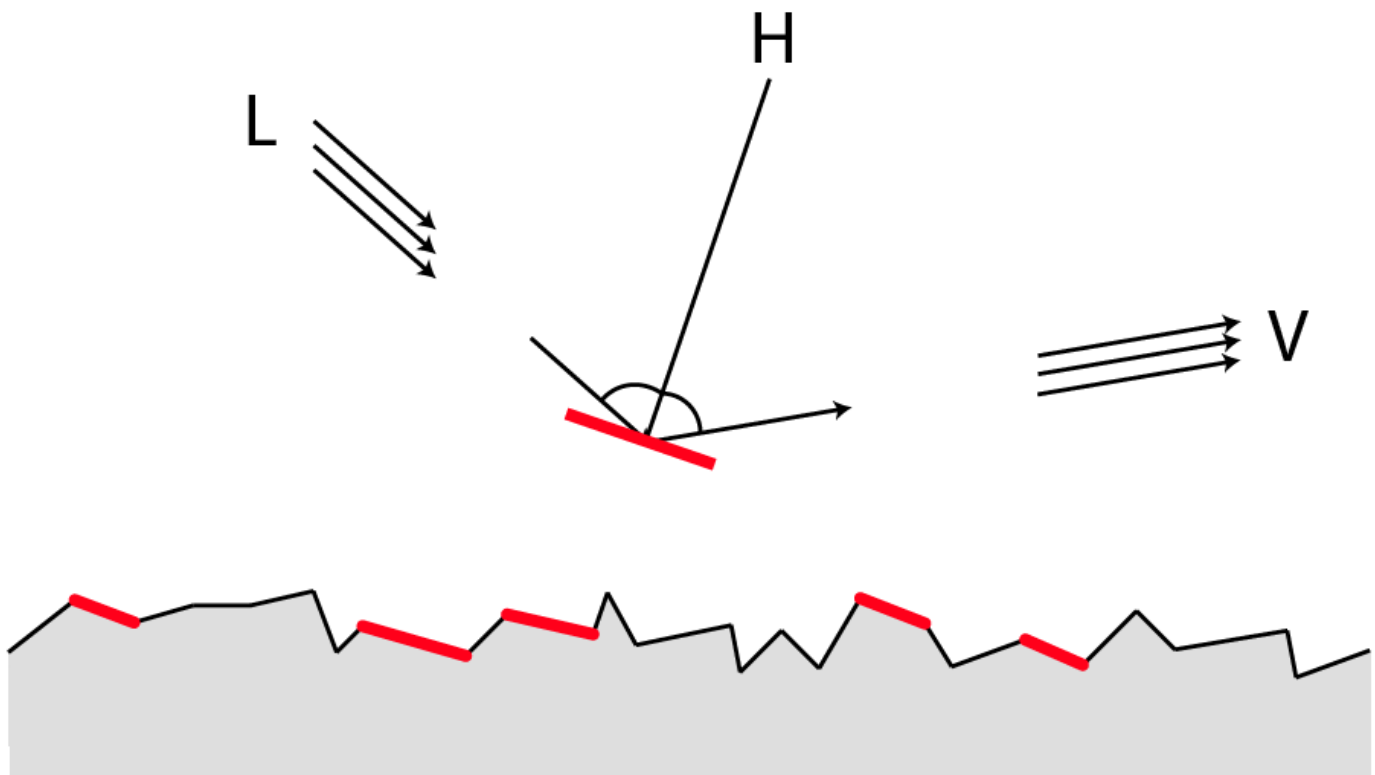
Microfacet Theory

- Value of BRDF at (L,V) is a product of
  - number of mirrors oriented halfway between L and V



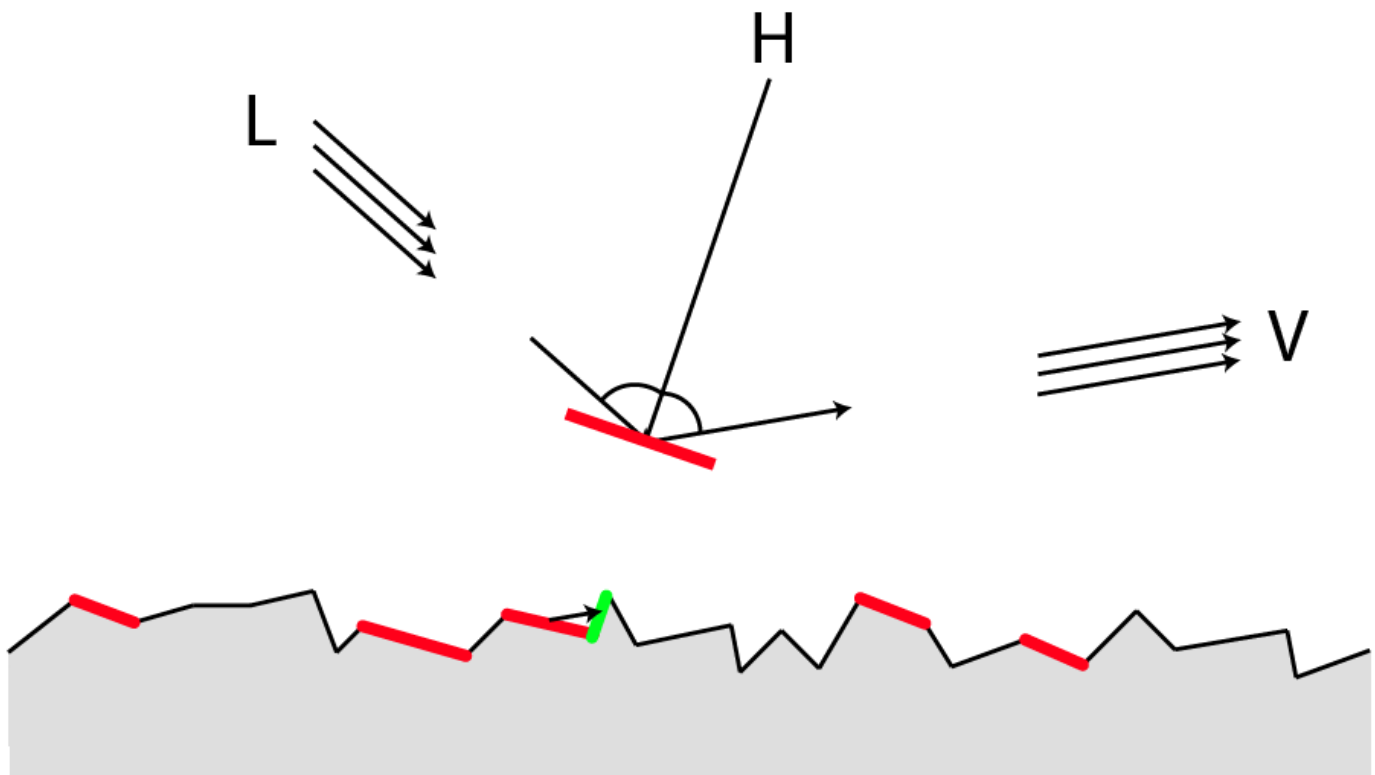
Microfacet Theory

- Value of BRDF at (L,V) is a product of
  - number of mirrors oriented halfway between L and V



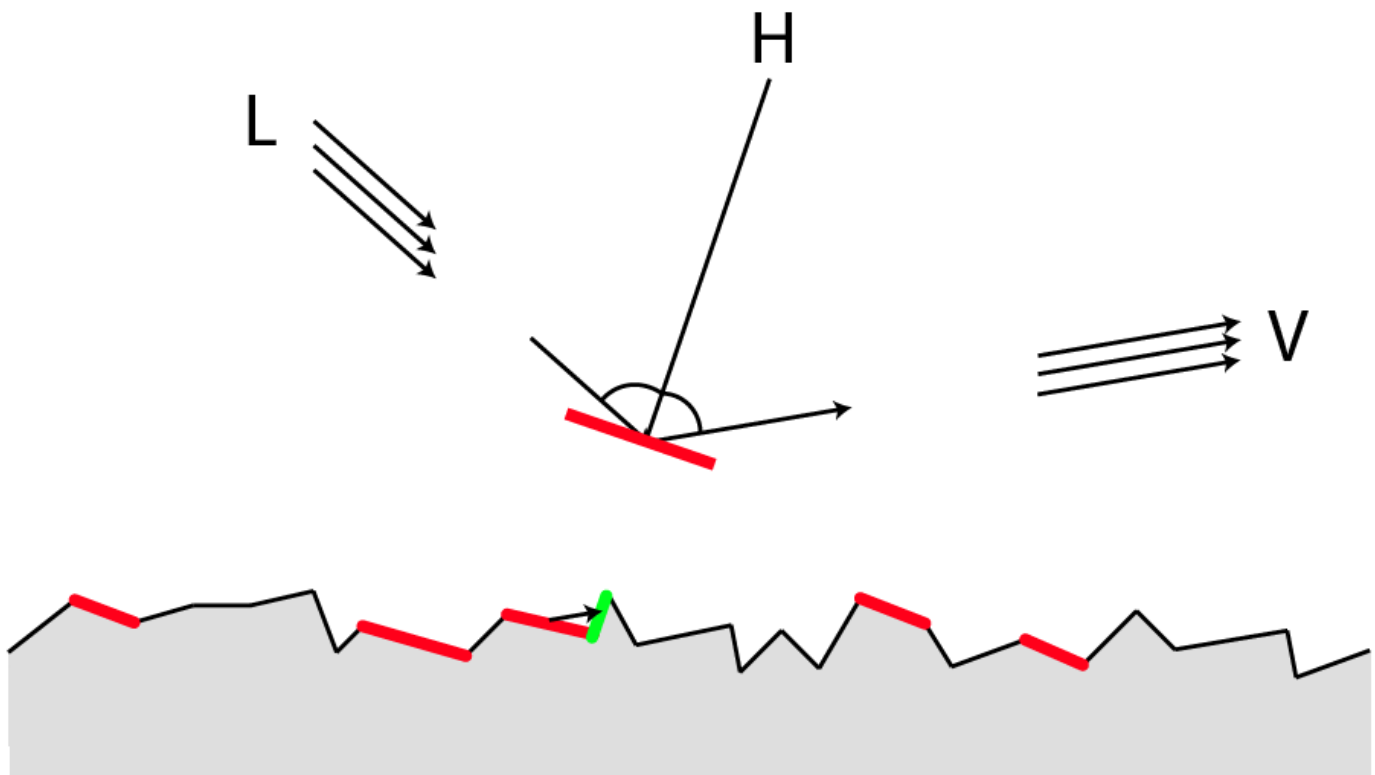
Microfacet Theory

- Value of BRDF at (L,V) is a product of
  - number of mirrors oriented halfway between L and V
  - ratio of the un(shadowed/masked) mirrors



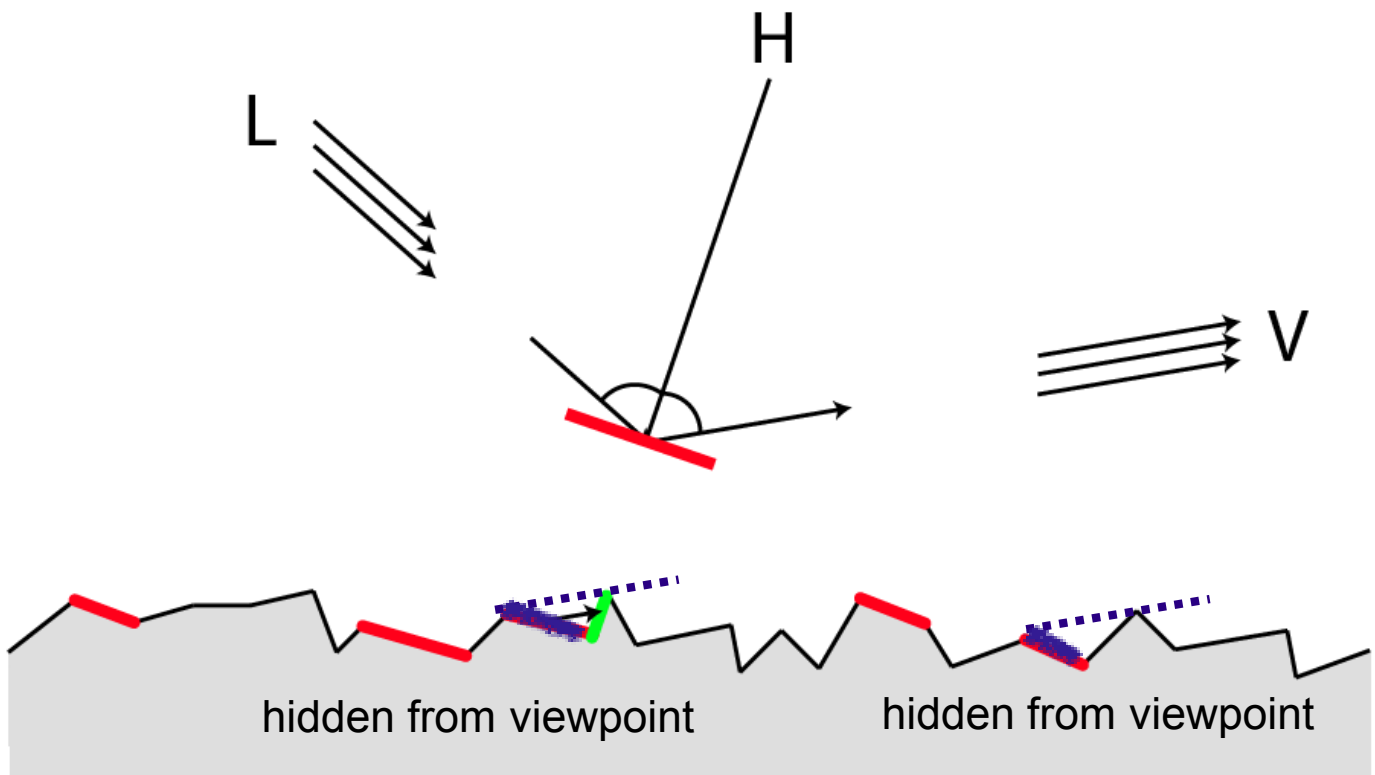
Microfacet Theory

- Value of BRDF at (L,V) is a product of
  - number of mirrors oriented halfway between L and V
  - ratio of the un(shadowed/masked) mirrors
  - Fresnel coefficient



Shadowing and Masking

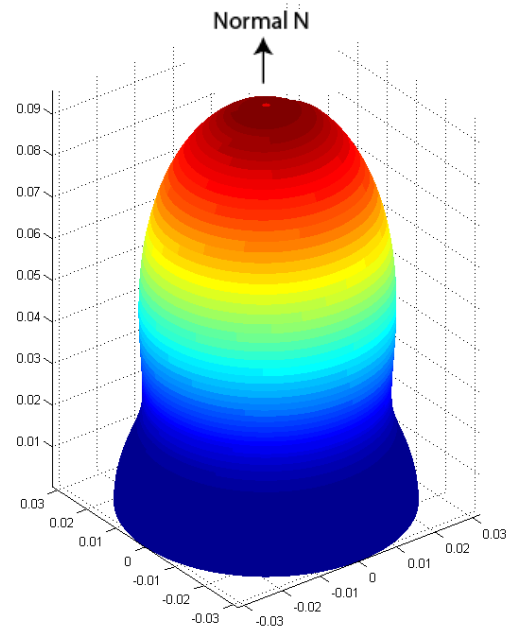
- Some facets are hidden from viewpoint
- Some are hidden from the light



Microfacet Theory-based Models

- Develop BRDF models by imposing simplifications [Torrance-Sparrow 67], [Blinn 77], [Cook-Torrance 81], [Ashikhmin et al. 2000]

- Model the distribution p(H) of microfacet normals
  - Also, statistical models for shadows and masking



spherical plot of a Gaussian-like p(H)

Full Cook-Torrance Lobe

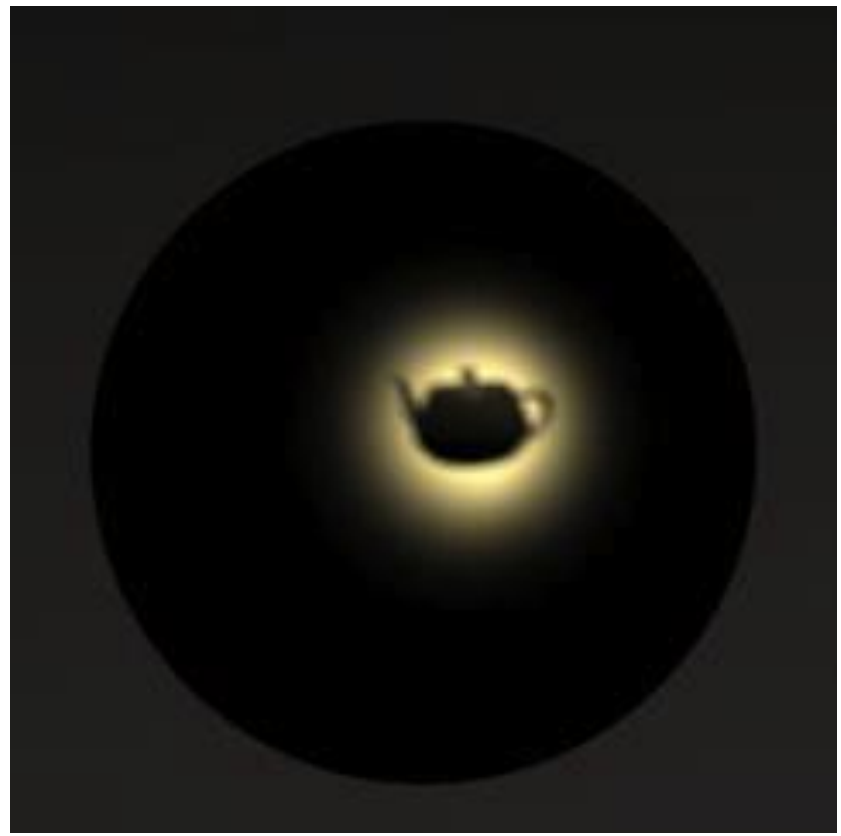
- ρ_s is the specular coefficient (3 numbers RGB)
- D is the microfacet distribution
  - δ is the angle between the half vector H and the normal N
  - m defines the roughness (width of lobe)
- G is the shadowing and masking term
- Need to add a diffuse term

$$K = \frac{\rho_s}{\pi} \frac{DG}{(N \cdot L)(N \cdot V)} Fresnel(F_0, V \cdot H)$$

where $G = min\{1, \frac{2(N \cdot H)(N \cdot V)}{(V \cdot H)}, \frac{2(N \cdot H)(N \cdot L)}{(V \cdot H)}\}$ and $D = \frac{1}{m^2 \cos^4 \delta} e^{-[(tan\delta)/m]^2}$

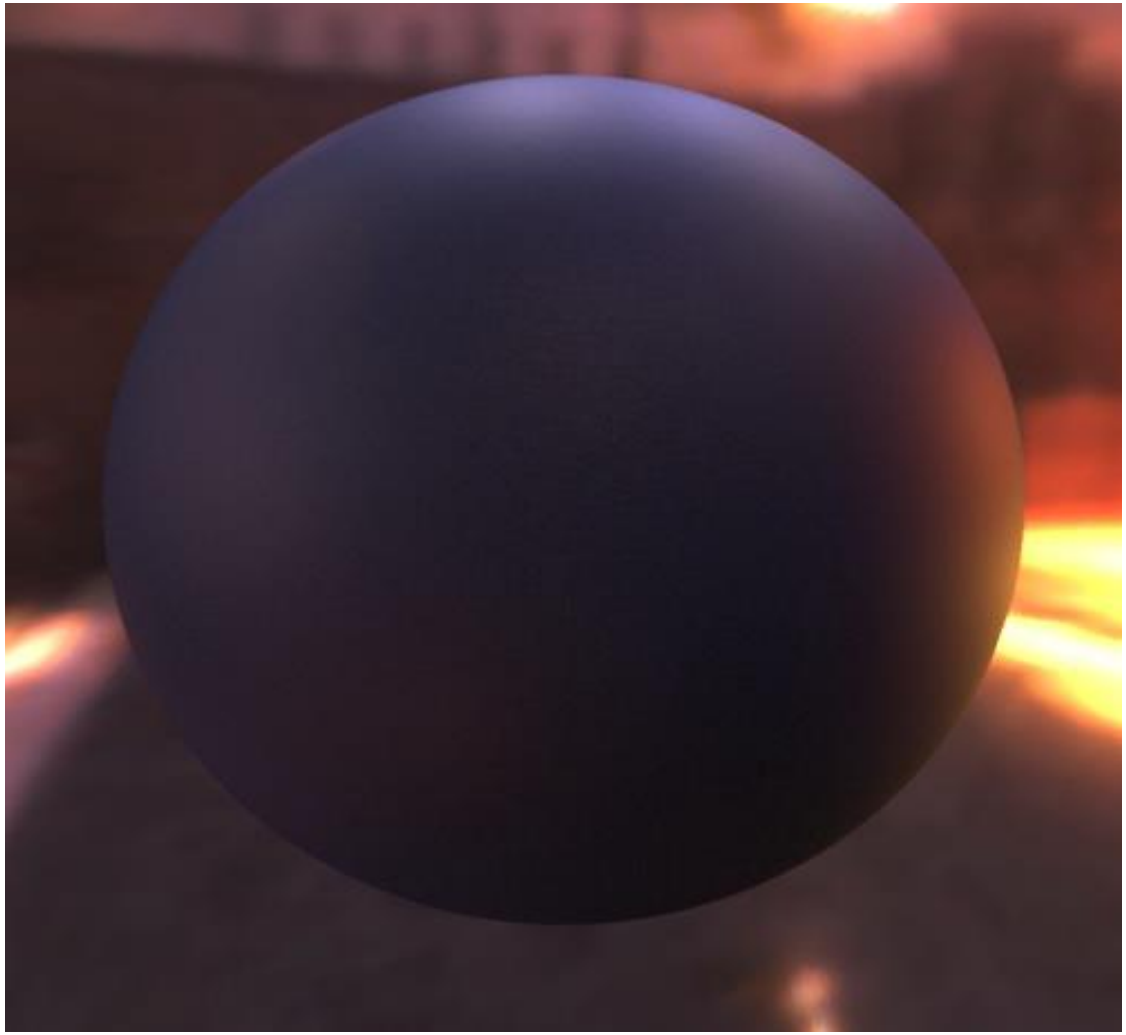
Questions?

- "Designer BRDFs" by Ashikhmin et al.



© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

BRDF Examples from Ngan et al.



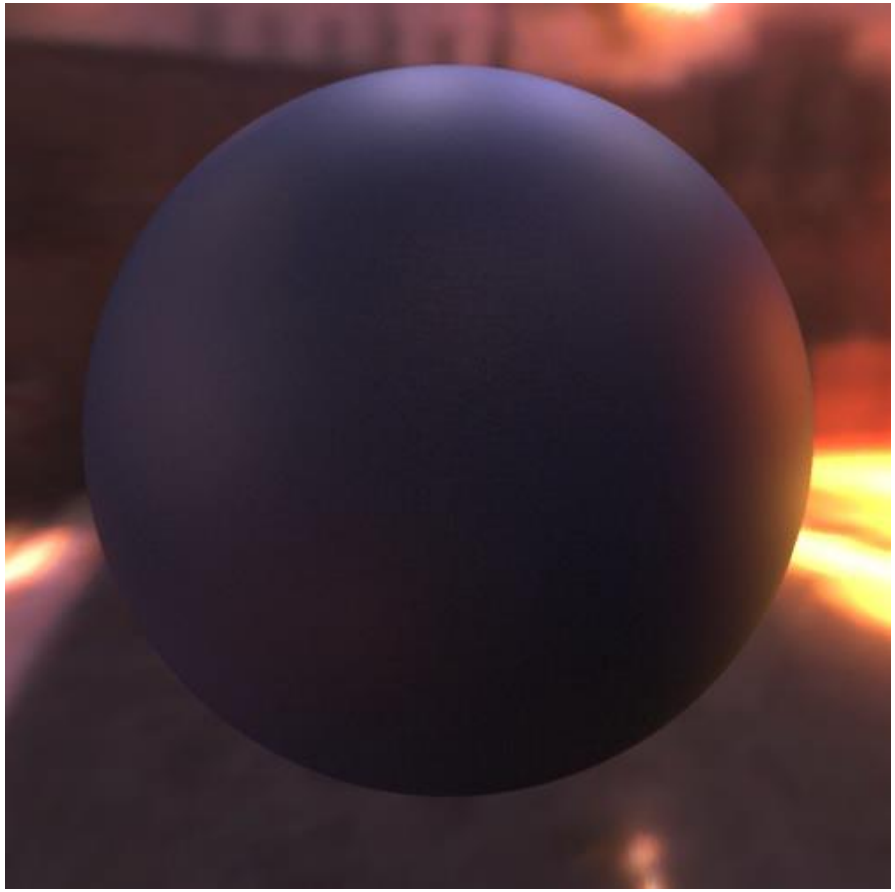
Material – Dark blue paint

Lighting

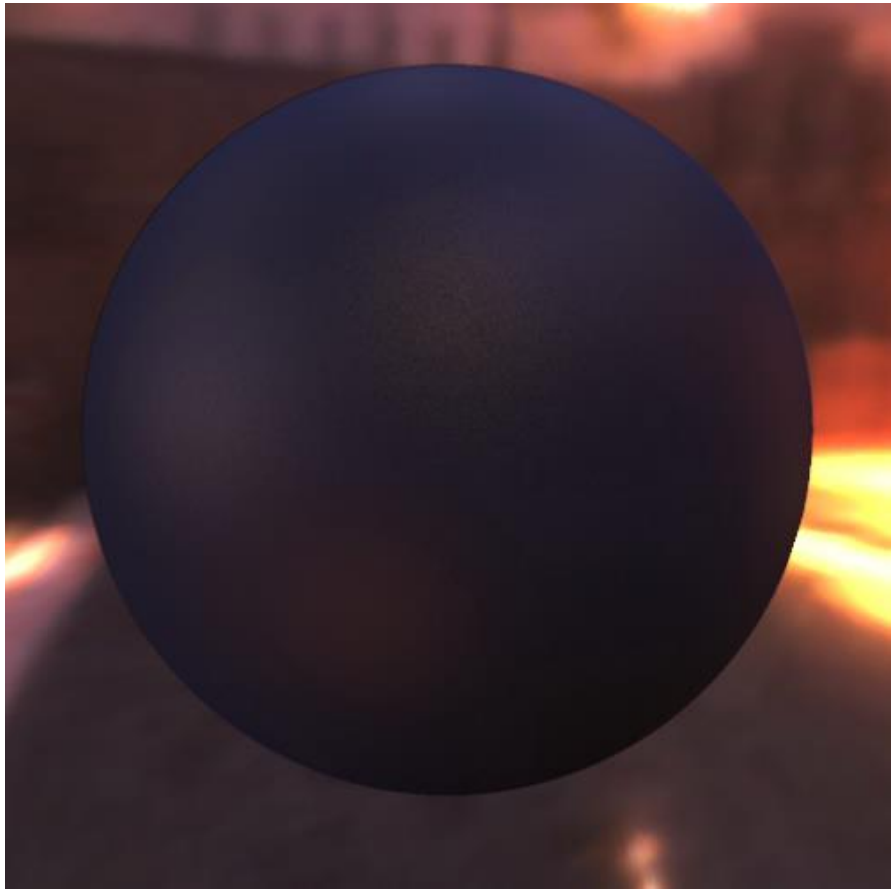
Courtesy of Mitsubishi Electric Researh Laboratories, Inc. Used with permission.

Dark Blue Paint

Acquired data



Blinn-Phong



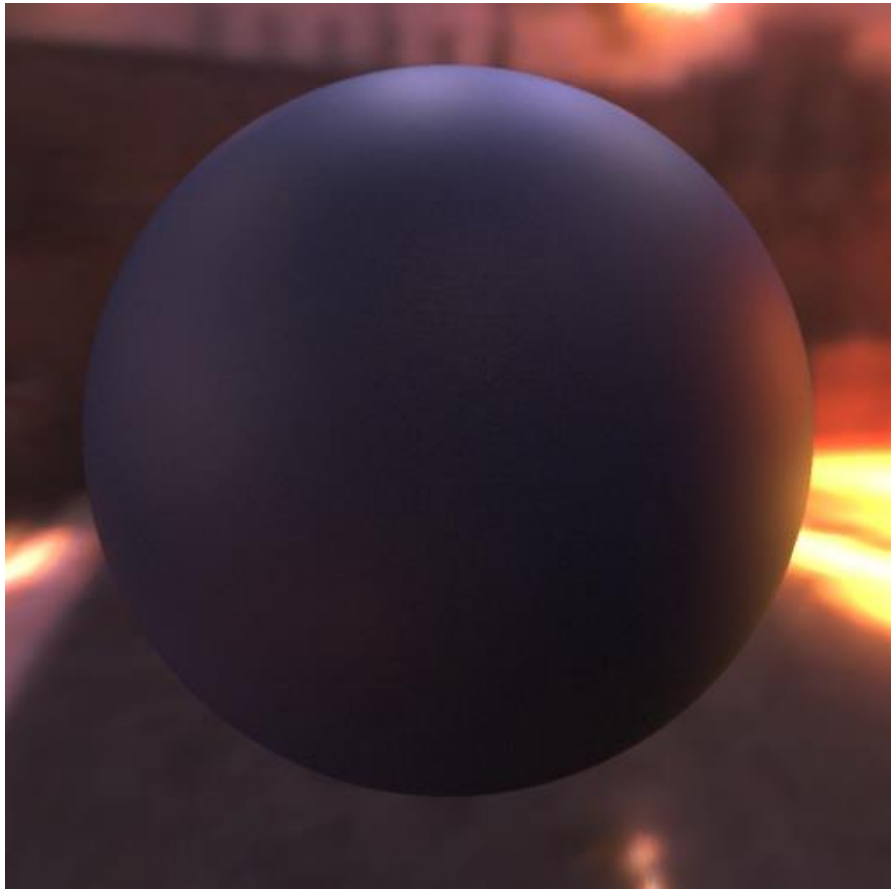
Courtesy of Mitsubishi Electric Researh Laboratories, Inc. Used with permission.

Finding the BRDF model parameters that best reproduce the real material

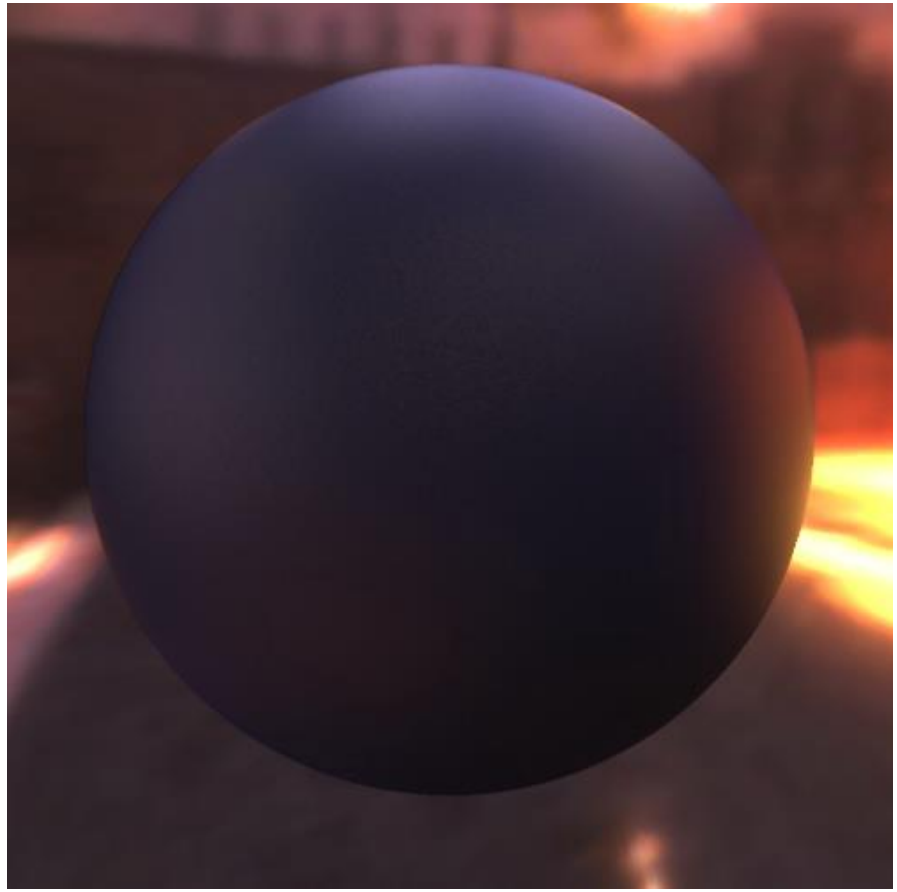
Material – Dark blue paint

Dark Blue Paint

Acquired data



Cook-Torrance



Courtesy of Mitsubishi Electric Researh Laboratories, Inc. Used with permission.

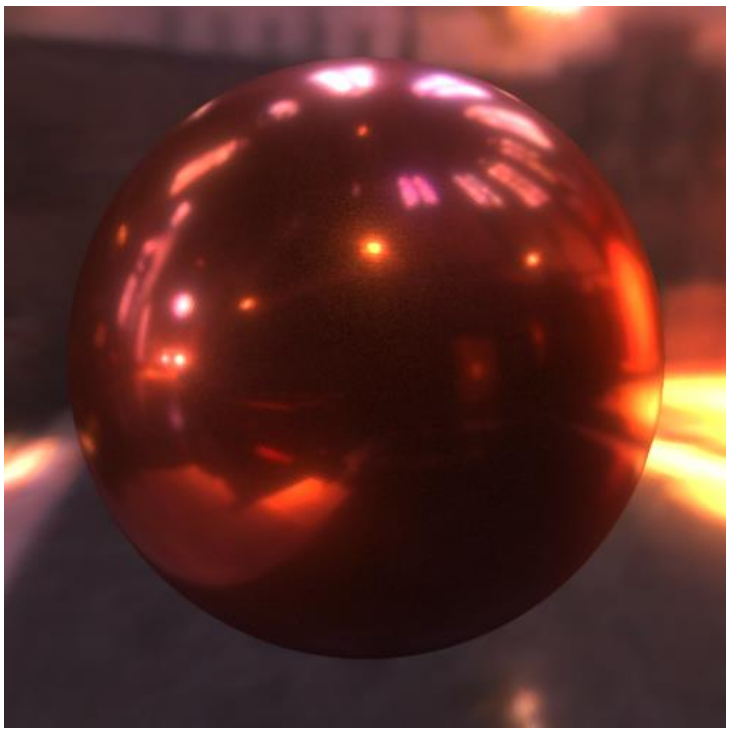
Finding the BRDF model parameters that best reproduce the real material

Material – Dark blue paint

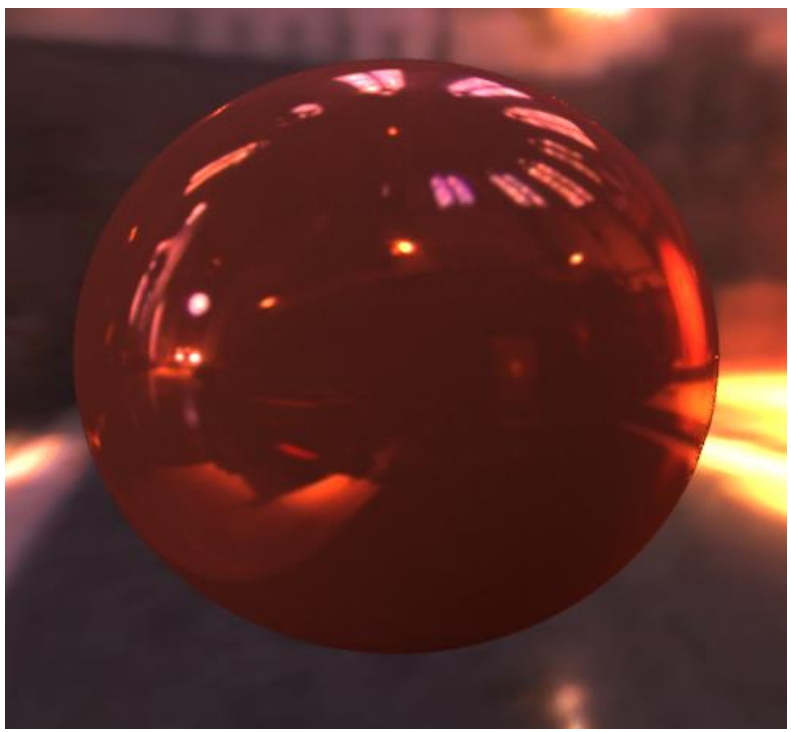
Observations

- Some materials impossible to represent with a single lobe

Acquired data



Cook-Torrance



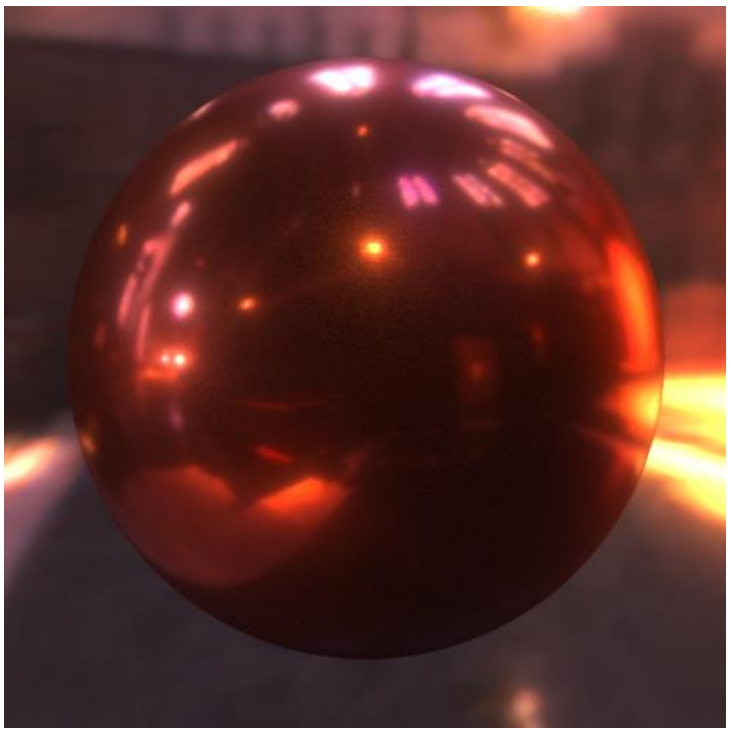
Material – Red Christmas Ball

Courtesy of Mitsubishi Electric Researh Laboratories, Inc. Used with permission.

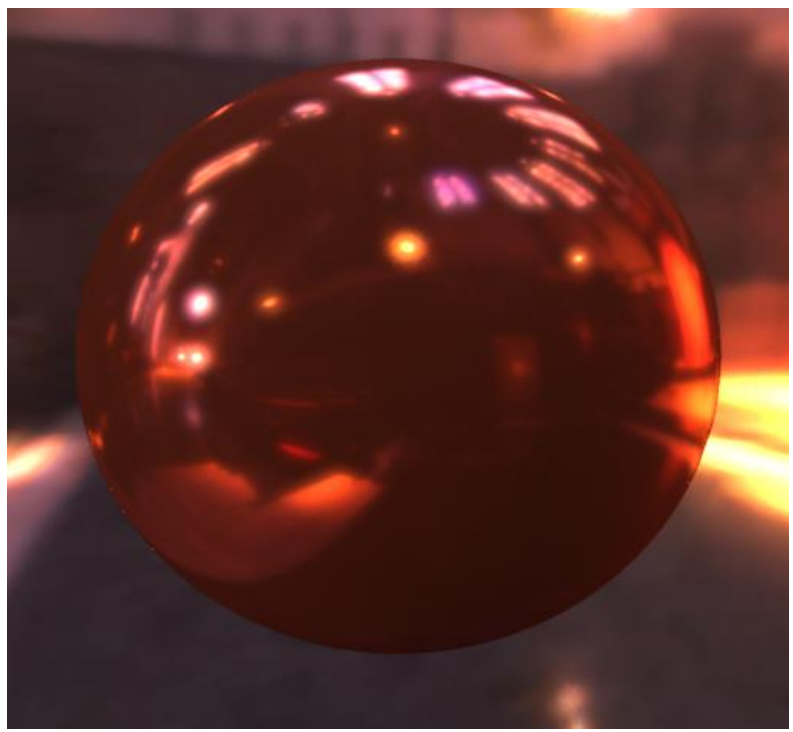
Adding a Second Lobe

- Some materials impossible to represent with a single lobe

Acquired data



Cook-Torrance 2 lobes



Material – Red Christmas Ball

Courtesy of Mitsubishi Electric Researh Laboratories, Inc. Used with permission.

Image-Based Acquisition

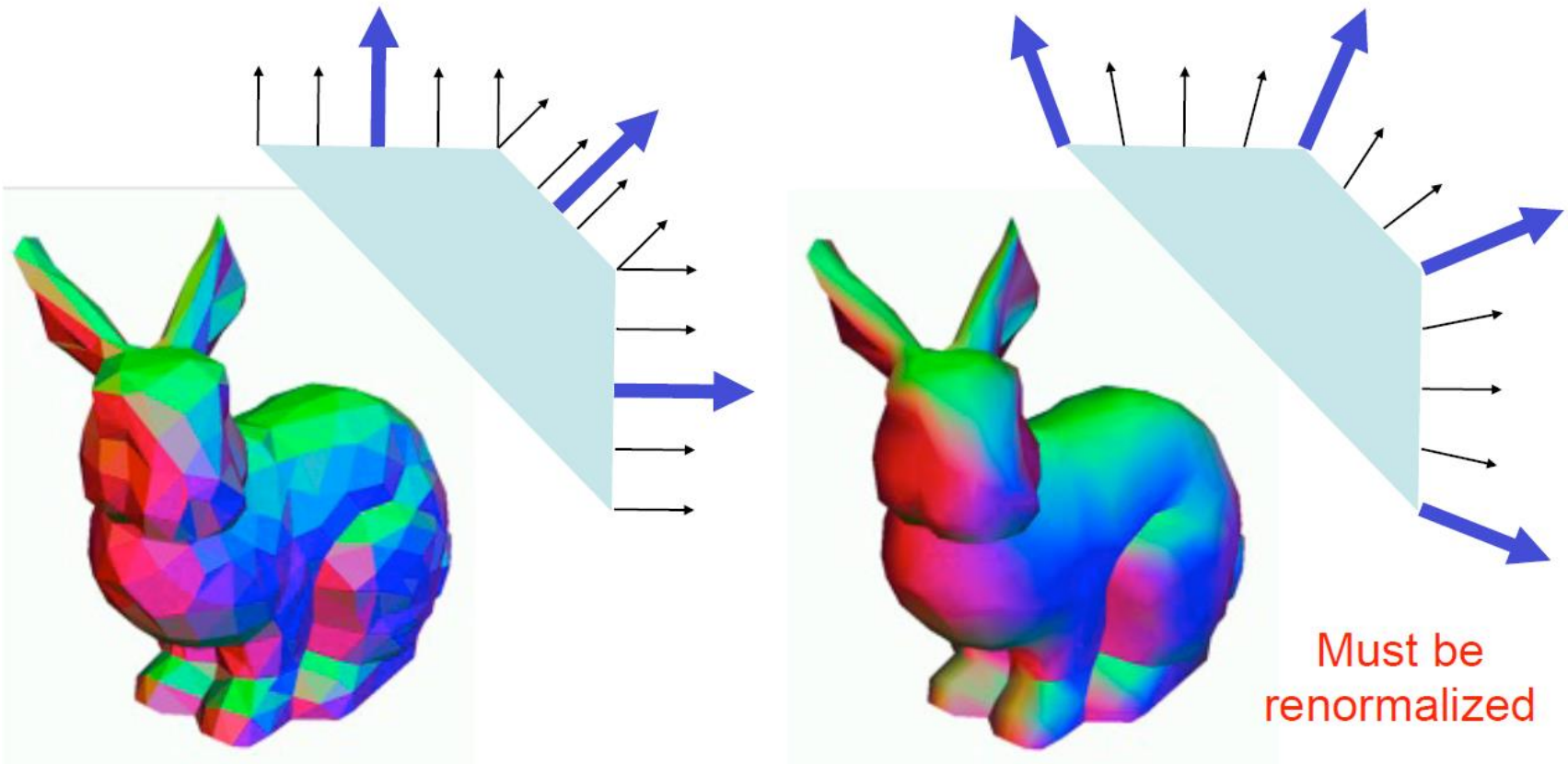
- A Data-Driven Reflectance Model, SIGGRAPH 2003
  - The data is available http://people.csail.mit.edu/wojciech/BRDFDatabase/

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Phong Normal Interpolation

(Not Phong *Shading*)

- Interpolate the average vertex normals across the face and use this in shading computations
  - Again, use barycentric interpolation!



That’s All for Today!

Images from the movie, “The Matrix,” removed due to copyright restrictions.

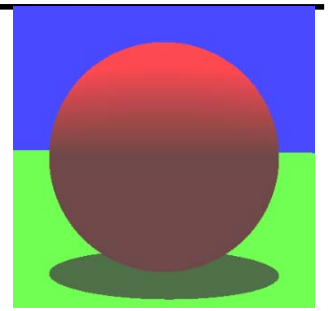
TIEA311 - Today in Jyväskylä

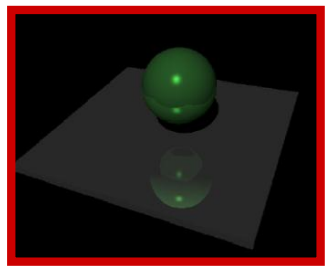
Last lecture plan:

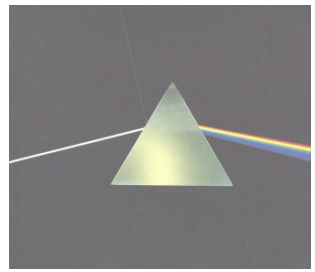
- Shading, texture mapping: Cover the principles up to Phong model and texture coordinates.
- Cherry-pick title slides from advanced stuff that we mostly defer to the follow-up course (starts next week) and/or future self-study.
- When **learning**, don't think about deadlines. **Think about learning!**
- My deadlines are flexible, since this course is about graphics and programming and **not about meeting time-to-market**. Leave that to project courses and traineeship periods!

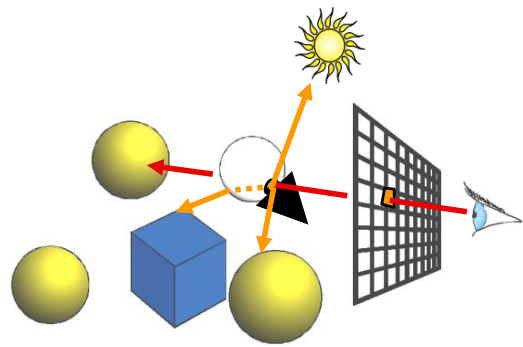
Overview of Today

- Shadows
- Reflection
- Refraction
- Recursive Ray Tracing



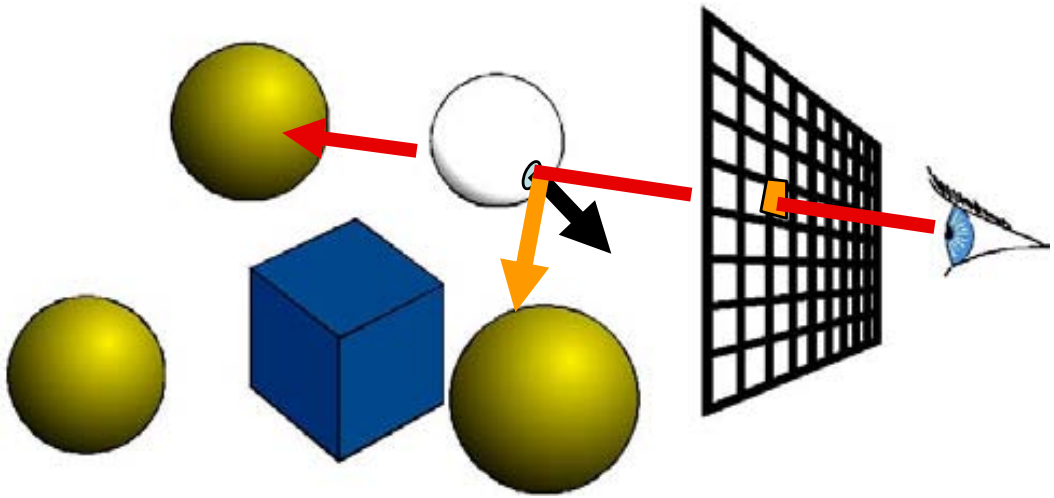


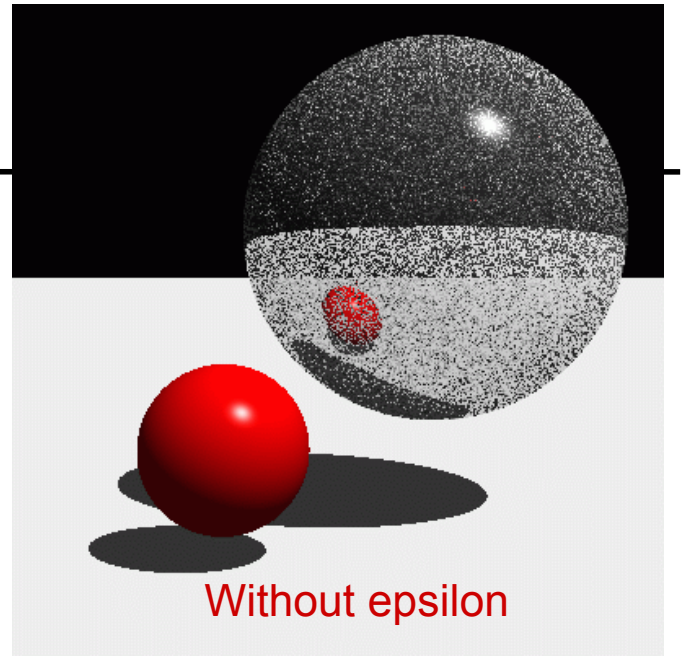


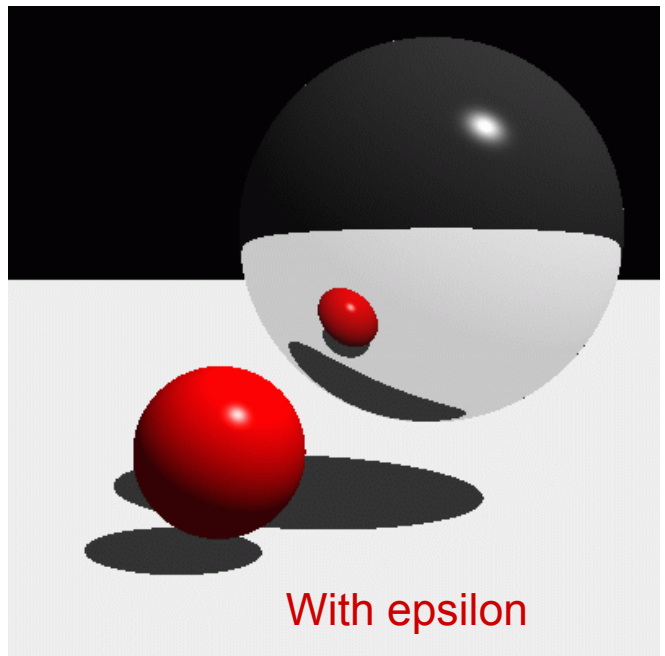


Mirror Reflection

- Cast ray symmetric with respect to the normal
- Multiply by reflection coefficient k_s (color)
- Don't forget to add epsilon to the ray!

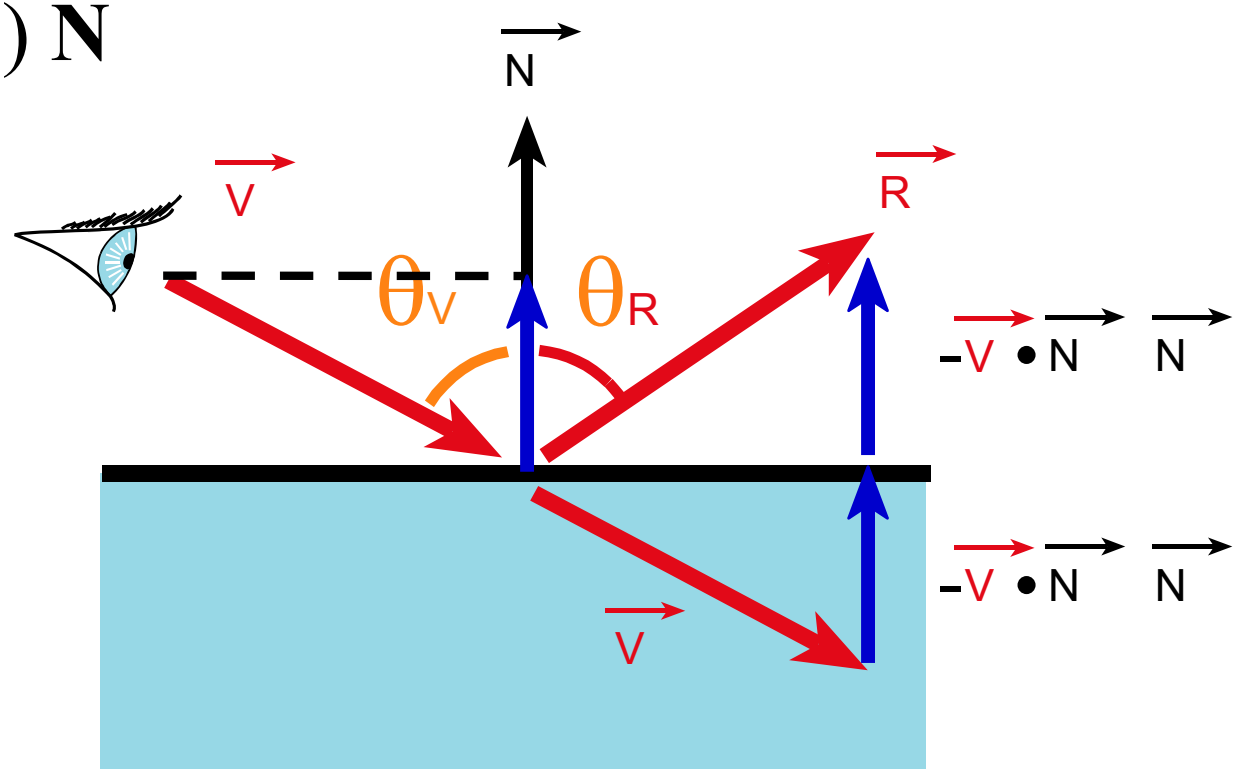






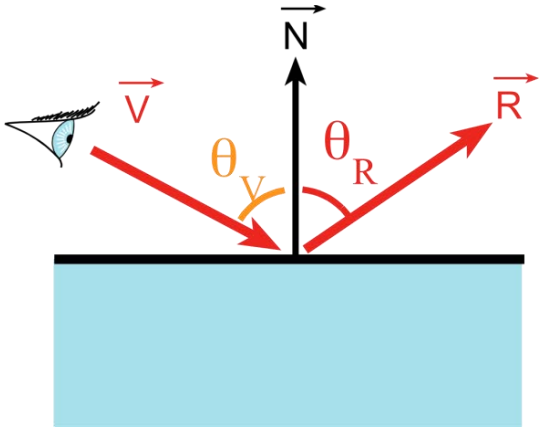
Perfect Mirror Reflection

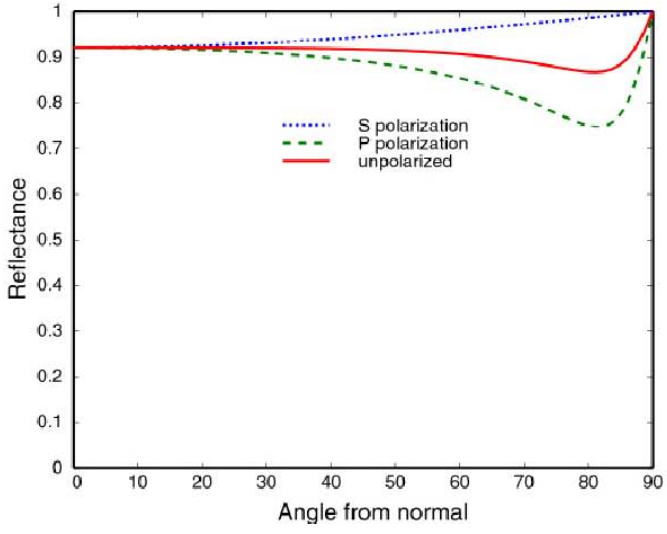
- Reflection angle = view angle
  - Normal component is negated
  - Remember particle collisions?
- $\mathbf{R} = \mathbf{V} - 2\,(\mathbf{V} \cdot \mathbf{N})\,\mathbf{N}$



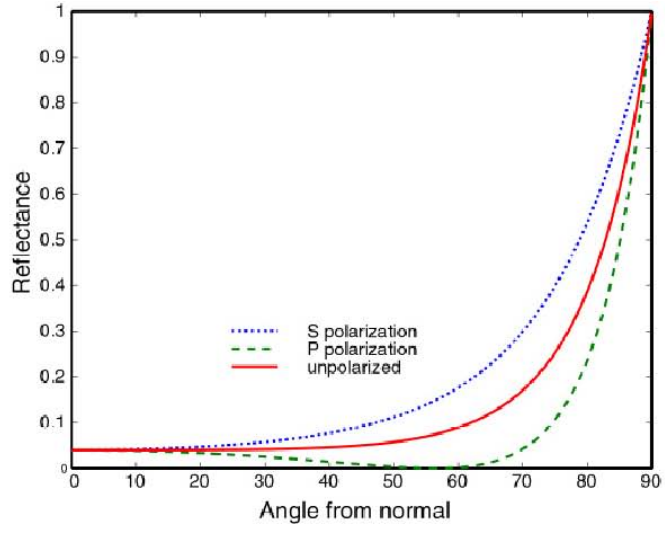
Amount of Reflection

- Traditional ray tracing (hack)
  - Constant k_s
- More realistic (we'll do this later):
  - Fresnel reflection term (more reflection at grazing angle)
  - Schlick's approximation: $R(\theta)=R_0+(1-R_0)(1-\cos\theta)^5$
- Fresnel makes a big difference!





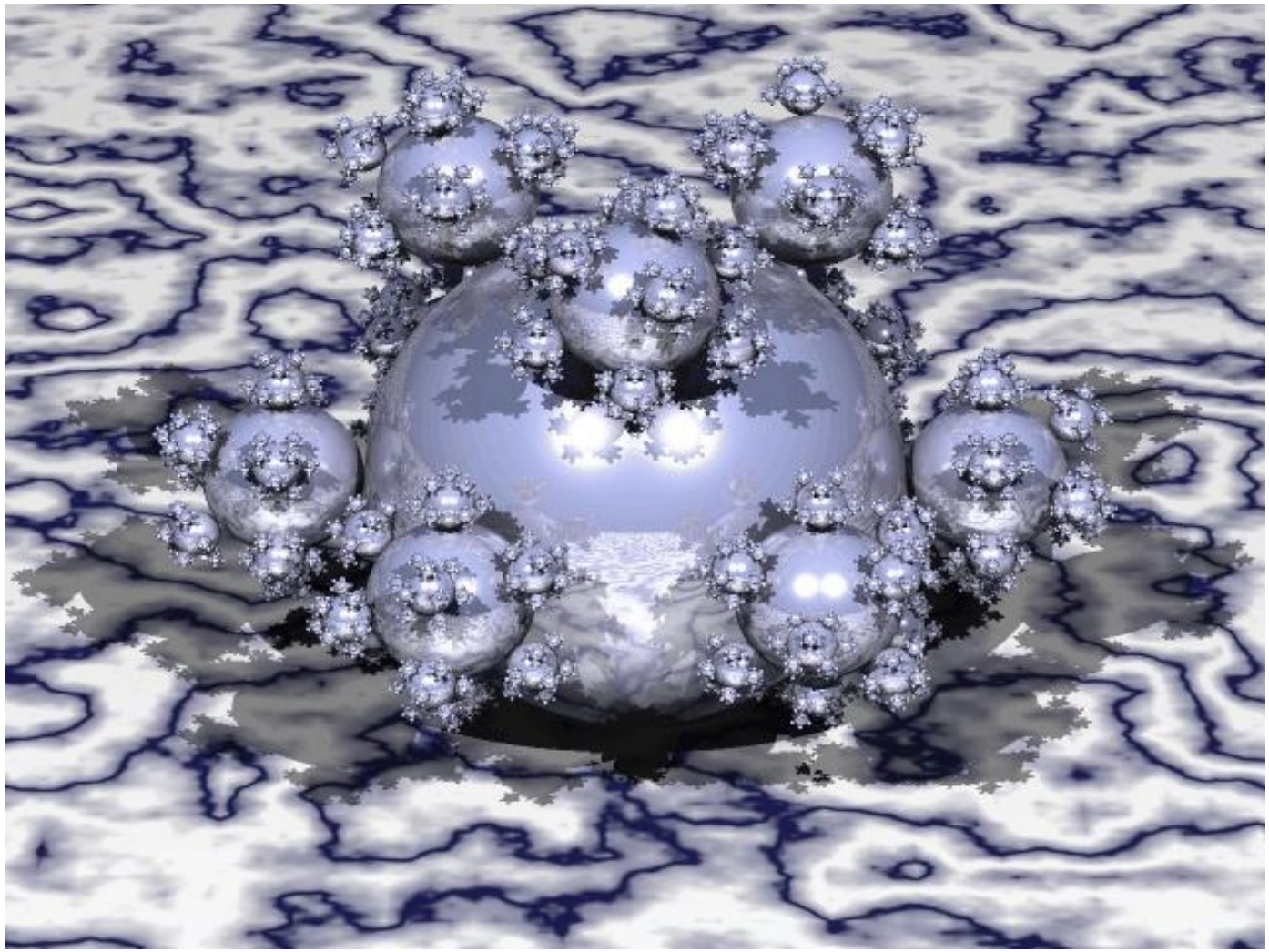
metal



Dielectric (glass)

Questions?

"Sphereflake" fractal

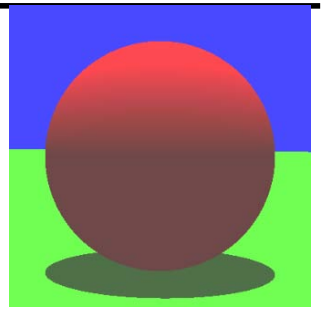


Henrik Wann Jensen

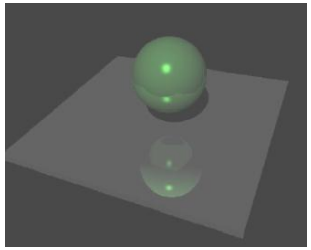
Courtesy of Henrik Wann Jensen. Used with permission.

Overview of Today

- Shadows

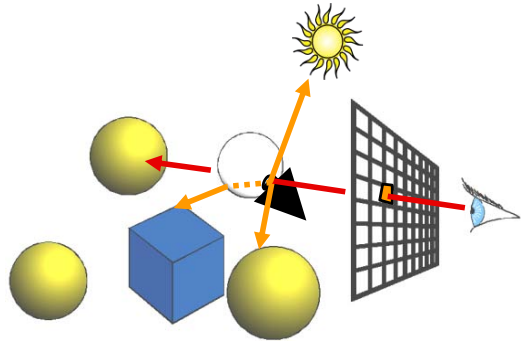


- Reflection



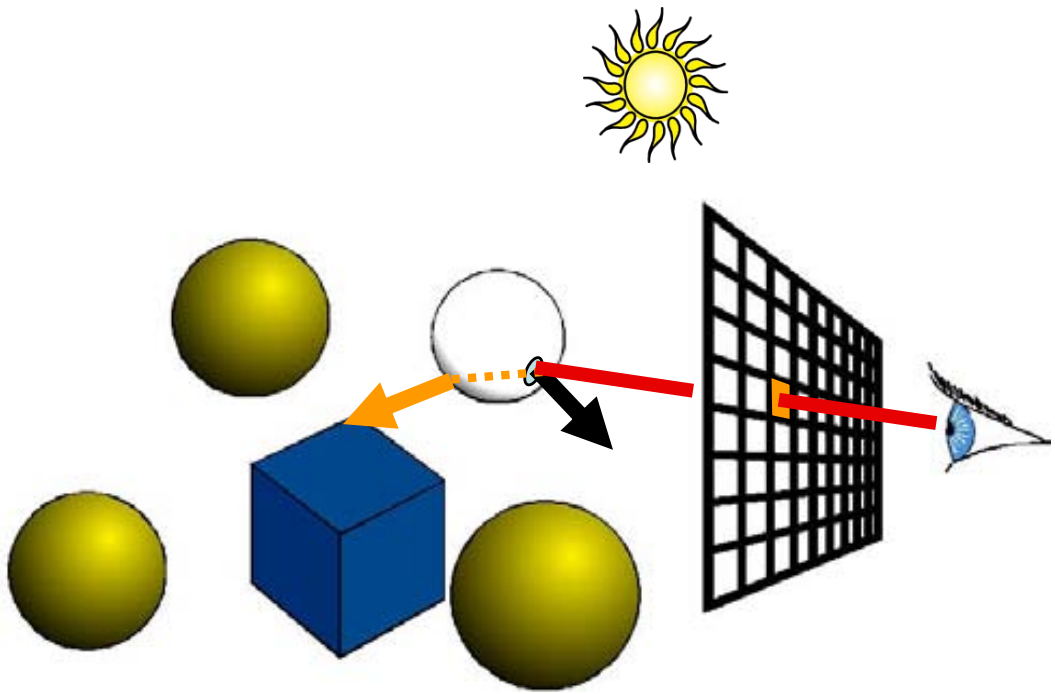
- Refraction

- Recursive Ray Tracing

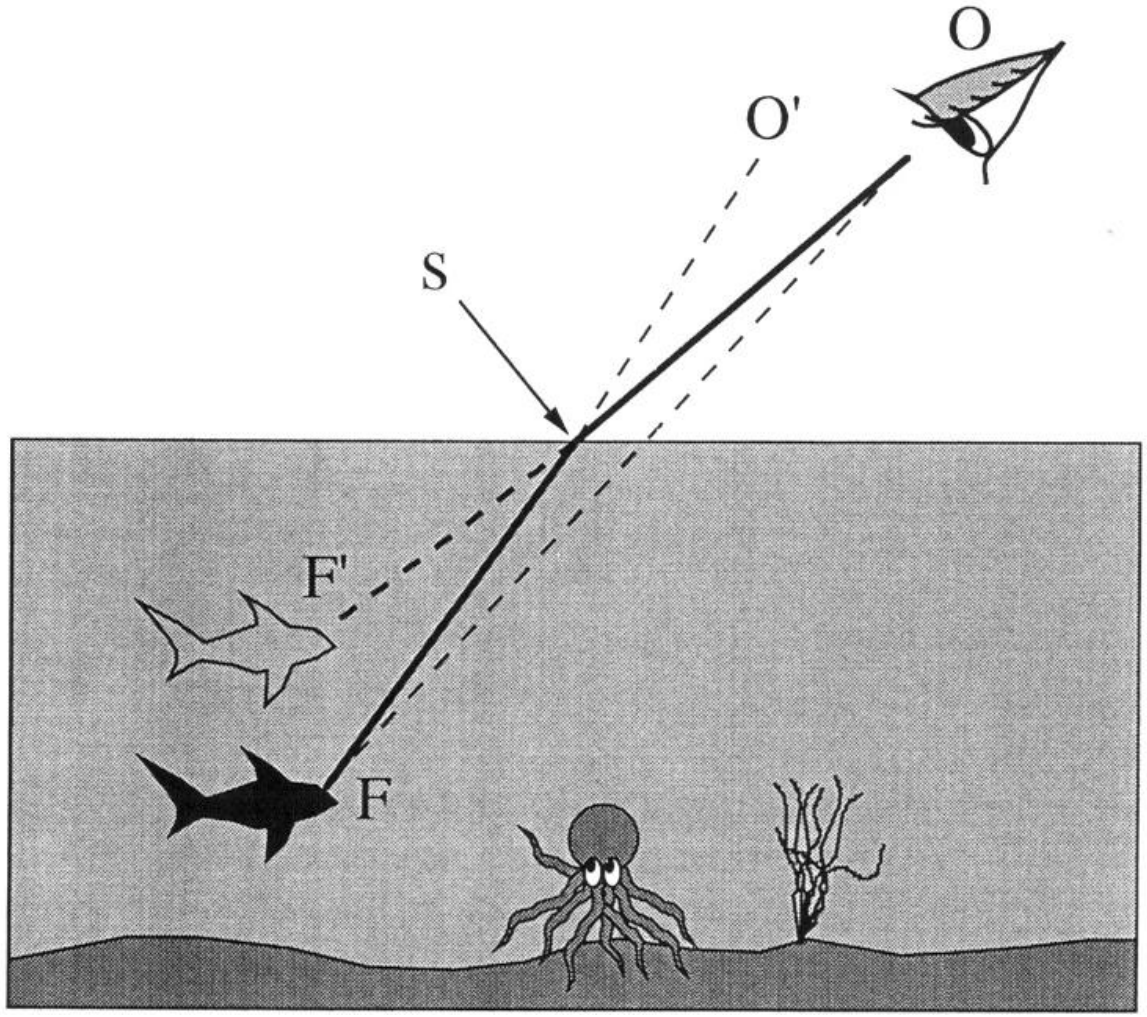


Transparency (Refraction)

- Cast ray in refracted direction
- Multiply by transparency coefficient k_t (color)

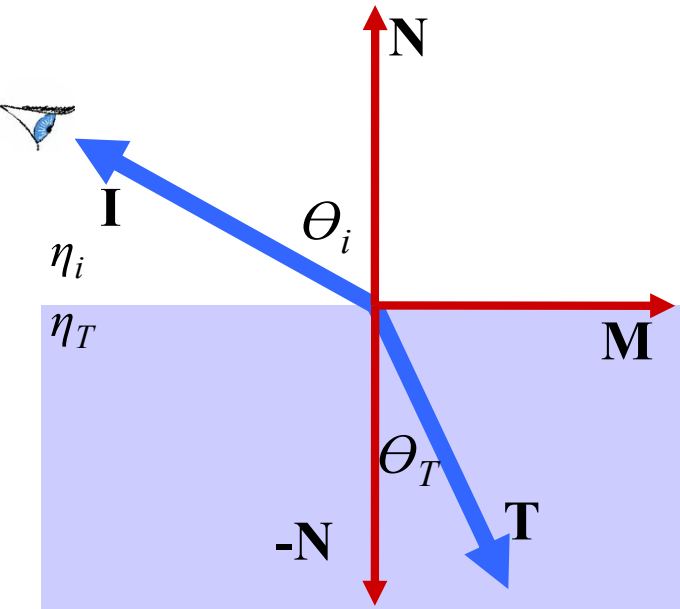


Qualitative Refraction



© Cambridge University Press. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Refraction



Material 1, index of refraction η_i

Material 2, index of refraction η_T

Snell-Descartes Law:

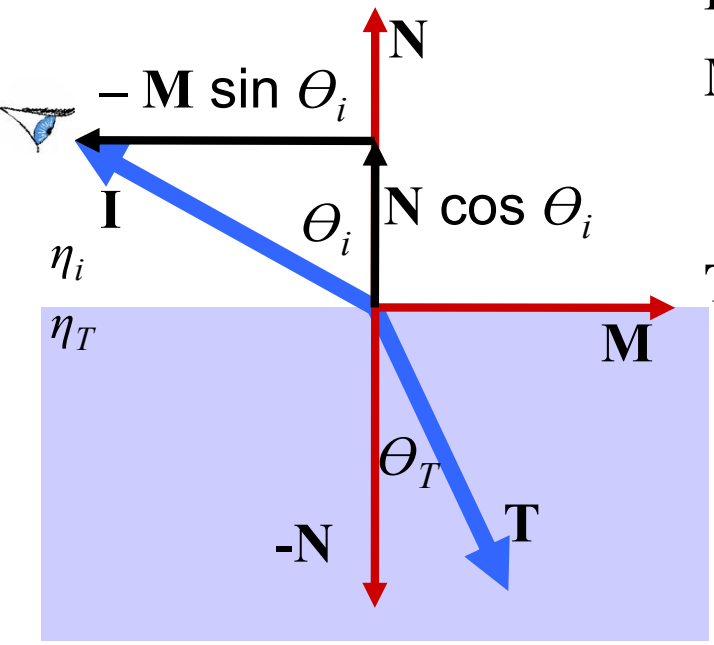
$$n_i \sin\theta_i = n_T \sin\theta_T$$

$$\frac{\sin\theta_T}{\sin\theta_i} = \frac{n_i}{n_T} = n_r$$

Refracted direction **T**?

Relative index of refraction

Refraction



$$\mathbf{I} = \mathbf{N}\cos\Theta_i - \mathbf{M}\sin\Theta_i$$

$$\mathbf{M} = (\mathbf{N}\cos\Theta_i - \mathbf{I}) / \sin\Theta_i$$

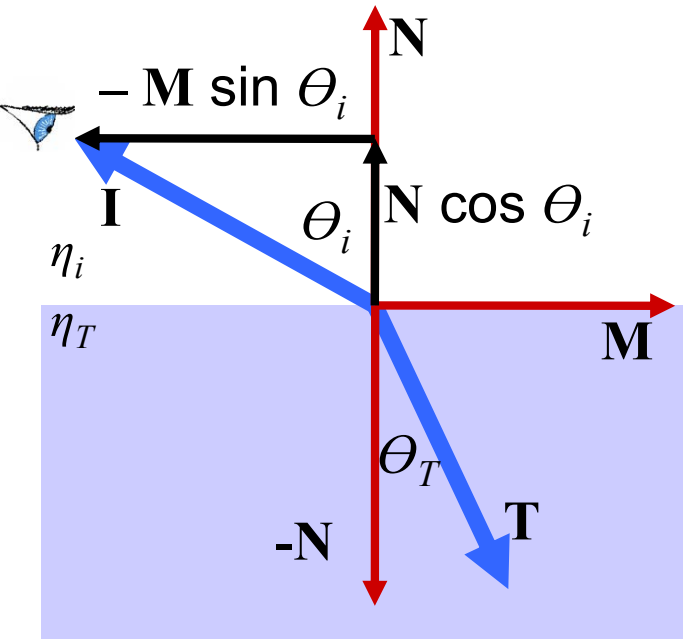
$$\begin{aligned}
\mathbf{T} &= -\mathbf{N}\cos\Theta_T + \mathbf{M}\sin\Theta_T \\
&= -\mathbf{N}\cos\Theta_T + (\mathbf{N}\cos\Theta_i - \mathbf{I})\sin\Theta_T / \sin\Theta_i \quad \textit{Plug M} \\
&= -\mathbf{N}\cos\Theta_T + (\mathbf{N}\cos\Theta_i - \mathbf{I})\,\eta_r \quad \textit{let's get rid of the cos \& sin} \\
&= [\,\eta_r\cos\Theta_i - \cos\Theta_T\,]\,\mathbf{N} - \eta_r\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{1-\sin^2\Theta_T}\,]\,\mathbf{N} - \eta_r\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{1-\eta_r^2\sin^2\Theta_i}\,]\,\mathbf{N} - \eta_r\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{1-\eta_r^2(1-\cos^2\Theta_i)}\,]\,\mathbf{N} - \eta_r\mathbf{I} \\
&= [\,\eta_r(\mathbf{N}\cdot\mathbf{I}) - \sqrt{1-\eta_r^2(1-(\mathbf{N}\cdot\mathbf{I})^2)}\,]\,\mathbf{N} - \eta_r\mathbf{I}
\end{aligned}$$

Snell-Descartes Law:

$$n_i \sin\theta_i = n_T \sin\theta_T$$

$$\frac{\sin\theta_T}{\sin\theta_i} = \frac{n_i}{n_T} = n_r$$

Refraction



$$\mathbf{I} = \mathbf{N}\cos\Theta_i - \mathbf{M}\sin\Theta_i$$

$$\mathbf{M} = (\mathbf{N}\cos\Theta_i - \mathbf{I}) / \sin\Theta_i$$

- **Total internal reflection** when the square root is imaginary (no refraction, just reflection)

Snell-Descartes Law:

$$n_i \sin\theta_i = n_T \sin\theta_T$$

$$\frac{\sin\theta_T}{\sin\theta_i} = \frac{n_i}{n_T} = n_r$$

$$= [\, \eta_r (\mathbf{N}\cdot\mathbf{I}) - \sqrt{1 - \eta_r^2 (1 - (\mathbf{N}\cdot\mathbf{I})^2)}\,]\, \mathbf{N} - \eta_r \mathbf{I}$$

Total Internal Reflection

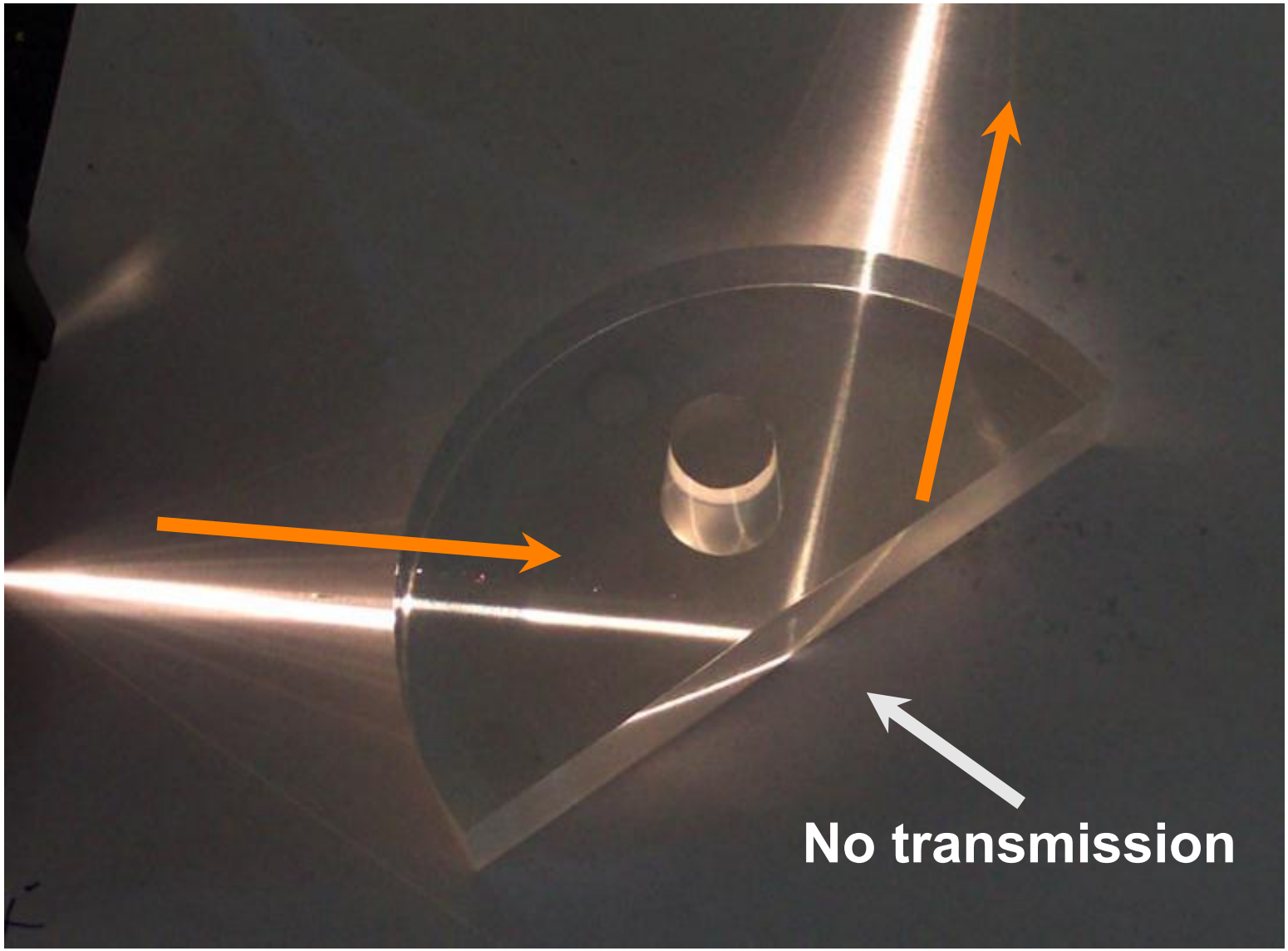


Image courtesy of Frazzydee on Wikimedia Commons. License: CC-BY-SA. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Total Internal Reflection

Fig. 3.7A The optical manhole. From under water, the entire celestial hemisphere is compressed into a circle only 97.2° across. The dark boundary defining the edges of the manhole is not sharp due to surface waves. The rays are analogous to the crepuscular type seen in hazy air, Section 1.9. (Photo by D. Granger)

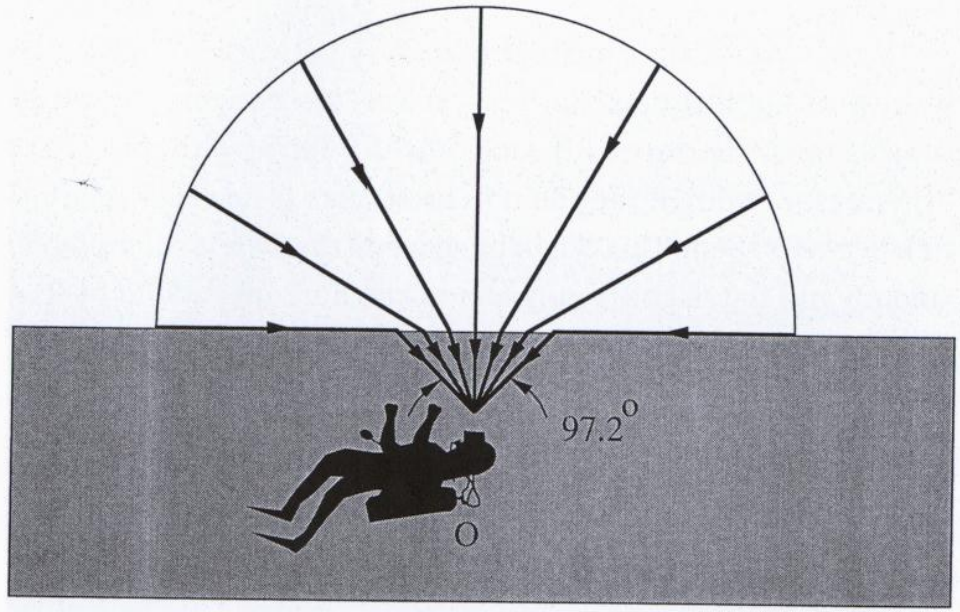
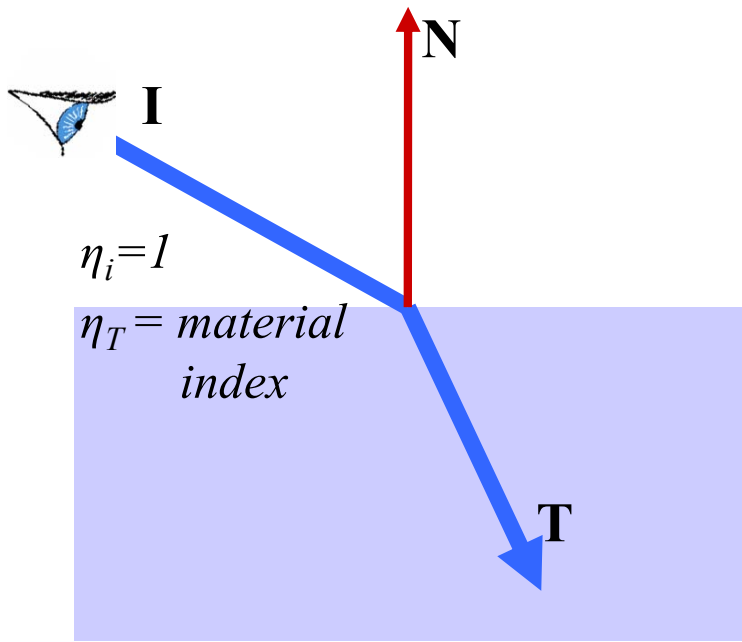


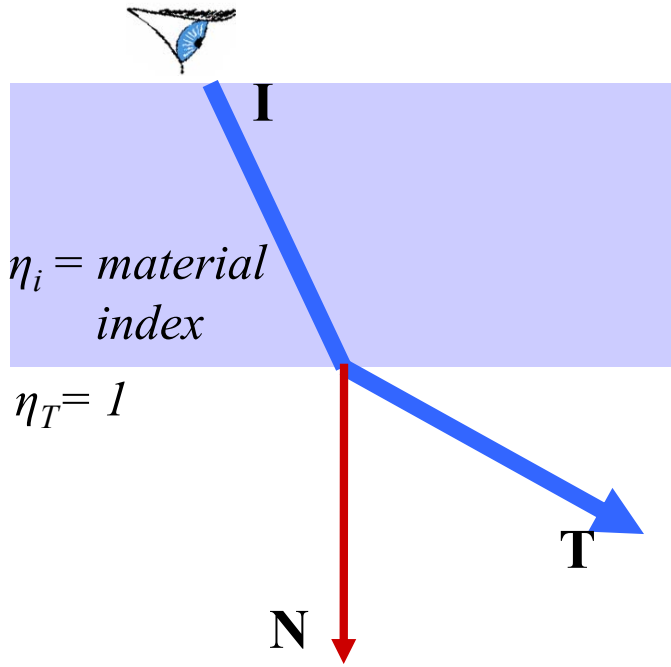
Fig. 3.7B The optical manhole. Light from the horizon (angle of incidence = 90°) is refracted downward at an angle of 48.6°. This compresses the sky into a circle with a diameter of 97.2° instead of its usual 180°.

© Cambridge University Press. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Refraction & Sidedness of Objects

- Make sure you know whether you're entering or leaving the transmissive material:

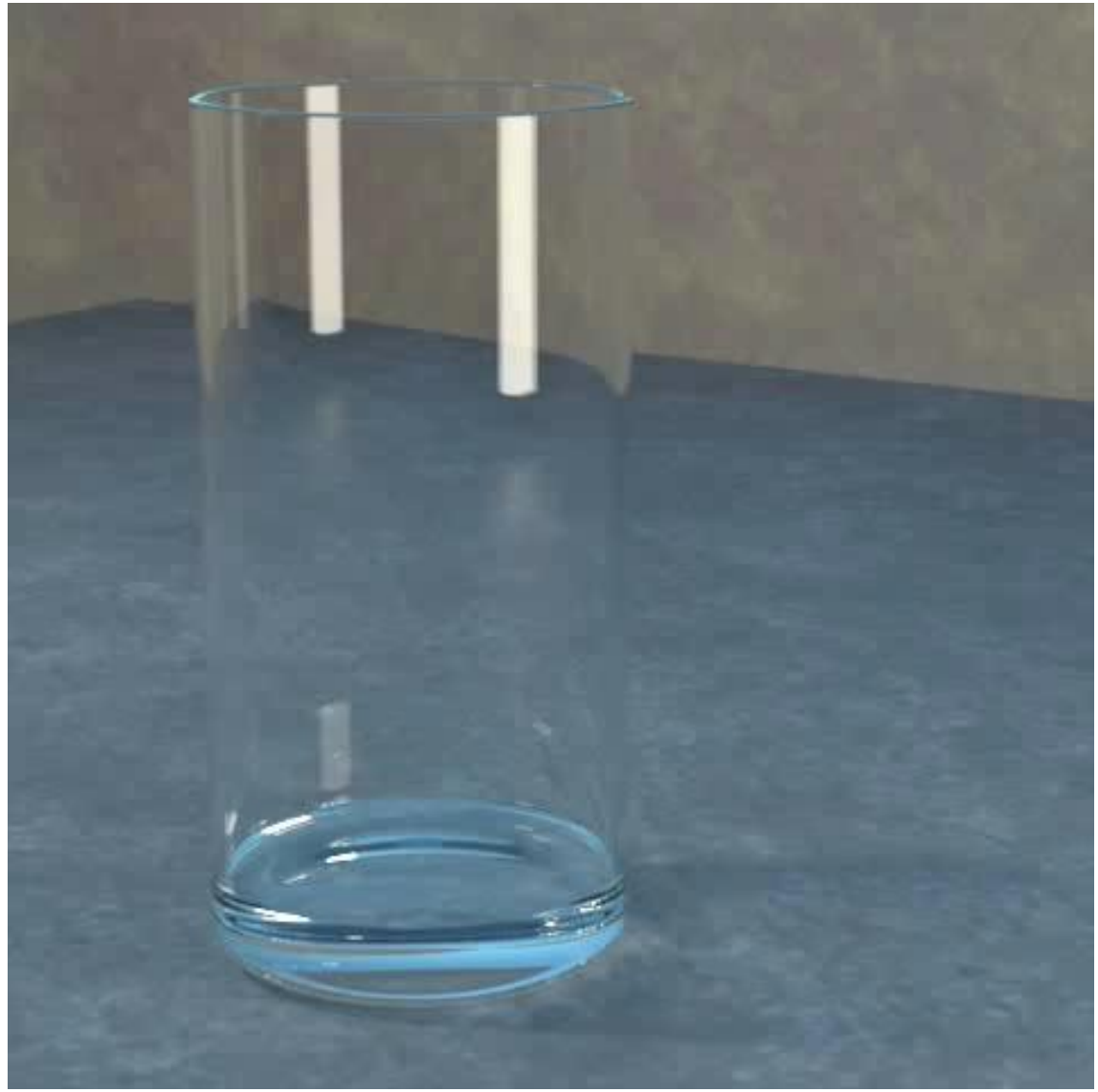




- Note: We won't ask you to trace rays through intersecting transparent objects **:-)**

Cool Refraction Demo

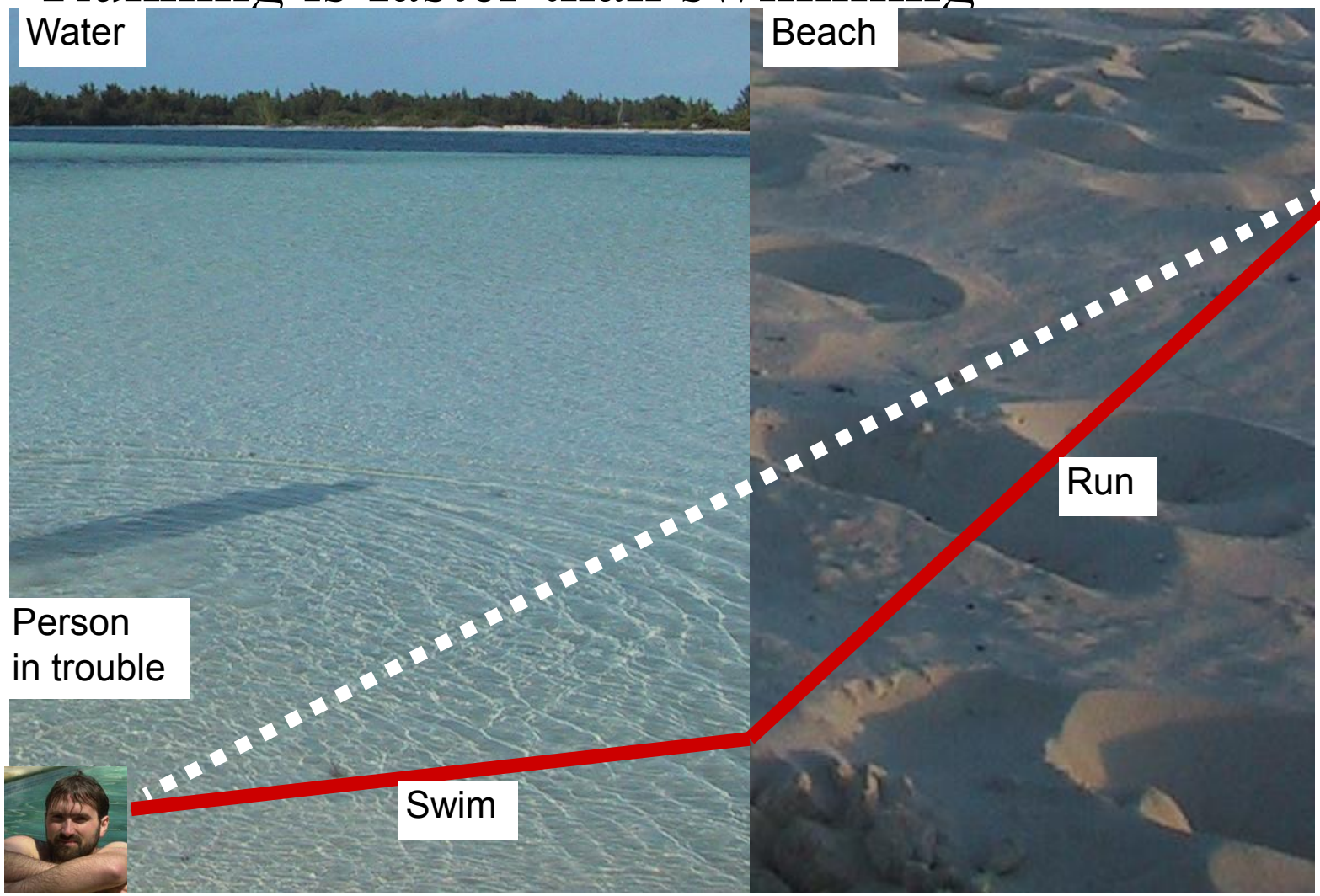
- Enright, D., Marschner, S. and Fedkiw, R., SIGGRAPH 2002



© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Refraction and the Lifeguard Problem

- Running is faster than swimming



Lifeguard

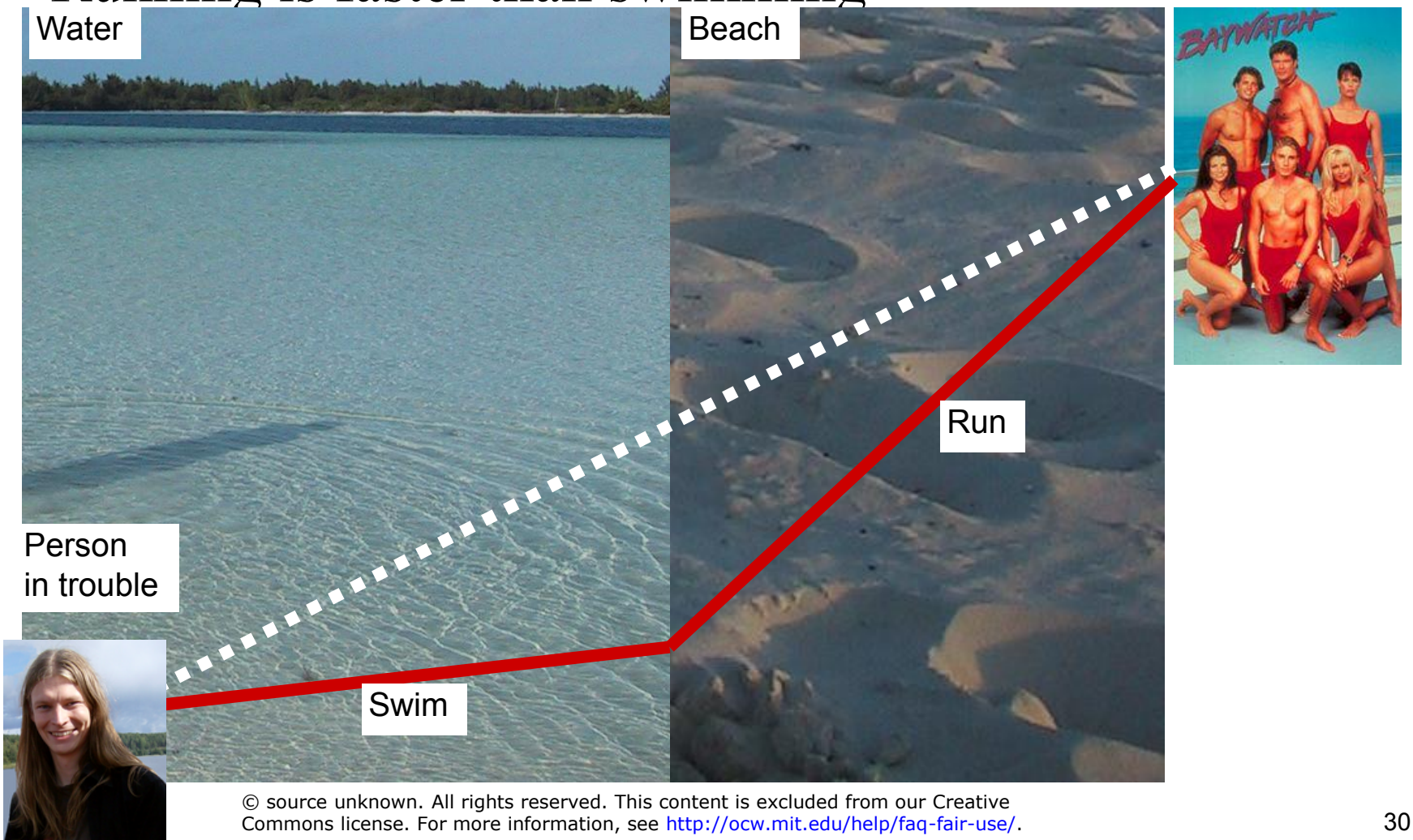
© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

TIEA311 - Today in Jyväskylä

Wait a moment. . . who is in trouble on this course?

Refraction and the Lifeguard Problem

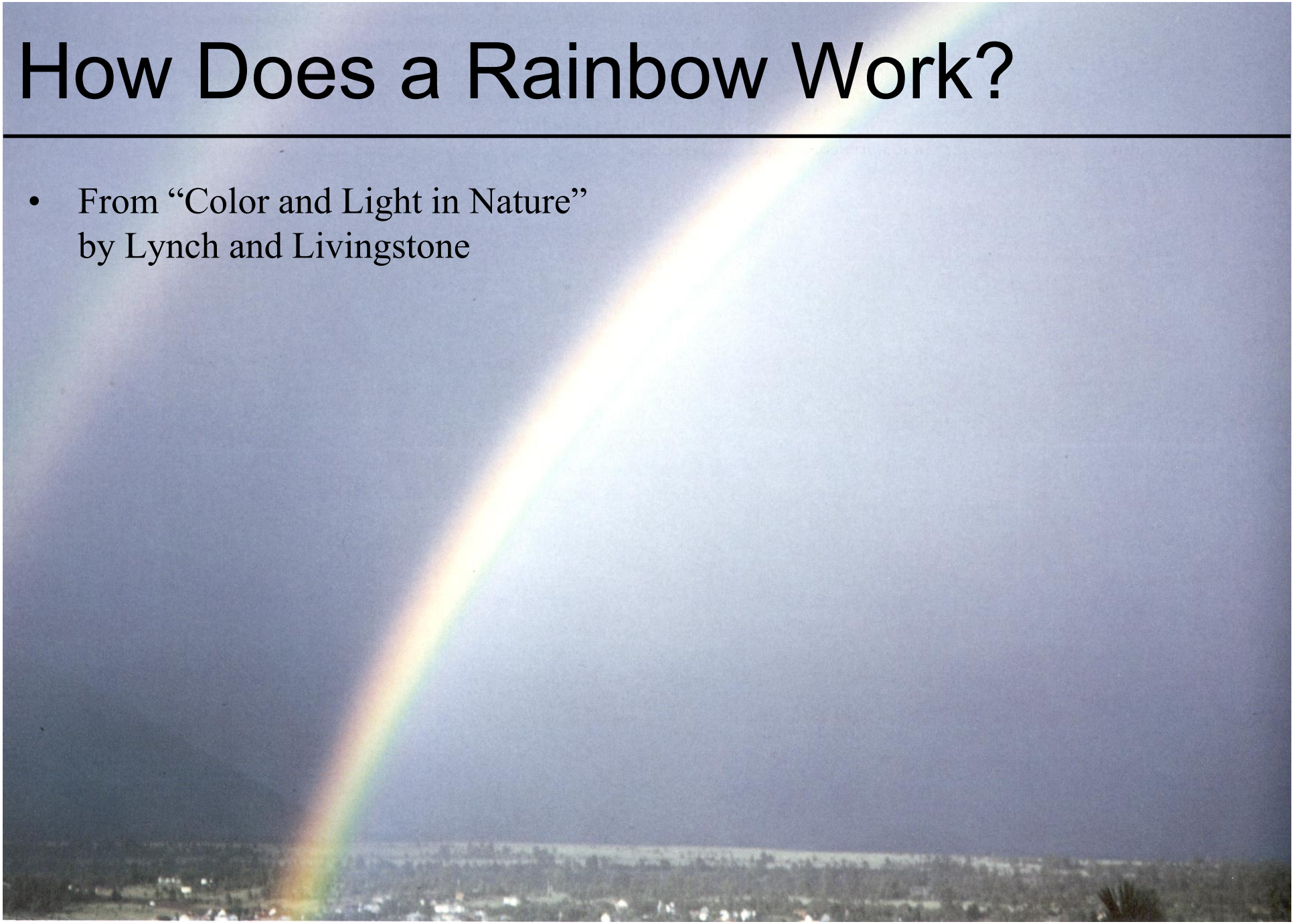
- Running is faster than swimming



© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

How Does a Rainbow Work?

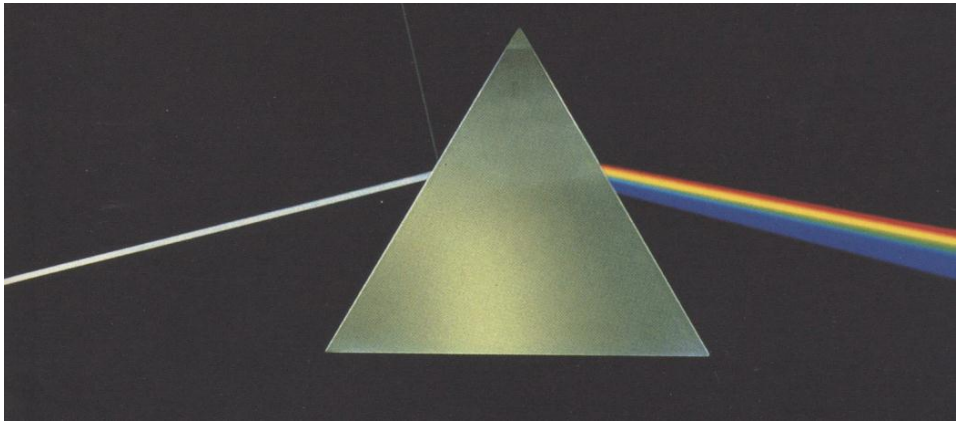
- From “Color and Light in Nature” by Lynch and Livingstone



© Cambridge University Press. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

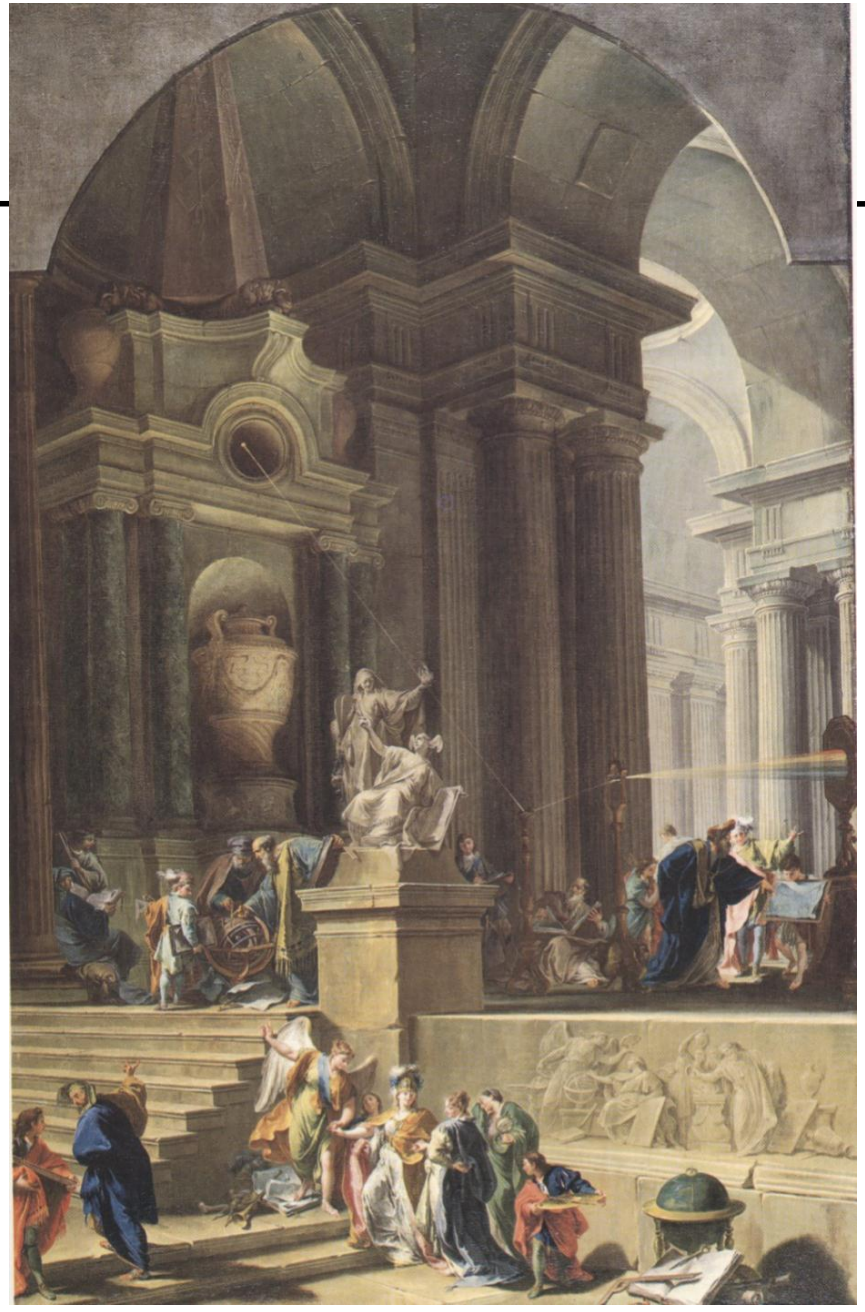
Wavelength

- Refraction is wavelength-dependent (dispersion)
  - Refraction increases as the wavelength of light decreases
  - violet and blue experience more bending than orange and red
- Newton's prism experiment
- **Usually ignored in graphics**



Pink Floyd, *The Dark Side of the Moon*

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

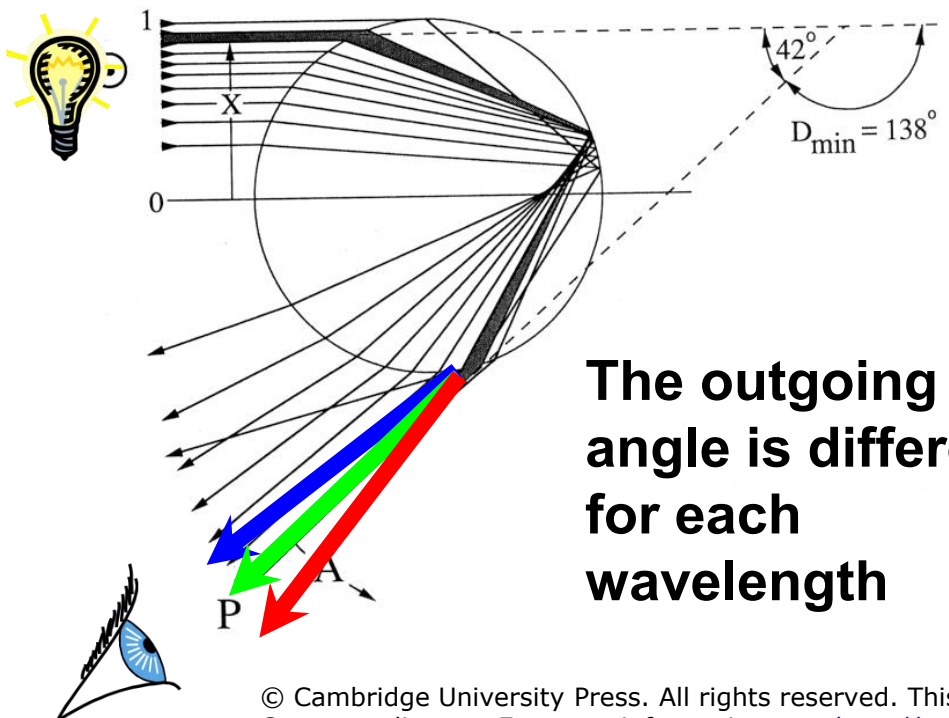


Pittoni, 1725, Allegory to Newton

© The Fitzwilliam Museum. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Rainbow

- Rainbow is caused by
  refraction + internal reflection + refraction
- Maximum for angle around 42 degrees
- Refraction depends on wavelength (dispersion)



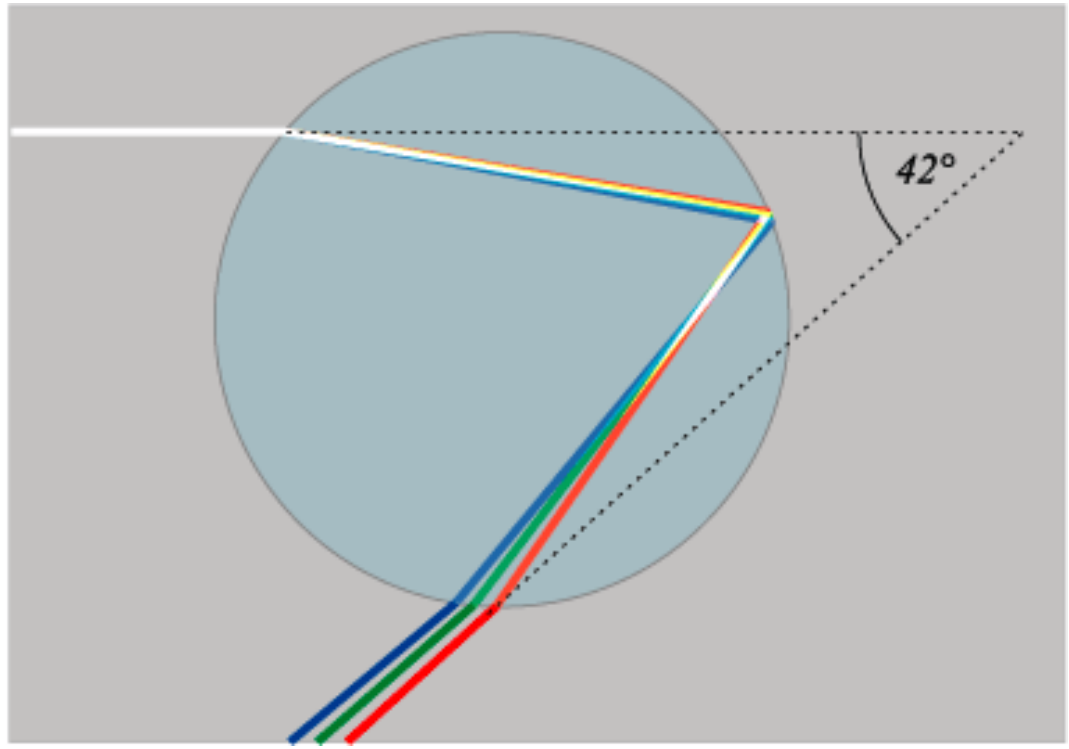
The outgoing angle is different for each wavelength

"Color and Light in Nature" by Lynch and Livingstone

© Cambridge University Press. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Rainbow

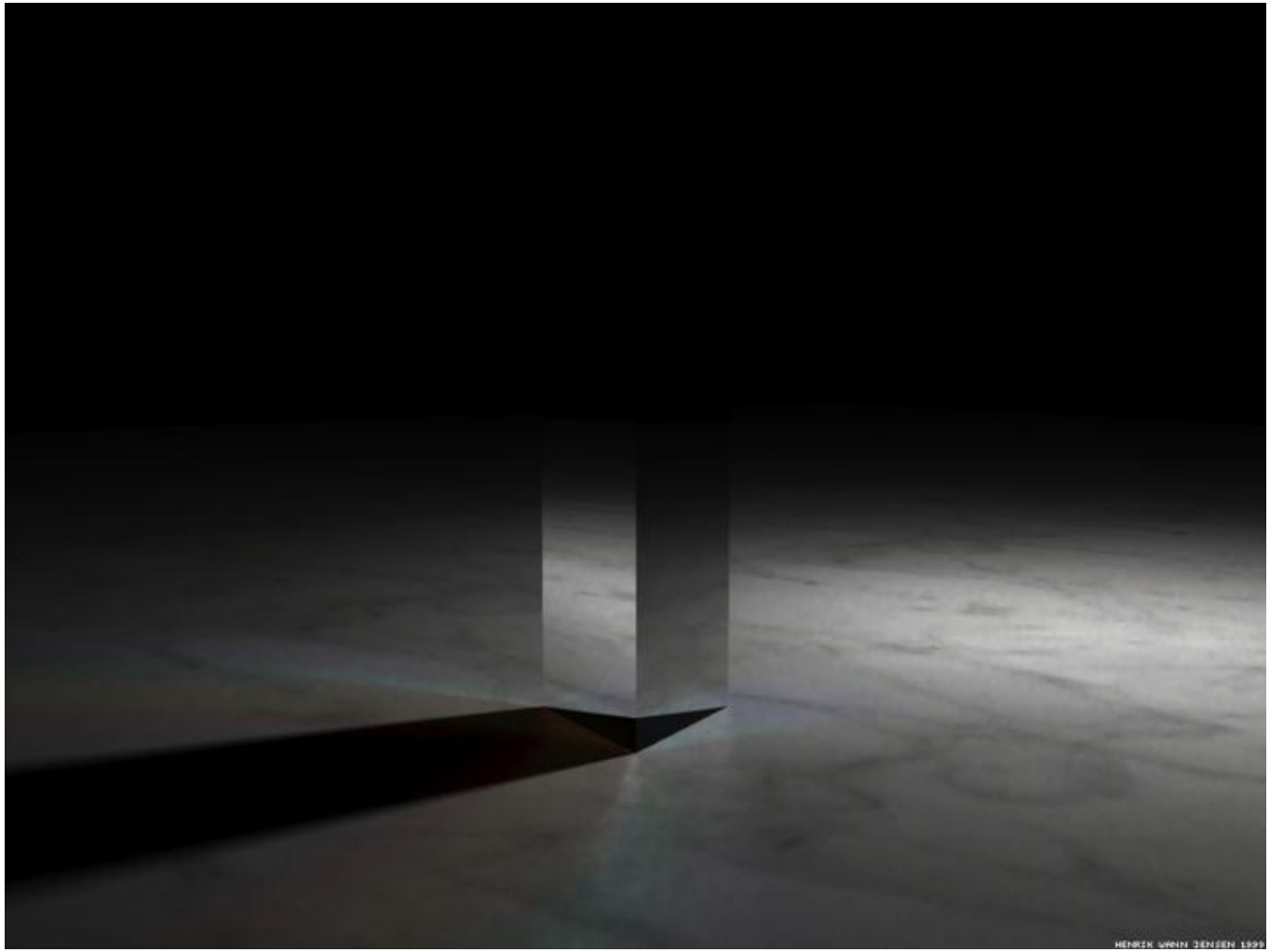
- Rainbow is caused by refraction + internal reflection + refraction
- Maximum for angle around 42 degrees
- Refraction depends on wavelength (dispersion)



This image is in the public domain. Source: Wikipedia.

Dispersion

- Image by Henrik Wann Jensen using Photon Mapping



Courtesy of Henrik Wann Jensen. Used with permission.

Application: CAD for lenses

- Has revolutionized lens design
  - E.g. zoom lenses are good now

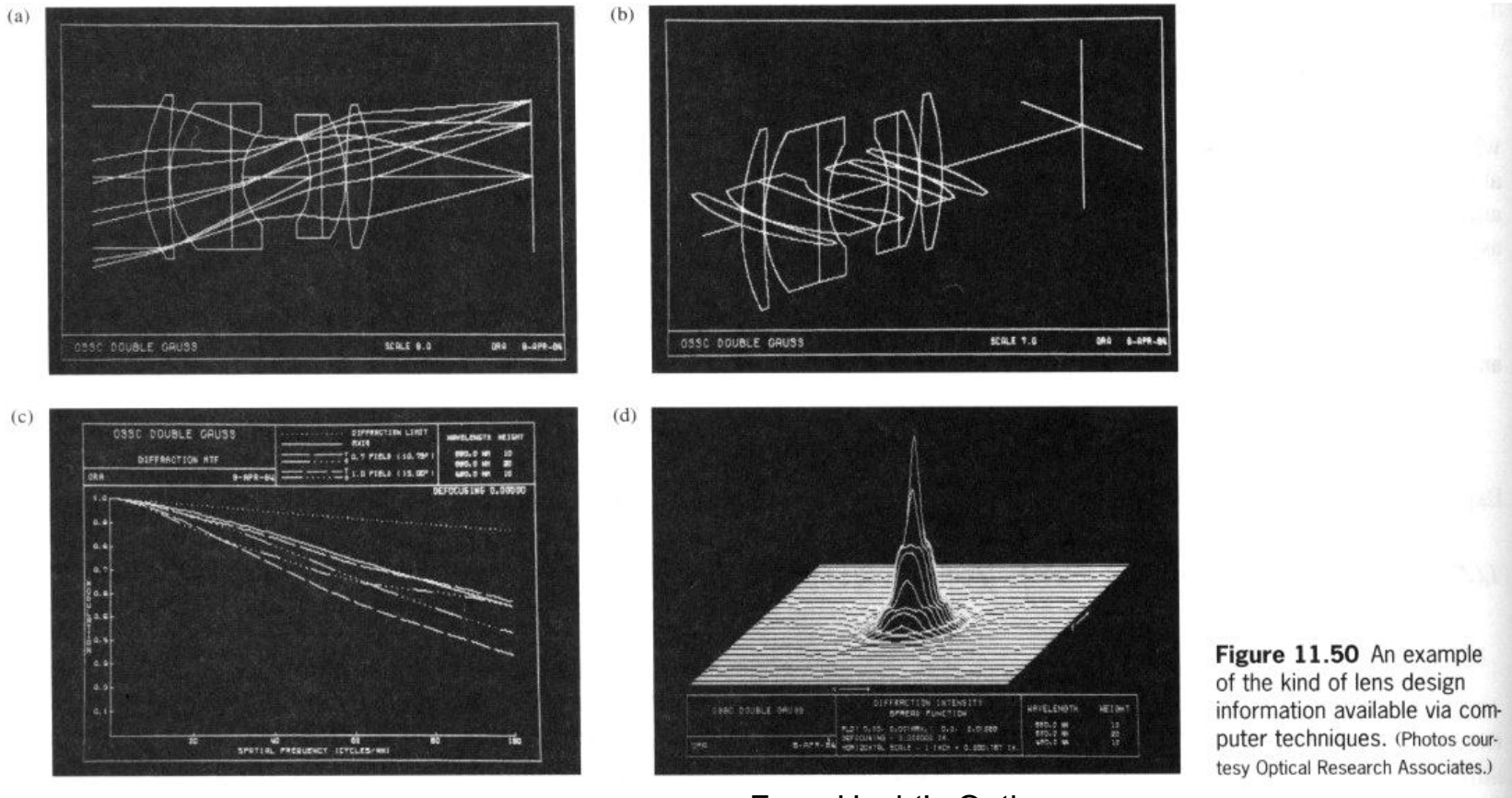


Figure 11.50 An example of the kind of lens design information available via computer techniques. (Photos courtesy Optical Research Associates.)

From Hecht's Optics

© Addison-Wesley. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Lens design by Ray Tracing

- Used to be done manually, by rooms full of engineers who would trace rays.
- Now software, e.g. Zemax
- More in 6.815/6.865 Computational Photography

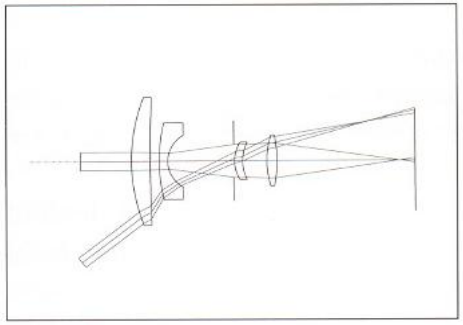


Figure-5

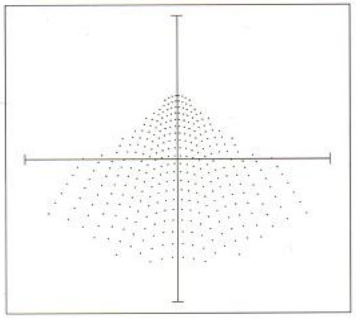


Figure-8

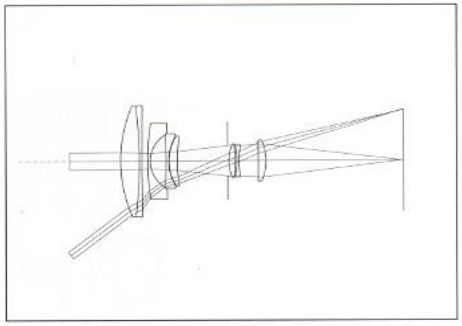


Figure-6

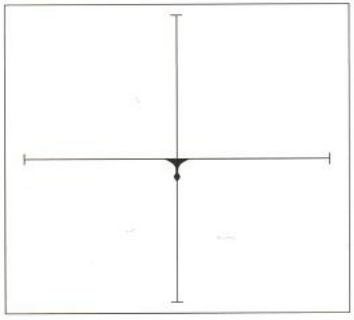


Figure-9

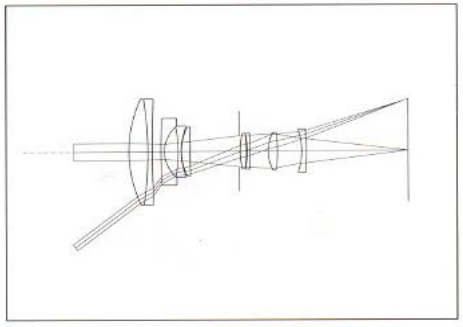


Figure-7

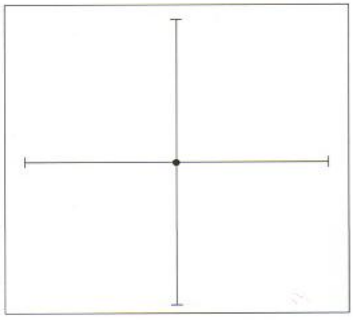


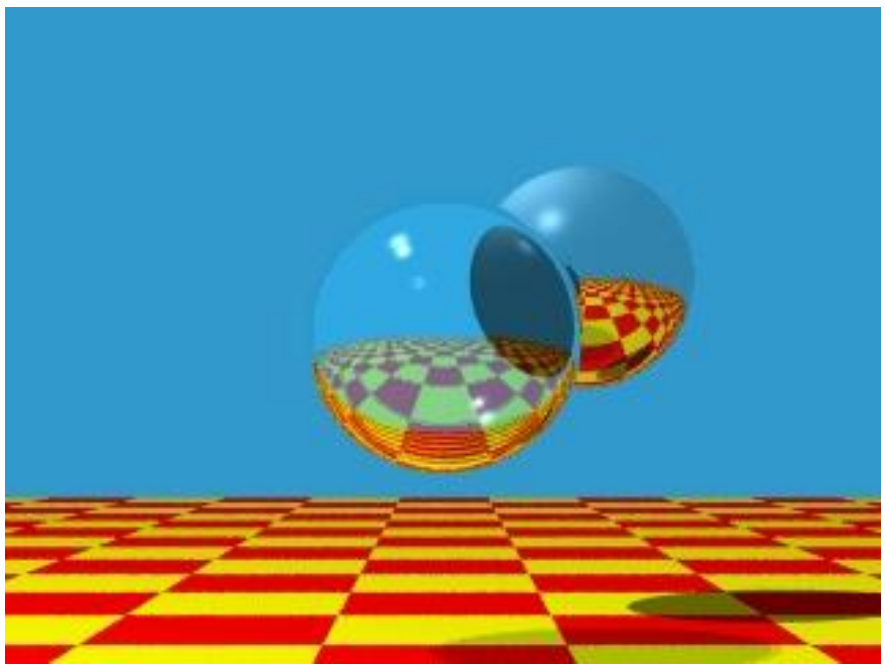
Figure-10

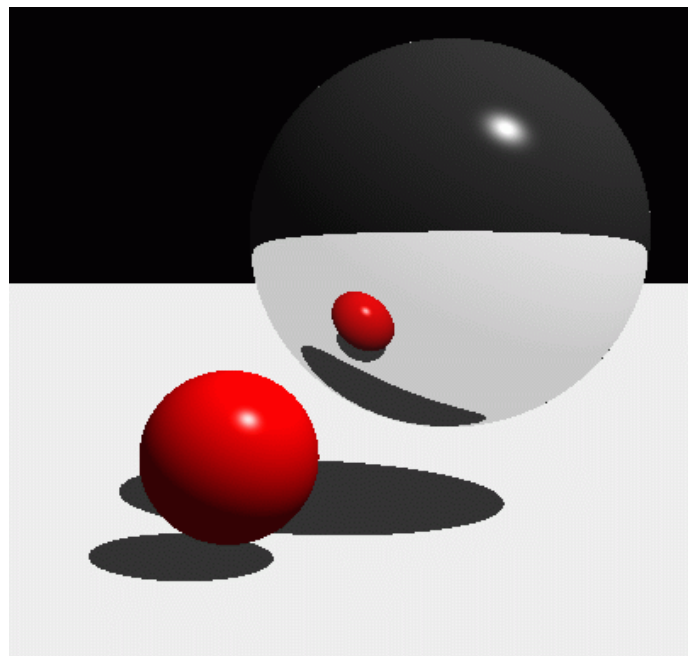
source: canon red book

© BBG Photographica. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Let's Pause for a Moment...

- Do these pictures look real?

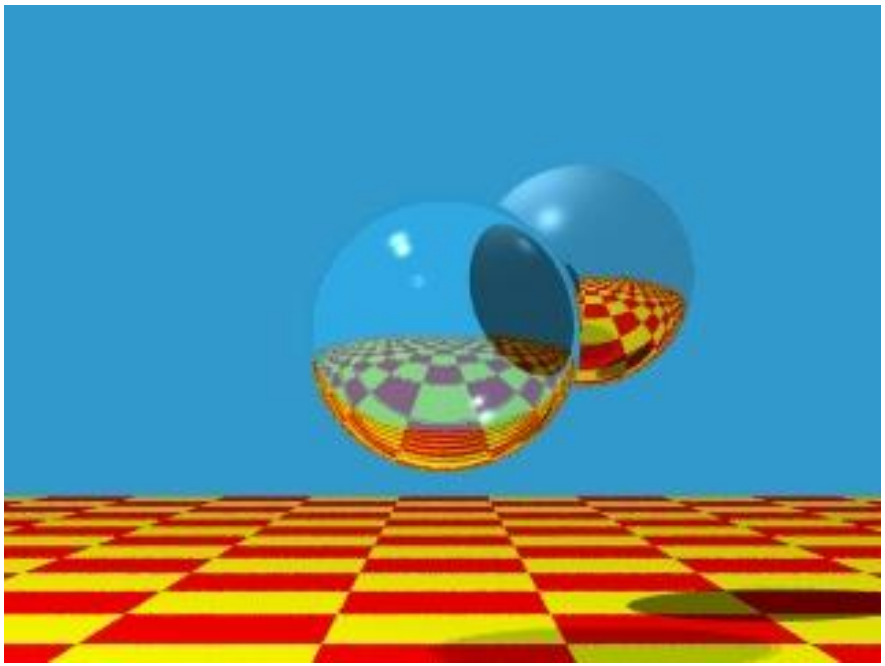


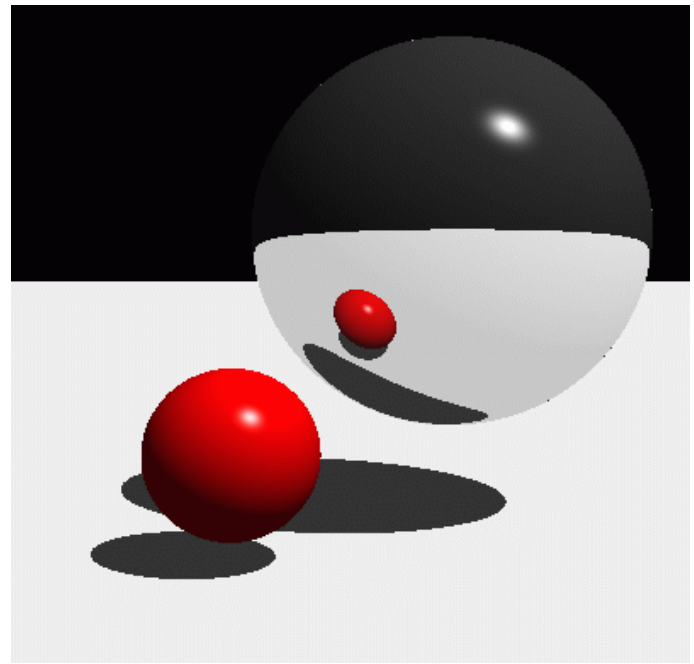


© Turner Whitted, Bell Laboratories. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

What's Wrong then?

- No surface is a perfect mirror,
  no material interface is perfectly smooth

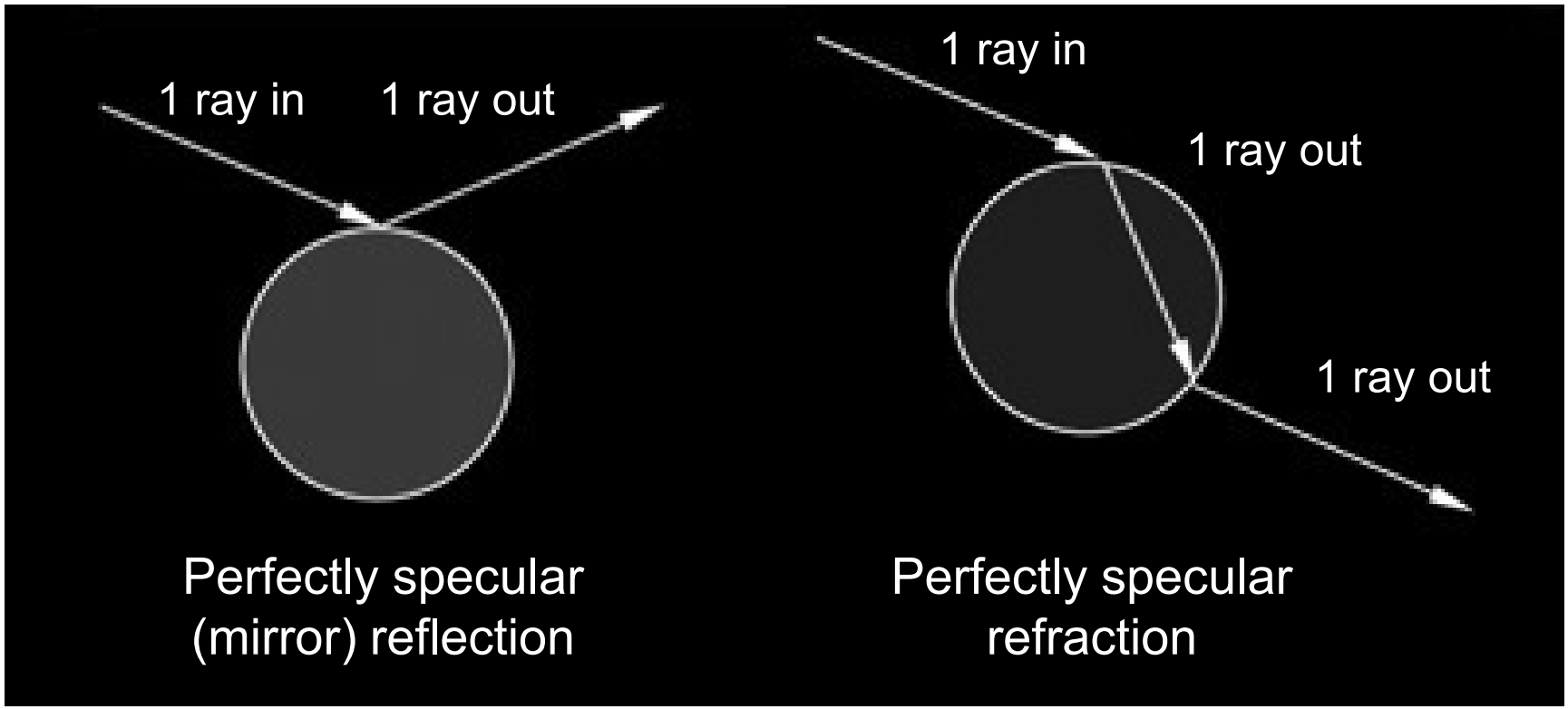




© Turner Whitted, Bell Laboratories. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

What's Wrong then?

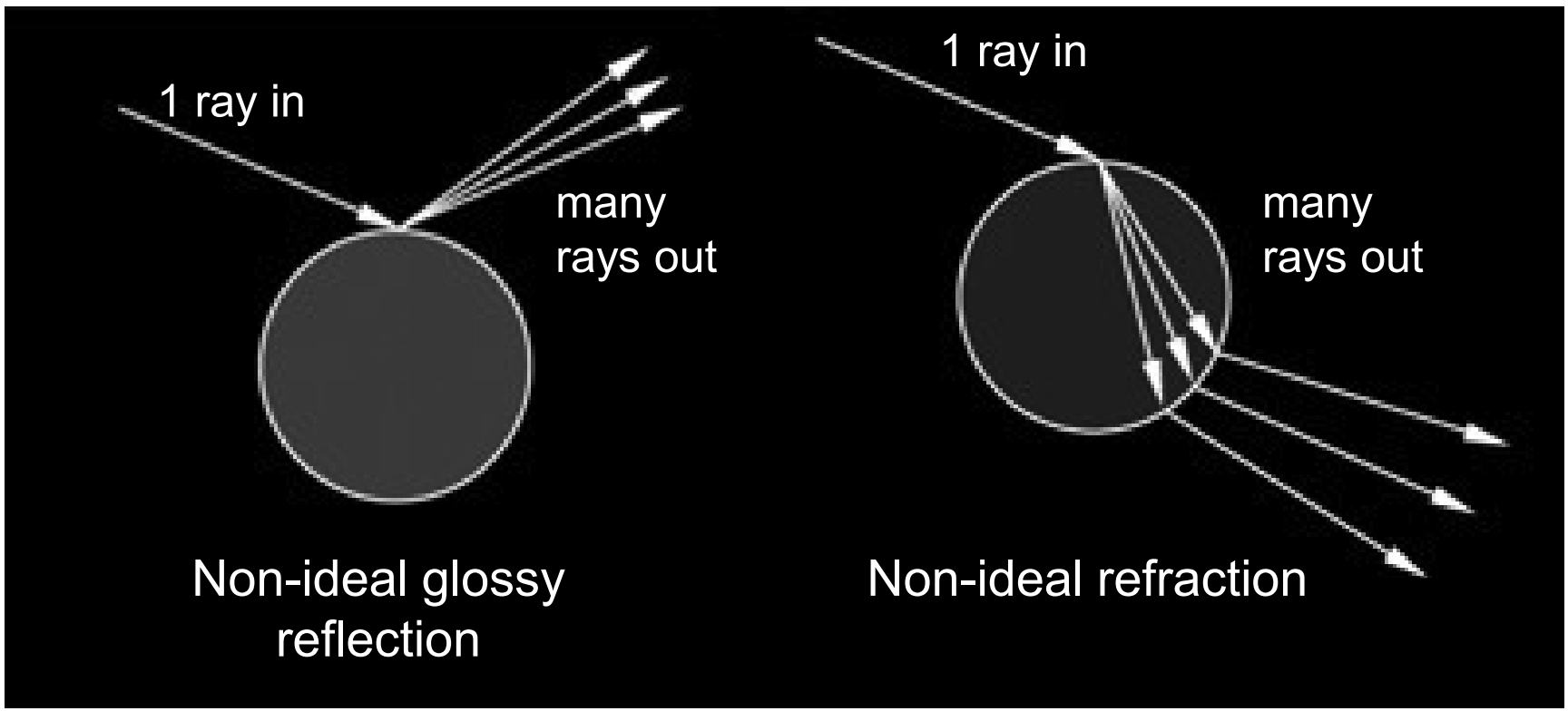
- No surface is a perfect mirror,
  no material interface is perfectly smooth



Adapted from blender.org

Non-Ideal Reflection/Refraction

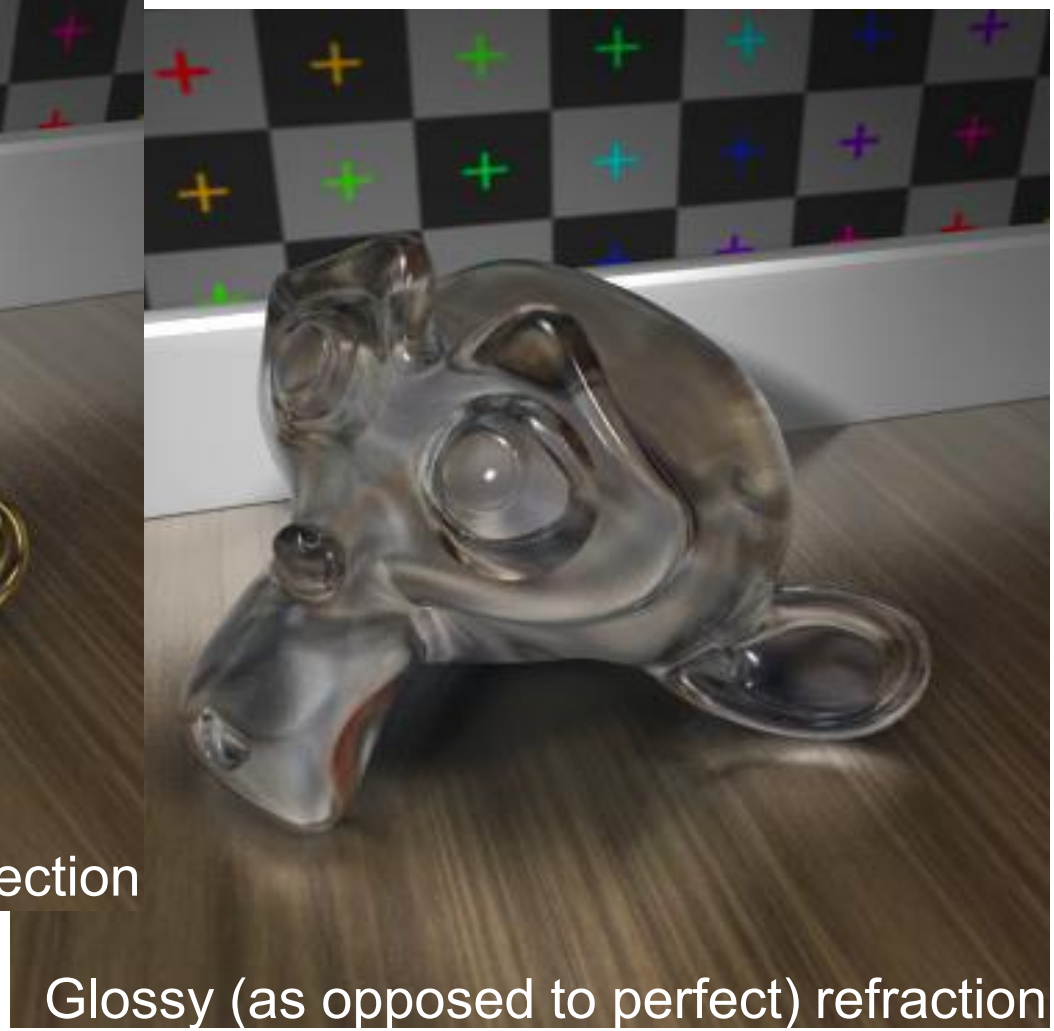
- No surface is a perfect mirror,
  no material interface is perfectly smooth



Adapted from blender.org

Non-Ideal Reflection/Refraction

Glossy (as opposed to mirror) reflection



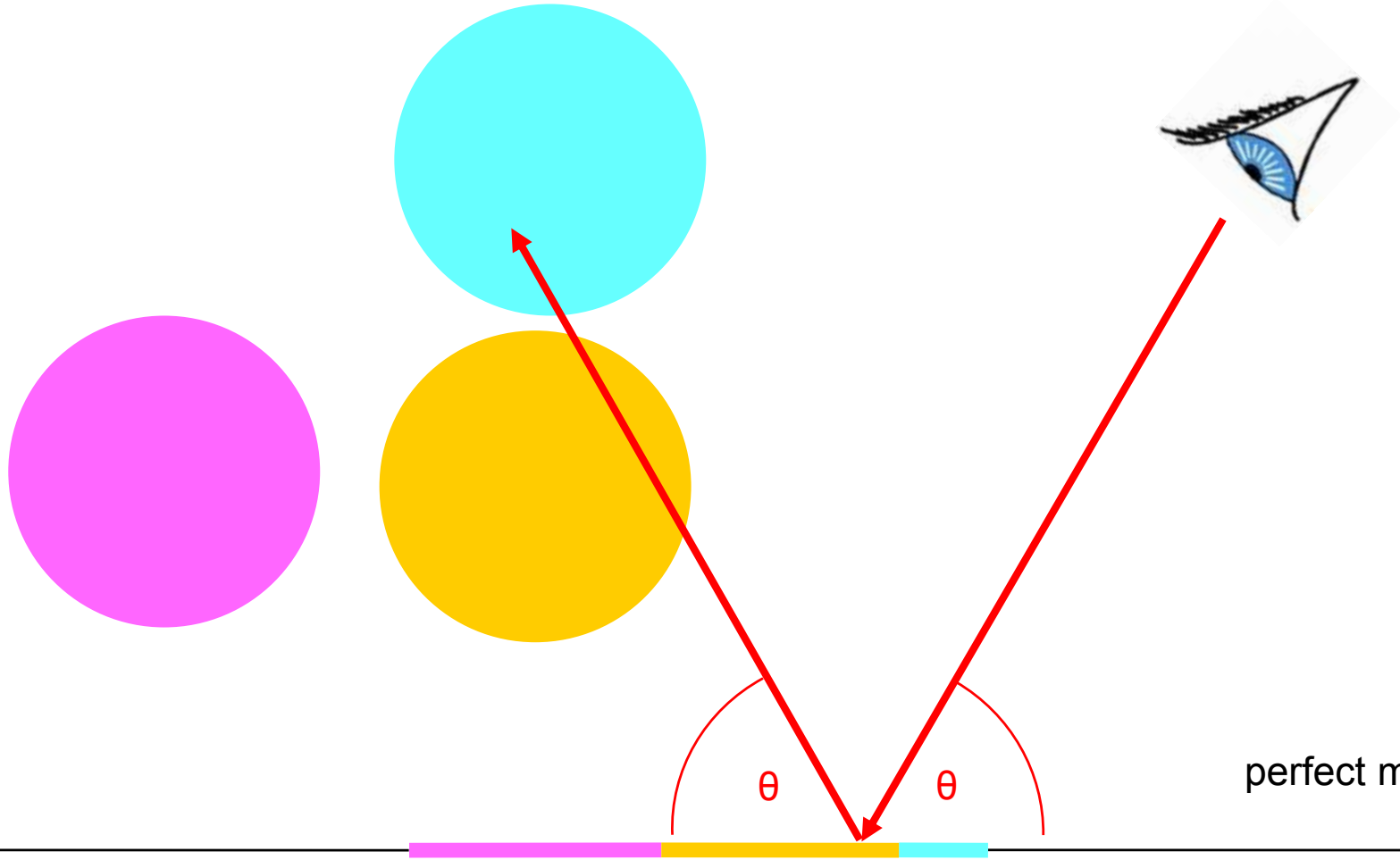
Glossy (as opposed to perfect) refraction

Courtesy of Blender Foundation. License CC-BY. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

images from blender.org

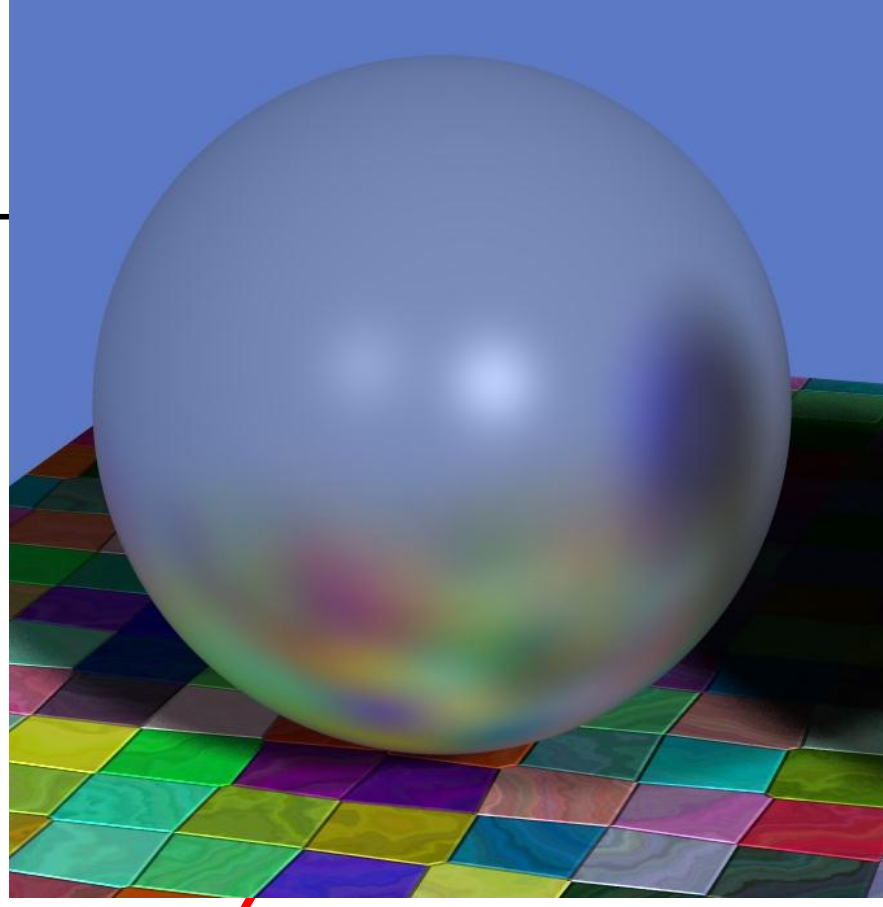
Reflection

- One reflection ray per intersection



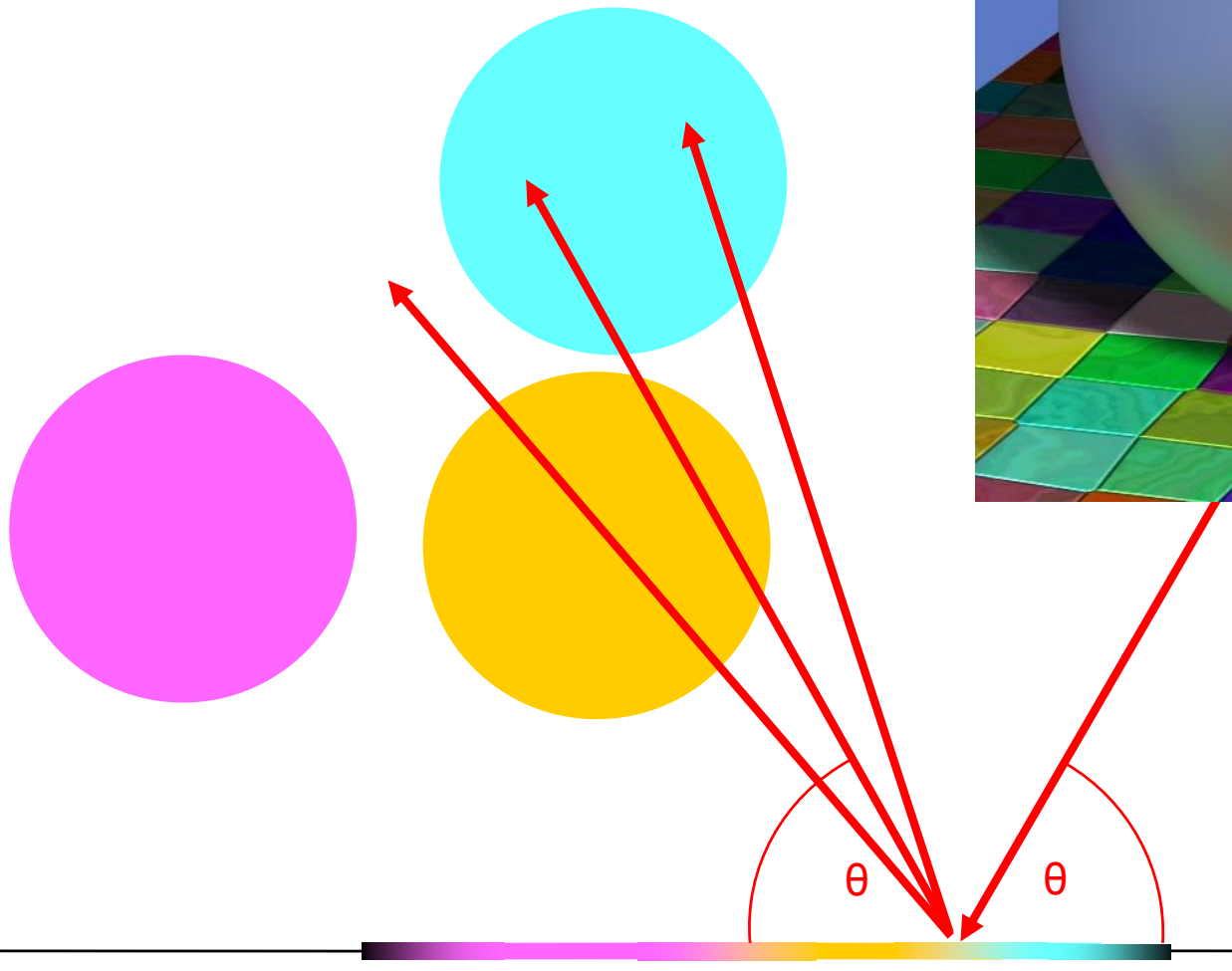
Glossy Reflection

- Multiple reflection rays



Courtesy of Justin Legakis.

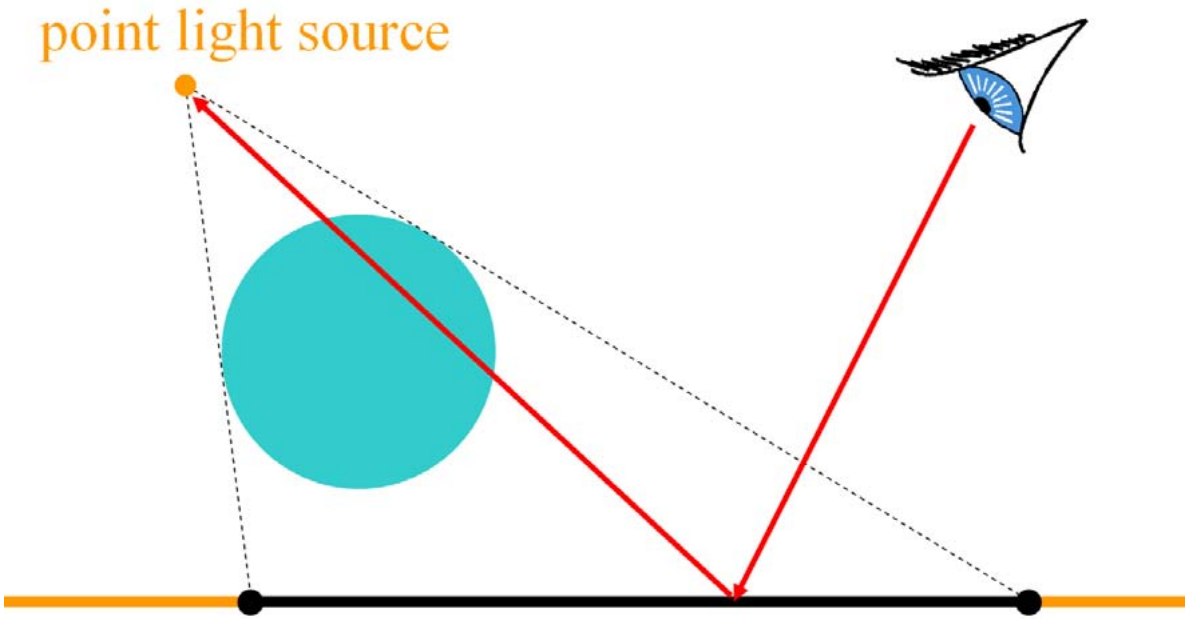
Justin Legakis

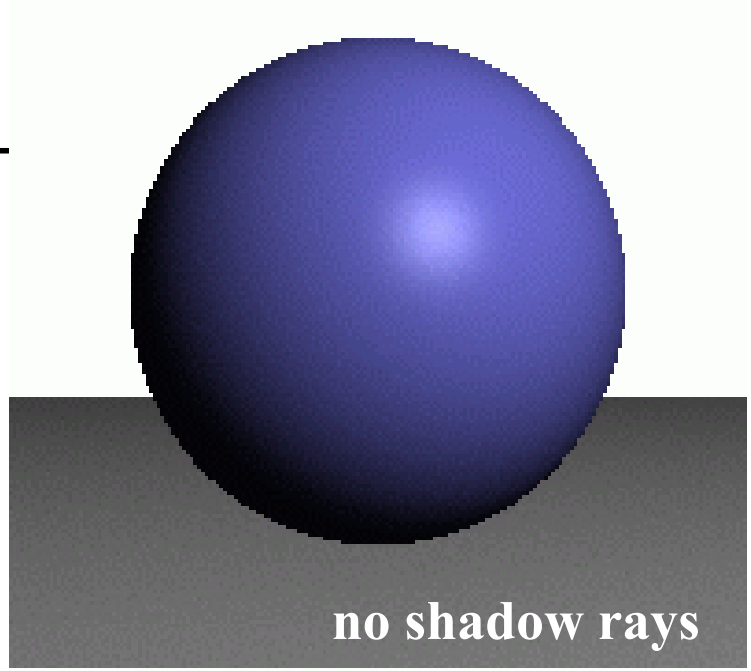


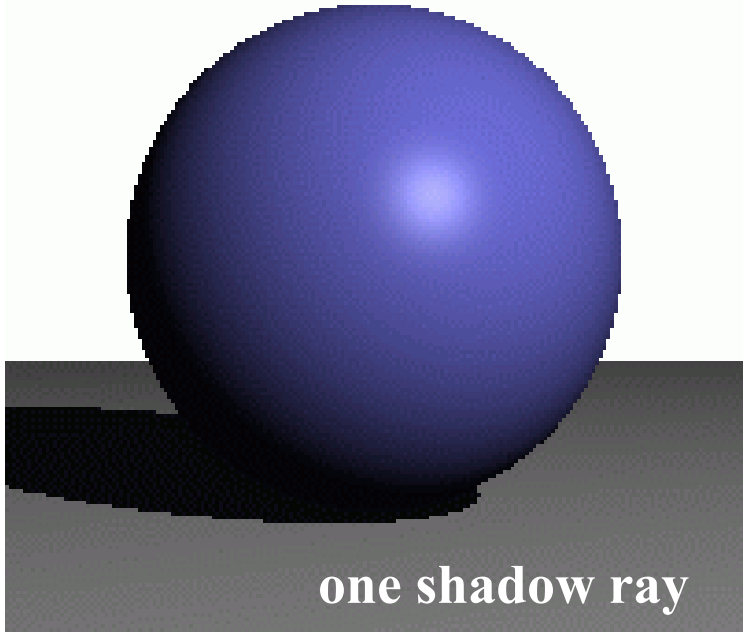
polished surface

Shadows

- One shadow ray per intersection per point light source







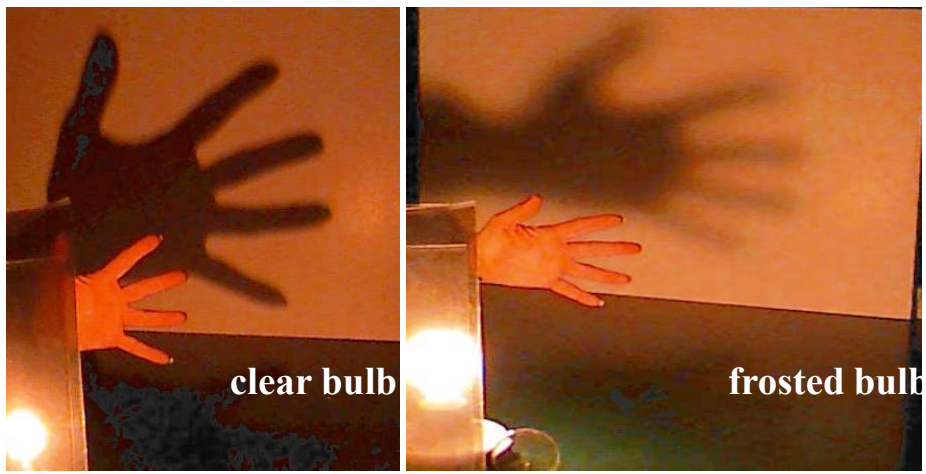
Shadows & Light Sources

Image removed due to copyright restrictions.

© David Fay Custom Furniture. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

http://www.davidfay.com/index.php

http://3media.initialized.org/photos/2000-10-18/index_gall.htm

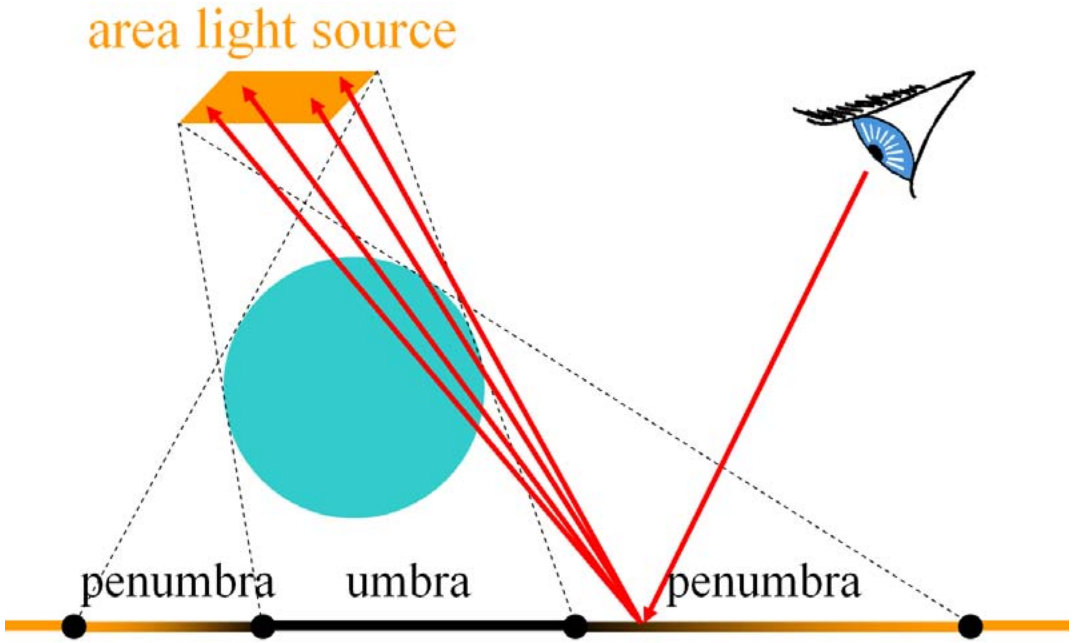


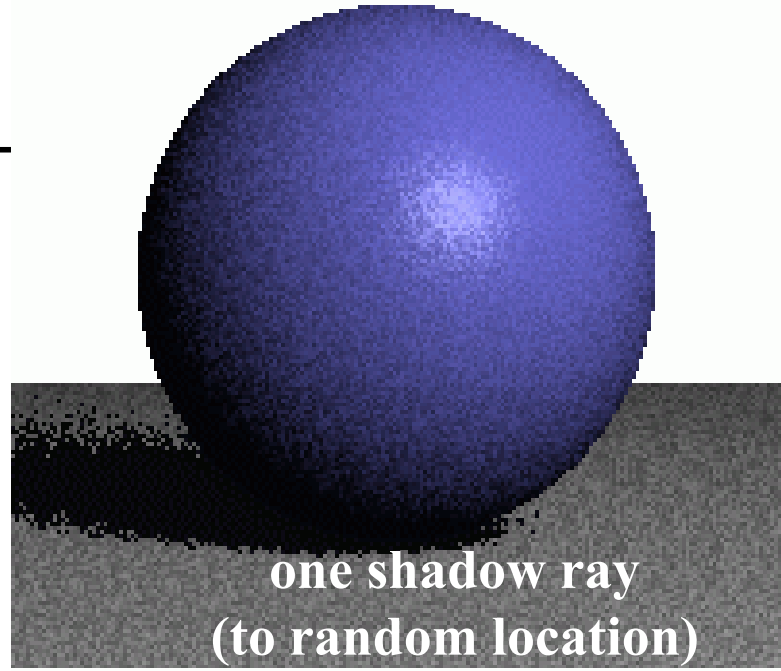
© Joseph Straley and Sally Shafer Kovash. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

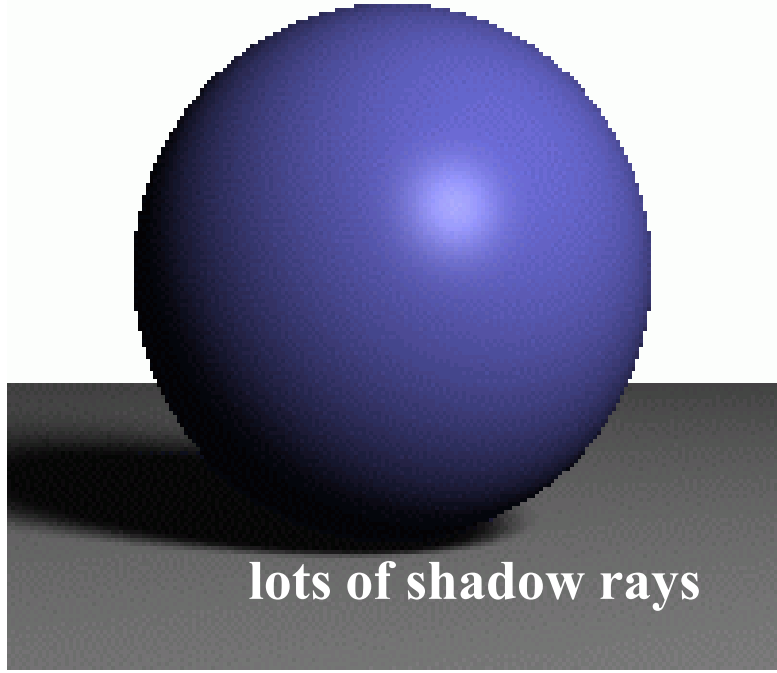
http://www.pa.uky.edu/~sciworks/light/preview/bulb2.htm

Soft Shadows

- Multiple shadow rays to sample area light source

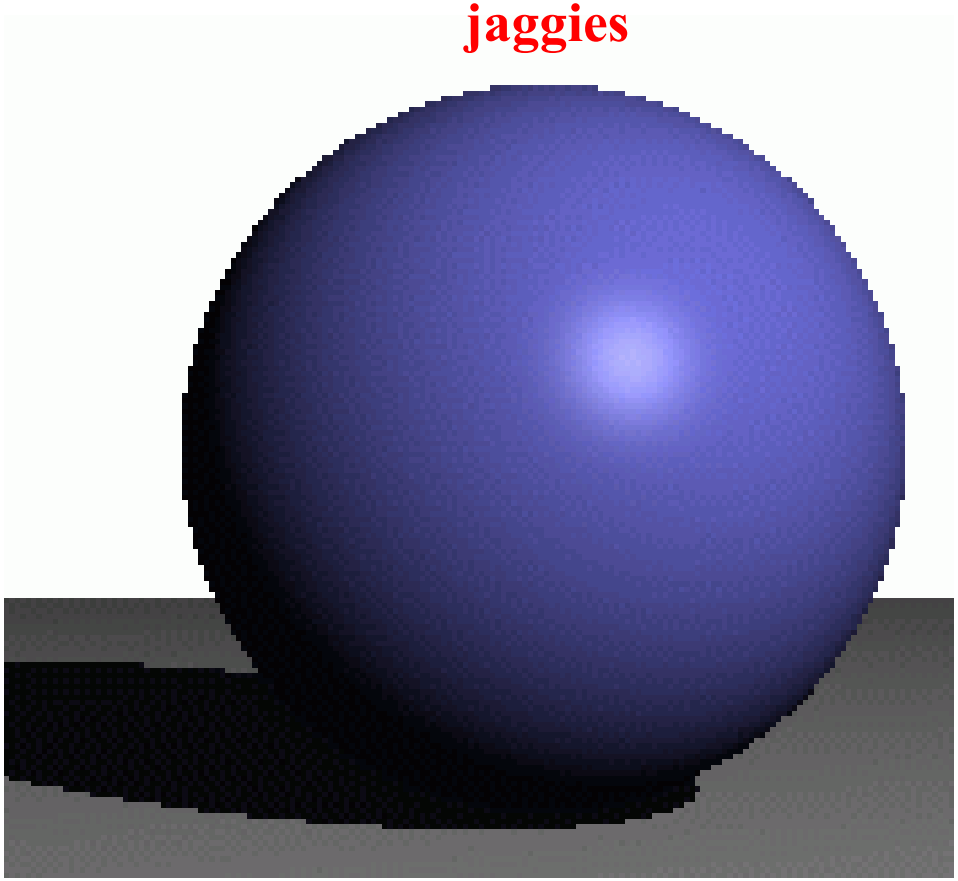


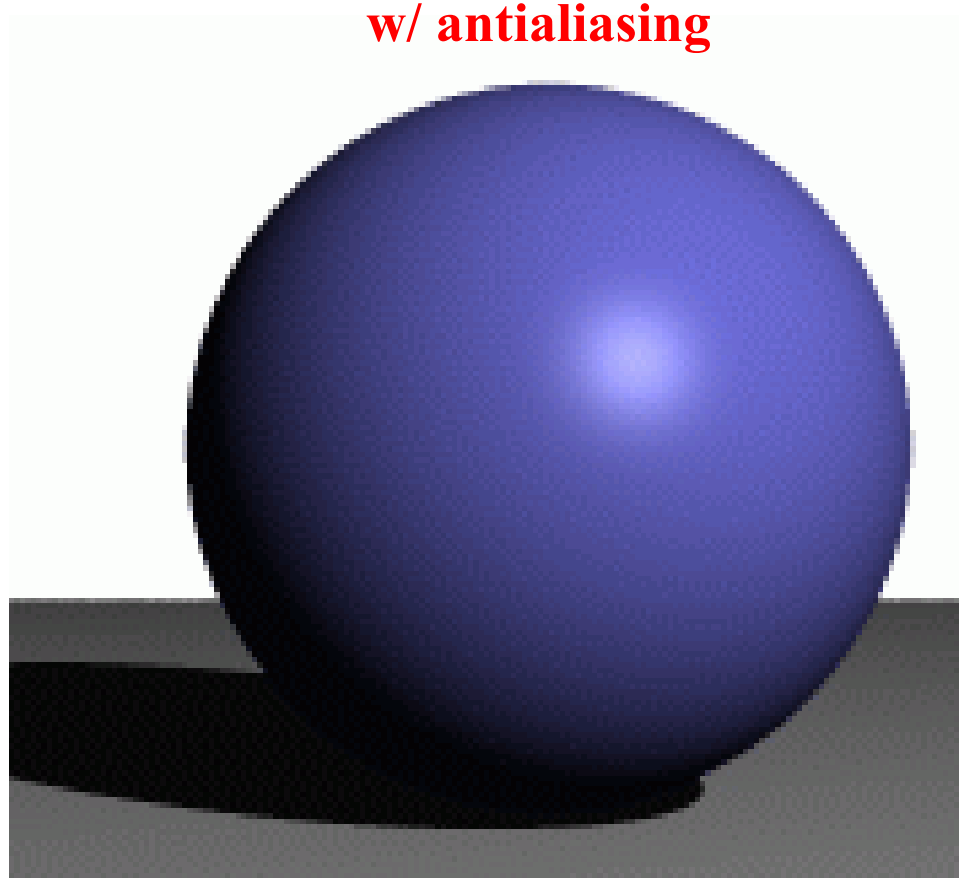




Antialiasing – Supersampling

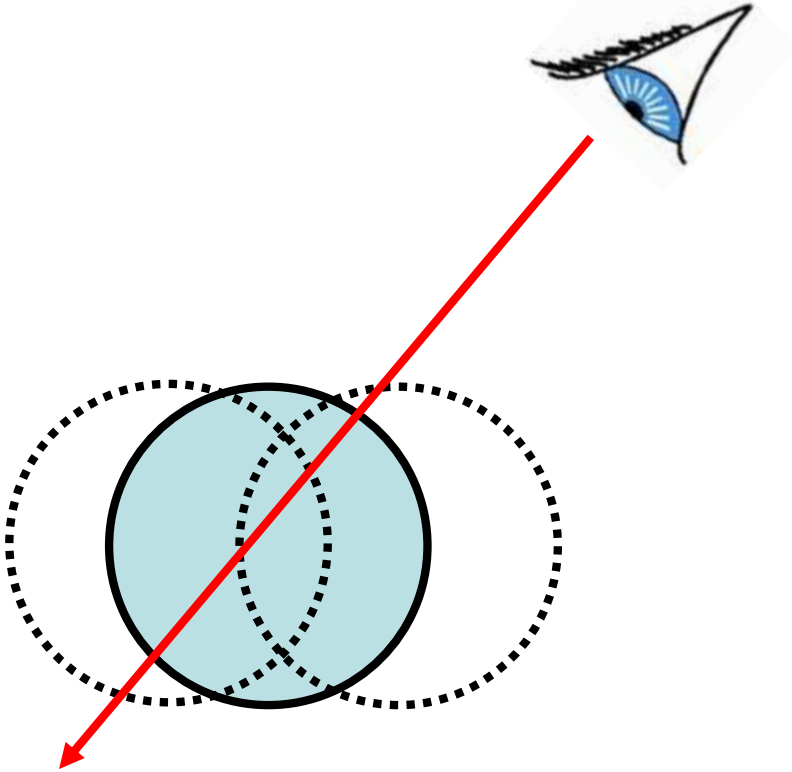
- Multiple rays per pixel





Motion Blur

- Sample objects temporally over time interval

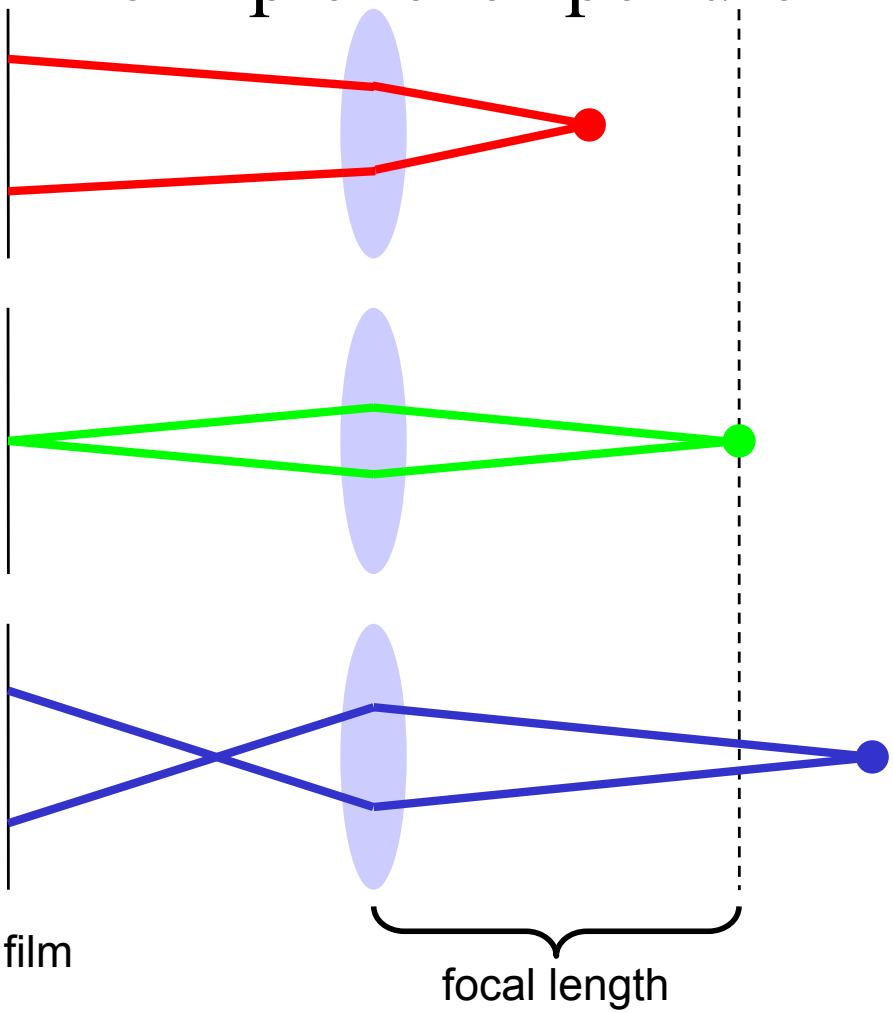


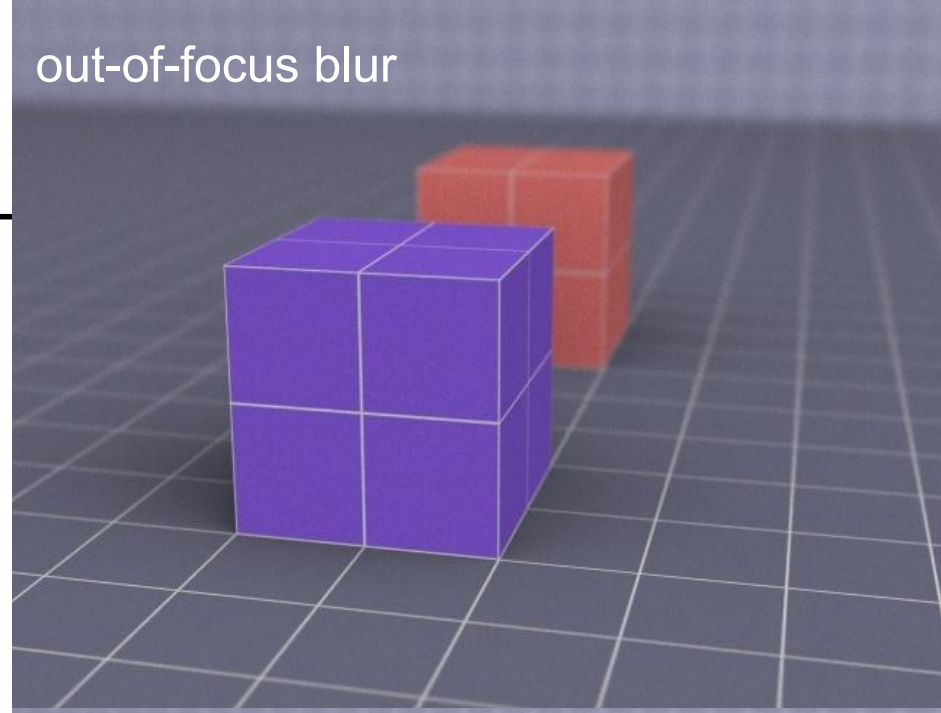
Rob Cook

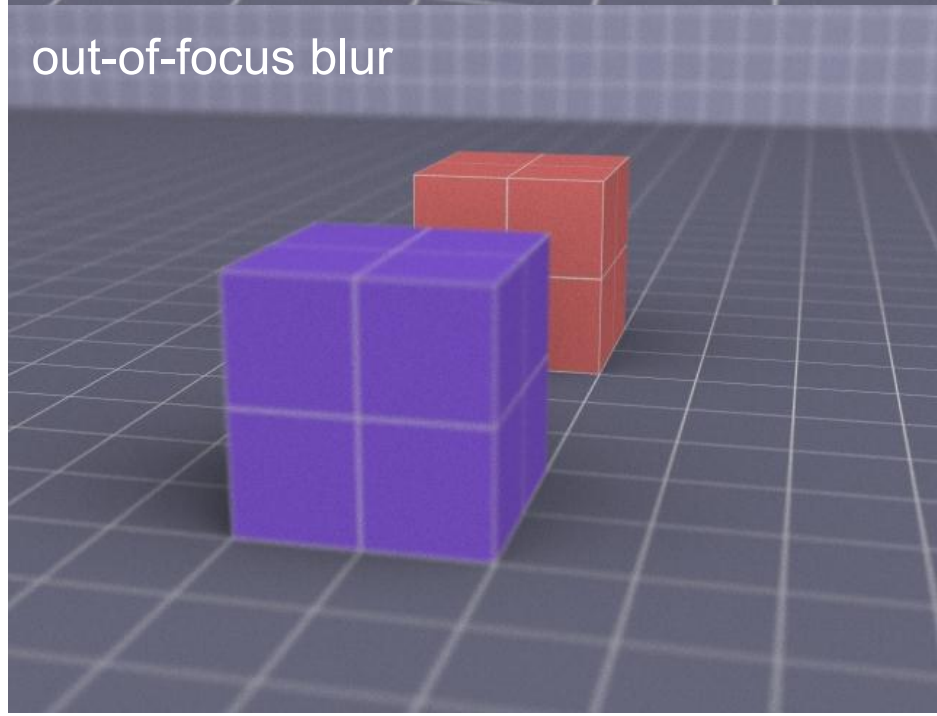
© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Depth of Field

- Multiple rays per pixel: sample lens aperture







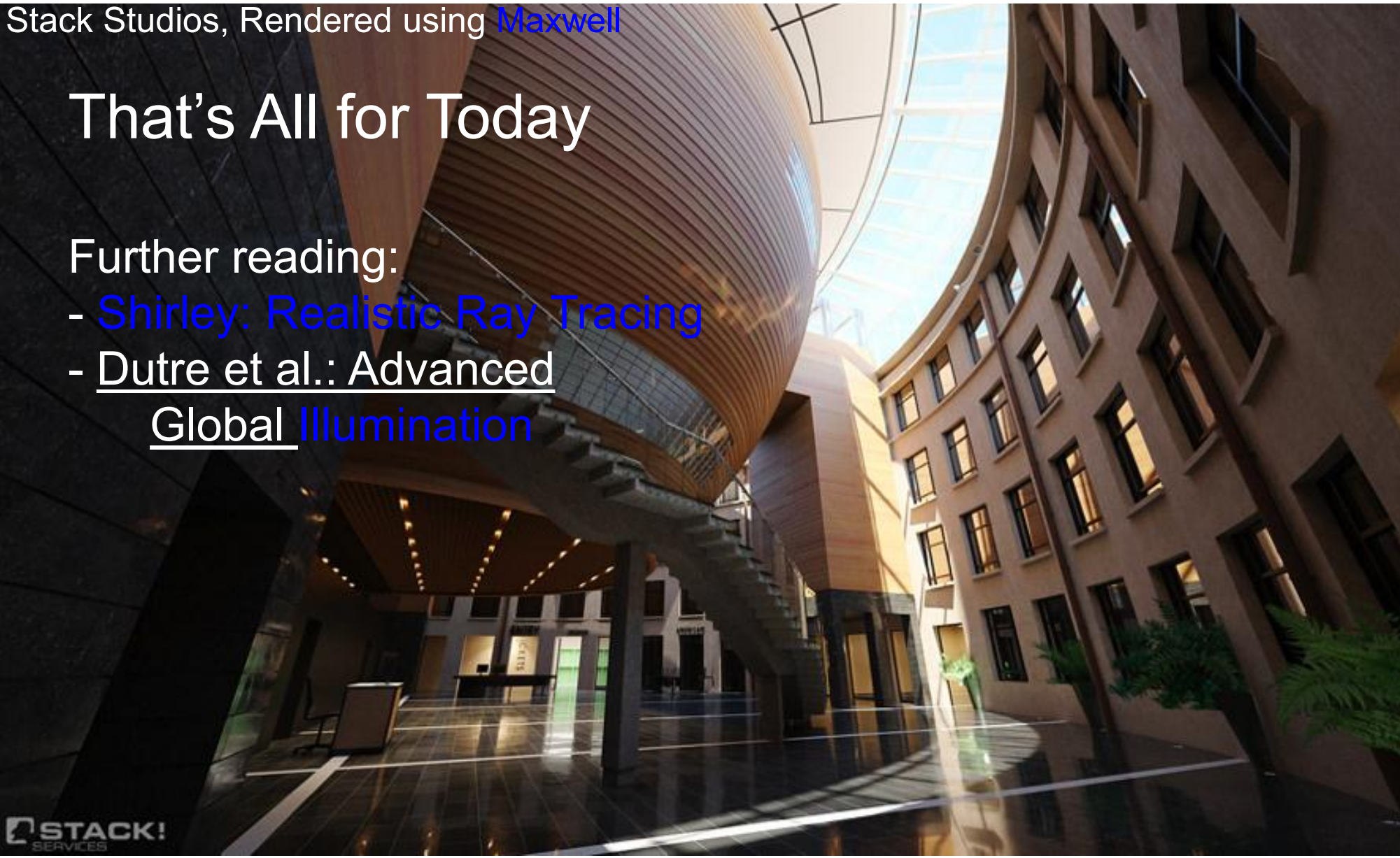
Justin Legakis

Courtesy of Justin Legakis.

Questions?

Henrik Wann Jensen

Courtesy of Henrik Wann Jensen. Used with permission.



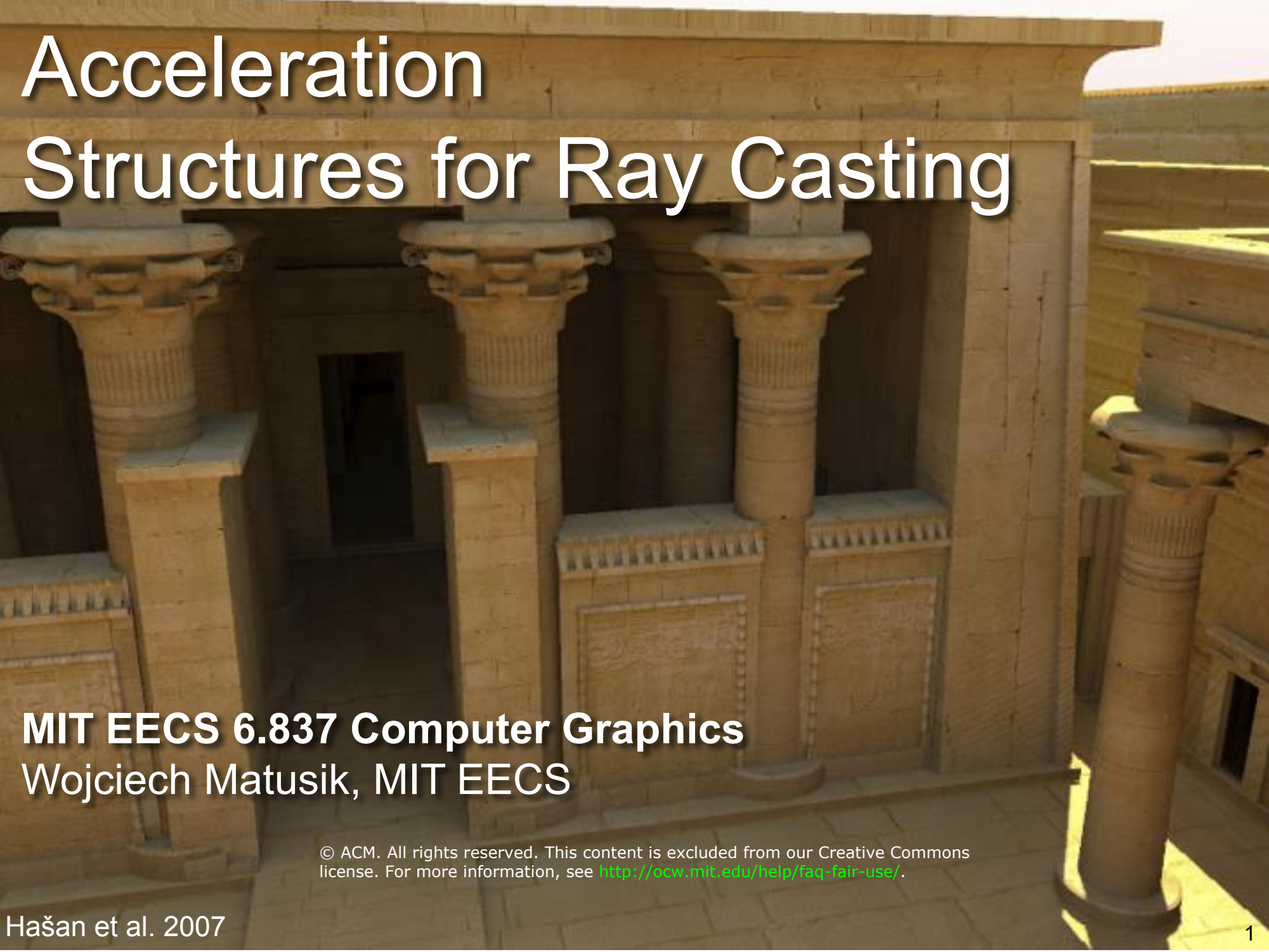
That's All for Today

Further reading:

- Shirley: Realistic Ray Tracing
- Dutre et al.: Advanced Global Illumination

© Next Limit S.L. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Acceleration Structures for Ray Casting



MIT EECS 6.837 Computer Graphics

Wojciech Matusik, MIT EECS

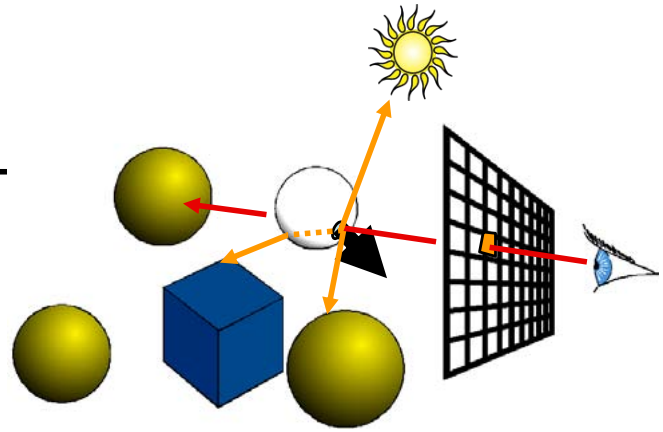
© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Hašan et al. 2007

Recap: Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
                 trace reflected ray
    If transparent
        color += color_trans *
                 trace transmitted ray
```

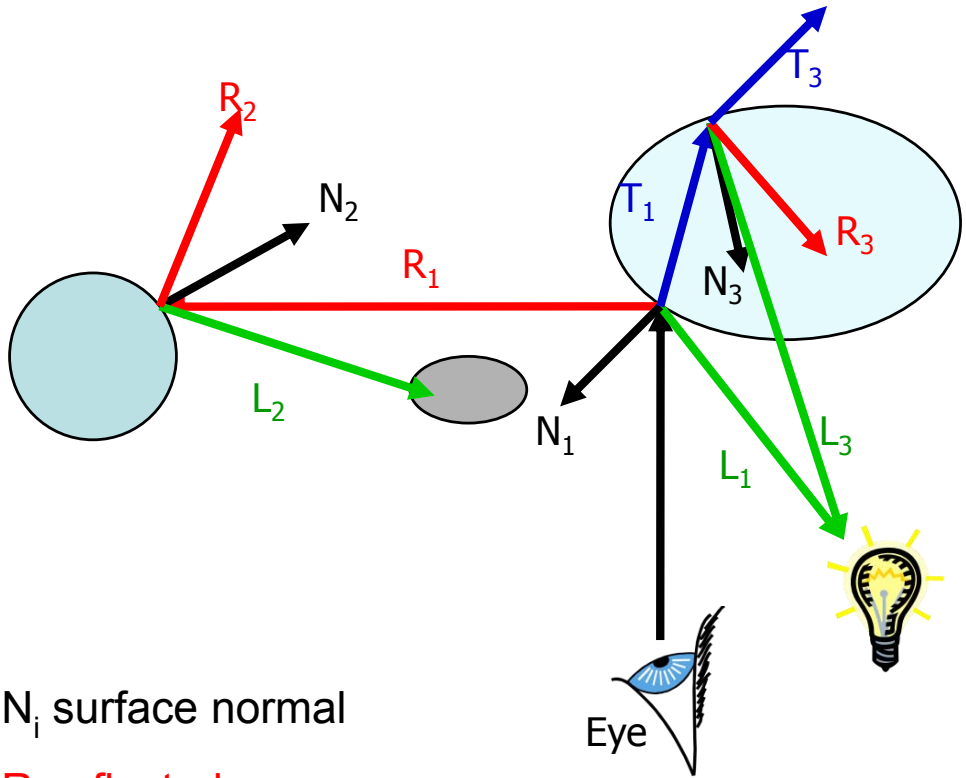
- ***Does it ever end?***



Stopping criteria:

- Recursion depth
  - Stop after a number of bounces
- Ray contribution
  - Stop if reflected / transmitted contribution becomes too small

The Ray Tree

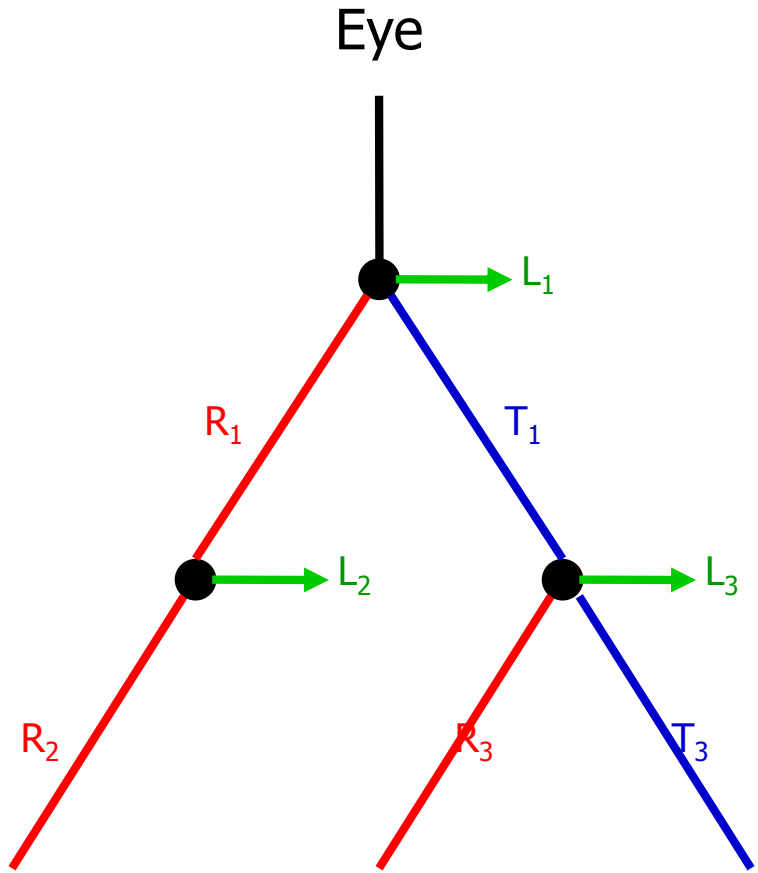


N_i surface normal

R_i reflected ray

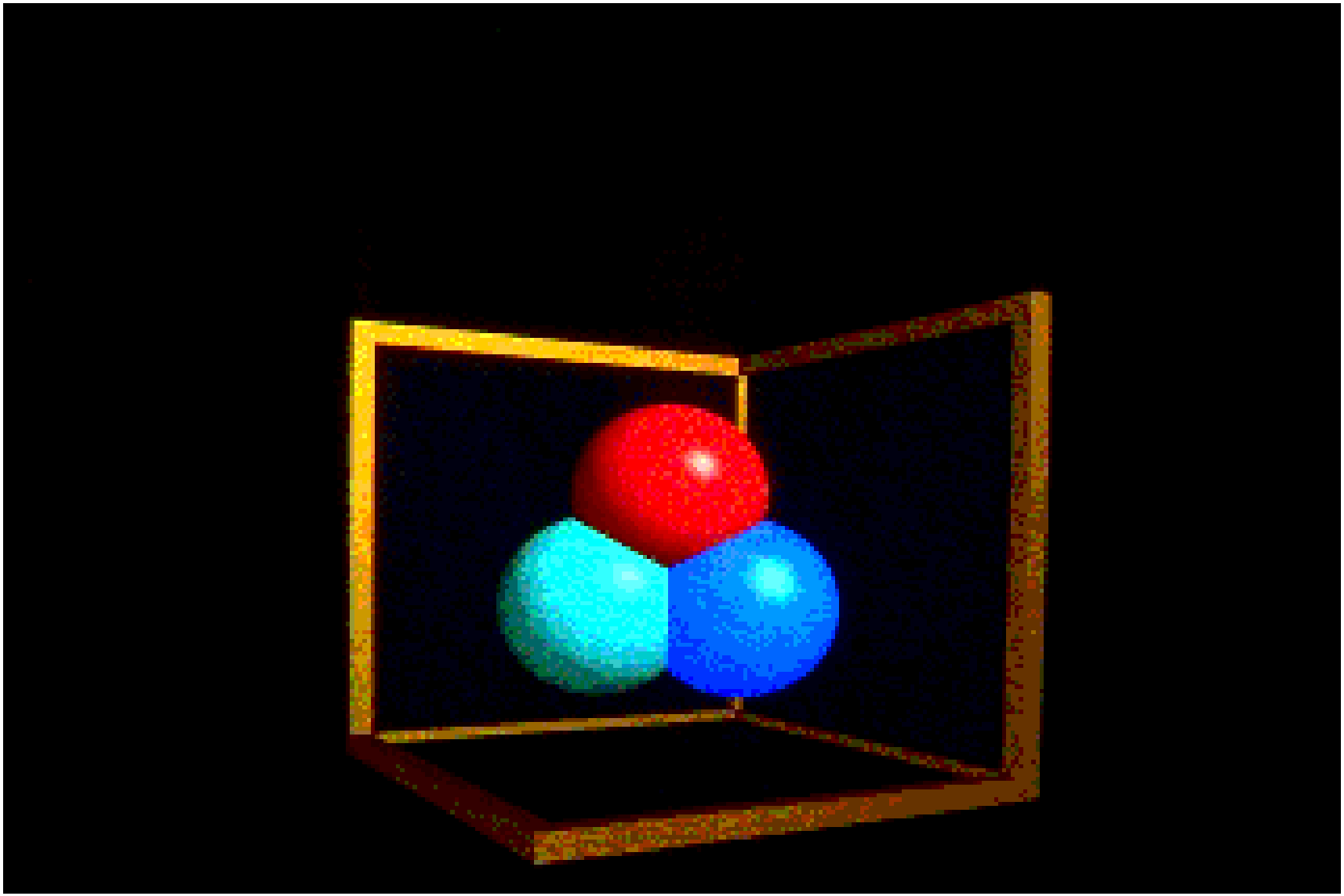
L_i shadow ray

T_i transmitted (refracted) ray

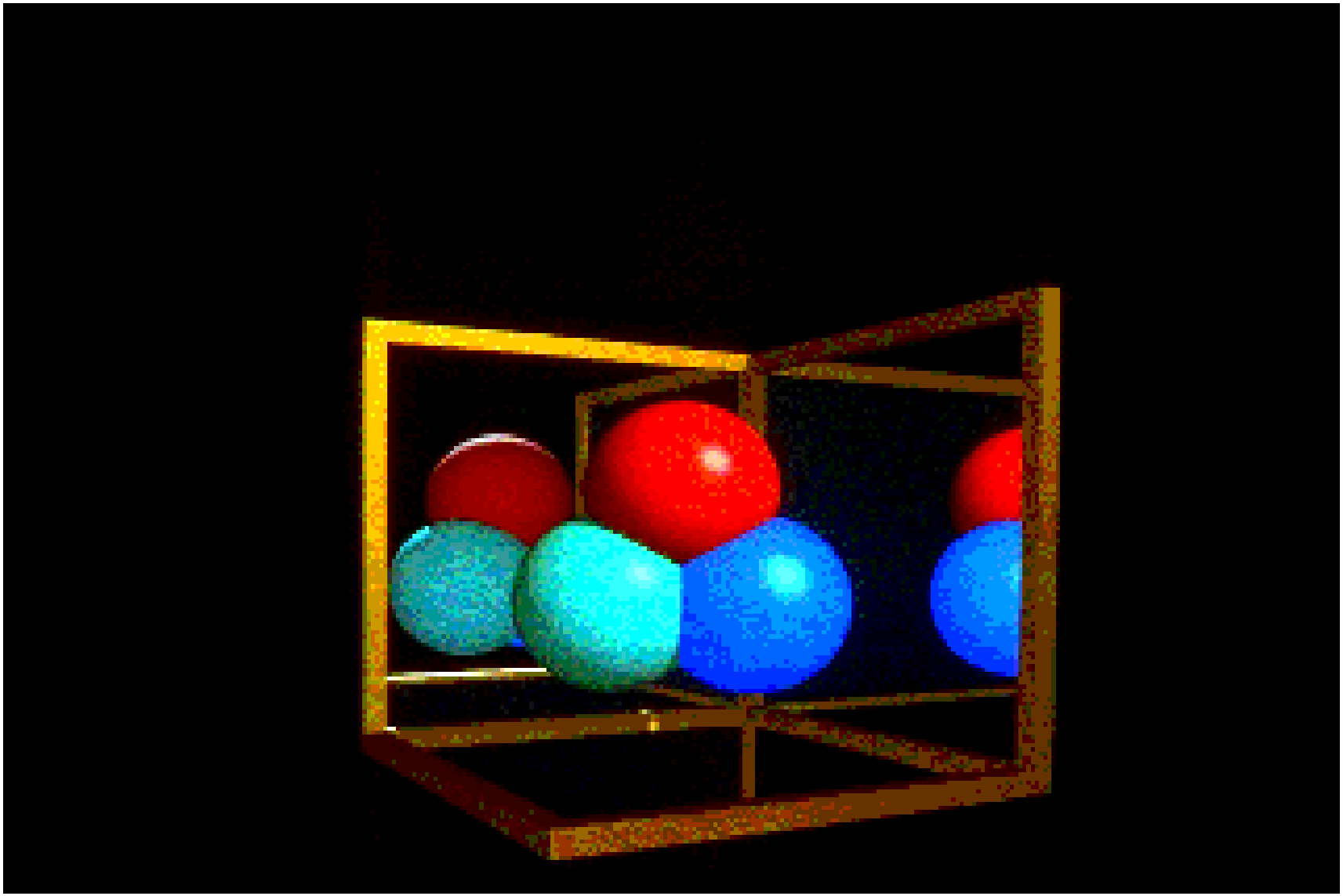


Complexity?

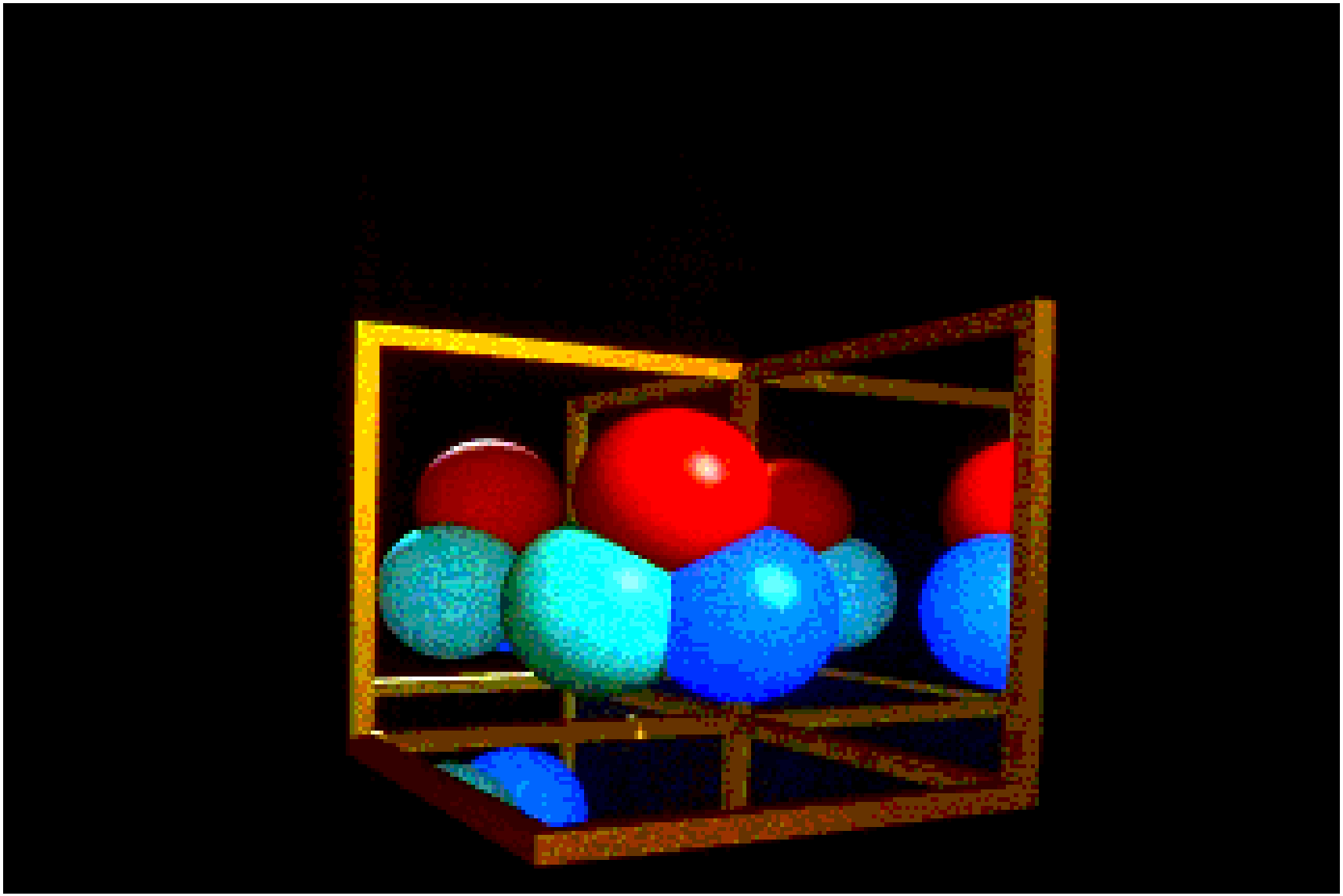
Recursion For Reflection: None



Recursion For Reflection: 1

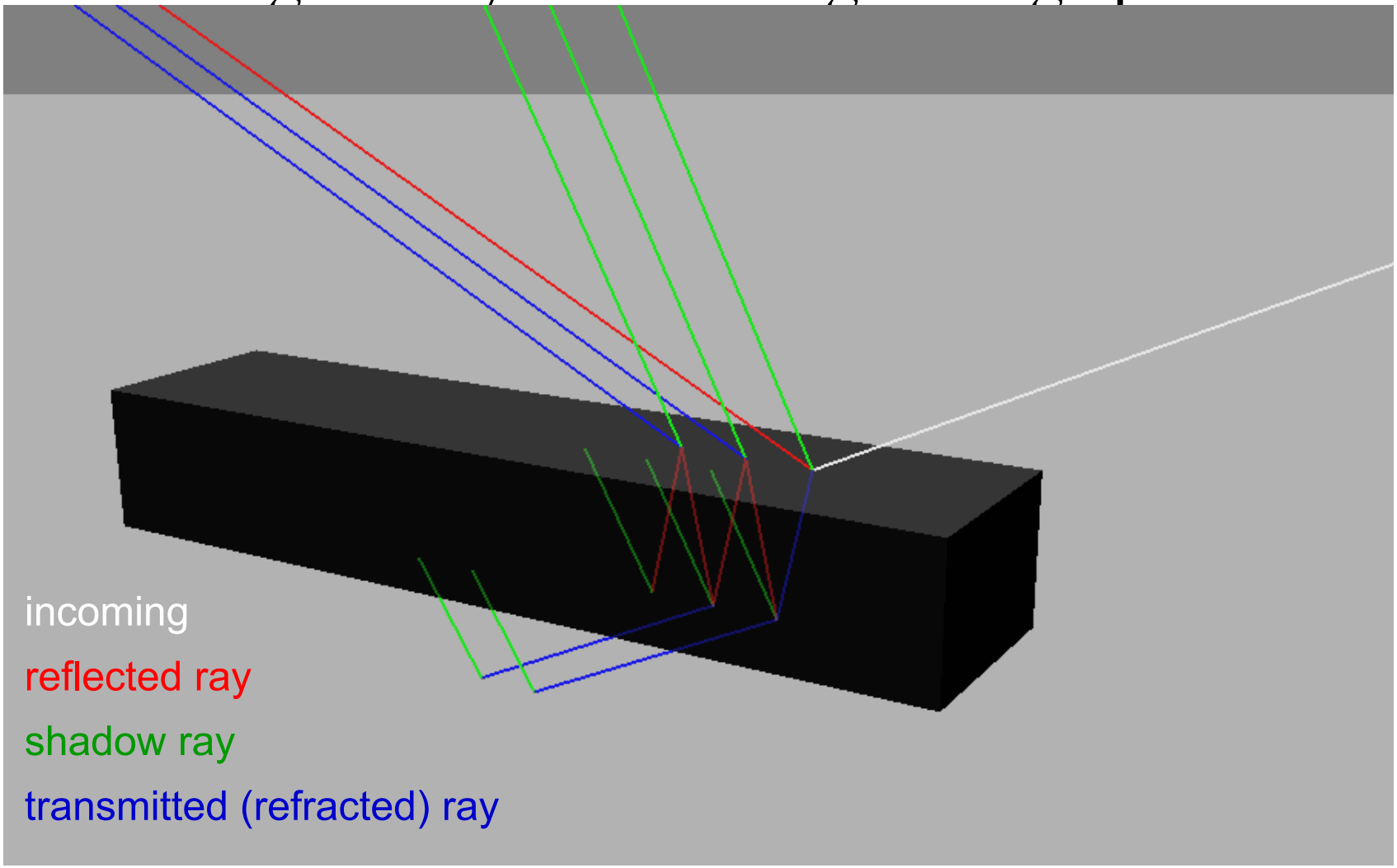


Recursion For Reflection: 2



Ray tree

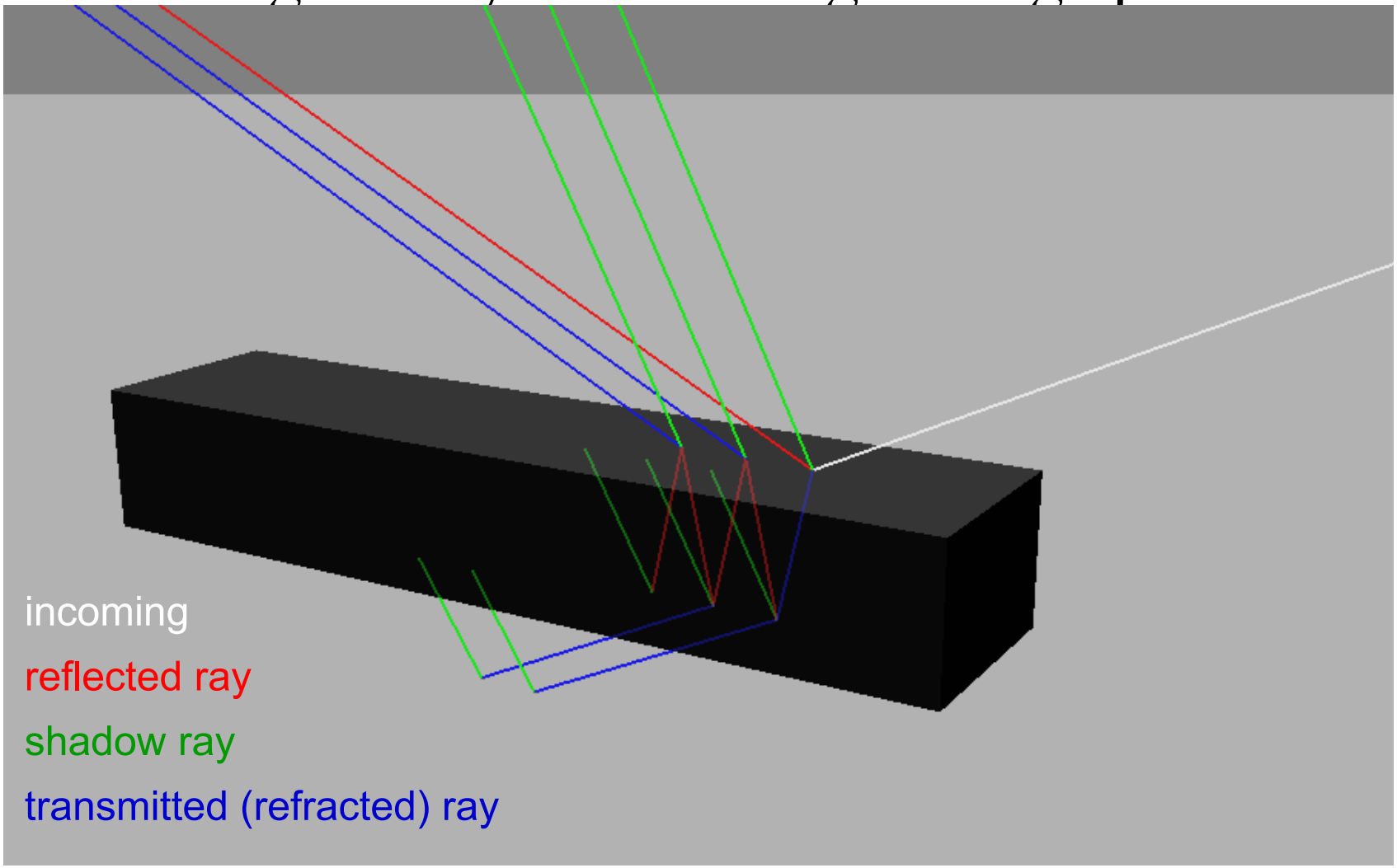
- Visualizing the ray tree for single image pixel



Ray tree

This gets pretty complicated pretty fast!

- Visualizing the ray tree for single image pixel



Ray Tracing Algorithm Analysis

- Lots of primitives
- Recursive
- Distributed Ray Tracing
  - Means using many rays for non-ideal/non-pointlike phenomena
    - Soft shadows
    - Anti-aliasing
    - Glossy reflection
    - Motion blur
    - Depth of field

cost ≈ height * width *
num primitives *
intersection cost *
size of recursive ray tree *
num shadow rays *
num supersamples *
num glossy rays *
num temporal samples *
num aperture samples *
. . .

Can we reduce this?

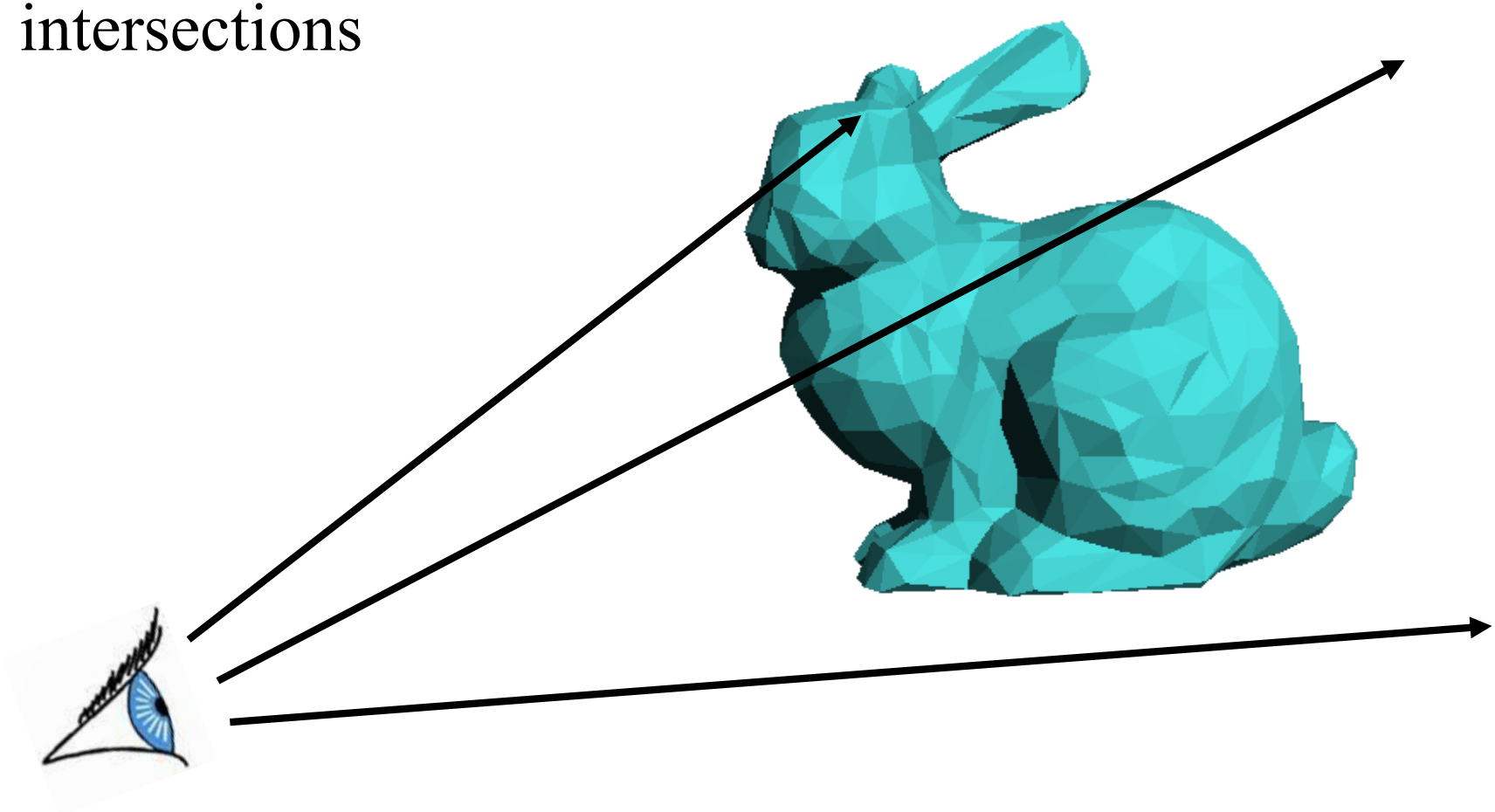
Today

- Motivation
  - You need LOTS of rays to generate nice pictures
  - Intersecting every ray with every primitive becomes the bottleneck
- Bounding volumes
- Bounding Volume Hierarchies, Kd-trees

```
For every pixel
   Construct a ray from the eye
   For every object in the scene
       Find intersection with the ray
       Keep if closest
        Shade
```

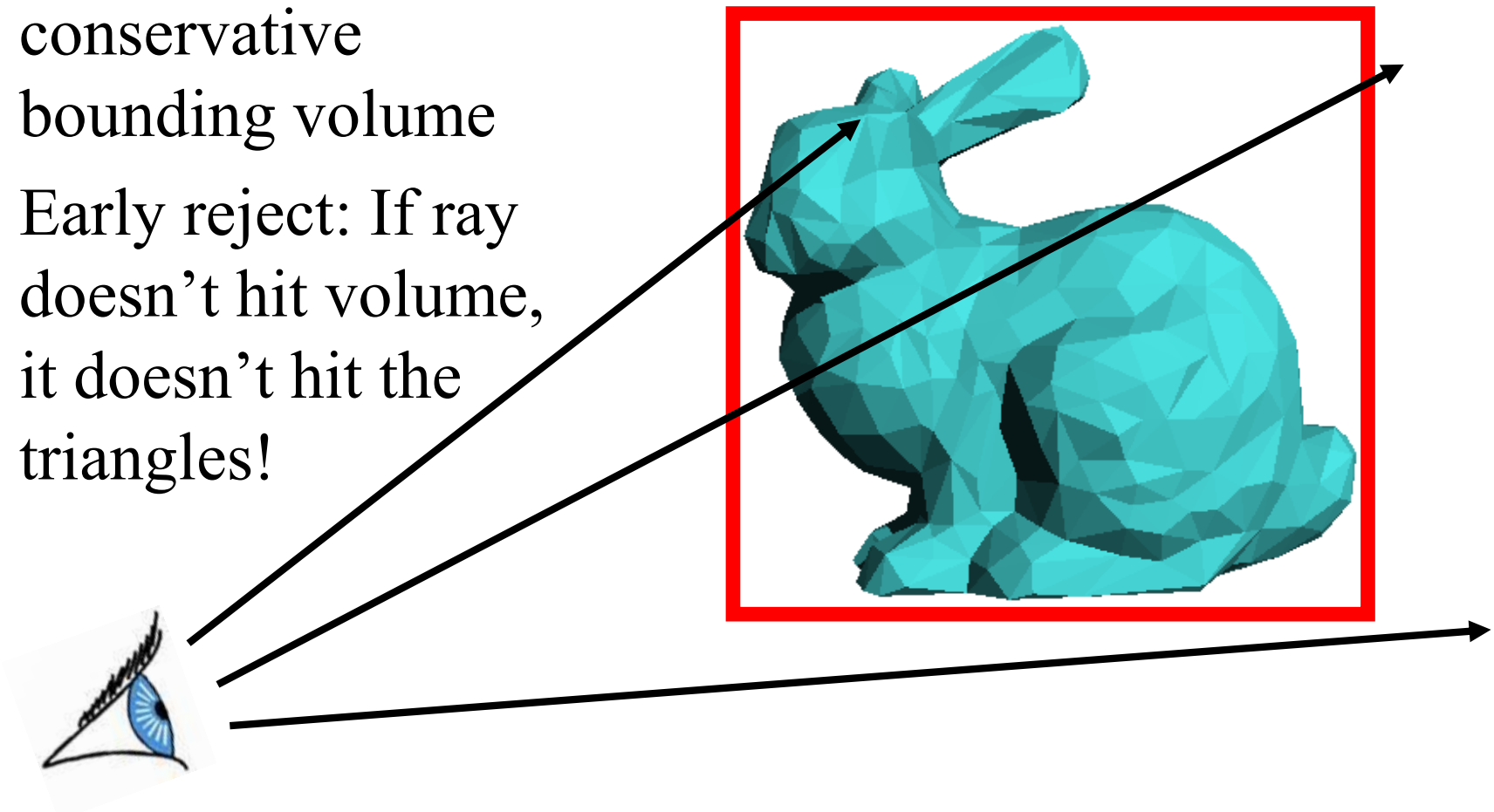
Accelerating Ray Casting

- Goal: Reduce the number of ray/primitive intersections



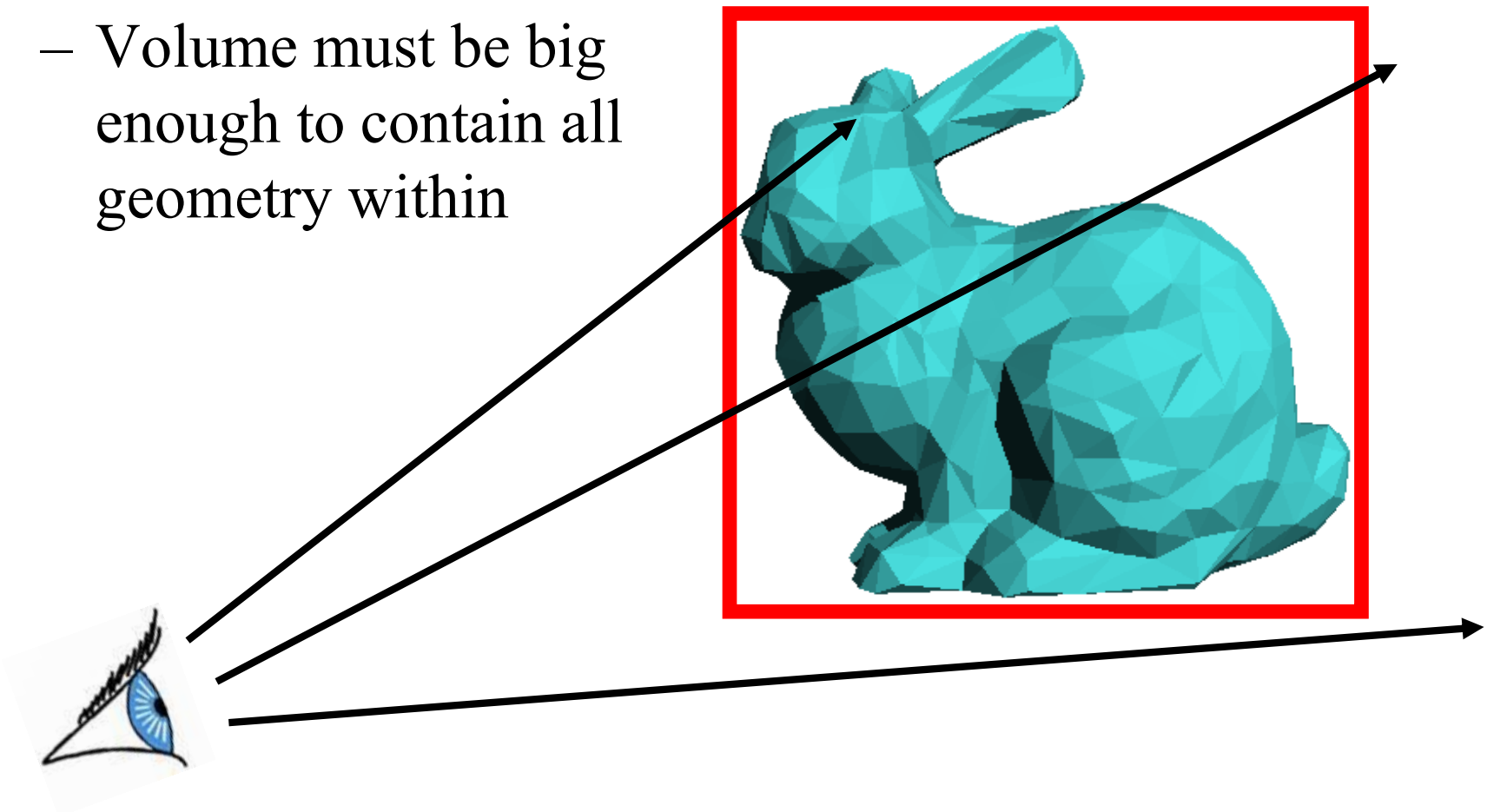
Conservative Bounding Volume

- First check for an intersection with a conservative bounding volume
- Early reject: If ray doesn't hit volume, it doesn't hit the triangles!



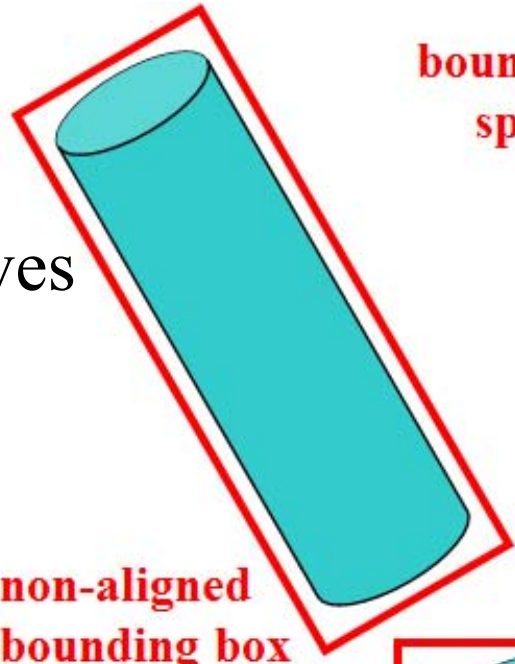
Conservative Bounding Volume

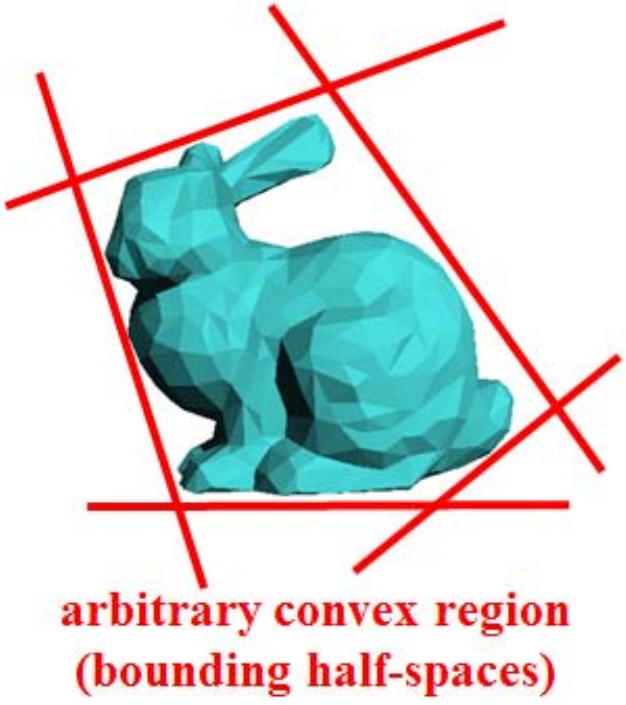
- What does "conservative" mean?
  - Volume must be big enough to contain all geometry within

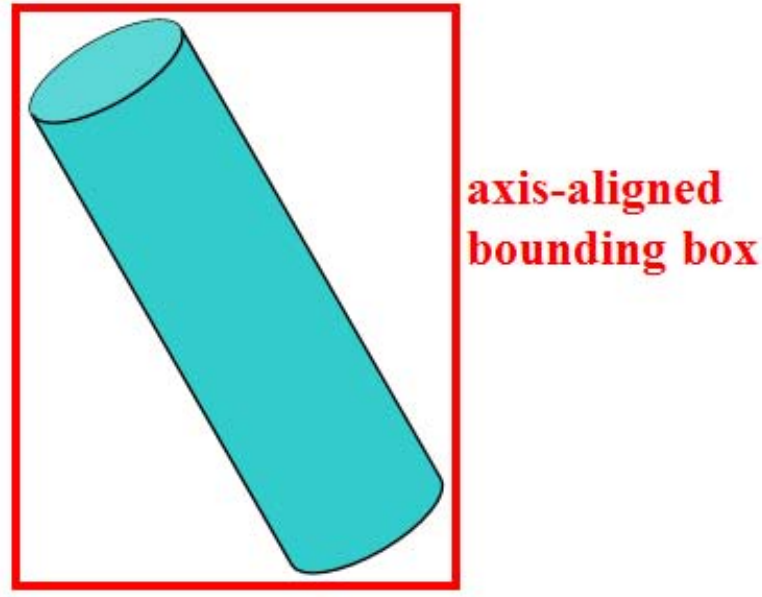


Conservative Bounding Regions

- Desiderata
  - Tight → avoid false positives
  - Fast to intersect

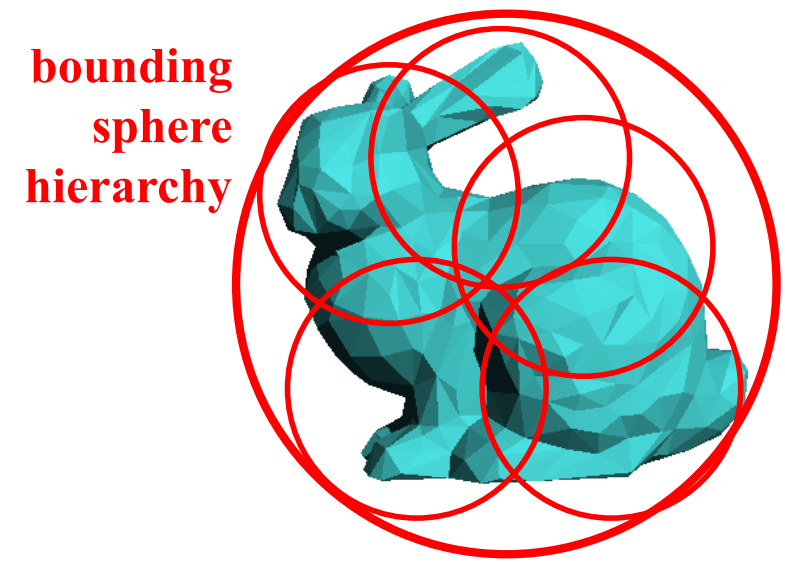






Bounding Volume Hierarchies

- If ray hits bounding volume, must we test all primitives inside it?
  - Lots of work, think of a 1M-triangle mesh
- You guessed it already, we'll split the primitives in groups and build recursive bounding volumes
  - Like collision detection, remember?



TIEA311 - Today in Jyväskylä

Sorry, guys. . . If you want to learn more about this stuff, you need to do it all by yourselves. At this point, we skip a lot of material about optimizing algorithms (all sorts of similar algorithms, even if graphics is an example, btw.).

To learn more, you may want to check out (on your own time) “Lecture 14” of the original course, and some books and articles about ray tracing. Should you want to use this stuff in your own hobby projects, do ask about possibilities of getting credit points. State-of-the-art methods are good topics for Bachelor / Master thesis projects.

This course will now teleport over algorithmic optimizations. (This is, of course, called “course optimization”!) – we haven’t covered all the fundamentals yet, so let’s not get stuck with details. The next few slides from the MIT course give references for further study (not necessary for our course).

Questions?

- Further reading on efficient Kd-tree construction
  - Hunt, Mark & Stoll, IRT 2006
  - Zhou et al., SIGGRAPH Asia 2008

Zhou et al.

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Optimizing Splitting Planes

- Most people use the Surface Area Heuristic (SAH)
  - MacDonald and Booth 1990, "Heuristic for ray tracing using space subdivision", Visual Computer
- Idea: simple probabilistic prediction of traversal cost based on split distance
- Then try different possible splits and keep the one with lowest cost
- Further reading on efficient Kd-tree construction
  - Hunt, Mark & Stoll, IRT 2006
  - Zhou et al., SIGGRAPH Asia 2008

Hard-core efficiency considerations

- See e.g. Ingo Wald's PhD thesis
  - http://www.sci.utah.edu/~wald/PhD/
- Calculation
  - Optimized barycentric ray-triangle intersection
- Memory
  - Make kd-tree node as small as possible (dirty bit packing, make it 8 bytes)
- Parallelism
  - SIMD extensions, trace 4 rays at a time, mask results where they disagree

Pros and Cons of Kd trees

- Pros
  - Simple code
  - Efficient traversal
  - Can conform to data

- Cons
  - costly construction, not great if you work with moving objects

Questions?

- For extensions to moving scenes, see Real-Time KD-Tree Construction on Graphics Hardware, Zhou et al., SIGGRAPH 2008

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

TIEA311 - Today in Jyväskylä

Sorry, guys. . . If you want to learn more about this stuff, you need to do it all by yourselves. At this point, we skip a lot of material about optimizing algorithms (all sorts of similar algorithms, even if graphics is an example, btw.).

To learn more, you may want to check out (on your own time) “Lecture 14” of the original course, and some books and articles about ray tracing. Should you want to use this stuff in your own hobby projects, do ask about possibilities of getting credit points. State-of-the-art methods are good topics for Bachelor / Master thesis projects.

This course will now teleport over algorithmic optimizations. (This is, of course, called “course optimization”!) – we haven’t covered all the fundamentals yet, so let’s not get stuck with details. The next few slides from the MIT course give references for further study (not necessary for our course).

Texture Mapping & Shaders

© Remedy Enterainment. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

MIT EECS 6.837 Computer Graphics

Spatial Variation

- All materials seen so far are the same everywhere
  - In other words, we are assuming the BRDF is independent of the surface point **x**
  - No real reason to make that assumption

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Courtesy of Fredo Durand. Used with permission.

Spatial Variation

- We will allow BRDF parameters to vary over space
  - This will give us much more complex surface appearance
  - e.g. diffuse color k_d vary with x
  - Other parameters/info can vary too: k_s, exponent, normal

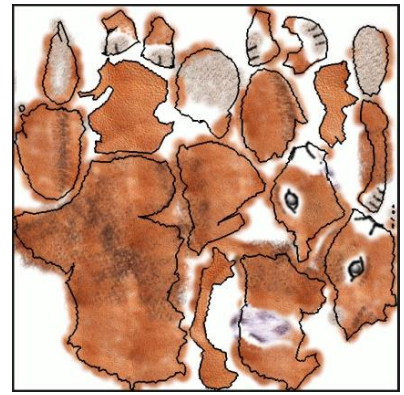
© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Courtesy of Fredo Durand. Used with permission.

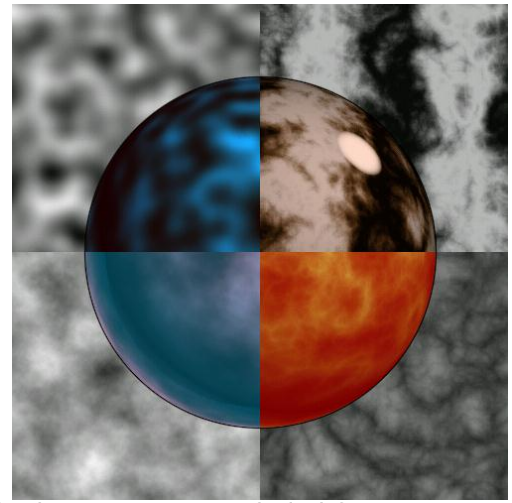
Two Approaches

- From data : texture mapping
  - read color and other information from 2D images



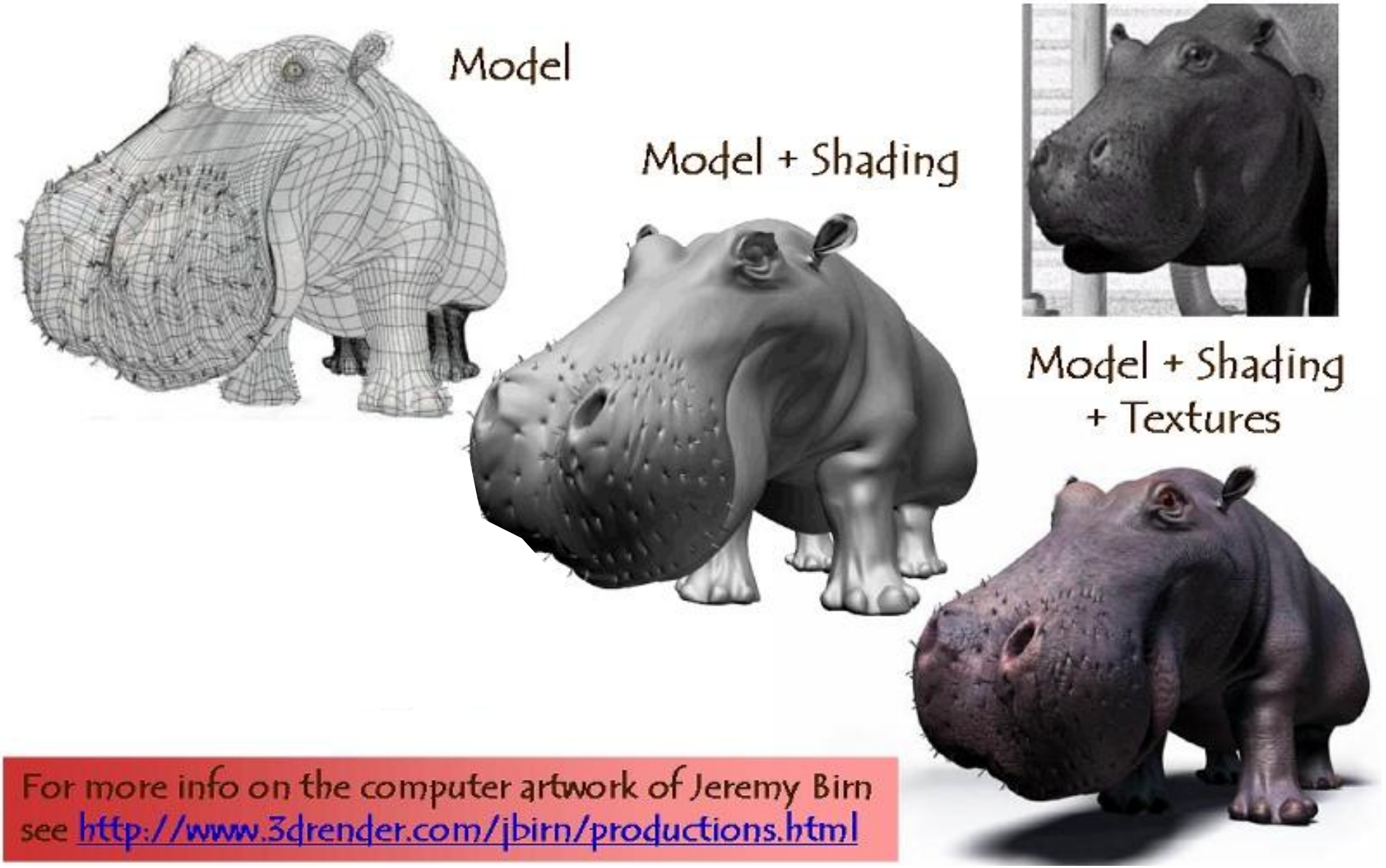
© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

- Procedural : shader
  - write little programs that compute color/info as a function of location



© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Effect of Textures



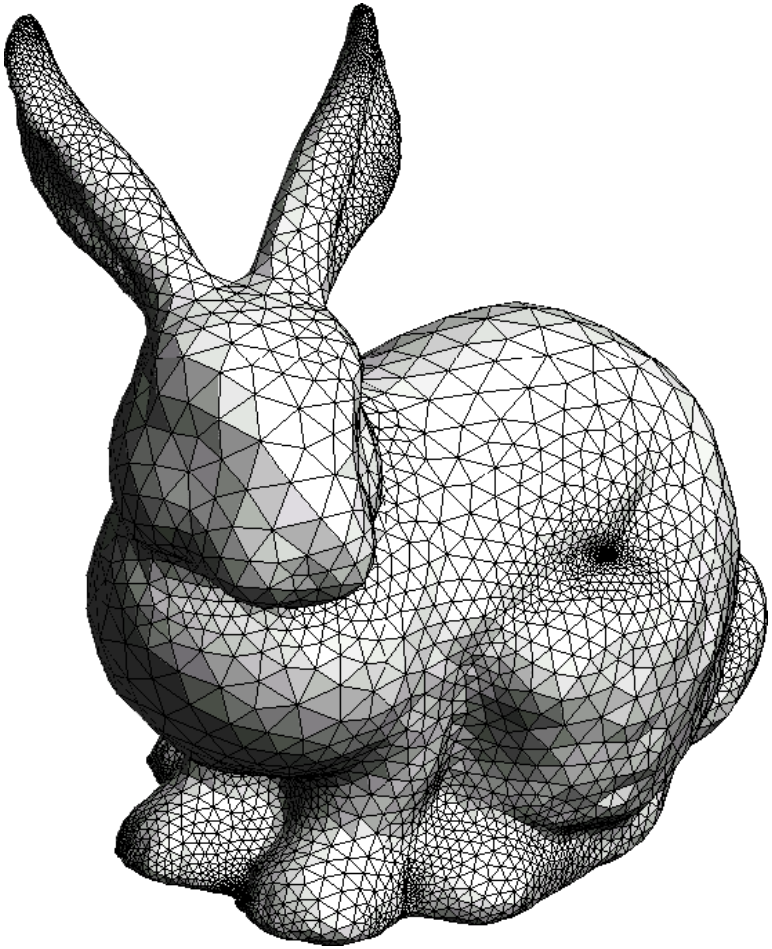
Courtesy of Jeremy Birn.

Texture Mapping

Image of a cartoon of a man applying wall paper has been removed due to copyright restrictions.

Texture Mapping

3D model



© Oscar Meruvia-Pastor, Daniel Rypl. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Texture mapped model

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Image: Praun et al.

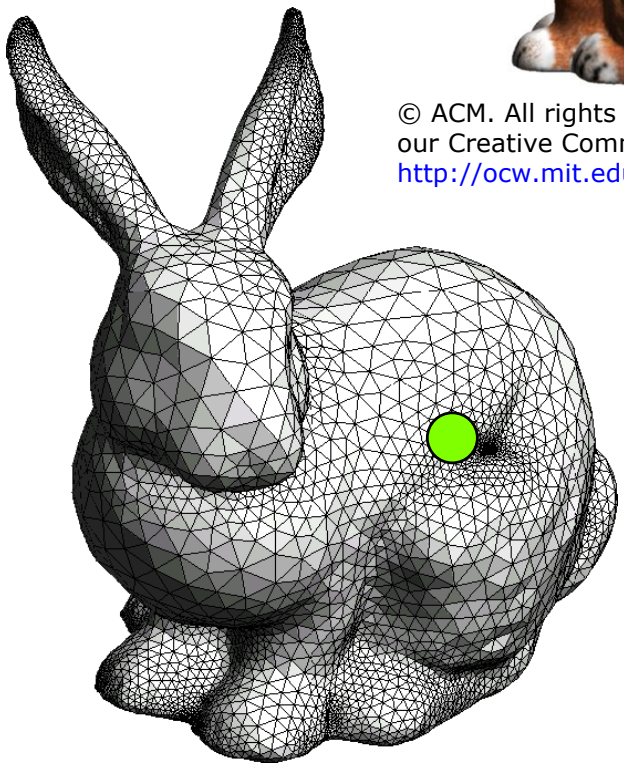
Texture Mapping

Image: Praun et al.

Texture mapped model

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

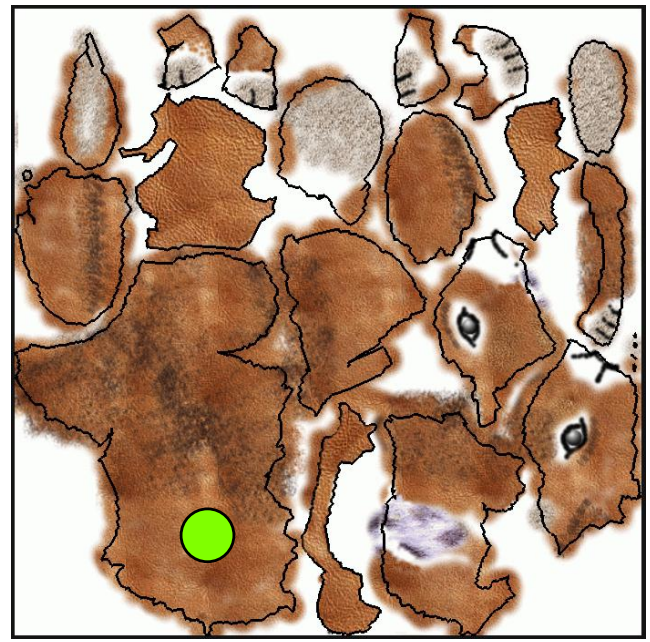
We need a function that associates each surface point with a 2D coordinate in the texture map



© Oscar Meruvia-Pastor, Daniel Rypl. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

?

Texture map (2D image)



© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Texture Mapping

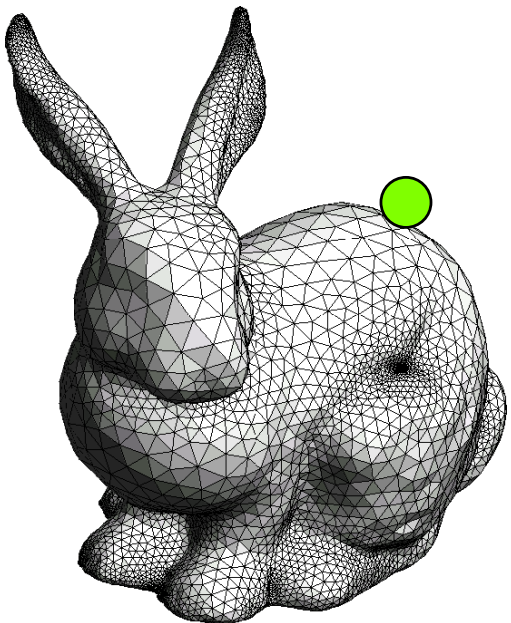
Image: Praun et al.

Texture mapped model

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

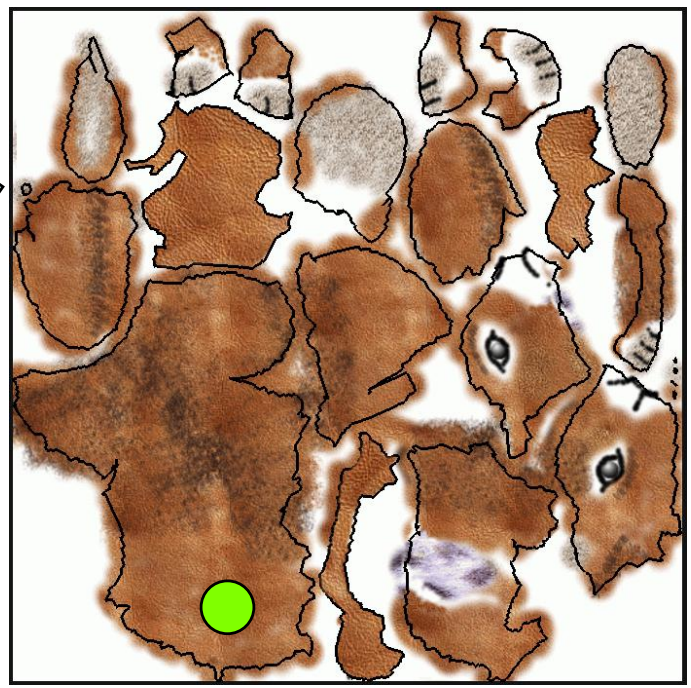
For each point rendered, look up color in texture map

Texture map (2D image)



© Oscar Meruvia-Pastor, Daniel Rypl. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

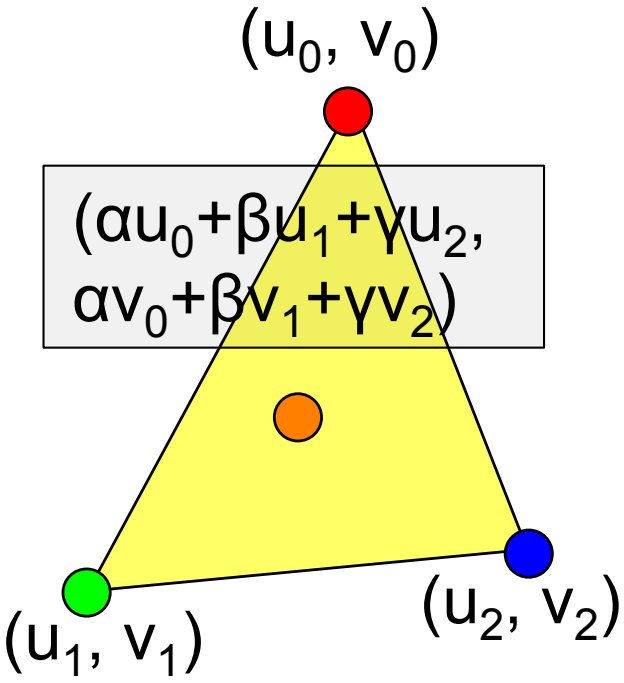
?

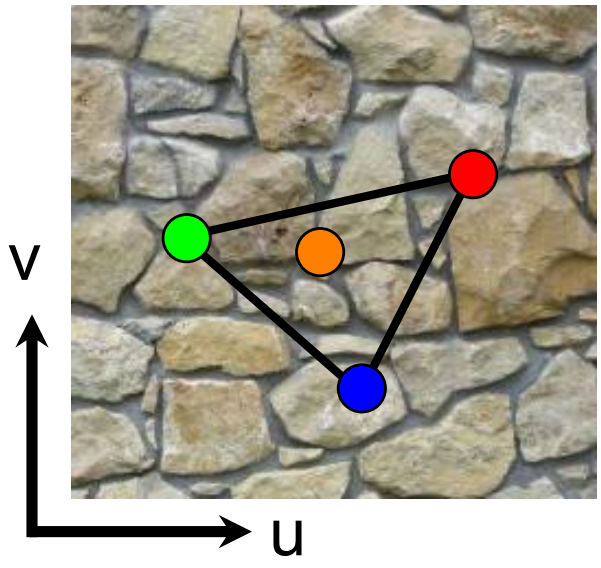


© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

UV Coordinates

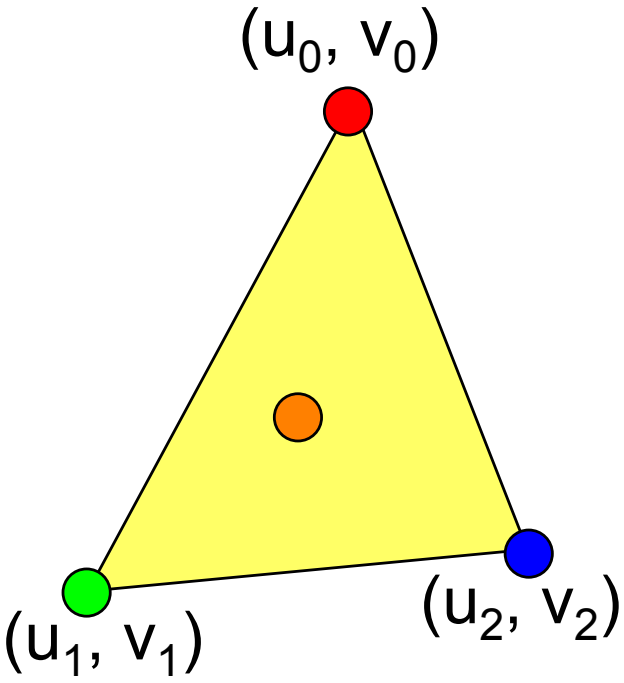
- Each vertex P stores 2D (u, v) "texture coordinates"
  - UVs determine the 2D location in the texture for the vertex
  - We will see how to specify them later
- Then we interpolate using barycentrics

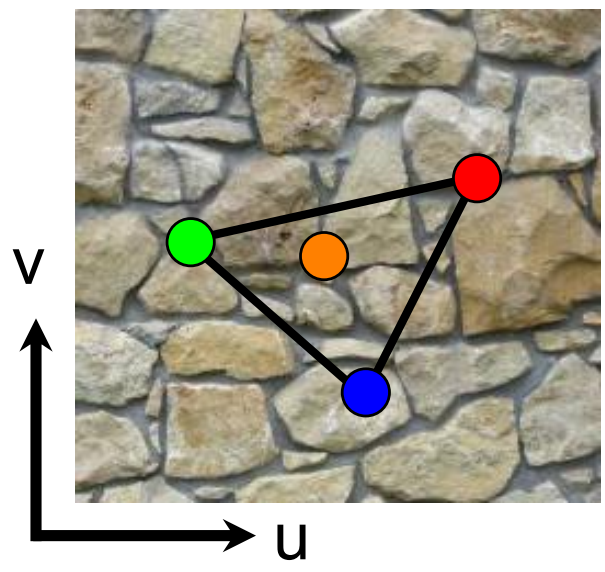




UV Coordinates

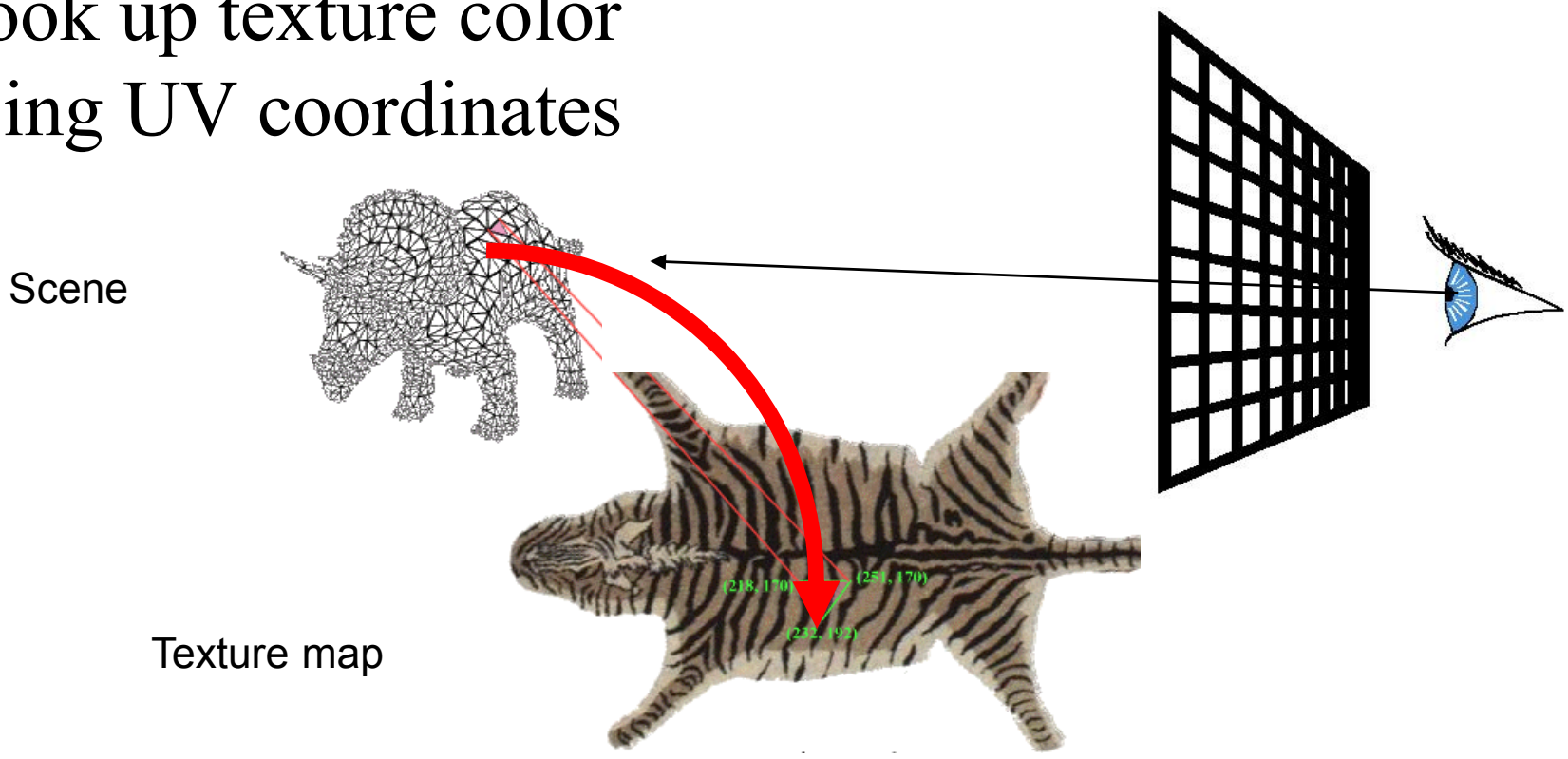
- Each vertex P stores 2D (u, v) "texture coordinates"
  - UVs determine the 2D location in the texture for the vertex
  - We will see how to specify them later
- Then we interpolate using barycentrics





Pseudocode – Ray Casting

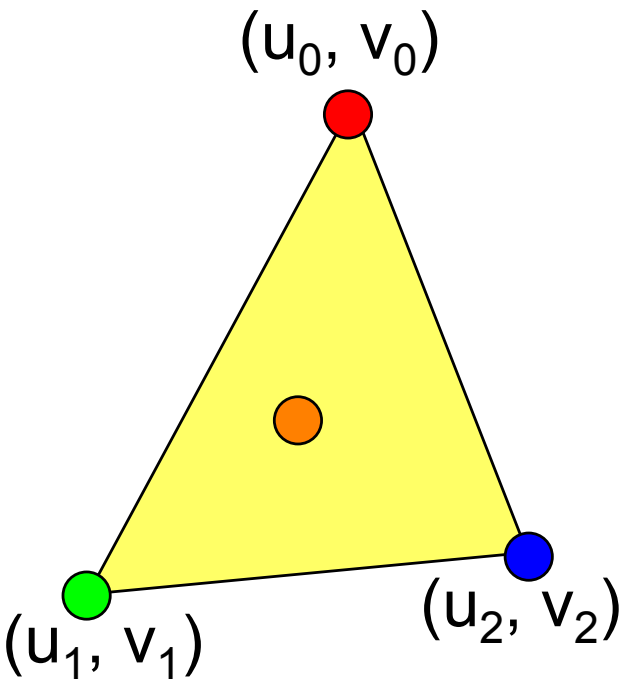
- Ray cast pixel (x, y), get visible point and α, β, γ
- Get texture coordinates (u, v) at that point
  - Interpolate from vertices using barycentrics
- Look up texture color using UV coordinates

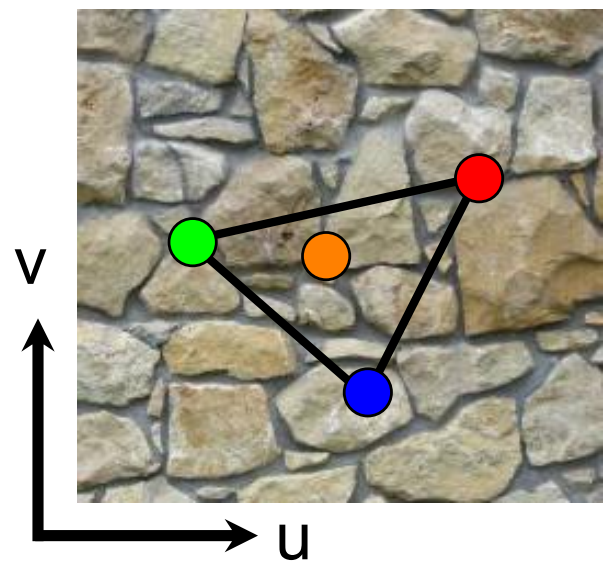


Leonard McMillan, Computer Science at the University of North Carolina in Chapel Hill.

UV Coordinates?

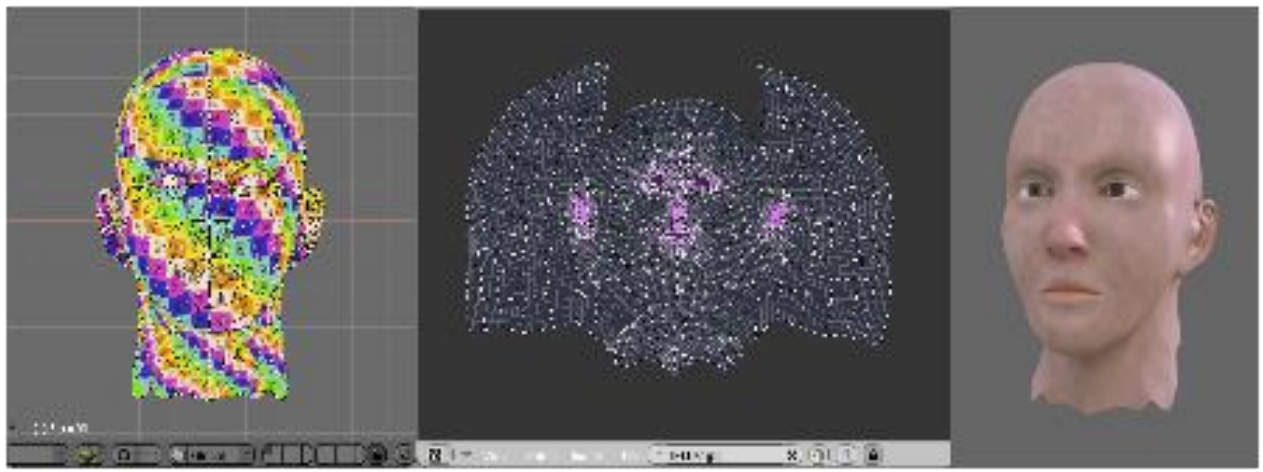
- Per-vertex (u, v) "texture coordinates" are specified:
  - Manually, provided by user (tedious!)
  - Automatically using parameterization optimization
  - Mathematical mapping (independent of vertices)





Texture UV Optimization

- Goal : “flatten” 3D object onto 2D UV coordinates
- For each vertex, find coordinates U,V such that distortion is minimized
  - distances in UV correspond to distances on mesh
  - angle of 3D triangle same as angle of triangle in UV plane
- Cuts are usually required (discontinuities)



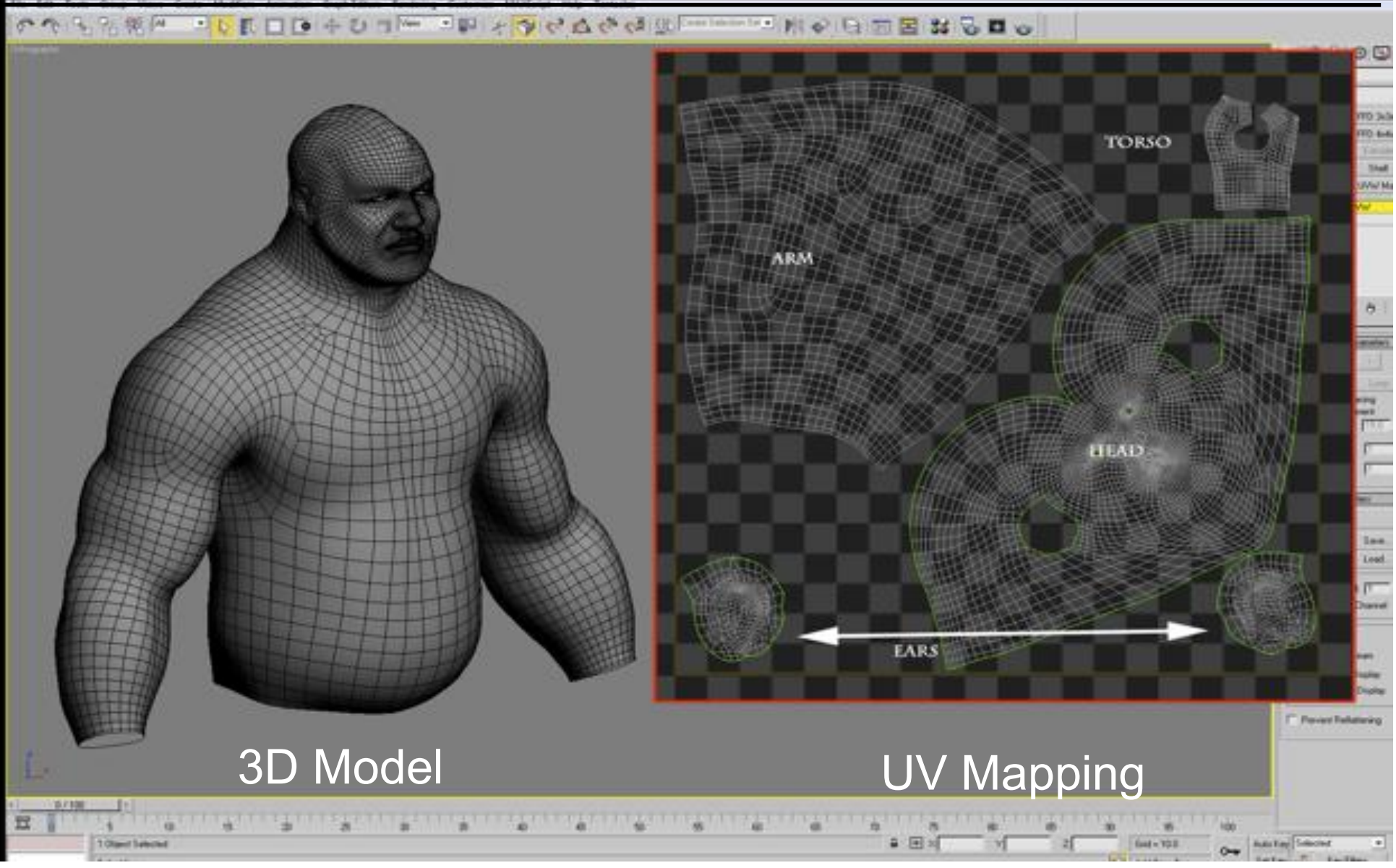
© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

To Learn More

- For this course, assume UV given per vertex
- Mesh Parameterization: Theory and Practice”
  - Kai Hormann, Bruno Lévy and Alla Sheffer *ACM SIGGRAPH Course Notes, 2007*
- http://alice.loria.fr/index.php/publications.html?redirect=0&Paper=SigCourseParam@2007&Author=Levy

Slide from Epic Games

Creating Torso Portion in Max

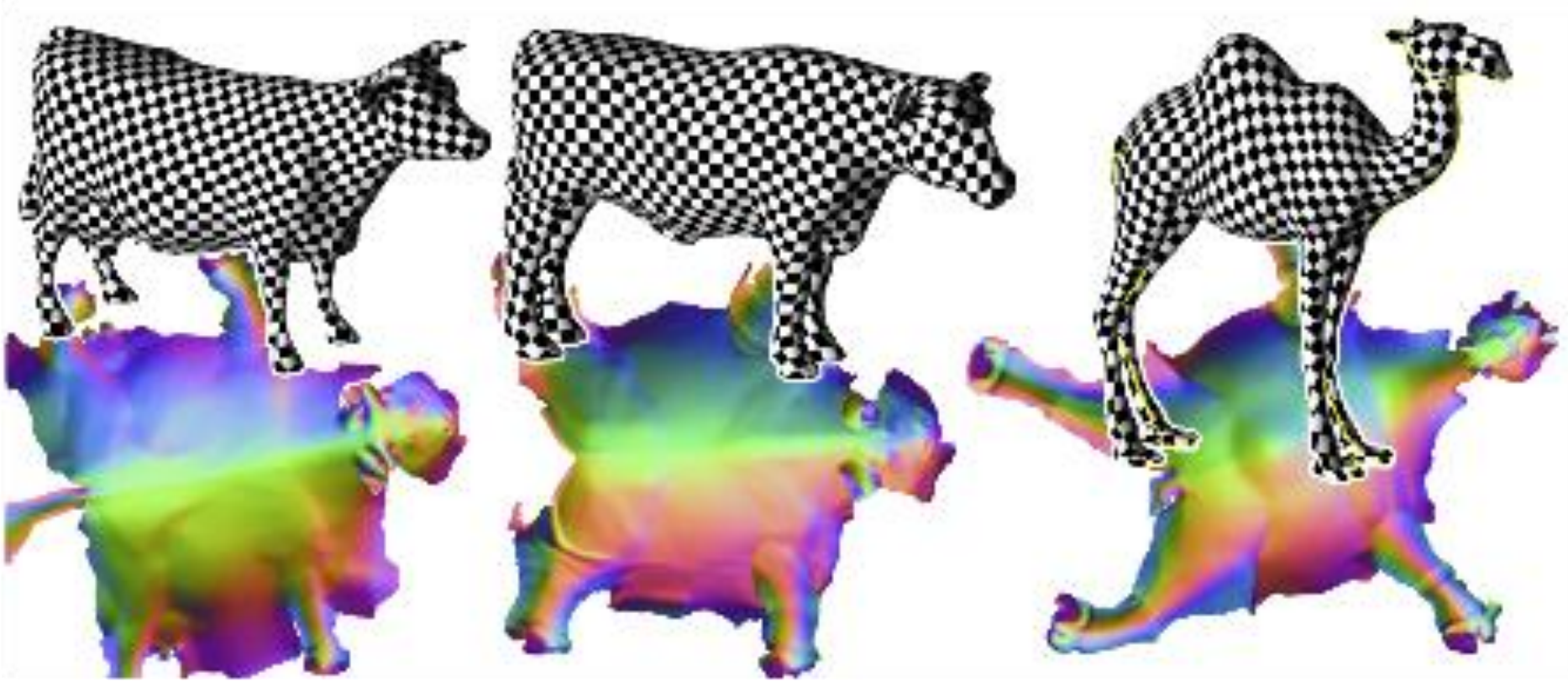


© Epic Games Inc. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

3D Model

- Information we need:
- Per vertex
  - 3D coordinates
  - Normal
  - 2D UV coordinates
- Other information
  - BRDF (often same for the whole object, but could vary)
  - 2D Image for the texture map

Questions?



Some results computed by stretch L_2 minimization (parameterized models courtesy of Pedro Sander and Alla Sheffer).

© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Mathematical Mapping

- What of non-triangular geometry?
  - Spheres, etc.
- No vertices, cannot specify UVs that way!
- Solution: Parametric Texturing
  - Deduce (u, v) from (x, y, z)
  - Various mappings are possible....

Common Texture Coordinate Mappings

- Planar
  - Vertex UVs and linear interpolation is a special case!
- Cylindrical
- Spherical
- Perspective Projection

Images removed due to copyright restrictions.

Projective Texture Example

- Modeling from photographs
- Using input photos as textures

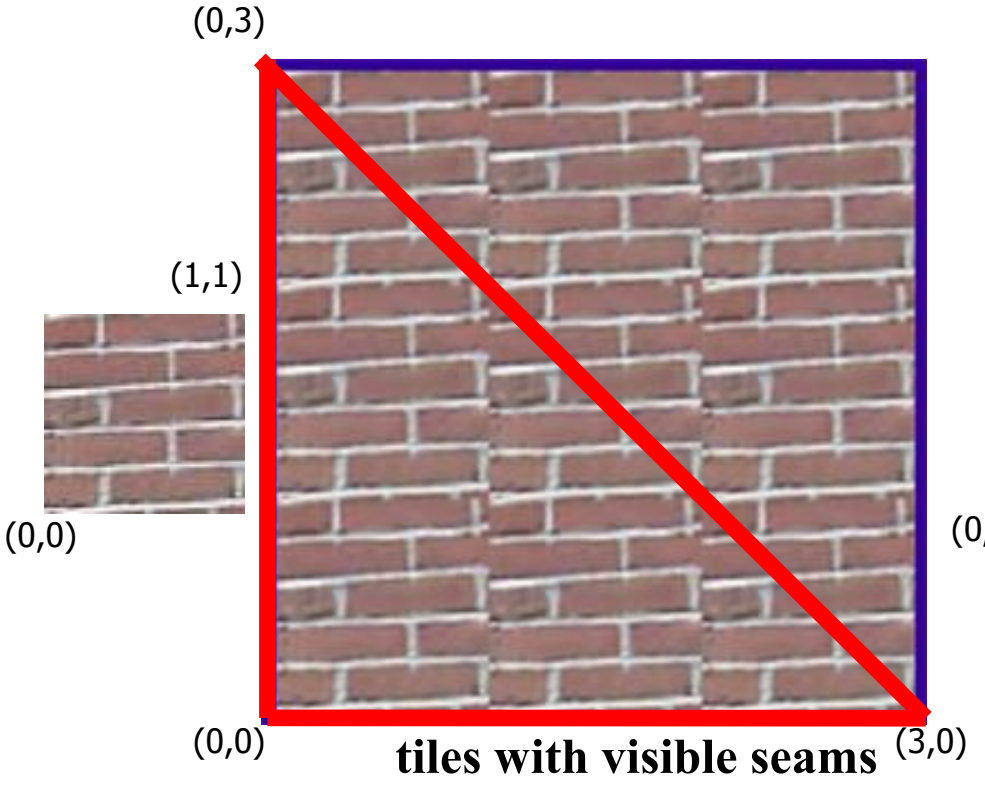
© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

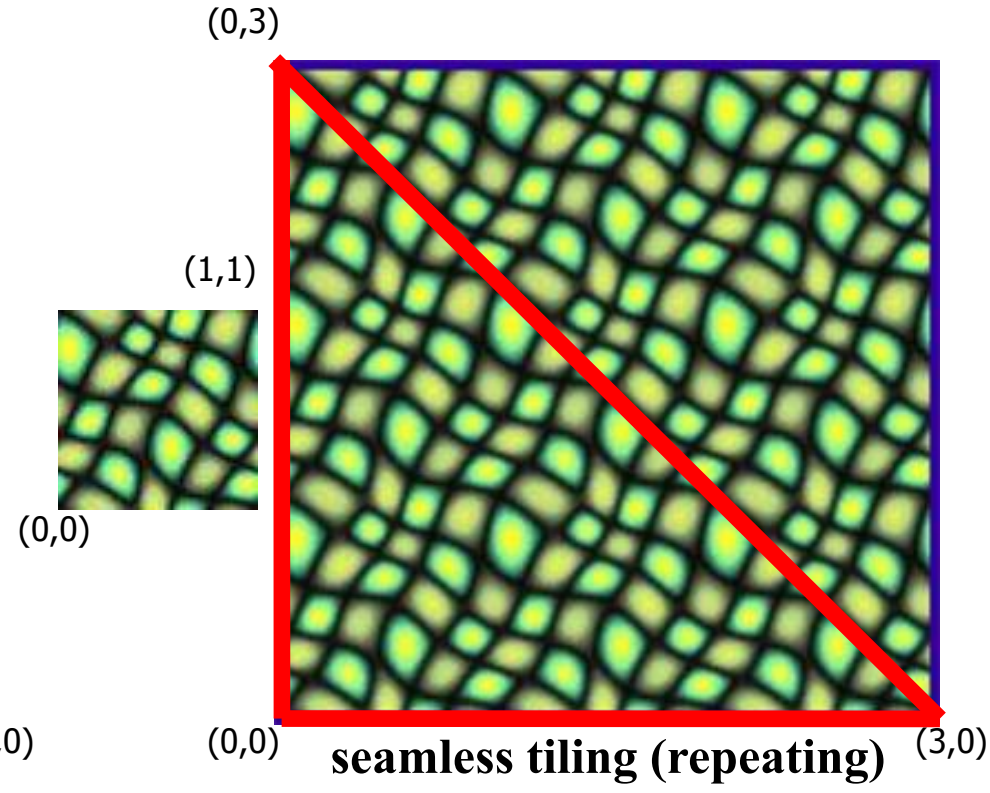
Figure from Debevec, Taylor & Malik
http://www.debevec.org/Research

Texture Tiling

Note the range (0,1) unlike normalized screen coordinates!

- Specify texture coordinates (u,v) at each vertex
- Canonical texture coordinates (0,0) → (1,1)
  - Wrap around when coordinates are outside (0, 1)



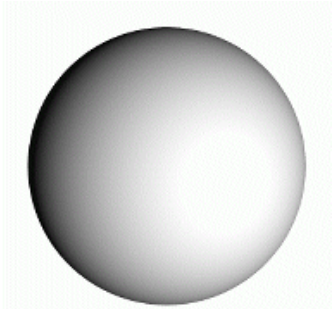


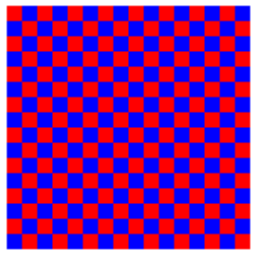
Texture Mapping & Illumination

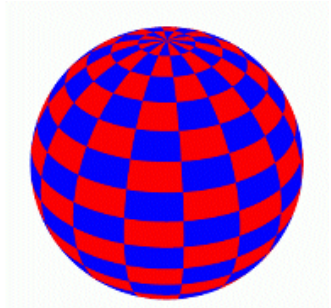
- Texture mapping can be used to alter some or all of the constants in the illumination equation
  - Diffuse color k_d, specular exponent q, specular color k_s...
  - Any parameter in any BRDF model!

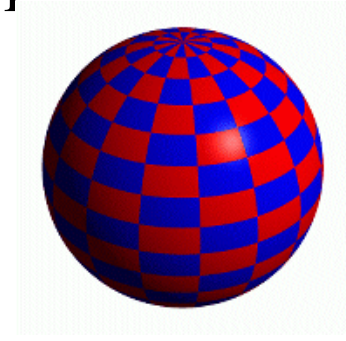
$$L_o = \left[k_a + k_d\left(\boldsymbol{n}\cdot\boldsymbol{l}\right) + k_s\left(\boldsymbol{v}\cdot\boldsymbol{r}\right)^q\right]\frac{L_i}{r^2}$$

  - k_d in particular is often read from a texture map


Constant Diffuse Color


Diffuse Texture Color


Texture used as Label


Texture used as Diffuse Color

Gloss Mapping Example

Ron Frazier

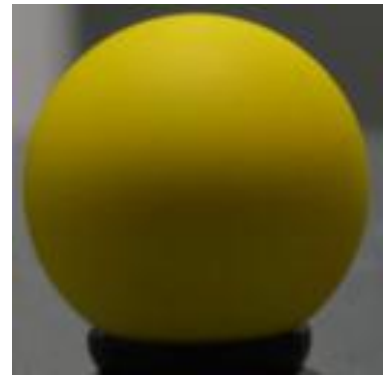
© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Spatially varying k_d and k_s

Questions?

We Can Go Even Further...

- The normal vector is really important in conveying the small-scale surface detail
  - Remember cosine dependence
  - The human eye is really good at picking up shape cues from lighting!



- We can exploit this and look up also the normal vector from a texture map
  - This is called “normal mapping” or “bump mapping”
  - A coarse mesh combined with detailed normal maps can convey the shape very well!

Normal Mapping

- For each shaded point, normal is given by a 2D image normalMap that stores the 3D normal

For a visible point

interpolate UV using barycentric

// same as texture mapping

Normal = normalMap[U,V]

compute shading (BRDF) using this normal

$$L_o = \left[k_a + k_d\left(\boldsymbol{n}\cdot\boldsymbol{l}\right) + k_s\left(\boldsymbol{v}\cdot\boldsymbol{r}\right)^q\right]\frac{L_i}{r^2}$$

Normal Map Example

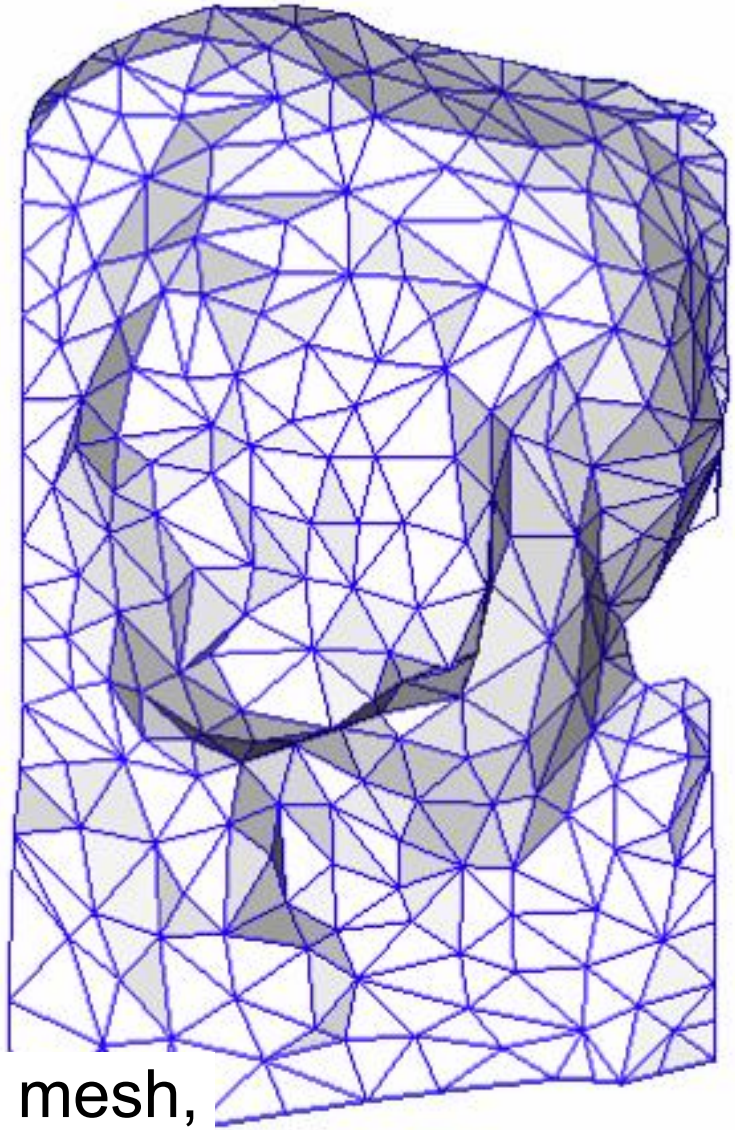
Paolo Cignoni

Original Mesh
4M triangles

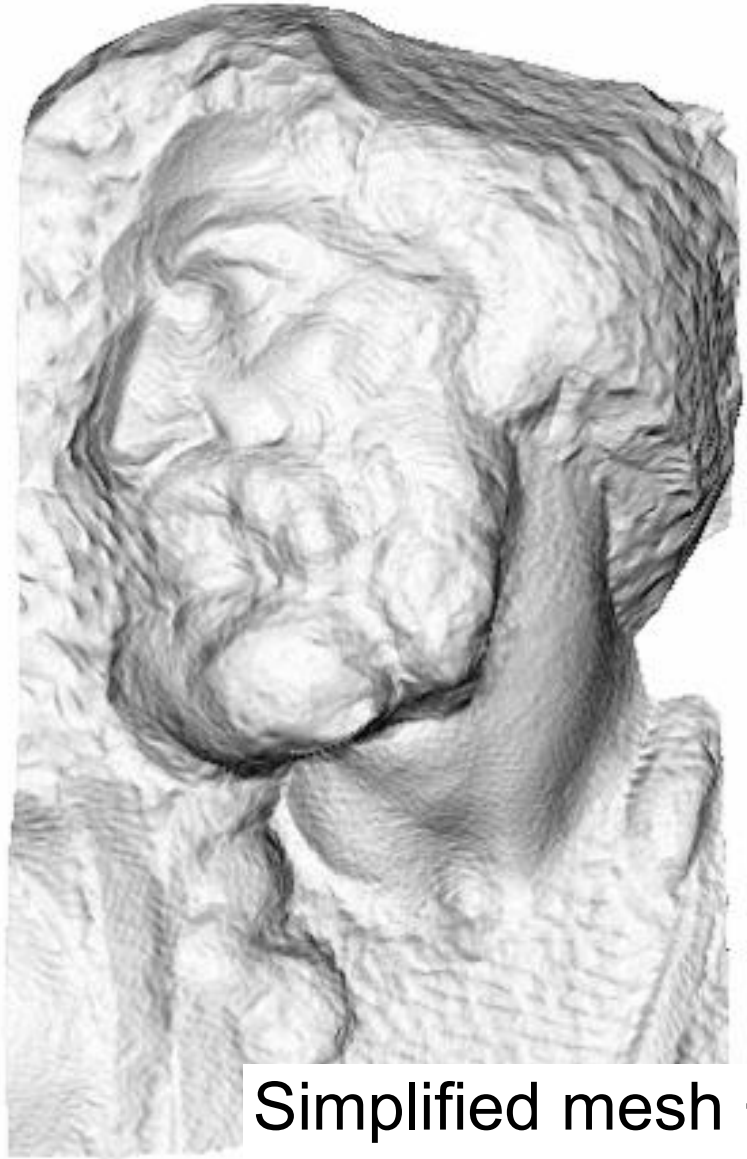
Image courtesy of Maksim on Wikimedia Commons. License: CC-BY-SA. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Normal Map Example

Paolo Cignoni



Simplified mesh,
500 triangles

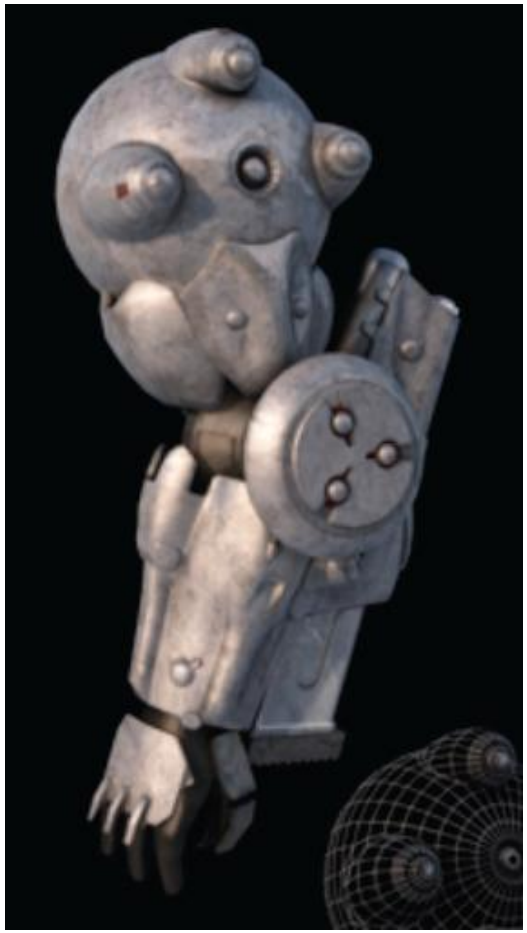


Simplified mesh +
normal mapping

Image courtesy of Maksim on Wikimedia Commons. License: CC-BY-SA. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Normal Map Example

Models and images: Trevor Taylor



Final render

Diffuse texture k_d

Normal Map

© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Generating Normal Maps

- Model a detailed mesh
- Generate a UV parameterization for the mesh
  - A UV mapping such that each 3D point has **unique** image coordinates in the 2D texture map
  - This is a difficult problem, but tools are available
    - E.g., the DirectX SDK has functionality to do this
- Simplify the mesh (again, see DirectX SDK)
- Overlay simplified and original model
- For each point **P** on the simplified mesh, find closest point **P'** on original model (ray casting)
- Store the normal at **P'** in the normal map. **Done!**

Normal Map Details

- You can store an object-space normal
  - Convenient if you have a unique parameterization
- ....but if you want to use a tiling normal map, this will not work
  - Must account for the curvature of the object!
  - Think of mapping this diffuse+normal map combination on a cylindrical tower
- Solution: Tangent space normal map
  - Encode a “difference” from the geometric normal in a local coord. system

Questions?

Image from Epic Games has been removed due to copyright restrictions.

Shaders (`Material` class)

- Functions executed when light interacts with a surface
- Constructor:
  - set shader parameters
- Inputs:
  - Incident radiance
  - Incident and reflected light directions
  - Surface tangent basis (anisotropic shaders only)
- Output:
  - Reflected radiance

Shader

- Initially for production (slow) rendering
  - Renderman in particular
- Now used for real-time (Games)
  - Evaluated by graphics hardware
  - More later in the course

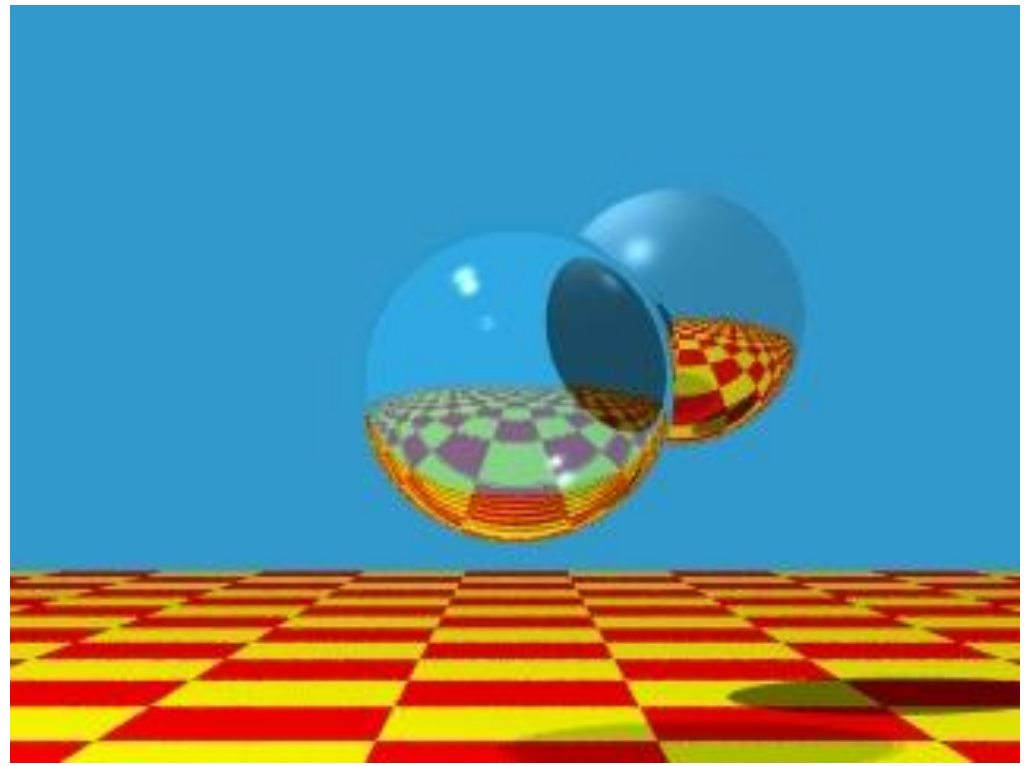
- Often makes heavy use of texture mapping

Questions?

Procedural Textures

- Alternative to texture mapping
- Little program that computes color as a function of *x,y,z*:

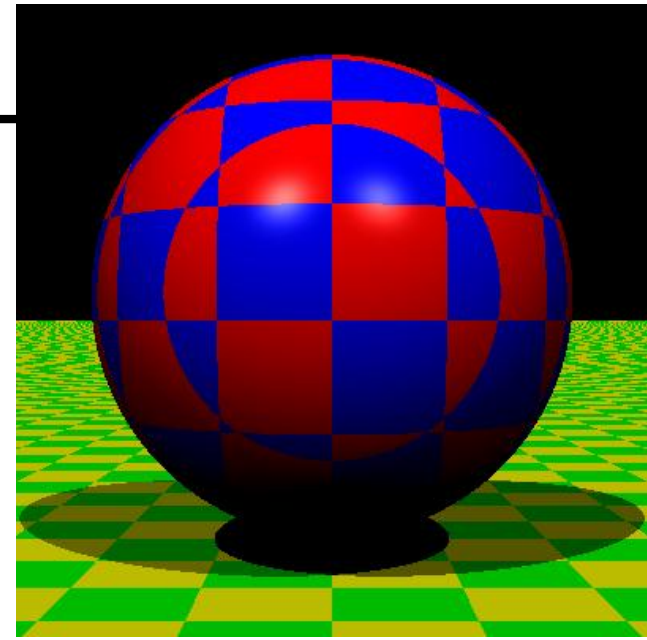
$f(x,y,z) \rightarrow color$



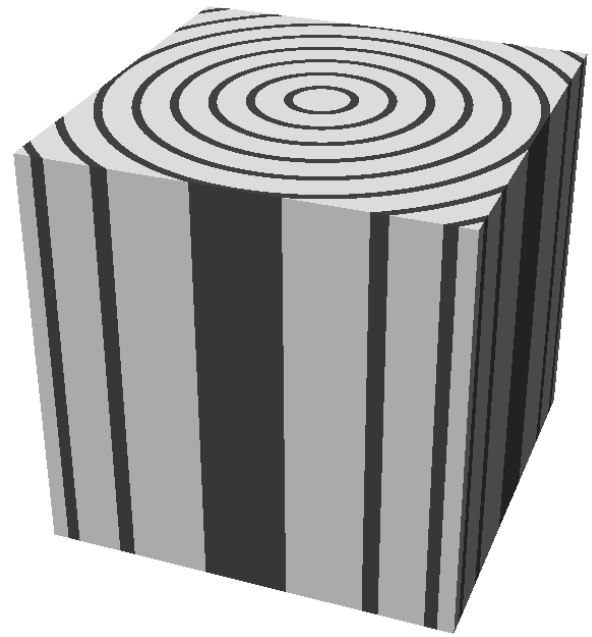
© Turner Whitted, Bell Laboratories. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Image by Turner Whitted

Procedural Textures



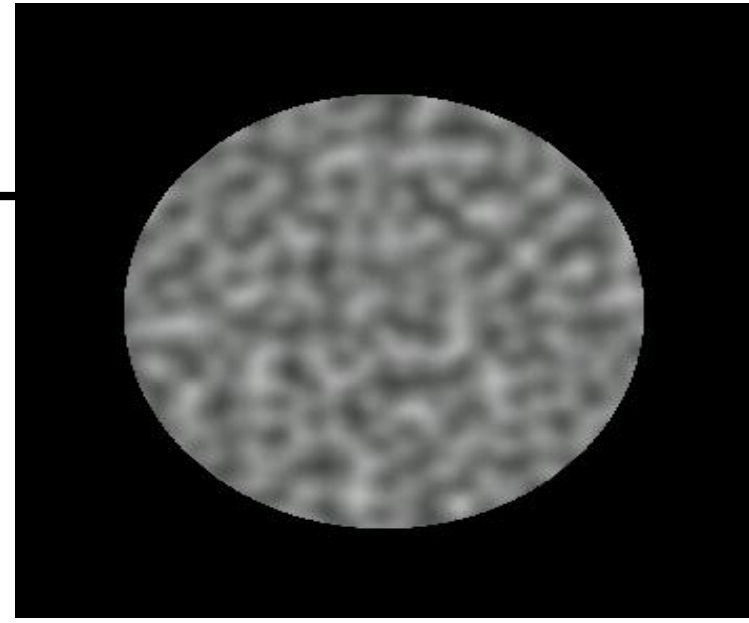
- Advantages:
  - easy to implement in ray tracer
  - more compact than texture maps (especially for solid textures)
  - infinite resolution



- Disadvantages
  - non-intuitive
  - difficult to match existing texture

Questions?

Perlin Noise



© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

- Critical component of procedural textures
- Pseudo-random function
  - But continuous
  - band pass (single scale)
- Useful to add lots of visual detail

http://www.noisemachine.com/talk1/index.html

http://mrl.nyu.edu/~perlin/doc/oscar.html

http://mrl.nyu.edu/~perlin/noise/

http://en.wikipedia.org/wiki/Perlin_noise

http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

(not really Perlin noise but very good)

http://portal.acm.org/citation.cfm?id=325247

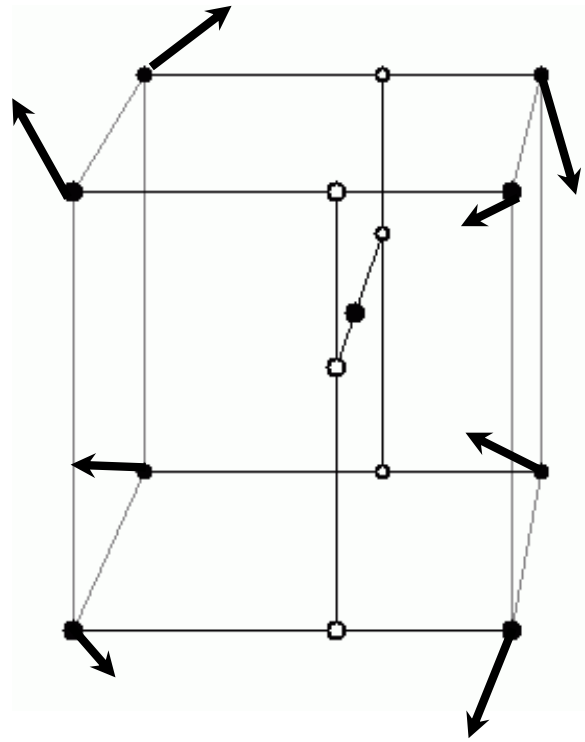
Requirements

- Pseudo random
- For arbitrary dimension
  - 4D is common for animation
- Smooth
- Band pass (single scale)
- Little memory usage

- How would you do it?

Perlin Noise

- Cubic lattice
- Zero at vertices
  - To avoid low frequencies
- Pseudo-random gradient at vertices
  - define local linear functions
- Splines to interpolate the values to arbitrary 3D points

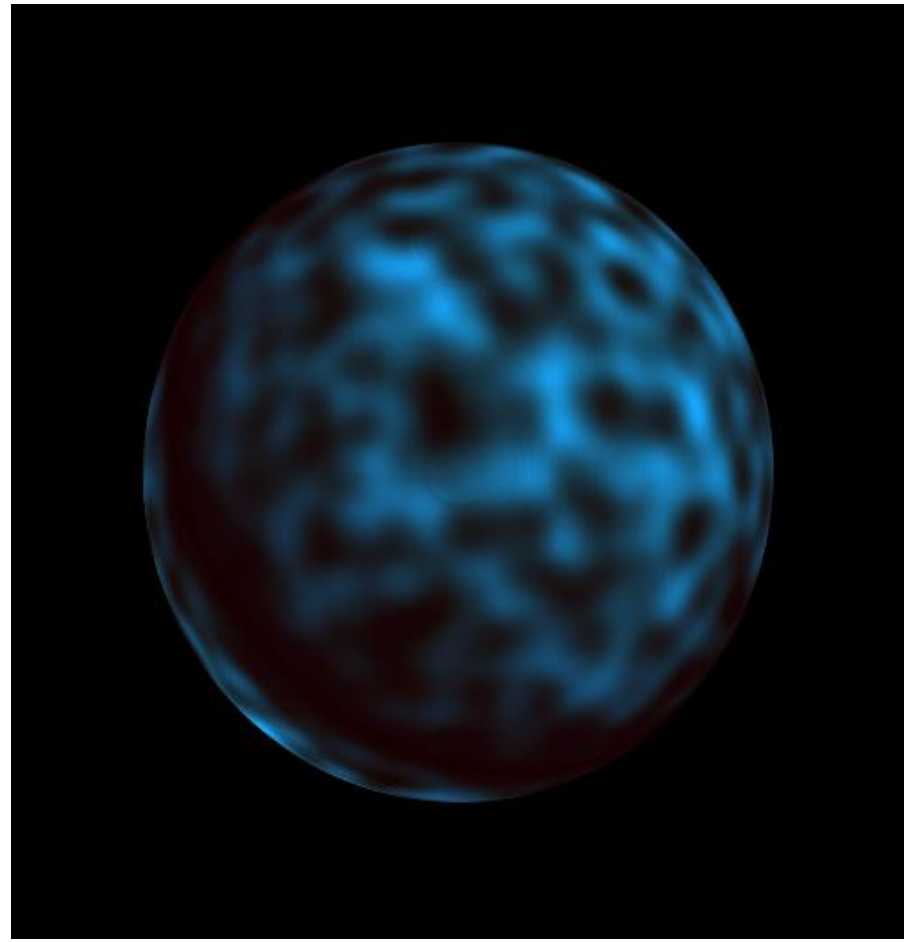


TIEA311 - Today in Jyväskylä

- Basic idea of Perlin noise is nicely introduced on “Lecture 16” of the original course material.
- We skip it here. I hope the follow-up course starting next week has time for this, among many other wonderful things.
- Pseudo-random noise is very easy to incorporate in real-time graphics shaders. If you want, you can just “copy-paste” code that you trust (and that has a license that allows inclusion in your current work!)
- Next, we proceed directly to applications of Perlin noise.

Noise At One Scale

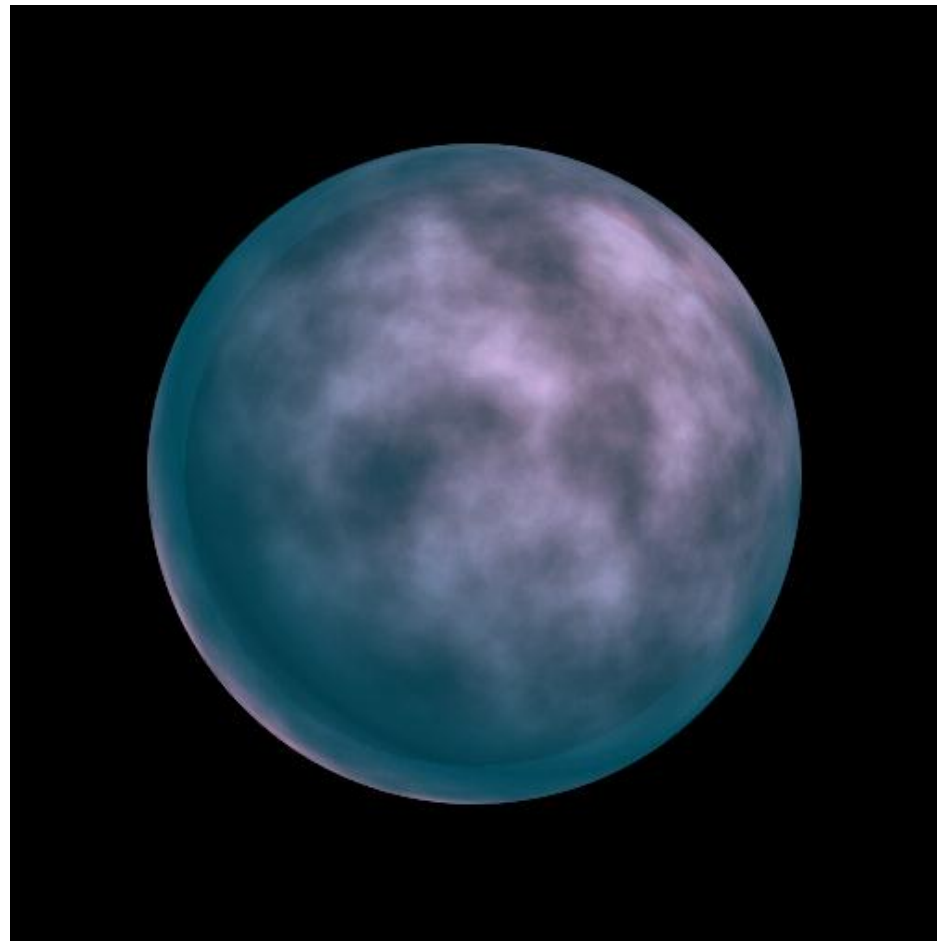
- A scale is also called an octave in noise parlance



© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Noise At Multiple Scales

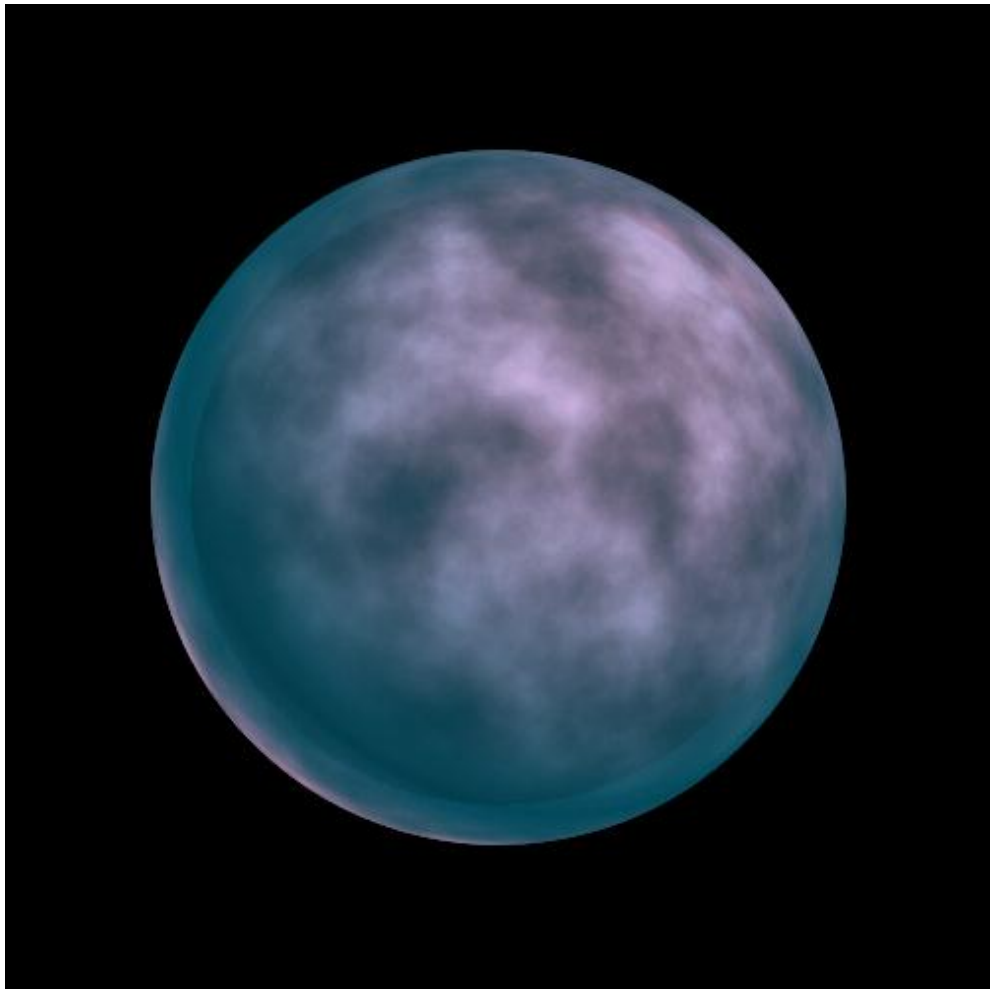
- A scale is also called an octave in noise parlance
- But multiple octaves are usually used, where the scale between two octaves is multiplied by 2
  - hence the name octave



© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Sum $1/f$ noise

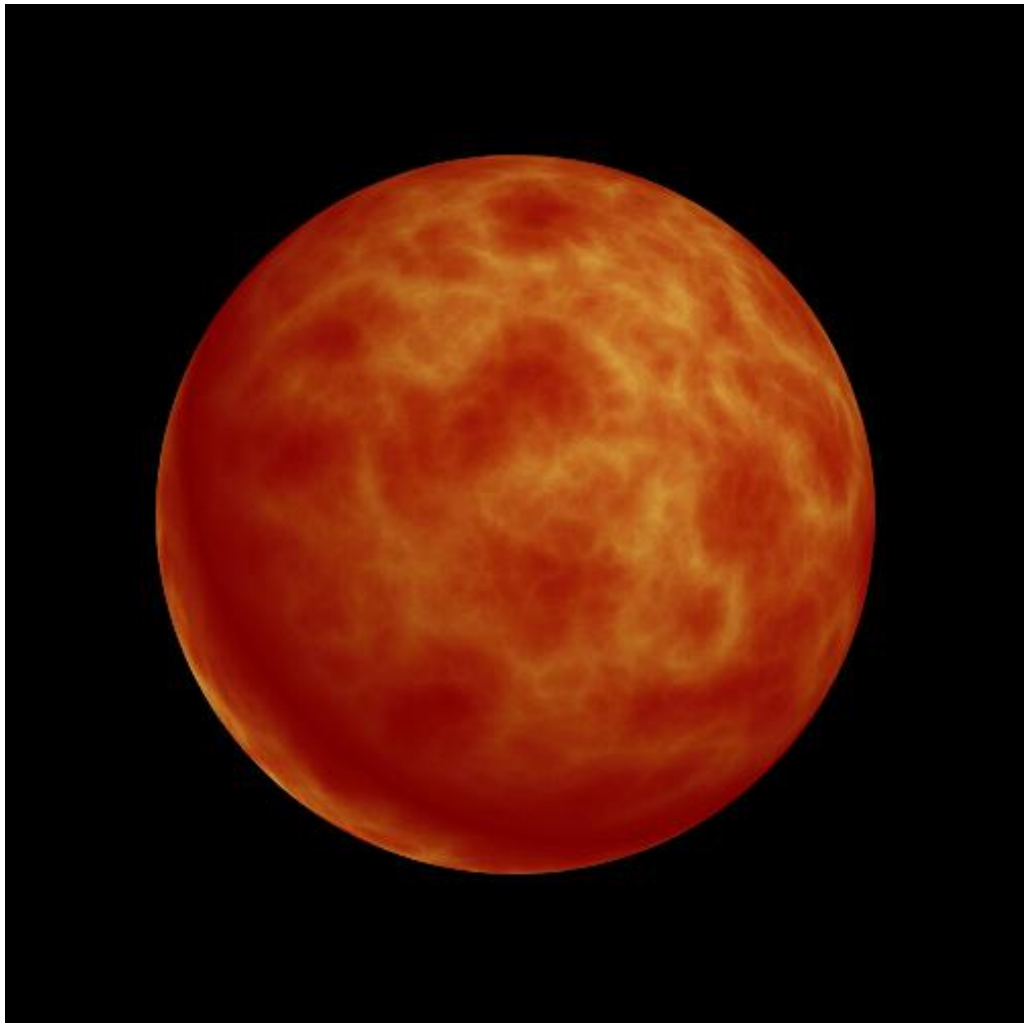
- That is, each octave f has weight $1/f$



© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

sum $1/f\,|noise|$

- Absolute value introduces *C1* discontinuities

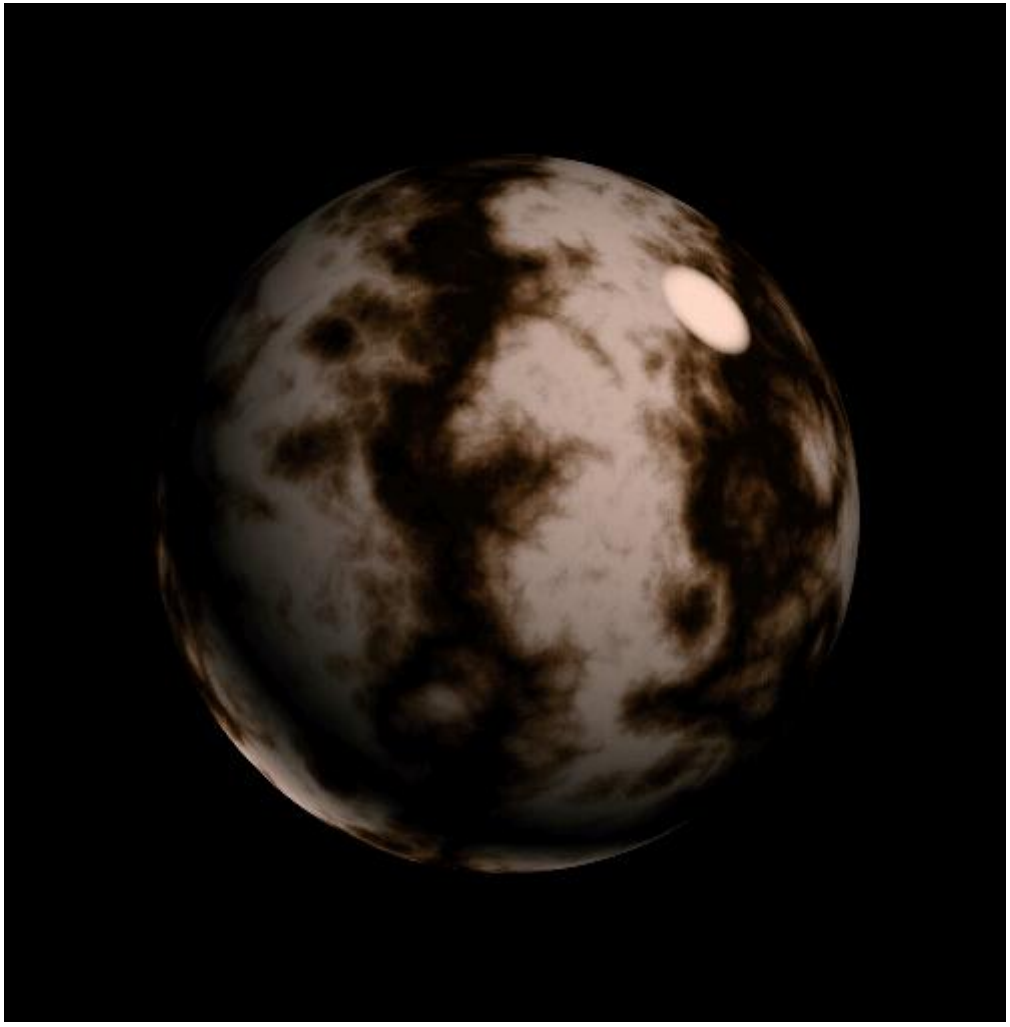


- a.k.a. turbulence

© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

$sin\ (x + \text{sum}\ 1/f\ |noise|)$

- Looks like marble!

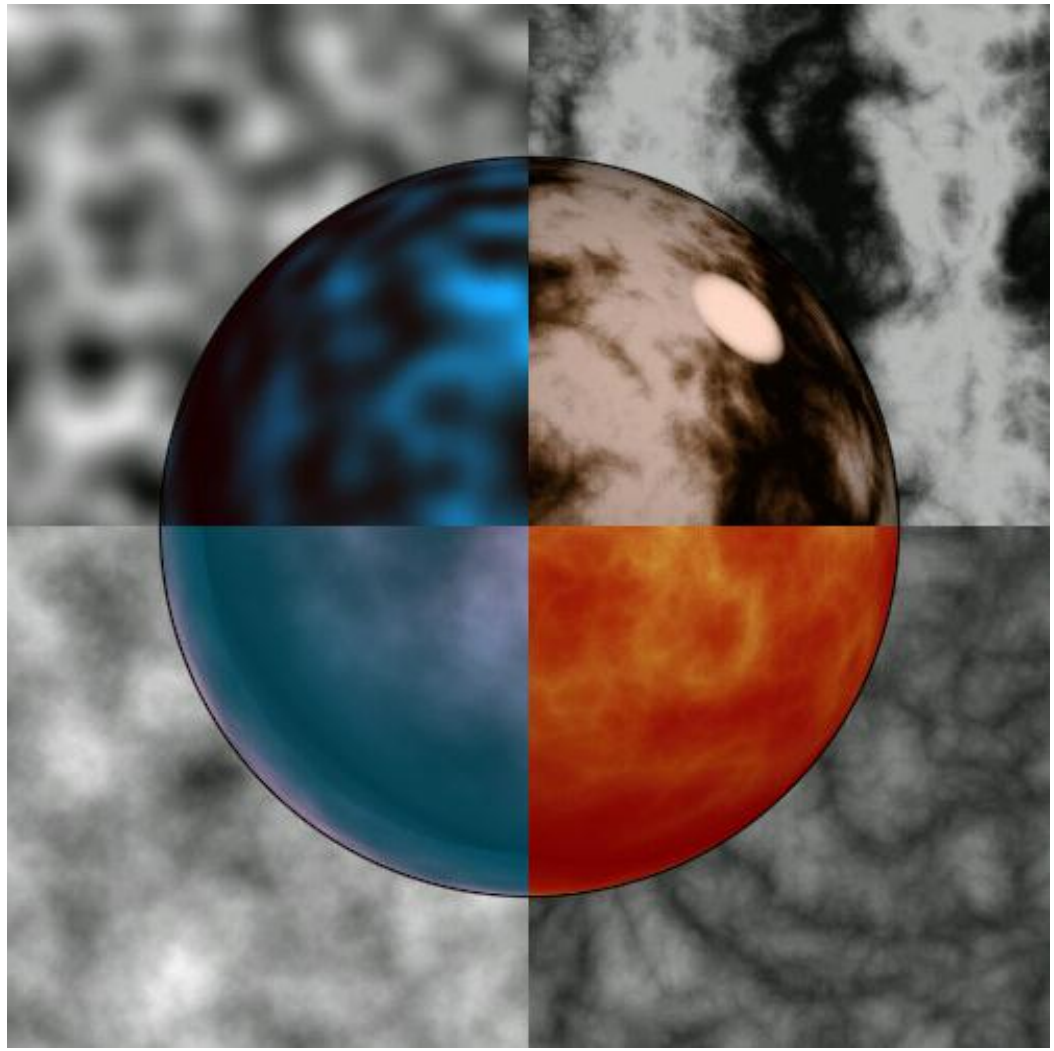


© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Comparison

•*noise*

sin(*x* + sum 1/*f*(|*noise*|))



sum 1/*f*(*noise*)

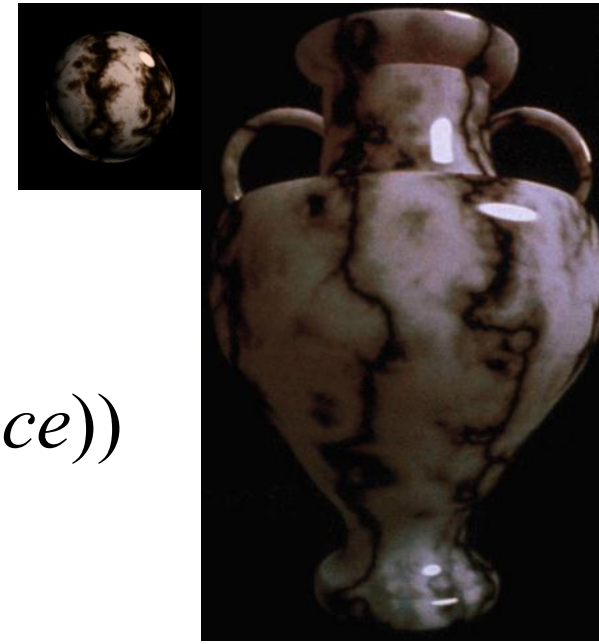
sum 1/*f*(|*noise*|)

© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

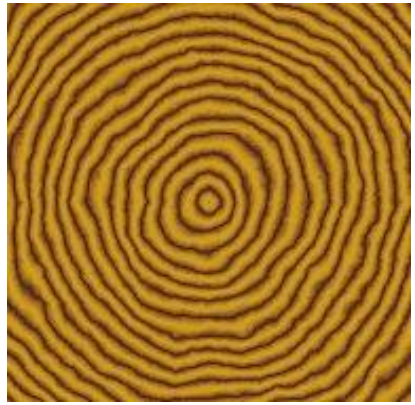
Questions?

Noise For Solid Textures

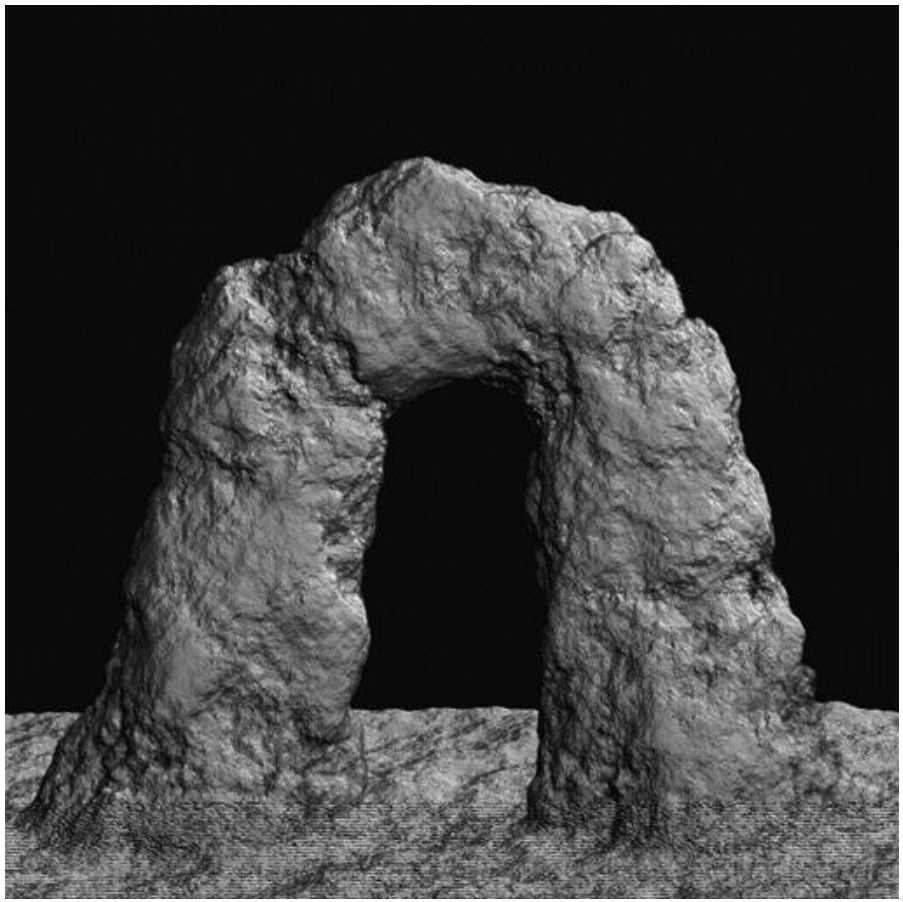
- Marble
  - recall *sin* (*x*[0] + sum 1/*f* |*noise*|)
  - *BoringMarble* = *colormap* (*sin*(*x*[0])
  - *Marble* = *colormap* (*sin*(*x*[0]+*turbulence*))
  - http://legakis.net/justin/MarbleApplet/
- Wood
  - replace *x* (or parallel plane) by radius
  - *Wood* = *colormap* (*sin*(*r*+*turbulence*))
  - http://www.connectedpixel.com/blog/texture/wood

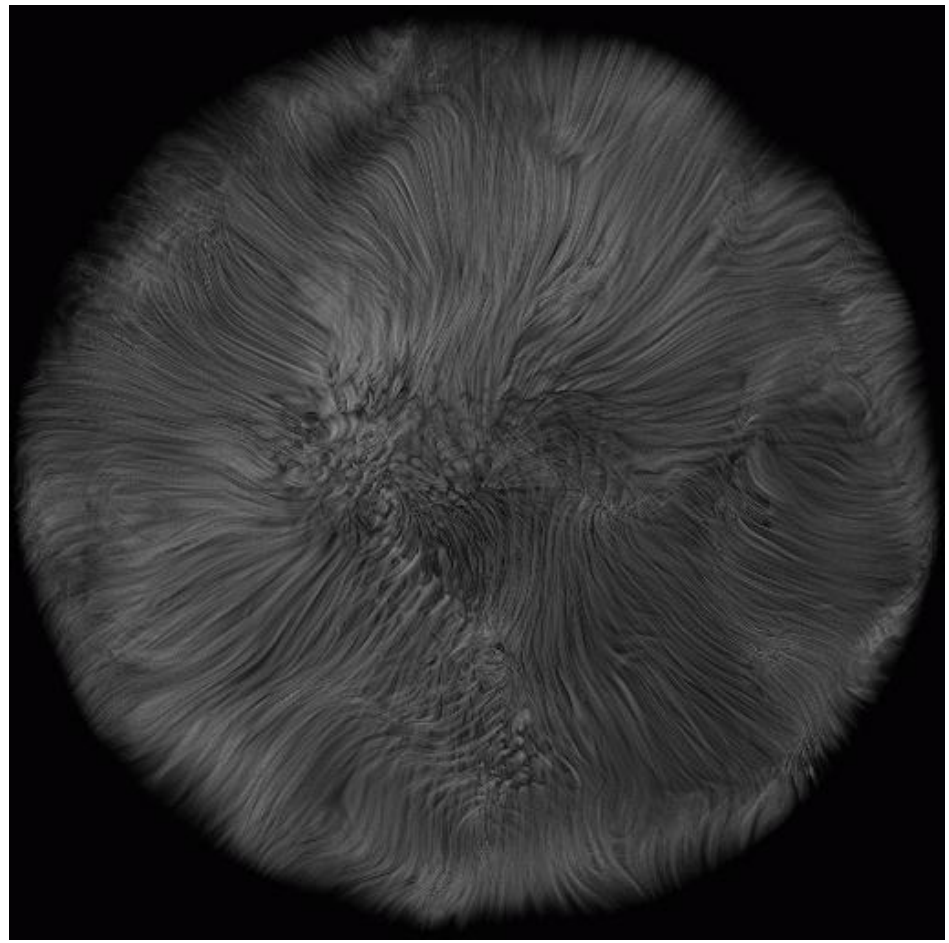


© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.



Other Cool Usage: Displacement, Fur





© Ken Perlin. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Questions?

Image removed due to copyright restrictions. Please the image of “blueglass.gif” from http://mrl.nyu.edu/~perlin/imgs/imgs.html.

Shaders

- Noise: one ingredient of shaders
- Can also use textures
- Shaders control diffuse color, but also specular components, maybe even roughness (exponent), transparency, etc.
- Shaders can be layered (e.g. a layer of dust, peeling paint, mortar between bricks).
- Notion of shade tree
  - Pretty much algebraic tree
- Assignment 5:
  checkerboard shader based on two shaders

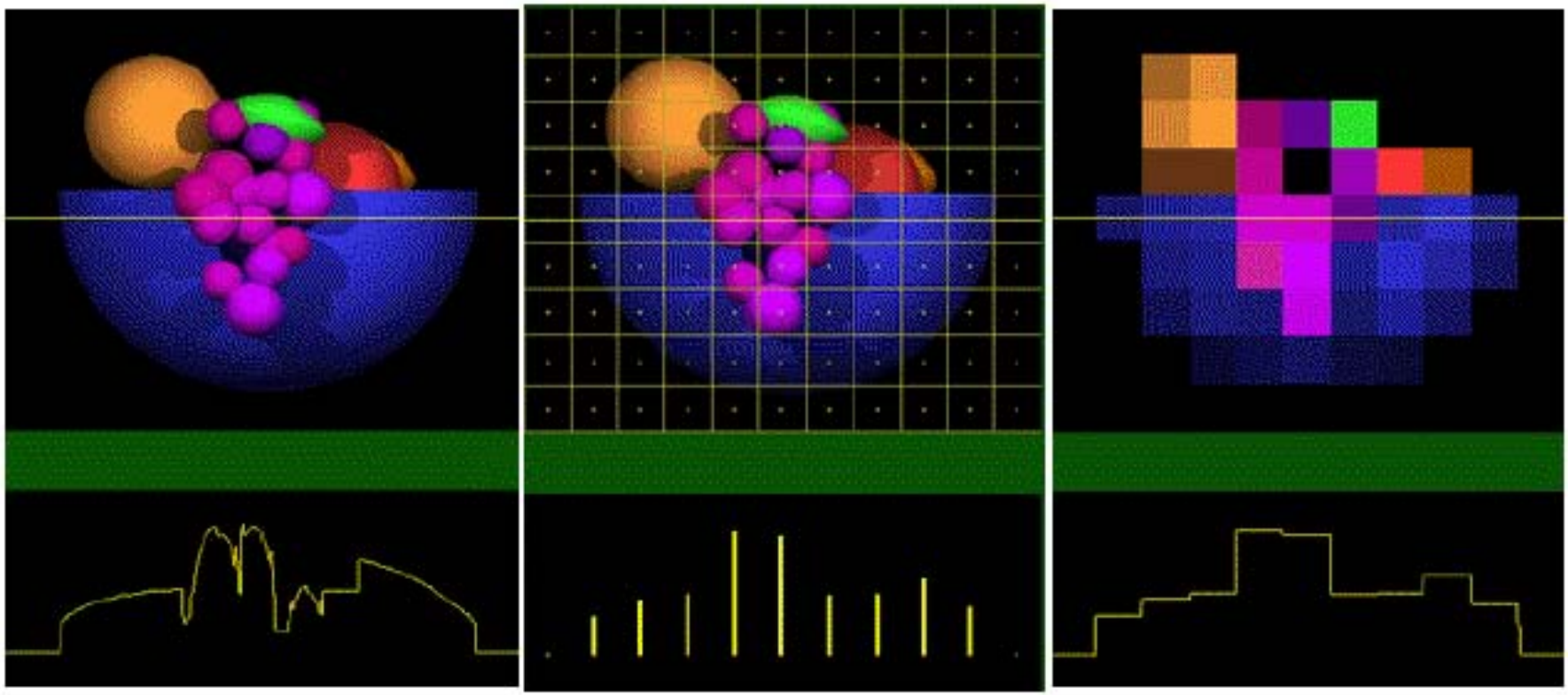
Bottom Line

- Programmable shader provide great flexibility
- Shaders can be extremely complex
  - 10,000 lines of code!
- Writing shaders is a black art

Sampling, Aliasing, & Mipmaps

MIT EECS 6.837 Computer Graphics
Wojciech Matusik, MIT EECS

Examples of Aliasing

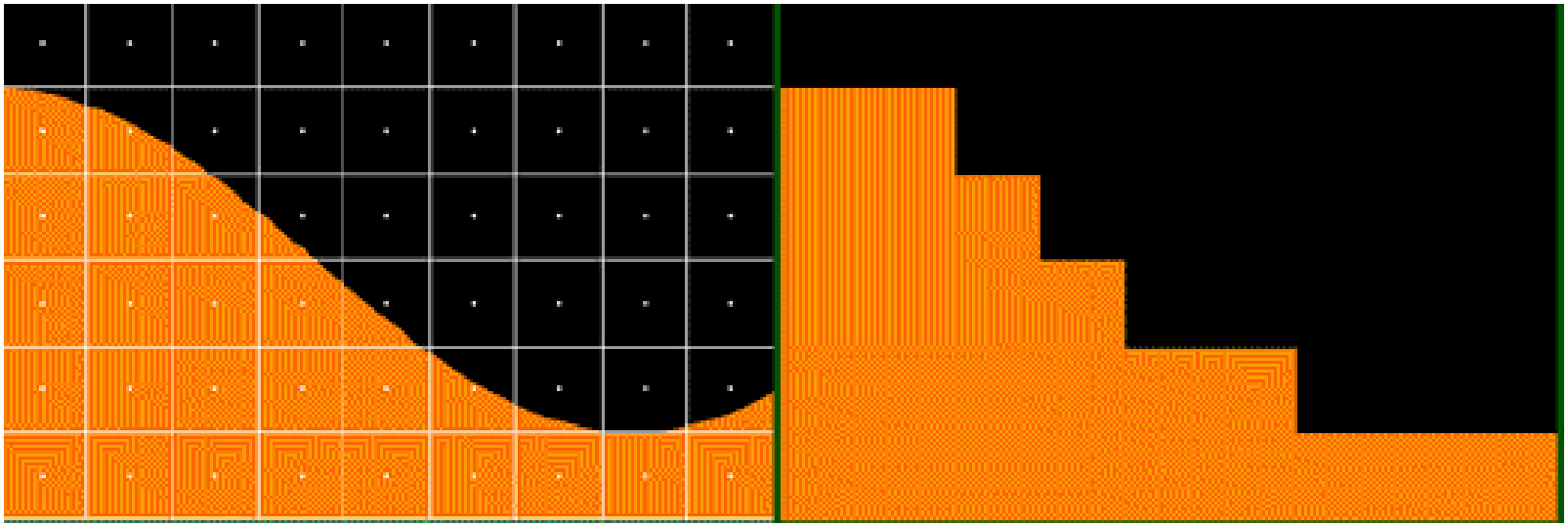


Original Image Samples Reconstruction

© Rosalee Nerheim-Wolfe, Toby Howard, Stephen Spencer. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Examples of Aliasing

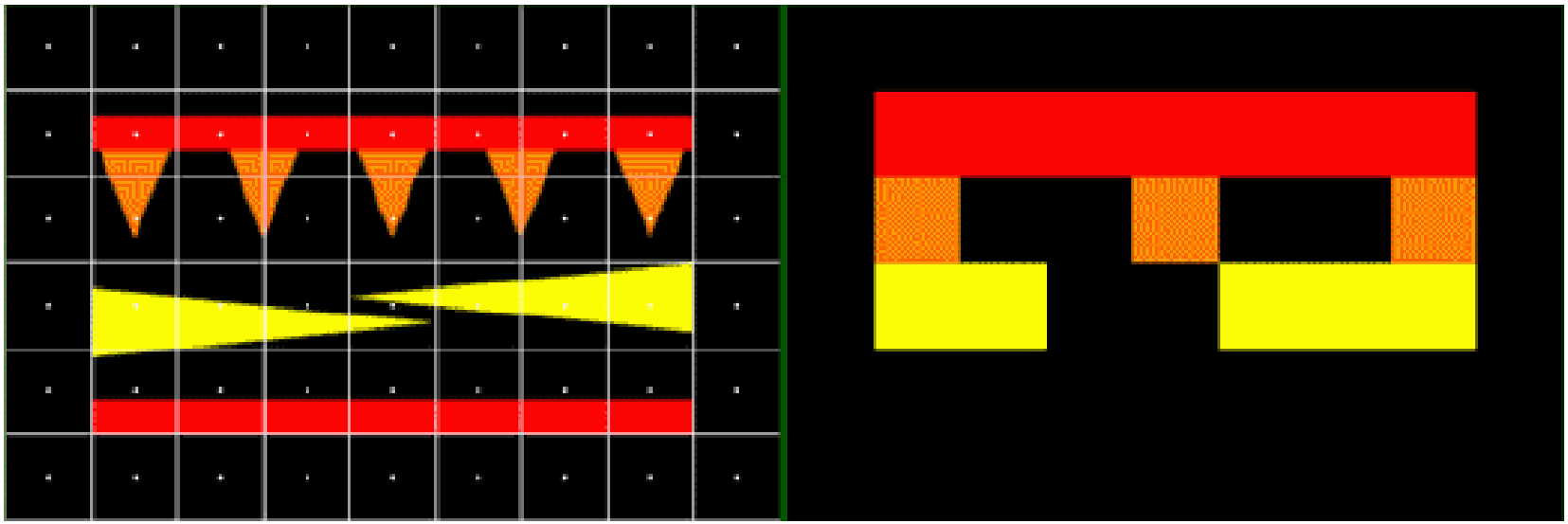
Jagged boundaries



© Rosalee Nerheim-Wolfe, Toby Howard, Stephen Spencer. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Examples of Aliasing

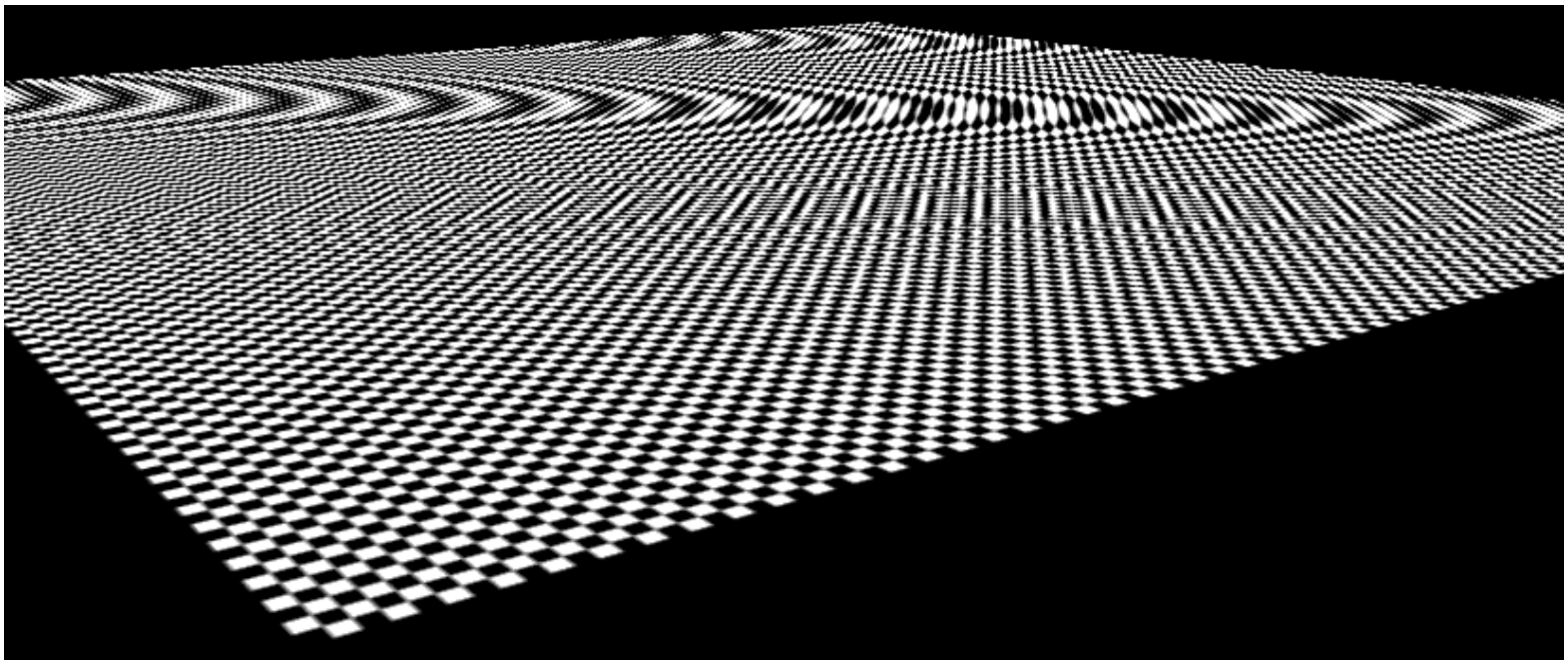
Improperly rendered detail



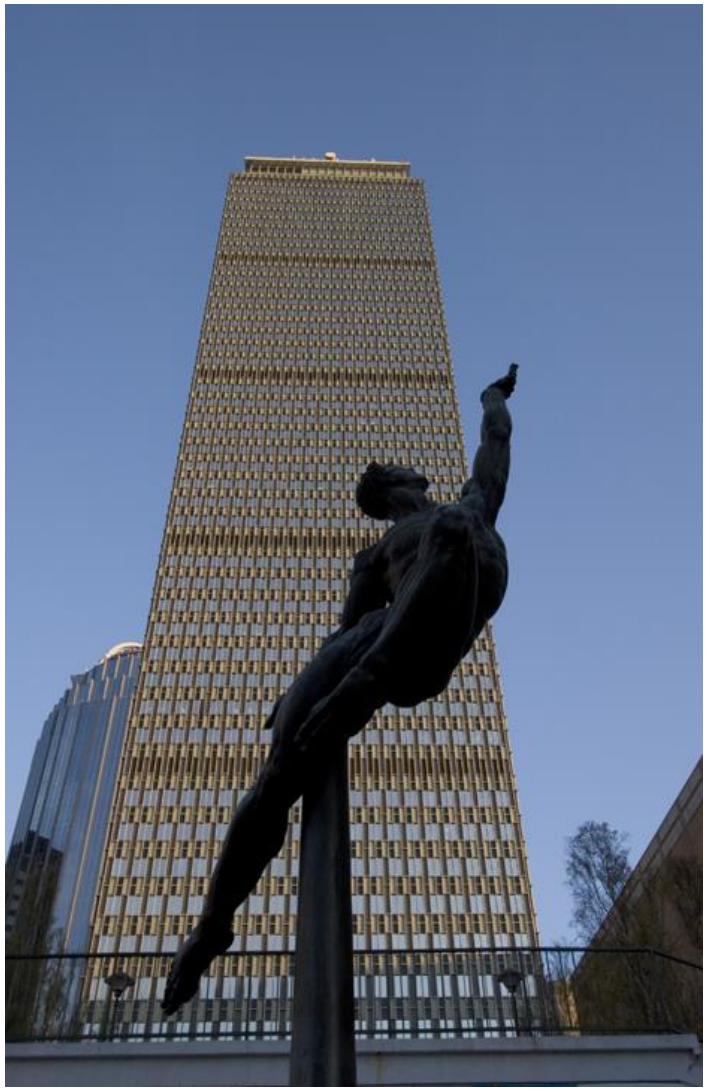
© Rosalee Nerheim-Wolfe, Toby Howard, Stephen Spencer. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Examples of Aliasing

Texture Errors



In photos too



© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

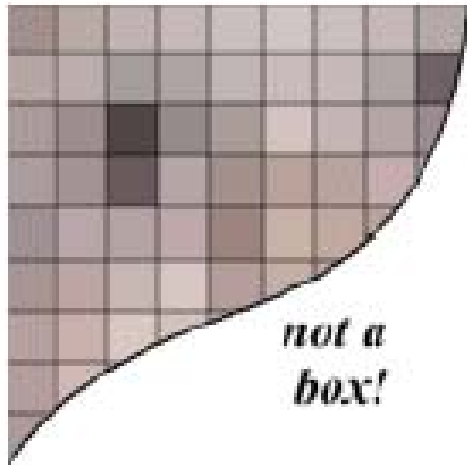
See also http://vimeo.com/26299355

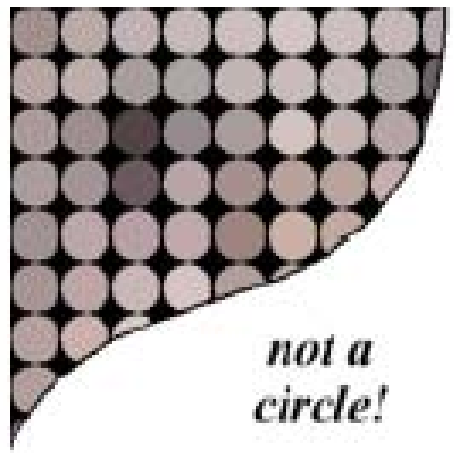
Philosophical perspective

- The physical world is continuous, inside the computer things need to be discrete
- Lots of computer graphics is about translating continuous problems into discrete solutions
  - e.g. ODEs for physically-based animation, global illumination, meshes to represent smooth surfaces, rasterization, antialiasing
- Careful mathematical understanding helps do the right thing

What is a Pixel?

- A pixel is not:
  - a box
  - a disk
  - a teeny tiny little light
- A pixel "looks different" on different display devices
- A pixel is a sample
  - it has no dimension
  - it occupies no area
  - it cannot be seen
  - it has a coordinate
  - it has a value

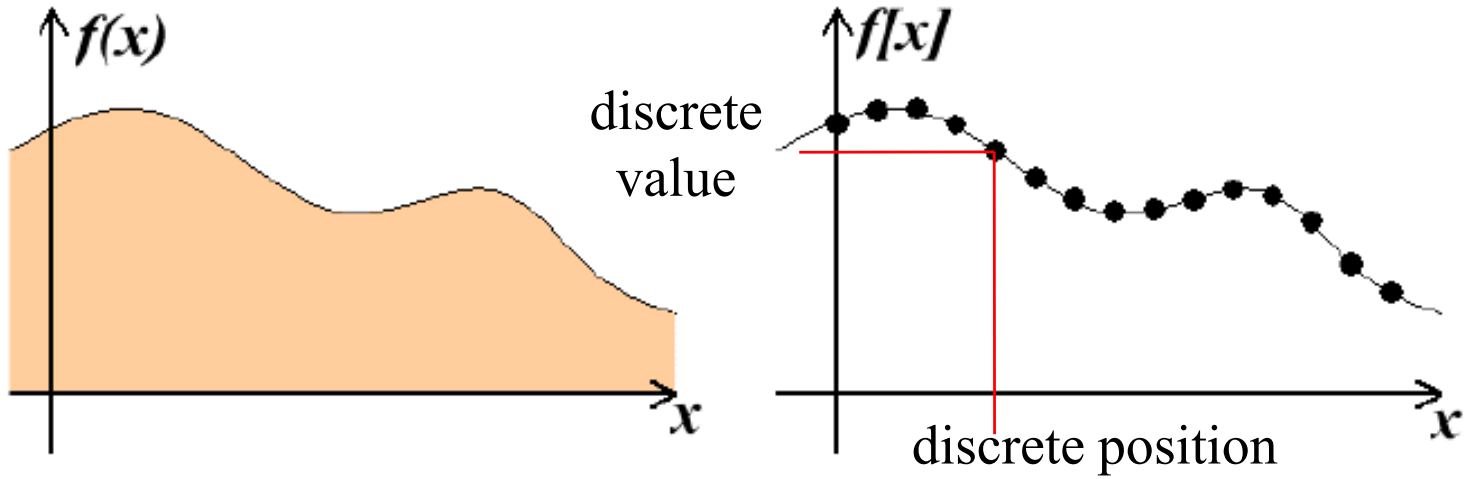




© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

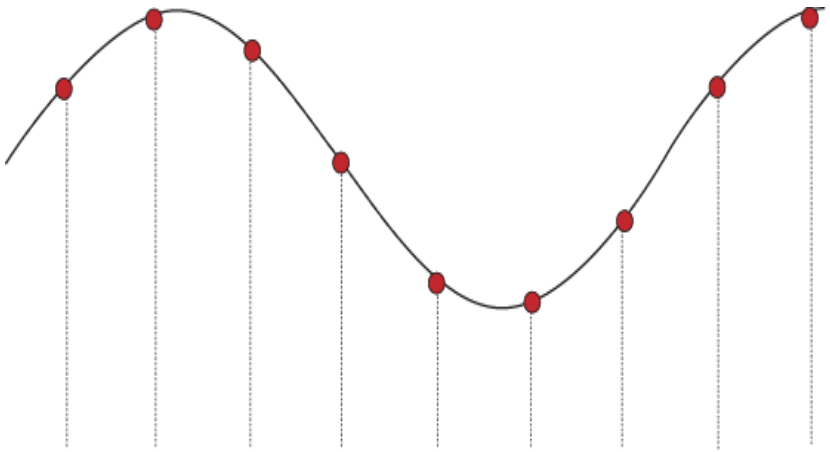
More on Samples

- In signal processing, the process of mapping a continuous function to a discrete one is called *sampling*
- The process of mapping a continuous variable to a discrete one is called *quantization*
  - Gamma helps quantization
- To represent or render an image using a computer, we must both sample and quantize
  - Today we focus on the effects of sampling and how to fight them

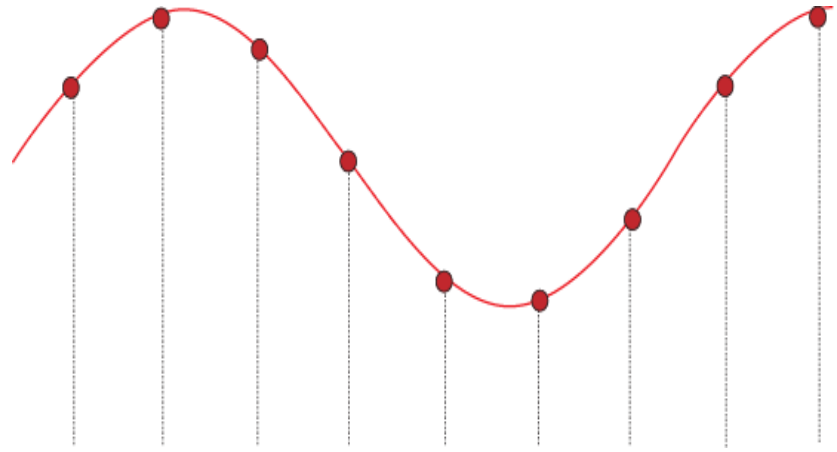


Sampling Density

- If we're lucky, sampling density is enough



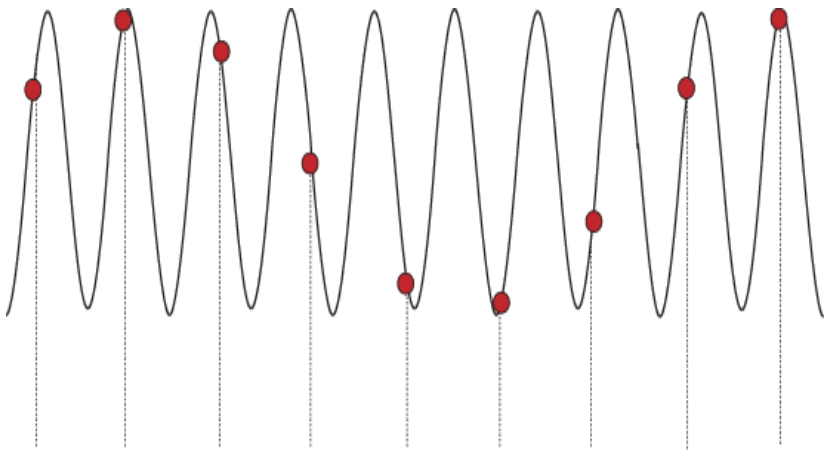
Input

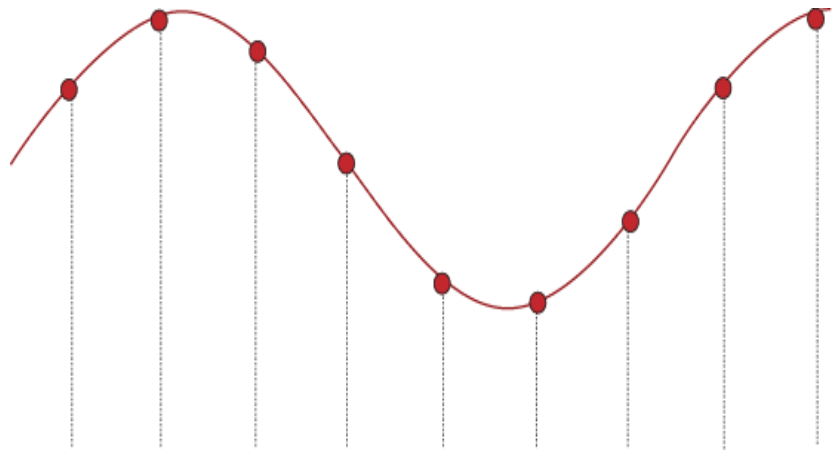


Reconstructed

Sampling Density

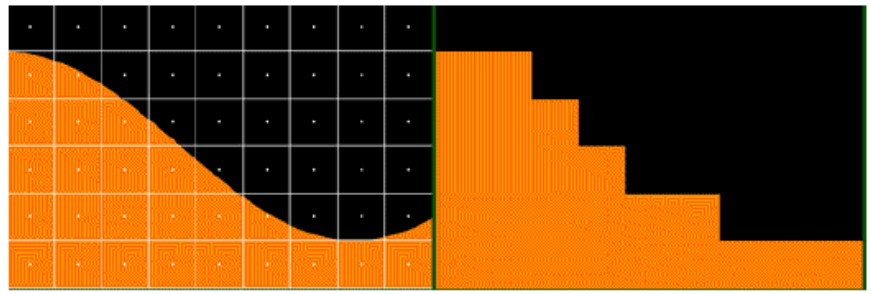
- If we insufficiently sample the signal, it may be mistaken for something simpler during reconstruction (that's aliasing!)
- This is why it’s called aliasing: the new low-frequency sine wave is an alias/ghost of the high-frequency one



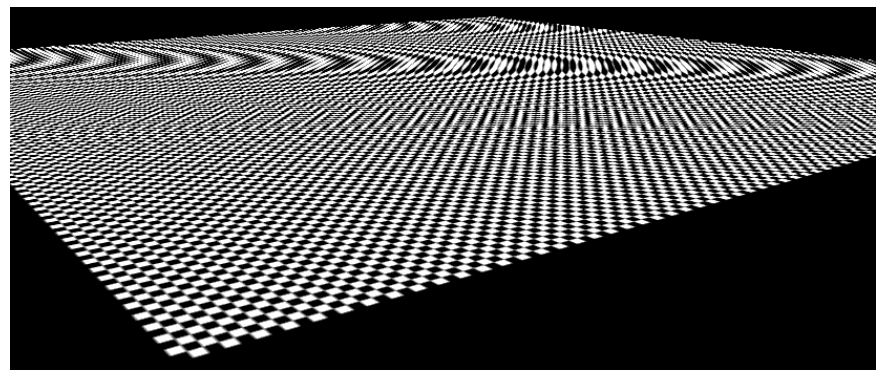


Discussion

- Types of aliasing
  - Edges
    - mostly directional aliasing (vertical and horizontal edges rather than actual slope)
  - Repetitive textures
    - Paradigm of aliasing
    - Harder to solve right
    - Motivates fun mathematics



© Rosalee Nerheim-Wolfe, Toby Howard, Stephen Spencer. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.



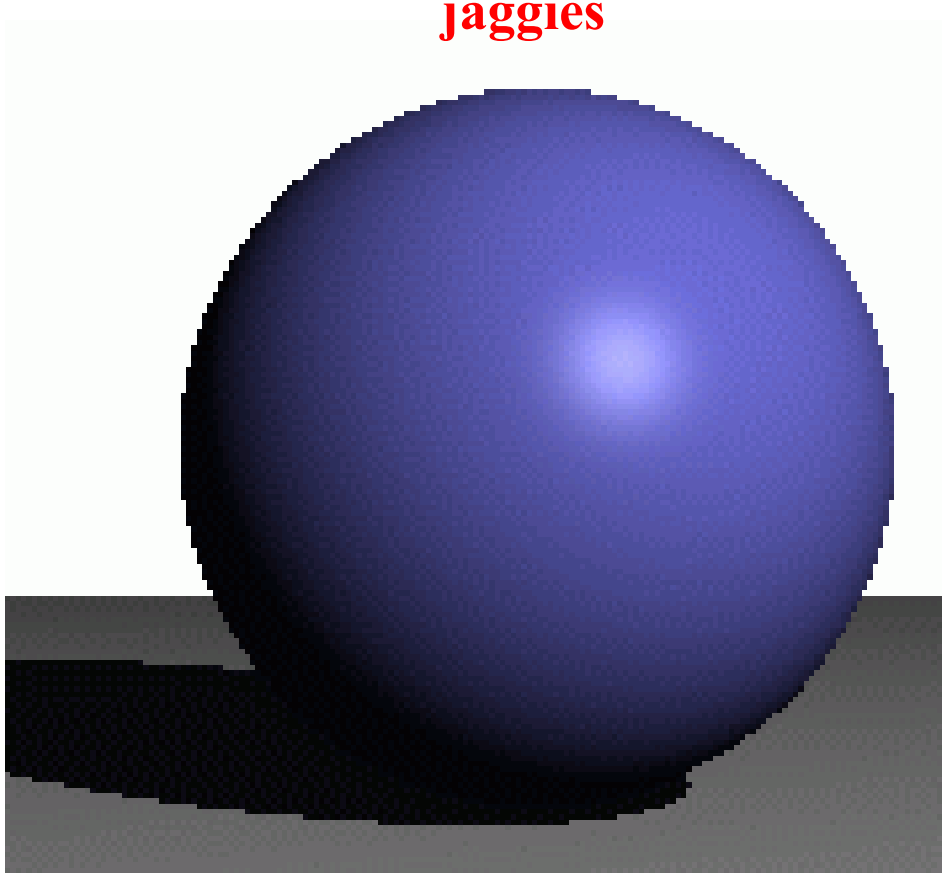
Solution?

- How do we avoid that high-frequency patterns mess up our image?
- We blur!
  - In the case of audio, people first include an analog low-pass filter before sampling
  - For ray tracing/rasterization: compute at higher resolution, blur, resample at lower resolution
  - For textures, we can also blur the texture image before doing the lookup
- To understand what really happens, we need serious math

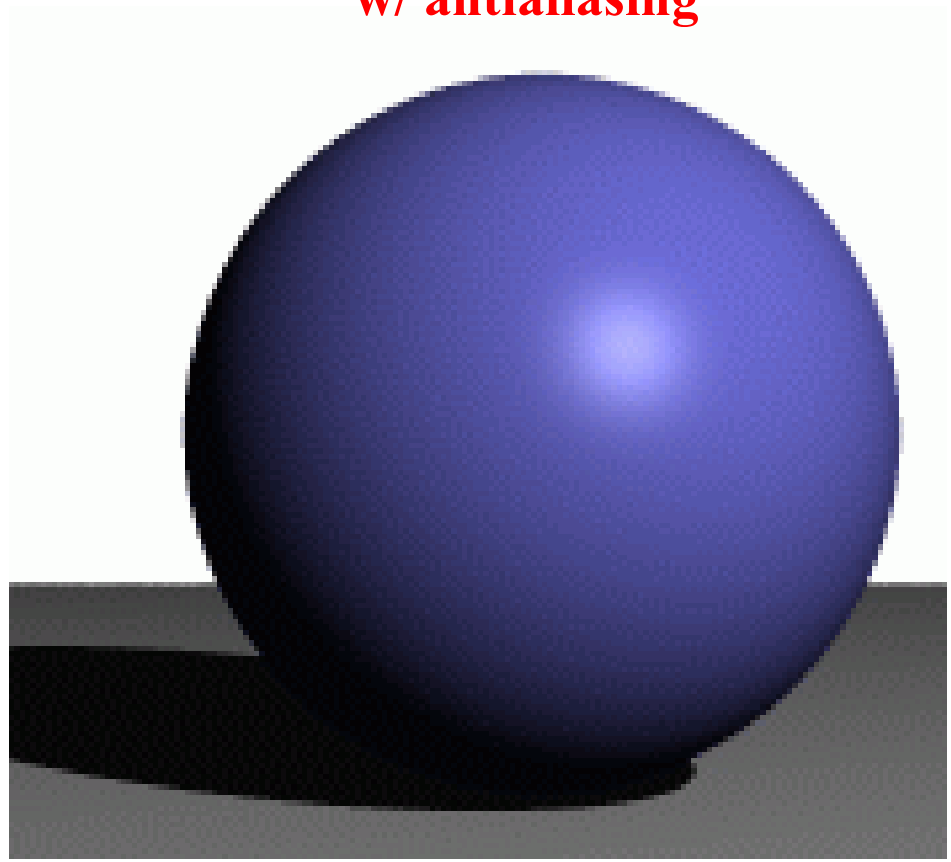
In practice: Supersampling

- Your intuitive solution is to compute multiple color values per pixel and average them

jaggies

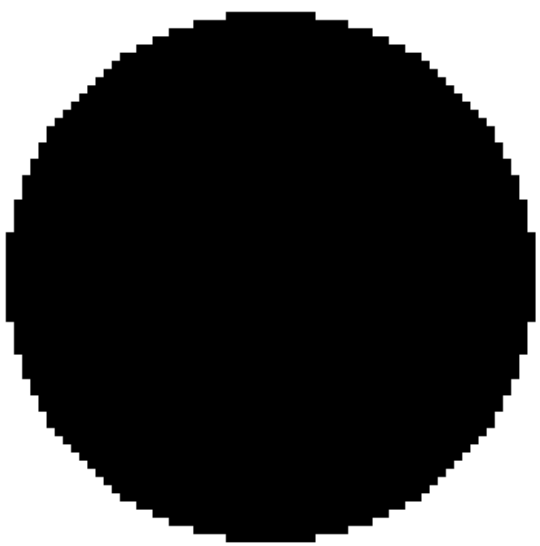


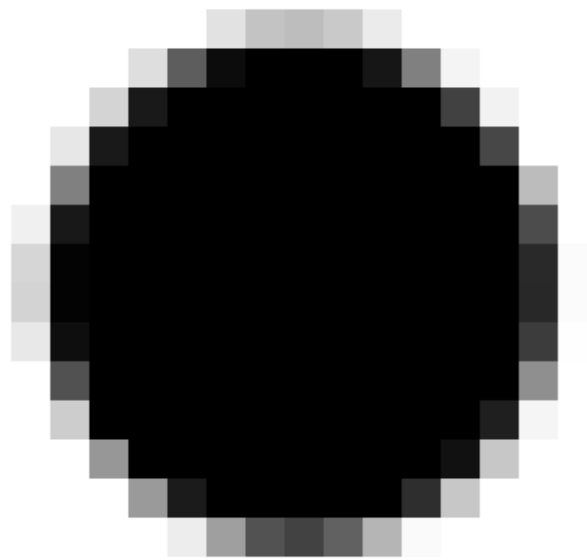
w/ antialiasing



Uniform supersampling

- Compute image at resolution k*width, k*height
- Downsample using low-pass filter
  (e.g. Gaussian, sinc, bicubic)



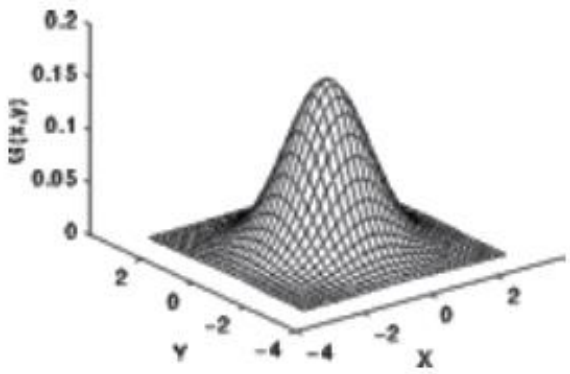


Low pass / convolution

- Each output (low-res) pixel is a weighted average of input subsamples
- Weight depends on relative spatial position
- For example:
  - Gaussian as a function of distance
  - 1 inside a square, zero outside (box)

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1



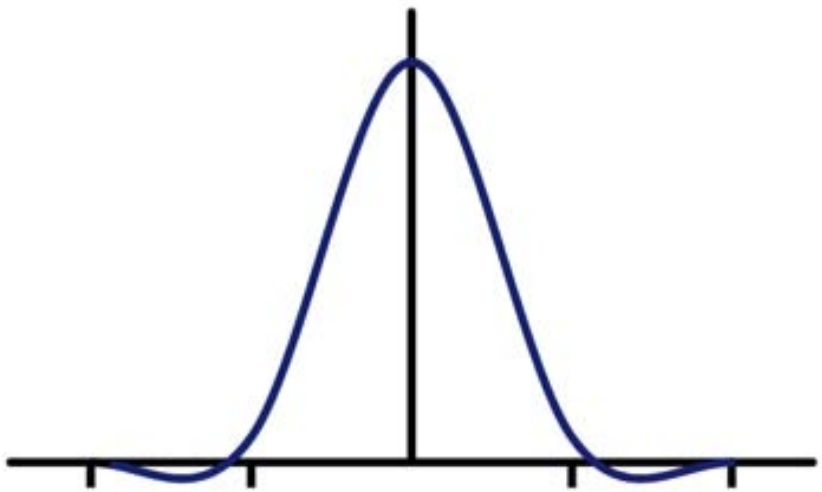
Gaussian

© 2003 R. Fisher, S. Perkins, A. Walker and E. Wolfart. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

Recommended filter

- Bicubic
  - http://www.mentallandscape.com/Papers_siggraph88.pdf
- Good tradeoff between sharpness and aliasing



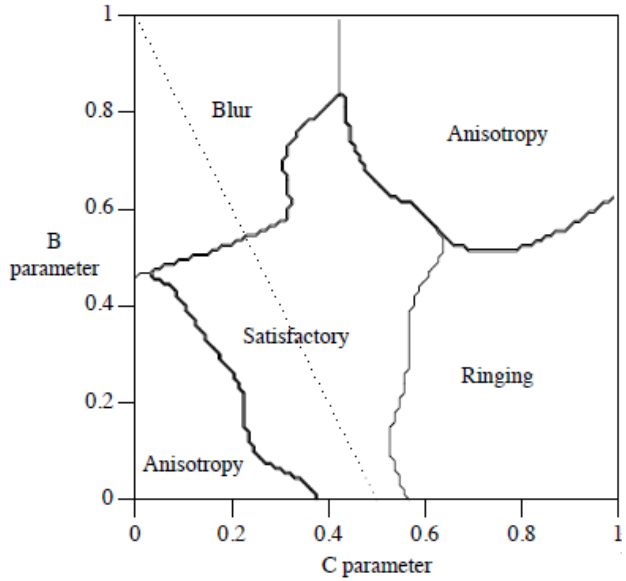
http://de.wikipedia.org/wiki/Datei:Mitchell_Filter.svg

Choosing the parameters

- Empirical tests determined usable parameters
  - Mitchell, Don and Arun Netravali, "Reconstruction Filters in Computer Graphics", SIGGRAPH 88.

    http://www.mentallandscape.com/Papers_siggraph88.pdf

    http://dl.acm.org/citation.cfm?id=378514

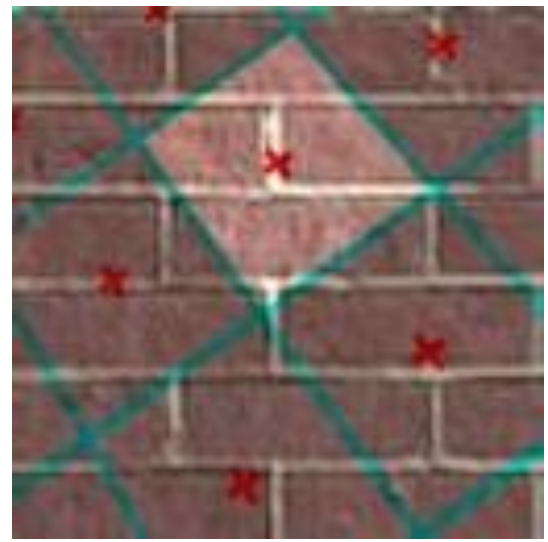


© ACM. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

Spatial Filtering

- Remove the high frequencies which cause artifacts in texture minification.
- Compute a spatial integration over the extent of the pixel
- This is equivalent to convolving the texture with a filter kernel centered at the sample (i.e., pixel center)!
- Expensive to do during rasterization, but an approximation it can be precomputed

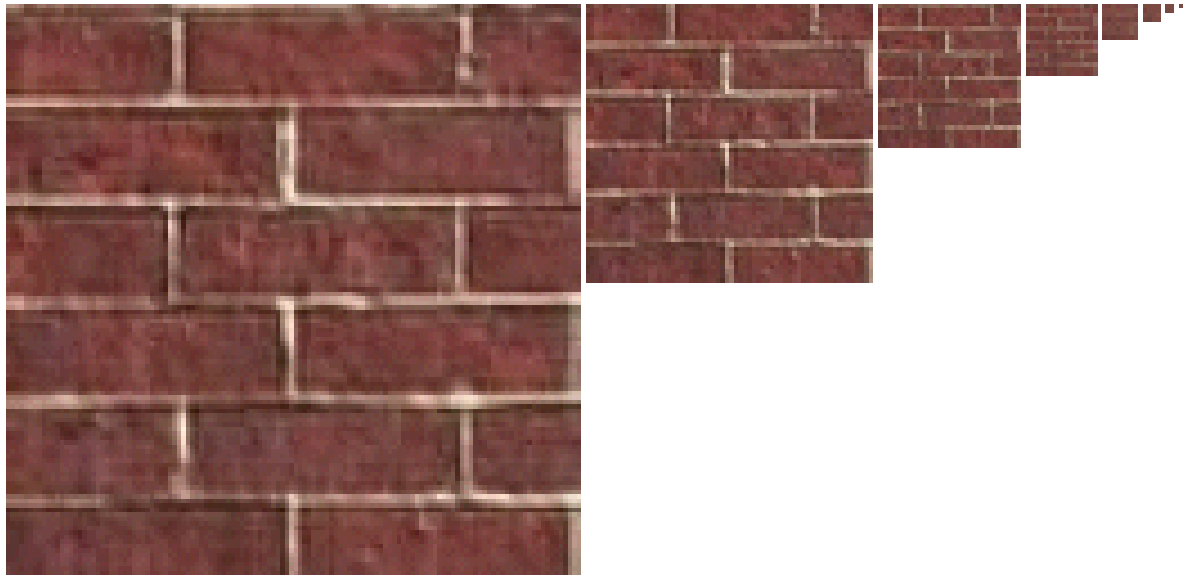
projected texture in image plane



pixels projected in texture plane

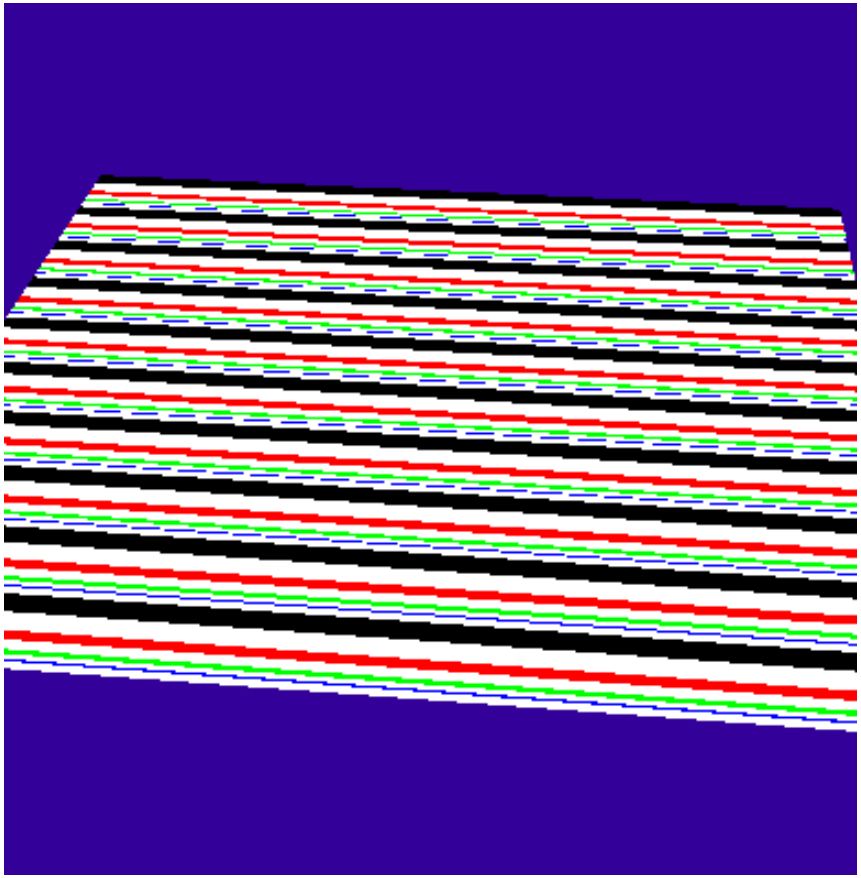
MIP Mapping

- Construct a pyramid of images that are pre-filtered and re-sampled at 1/2, 1/4, 1/8, etc., of the original image's sampling

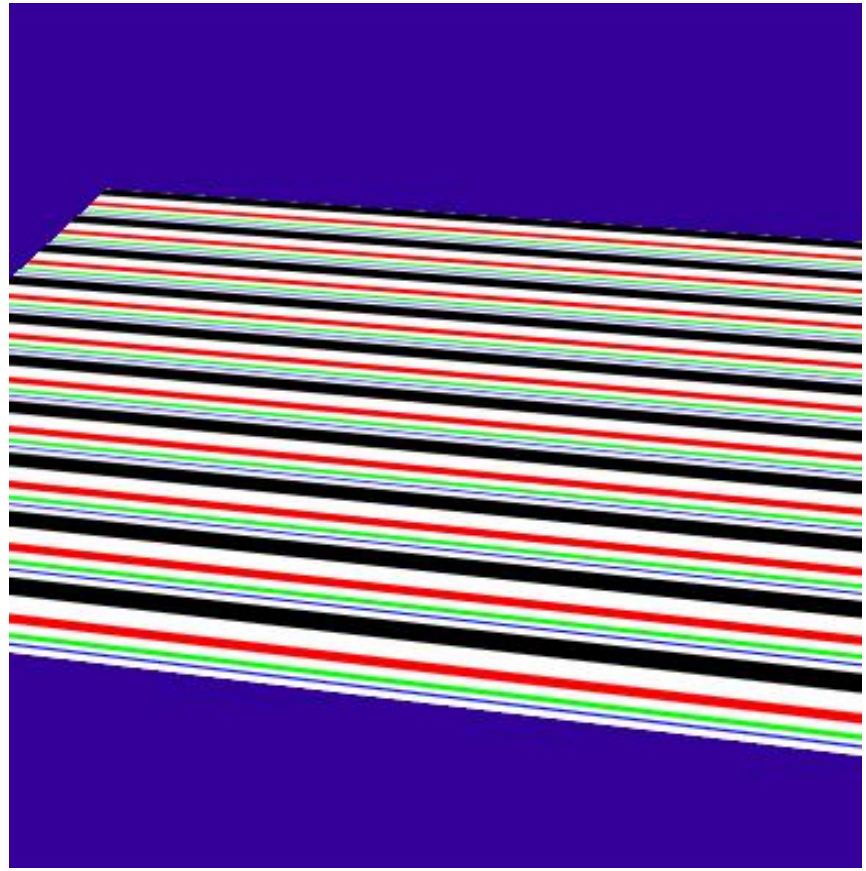


- During rasterization we compute the index of the decimated image that is sampled at a rate closest to the density of our desired sampling rate
- MIP stands for *multum in parvo* which means *many in a small place*

MIP Mapping Example



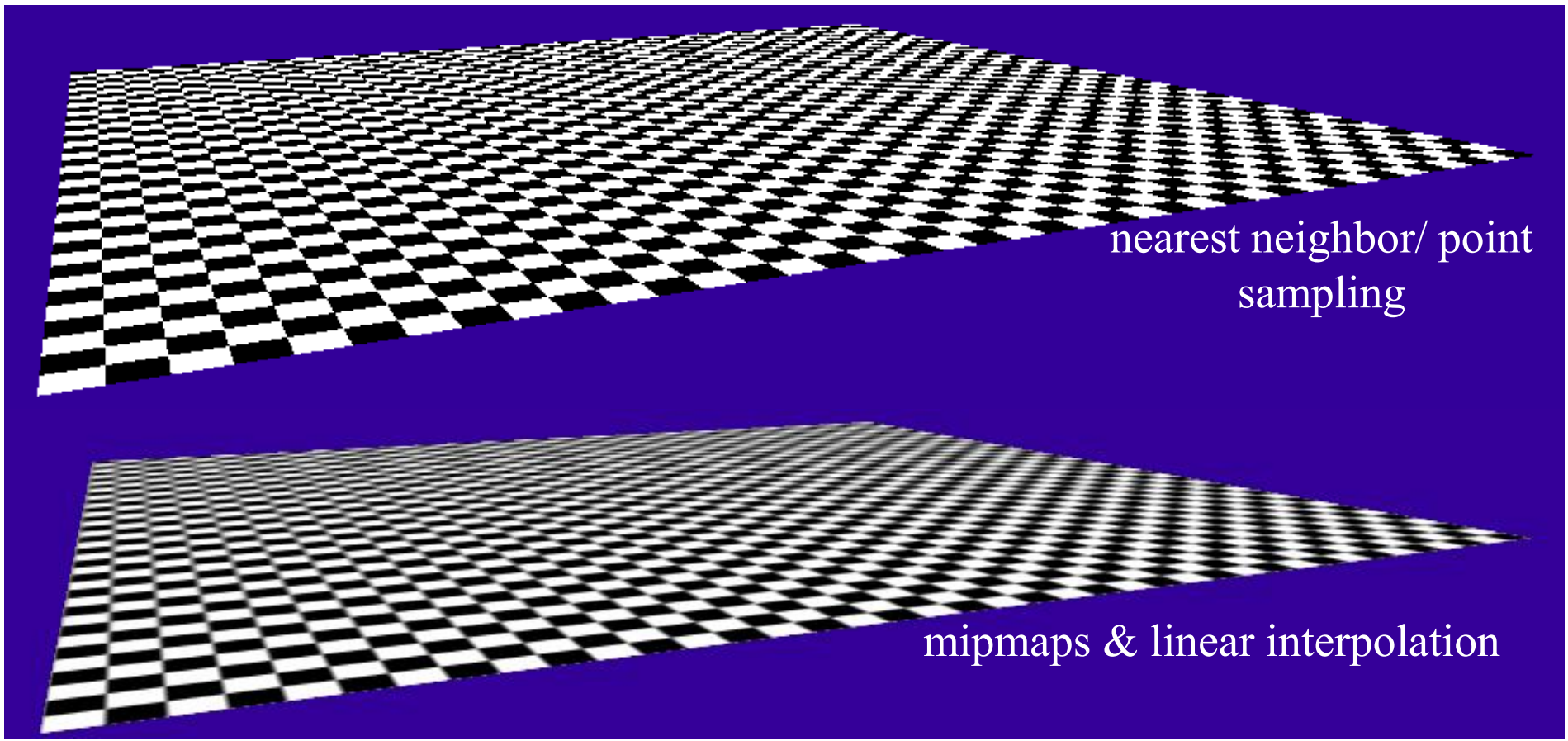
Nearest Neighbor



MIP Mapped (Bi-Linear)

Examples of Aliasing

Texture Errors



TIEA311 - Today in Jyväskylä

- Much more about sampling issues and antialiasing on “Lecture 17” of the original course material.
- The previous few slides were just a low-resolution sample of the original slide set – (pun, intended, funny).
- As mentioned earlier, we gladly defer the theory to our local courses “TIES324 Signaalinkäsittely” and techniques to “TIES471 Reaaliaikainen renderöinti”.

TIEA311 - last slide

Thus, we scratched the surface of something massive.

Where, when, and how to continue:

- Either TIES471 **next week**
- Or TIES471 **next year** (hobby projects and more math in-between?)
- Theses, hobby projects
- The more math you can do, the better – to move from “copy-paster” to “developer” of algorithms!
- Remember that the **tax payers spend a lot of money** every day to pay for **your subscription** to research articles! **Make it count! Learn!** Just browse journals through the university IP, using VPN from home.

THANK YOU!