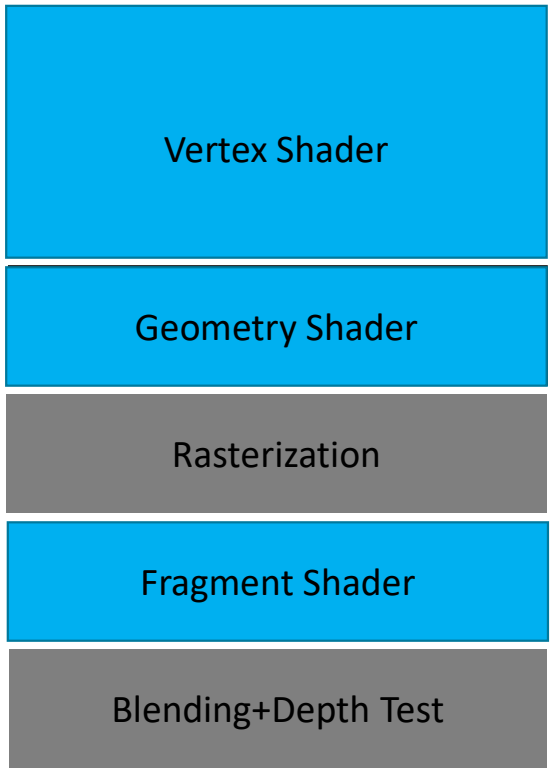
3D Computer Graphics and Animation

Materials & Shading
All that glitters is not gold

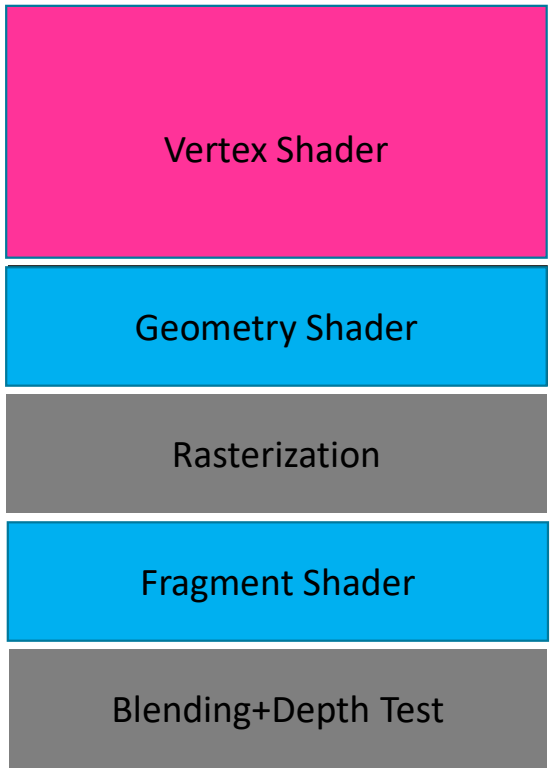
Elmar Eisemann

Delft University of Technology

Today's Standard Graphics Pipeline

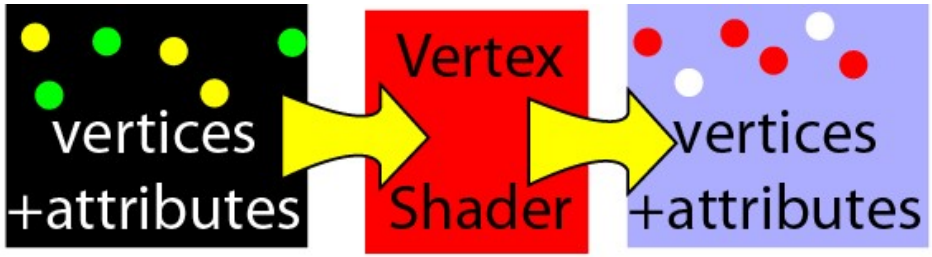


Today's Standard Graphics Pipeline



Vertex Shader

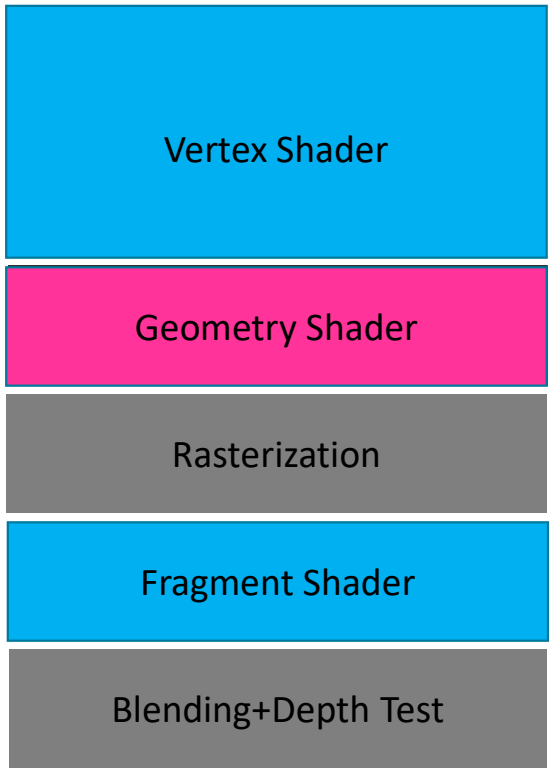
- Input: Vertex with attributes
- Output: Vertex with attributes



- What operation is done in standard pipeline?

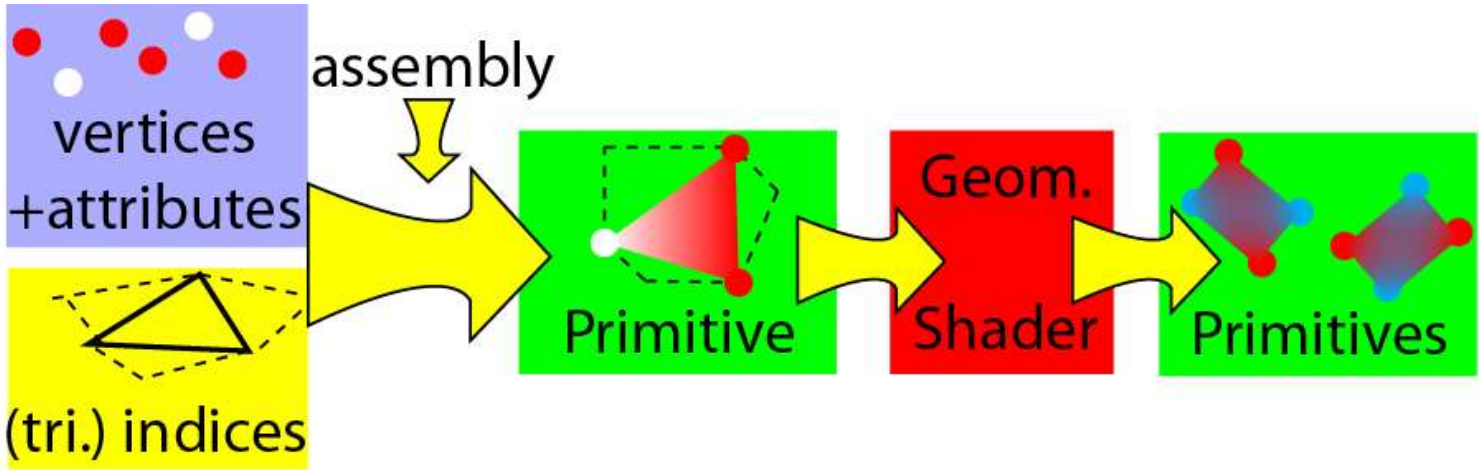
  Usually camera projection / object placement!

Today's Standard Graphics Pipeline

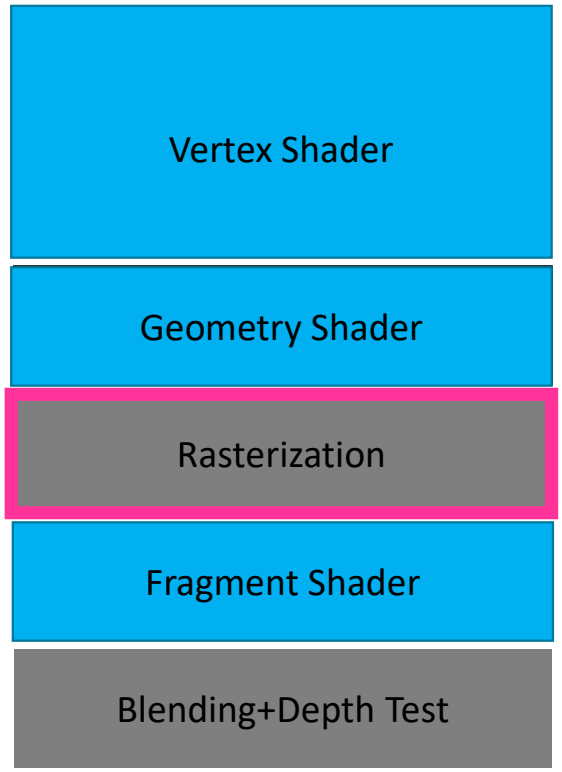


Geometry Shader

- Input: Vertex/Attribute **array** of current primitive (e.g., triangle or its immediate neighborhood)
- Output: Several primitives (vertices + attributes)

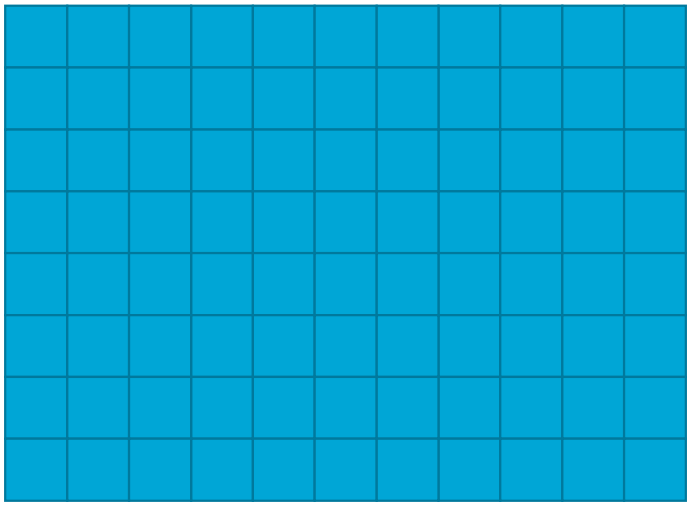


Today's Standard Graphics Pipeline



Rasterization

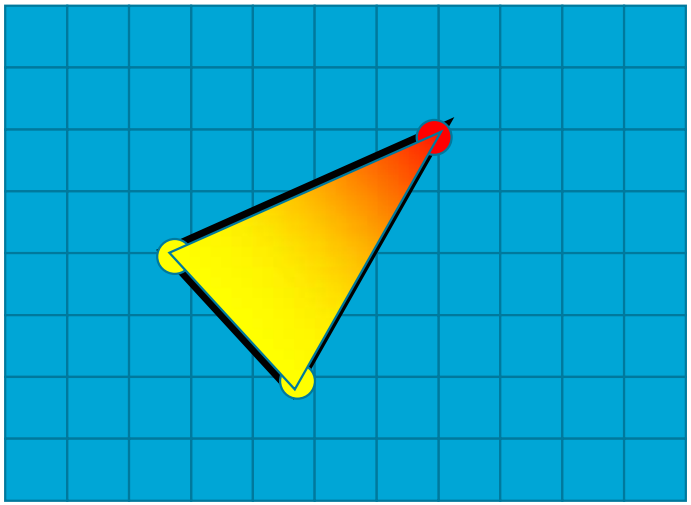
- Triangles into fragments (pixels+attributes)



- Fragment data determined by interpolation
- Values are extracted at pixel centers

Rasterization

- Triangles into fragments (pixels+attributes)



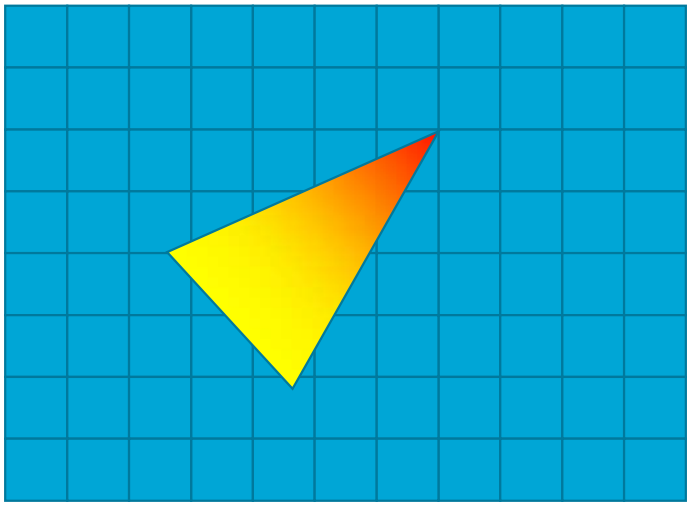
Two yellow,
one red vertex

Colors are
interpolated over
triangle

- Fragment data determined by interpolation
- Values are extracted at pixel centers

Rasterization

- Triangles into fragments (pixels+attributes)



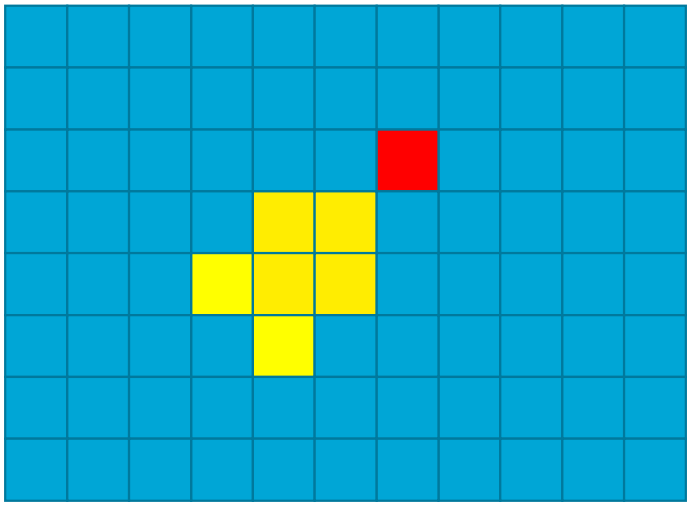
Two yellow,
one red vertex

Colors are
interpolated over
triangle

- Fragment data determined by interpolation
- Values are extracted at pixel centers

Rasterization

- Triangles into fragments (pixels+attributes)

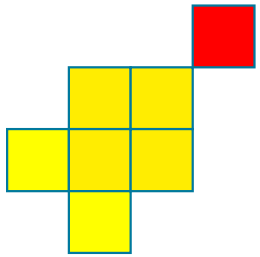


Colors sampled
at pixel centers

- Fragment data determined by interpolation
- Values are extracted at pixel centers

Rasterization

- Triangles into fragments (pixels+attributes)



The input to the fragment shader

- Fragment data determined by interpolation
- Values are extracted at pixel centers

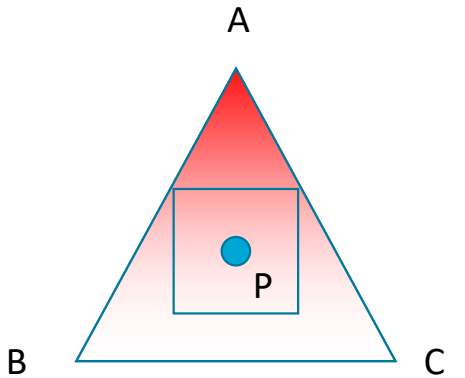
Example: Interpolation

- What are “interpolated” vertex attributes in each Fragment?

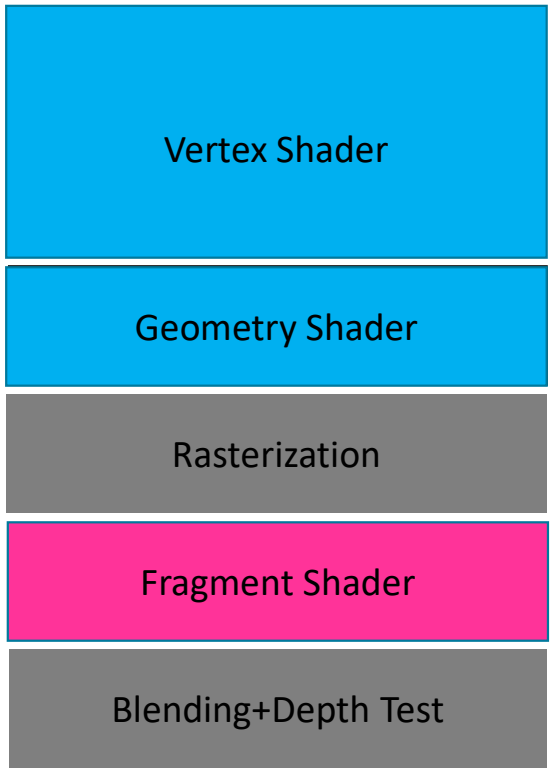
- Imagine P=1/3 A + 1/3 B + 1/3 C
- If the colors at A, B, C are red (1,0,0), white (1,1,1), white (1,1,1) respectively

- The interpolated color at P would be:

  (1, 2/3, 2/3), which is pink

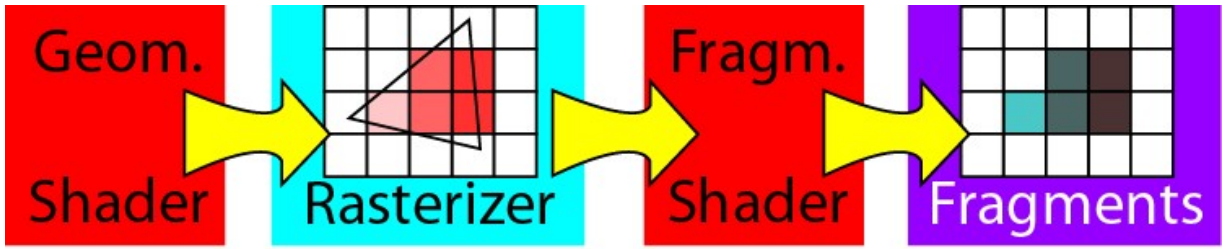


Today's Standard Graphics Pipeline

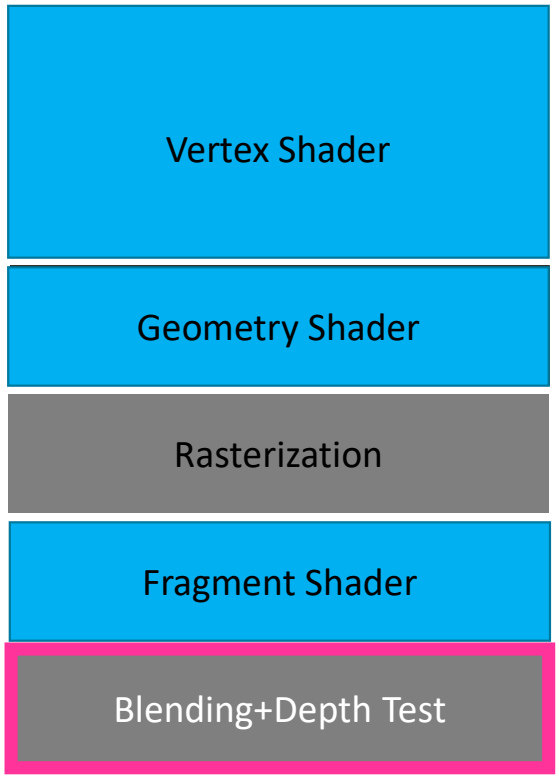


Fragment Shader

- Input: “Pixel” with interpolated vertex/attributes
- Output: New fragment for this position
  - Typically a new pixel color is computed

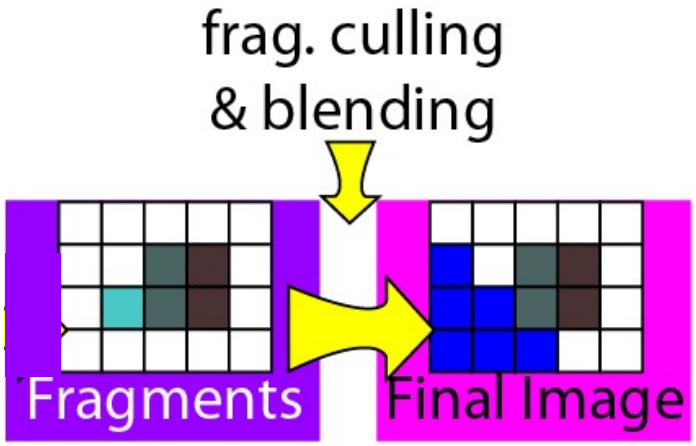


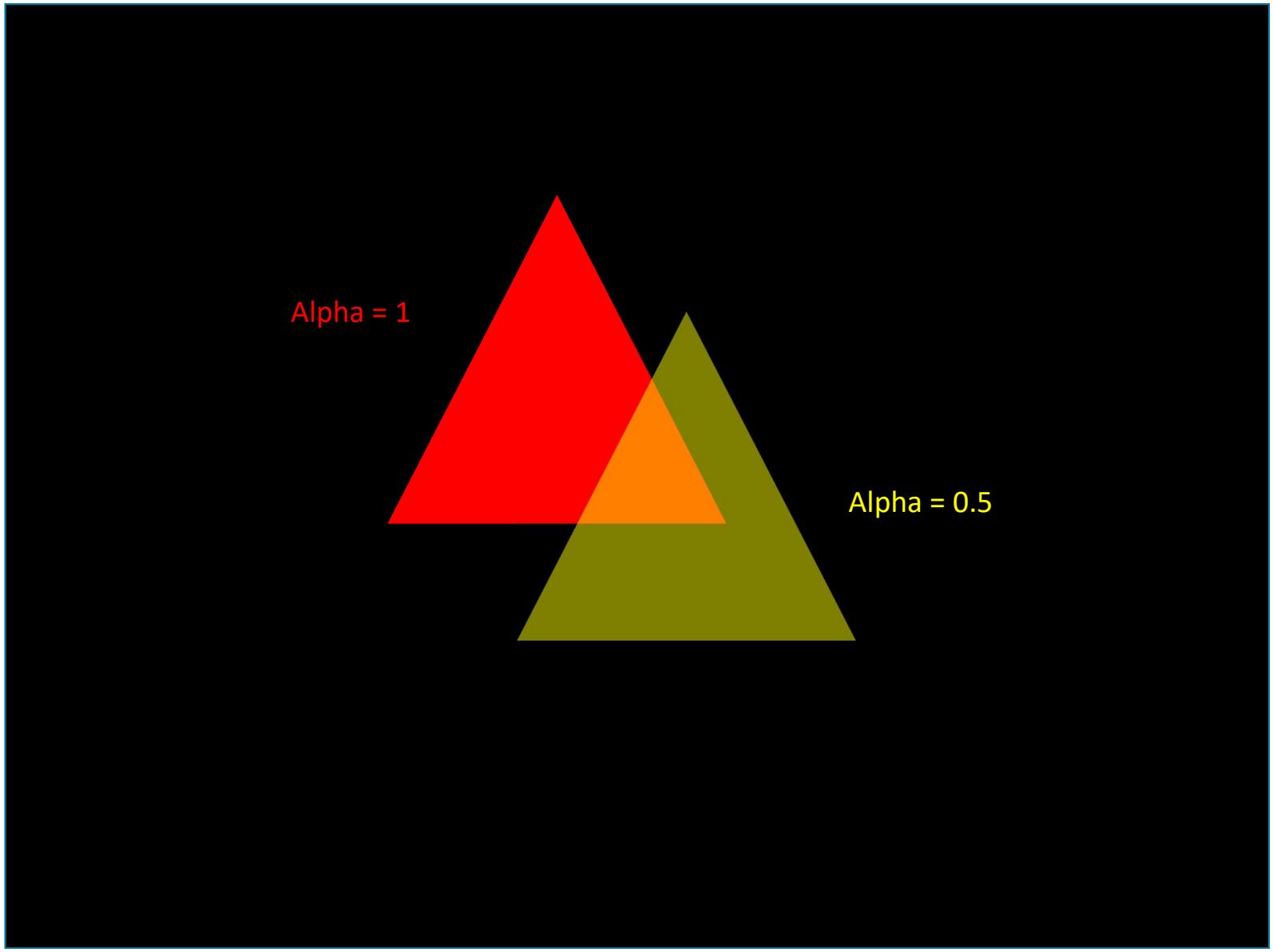
Today's Standard Graphics Pipeline



Blending Stage

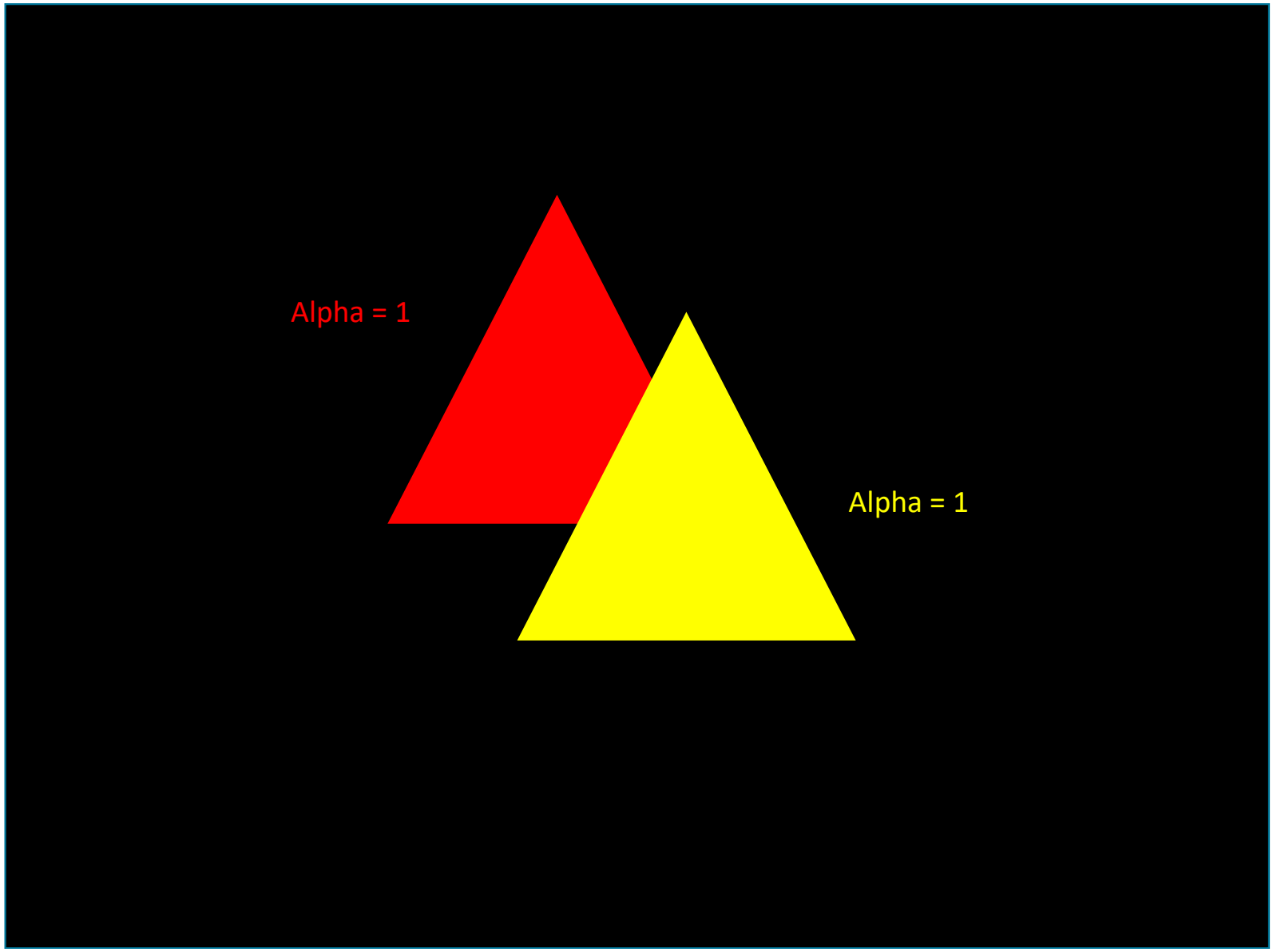
- Blending stage cannot be programmed.
- It takes the produced fragment and combines it with the current image (e.g., depth test, which culls hidden fragments)

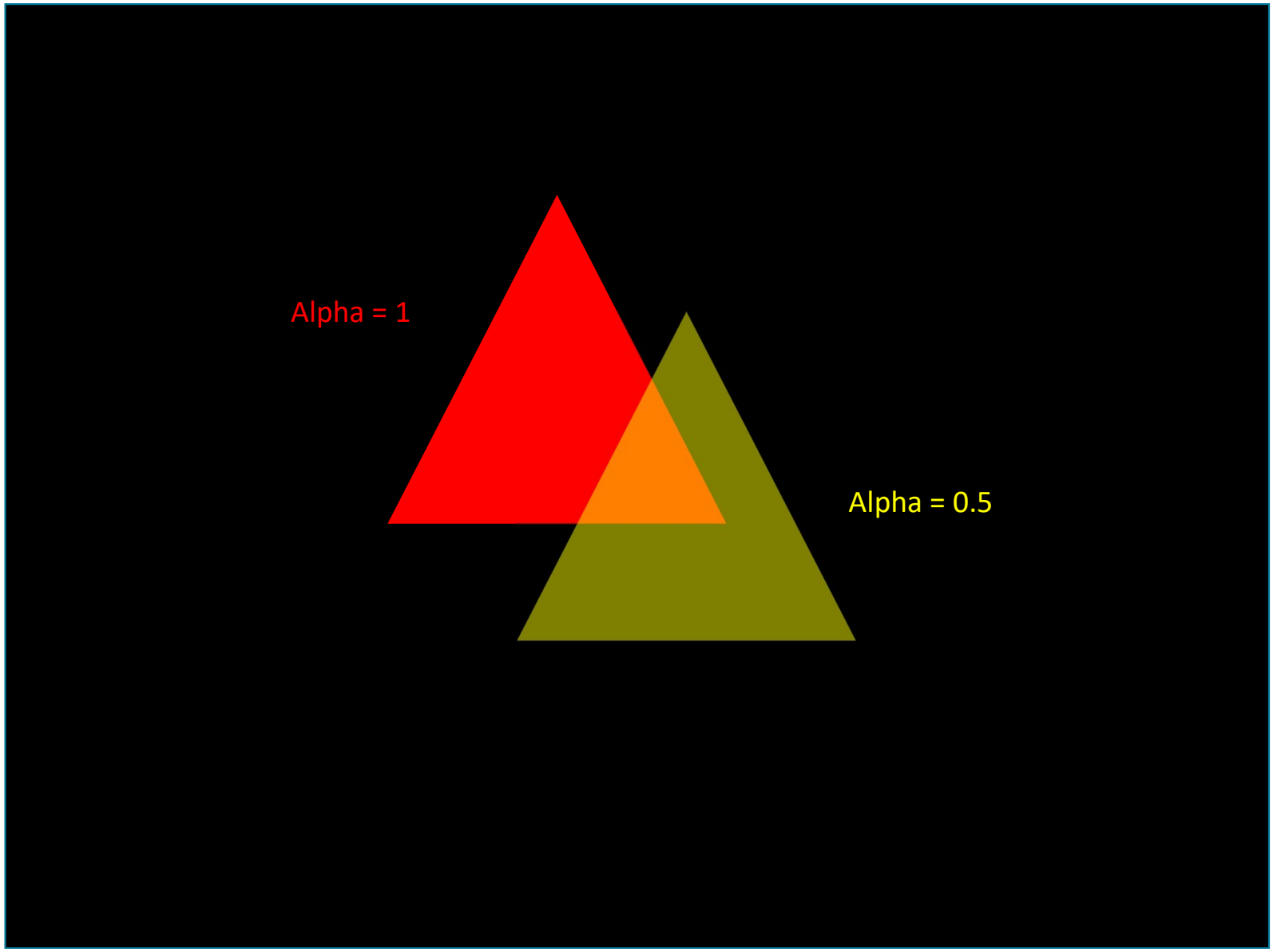


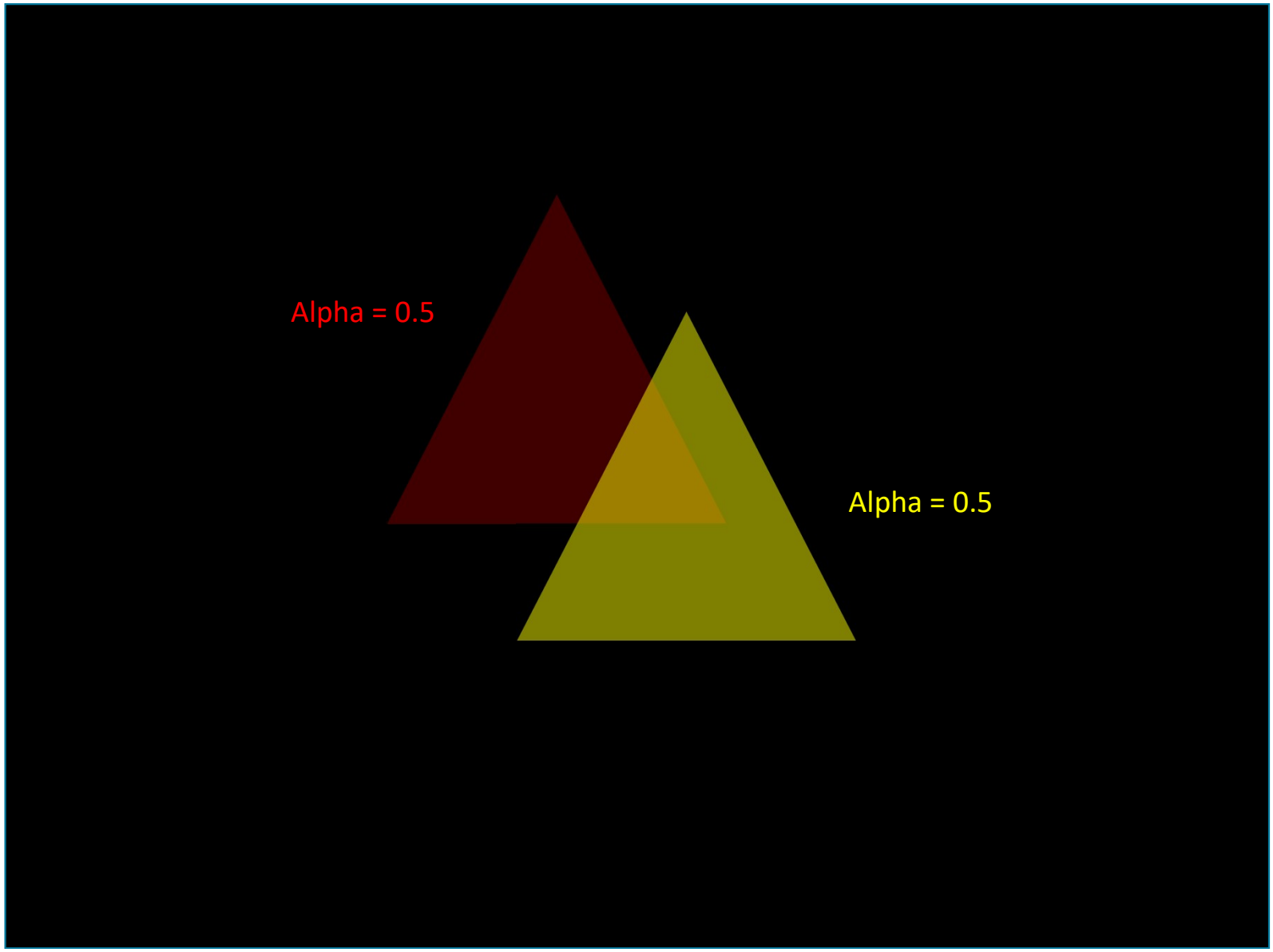

Alpha = 1
Alpha = 0.5

Standard Blending

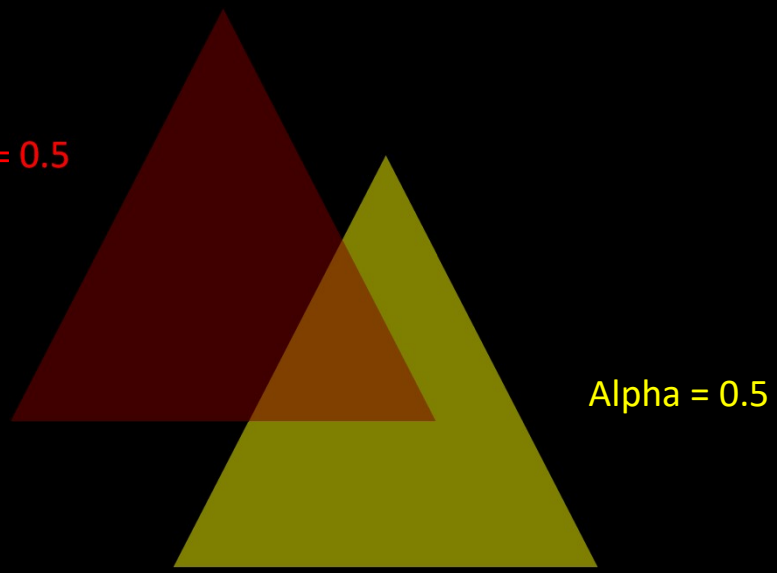
- Color can be a vec4 : RGB and an alpha value
- Alpha describes the opacity
  - 0 : object is completely transparent
  - 1 : object is completely opaque
  - 0.5: 50% of fragment and 50% background
- In general, incoming fragment (R,G,B,A) and current pixel color is ($R_B G_B B_B$) then
  the new pixel color is: A * RGB+ (1-A) * $R_B G_B B_B$


Alpha = 1
Alpha = 1


Alpha = 1
Alpha = 0.5

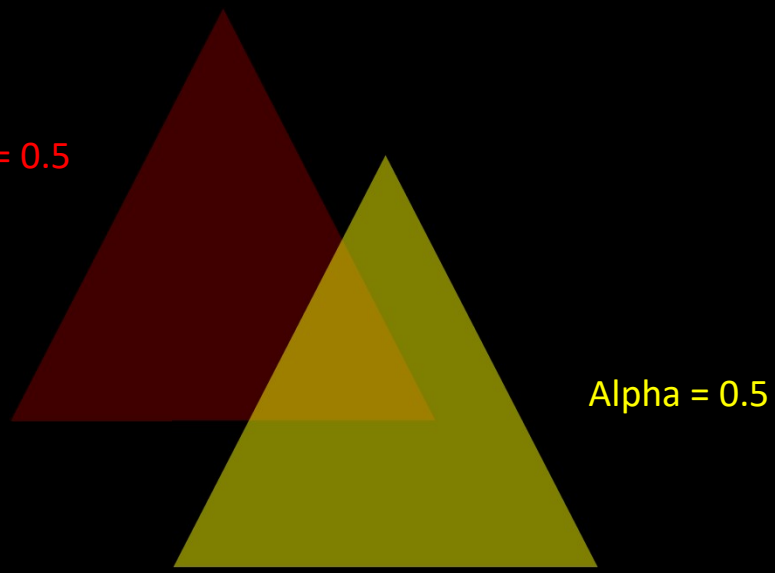

Alpha = 0.5
Alpha = 0.5

Attention: ORDER is important



RED drawn last

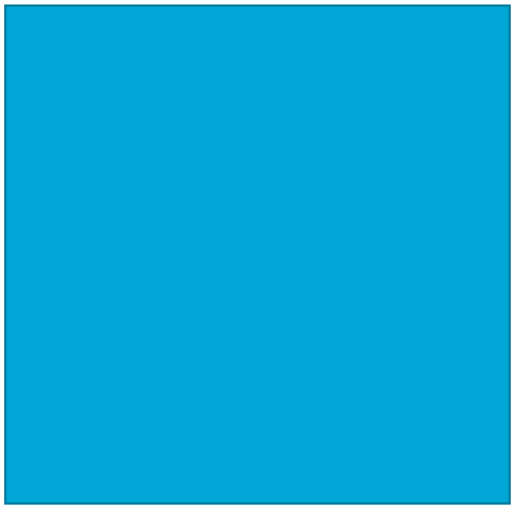
Attention: ORDER is important



YELLOW drawn last

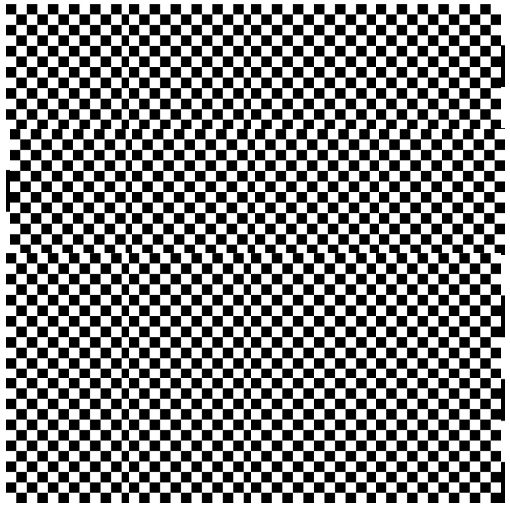
Why this formula?

- Alpha is the opacity of the object
- Imagine the object to have tiny holes with probability Alpha
- 0.5 could be imagined as a checkerboard



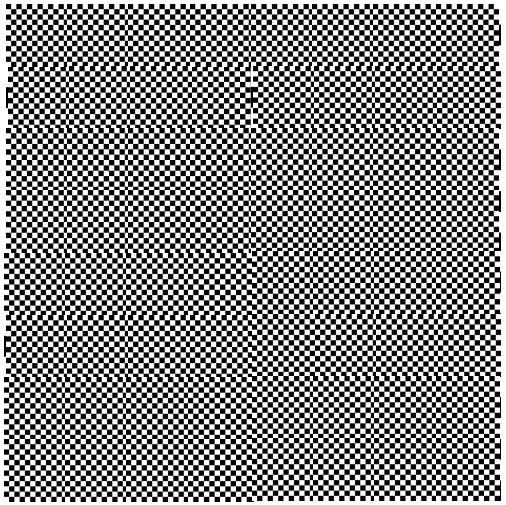
Why this formula?

- Alpha is the opacity of the object
- Imagine the object to have tiny holes with probability Alpha
- 0.5 could be imagined as a checkerboard



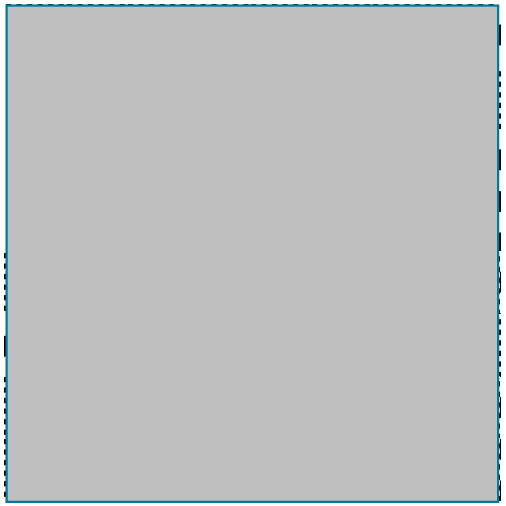
Why this formula?

- Alpha is the opacity of the object
- Imagine the object to have tiny holes with probability Alpha
- 0.5 could be imagined as a checkerboard



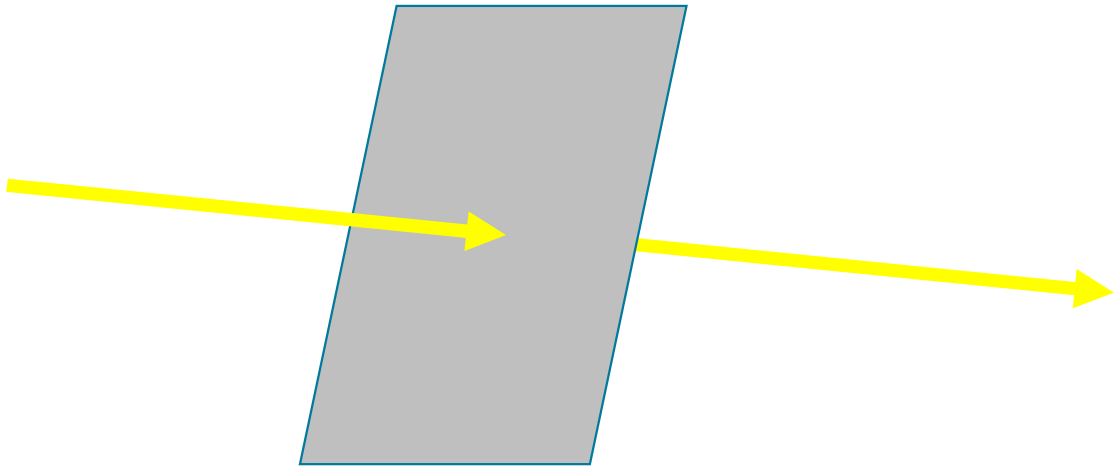
Why this formula?

- Alpha is the opacity of the object
- Imagine the object to have tiny holes with probability Alpha
- 0.5 could be imagined as a checkerboard



Why this formula?

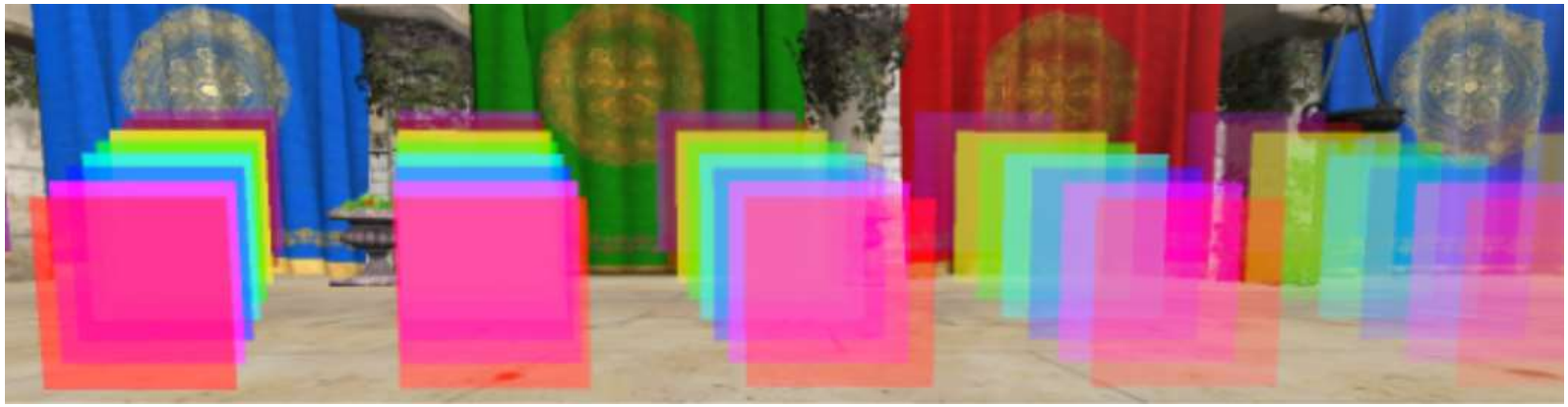
- A light ray hits with probability A
  and passes with probability (1-A),
  hence, A * color + (1-A) * Background



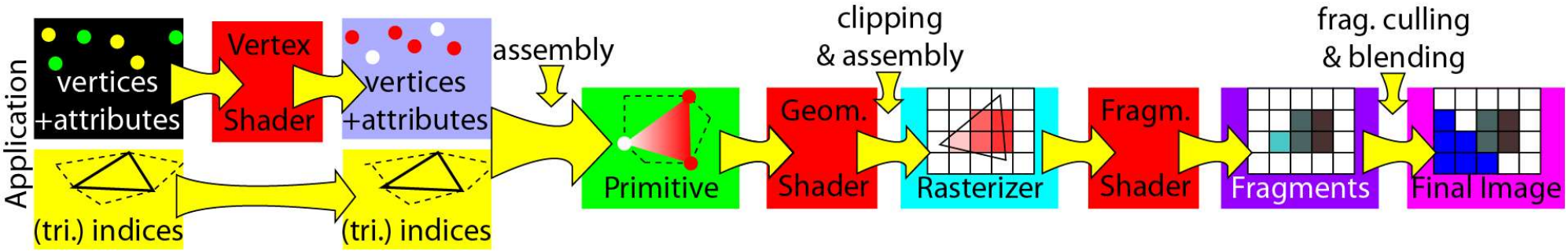
Attention!

- If you have several transparent objects,
  it might be needed to sort the triangles.

- Be aware that when drawing transparent objects you should draw them last and deactivate the depth buffer



Programmable Graphics Pipeline



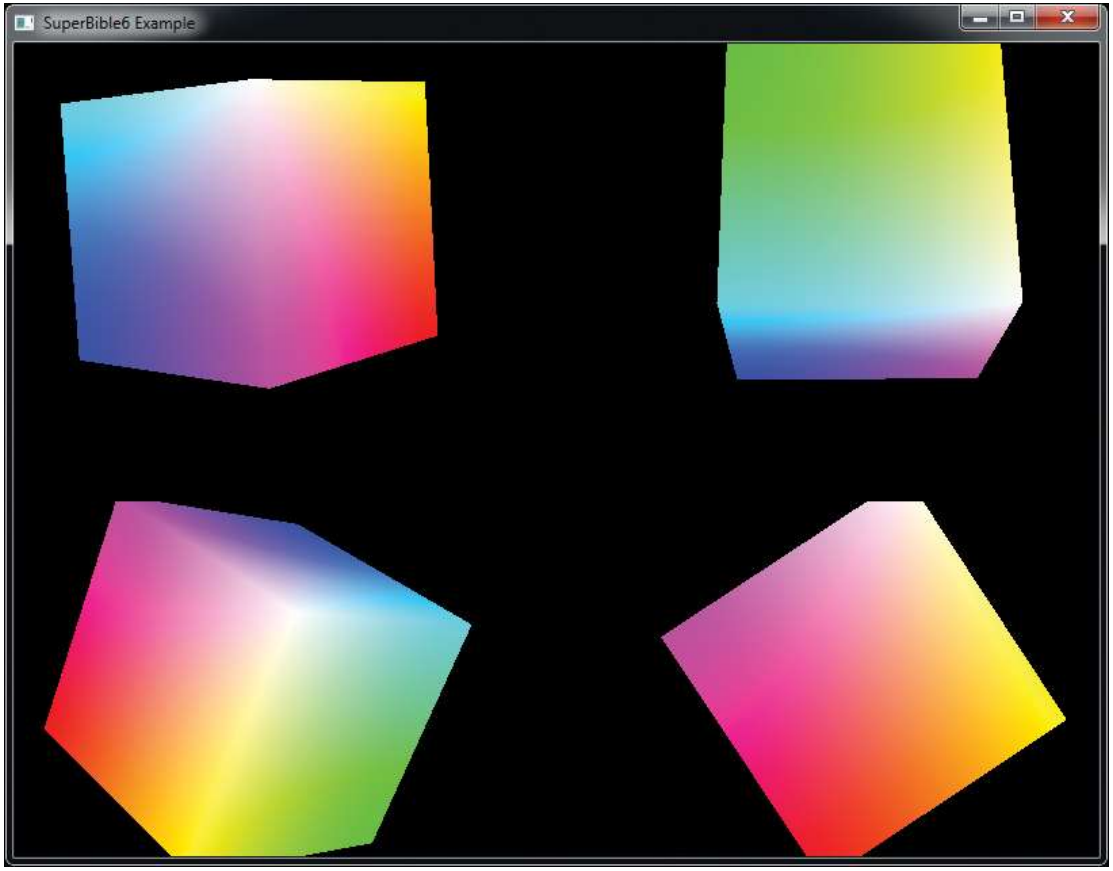
More Modern Extensions

Modern GPUs support even more
(but out of scope of this lecture)

- Tessellation Shader: Subdivide surfaces
- Compute Shader: For general purpose computing
- RTX: Raytracing extensions, e.g., via Optix

We saw the full pipeline but it produces these images...

A first step towards beauty…



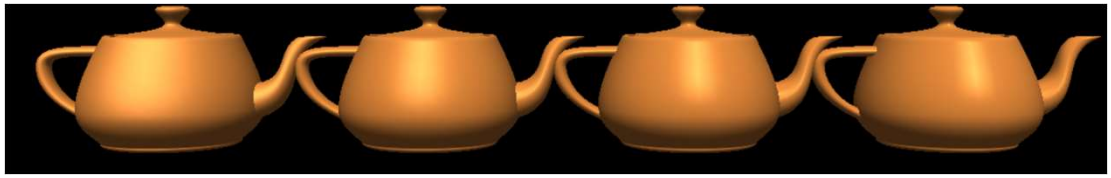
OpenGL SuperBible: Comprehensive Tutorial and Reference, 6th Edition

This whole pipeline and still no beautiful pictures?

Shading

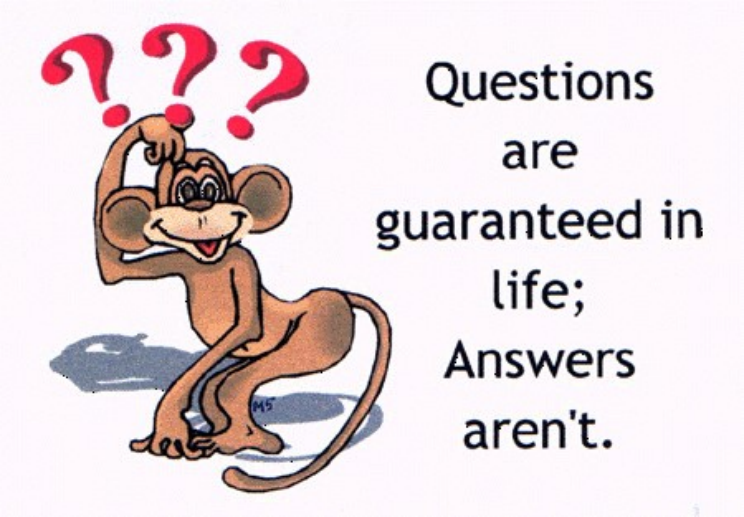
How to transform

in



?

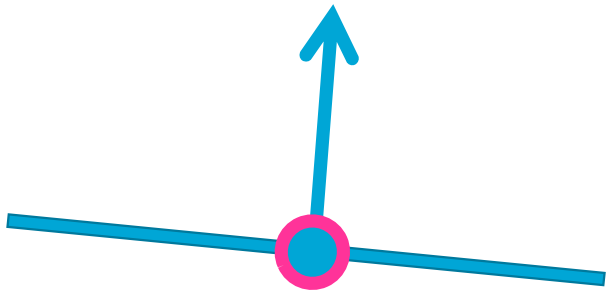
Questions?



Making beautiful pictures…

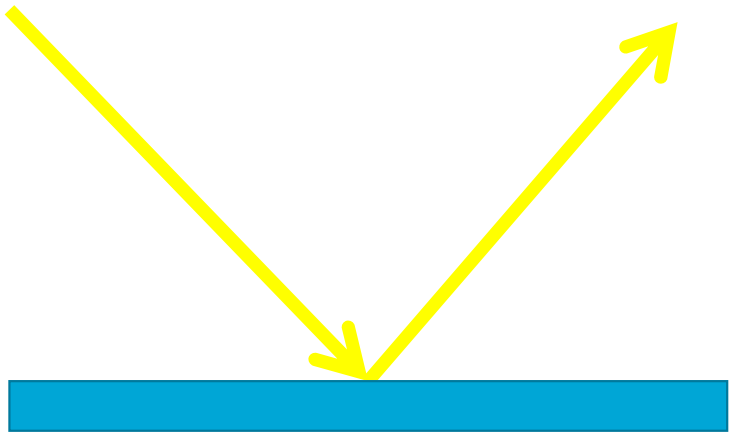
TODAY'S QUESTION:

- Given a surface point
  (position, its normal, and potential attributes)
  and a point emitting light…
- How to compute a realistic color???



Reflection

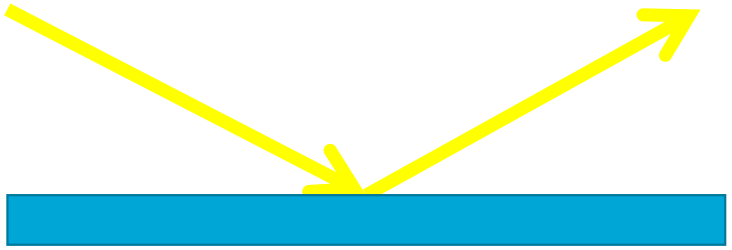
- What happens when the light hits a surface?



Mirror

Reflection

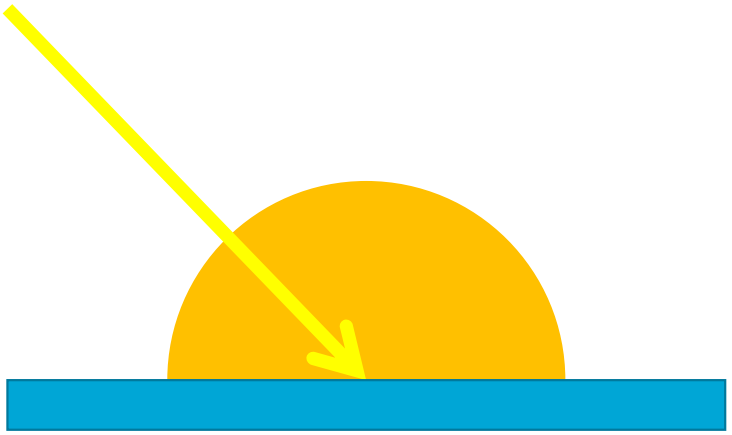
- What happens when the light hits a surface?



Mirror

Reflection

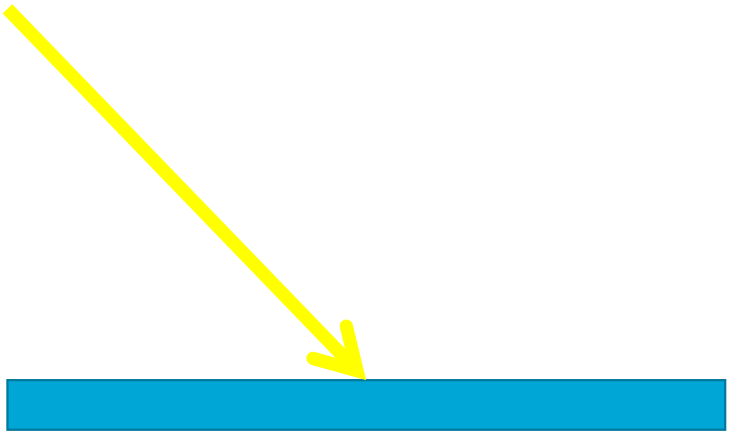
- What happens when the light hits a surface?



perfectly diffuse

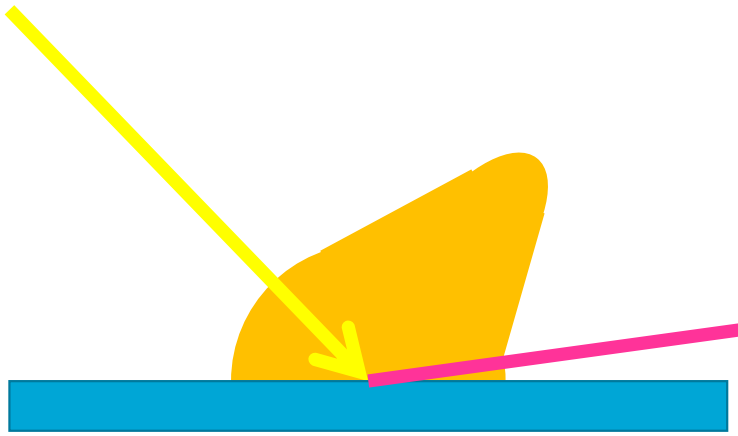
Reflection

- What happens when the light hits a surface?

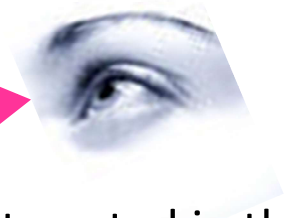


Reflection

- What happens when the light hits a surface?



general

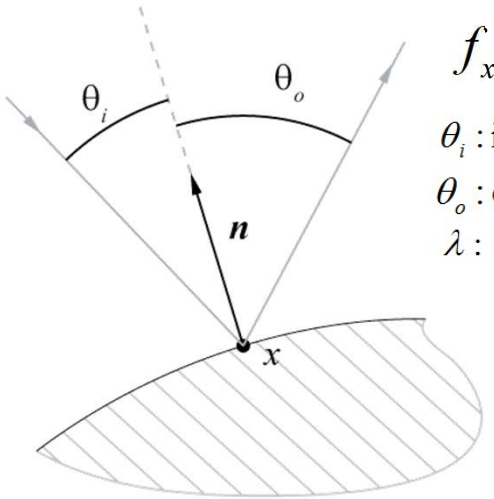


Today:
Mostly interested in the direction towards the camera!

How are materials represented?

BRDF *(Bidirectional Reflectance Distribution Function)*

Function from IR^5 into IR indicating how much incident light is reflected in a point (ratio between in- and outgoing energy)

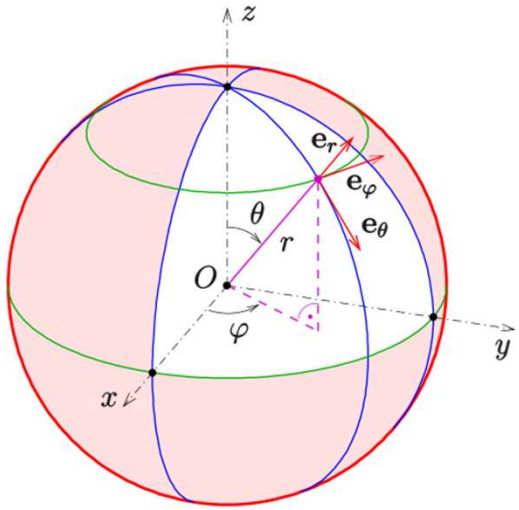


$$f_x(\theta_i, \theta_o, \lambda)$$

θ_i : incoming light

θ_o : outgoing light

λ : wavelength



Wikipedia – a direction is 2D

Jensen, H. W et al. *A practical model for subsurface light transport*. 2001

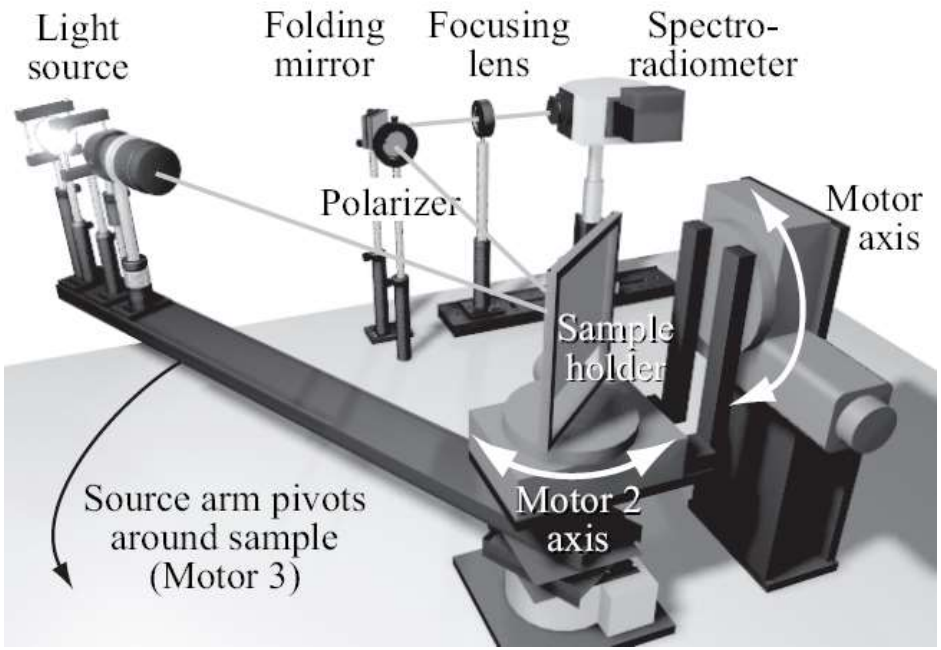
TU Delft

Acquired Materials

- Big databases

Material Acquisition

- Use a gonioreflectometer – yes, that is the name…



http://www.graphics.cornell.edu/~westin/

Acquired Materials

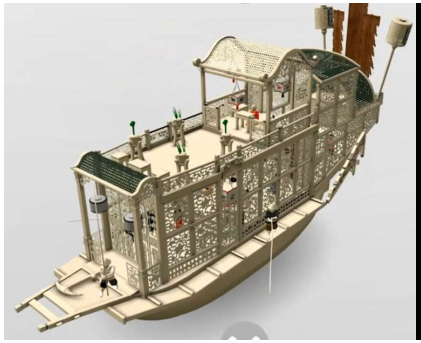
- Big databases

Often costly, much data

Hard to use for artists

e.g., “Can you make the blue darker?”

Very important when
acquiring real-world objects



Mathematical Models

- Describe light interaction as a function
- Usually more lightweight
- Has parameters to control appearance
- Acquired materials can be approximated

Published recently:
Effect of the additional glaze layer (left) that Vermeer placed over the black background.
graphics.tudelft.nl/WebPearl/

Girl With A Pearl Earring

Exhibition in the Mauritshuis
until 8th of January 24 (you missed it)

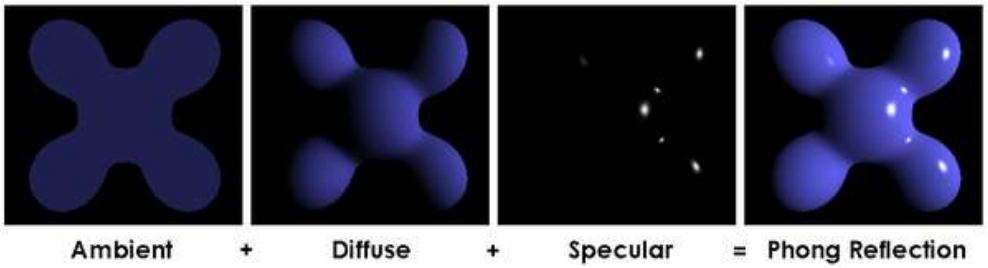
World's largest 3D print

vlnr: Elmar Eisemann (TUD),Ruben Wiersma (TUD), Lucie Oude Luttikhuis (TUD), Liselore Tissen (TUD/UL) Mane van Veldhuizen (UVA), Abbie Vandivere (MH), Mathijs Lefferts (TUD), Emilien Leonhardt (Hirox), Clemens Weijkamp (Canon), Camiel Rejhons (Canon)

Simplified Models

- Mathematically describe Material Properties

- Phong Model: Sum of 3 terms
  - Ambient
  - Diffuse
  - Specular



Color - Recap

- Remember:

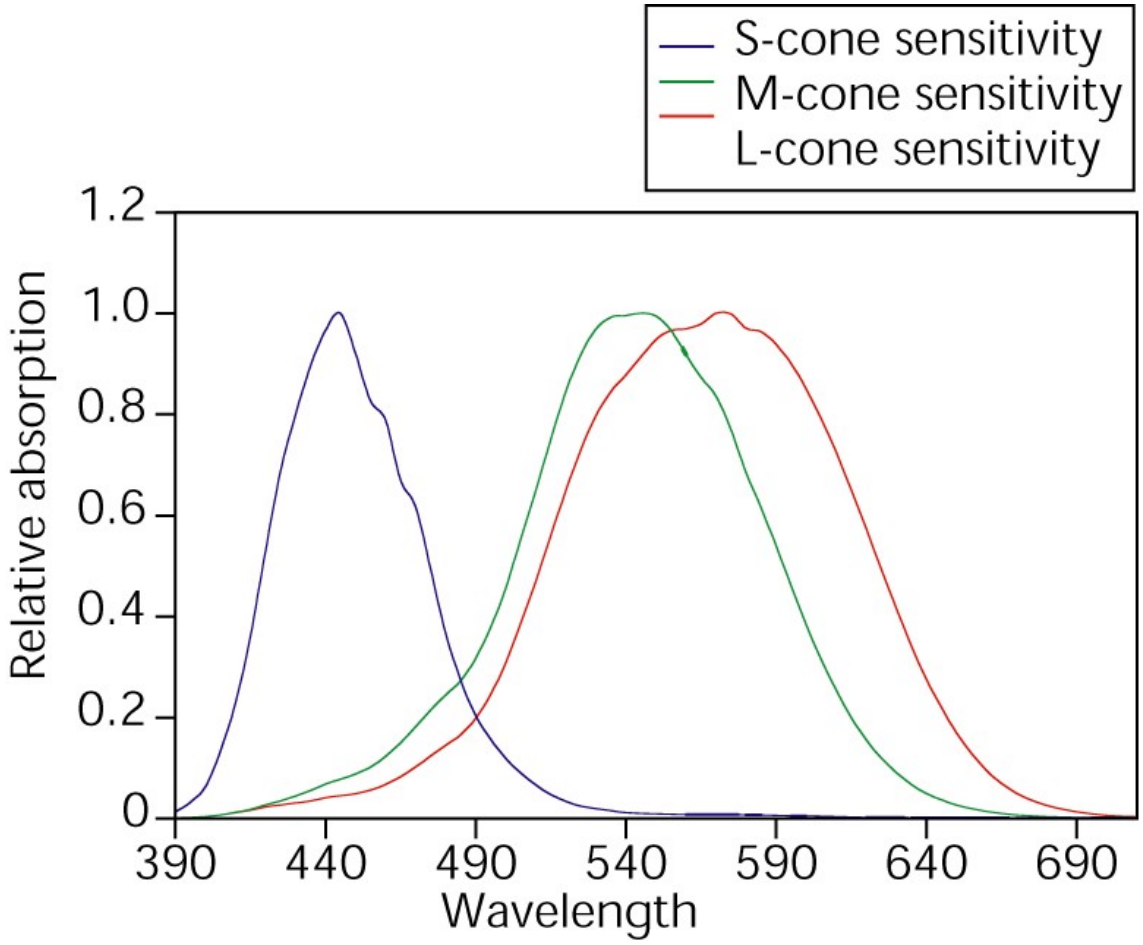
  Visual system uses 3 cone types for color

  In our model, we will treat wavelengths separately (in practice: **Red, Green, Blue**).

  In the following, we usually describe the model for a single wavelength/color channel

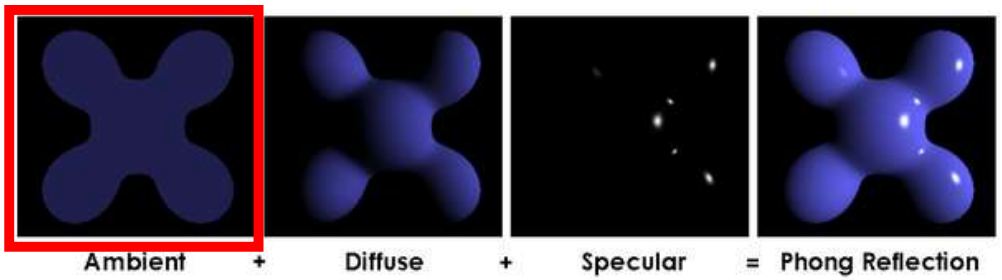
  (do it 3 times for red, green, blue…)

3 Cone types


S-cone sensitivity
M-cone sensitivity
L-cone sensitivity
Relative absorption
1.2
1.0
0.8
0.6
0.4
0.2
0
390
440
490
540
590
640
690
Wavelength

Phong Model

- Sum of 3 terms
  - Ambient
  - Diffuse
  - Specular

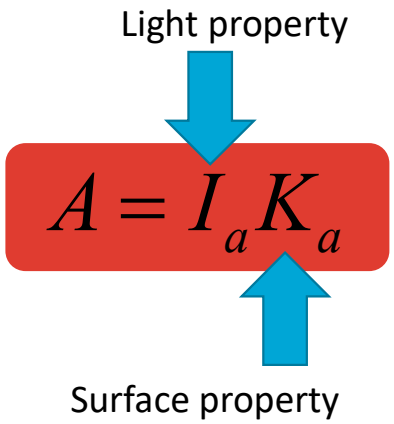


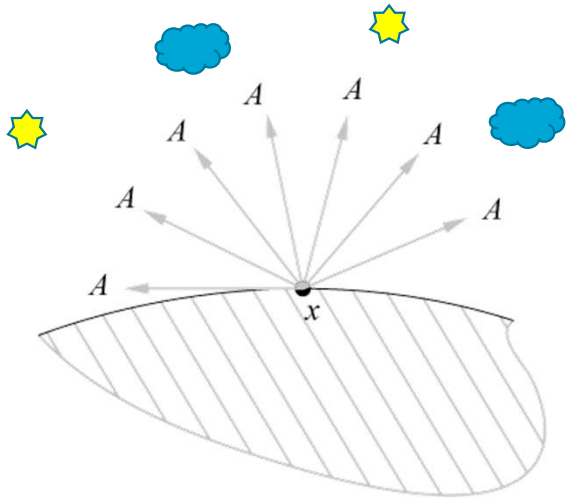
Ambient Term

- Is supposed to mimic “scene light”:
  - Skylight
  - Reflections from neighboring surfaces

- Model:
  - Very simple:

Formula is used for Red, Green & Blue.





Ambient Term

$$A = I_a K_a$$

K_a increasing

Ambient Term

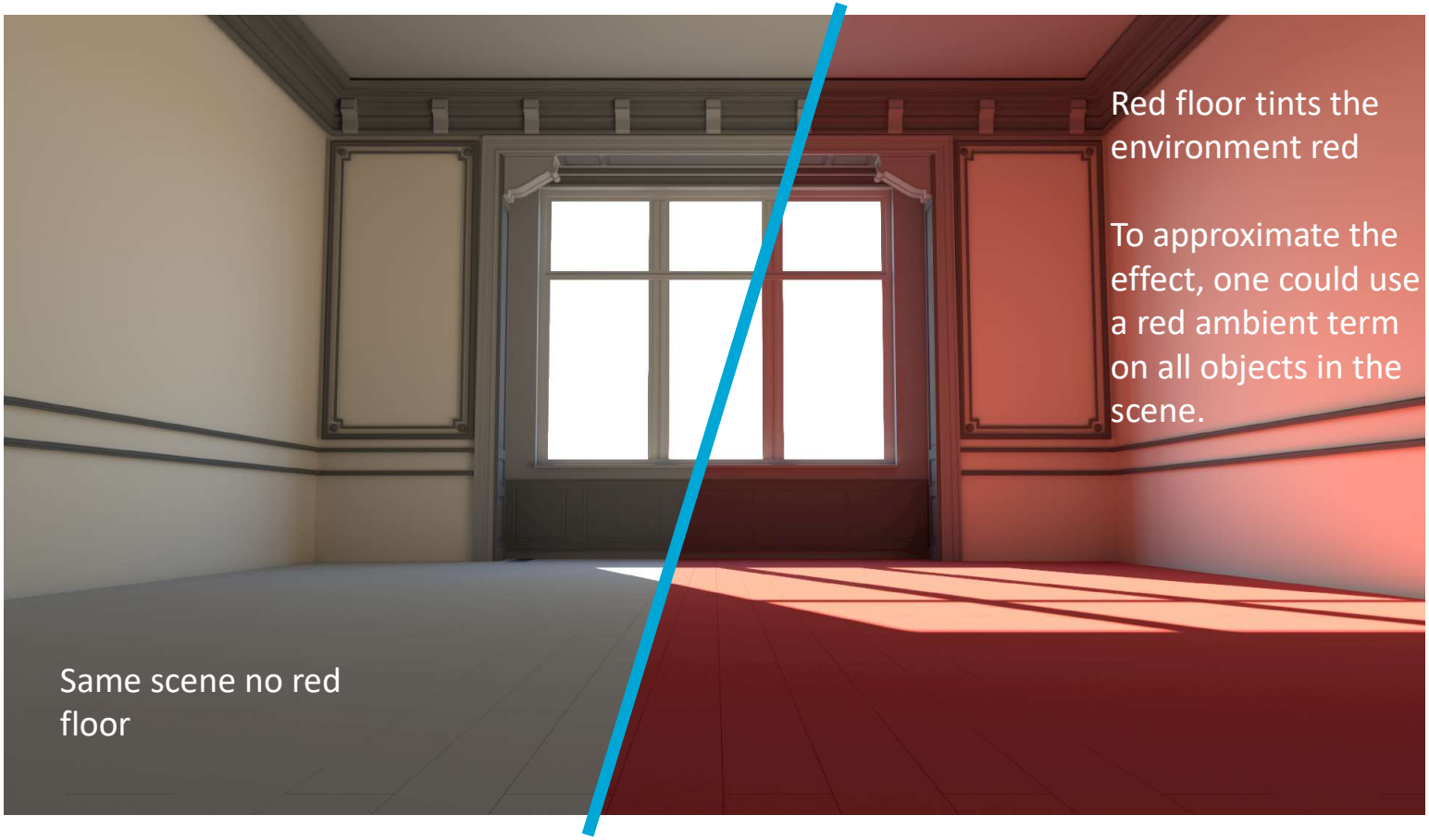
$$A = I_a K_a$$

- Very simplistic
  - No indications on the shape of an object!

K_a increasing

- Used often in practice as a strong approximation of indirect light

Example of Indirect Light



$$A = I_a K_a$$

Example Ambient

- Typically, values between [0,1] are used
- $I_a = (0.9,0.9,0.9)$ "white light"
  (Red, green and blue are close to one)

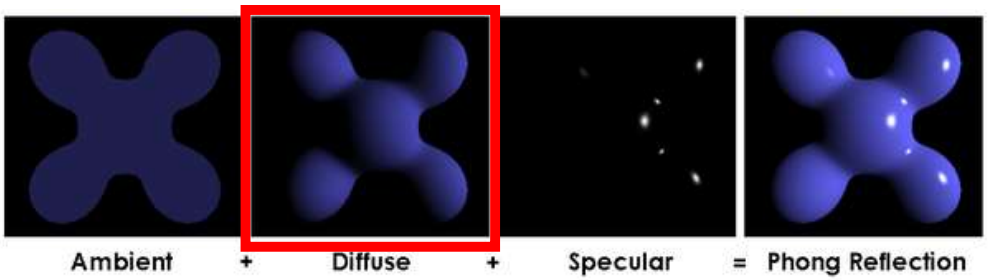
- $K_a = (0.5,0,0)$ the surface is "dark red"

- The ambient term is?

$$A = I_a K_a = (0.9,0.9,0.9) * (0.5,0,0) = (0.45,0,0)$$ ...also dark red.

Phong Model

- Sum of 3 terms
  - Ambient
  - Diffuse
  - Specular



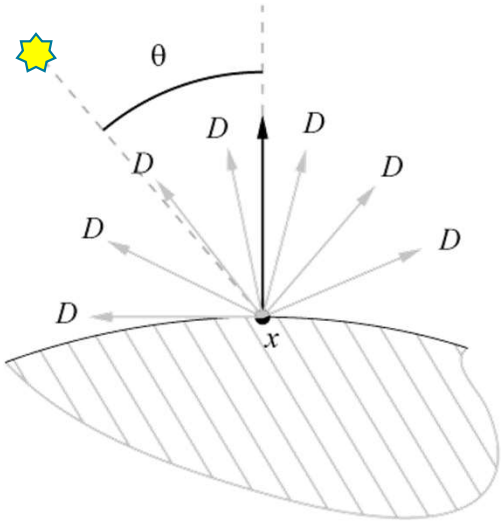
Diffuse Term

- Lambert Surfaces
  - Light is reflected uniformly in all directions

- Model:
  - Uses local surface orientation

$$D = I_d K_d \cos\theta$$

Light property (RGB) Surface property (RGB)



Diffuse Term

$$D = I_d K_d \cos\theta$$

K_d increasing

Diffuse Term

$$D = I_d K_d \cos\theta$$

- Shading varies along surface
  - Gives information about shape

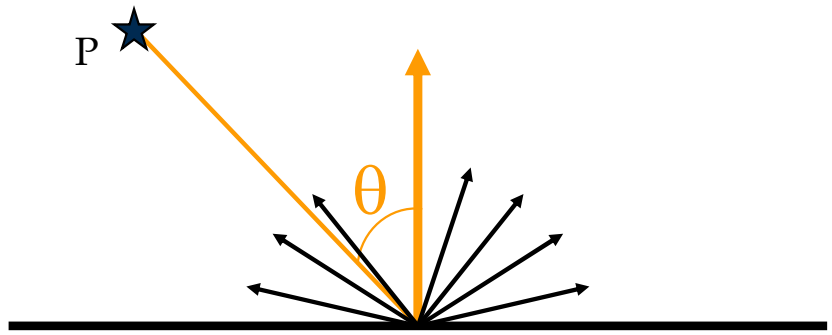
K_d increasing

  - Does not depend on observer position
    (looks the same from all positions)
  - **Careful:**
    the light should always come from above the surface, otherwise, it should stay black.
    What does this mean for the angle θ?

Diffuse Term

- Where does the cosine come from?

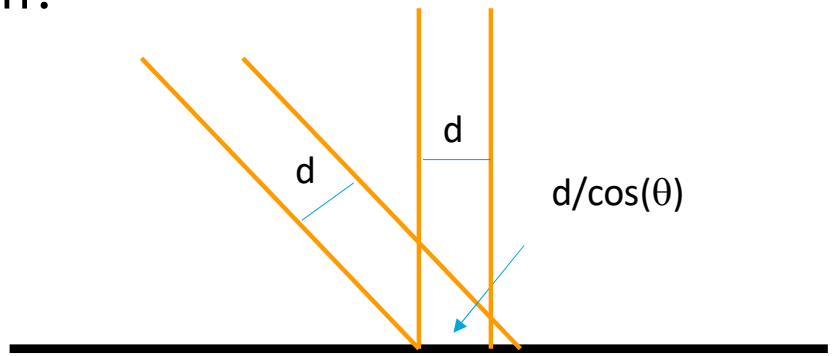
$$D = I_d K_d \cos\theta$$



Diffuse Term

- Where does the cosine come from?

$$D = I_d K_d \cos\theta$$

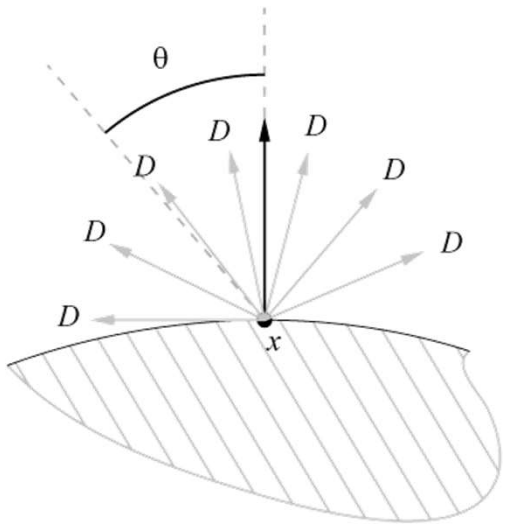


- What do you observe when tilting a flashlight?
- Imagine light arriving as parallel rays

Example

$$A = I_a K_a \qquad D = I_d K_d \cos\theta$$

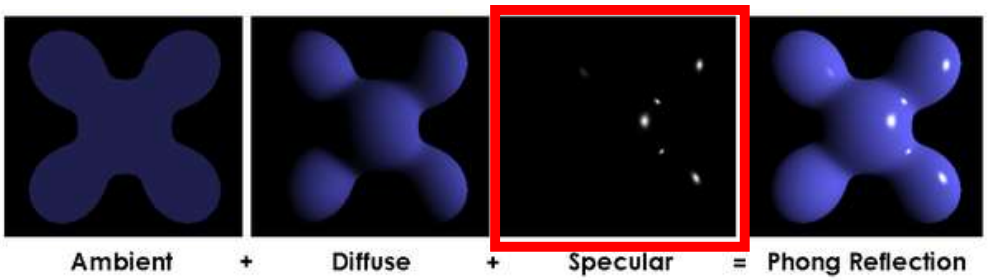
- K_d = (0.9,0,0)
- I_d = (0.9,0.5,1.0)
- K_a = (0,0,0.1)
- I_a = (1,1,0.1)



- Normal at x: (0,0,1)
- Position x: (0,0,0)
- light position : (0,0,10)
- Resulting color: (0.81,0,0.01)

Phong Model

- Sum of 3 terms
  - Ambient
  - Diffuse
  - Specular

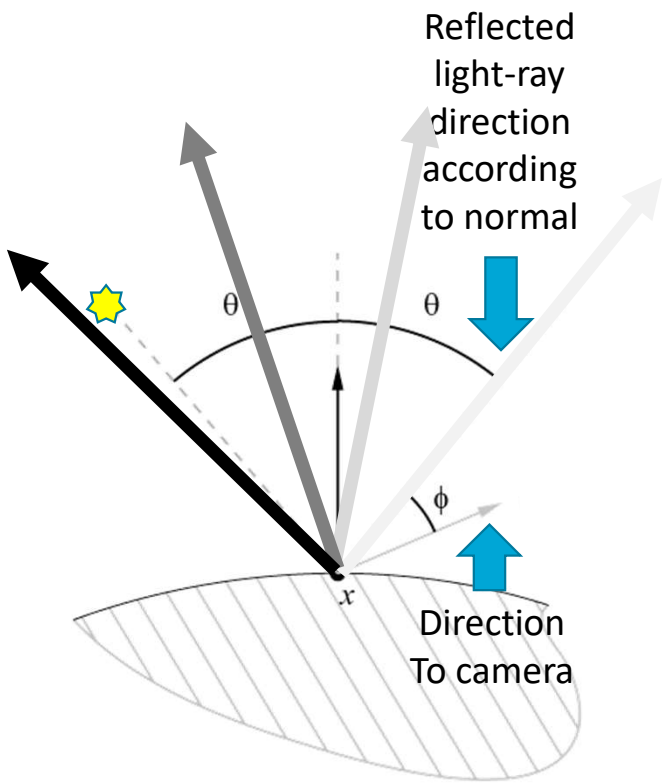


Specular Term

- Represent glossy surfaces
  - Ideal case: mirror
- Model:
  - Highlights are around the perfectly-mirrored ray
  - Exponential control over the highlight

$$S(\phi) = I_s K_s (\cos\phi)^n$$

n : *shininess*

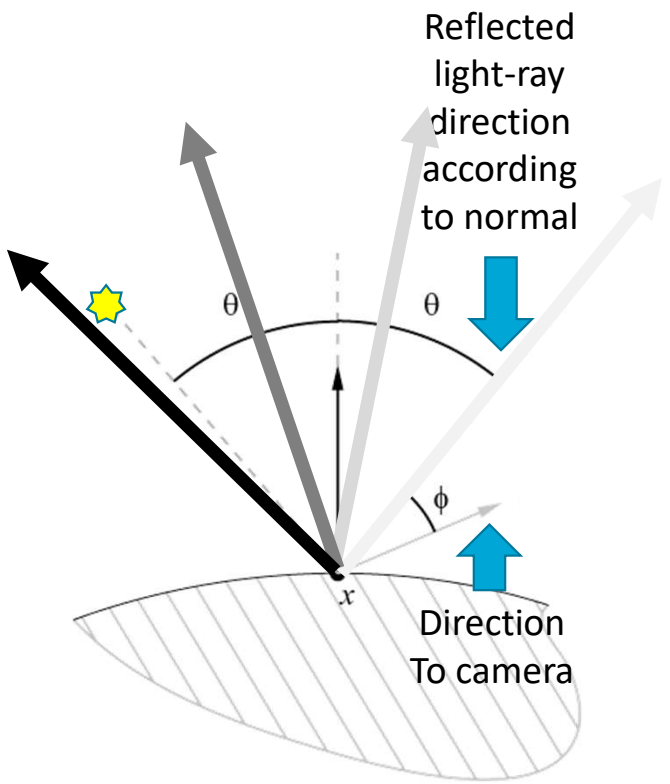


Specular Term

- Represent glossy surfaces
  - Ideal case: mirror
- Model:
  - Highlights are around the perfectly-mirrored ray
  - Exponential control over the highlight

$$S(\phi) = I_s K_s (\cos\phi)^n$$

n : *shininess*

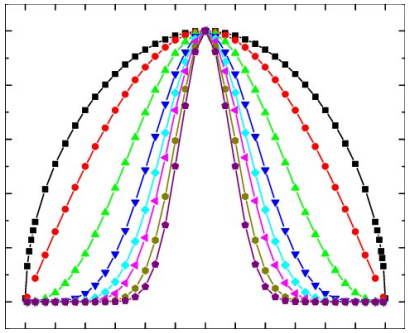


Specular Term

- Represent glossy surfaces
  - Ideal case: mirror
- Model:
  - Highlights are around the perfectly-mirrored ray
  - Exponential control over the highlight

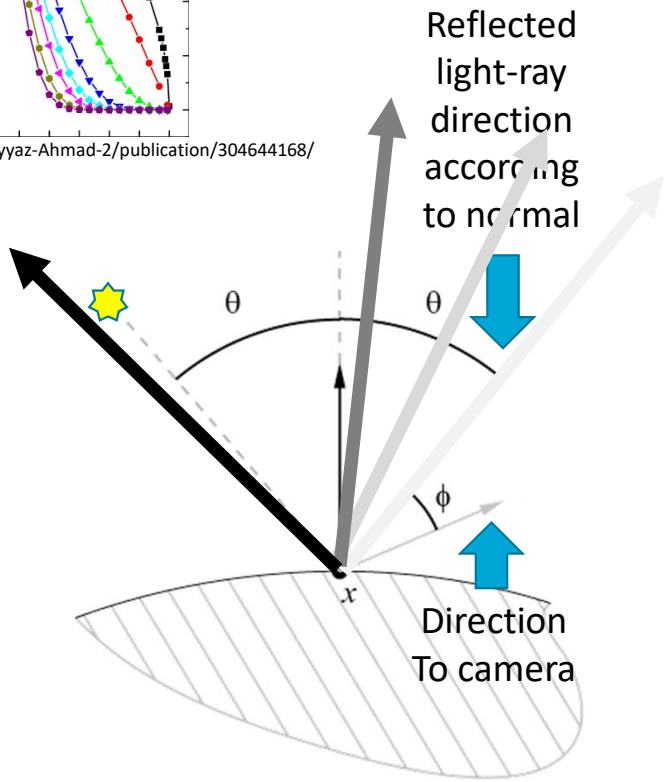
$$S(\phi) = I_s K_s (\cos\phi)^n$$

n : *shininess*



Cosine over
[-pi/2,pi/2]
to the power of n

https://www.researchgate.net/profile/Fayyaz-Ahmad-2/publication/304644168/

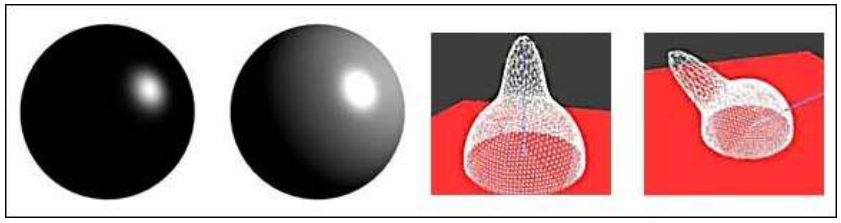


Specular Term

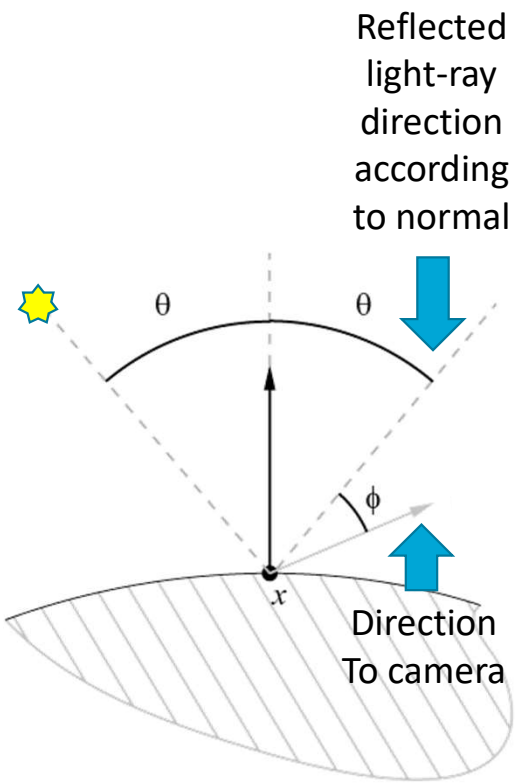
- Represent glossy surfaces
  - Ideal case: mirror
- Model:
  - Highlights are around the perfectly-mirrored ray
  - Exponential control over the highlight

$$S(\phi) = I_s K_s (\cos\phi)^n$$

n : *shininess*

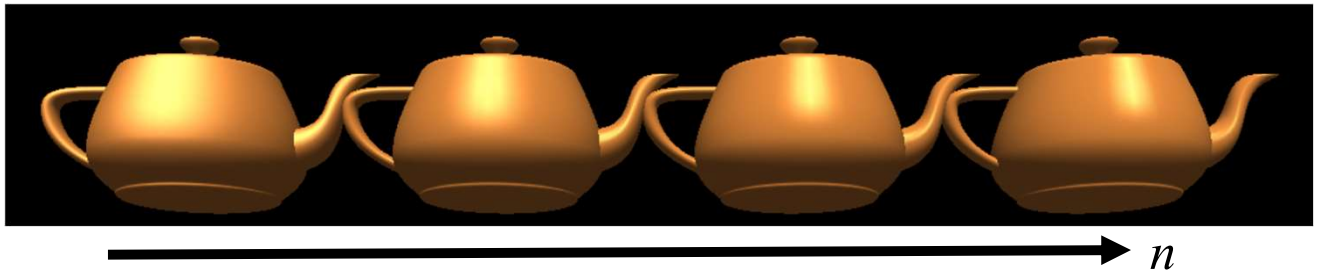


specular | specular+diffuse | S(ϕ), called the *lobe*



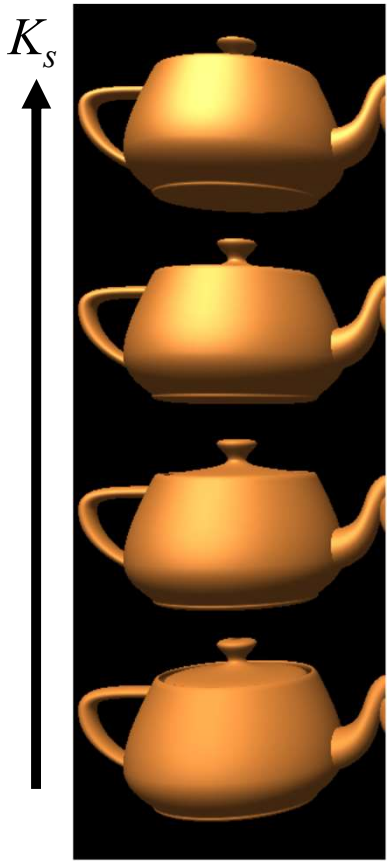
Specular Term

$$S(\phi) = I_s K_s (\cos\phi)^n$$



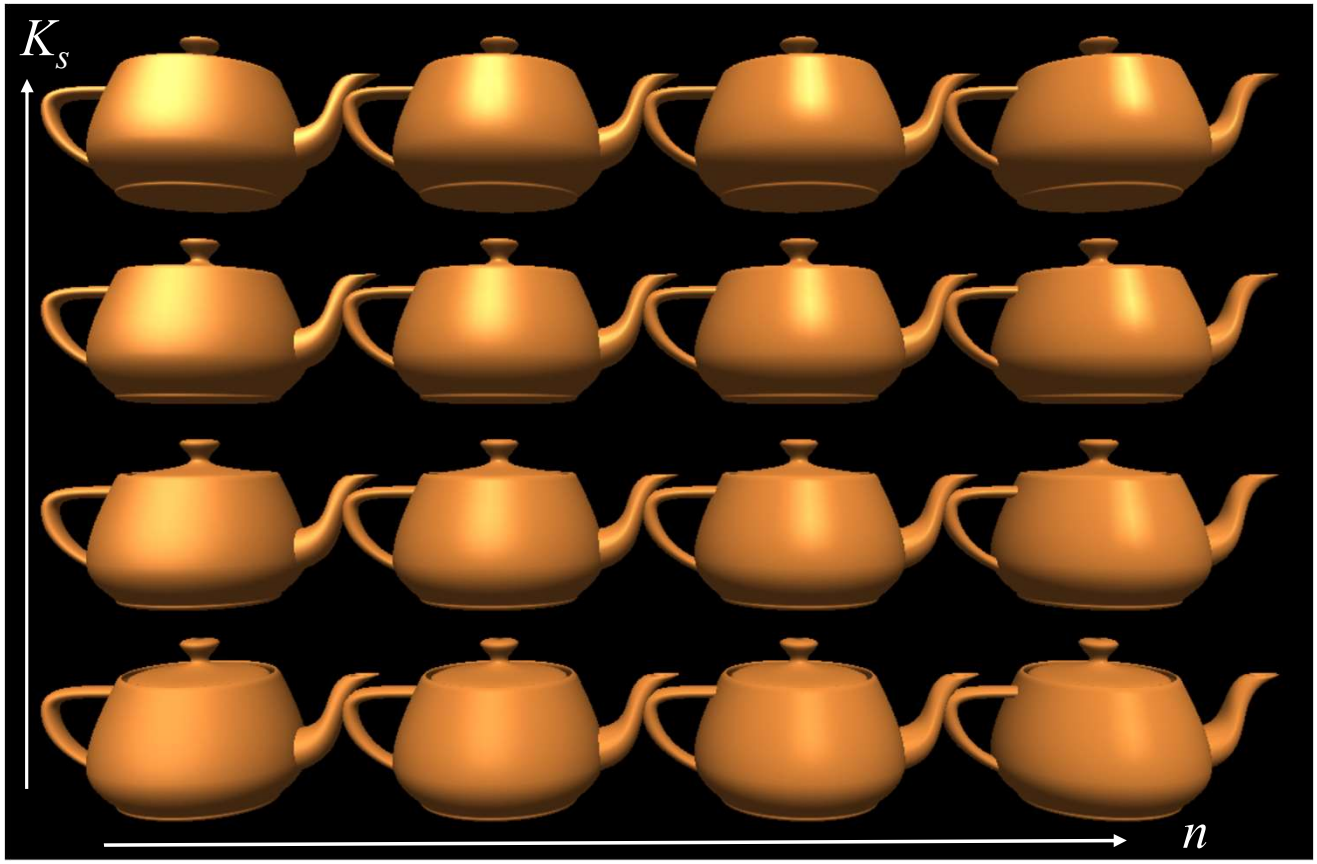
Specular Term

$$S(\phi) = I_s K_s (\cos\phi)^n$$



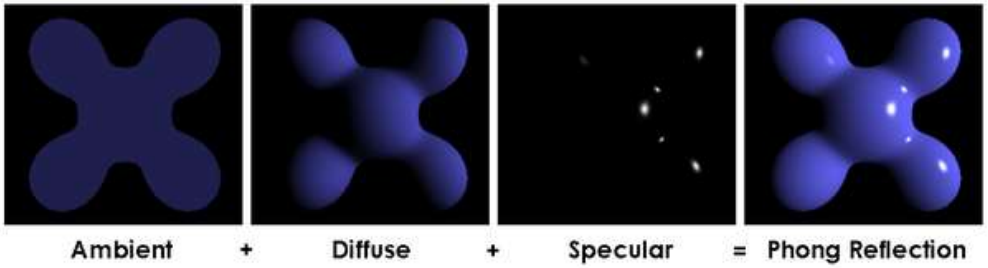
Specular Term

$$S(\phi) = I_s K_s (\cos\phi)^n$$



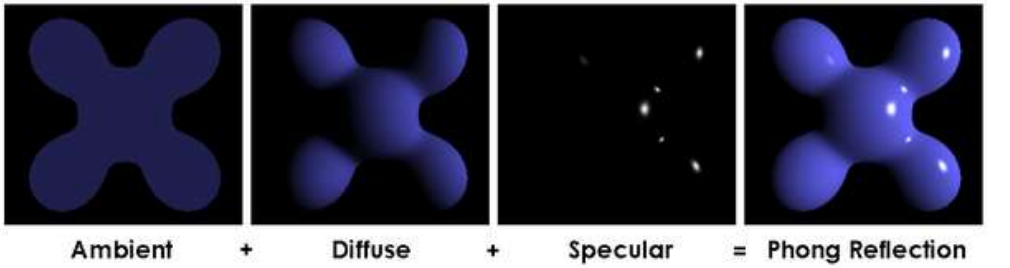
Phong Model

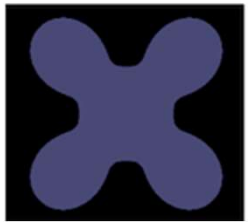
- Sum of 3 terms
  - Ambient
  - Diffuse
  - Specular



Phong Model

- Sum of 3 terms
  - Ambient
  - Diffuse
  - Specular



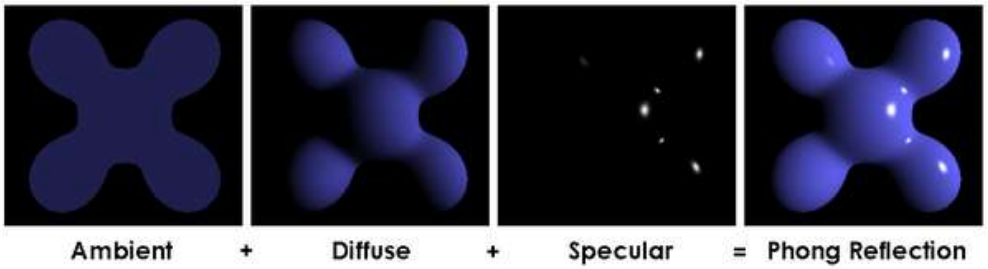


- Optional Extension:
  - Emission = Ambient with a Light set to 1
    - Idea: object is emitting light (e.g., hot glowing metal)

Mathematical Models

- Mathematically describe Material Properties

- Phong Model: Sum of 3 terms
  - Ambient
  - Diffuse
  - Specular



- In the literature: Many more material models

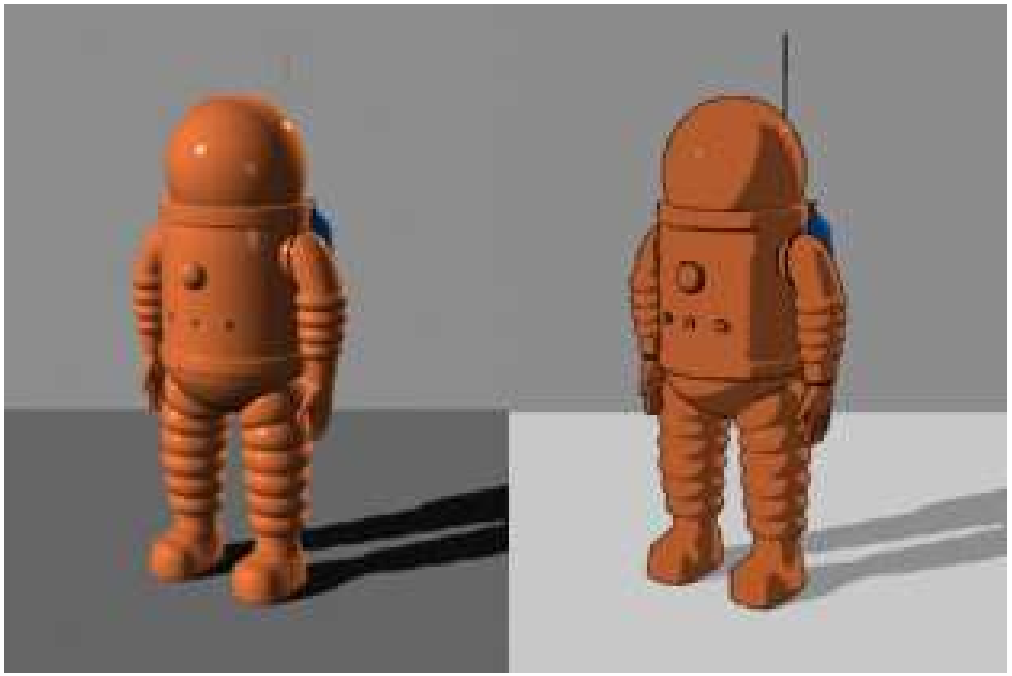
  Tradeoff between efficiency and accuracy

Advanced Material Models

https://blog.playcanvas.com/physically-based-rendering-comes-to-webgl/

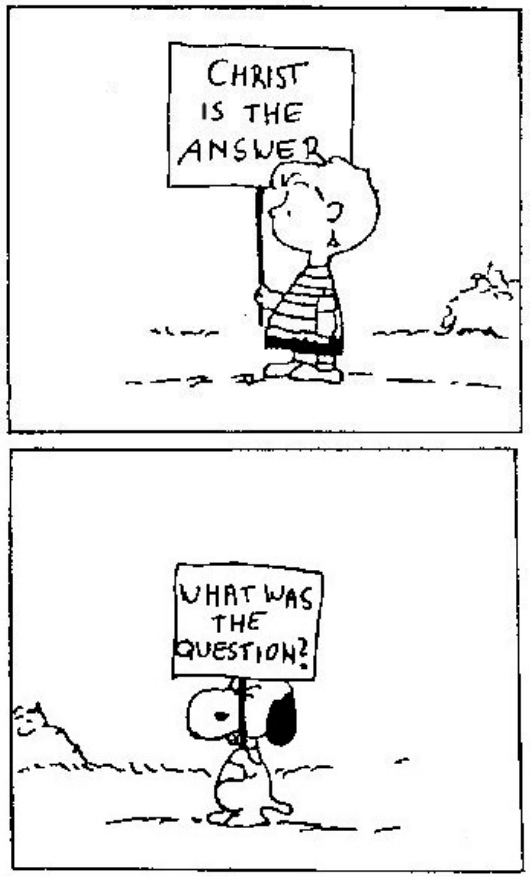
Extreme Example…

- Non-photorealistic materials



- Cel-shading: threshold on the diffuse shading

Questions?



How to apply Phong Model ?

- We know how to compute shading of a point, but how is it applied on a triangle mesh?

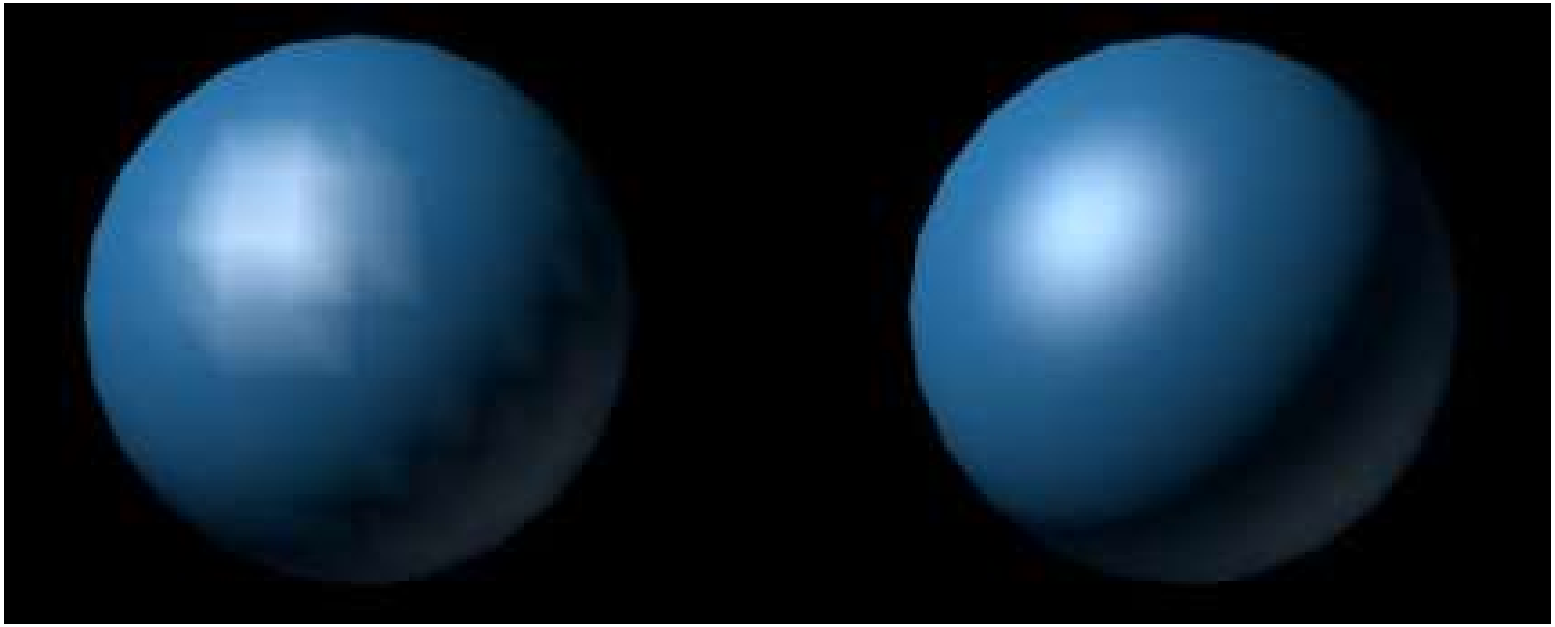
Shading

- *Early days - compute color per face:*
  Flat shading produces “facets”

- Later – compute color per vertex:
  produces *Gouraud Shading* produces a smooth look

Phong shading

- Today: compute result per pixel
- Phong Shading leads to smooth specularities

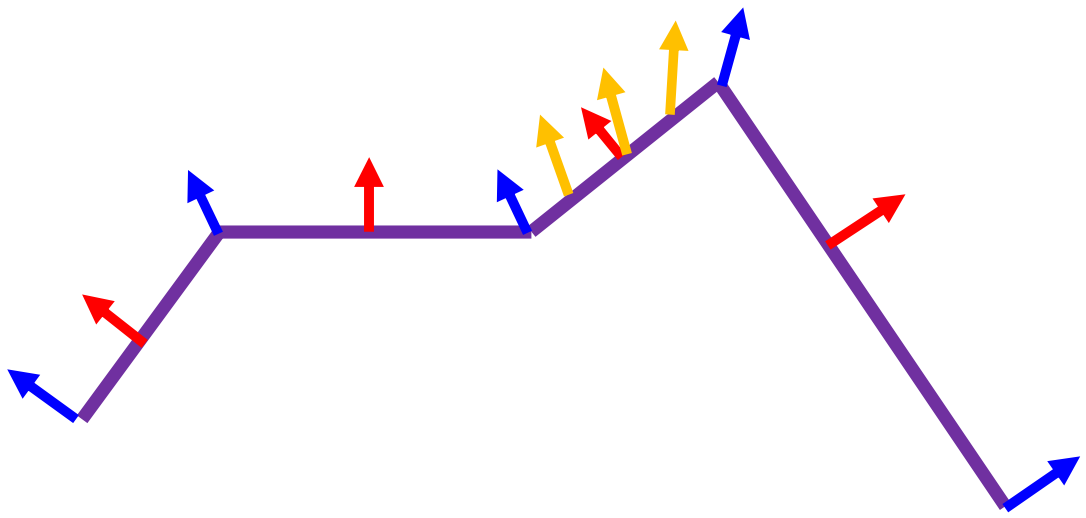


Diccan.com

- Phong interpolates normals from vertices over pixels

Normals on Meshes

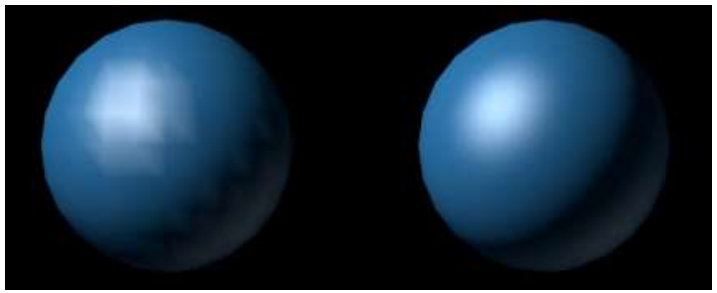
- Face normals (normal of the plane containing triangle)
- Vertex normals (e.g., average neighboring face normal)
- Interpolated normal (interpolate vertex normals over triangle)



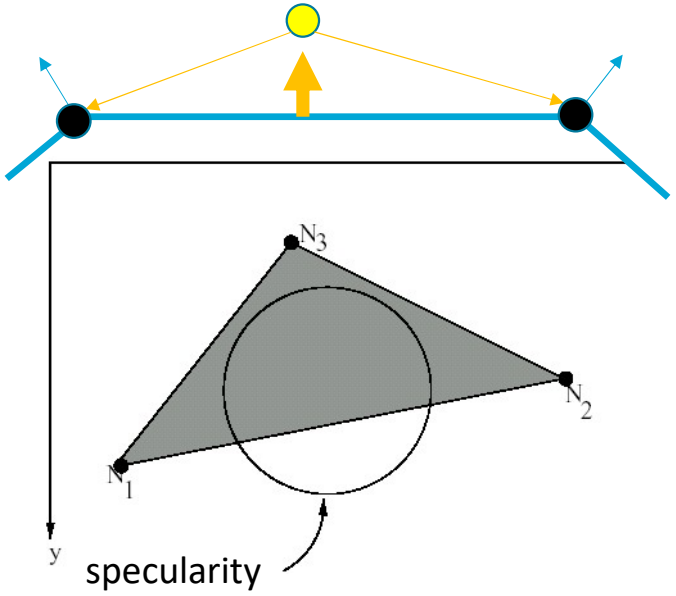
Per vertex *Per pixel*

Gouraud *vs.* Phong shading

- Phong usually more expensive than Gouraud
  - Because there are often more pixels than vertices
- Phong is more beautiful and minimal standard
  - Captures specularities between faces

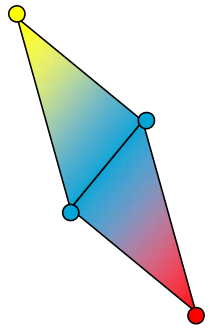


Diccan.com



On the practical side: Shading types

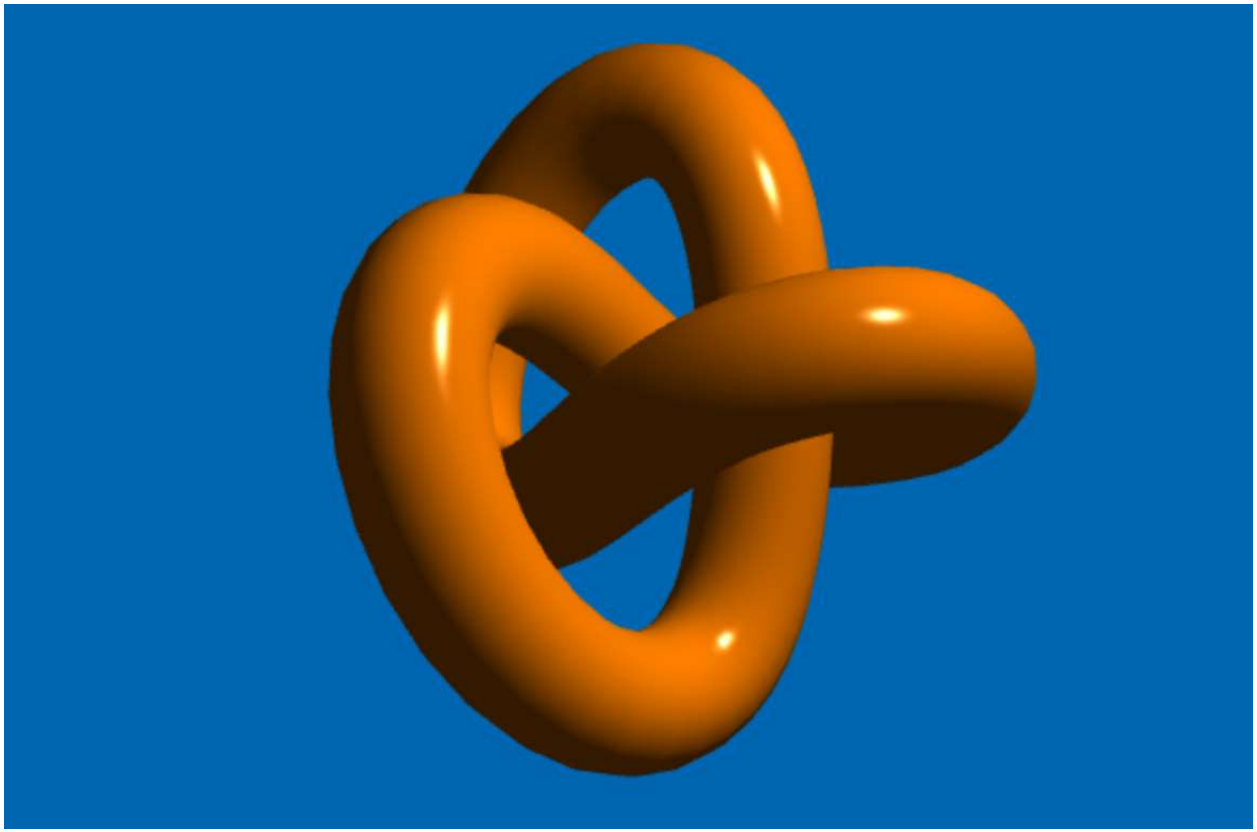
- How are the three different types computed?
  - *Flat shading*
    - Applies Phong Model to produce a color per face
  - *Gouraud shading*
    - Applies Phong to produce a color per vertex
    - Interpolate color from vertices over triangle
  - *Phong shading* ← 2 MEANINGS!!!
    - Interpolate parameters of Phong model
    - Applies Phong to produce a color per pixel



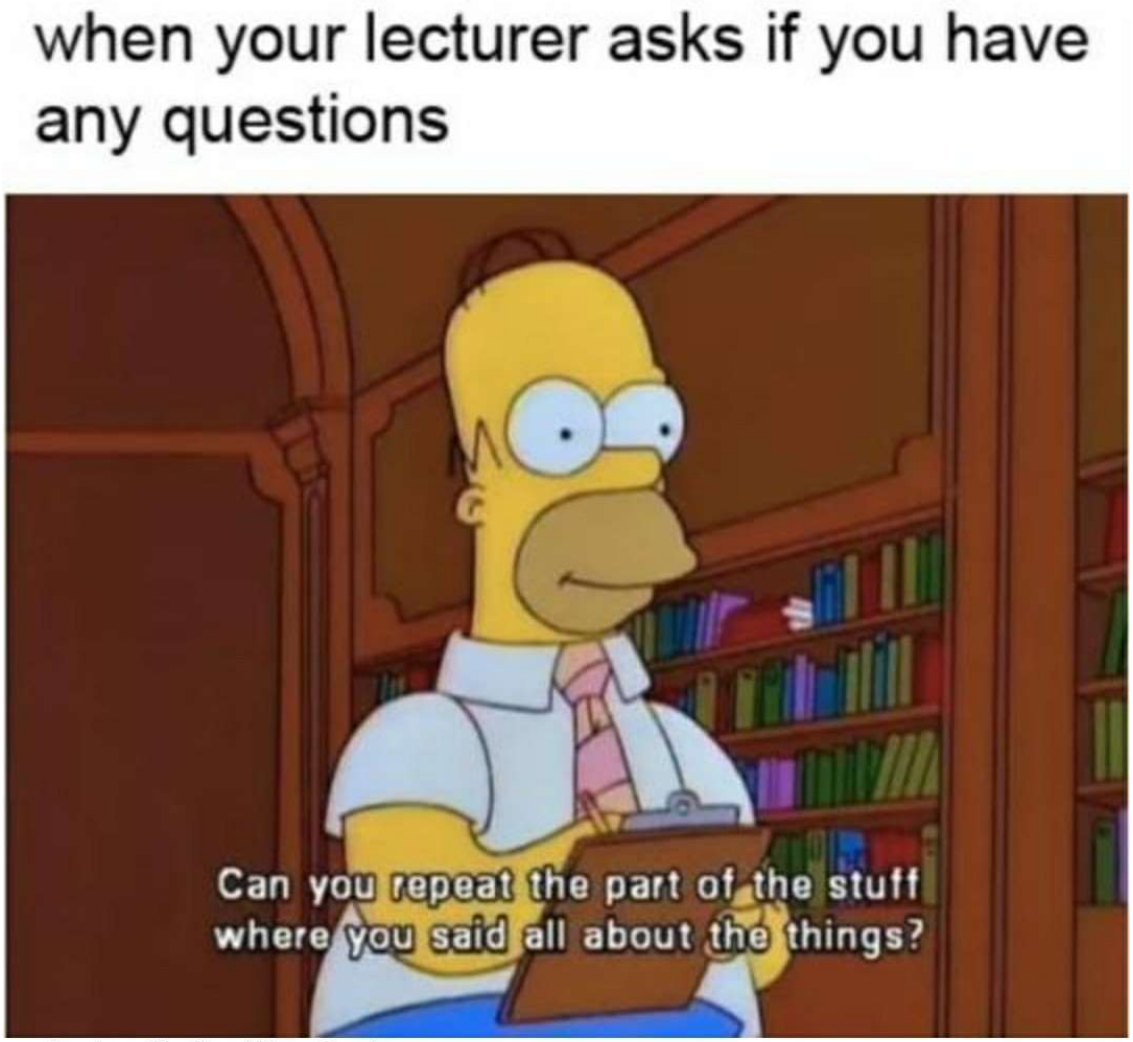
Demo Time

Demo Time

- http://multivis.net/lecture/phong.html



Questions?



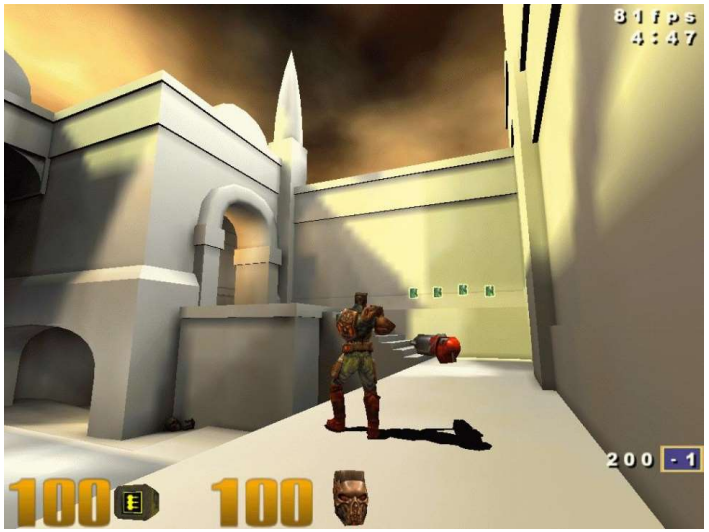
Jemima Skelley / BuzzFeed

Summary

- Graphics Pipeline Core:
  - Geometry
    - Transformation and Projection
  - Shading
    - Material model (Ambient, Diffuse, Specular)
    - Shading interpolation (Flat, Gouraud, Phong)

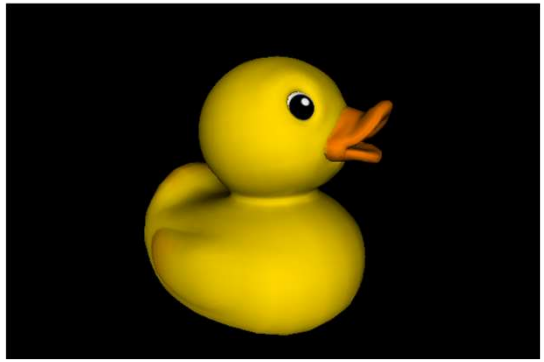
Something is still missing...

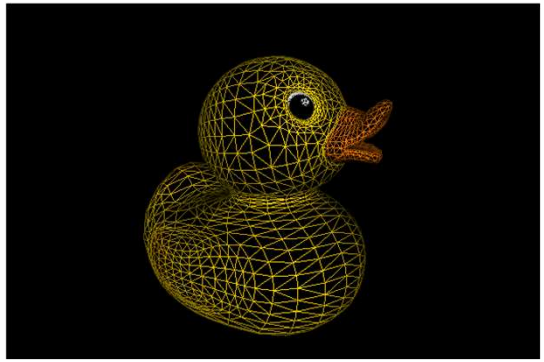
Misses quick material changes



Textures

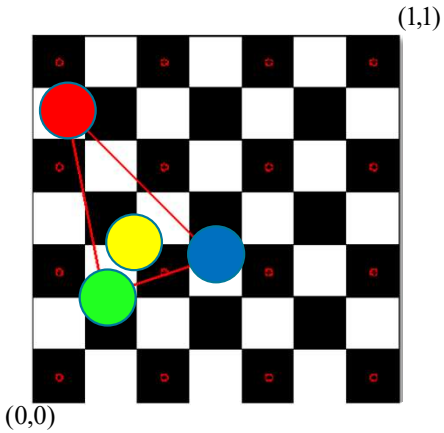
- Can be used to provide, e.g.,
  - parameters for ambient/diffuse material models





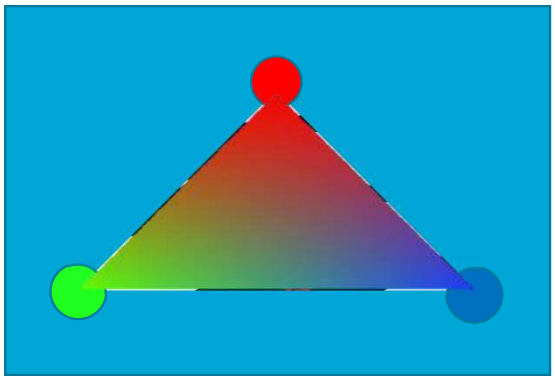
Textures

- Image mapped on the surface via texture coordinates
  - Specified at each vertex `glTexCoord{123}{fi}`
  - Interpolated over triangles



domain

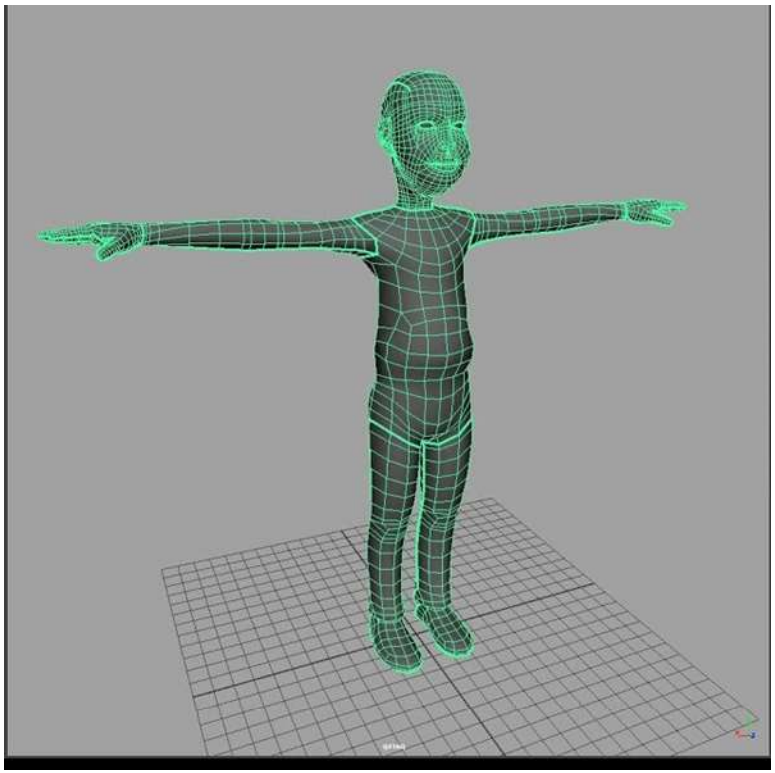
```
TextureCoords(0.2f,0.3f);
Vertex(-1.0f,0.0f);

TextureCoords(0.5f,0.4f);
Vertex(+1.0f,0.0f);

TextureCoords(0.1f,0.8f);
Vertex( 0.0f,1.0f);
```



3D View

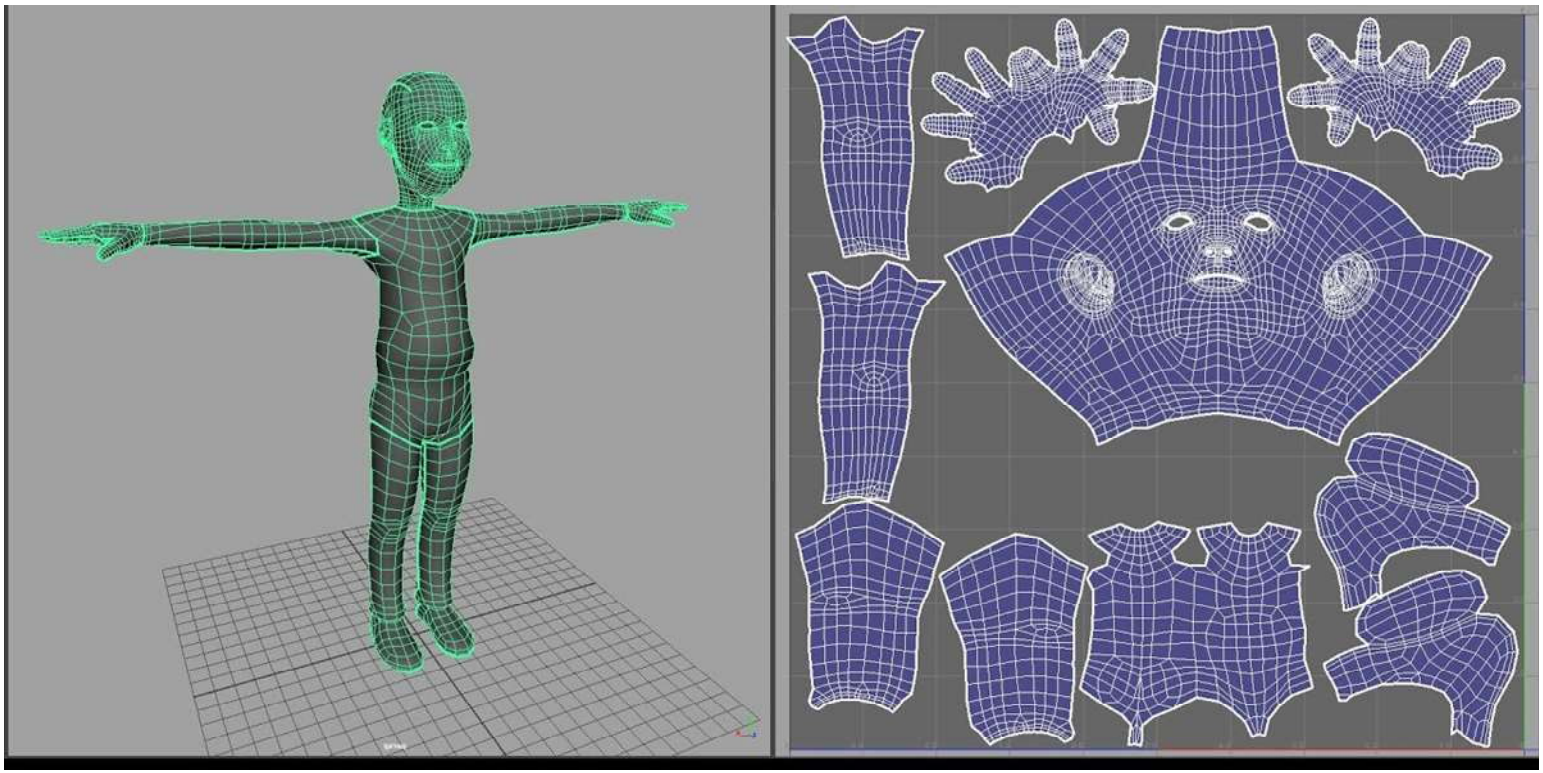
How to define Texture Coordinates?

- Common start: Mesh Unwrapping

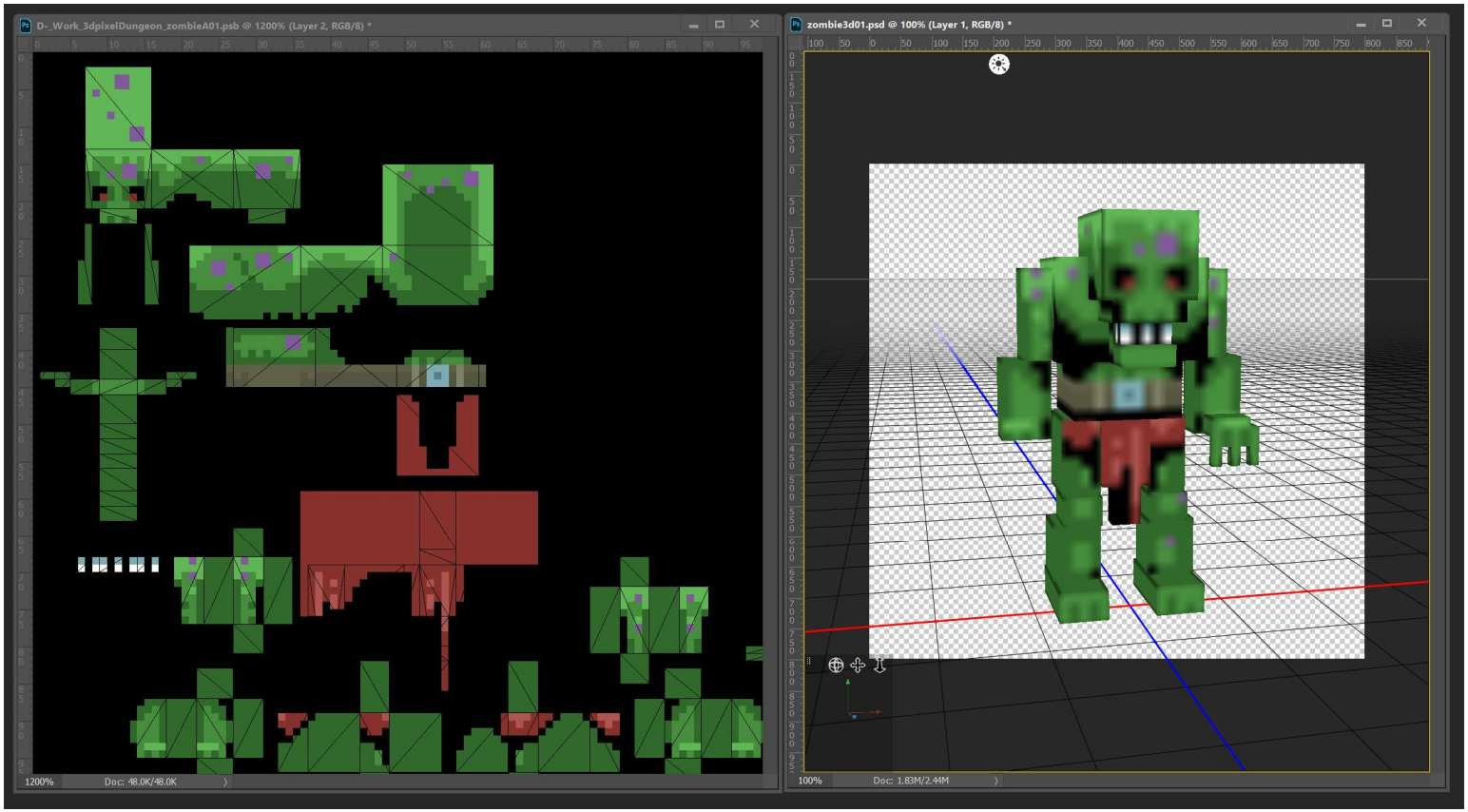


How to define Texture Coordinates?

- Common start: Mesh Unwrapping



Specialized Software

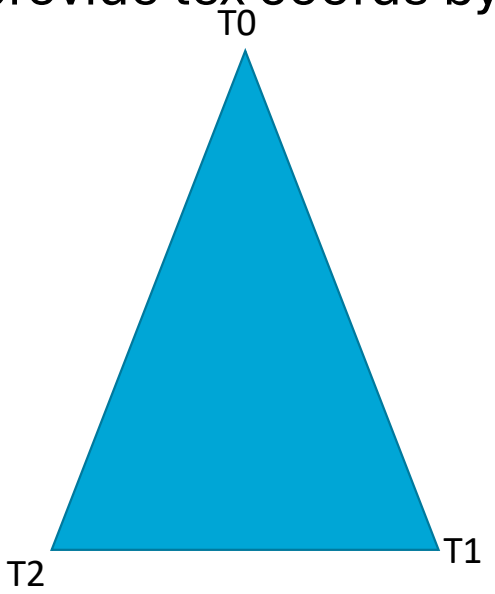


Unfold a given mesh onto a plane… possible to draw directly on the mesh

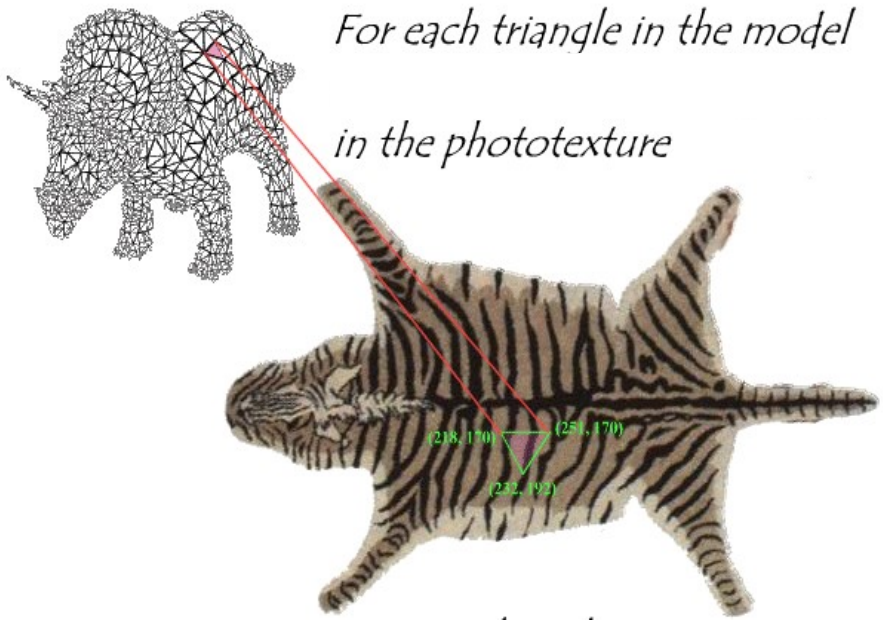
Specialized Software

How to define Texture Coordinates?

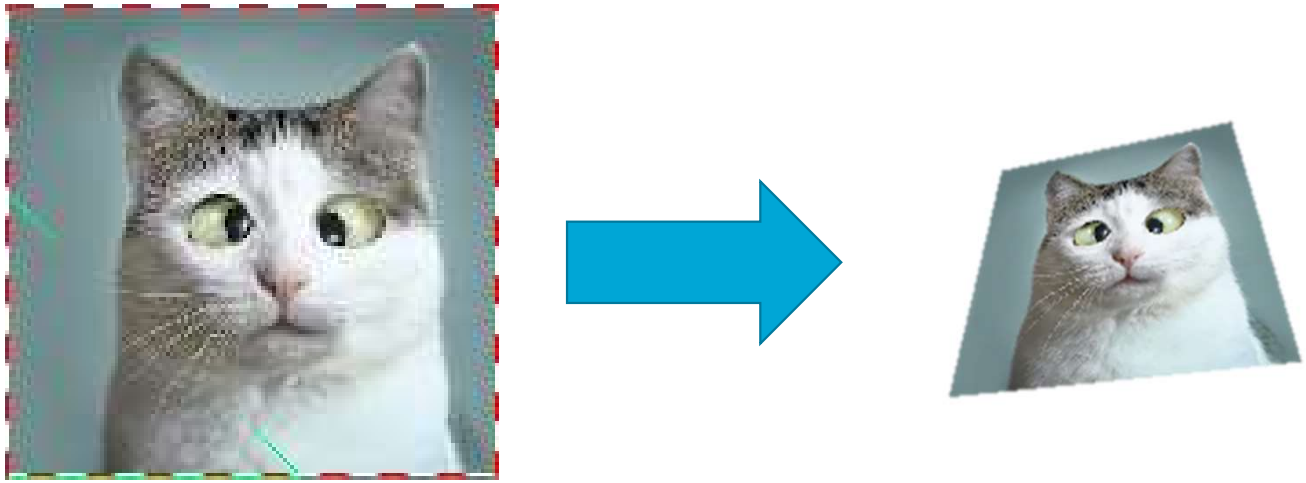
- We will provide tex coords by hand!



Texture coordinates are handled like attributes and interpolated over triangle!!!



Example of a mapped texture



Questions?


It is our duty as scientists to question everything.
Says who?
CARTOONSTOCK .com

Thank you very much for your attention!