Test Framework for Shaders

Project for the course
„Visuelle Effekte" (Visual Effects)

Kaveh Yousefi

Version History

Version	Date	Changes
1.0	10.07.2015	First version, document created.
1.1	23.09.2015	Added parameter description for extended Lommel-Seeliger model, and corrected mistake having mixed-up the Schlick approximation for the Fresnel term with the empirical approximation.
1.2	23.09.2015	Corrected a mistake in the empirical approximation formula for the Fresnel term, having forgotten the exponent ("power").
1.3	23.09.2015	Added further problem information and hyperlinks to the project files.
1.4	23.09.2015	Correct a mistake on page 50, calling the normal distribution function the "standard distribution function".

Content

- Goals

- Definitions

- Presentation of implemented shaders

- Description of the test framework

- Evaluation and future work

Goal

- The goal is the implementation of a shader test framework.

- Utilized technologies:

  - HTML5

  - JavaScript

  - WebGL

  - CSS (*Cascading Stylesheets*).

- Contains implementations of various shaders*.

- Allows

  - experimenting with different lighting models

  - comparing different lighting models.

*) Note: A shading model is usually a rephrasing and simplification of a BRDF (*Bidirection Reflectance Distribution Function*). See [dempski2005advanced], page 90.

Definitions

- Anisotropy:

  - A material's reflective property depends on the angle of the viewer looking at the surface.

  - Opposite: isotropy.

- Fresnel effect:

  - The fraction of the reflection/refracted light depends on the viewing angle.

  - Example: At some angles water seems "less transparent".

Sources: [3drender2001]; [dempski2005advanced], page 89.

Example - Anisotropy
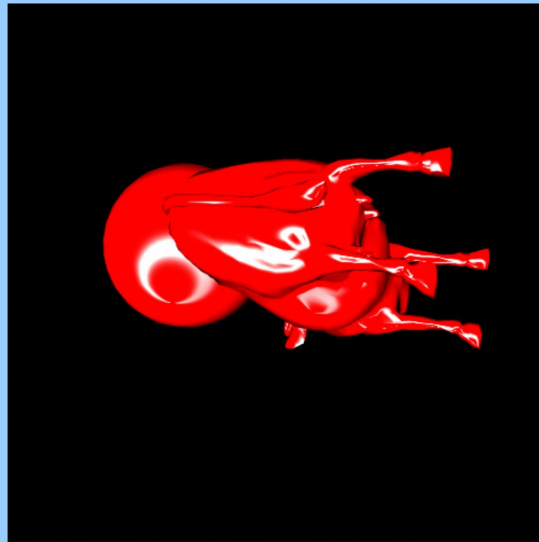
Example of anisotropy: The lighted area's shape changes with the viewing angle.
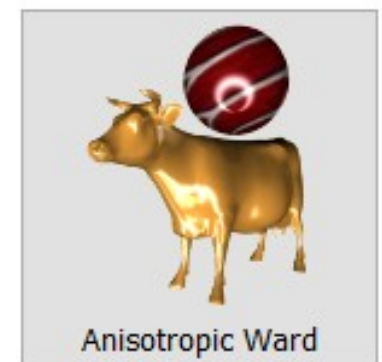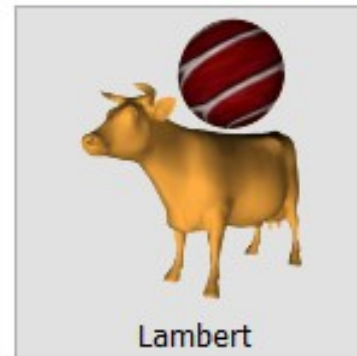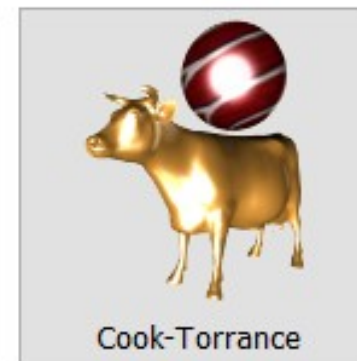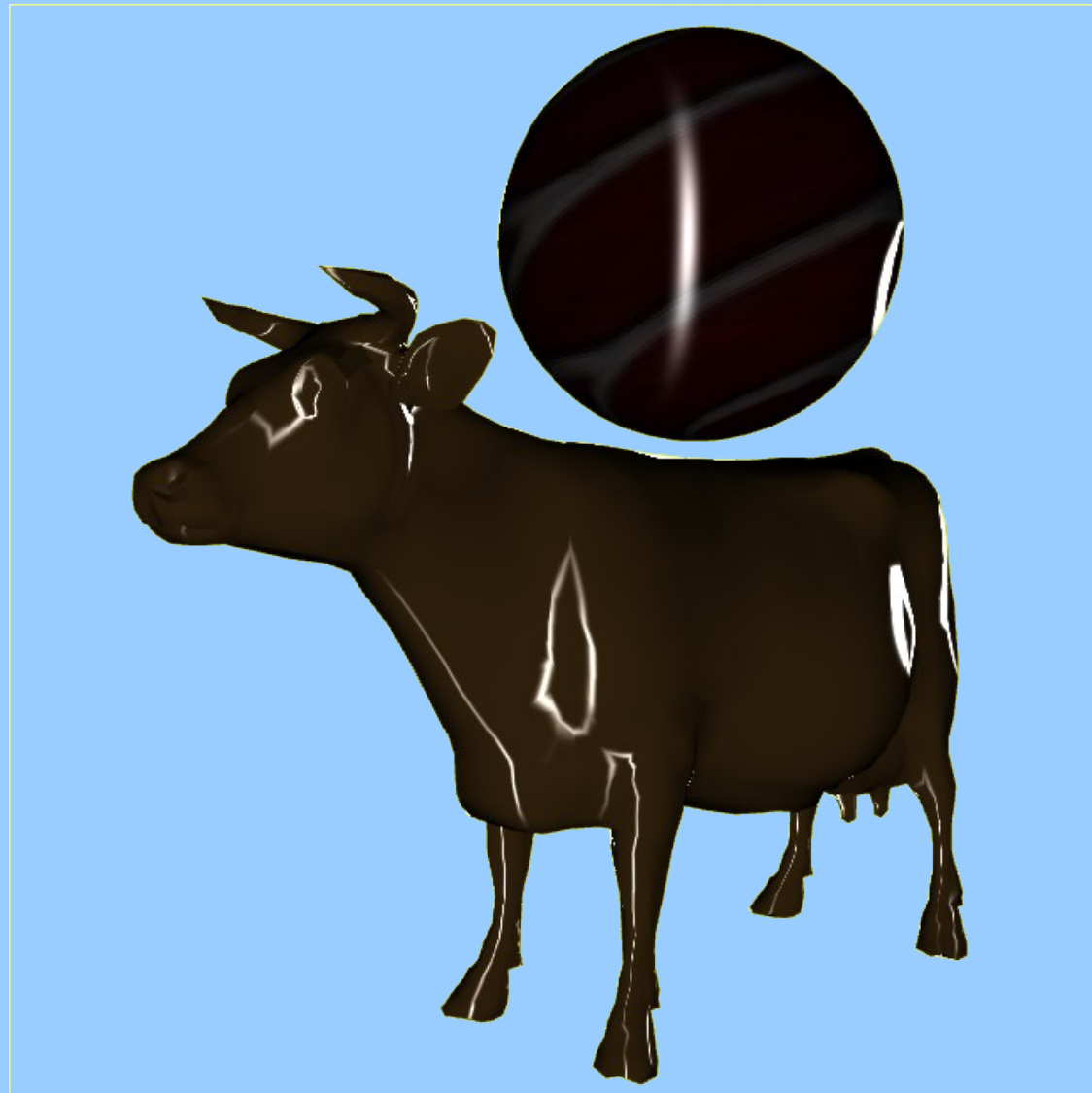
Example – Fresnel Effect
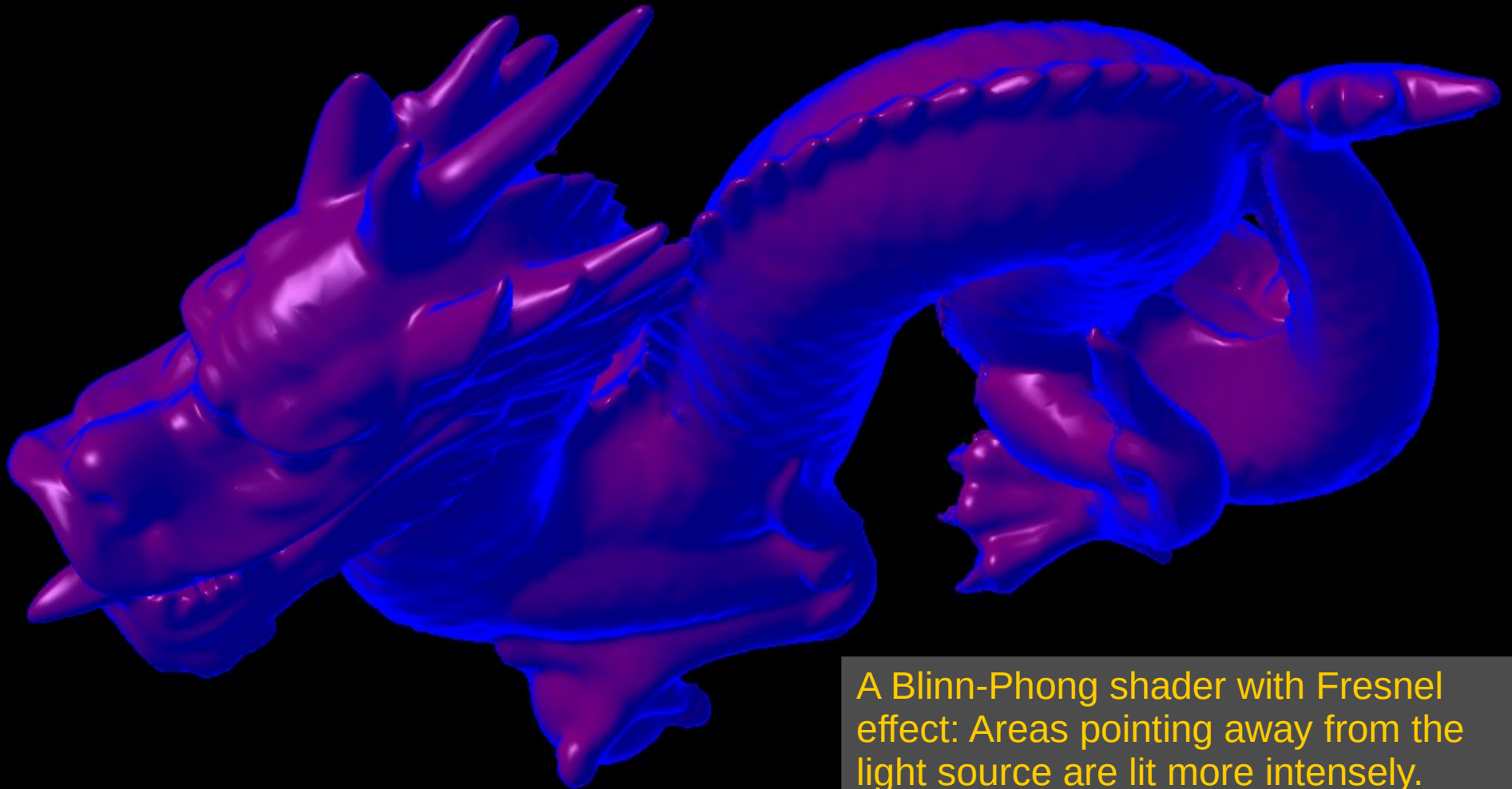
A Blinn-Phong shader with Fresnel effect: Areas pointing away from the light source are lit more intensely.

Presented Models

- Ashikhmin-Shirley
- Cook-Torrance
- Gooch
- Lommel-Seeliger
- Minnaert
- Oren-Nayar
- Phong (Lambert) variants
  - Phong
  - Blinn-Phong
  - Schlick
  - Gaussian
- Strauss
- Ward (anisotropic)

Ashikhmin-Shirley

Ashikhmin-Shirley

- An anisotropic, specular model.

- Comprised of a diffuse ρ_d and a specular term ρ_s.

- Uses two exponential factors (n_u, n_v):

  - Determine the specular reflection's shape.

  - Can be very large, for instance, larger than 10000.

  - If $n_u = n_v$: specular area is circular, similar to Phong model.

- Formula:

$$\rho(\vec{l}, \vec{v}) = \rho_d(\vec{l}, \vec{v}) + \rho_s(\vec{l}, \vec{v})$$

Ashikhmin-Shirley

- Diffuse term:

  - May optionally be replaced by a more efficient, but less realistic, term, e.g., a Lambertian.

  - Formula:

$$\rho_d = \frac{28 \times R_d}{23 \times \pi} \times (1 - R_s) \times \left(1 - \left(1 - \frac{\vec{n} \cdot \vec{l}}{2}\right)^5\right) \times \left(1 - \left(1 - \frac{\vec{n} \cdot \vec{v}}{2}\right)^5\right)$$

Ashikhmin-Shirley

- Specular term:
  - Contains a Fresnel term, usually the Schlick approximation.
  - Uses the two exponential factors n_u, n_v.
  - Needs two additional vectors besides the surface normal:
    - Tangent (U)
    - Bitangent (V).
  - Formula:

$$\rho_s = \frac{\overline{\sqrt{(n_u+1)(n_v+1)}} \times (\vec{n} \cdot \vec{h})^{n_u \cos^2 \varphi + n_v \sin^2 \varphi}}{8 \times \pi \times (\vec{h} \cdot \vec{l}) \times max((\vec{n} \cdot \vec{l}),(\vec{n} \cdot \vec{v}))} \times F(\vec{h} \cdot \vec{l})$$

Ashikhmin-Shirley - Parameters

| Parameter	Description	Notes
diffuseColor (Rd)	The diffuse surface color.	
specularColor (Rs)	The specular surface color.	
exponentX (Nu)	Exponent to control lobe.	Creates anisotropy.
exponentY (Nv)	Exponent to control lobe.	Creates anisotropy.
referenceAxis (epsilon)	Basic 3D vector to define the reference frame together with the surface normal.	Usually no parameter, but the vector (1, 0, 0).

Cook-Torrance
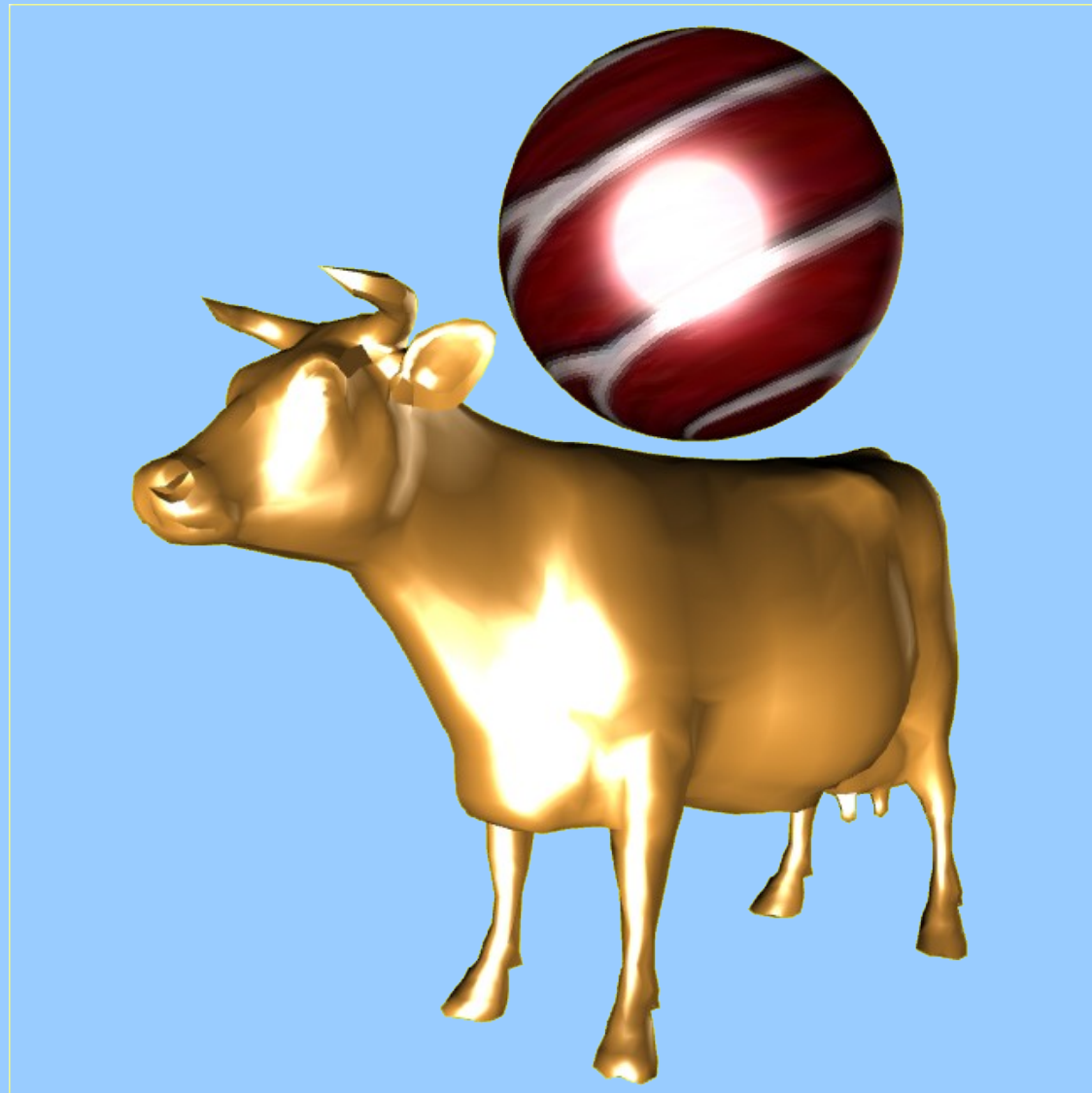
Cook-Torrance

- Physically based lighting model.

- Regards the surface as a set of microfacets, each pointing to a certain direction.

- Simulates **shadowing** and **masking**:

  - *shadowing*: Microfacets can prevent light from reaching other microfacets.

  - *masking*: Microfacets can block light from being reflected from other microfacets to the viewer.

  - Both effects influence the diffuse and specular term.

Cook-Torrance - shadowing
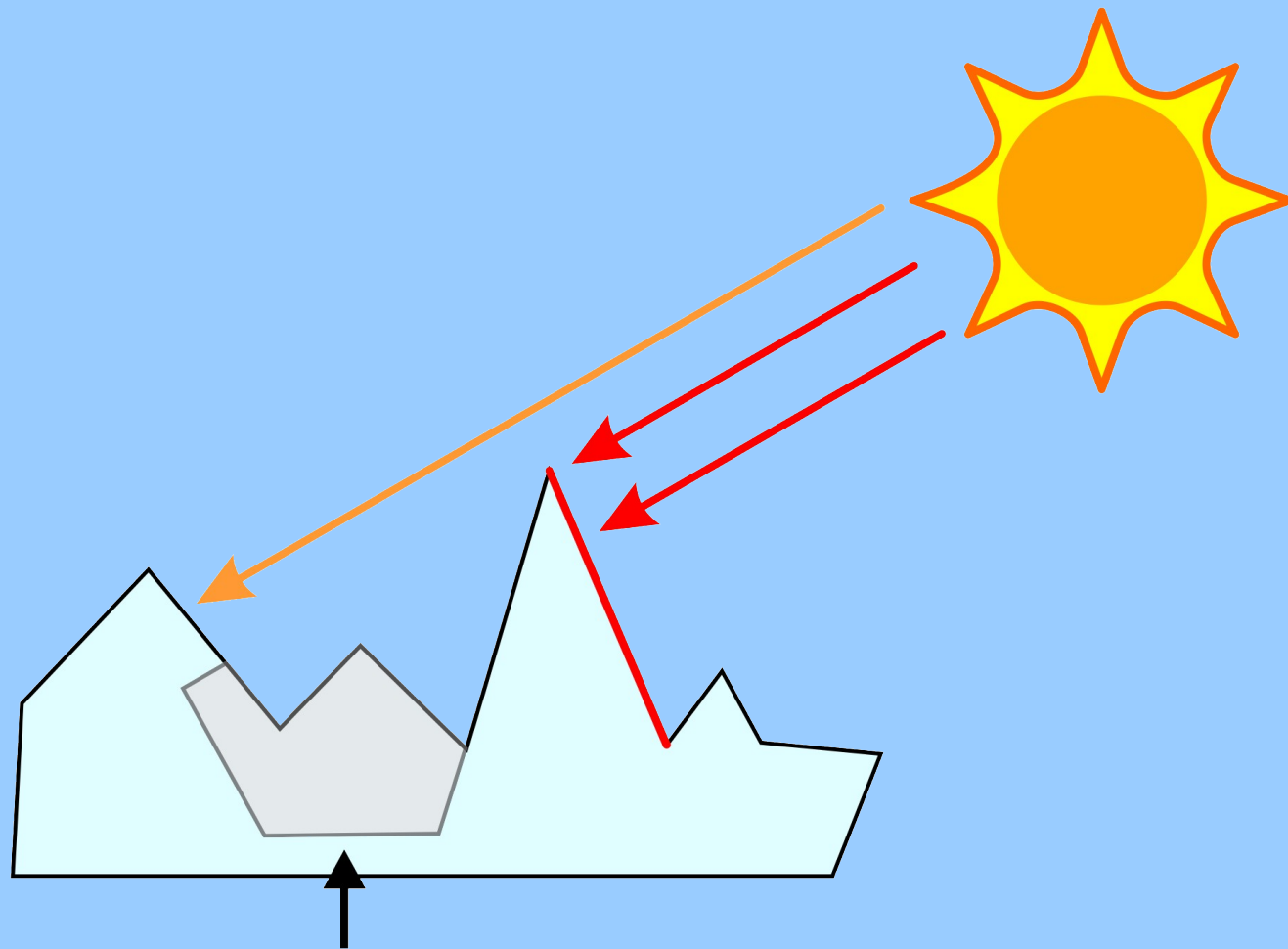
The right microfacet blocks incoming light and thus **shadows** the microfacet to its left.

Cook-Torrance - masking
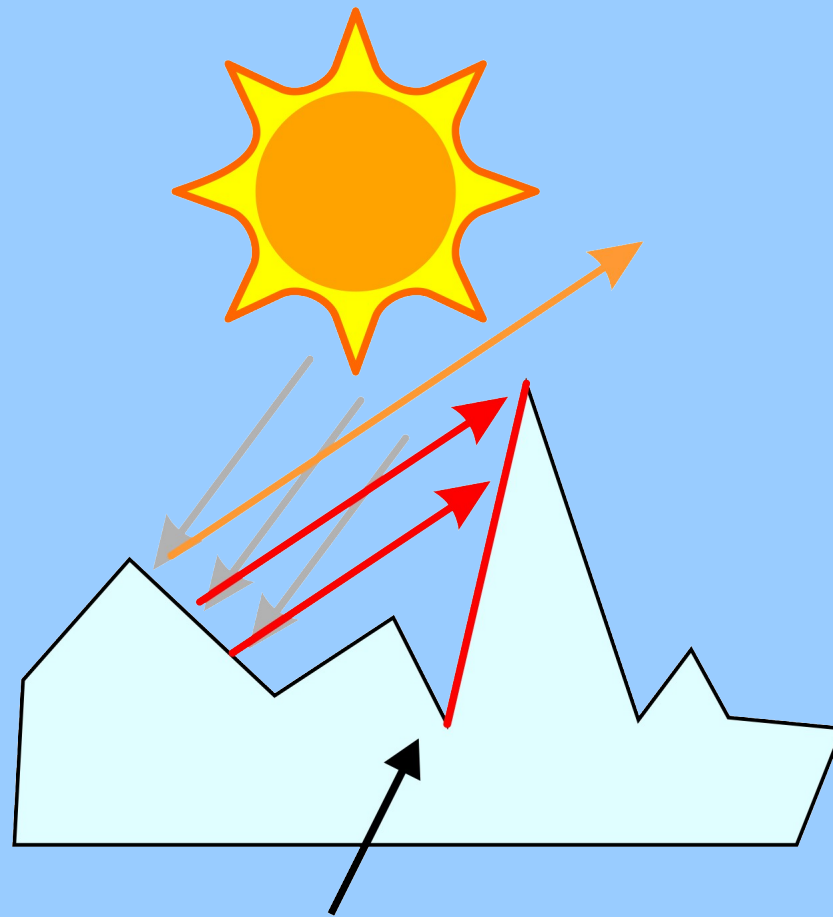
The right microfacet blocks the reflected light
leaving the microfacet to its left, thus **masking** it.

Cook-Torrance

- Formula:

$$I = \frac{F \times D \times G}{\pi (\vec{n} \cdot \vec{l})(\vec{n} \cdot \vec{v})}$$

- Three terms define the appearance:

  - Geometric term *G*

  - Fresnel term *F*

  - Roughness term *D*

Cook-Torrance
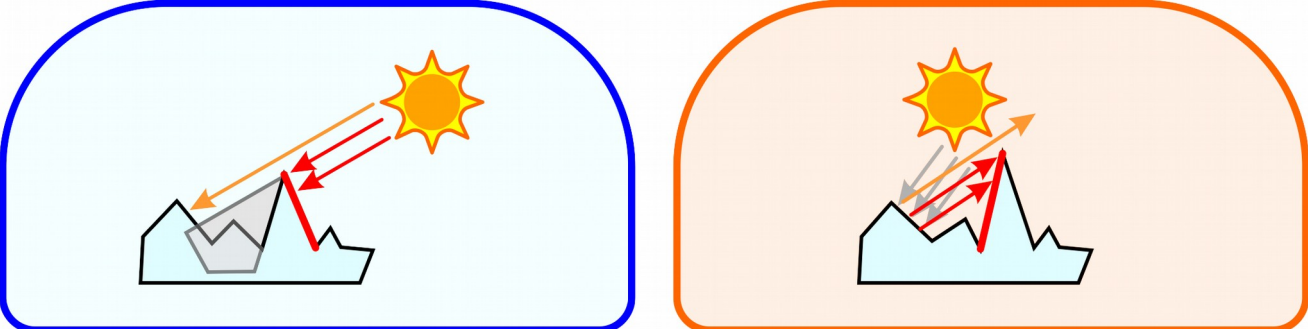
- Geometric term:
    - Takes *shadowing* and *masking* into account.

$$G = min\left(1, \frac{2(\vec{n}\cdot\vec{h})(\vec{n}\cdot\vec{v})}{(\vec{v}\cdot\vec{h})}, \frac{2(\vec{n}\cdot\vec{h})(\vec{n}\cdot\vec{l})}{(\vec{v}\cdot\vec{h})}\right)$$

Light reflected
without
interference

Shadowing
(how much incoming
light is blocked)

Masking
(how much reflected
light is blocked)

Cook-Torrance

- Fresnel term:

    - Controls the amount of reflected light.

    - Determines reflection of microfacets, and how metallic the surface appears.

    - Replaces specular light of other models. $\Rightarrow$ More realistic.

    $$F = \frac{1}{2} \frac{(g-c)^2}{(g+c)^2} \left(1 + \frac{(c(g+c)-1)^2}{(c(g+c)+1)^2} \right)$$

    where

    $$c = \cos(\theta) = (\vec{v} \cdot \vec{h})$$
    $$g = \sqrt{\eta^2 + c^2 - 1}$$

    - Usually implemented by Schlick's approximation.

    $$F = F_0 + (1 - (\vec{h} \cdot \vec{v}))^5 \times (1 - F_0)$$

    where

    F_0 - reflectance at normal incidence

Cook-Torrance

- Roughness term:

  - Roughness depends on the distribution of the microfacets' slopes.

  - Calculation is based on a distribution function.

  - For instance the Beckmann distribution:

$$D = \frac{e^{-\left(\frac{\tan\alpha}{m}\right)^2}}{m^2 \cos^4 \alpha}$$

- Application:

  - Various materials, but especially metals.

Cook-Torrance - Parameters

Parameter	Description	Notes
distributionParameter	Roughness factor for the distribution function.	In sources usually named *c*.
roughness	Roughness factor for the roughness term.	
reflectionCoefficient	Factor for the Fresnel term.	In sources usually named *R0*.

Gooch
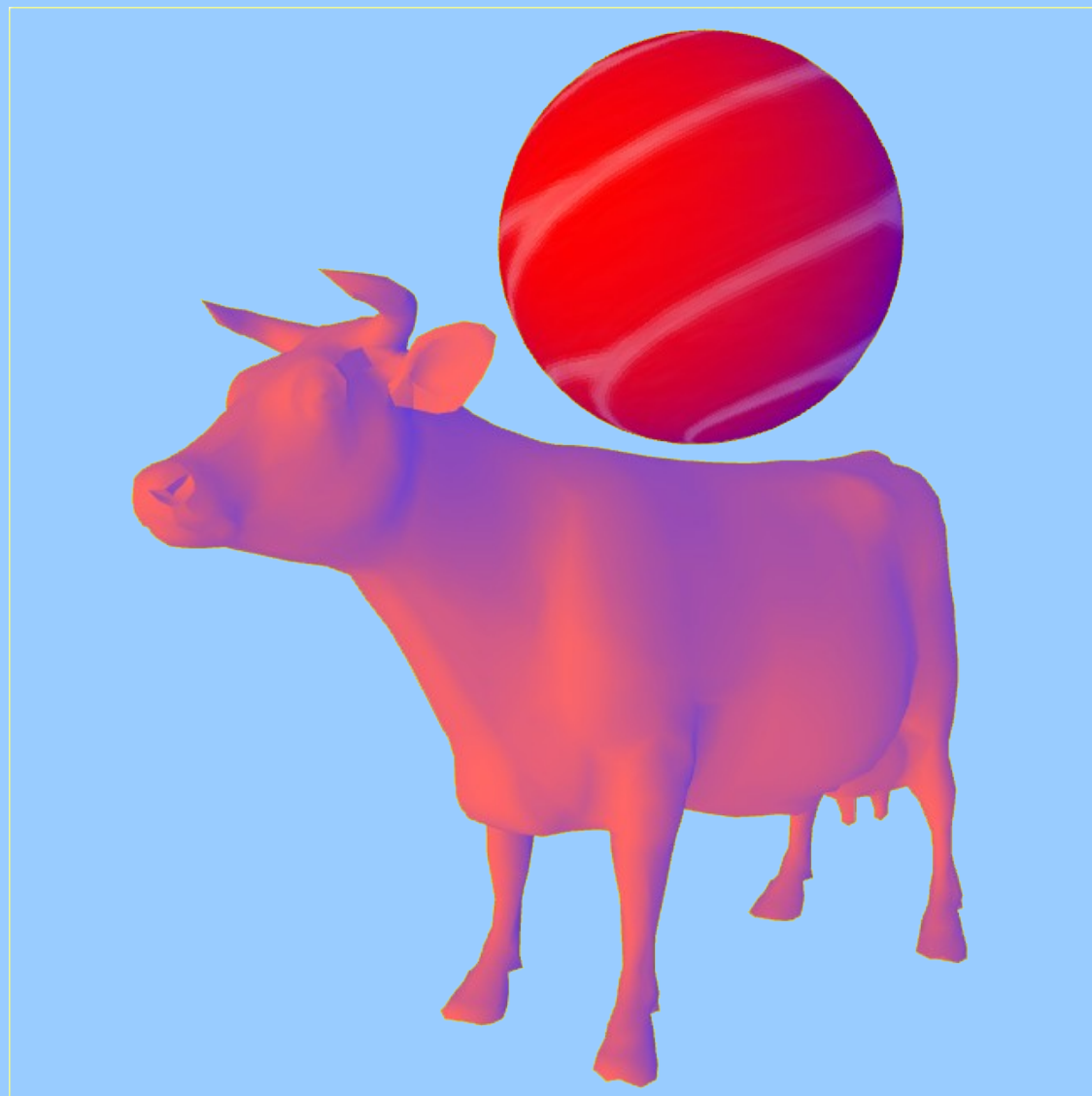
Gooch

- Introduced in 1998 by Amy Gooch, Bruce Gooch, Peter Shirley and Elaine Cohen.

- A lighting model for *non-photorealistic rendering* (*NPR*).

- Goal:
  - Articulate contours by means of colors.

- Setup:
  - Usually only one light source.
  - The paper describes further configurations, for instance:
    - A second light source with special properties.
    - Additional drawing of the object edges as lines.

Gooch

- Formula:

$$coolComponent = coolColor + (coolMixValue * surfaceColor)$$

$$warmComponent = warmColor + (warmMixValue * surfaceColor)$$

$$finalColor = \left(\frac{1 + \vec{l} \cdot \vec{n}}{2}\right) * coolComponent + \left(1 - \frac{1 + \vec{l} \cdot \vec{n}}{2}\right) * warmComponent$$

Gooch - Parameters

Parameter	Description	Notes
surfaceColor	Basic surface color.	
coolColor	Color for "cold" areas.	
warmColor	Color for "warm" areas.	
coolMixValue	Determines how much of the surface color should be mixed into the cool color.	
warmMixValue	Determines how much of the surface color should be mixed into the warm color.	

Lommel-Seeliger
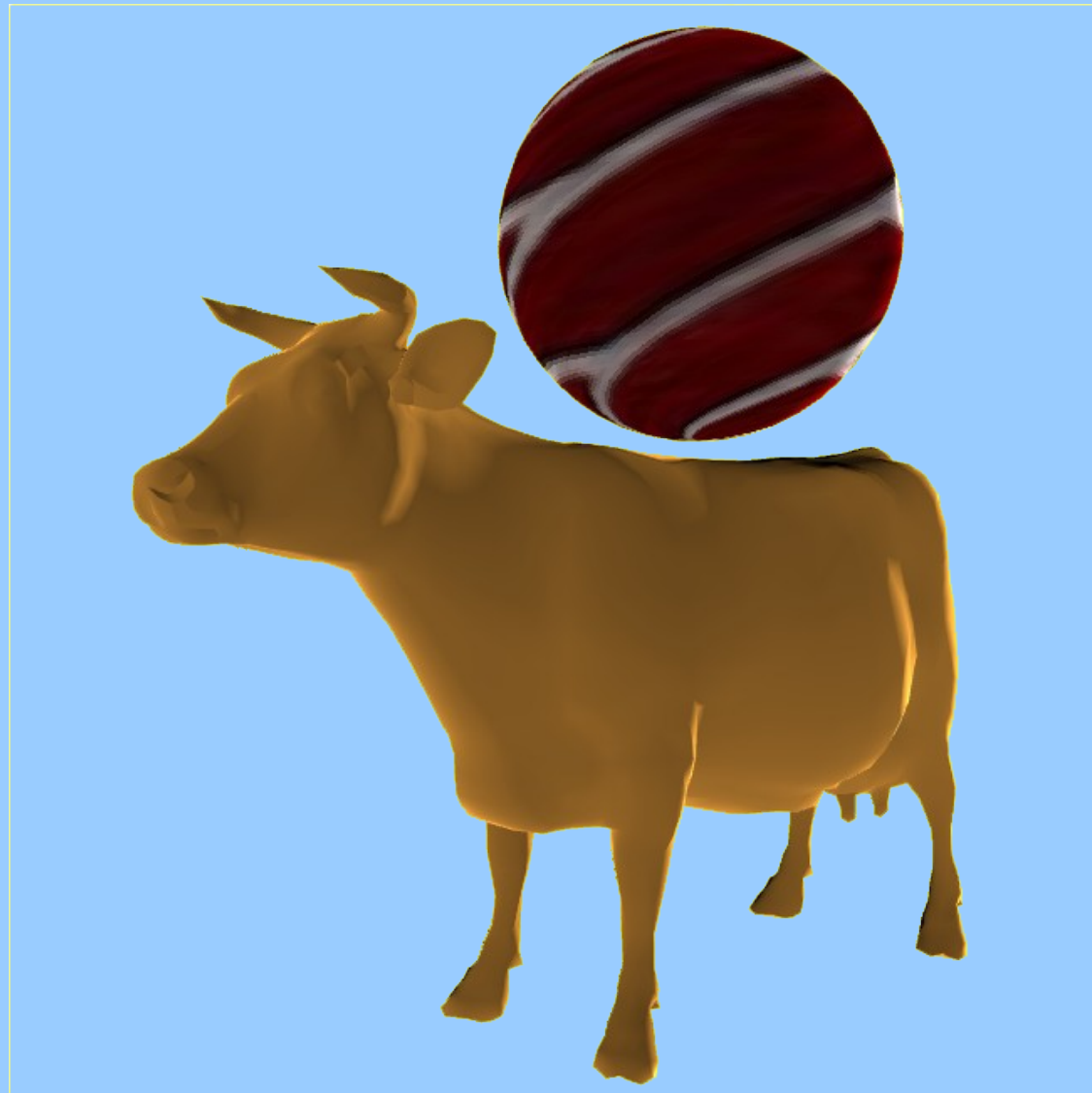
Lommel-Seeliger

- Diffuse model.

- Based on the Hapke function.

  - But avoids the more complex term.

- Rough and dusty look.

- Formula:

$$I_{diff} = \frac{\vec{l} \cdot \vec{n}}{(\vec{l} \cdot \vec{n}) + (\vec{v} \cdot \vec{n})}$$

- Applications:

  - Dusty surfaces, for instance, the moon.

Lommel-Seeliger (extended)

- Extension:
  - No general term found for configuring the roughness.
  - Thus: own experiments on parametrization.
  - Two insights called up:
    - The Minnaert model uses a roughness parameter as an exponent.
    - The empirical approximation for the Fresnel term [fernando2003cg] allows for good configuration, also using an exponent.

    $\Rightarrow$ Thus: Reproduction of the empirical approximation formula to extend the Lommel-Seeliger model.

Lommel-Seeliger (extended)

- Resulting formula:

$$I_{diff} = diffuseBias + diffuseScale * \left(\frac{\vec{l} \cdot \vec{n}}{(\vec{l} \cdot \vec{n}) + (\vec{v} \cdot \vec{n})} \right)^{diffuseExponent}$$

- To allow comparison, the empirical approximation formula:

$$F = bias + scale \times (1 + \vec{l} \cdot \vec{n})^{power}$$

Lommel-Seeliger (extended)

Parameter	Description	Notes
diffuseBias	Basic brightness/darkness.	Very sensitive, as it is an additive term.
diffuseScale	Increases or decreases the intensity of the lit area. The larger the value, the more intense.	
diffuseExponent	Decreases or increases the size of the area to be lit. The larger the value, the smaller the area.	

- Notes:
  - An experimental approach is necessary.
    - The descriptions above are only observations; they are not mathematically proven.
    - Actually all three parameters influence the light intensity.
  - Especially combinations of negative and positive values yield interesting results.

Lommel-Seeliger (extended)

- Advantages of the **extended** model:

  - Wide range of effects possible:

    - very rough (like clothes)

    - approximately metallic (if using specular term).

- Disadvantages of the **extended** model:

  - Difficult configuration, as parameters are very sensible and not intuitive.

  - Intense bright and dark areas show up unexpectedly:

    - Reason: Little mathematical experience and insight of the author.

  - Very few configurations achieve useful results.

Minnaert
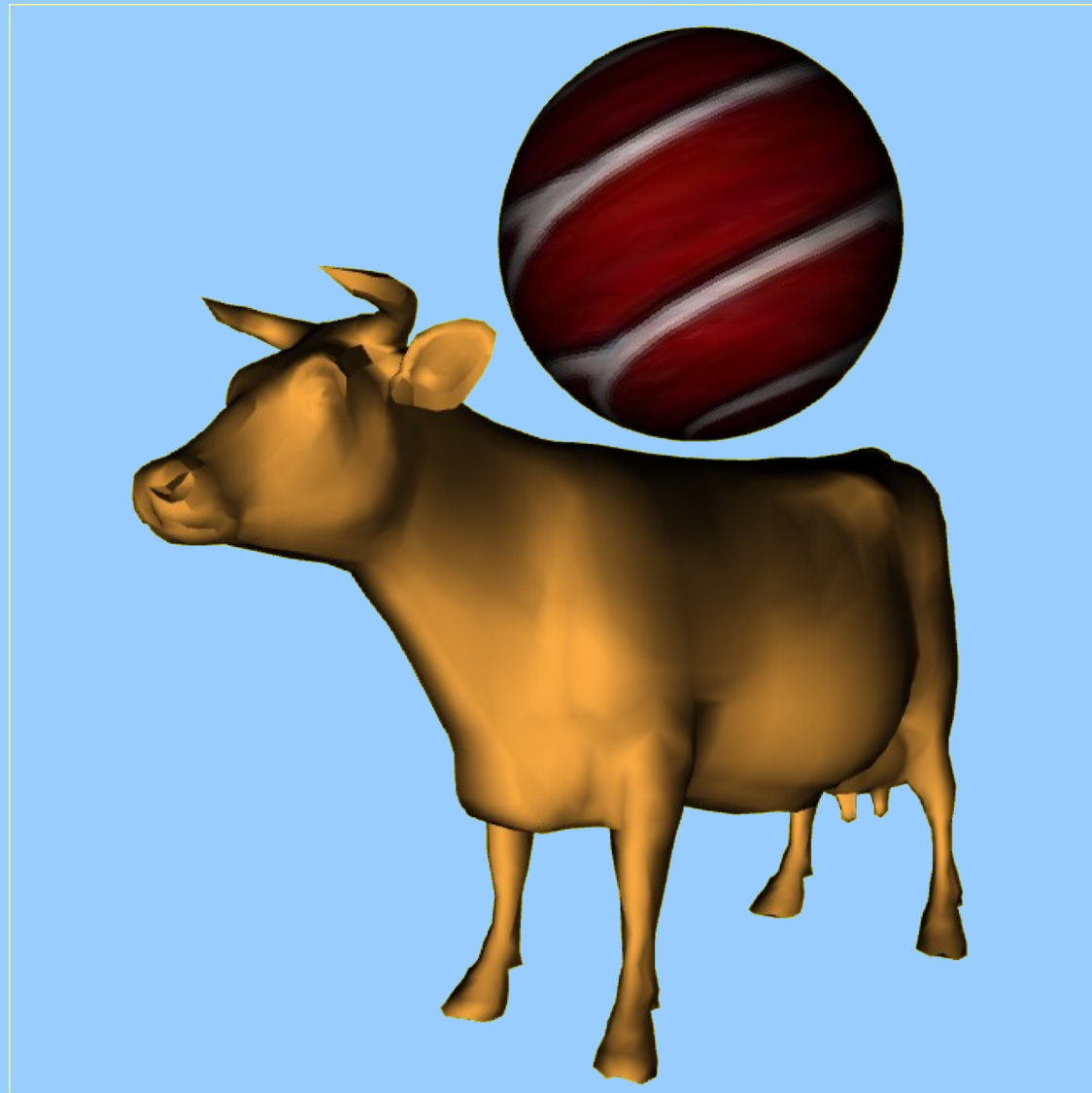
Minnaert

- Diffuse model.

- Actually developed for other purpose.

- Uses a roughness factor.

- Characteristic dark stripes around edges.

- Applications:

  – Simulate clothes, especially **velvet**.

- Formula:

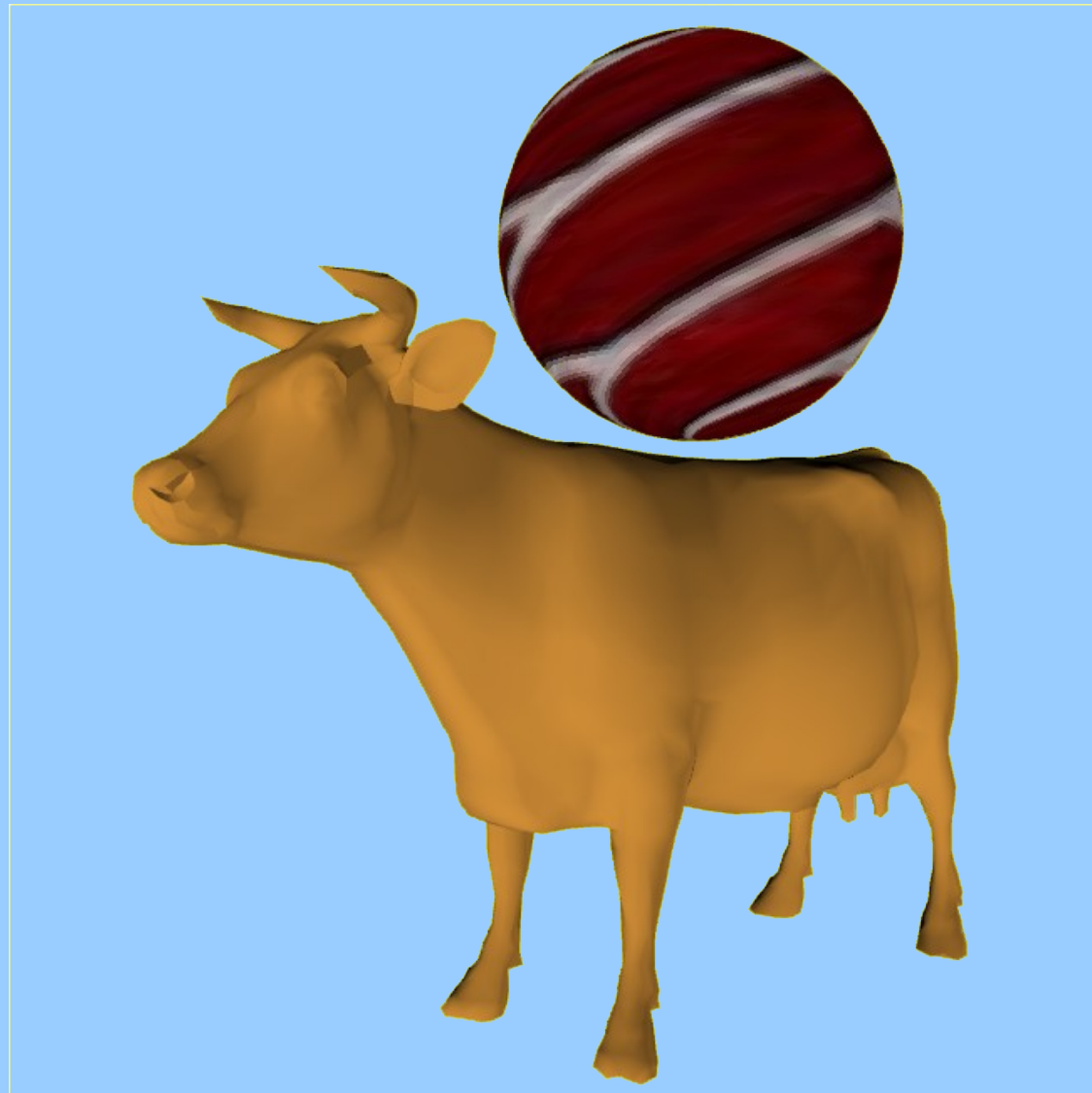$$I_{diff} = ((\vec{n} \cdot \vec{l})(\vec{n} \cdot \vec{v}))^{m-1}$$
where

$$m \text{ - roughness}$$

Minnaert - Parameters

Parameter	Description	Notes
diffuse color (D)	Diffuse surface color.	
roughness (m)	Roughness factor.	

Oren-Nayar

Oren-Nayar

- A purely diffuse model (no specular component).

- Extends the Lambert concept.

- Adds **roughness**:
  - Surface is comprised of tiny microfacets.
  - Roughness is based on normal distribution of the microfacets:
    - High distribution $\Rightarrow$ great differences in the directions.

- Results are usually "flatter" than the Lambert model.

- Applications:
  - **Rough** surfaces.
    - For example: moon, clay, dirt, clothes.

Oren-Nayar

- Formula:

$$I_0 =$$

$$D * (N \bullet L) * (A + B * \sin(\alpha) * \tan(\beta) * MAX(0, \cos(C))) * I_1$$

$$A = 1 - 0.5 \frac{\sigma^2}{\sigma^2 + 0.33}$$

$$B = 0.45 \frac{\sigma^2}{\sigma^2 + 0.09}$$

$$\alpha = MAX(\text{acos}(N \bullet L), \text{acos}(N \bullet V))$$

$$\beta = MIN(\text{acos}(N \bullet L), \text{acos}(N \bullet V))$$

$$C = \text{the azimuth angle between the}$$

light vector and the view vector

Oren-Nayar - Parameters

| Parameter	Description	Notes
diffuse color (D)	Diffuse surface color.	
roughness (σ)	Roughness term.	

Phong
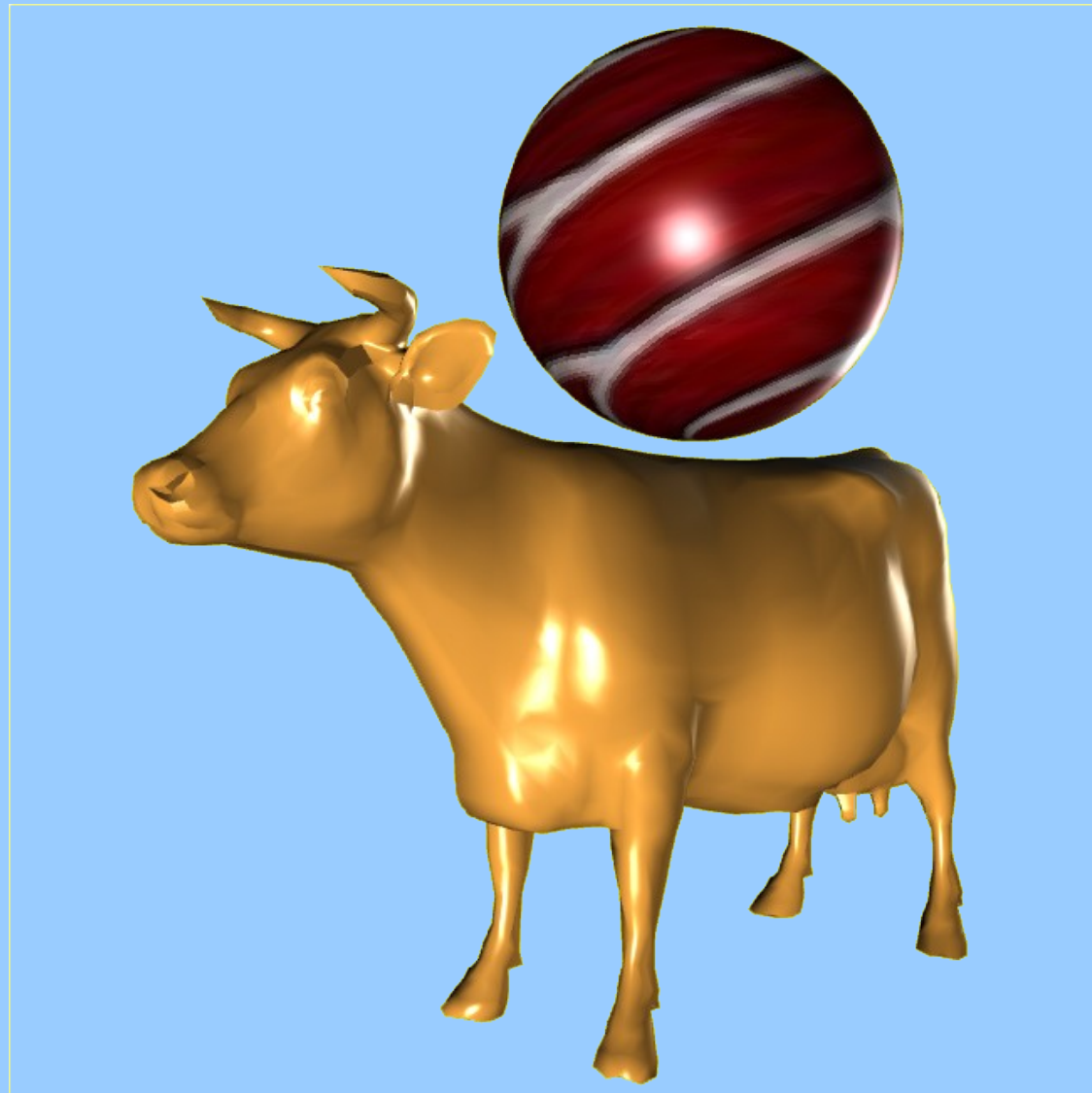
Phong

- One of the oldest models.

- Important, as it introduces a specular reflection.

  - Calculated via a reflection vector.

- Rather not physically plausible.

- But easy to implement.

- Application:

  - Basically all materials.

  - But especially plastics.

Phong - Parameters

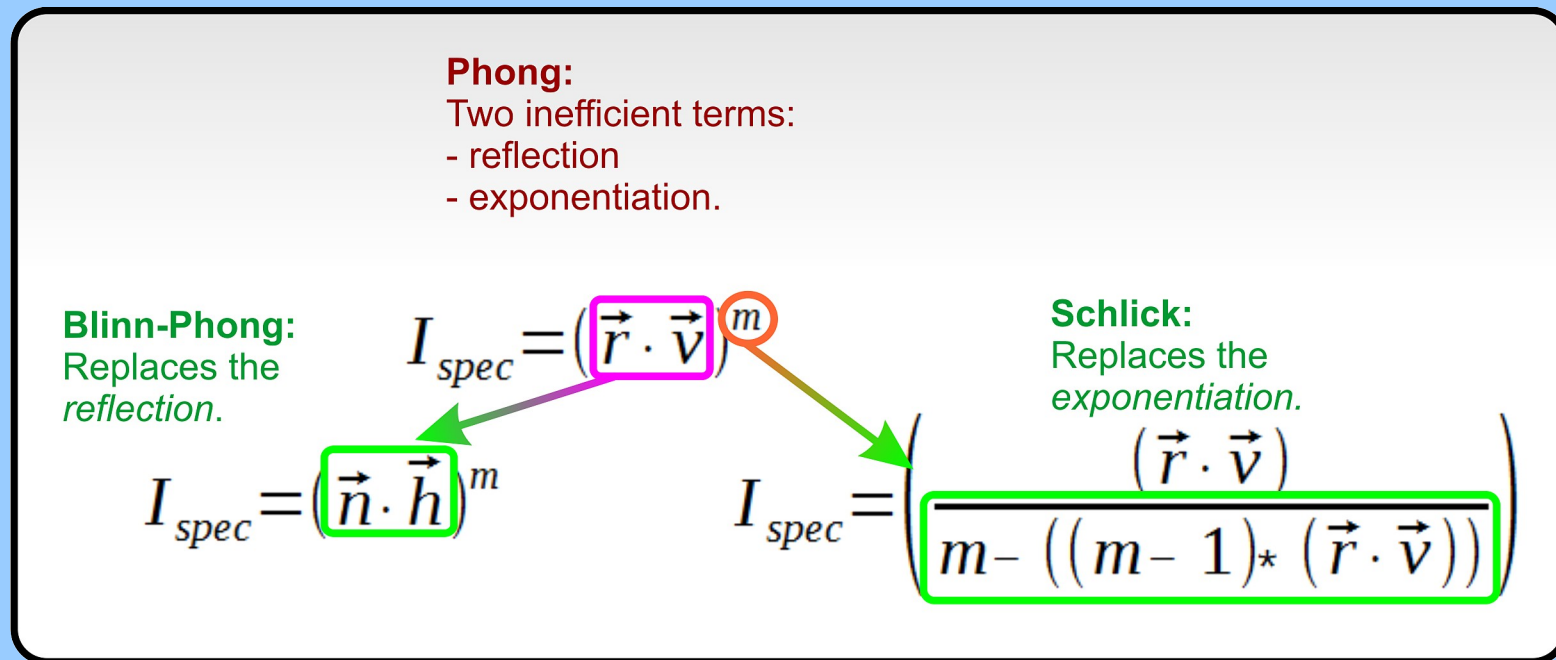| Parameter	Description	Notes
ambient color	Color from the surrounding.	
diffuse color	Diffuse material color.	
emissive color	Color emitted from the object.	
specular color	Specular material color.	
shininess	Intensity of the surface's shining.	

Blinn-Phong
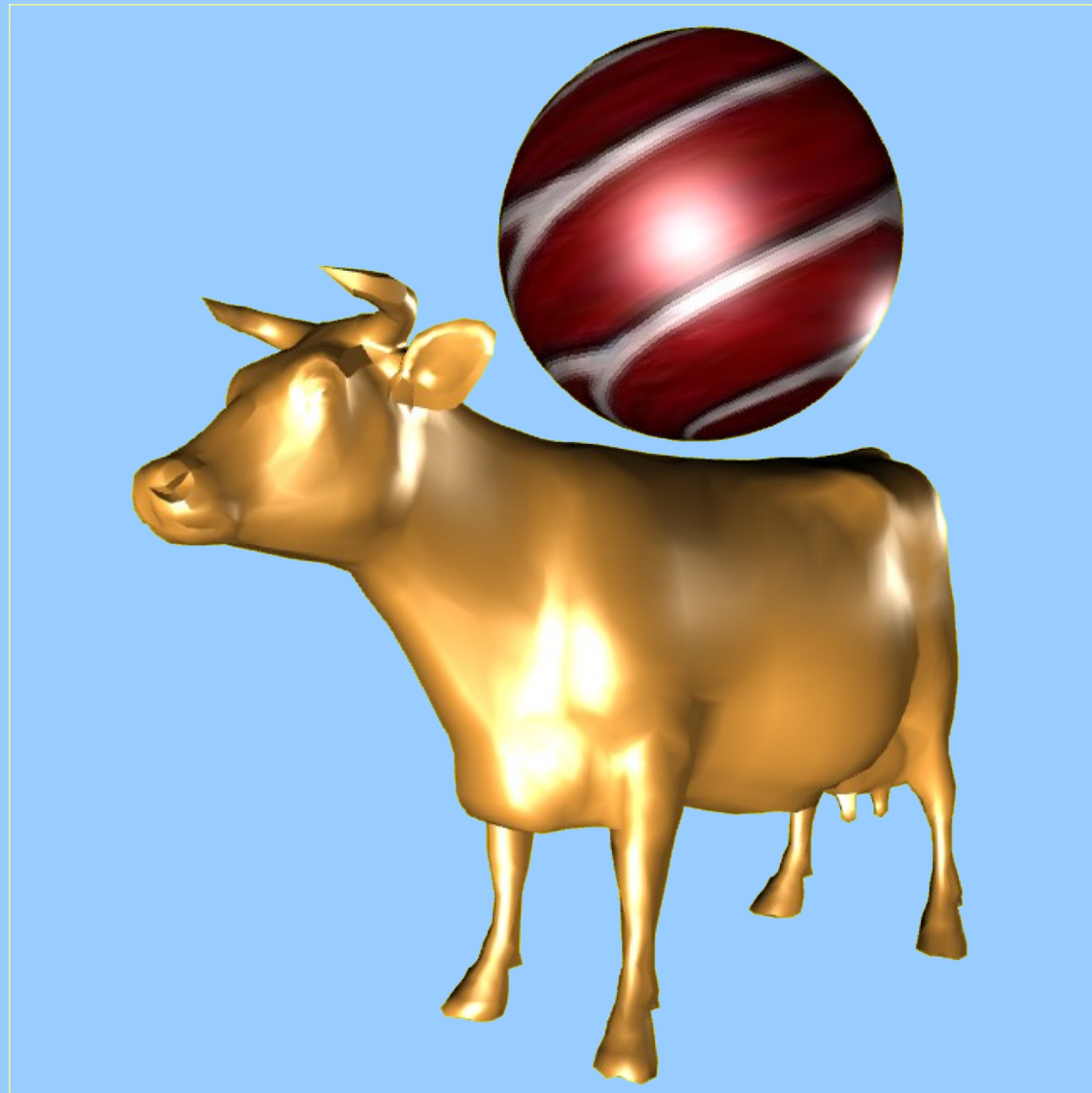
Blinn-Phong

- Derived from the Phong lighting model.

- Uses a half vector to increase efficiency.

- Can yield more realistic results.

- Applications:

    - Generally all materials.

    - But especially plastics.

Schlick Specular Model
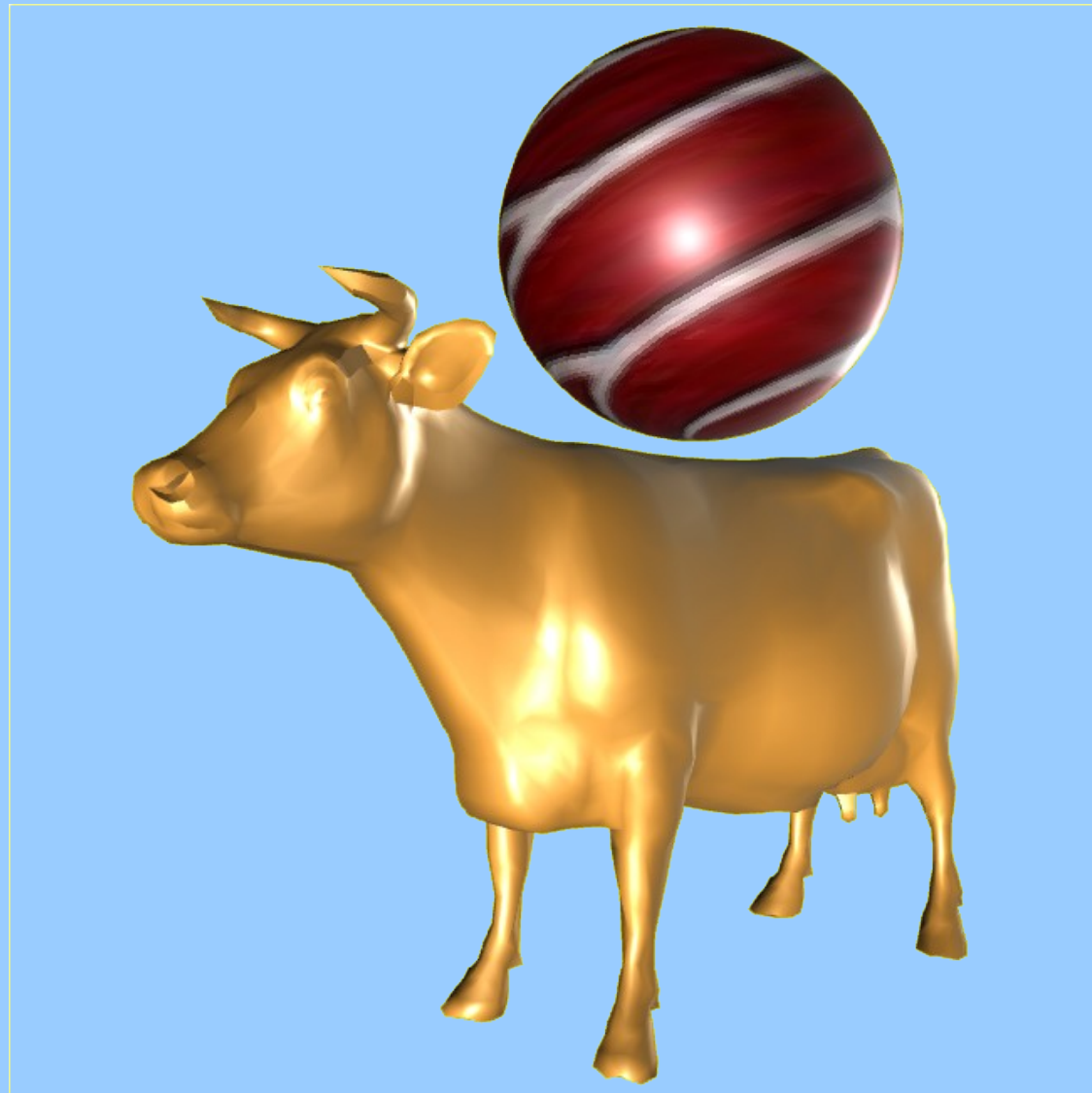
Schlick Specular Model

- Variation of the Phong lighting model.

- Goal: more efficient calculation.
  - Replaces the expensive exponentiation.

Phong:
Two inefficient terms:
- reflection
- exponentiation.

$$I_{spec} = (\vec{r} \cdot \vec{v})^m$$

Blinn-Phong:
Replaces the *reflection*.

$$I_{spec} = (\vec{n} \cdot \vec{h})^m$$

Schlick:
Replaces the *exponentiation*.

$$I_{spec} = \left(\frac{(\vec{r} \cdot \vec{v})}{m - ((m-1) * (\vec{r} \cdot \vec{v}))} \right)$$

Schlick Specular Model

- Problem:

    - Exponentiation with non-integer exponent is expensive.

- Solution:

    - Replace the exponentiation with an approximating term:

$$I_{spec} = \left(\frac{(\vec{r} \cdot \vec{v})}{m - ((m-1) * (\vec{r} \cdot \vec{v}))} \right)$$

Schlick Specular Model

- Result:
    - Reflection is similar to Phong lighting model.
        - Shape of specularly lit areas are almost identical.
    - Though the light is more scattered.
        - Objects are "brighter".

Gaussian Specular Model
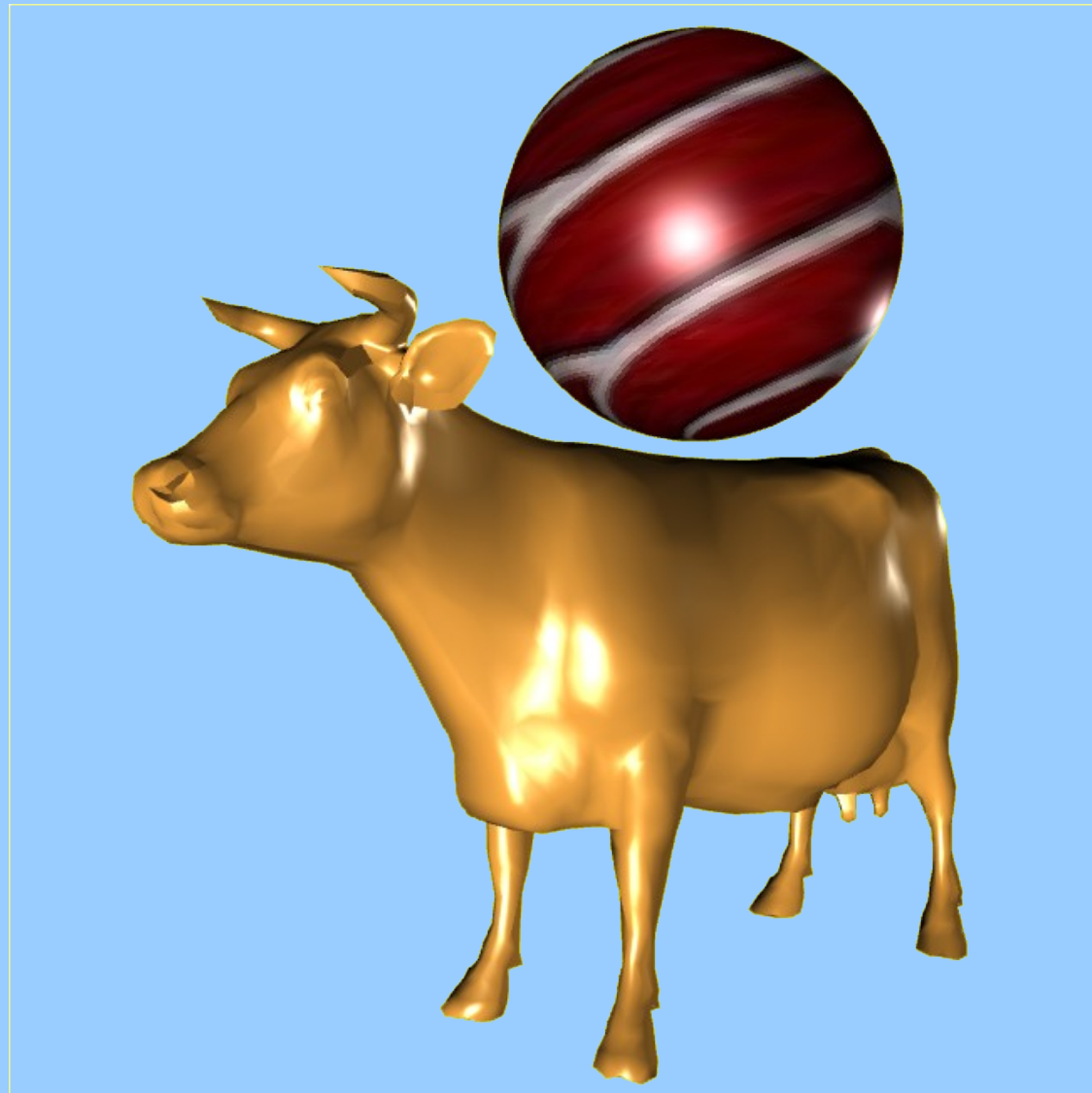
Gaussian Specular Model

- Variation of the Phong/Blinn-Phong lighting model.

- Goal:

  - Simulation of microfacets.

- Problem:

  - The Phong model uses a Lambertian diffuse term.

  - This term does not take microfacets into account.

    $\Rightarrow$ Rather unrealistic.

- Solution:

  - Microfacets are being simulated by the normal distribution (Gaussian distribution) function.

Gaussian Specular Model

- Formula:

$$I_{diff} = e^{-\left(\frac{\sigma}{m}\right)^2}$$

$$\sigma = acos(\vec{h} \cdot \vec{n})$$

m - shininess (roughness)

- Attention:

  - The term "shininess" is being used in compliance to the Phong terminology.

  - Actually its meaning is exactly the opposite:
    - The higher the shininess, the rougher the surface.

 $\Rightarrow$ The parameter **m is a roughness** factor.

Gaussian Specular Model

- Result:
  - Objects look less like plastic.

- Advantage:
  - More realistic.

- Disadvantage:
  - Less efficient than the default Phong/Blinn-Phong model.
    - Especially because of the *arcus cosinus* function.

Summary – Phong Variations

| Model	Characteristics
Phong	Standard model with three issues: - inefficient reflection formula - inefficient exponentiation formula - unrealistic (Lambert) diffuse term.
Blinn-Phong	More efficient by replacing the reflection formula.
Schlick Specular Model	More realistic by replacing the exponentiation.
Gaussian Specular Model	More realistic, but less efficient, by modeling microfacets.

Strauss
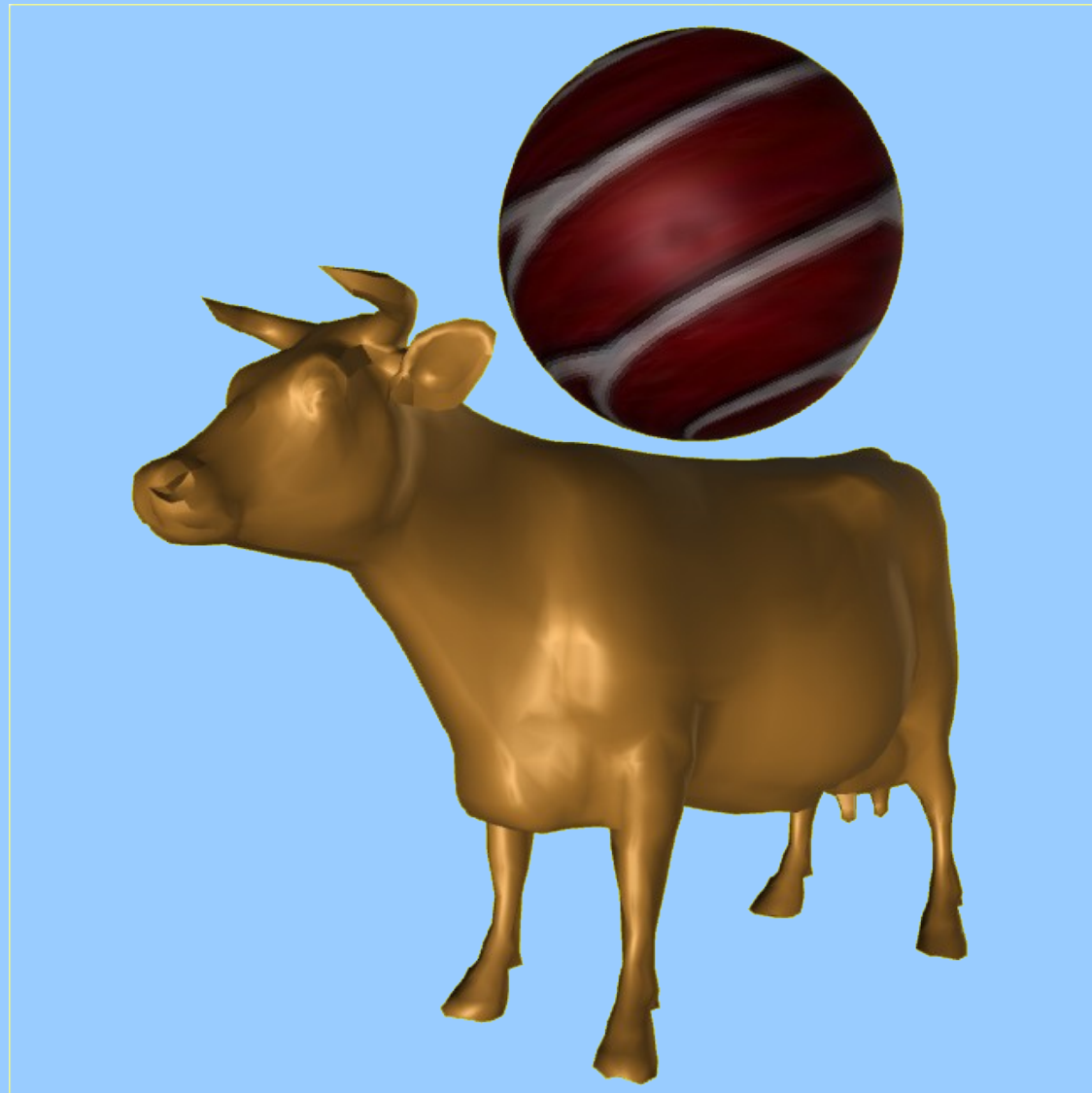
Strauss

- Goals:
    - Simple to understand and to handle parameters.
    - At the same time simulation of a broad range of materials.
- Combines parts of existing lighting models.
- Application:
    - Various materials.
    - Especially well-suited for metals.

Strauss - Parameters

Parameter	Description	Notes
metalness m	How metallic does the surface look?	Range: [0, 1].
smoothness s	How smooth is the surface?	Range: [0, 1].
transparency t	Simulates the effect of lighting.	Not identical to the RGBA alpha transparency, but should be equal. Range: [0, 1].
surface color c	(Diffuse) RGB surface color.	Usually the only color to be set.
index of refraction n	Index of refraction.	Not used in the shader, but part of the ray-tracing or global illumination.
fresnel constant Kf	Affects the Fresnel term.	Usually not a parameter, but a constant with $Kf = 1.12$.
shadow constant Ks	Affects the simulation of shadowing.	Usually not a parameter, but a constant with $Ks = 1.01$.
off-specular peak k	Additional value for very rough surfaces.	Usually not a parameter, but a constant with $k = 0.1$.
highlight base color $C1$	Base color for the specular term.	Usually constant white color.

Ward (anisotropic)
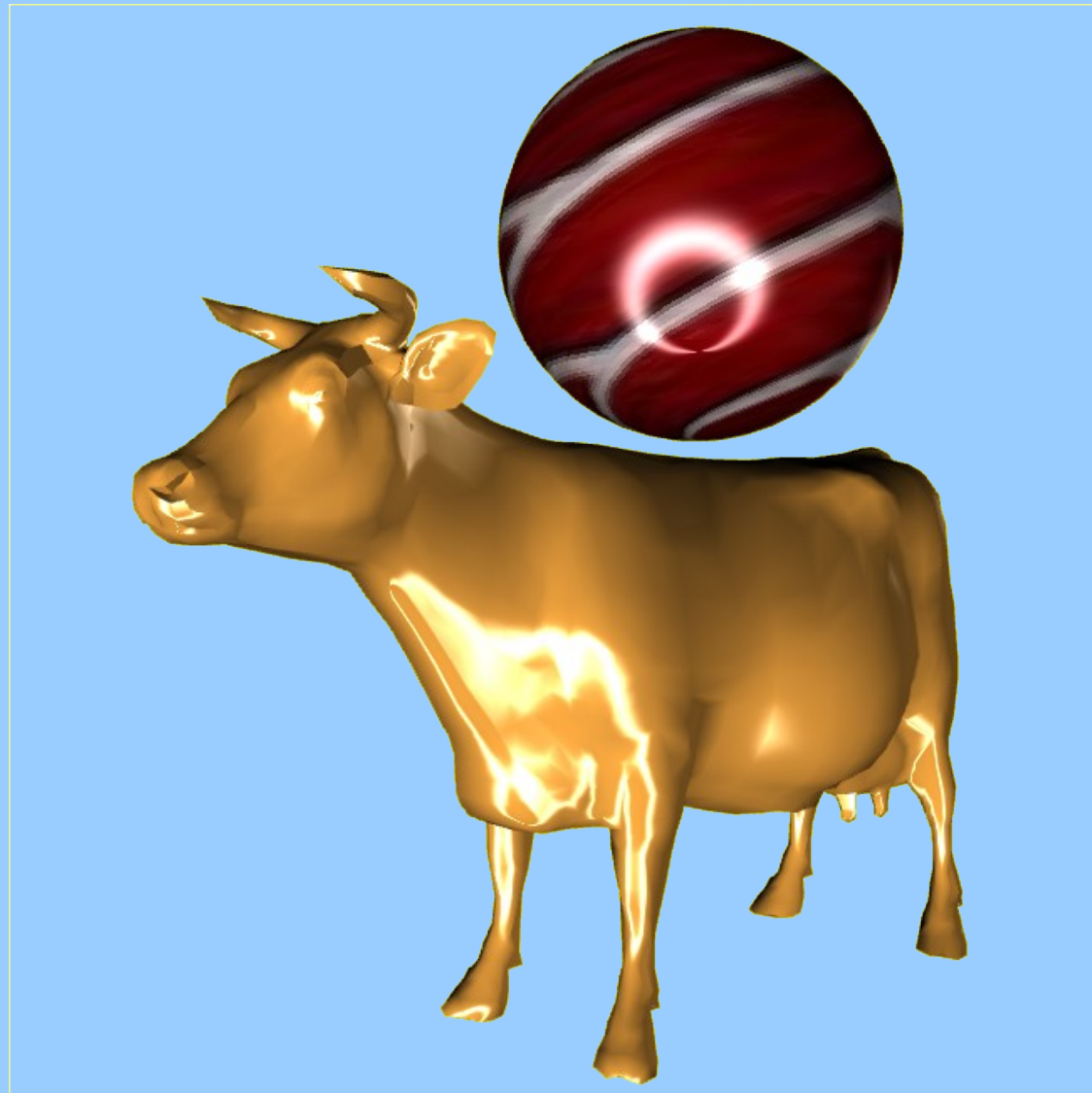
Ward (anisotropic)

- Empirical model: based on observations.

  - Equations created from retrieved data.

- Rather physically plausible.

- Two variations exist:

  - isotropic

  - anisotropic.

- Anisotropy is achieved through two orthogonal roughness coefficients (X- and Y-direction).

- Applications:

  - Metals, especially brushed ones.

Ward (anisotropic)

- Formula:

$$I_o = \left(D + S * \frac{e^{-\tan^2\gamma\left(\frac{\cos^2\phi}{\sigma_x^2} + \frac{\sin^2\phi}{\sigma_y^2}\right)}}{2\pi\sigma_x\sigma_y\sqrt{(N \bullet L)(N \bullet V)}} \right) I_i(N \bullet L)$$

$$\gamma = \text{acos}(N \bullet H)$$

ϕ = the azimuth angle of the light vector on the tangent plane

- where

  - D – diffuse surface color

  - S – specular surface color

  - σx – roughness coefficient along the X-axis

  - σy – roughness coefficient along the Y-axis

Ward - Parameters

Parameter	Description	Notes
direction	Material direction as 3D vector.	
roughness.x	Roughness coefficient along the X-axis.	Responsible for anisotropy.
roughness.y	Roughness coefficient along the Y-axis.	Responsible anisotropy.

Framework – Concept (1/3)

- Various lighting models are offered.

  - Each vertex or fragment shader is implemented in its own file.

  - Its content is read by a PHP script and inserted into the dynamic web page.

- The lighting models can be selected in an HTML form, part of the *Shader Chooser* page.

  - Each model is represented by its own button.

  - Click it sends an unique shader identifier via GET request to the HTML page containing the WebGL canvas.

  - The PHP script mentioned above selects the fragment shader associated the identifier, creating the GLSL code block.
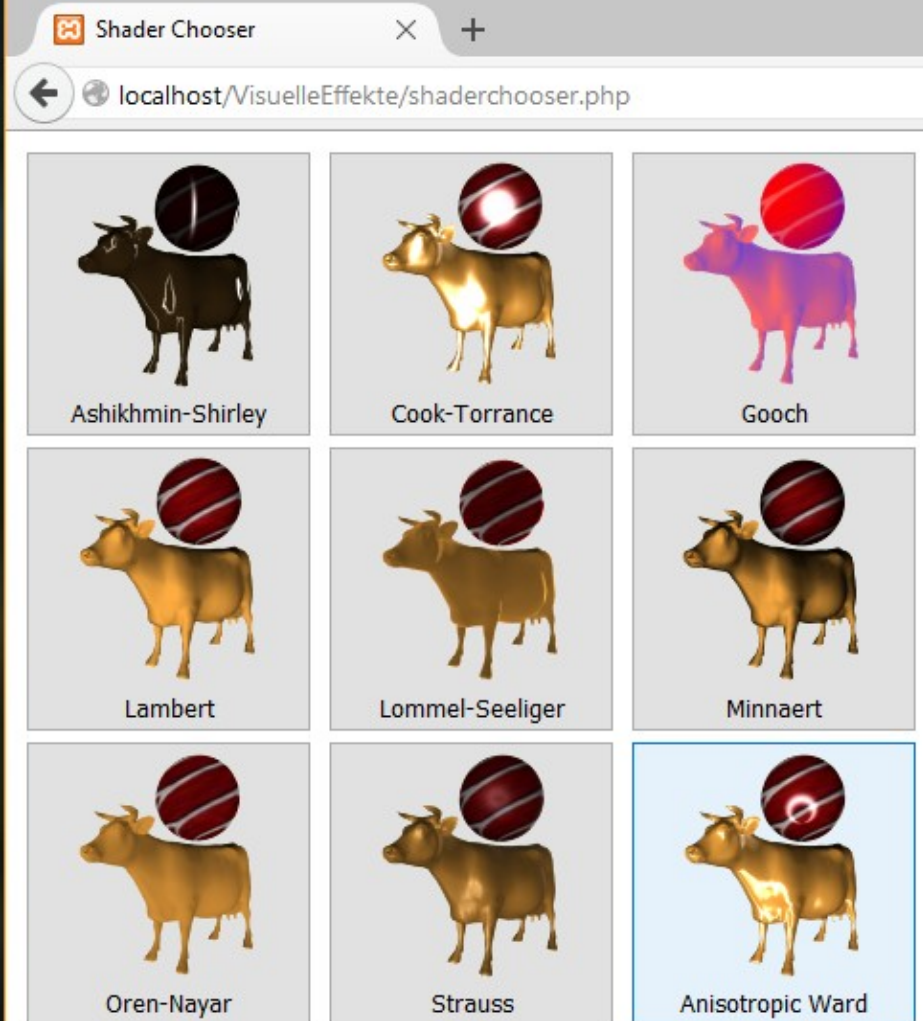
Framework – Concept (2/3)

The *Shader Chooser* form represents each implemented lighting model by an own button. Moving the mouse into such a button displays information regarding the model.

# Framework – Concept (3/3)

- The scene contains 3D objects.

  - They can be selected with a menu structure.

  $\Rightarrow$ Simulates a flat scene graph (actually a "scene list").

- The navigation (rotation) is possible by using a computer mouse.

Framework – Configuration (1/2)

- Each lighting model is associated with a set of dynamically created HTML configuration elements.

  - For example: The Minnaert shader needs a roughness spinner, the Cook-Torrance shader a set of radiobuttons to set the distribution function.

- Shared properties exist.

  - For instance: transparency (slider), Fresnel block.

- Further properties, independent of the lighting model are

  - lights

  - fog.

Framework – Configuration (2/2)
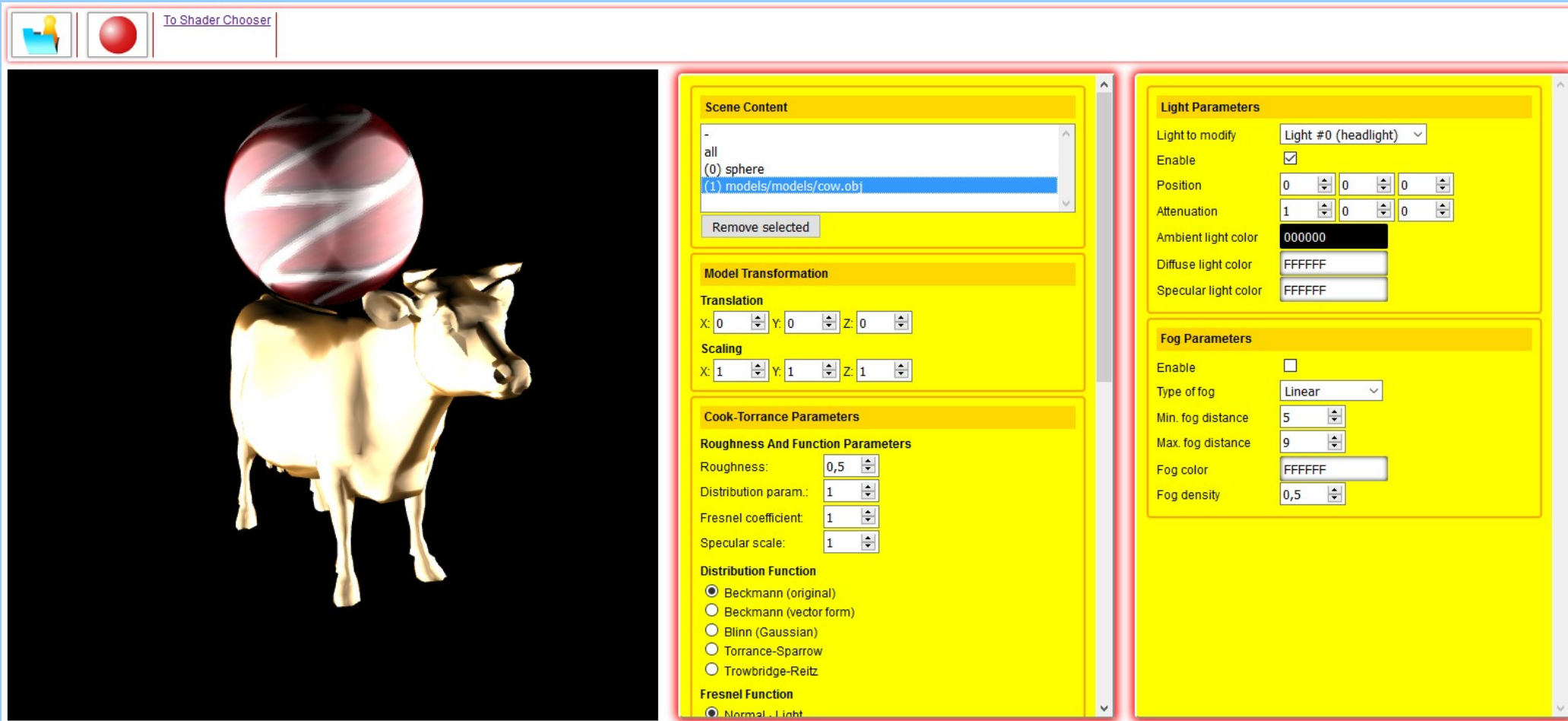
A capture of the configuration screen for the Cook-Torrance shader. The yellow block to the left contains shader-specific properties, as well as the "scene graph" (top of it). The right block contains light and fog parameters.
At the top, the toolbar for inserting external models (OBJ format) and spheres is visible.

Framework – Architecture (1/2)

- The architecture is loosely leaned on the Java 3D specification. [sowizral1997java].

- Thus, a great number of "classes" (types) with clearly defined responsibilities have been defined.

- Primary goals:
  - Extensibility
    - E.g., new shader materials.
    - E.g., abstract "superclass" for geometries.
  - Readability
    - E.g., by utilizing JavaScript files as modules.
    - E.g., by defining one "class" per uniform variable type.

Framework – Architecture (2/2)
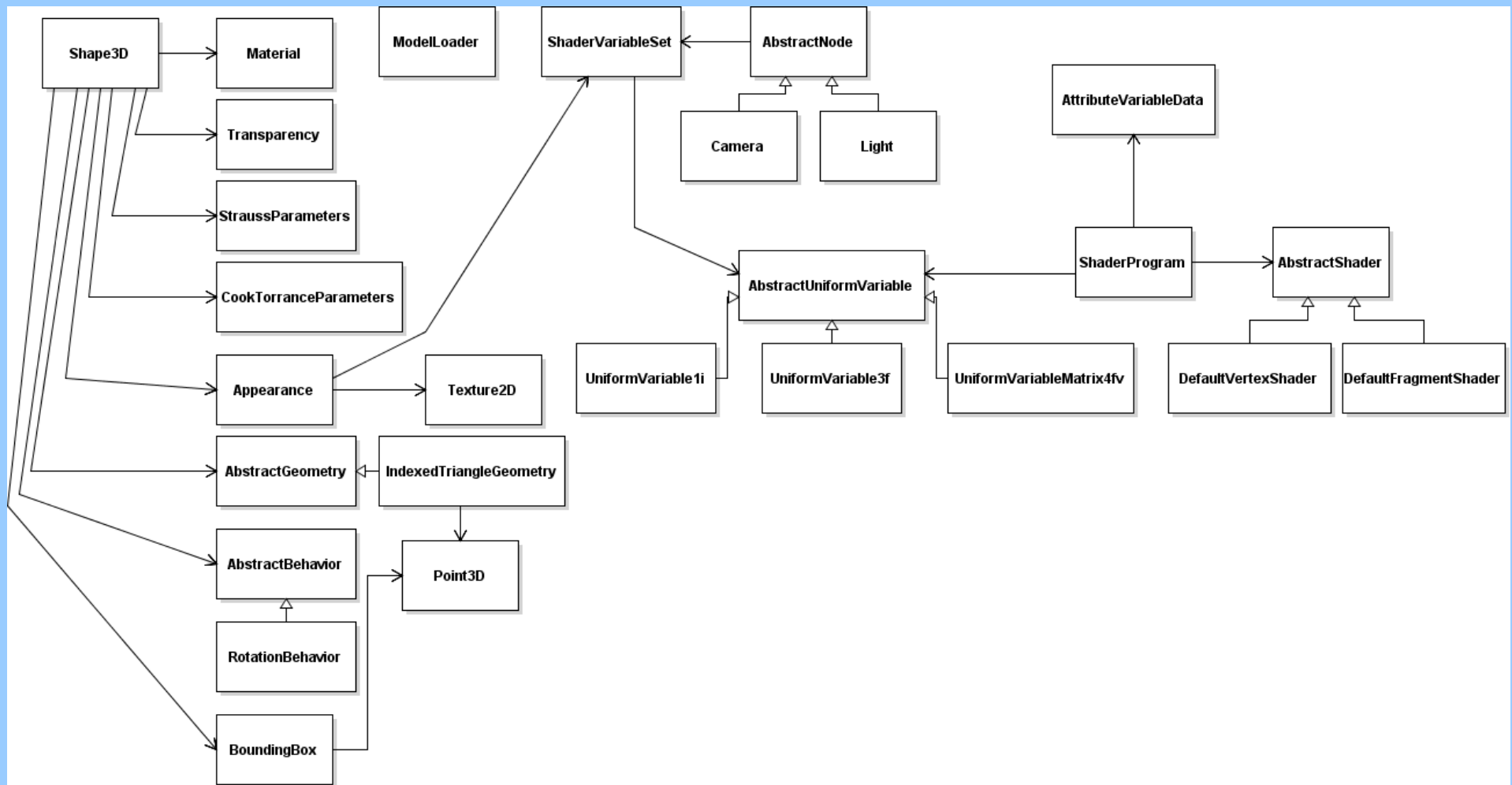
Excerpt from the current system architecture as class UML diagram.

Development Status

Feature	Type	Status
Mouse based navigation	**mandatory**	**Partially implemented.**
Textures	**mandatory**	**Implemented.**
Fresnel effect	**mandatory**	**Implemented.**
Dynamic configuration elements	**mandatory**	**Implemented.**
Model transformations	**mandatory**	**Partially implemented.**
Ashikhmin-Shirley shader	optional	Implemented.
Lafortune shader	optional	Not implemented.
Schlick specular shader	optional	Implemented.
Shader per object	optional	Not implemented.
Insertion of 3D objects	optional	Partially implemented.

Problems (1/3)

- WebGL error messages
  - Setting the light properties generates WebGL exceptions.
  - But changes seem to work none the less.

- Shader appearance
  - Some shaders do not look very realistic and/or correct.
  - Especially the Ashikhmin-Shirley shader is too dark.
    - An implementation working with Java 3D does not share this problem.
  - The Cook-Torrance shader's reflectance is not "metallic" enough compared to the Java 3D version.

Problems (2/3)
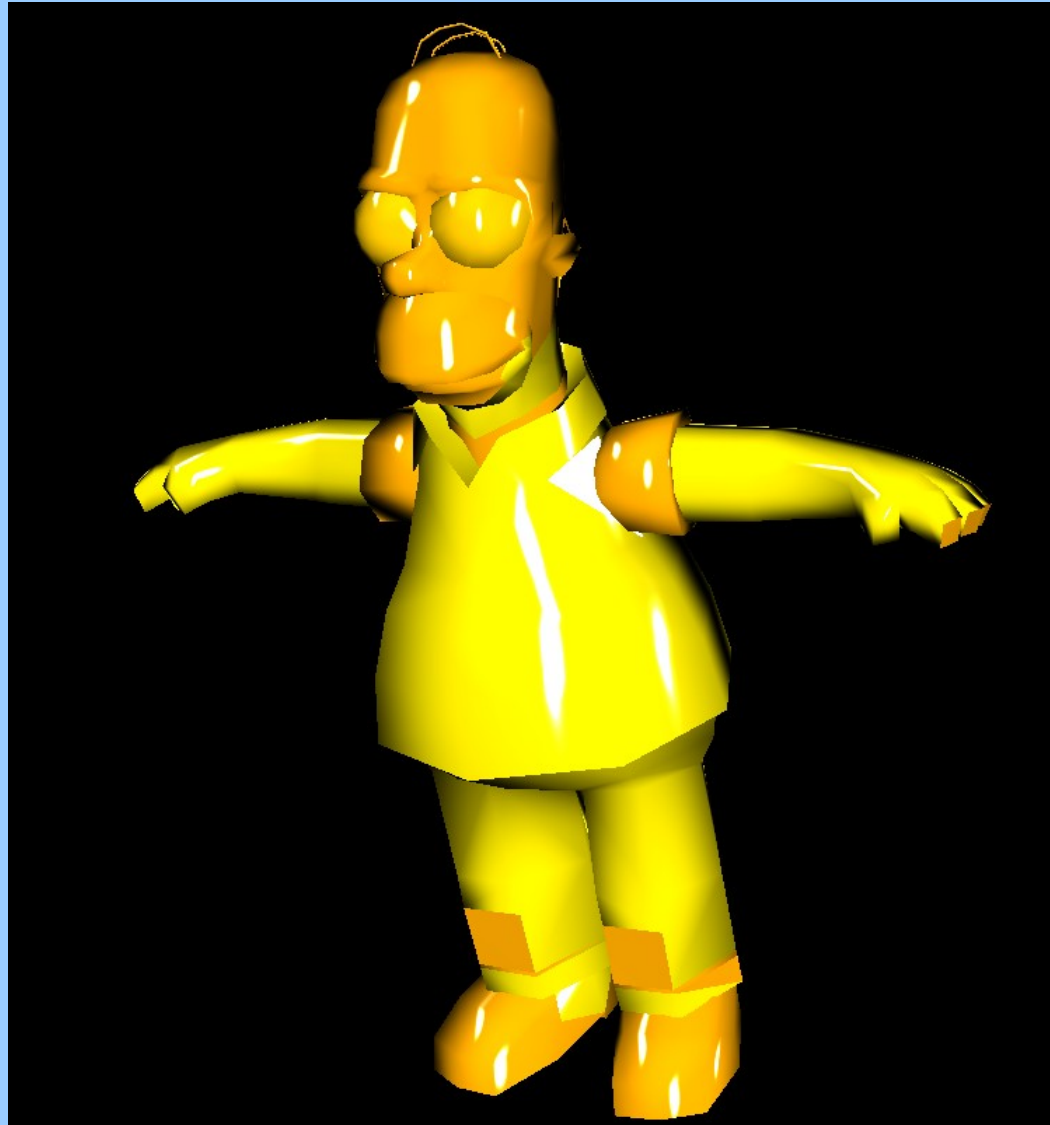
Example for a correct looking Ashikhmin-Shirley result. Taken from an own Java 3D project.

Problems (3/3)

- Fog update problem
  - The fog parameters do not update properly
- Two types of lights missing
  - No headlight implemented.
  - No spotlight implemented.
- Scene object updating problem
  - Inserting a new object or a sphere does not lead to updating the scene without navigation a bit.
- XAMPP
  - Used for dynamic content creation.
  - But obstacle for new users.

Conclusion

- The project is modestly successful.

- Lacks many features.

- At least the implemented shaders can be configured and tested.

- The files can be found on the internet:

  – Main page: https://github.com/KY-ARVR/VisuelleEffekte_SS-2015

  – Project data:
    https://github.com/KY-ARVR/VisuelleEffekte_SS-2015/tree/master/Project/VisuelleEffekte

  – This presentation:
    https://github.com/KY-ARVR/VisuelleEffekte_SS-2015/tree/master/Final_Presentation

Future Work

- Possible extensions may be:
    - Interactive picking by mouse.
    - Shadow calculation.
    - Efficiency comparisons between shaders.
    - Further shaders, for instance:
        - Heidrich-Seidel
        - Lafortune.

Bibliography (1/2)

[3drender2001], Jeremy Birn, Fresnel Effect, http://www.3drender.com/glossary/fresneleffect.htm

[cook1981reflectance] Robert L. Cook, Kenneth E. Torrance, A Reflectance Model for Computer Graphics, 1981, http://www.cs.columbia.edu/~belhumeur/courses/appearance/cook-torrance.pdf

[dempski2005advanced] Dempski, Kelly and Viale, Emmanuel, Advanced Lighting and Materials with Shaders, 2005

[engel2008programming] Wolfgang F. Engel, Jack Hoxley, Ralf Kornmann, Niko Suni, Jason Zink, Programming Vertex, Geometry, and Pixel Shaders, 2008

Bibliography (2/2)

[fernando2003cg] Randima Fernando, Mark J. Kilgard, The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics, 2003, http://http.developer.nvidia.com/CgTutorial/cg_tutorial_chapter07.html

[kornmann2011d3dbook] Ralf Kornmann, D3DBook, 2011, http://content.gpwiki.org/D3DBook:Table_Of_Contents

[shlyaev2014] Sergey Shlyaev, Diffuse Shaders, 2014, http://www.shlyaev.com/rnd/37-cpp-category/60-diffuseshaders

[sowizral1997java] Kevin Sowizral, Kevin Rushforth, Henry Sowizral, The Java 3D API Specification, 1997