Computer Graphics 1

Ludwig-Maximilians-Universität München
Summer semester 2020

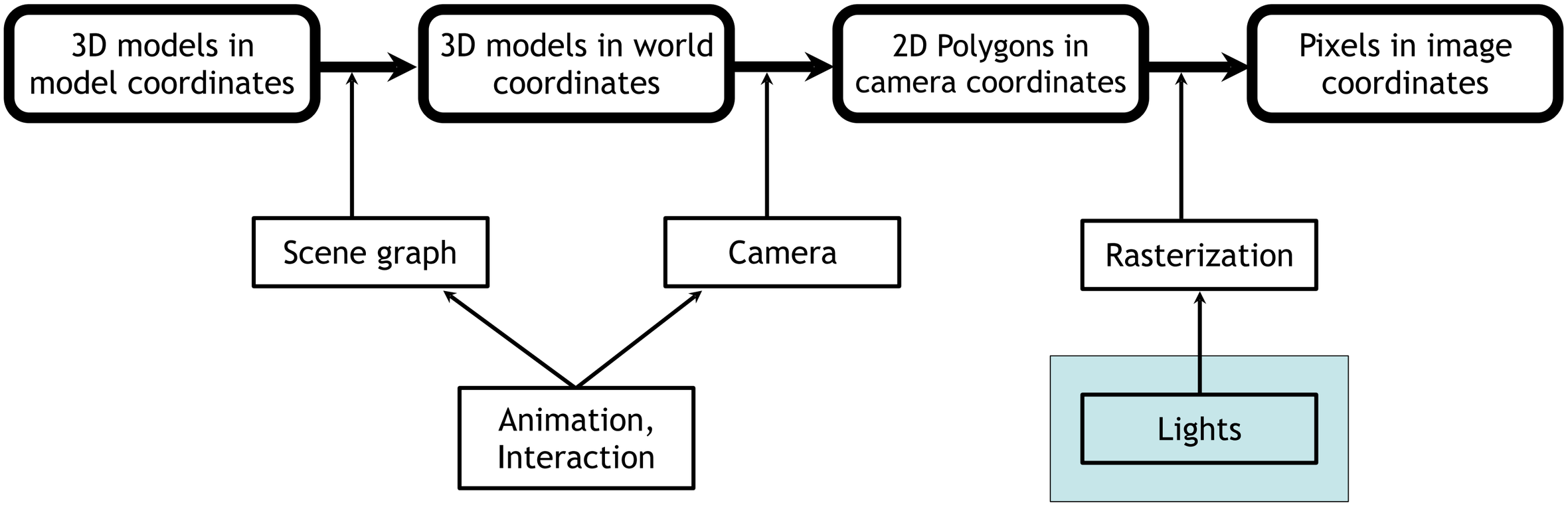
Prof. Dr.-Ing. Andreas Butz
lecture additions by Dr. Michael Krone, Univ. Stuttgart

https://commons.wikimedia.org/wiki/File:Stanford_bunny_qem.png

Chapter 6 - Light, Materials, Appearance

- Types of light in nature and in CG
- Shadows
- Using lights in CG
- Illumination models
- Textures and maps
- Procedural surface descriptions

The 3D Rendering Pipeline (our version for this class)

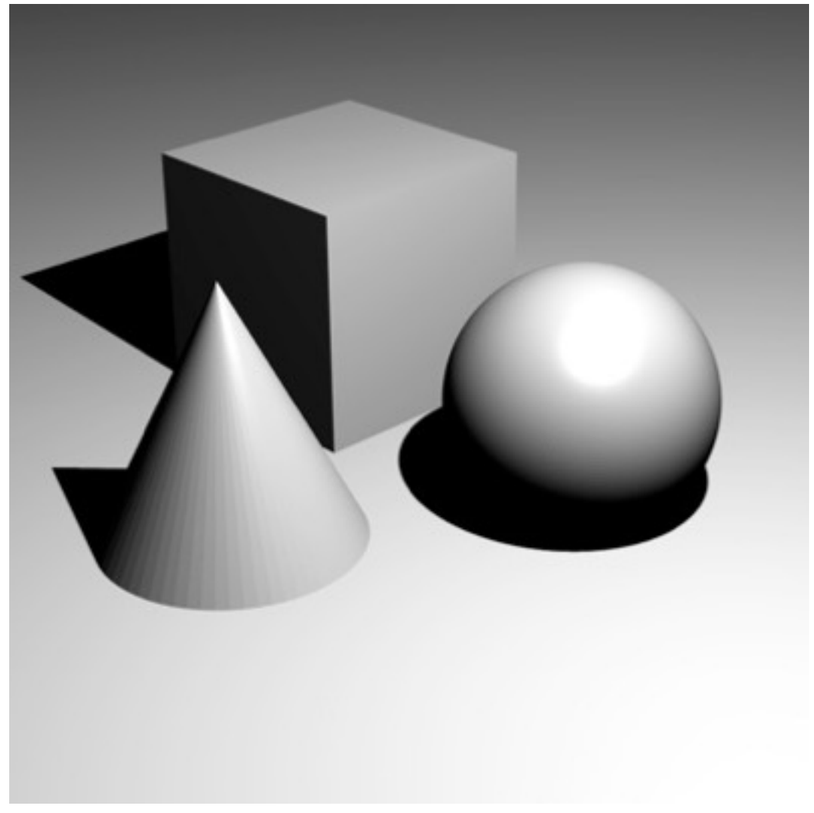


Light in Nature (Physics Refresher)

- Can be described as a electromagnetic wave
- Can also be described as a stream of photons
- Intensity drops with distance from the source
  - How? _____
  - Why? _____
- Monochromatic (1 color) light has 1 frequency
- White light is a mixture of many frequencies
- Can be simulated for the human eye by adding Red, Green and Blue
- The human eye can discriminate a max. dynamic range of $1:2^{30}$ with adaptation or $1:2^{16}$ without [Seetzen et al., „High dynamic range display systems“, ACM Siggraph 2004]
- Film, digital cameras, and computer screens can only deal with less!

http://www.lebjournal.com/newz/wp-content/candle-light-photography.jpg

Point Light Sources



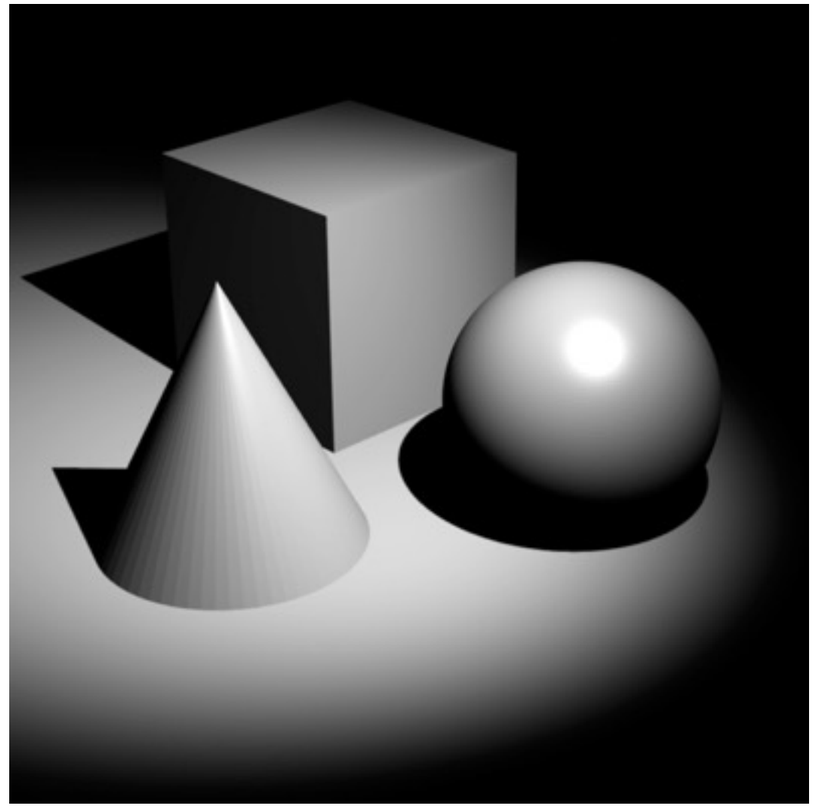
- Have just a position in space
- Emit light equally in all directions
- Intensity falloff with distance d is:
  $I = I_0 / (ad^2 + bd)$
  - This means that the falloff is less harsh than in nature.
  - Why? _____

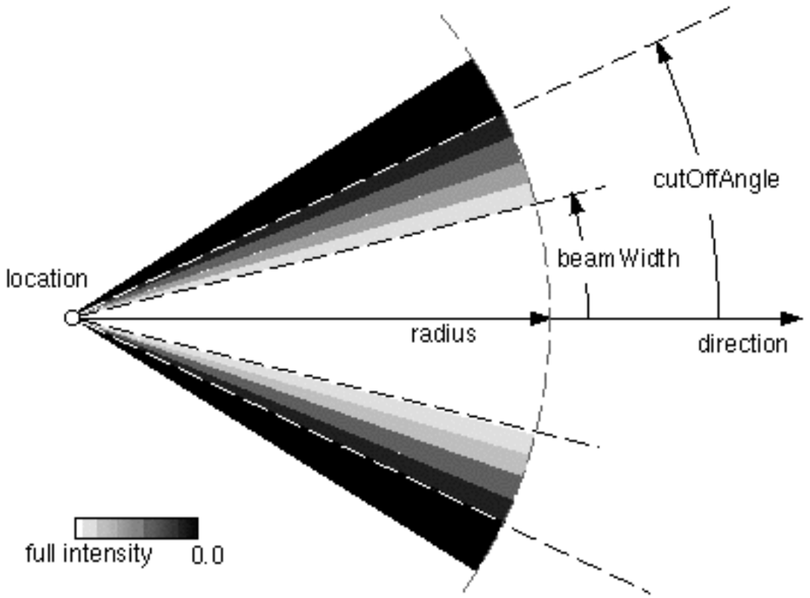
- Light source itself is invisible in the image
  - Since points are infinitely small

- Shadows have sharp edges
- Shadows get bigger with distance from object

Spot Lights

- Have a position and orientation in space
- Have an opening angle and a (optional) parameter controlling the softness of the beam‘s borders
- Intensity falloff with distance d is:
  $I = I_0 / (ad^2 + bd)$
  - This means that the falloff is less harsh than in nature. Why??
  -
- Intensity falloff with angle depends on exact model
- Light source itself invisible
- Object shadows have sharp edges
- Transition to surrounding shadow can be soft

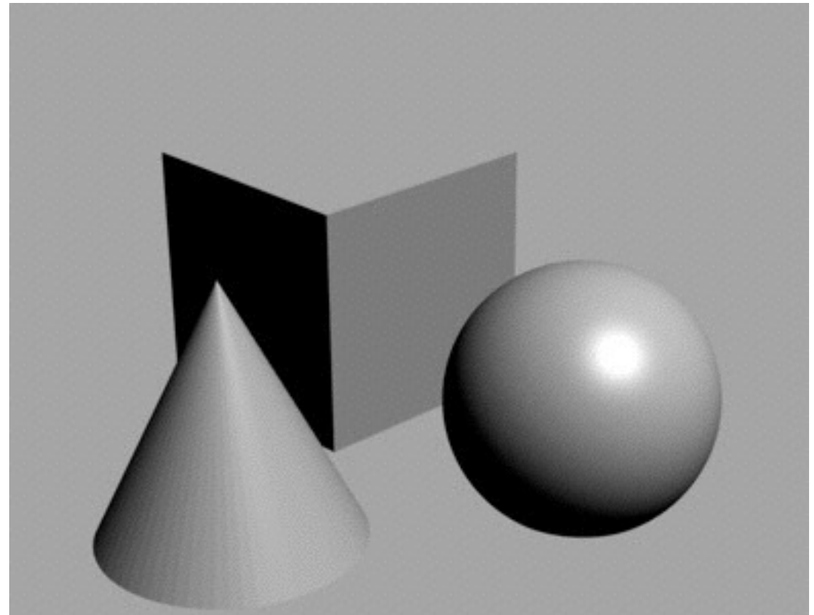




http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/Images/spotlight.gif

Distant Light Source (a.k.a. the sun)

- Size of the earth (radius) = 6.370 km
- Size of the sun (radius) = 695.700 km
- Distance from earth to sun = 150.000.000 km
  - Distance to the sun is practically equal for all points on earth
  - Hence light falloff with distance is not noticeable for sunlight…
  - …and all light rays are practically parallel
- Distant light source in 3D CG has only a direction and a fixed intensity
  - Directional light source
- Good and neutral first step for lighting a scene!
- Shadows should have sharp edges
  - And do not get bigger with distance from the object



Shadows are missing in this image!

Ambient Light

- Equivalent in nature:
  - Light emitted from the entire sky (scattering)
  - Indirect light reflected from objects in the scene

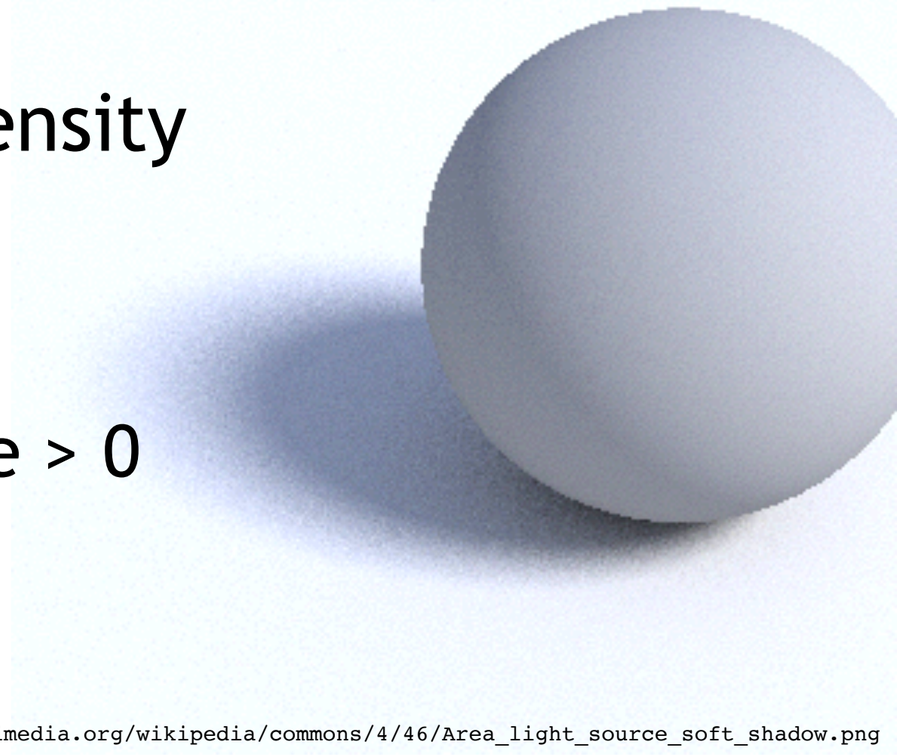
http://de.wikibooks.org/wiki/Datei:Blender3D_li_ambiant_light_occl.jpg

- Intensity is equal from all directions
- Creates low contrast images by itself

- Ambient light is a good way to light up harsh shadows
- Combination with one distant light can already create a decent daylight simulation (sun + sky)

Area Light Sources

- Described by object geometry and light intensity
- Entire area emits light
- All natural light sources are of this kind
  - Even a small light bulb or a LED have a surface > 0



http://upload.wikimedia.org/wikipedia/commons/4/46/Area_light_source_soft_shadow.png

- Light falloff with distance
- Shadows have soft edges

- Computationally difficult, take very long to render correctly
- Can be simulated by many point light sources
- Need global illumination techniques for correct rendering (see later)

Chapter 6 - Light, Materials, Appearance

- Types of light in nature and in CG
- Shadows
- Using lights in CG
- Illumination models
- Textures and maps
- Procedural surface descriptions

Shadows in Nature

- Very important for spatial vision

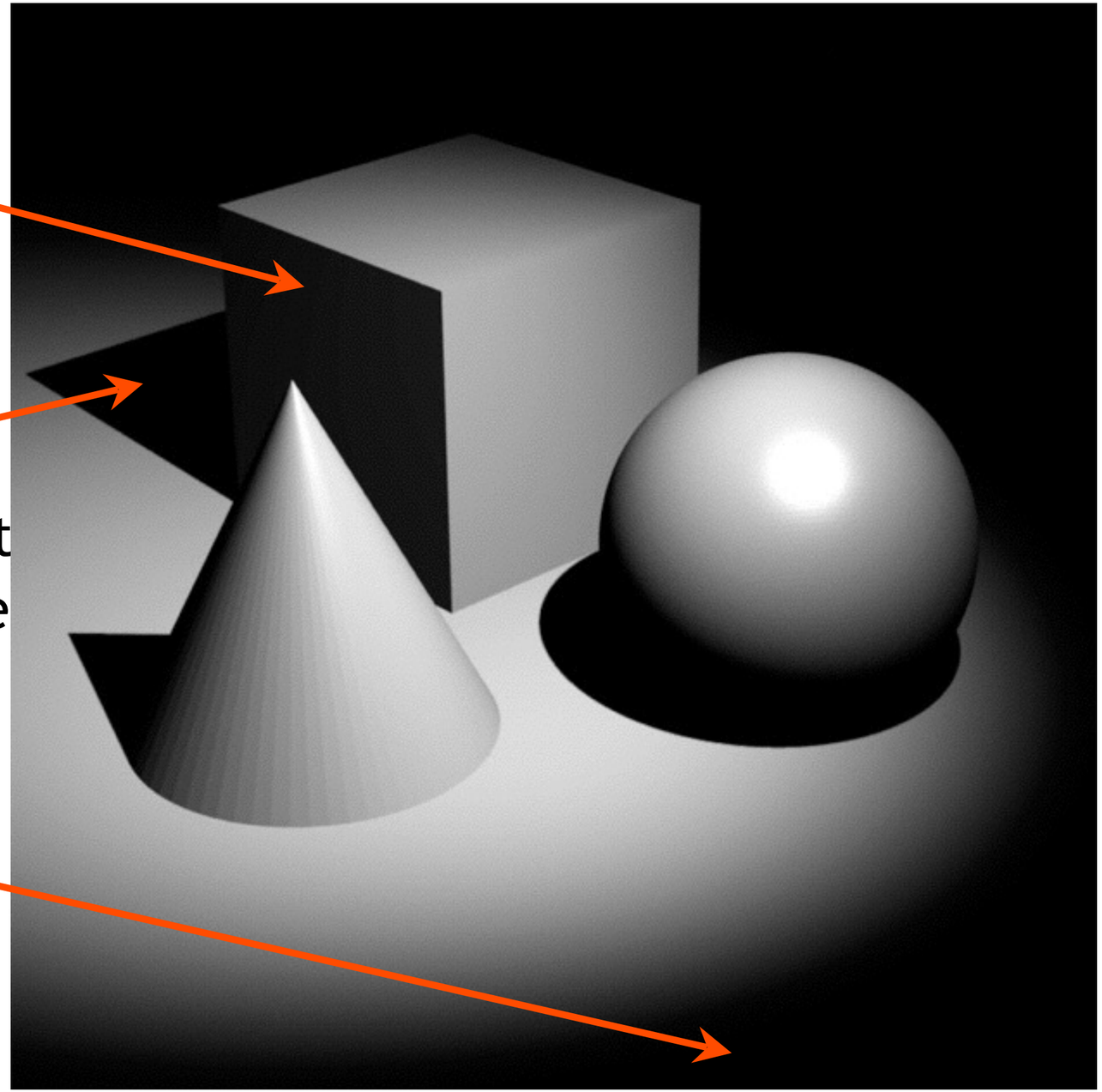
- Artistically used in all art forms
  - Drawing, painting
  - Photography
  - Cinematography

- Practically never really black

- Types of shadows in this image?

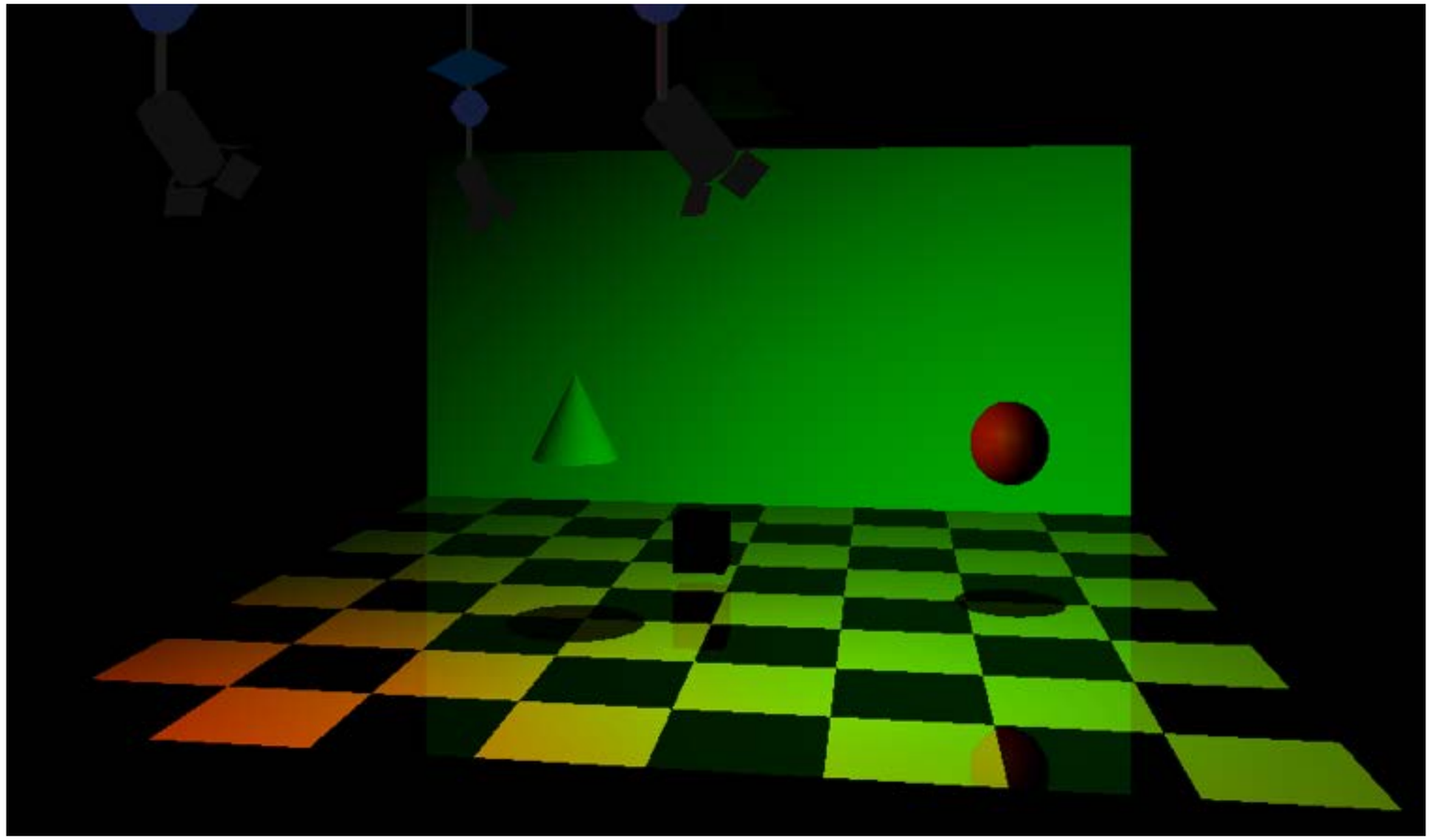
Types of Shadow

- Object shadow
  - The side of objects that points away from the light
  - Exists in free space
- Cast shadow / drop shadow
  - The shadow cast onto another object
  - Need another object or ground plane
- Shadow as the absence of light
  - No light source reaches this place



Cheating a Shadow (and a Reflection!)

- Real-time rendering APIs like OpenGL/WebGL usually only support local lighting (and no area lights)
  - Try to guess how this simple VRML world creates shadows and reflections in real time!

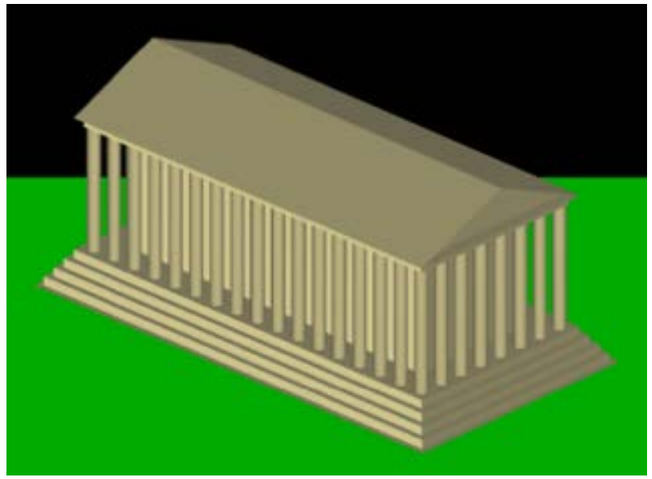


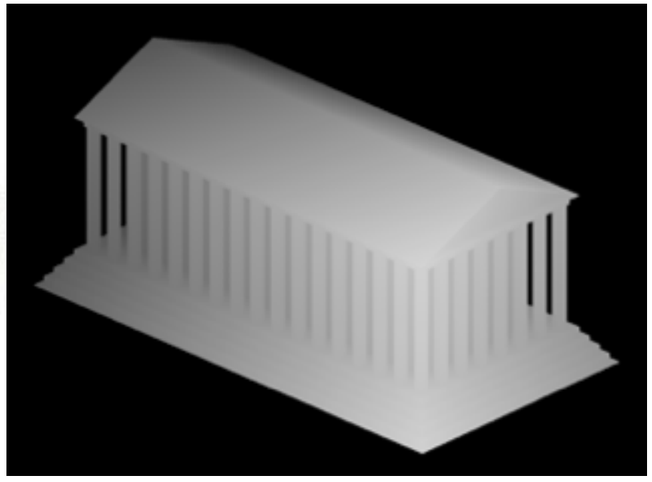
Shadow Maps

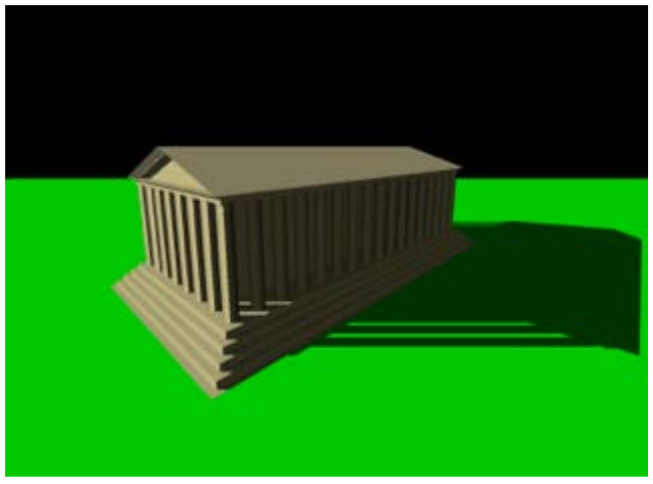
- From the position of a light source, record a depth buffer
  - For each pixel in buffer, we know how far from the light it is

- For each rendered pixel in the camera image, check distance of its surface point to the light
  - If closer than shadow buffer: in this light
  - If further away: in the shadow of this light

- If scene or lights change, shadow map must be recalculated
  - Usually computed per frame (e.g., games)







Chapter 6 - Light, Materials, Appearance

- Types of light in nature and in CG
- Shadows
- Using lights in CG
- Illumination models
- Textures and maps
- Procedural surface descriptions

Using Lights

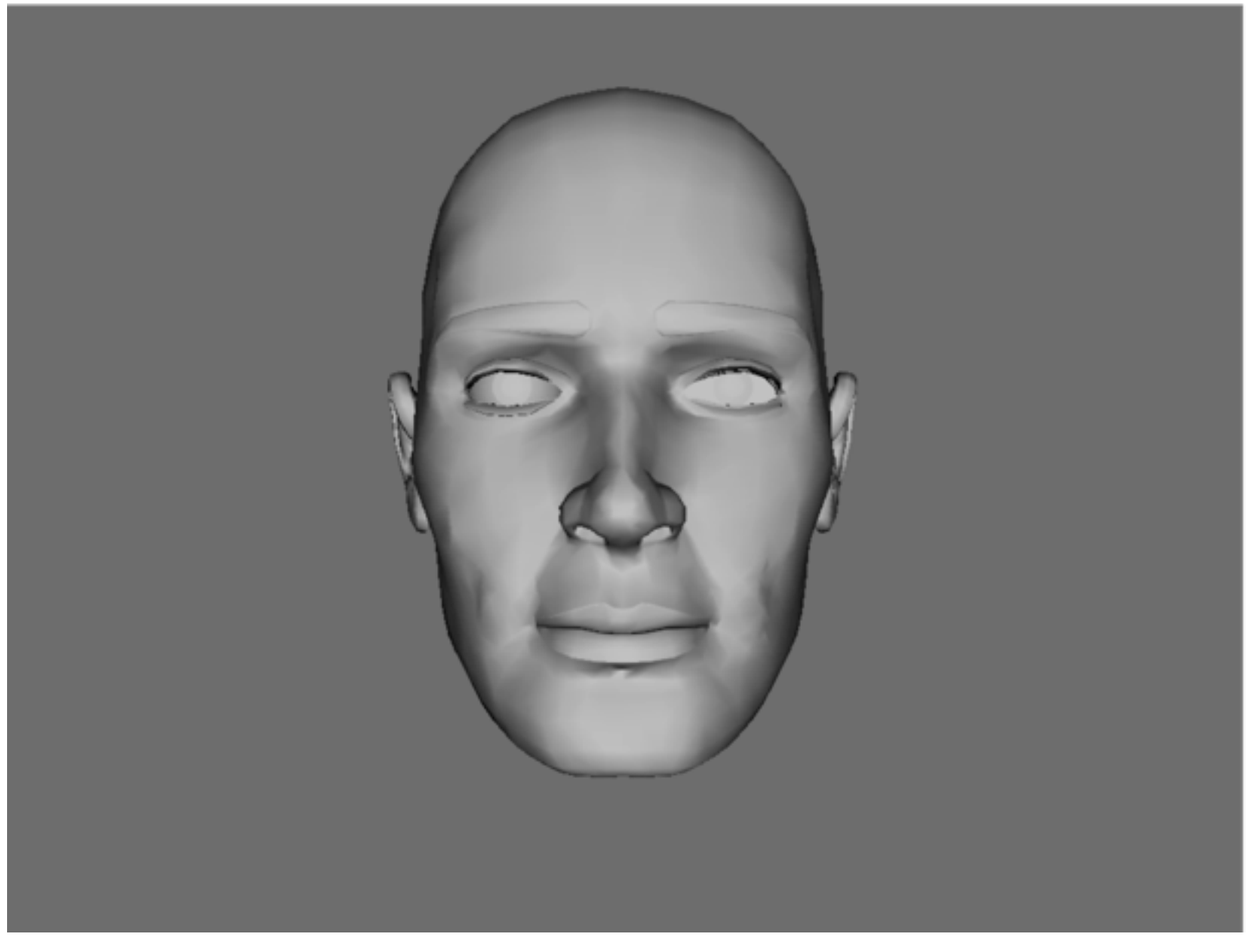
- A few recipes to get started with lighting
- Really good lighting design is an art in itself
  - 3D animated movies hire full time light designers

https://www.nyfa.edu/student-resources/best-cinematography-many-looks-avatar/

https://www.fxguide.com/featured/inside-out-rendering/

Headlight

- Default light setup in VRML
- Light source in camera position
- Scene can be viewed from arbitrary directions
- Creates no visible drop shadows
  - Why? Or does it?



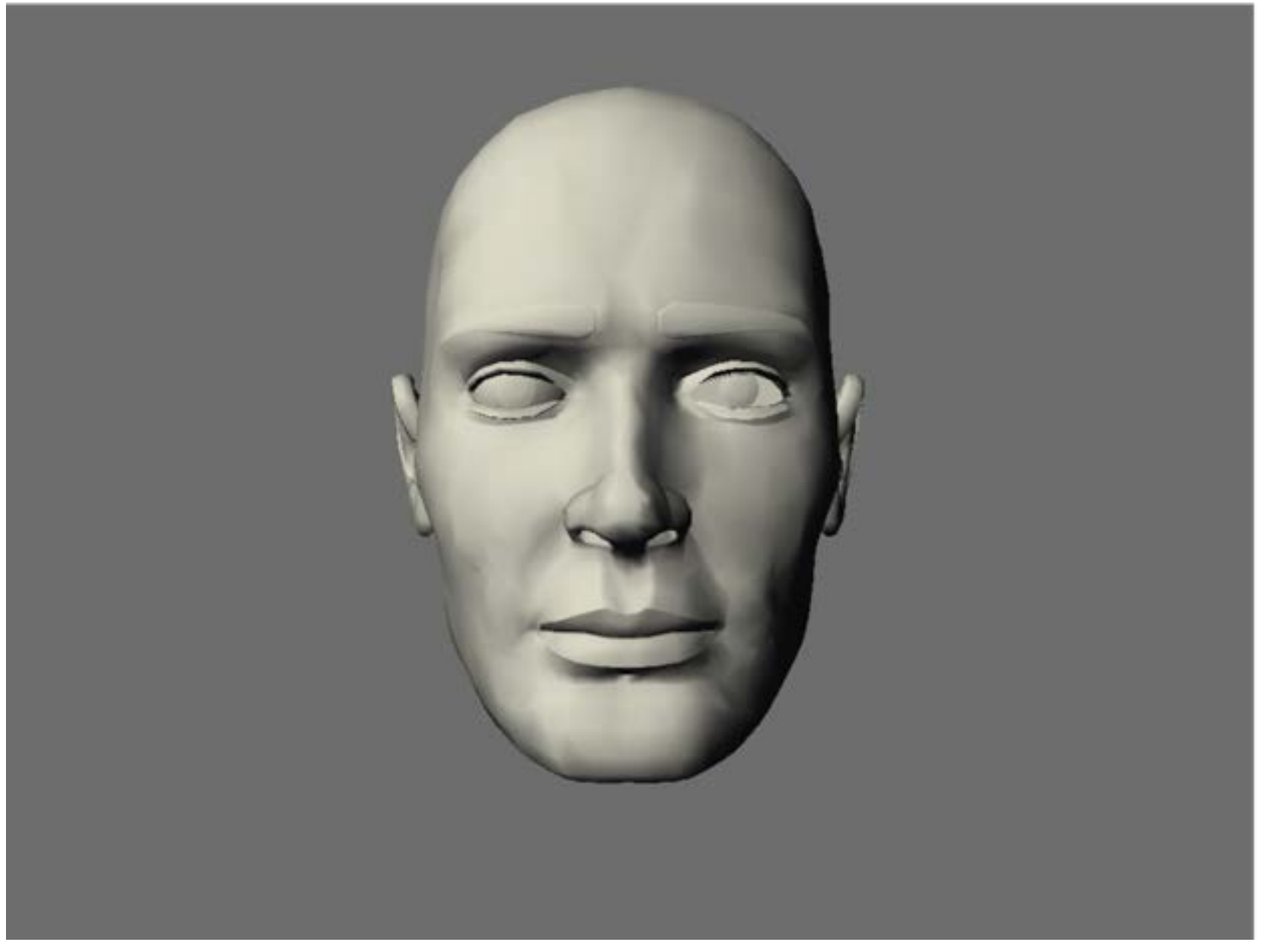
- Creates rather „flat“ images
- Unnatural „flashlight“ look
- Good in combination with other setups for lighting up the scene

http://www.online-superpreis.de/images/produkte/476-stirnlampearcas9er400.jpg

Daylight Simulation

- Sun
  - Distant (directional) light source
  - Warm color tint

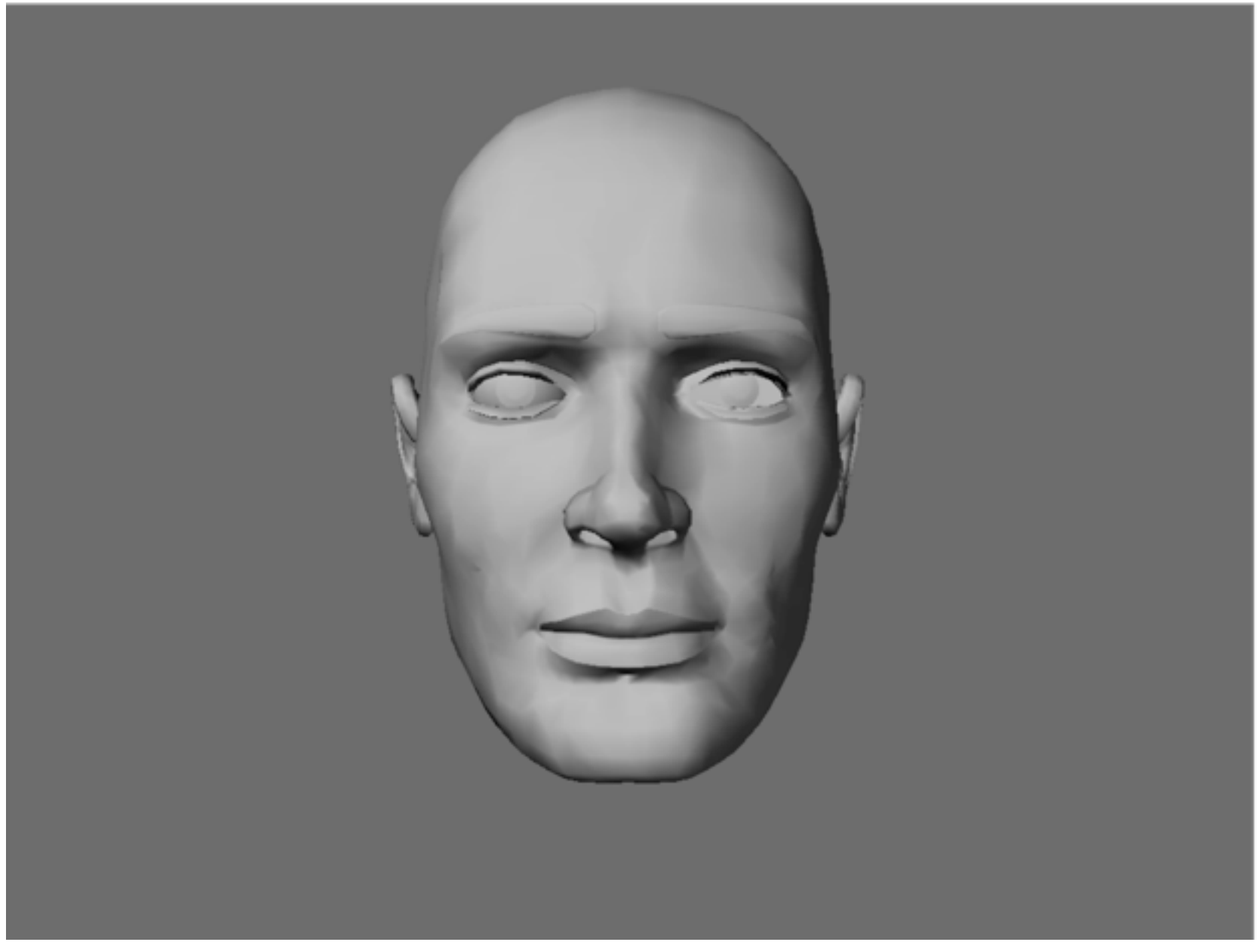
- Sky
  - Ambient light
  - Cool color tint
  - Can be simulated by directional light from opposite side



- Creates a natural look
- Can simulate daytimes (how?)
- Can simulate sunny/cloudy weather (how?)

Simple Portrait Light Setup

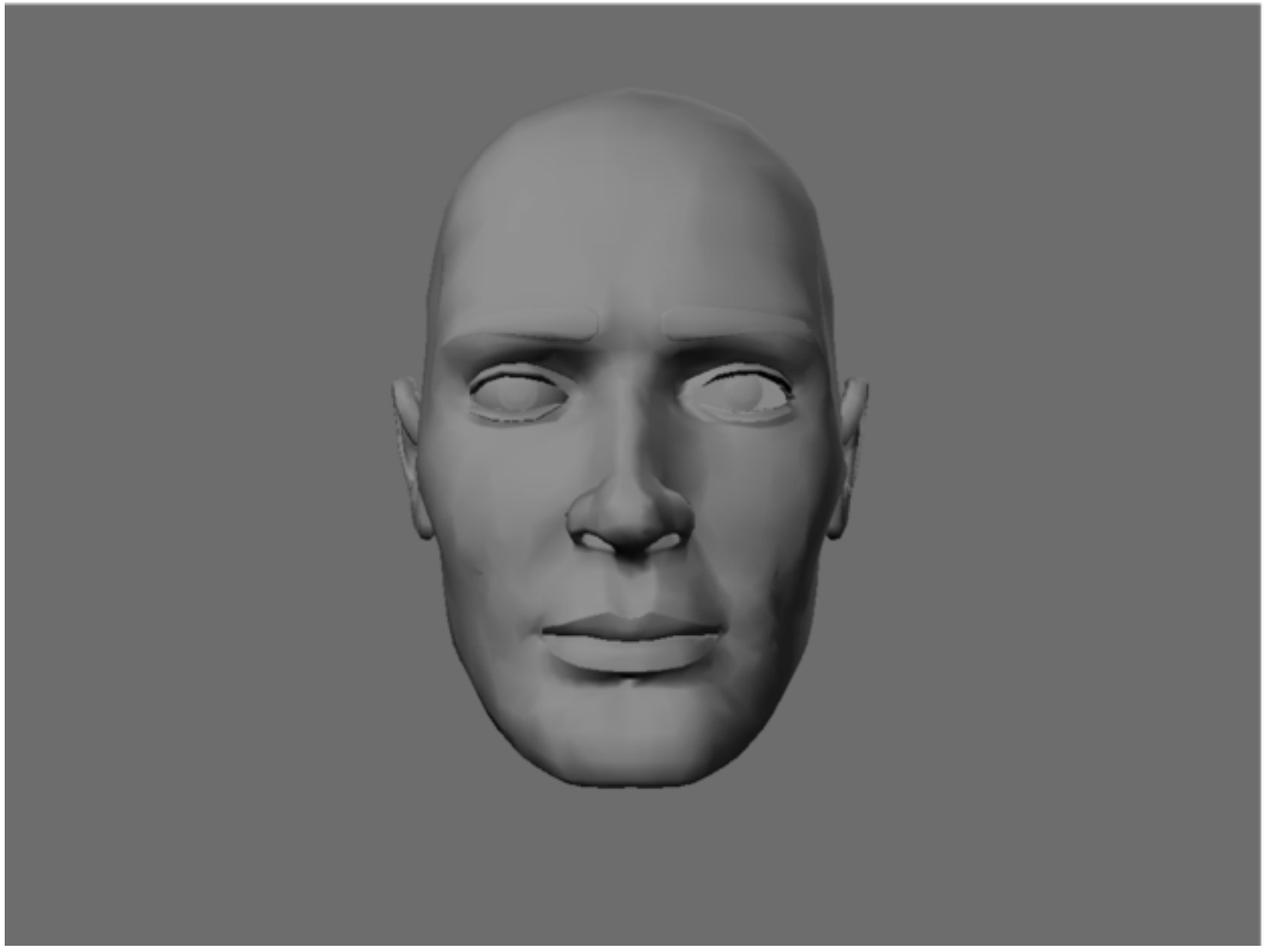
- Borrows ideas from daylight
  - 1 main light source
  - Direction: traditionally from top left
  - Creates overall basic brightness
- One or several brighteners
  - From opposite sides
  - To light up shadows
  - Sum of their brightness less than half of main light (why?)



- Basic setup for scenes viewed from just 1 direction

Sided Light / Grazing Light

- Effect light known from movies
  - Use only in addition to others
- Enhances object contours
- Placement behind the subject
  - Not straight behind, but off-axis
  - Positioning is difficult in real world
  - Easier in graphics, but still:
  - Highly position-dependent



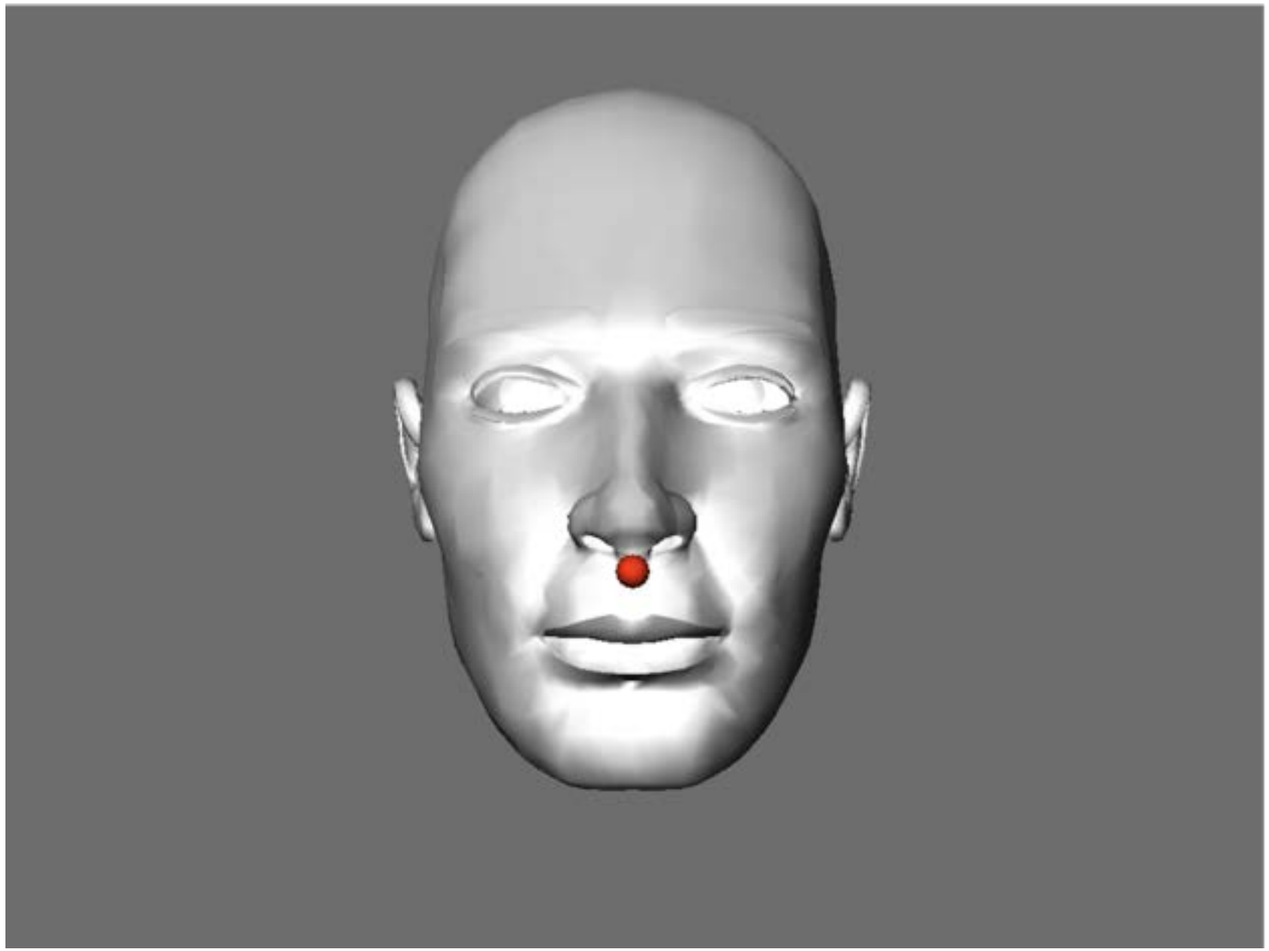
- Can be used to clearly separate an object from the background.
- Will highlight its silhouette.

Cheating with Light

- Light sources in computer graphics are invisible
  - Only their effects on objects are visible!

- Can be positioned anywhere in a scene to light up dark areas

- Example on this slide is exaggerated!



Dramatic Lighting

- Combination of unnatural lights
  - Coming from below
  - Strong colors
  - Mostly low key

- Unlit shadows can create mystery

- Can be supported by unnatural camera
  - From below
  - Wide angle and close up

High Key, Low Key

- High Key: all colors in image are bright
  - Start with very even lighting
  - Frontal light will remove shadows
  - Danger of saturated white
  - Communicates light and cleanliness

- Low Key: all colors are very dark
  - Often uses sided light
  - Objects can be reduced to their contours
  - Communicates e.g., mystery

Chapter 6 - Light, Materials, Appearance

- Types of light in nature and in CG
- Shadows
- Using lights in CG
- Illumination models
- Textures and maps
- Procedural surface descriptions

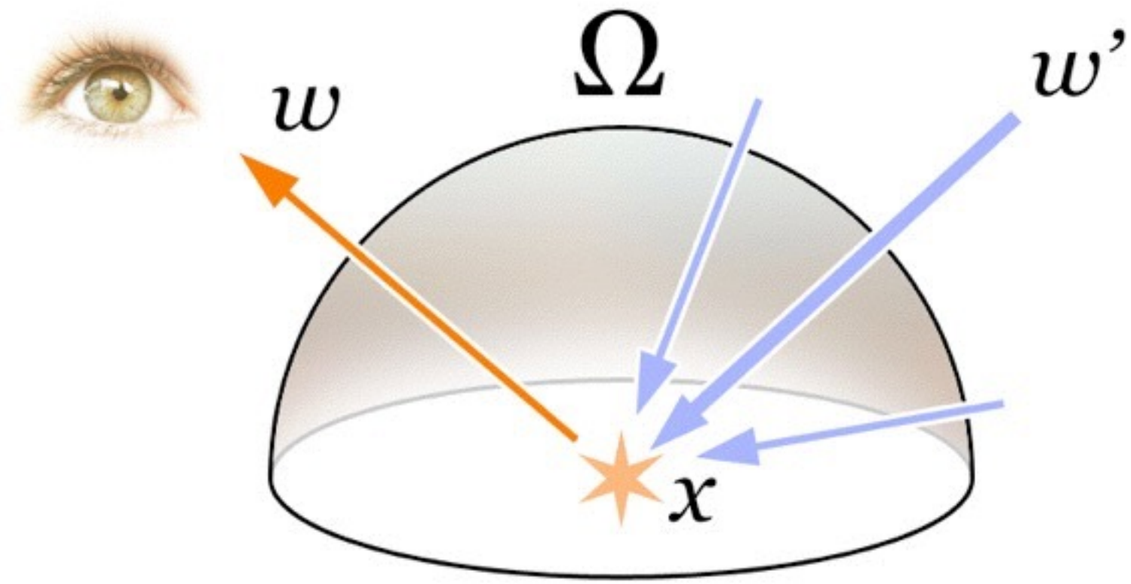
Surfaces in Nature

- What does a surface do to light? (mini-Brainstorming)

The Rendering Equation [Kajiya ‘86]

$$I_o(x,\vec{\omega}) = I_e(x,\vec{\omega}) + \int_\Omega f_r(x,\vec{\omega}',\vec{\omega}) I_i(x,\vec{\omega}')(\vec{\omega}' \cdot \vec{n}) d\vec{\omega}'$$

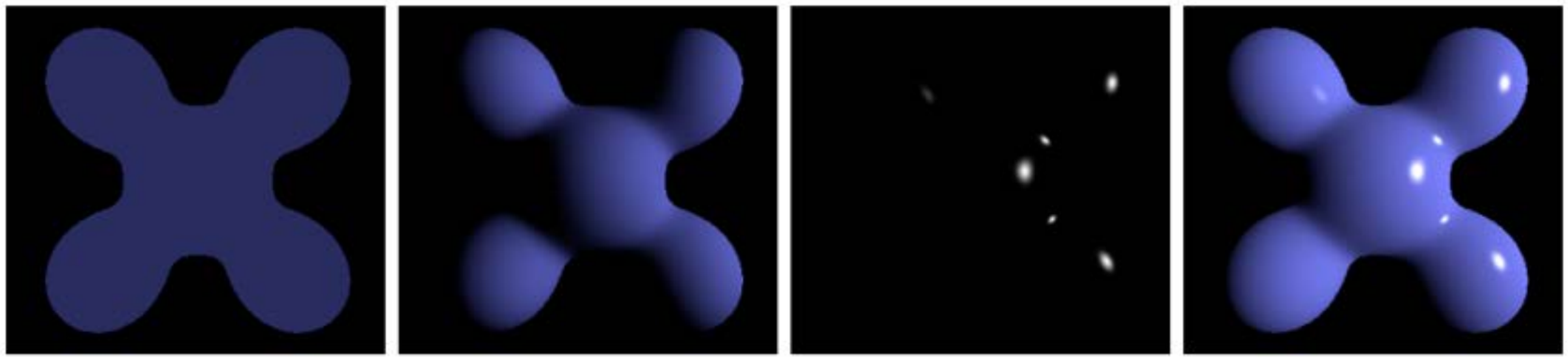
- I_o = outgoing light
- I_e = emitted light
- Reflectance Function
- I_i = incoming light
- Angle of incoming light
- Describes all flow of light in a scene in an abstract way
- Doesn‘t describe some effects of light:
  -
  -



http://en.wikipedia.org/wiki/File:Rendering_eq.png

Phong‘s Illumination Model [Bùi Tường Phong, 1973, PhD thesis]

$$I_o = I_{amb} + I_{diff} + I_{spec}$$

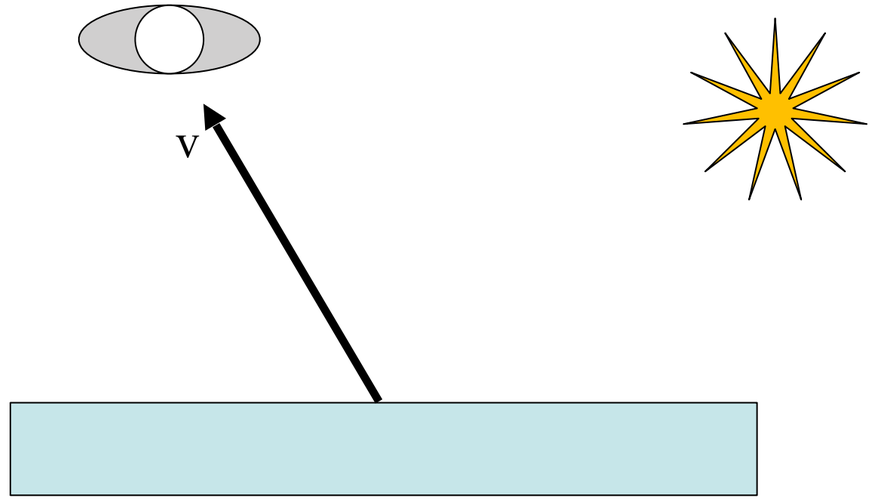


http://de.wikipedia.org/w/index.php?title=Datei:Phong_components_version_4.png

- Strong simplification and specialization of the situation
  - Just 1 light source from a clear direction l
  - Viewing direction is given as v
- Only 3 components:
  - Ambient component: reflection of ambient light source from and in all directions
  - Diffuse component: diffuse reflection of the given light source in all directions
  - Specular component: „glossy“ reflection creating specular highlights

Ambient Component

- I_a = Intensity of the ambient light source

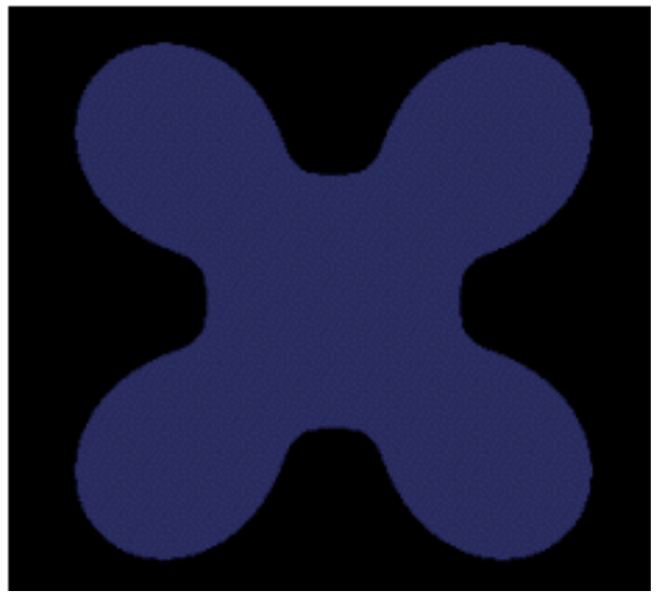


- Independent of any directions

- Can simulate a „glowing in the dark“

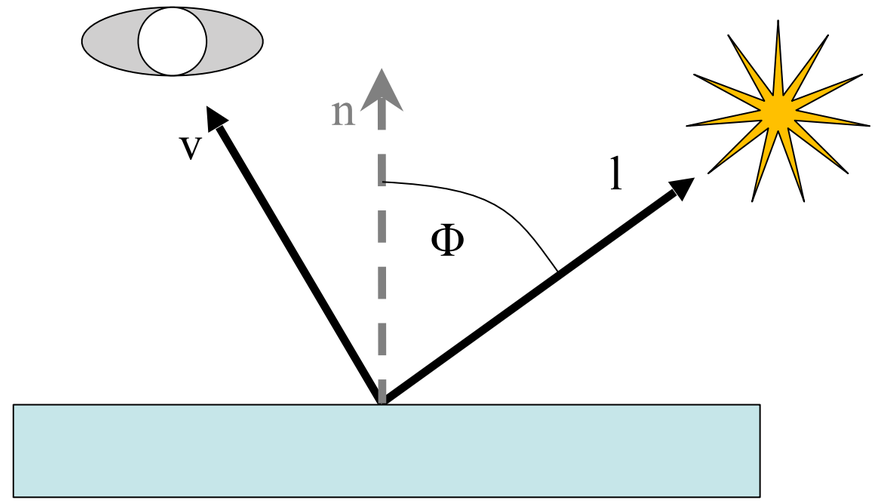
$$I_{amb} = I_a k_a$$

- Can be seen as the equivalent to emitted light I_e in the rendering equation



Diffuse Component

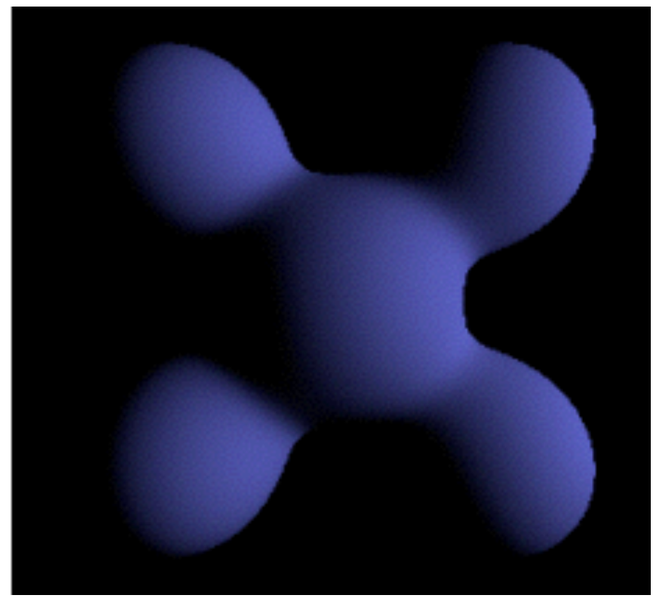
- Diffuse reflection is equal in all directions
- Depends on the angle of incident light
  - Light along the surface normal: maximum
  - Light perpendicular to the normal: 0



- Cosine function describes the energy by which a given area is lit, dep. on angle
  - Hence, cosine is used here

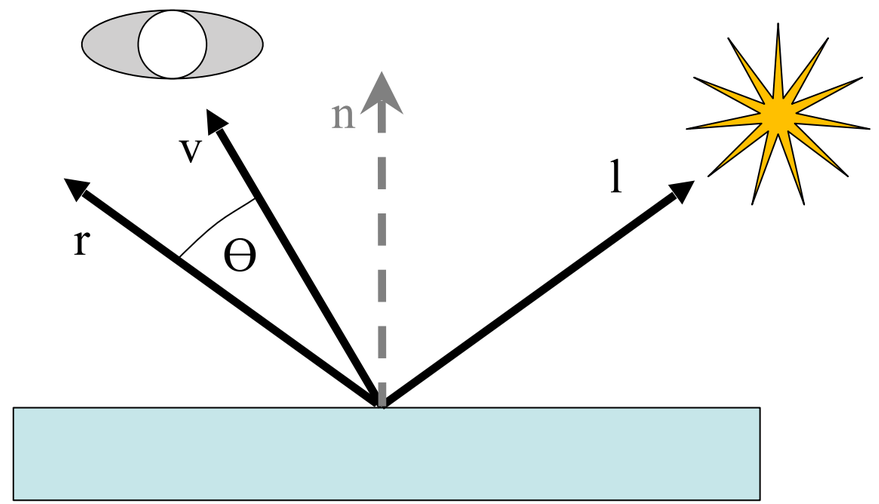
$$I_{diff} = I_i k_d cos\phi = I_i k_d (\vec{l} \cdot \vec{n})$$

- „Lambertian“ surface
- Visual equivalent in nature: paper



Specular Reflection

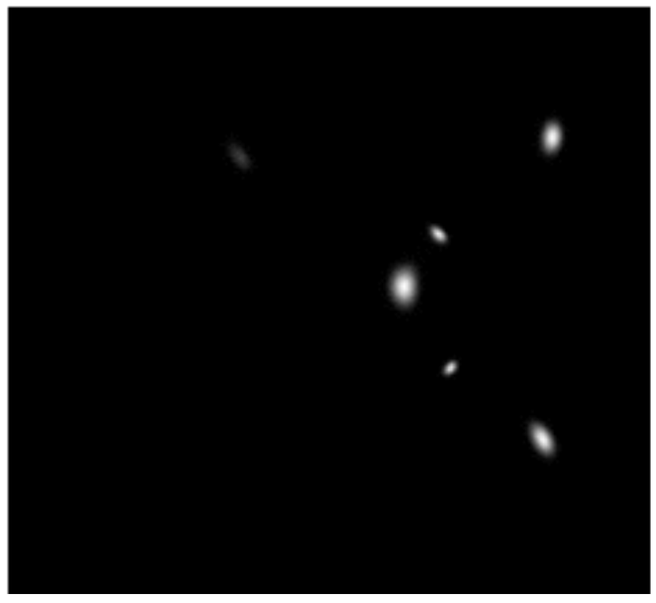
- Let r be the reflection of l on the surface



- Specular reflection depends on the angle between v and r
  - v = -r: maximum
  - v and r perpendicular: minimum

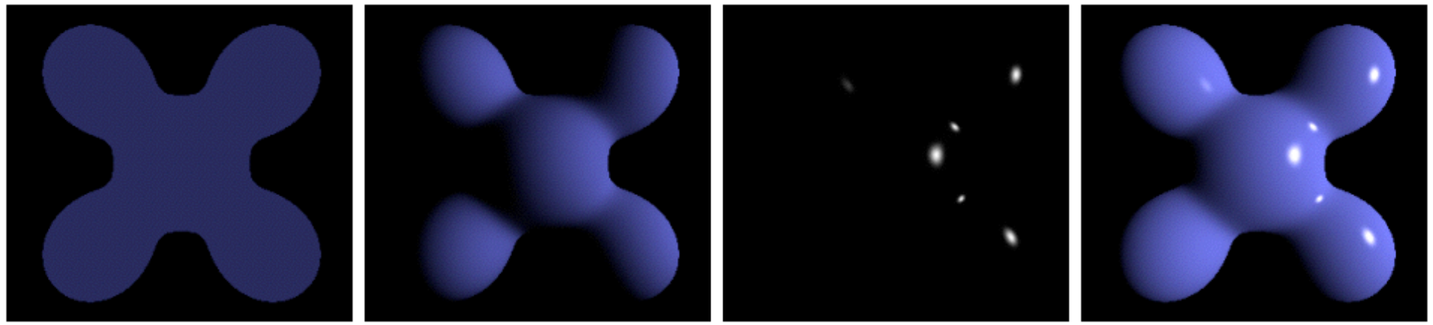
$$I_{spec} = I_i k_s cos^n \theta = I_i k_s (\vec{r} \cdot \vec{v})^n$$

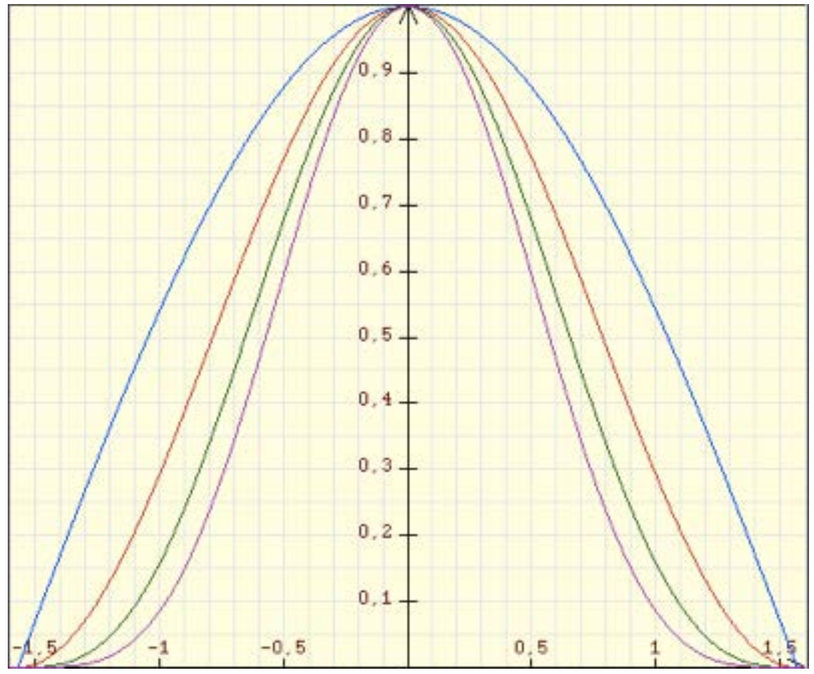
- Function cos^n behaves correctly
  - Exponent n determines how wide the resulting specular highlight is
  - Other functions could be used as well



Tweaking the Parameters

$$I_o = I_{amb} + I_{diff} + I_{spec} = I_a k_a + I_i k_d (\vec{l} \cdot \vec{n}) + I_i k_s (\vec{r} \cdot \vec{v})^n$$



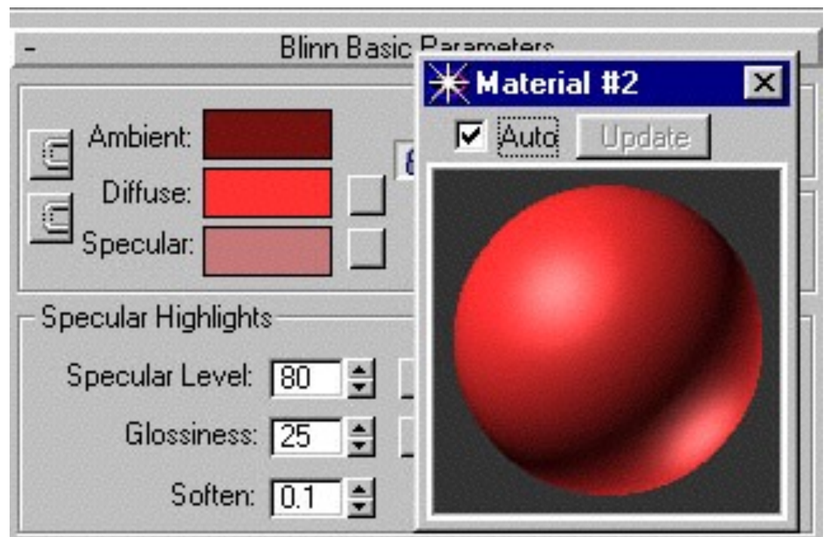


- Choose k_s = 0 for perfectly matte material
- Choose k_a > 0 to avoid harsh shadows
- Keep k_a small to avoid „glowing“ objects
- Add in some k_s > 0 to add gloss
- Adjust the size of specular highlights with n

- All of these calculations generalize to (RGB) color, of course!

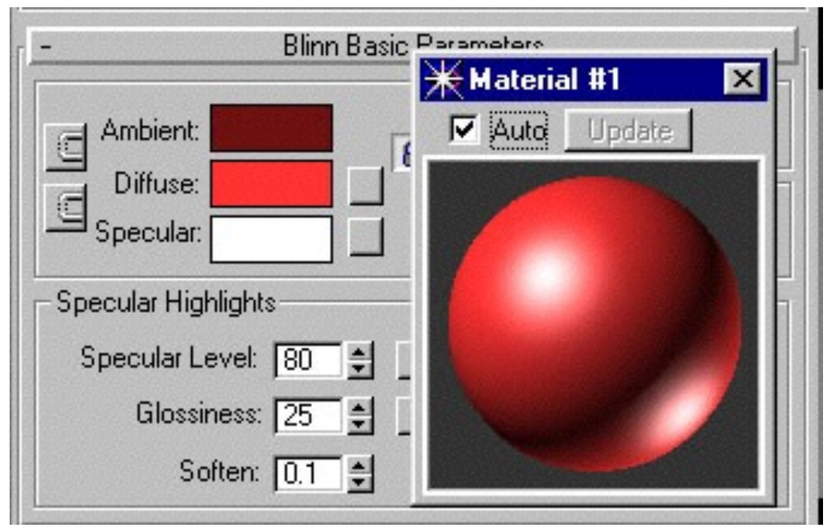
The VRML Material Node

```
Material {
  exposedField SFFloat ambientIntensity 0.2
  exposedField SFColor diffuseColor     0.8 0.0 0.0
  exposedField SFColor emissiveColor    0.2 0.0 0.0
  exposedField SFFloat shininess        0.5
  exposedField SFColor specularColor    1.0 1.0 1.0
  exposedField SFFloat transparency     0.0
}
```



→ All values in [0.0, 1.0]

→ Shininess in VRML is multiplied by 128 to produce n in the lighting model.



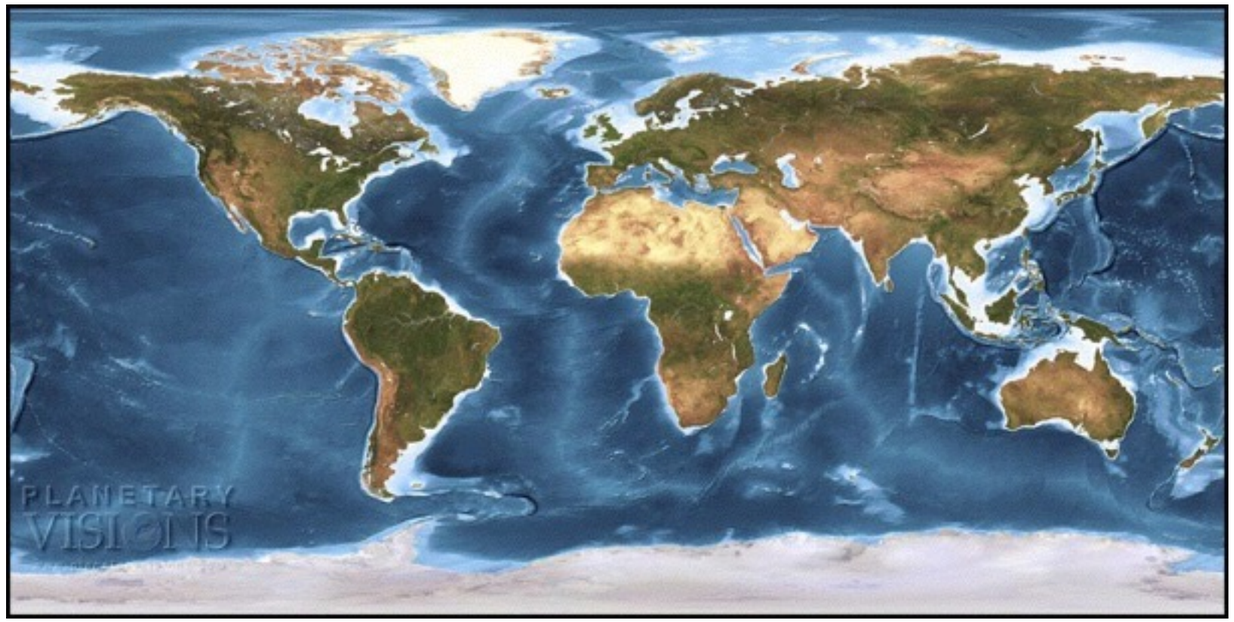
http://dxyner2000.com/tutorials/tut3.1.htm

Chapter 6 - Light, Materials, Appearance

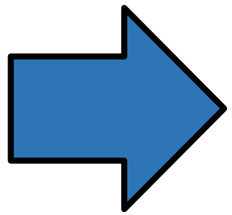
- Types of light in nature and in CG
- Shadows
- Using lights in CG
- Illumination models
- Textures and maps
- Procedural surface descriptions

Textures and Maps

- One of the simplest and oldest ways to achieve good looking objects with simple geometry
- Texture design is a very complex task, needs a lot of imagination!
- **Idea:** use a bitmap image, shrink wrap around the object
  - Use bitmap contents for object surface color: image map
  - Can be used for other parameters, e.g., normal, elevation, transparency, reflection
- **Problem:** what does shrink wrap mean exactly?



http://www.planetaryvisions.com/Texture_map.php?pid=4128

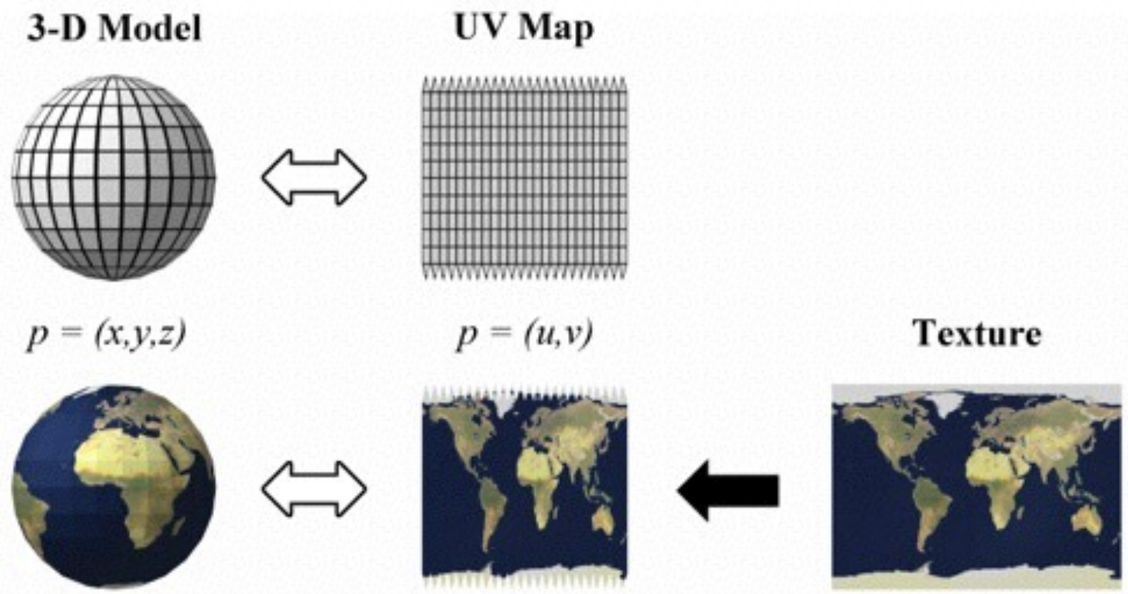


Texture Coordinates and UV Mapping

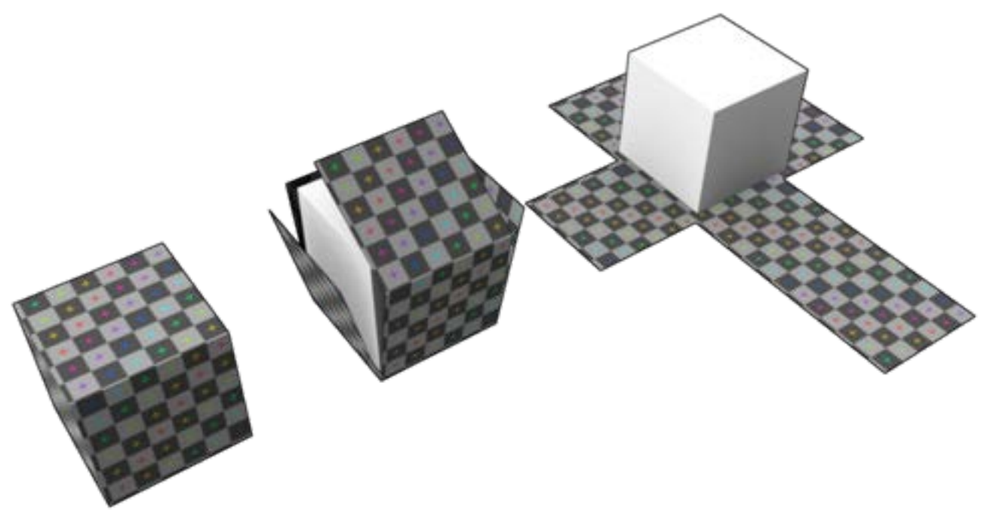
- Each texture is mapped to a 1x1 square
- Each object defines u,v coordinates
  - Such that u,v are both between 0 and 1

- Straightforward for geometric primitives
  - Different possibilities
  - Conventions exist

- Not so easy for polygon models
  - Can be defined per vertex
  - ...but who wants to do this?
  - Simplifications: shrink a sphere onto the object
    - Works fine with convex objects
    - Always tricky for complicated objects



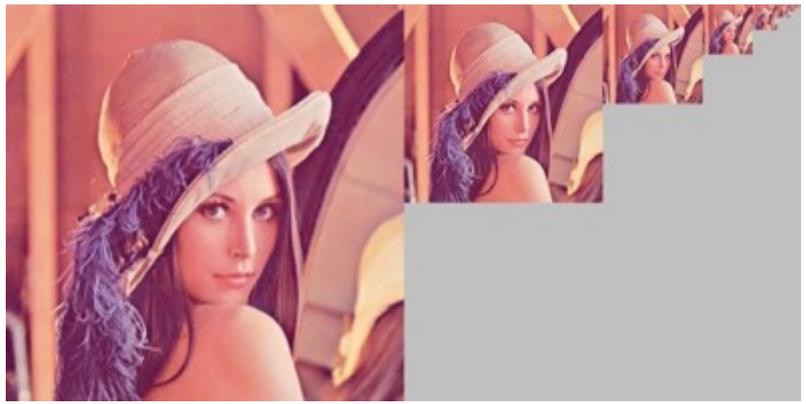
http://upload.wikimedia.org/wikipedia/commons/0/04/UVMapping.png



http://en.wikipedia.org/wiki/File:Cube_Representative_UV_Unwrapping.png

Texture Filtering

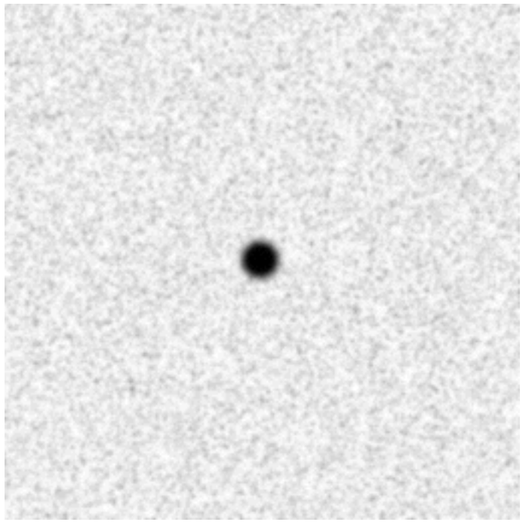
- During rasterization, for each rendered pixel of the textured object we need to look up a color value from the texture
  - Will almost always fall between texture pixels (texels)
  - Texture may have too much resolution: sampling or integration
  - Texture may have too little resolution: interpolation
- Naïve approach: pick the nearest neighbor pixel
  - Leads to blocky textures
- Better approach: bilinear filtering
  - Pick the 4 neighboring pixels and linearly interpolate
- Mip map: image pyramid with image scaled to 1/4 area in each step
  - Eliminates excessive integration over pixels
- Trilinear filtering: find the 2 best levels of the mip map and interpolate within and between them



http://wiki.aqsis.org/dev/texture_filtering

Bump Mapping

- Texture file is only greyscale
- Grey value determines the elevation of the surface
  - e.g., black = dent, white = bulge



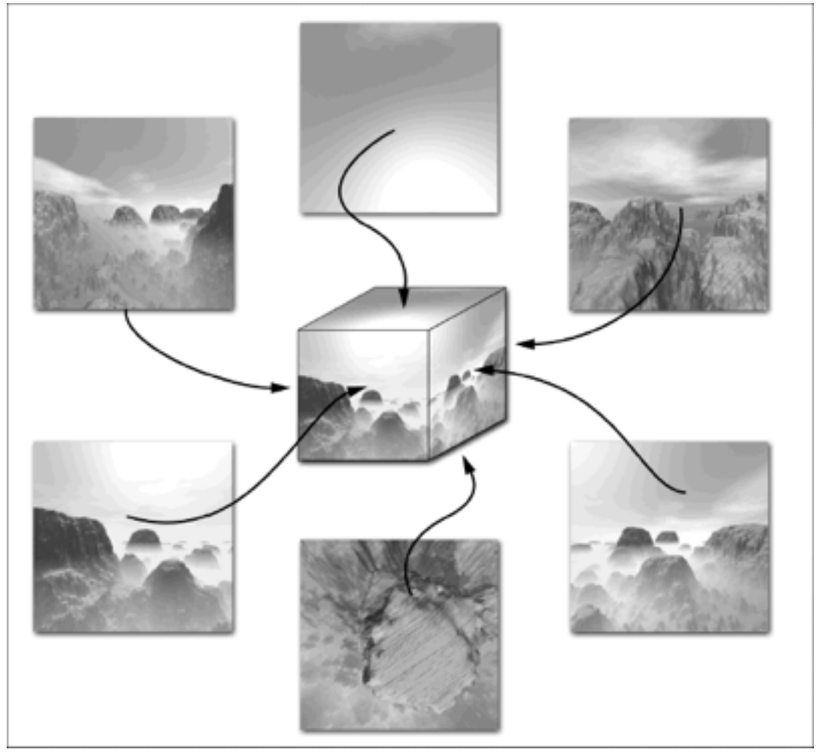
- Can simulate complex 3D surface structure on very simple geometry
- Often used together with image maps to enhance realism
- Only modifies surface color, not silhouette!

- Introduced by Jim Blinn in 1978
  - Related and improved techniques with similar look in use today: normal mapping, displacement mapping

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

Environment Maps

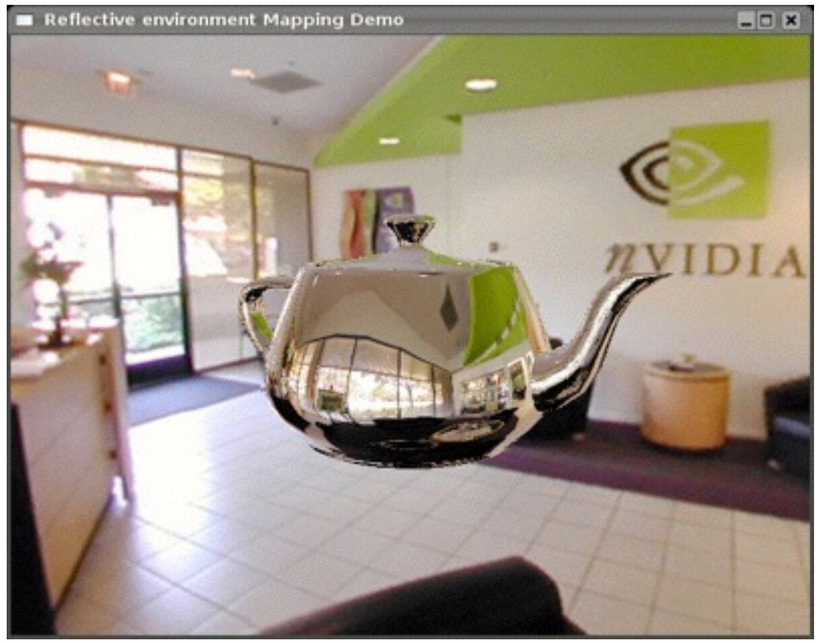
- Maps show the environment of the object
  - Inside out view, 360 degrees in all directions
  - Can be represented as 6 sides of a cube
  - Can be photographed in a real environment



http://www.developer.com/img/articles/2003/03/24/EnvMapTecIm01.gif

- Can be used to calculate appropriate reflections
  - Problem:

- Can also be used for lighting
  - Record map in real environment
  - Light a 3D model with it
  - This model will seem as if lit in the real environment
  - Useful for combining real and virtual objects



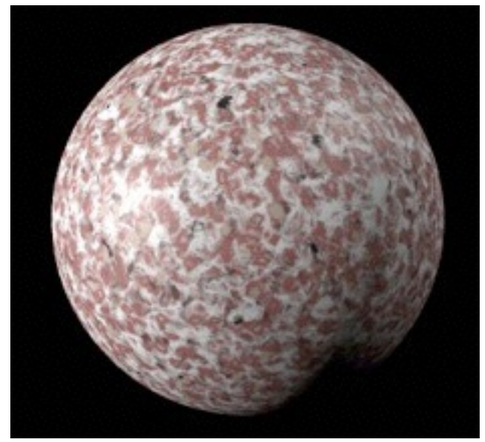
http://tfc.duke.free.fr/

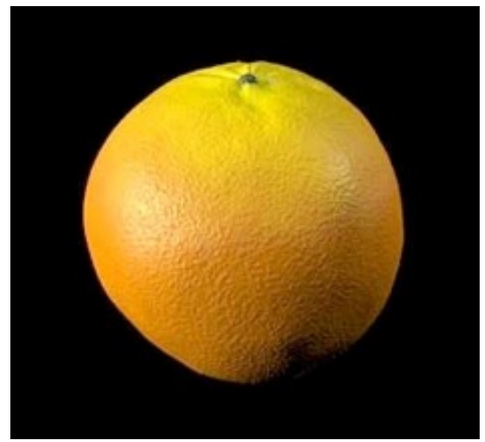
Chapter 6 - Light, Materials, Appearance

- Types of light in nature and in CG
- Shadows
- Using lights in CG
- Illumination models
- Textures and maps
- Procedural surface descriptions

Procedural Surface Descriptions

- Programming languages for surface descriptions
- Can influence various stages of the rendering pipeline
  - In particular: can implement textures and the Phong model
  - But also much more...
- Can describe real 3D structures
  - Not just surface color
- State of the art in high end 3D graphics
  - e.g., RenderMan, used in PIXAR movies
  - Also in OpenGL, DirectX
- Detailed implementation varies depending on the platform
- In OpenGL: vertex shaders and fragment shaders
  - Fragments = parts of an object that cover 1 screen pixel





https://renderman.pixar.com/products/tools/rms-slim.html

OpenGL Recap (see Chapter 4): Vertex and Fragment Shaders

- A vertex shader can do the following:
  - Transform the vertex position (e.g., using model-view and projection matrices)
  - Transform normals, and if required normalize them
  - Generate and transform texture coordinates
  - Lighting per vertex or compute values for lighting per pixel
  - Per-vertex color computations
- A fragment shader can do the following:
  - Compute colors, and texture coordinates per pixel
  - Apply a texture
  - Fog computation
  - Compute normals if you want lighting per pixel
- This, and more examples at: `http://www.lighthouse3d.com/opengl/glsl/`

Links, Various

- You‘ve been doing 3D computer graphics too long if ...when people ask you, "What's up?", you reply "Y": http://www.deakin.edu.au/~agoodman/scc308/toolong.html

- More detailed class material from one of the world‘s leading groups: http://graphics.stanford.edu/courses/

- More compact overviews from the wisdom of the masses ;-): http://en.wikipedia.org/wiki/Rendering_(computer_graphics)

- Kajiya, James T. (1986), "The rendering equation", SIGGRAPH 1986: 143, doi:10.1145/15922.15902

- Some nice related tutorials: http://www.lighthouse3d.com/tutorials/