CMSC 635

Lighting

Lighting & Illumination

- Interaction of light with surfaces
- Local Illumination
  - Each point independent of every other
- Global Illumination
  - Lighting at one point affects others

Lights

- $L = P_L - P_S = w_S\, p_L - w_L\, p_S$
- Directional: (x, y, z, 0)
  - Far enough away that rays are parallel
- Point: (x, y, z, 1)
  - Shines in all directions from point
  - Normally no falloff with distance
  - Physical: Attenuate I_L by $1/(L \bullet L)$
    - May require $I_L > 1$

Lights

- Spot
  - Point + direction and cone
    - Scale I_L by $L \cdot D^e$
- Area
  - Line: like florescent tube
  - Patch: like light fixture
- Environment

Environment map

- Approximate light from all directions as seen by each point on surface
- Instead use light from all directions as seen by one representative point
- Distant environments
- Direction-based texture map

Direction-based mapping

- Vector R = (x,y,z)
- Cube map
  - Six images on cube faces
  - Divide other two components by largest
  - Say it is y: (s,t,q) = (x, z, y)
    - S = x/y; T = z/y
  - Scale into texture: (S+1)/2, (T+1)/2

Direction-based mapping

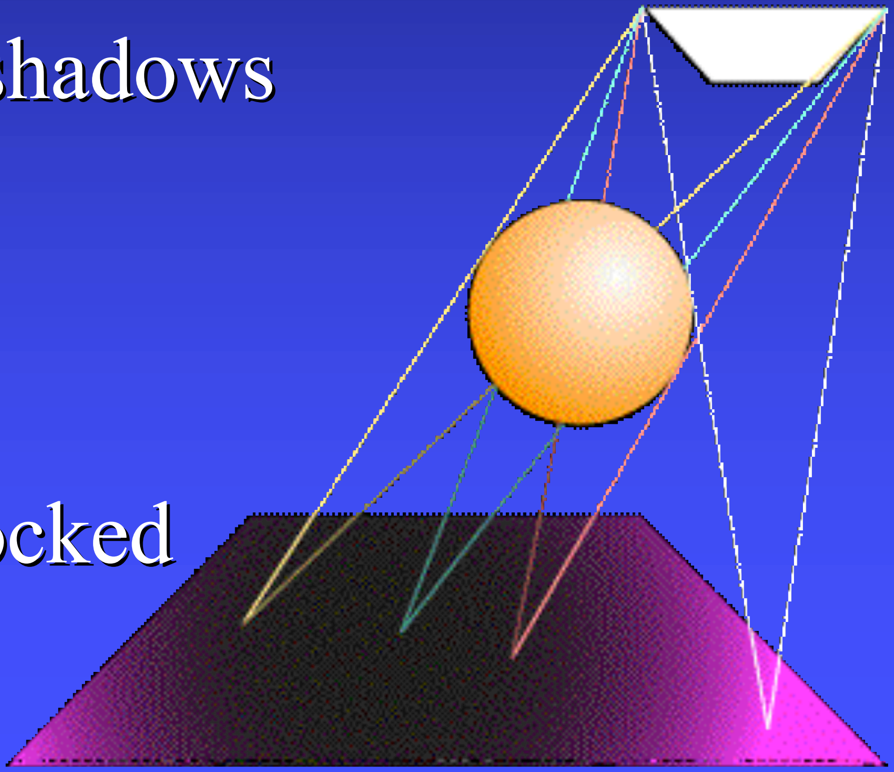
- Sphere map
  - (s,t) = (x,y) on shiny sphere refl. V to R
    - $V = (0,0,-1)$
    - $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$
  - N half way between V and R
    - $N = (V+R)/|V+R| = (2x, 2y, 2z)/2$
  - $(s,t,q) = x,y,\mathrm{sqrt}(x^2+y^2+(z-1)^2)$

Direction-based mapping

- Parabolic maps
  - $(s,t) = (x,y)$ on shiny parabola
    - Need two
      - $V=(0,0,1)$; $f(x,y,z) = z + (x^2 + y^2)/2=0$
      - $V=(0,0,-1)$; $f(x,y,z) = z - (x^2 + y^2)/2=0$
  - $(s,t,q) = (x, y, z - 1)$
  - $(s,t,q) = (x, y, 1 - z)$

Shadows

- *Occluder* blocks light
- Point lights: hard shadows
- Area lights: soft shadows
  - Umbra
    - blocked
  - Penumbra
    - partially blocked

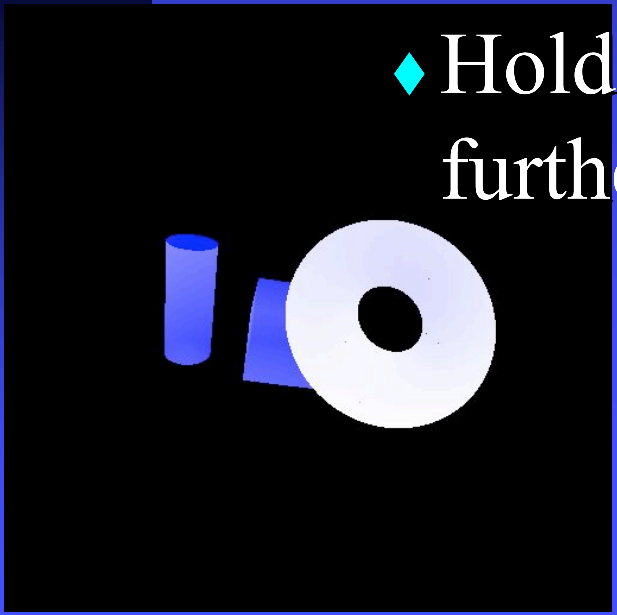


Blinn method

- Hard shadows on planar surfaces
- Project copy of object onto plane
    - Extra modeling transform matrix
- Avoid occlusion problems
    - Depends on rendering algorithm
    - Common to just translate “shadow” object slightly off surface

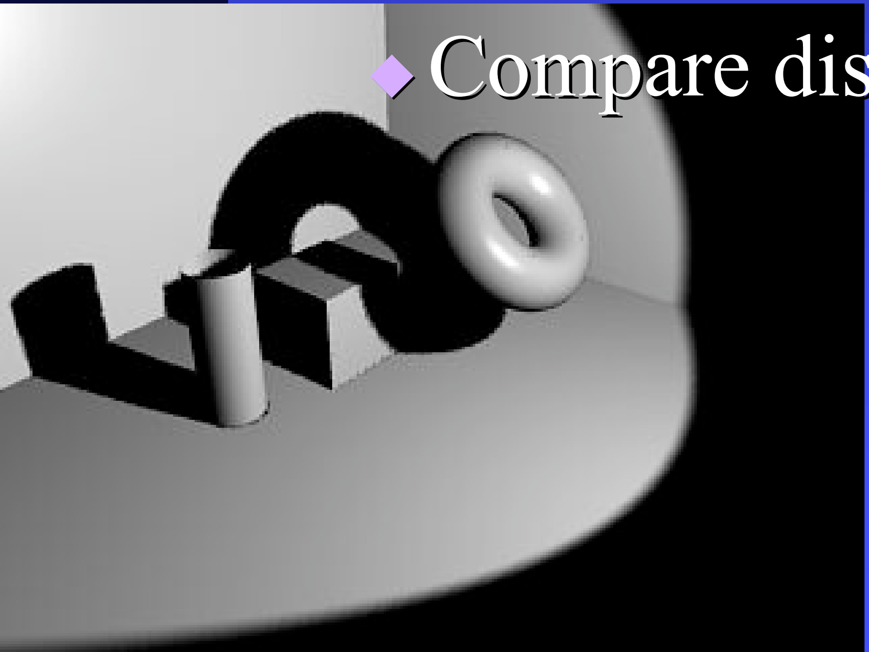
Shadow map

- In advance
  - Render scene from light position
  - Store depths in texture
    - Holds distance to lit surface, anything further than that is in shadow.



Shadow map

- To do shadow
  - Transform surface posns to *light space*
  - Project texture onto surface
  - Compare distance

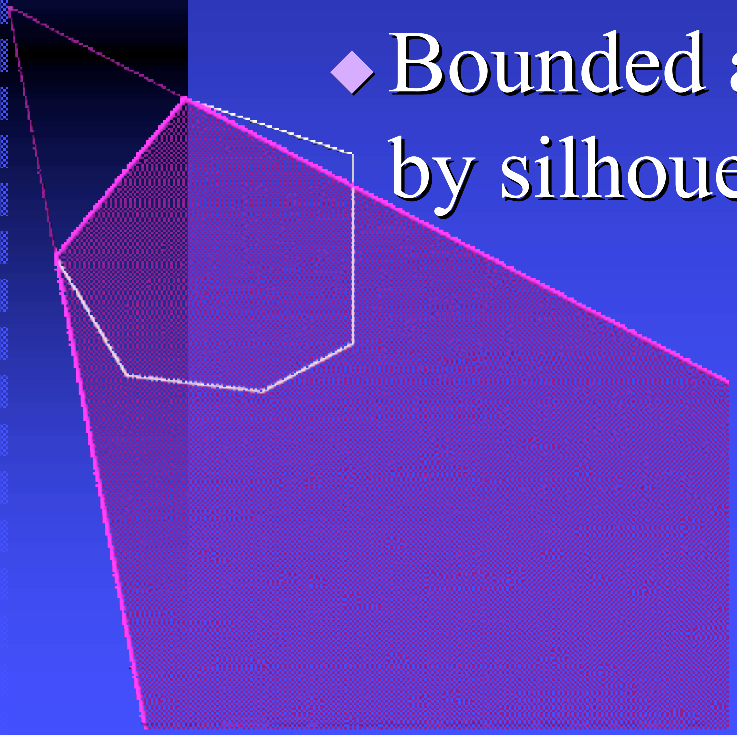


Shadow map

- Numeric problems
  - Lit surface mistakenly in shadow
  - Shadow bias: constant □in comparison
  - Store average of 1st & 2nd depth
- Filtering
  - Blending depth values does not work
  - Percentage closer filtering

Shadow volumes

- Occluder creates wedge of shadow
  - Bounded above by object
  - Bounded at sides by polygons defined by silhouette edge & light point



Stencil shadow volumes

- OpenGL *Stencil* increment & decrement
  - Draw object
  - Draw front-facing shadow polygons
    - Increment stencil for each
  - Draw back-facing shadow polygons
    - Decrement stencil for each
  - Non-zero stencil = in shadow

Soft shadows

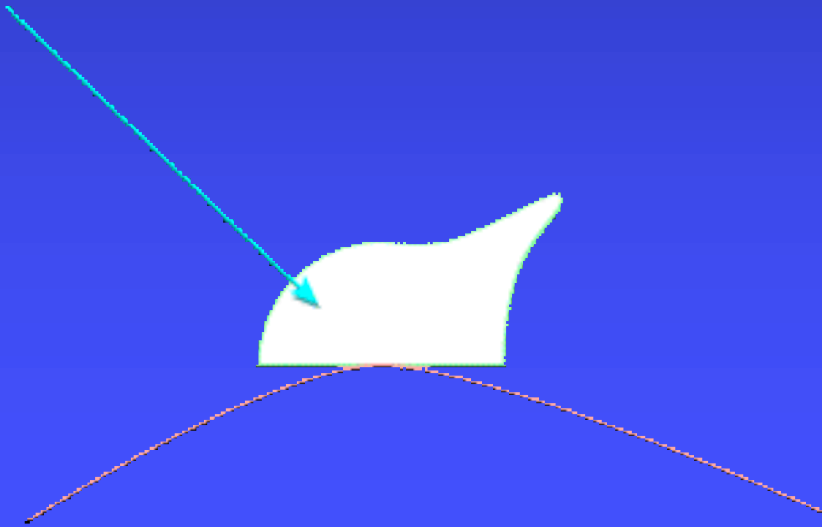
- Many point samplees
  - Monte-Carlo
- Shadow volumes with penumbra wedges

BRDF

- Bidirectional
  - Incoming & outgoing light directions
- Reflectance
  - Attenuation of reflected light
  - Not transmission or emission
- Distribution
  - Light in distributed to outgoing directions
  - Don’t create new light
- Function

BRDF

- In terms of local surface coordinates
  - Only above surface
  - Direction: □, □ or U, V (N)
  - f($\square_i$, $\square_i$, $\square_o$, $\square_o$)
- Polar/spherical plot

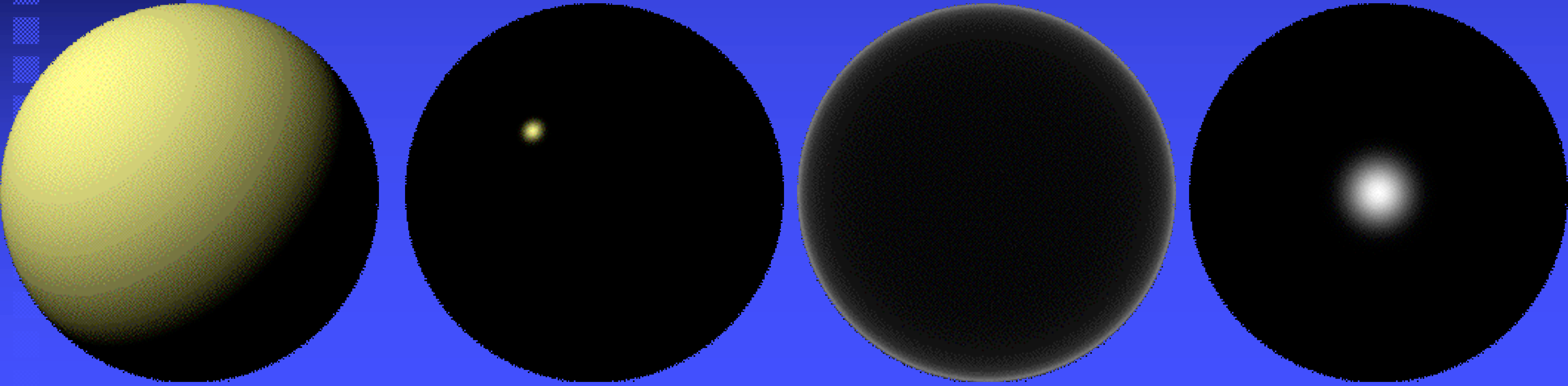


Physically plausible BRDF

- Positive everywhere
  - No negative light
- Conservation of Energy
  - No more light out than you put in
  - $\int f(V,L)\, dL \leq 1$
- Reciprocity
  - No one-way light valves
  - $f(V,L) = f(L,V)$

Decomposition

- Often decompose into components
  - $f_{\text{diffuse}} + f_{\text{specular}} + f_{\text{Fresnel}} + f_{\text{retroreflect}} + \ldots$

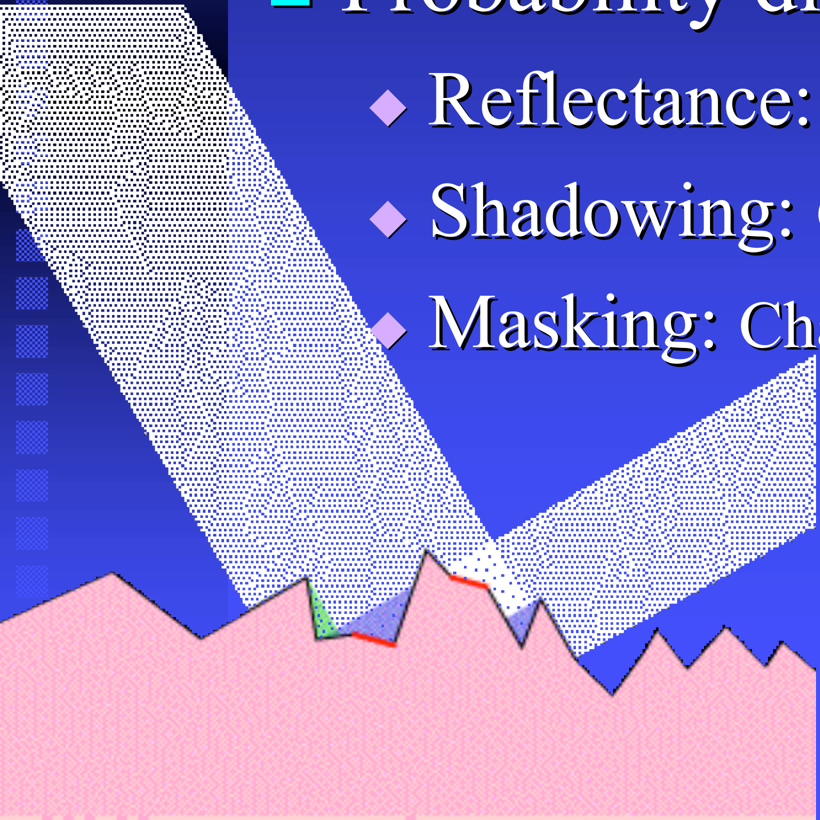


Rendering Equation

- $I(\square_o,\square_o) = \int f(\square_i,\square_i,\square_o,\square_o)\, I_L(\square_i, \square_i) \cos \square_i \, d\square_i \, d\square_i$
  - Add up all the light, modulated by BRDF
  - $\int \ldots \cos \square_i \, d\square_i \, d\square_i$ = spherical integration
  - $\cos \square_i = N \cdot L$
- $f_{\text{diffuse}} = 1/\pi$
- $f_{\text{Phong}} = R \cdot L^e / N \cdot L$
- $f_{\text{Blinn-Phong}} = N \cdot H^e / N \cdot L$

Microfacet models

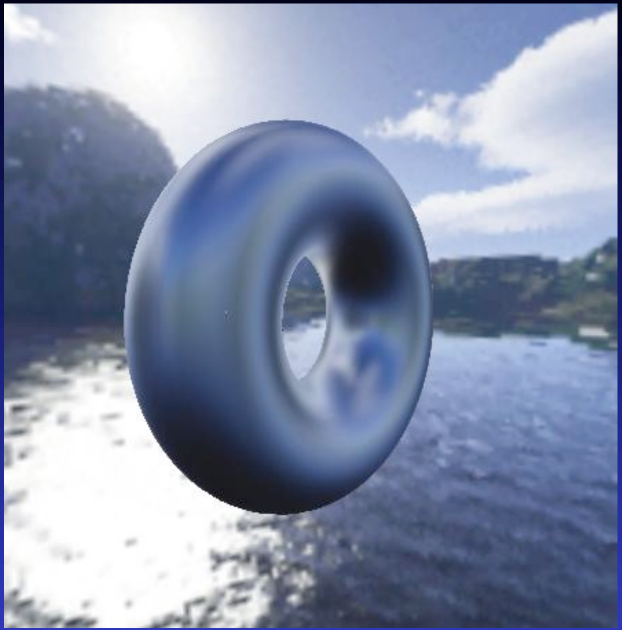
- Microscopic reflective facets
- Probability distributions
  - Reflectance: Chance a facet has normal H=V+L
  - Shadowing: Chance another facet blocks L
  - Masking: Chance another facet blocks V



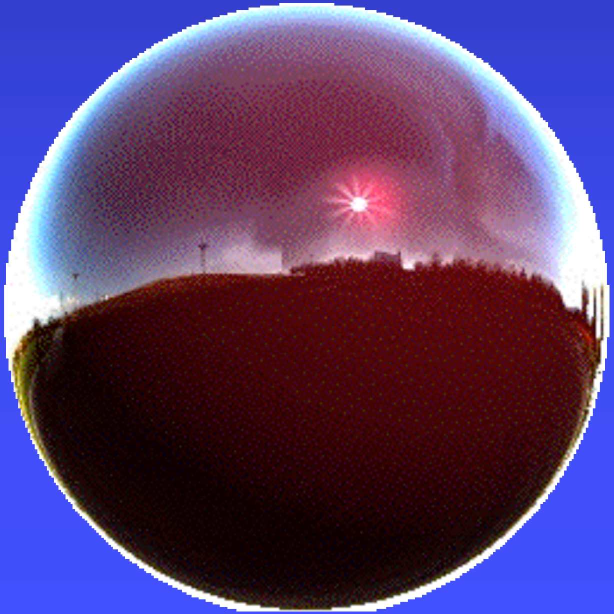
Cook-Torrance

- Symmetric V facets
- $F\,D\,G / (\pi\; N \bullet V\; N \bullet L)$
  - Fresnel, Distribution, Geometry
- Beckmann Distribution
  - $\exp(-\tan^2 \square / m^2) / (4\, m^2 \cos^4 \square)$
  - Gaussian distribution of facet slope

Reflectance map



- Diffuse: I(N) = texture
- Specular: I(H) = texture
  - Filtered environment map
  - BRDF as Filter

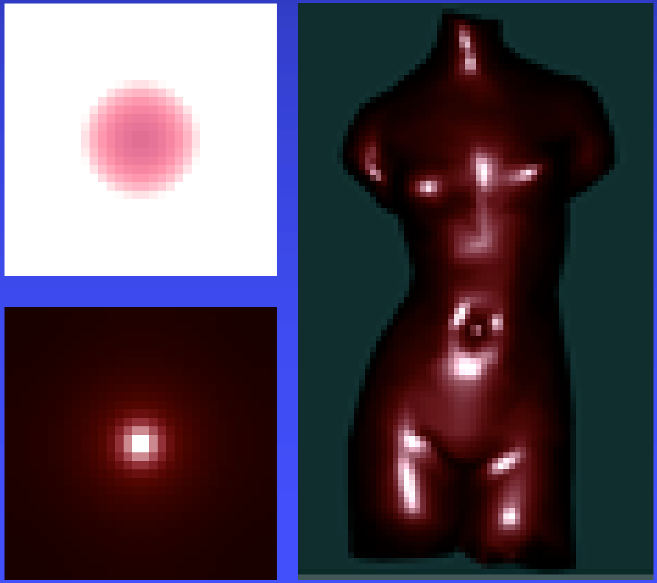


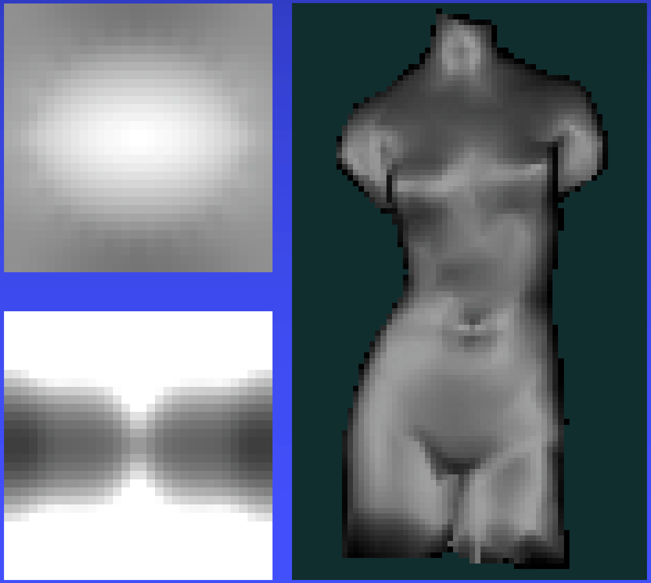
Homomorphic Factorization

- $f(V,L) = f_0(v_0)\ f_1(v_1)\ f_2(v_2) \ldots f_n(v_n)$
- Pick $v_0 \ldots v_n$, functions of V & L
- $\log(f) = \log(f_0\ f_1\ f_2 \ldots f_n)$
  - $= \log(f_0) + \log(f_1) + \log(f_2) + \ldots + \log(f_n)$
  - + smoothness terms
  - Solve for elements of $\log(f_i)$
    - Big least-squares problem
  - Use $\exp(\log(f_i))$ as texture & v_i as texture coordinates

Homomorphic + Microfacet

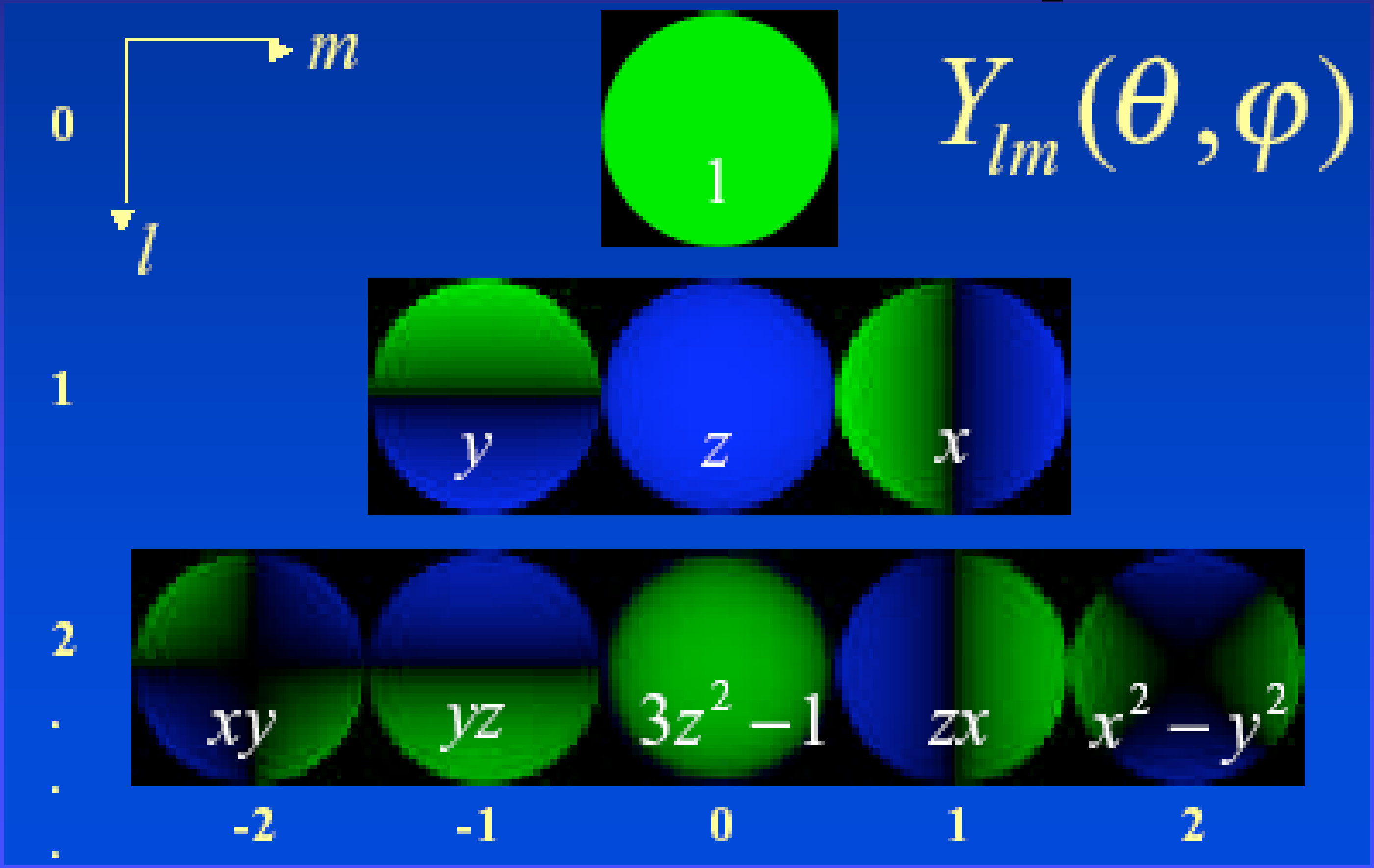
- Factor into f(V), f(H), f(L)
- f(V) = masking = f(L) = shadowing
- f(H) = reflectance





Spherical harmonics

- Like Fourier transform for spheres



Spherical harmonics

- Simulate lighting using harmonic basis functions as lighting environment
  - Take as long as necessary to find reflectance, shadowing, multi-bounce, etc.
- Store results in separate texture for each basis
- Decompose real environment into SH basis
- Scale per-pixel texture results by SH basis coefficients

Spherical harmonics