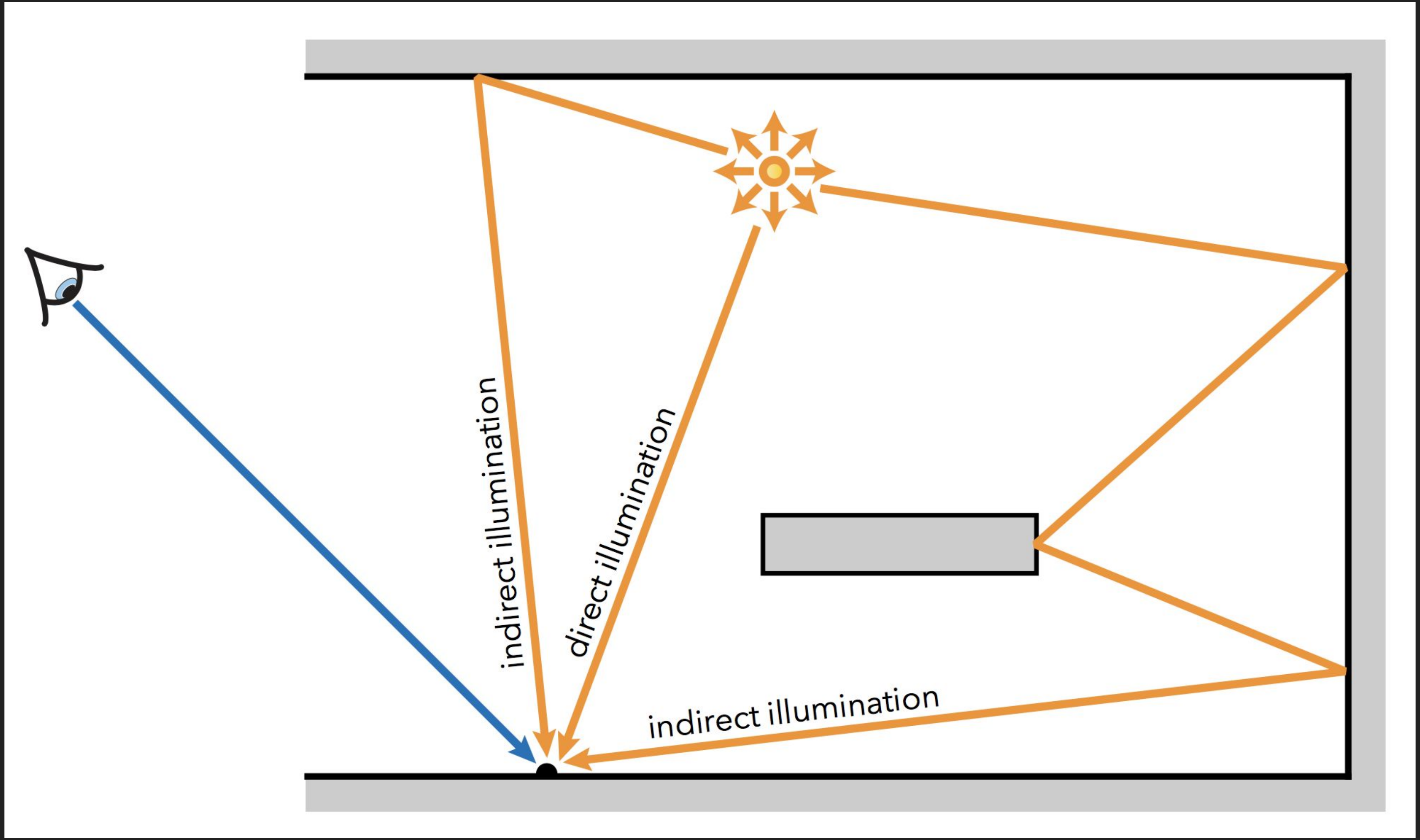
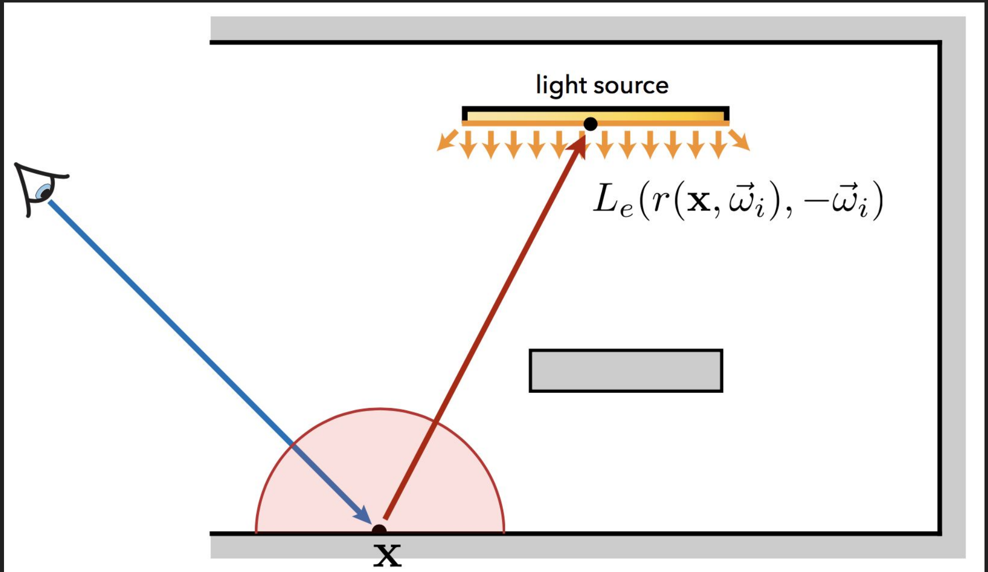
Illumination



Direct Illumination

$$L_r(\mathbf{x}, \vec{\omega}_r) = \int_{\boxed{\Omega}} f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_r) L_e(r(\mathbf{x}, \vec{\omega}_i), -\vec{\omega}_i) |\cos\theta_i| d\vec{\omega}_i$$



Direct Illumination

- Illumination problem > Solved using ray tracing and Monte-Carlo

$$\langle L_r(\mathbf{x},\vec{\omega}_r)^N\rangle = \frac{1}{N}\sum_{k=1}^{N}\frac{f_r(\mathbf{x},\vec{\omega}_{i,k},\vec{\omega}_r)L_e(r(\mathbf{x},\vec{\omega}_{i,k}),-\vec{\omega}_{i,k})\cos\theta_{i,k}}{p_\Omega(\vec{\omega}_{i,k})}$$

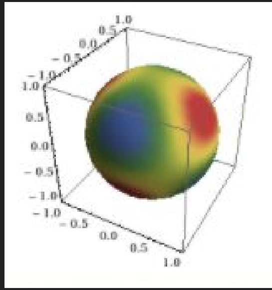
- Evaluate the integral at random values and use probabilities to make sure it converges

Precomputed Radiance Transfer (PRT)

- **Global Illumination** (soft shadows, interreflections, caustics) at **run-time**
  - No ray tracing needed at run-time
  - Independent of the number of lights
  - Static scene with dynamic lights
  - Simple BRDF like Diffuse & Specular
- Taking advantage of some properties of **Spherical Harmonics** (SH)
  - Rendering becomes a simple **dot product**!

Spherical Harmonics (SH)

- Function that assigns a **value** to every point on the surface of a unit sphere



- Used to represent a function defined on the surface of a sphere (like radiance)

Spherical Harmonics

F(x) = sum of spherical functions

Fourier Transform

F(x) = sum of circular functions (sinusoids)

Projection

$$f_l^m = \int f(s)\, y_l^m(s)\, ds$$

$$S(f) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} s(t) \cdot e^{-2i\pi ft}\, dt$$

Reconstruction

$$\tilde{f}(s) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} f_l^m \; y_l^m(s)$$

$$s(t) = \int_{-\infty}^{\infty} S(f) \cdot e^{2i\pi ft}\, df$$

Spherical Harmonics (SH)

Useful property:

$$\int \tilde{a}(s)\, \tilde{b}(s)\, ds = \sum_{i=1}^{n^2} a_i\, b_i$$

n-th order involves n^2 coefficients

$$i = l(l+1) + m + 1$$

$$\tilde{f}(s) = \sum_{i=1}^{n^2} f_i\, y_i(s)$$

SH (Definition)

Complex

$$Y_l^m(\theta,\varphi) = K_l^m e^{im\varphi} P_l^{|m|}(\cos\theta), l \in \mathbf{N}, -l \le m \le l$$

P_l^m are the associated Legendre polynomials

K_l^m are the normalization constants

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$$

Real

$$y_l^m = \begin{cases} \sqrt{2}\mathrm{Re}(Y_l^m) & m > 0 \\ \sqrt{2}\mathrm{Im}(Y_l^m) & m < 0 \\ Y_l^0 & m = 0 \end{cases} = \begin{cases} \sqrt{2}K_l^m \cos m\varphi \; P_l^m(\cos\theta) & m > 0 \\ \sqrt{2}K_l^m \sin|m|\varphi \; P_l^{|m|}(\cos\theta) & m < 0 \\ K_l^0 P_l^0(\cos\theta) & m = 0 \end{cases}$$

$$P_0^0(z) = 1,$$

$$P_m^m(z) = (2m-1)!!(1-z^2)^{m/2},$$

$$P_{m+1}^m(z) = z(2m+1)P_m^m(z),$$

$$P_l^m(z) = \frac{z(2l-1)}{l-m}P_{l-1}^m(z) - \frac{(l+m-1)}{l-m}P_{l-2}^m(z)$$

SH (Implementation)

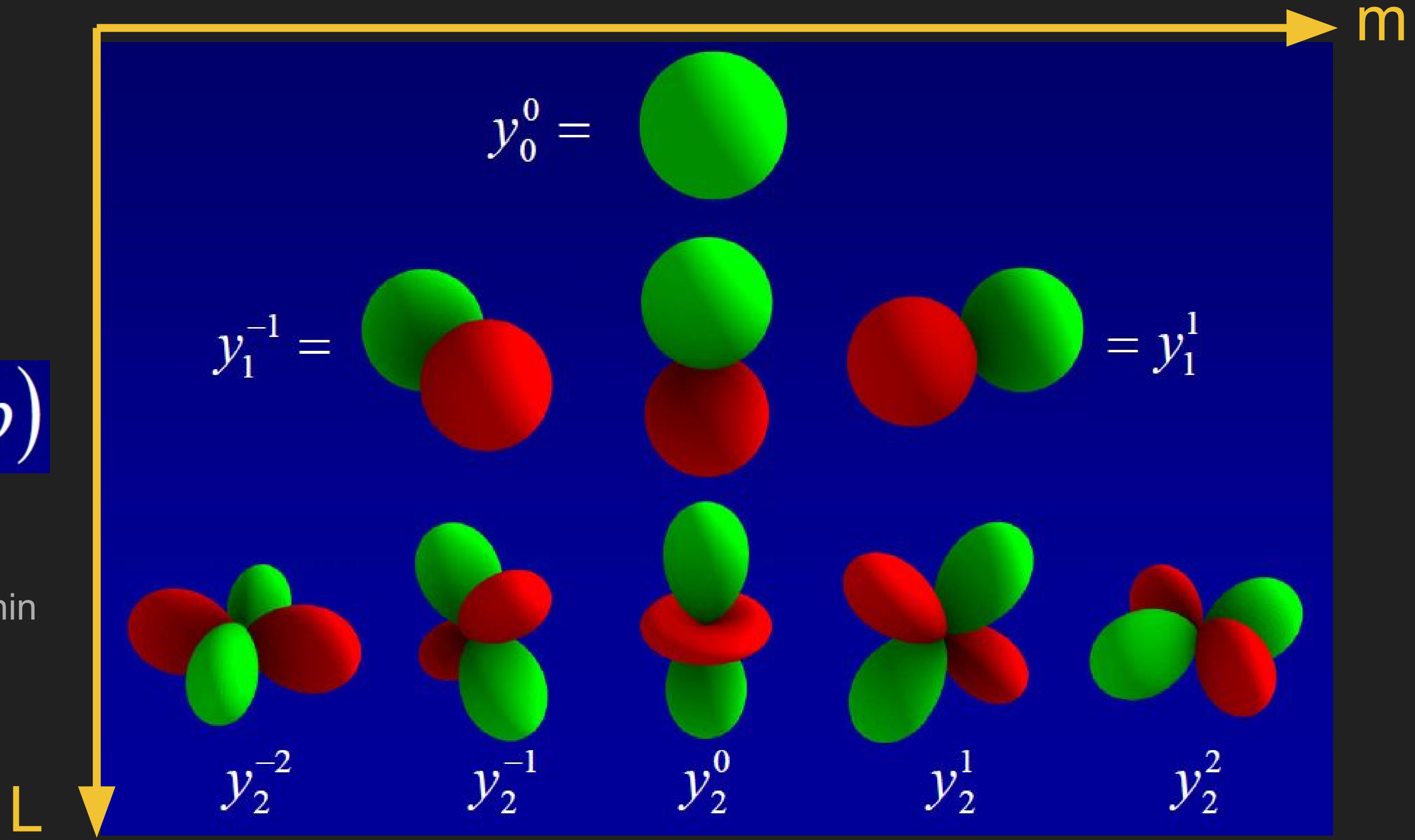
```
var fC0,fC1,fS0,fS1,fTmpA,fTmpB,fTmpC;
var fZ2 = fZ*fZ;
var pSH = new Array(9);

pSH[0] = 0.2820947917738781;
pSH[2] = 0.4886025119029199*fZ;
pSH[6] = 0.9461746957575601*fZ2 + -0.3153915652525201;
fC0 = fX;
fS0 = fY;

fTmpA = -0.48860251190292;
pSH[3] = fTmpA*fC0;
pSH[1] = fTmpA*fS0;
fTmpB = -1.092548430592079*fZ;
pSH[7] = fTmpB*fC0;
pSH[5] = fTmpB*fS0;
fC1 = fX*fC0 - fY*fS0;
fS1 = fX*fS0 + fY*fC0;

fTmpC = 0.5462742152960395;
pSH[8] = fTmpC*fC1;
pSH[4] = fTmpC*fS1;
```

SH (Graphical representation)

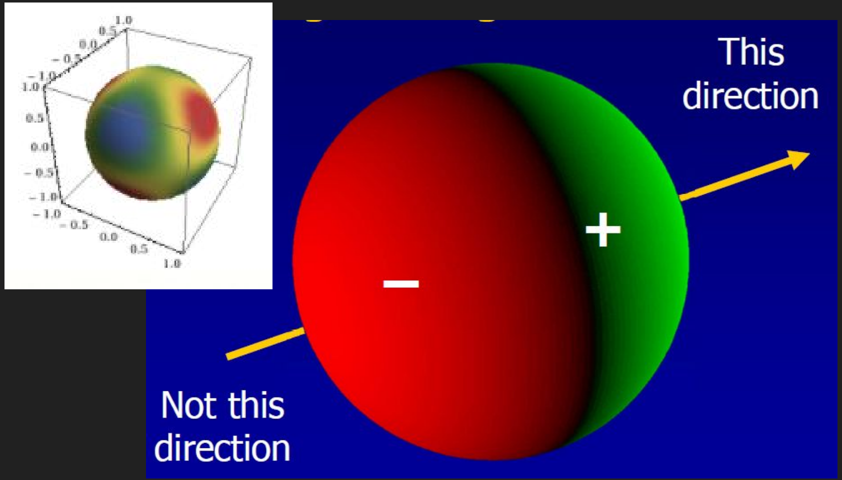


$$y_l^m(\theta, \varphi)$$

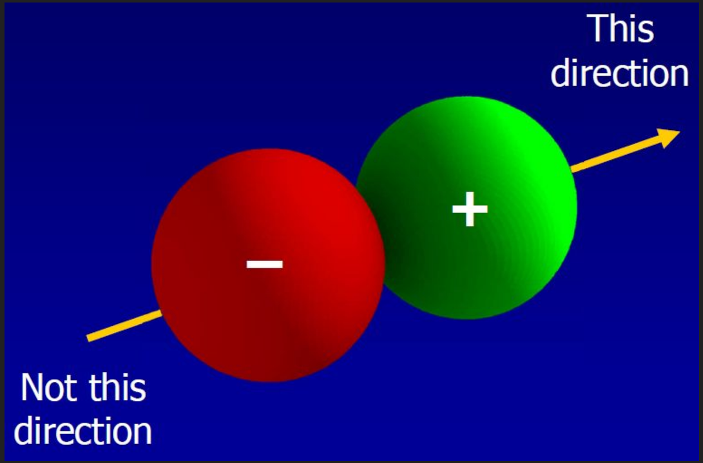
L = band

m = index within the band

Two graphical representations

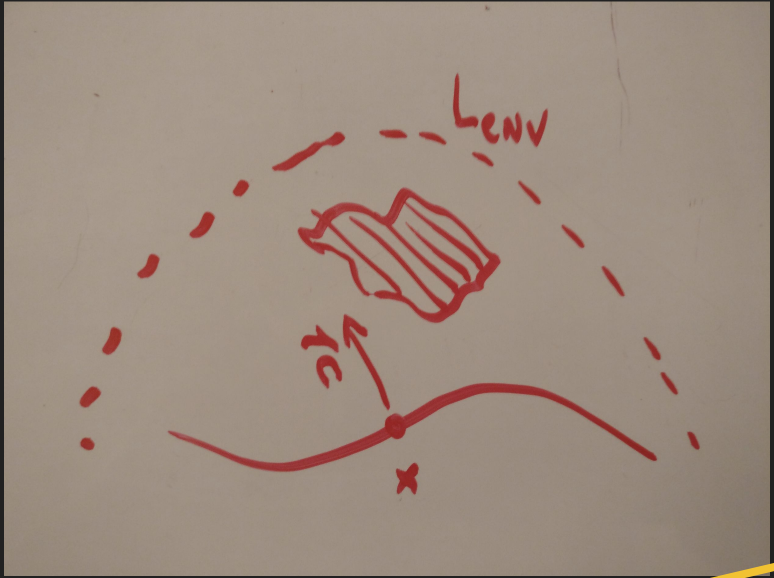


3D plot where color = value of SH



Sphere deformation according to the SH value

PRT



$$L_o = \int_{\Omega} f_r \cdot L_i \cdot \cos\theta_i \cdot d\vec{\omega}_i$$

$$= \int_{\Omega} \boxed{\frac{\rho}{\pi}} \cdot \boxed{L_{env}} \cdot \boxed{V \cdot \lfloor \vec{\omega}_i \cdot \vec{n} \rfloor} d\vec{\omega}_i$$

$$l_i = \int_{S^2} L_{env} \cdot y_i(\vec{\omega})\, d\vec{\omega}$$

$$g_i = \frac{\rho}{\pi} \int_{S^2} V \cdot \lfloor \vec{\omega} \cdot \vec{n} \rfloor \cdot y_i(\vec{\omega})\, d\vec{\omega}$$

PRT

$$f(x) = \sum l_i \overset{\text{SH}}{y_i}(\omega)$$

$$g(x) = \sum g_i y_i(\omega)$$

$$L_0 = \sum_k l_k g_k$$

PRT algorithm

1. **Precomputation**

$$l_i = \int_{S^2} L_{env} \cdot y_i(\vec{\omega})\, d\vec{\omega}$$

$$g_i = \frac{\rho}{\pi} \int_{S^2} V \cdot \lfloor \vec{\omega} \cdot \vec{n} \rfloor \cdot y_i(\vec{\omega})\, d\vec{\omega}$$

2. **Run-time**

$$L_o = \sum_k l_k g_k$$

Precomputation (Lights)

$$l_i = \int_{S^2} L_{env} \cdot y_i(\vec{\omega})\, d\vec{\omega}$$

- Projection of Sphere light onto an environment map = analytic solution

Precomputation (Visibility)

$$g_j = \frac{\rho}{\pi} \int_{S^2} V \cdot \lfloor \vec{\omega} \cdot \vec{n} \rfloor \cdot y_j(\vec{\omega}) \, d\vec{\omega}$$

$$= \frac{1}{N} \frac{\rho}{\pi} \sum_{k=1}^{N} \frac{V \cdot \lfloor \vec{\omega} \cdot \vec{n} \rfloor \cdot y_j(\vec{\omega})}{p(\vec{\omega})}$$

- Solved via Monte-Carlo
  - Sampling = uniform spherical
- n^2 coefficients / vertex

Precomputation (Visibility)

```
for(var obj : objects) {
    for(var v : obj.vertex) {
        obj.G[v] = MC(v);
    }
}
```

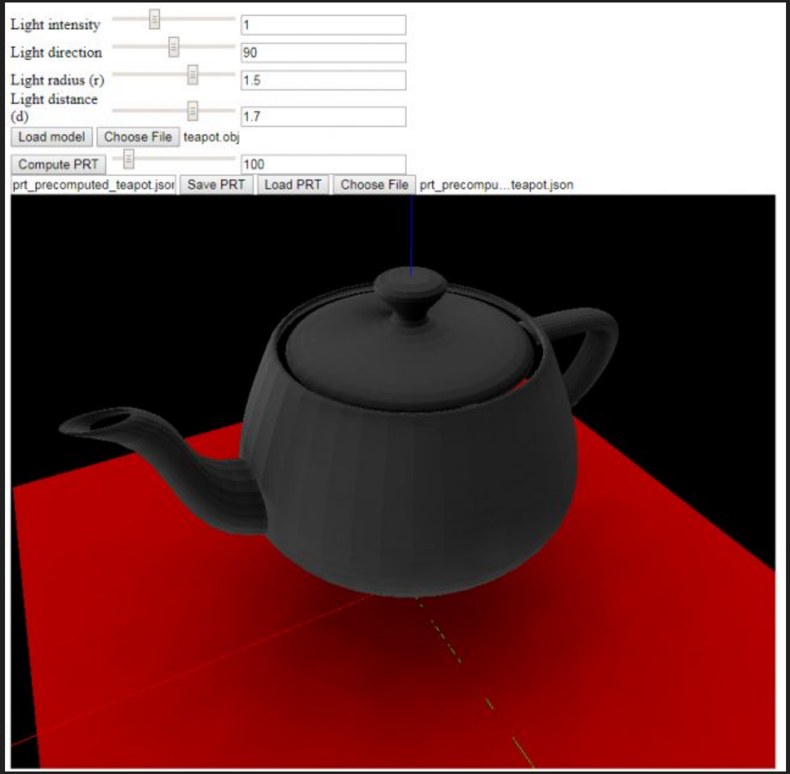
```
for(var i = 0; i < N; i++) {
    var w = sampleUniformSphere();
    if( !intersectRay(p,w) ) {
        var cosTheta = Math.max(0.0, w.dot(n));
        var yi = SHEval(w, N_ORDER);
        for(var k = 0; k < N_COEFFS; k++) {
            G[v][k] += cosTheta * yi[k];
        }
    }
}

var pWi = 1.0 / (4.0 * Math.PI);
for(var k = 0; k < N_COEFFS; k++) {
    G[v][k] /= (Math.PI * N * pWi);
}
```

Run-time

```
for(var obj : objects) {
    for(var v : obj.vertex) {
        v = obj.albedo * dot(obj.L, obj.G[v]);
    }
}
```

Implementation



- Using Three.js / WebGL
- 1 dynamic light
- 2 static objects
- Soft shadows

github.com/kevenv/prt