

ゲームグラフィックス特論

第10回 環境による照明

周囲の光による陰影

環境光

任意光源の放射測定

- \mathbf{l} 方向の微小立体角 $d\omega_i$ の放射輝度 (接空間 $\mathbf{t}, \mathbf{b}, \mathbf{n}$)

$$L_i(\mathbf{l}) = \frac{dE}{d\omega_i \overline{\cos \theta_i}}$$

- 物体表面上のこの光源による放射照度

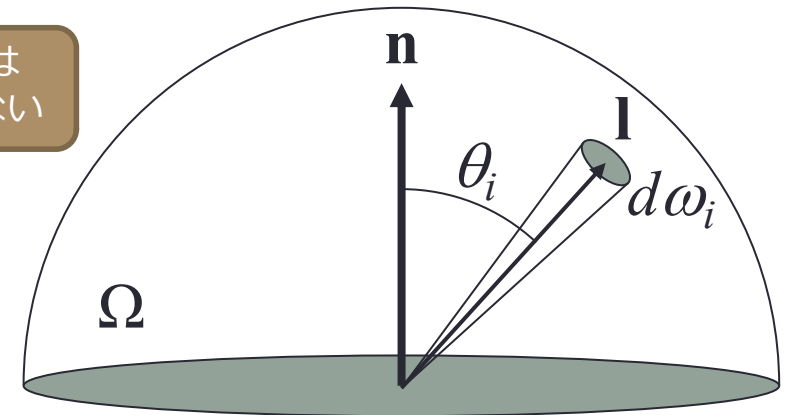
$$dE = L_i(\mathbf{l}) d\omega_i \overline{\cos \theta_i}$$

- したがって、物体表面上の天球全体からの放射照度は

天空上の放射輝度に入射角の余弦をかけたものの総和

$$E = \int_{\Omega} L_i(\mathbf{l}) d\omega_i \cos \theta_i$$

半球 Ω の範囲では
 $\cos \theta_i$ は負にならない



放射輝度

- BRDFの定義

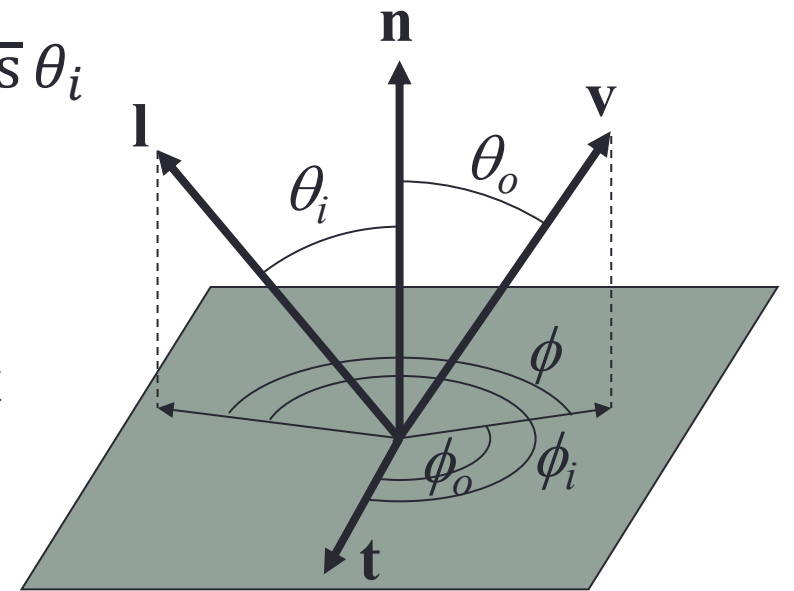
$$f(\mathbf{l}, \mathbf{v}) = \frac{dL_o(\mathbf{v})}{dE(\mathbf{l})}, \quad dE = L_i(\mathbf{l}) d\omega_i \overline{\cos \theta_i}$$

- \mathbf{v} 方向への放射輝度 $L_o(\mathbf{v})$ は

$$L_o(\mathbf{v}) = \int_{\Omega} f(\mathbf{l}, \mathbf{v}) \otimes L_i(\mathbf{l}) \cos \theta_i d\omega_i$$

- 極座標で表せば

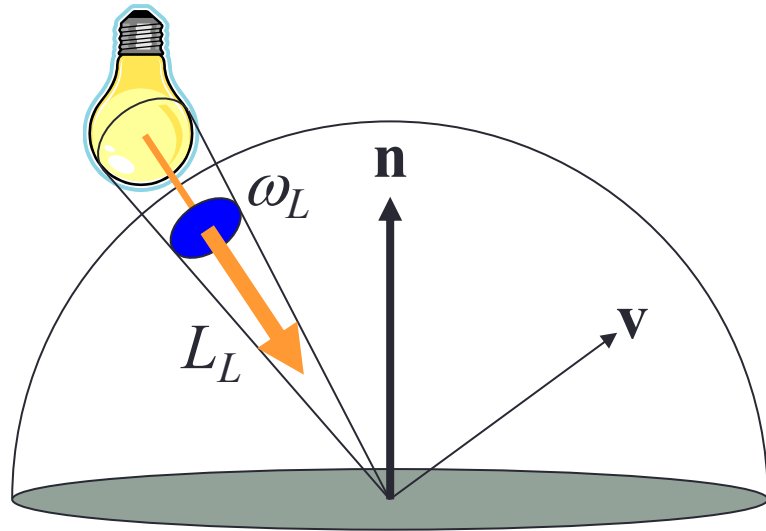
$$L_o(\theta_o, \phi_o) = \int_{\phi_i=0}^{2\pi} \int_{\theta_i=0}^{\frac{\pi}{2}} f(\theta_i, \phi_i, \theta_o, \phi_o) \otimes L_i(\theta_i, \phi_i) \cos \theta_i \sin \theta_i d\theta_i d\phi_i$$



$d\phi_i$ は $\sin \theta_i$ に比例する ($d\omega_i = \sin \theta_i d\theta_i d\phi_i$)

面光源

- 天球の一部を切り取ったもの

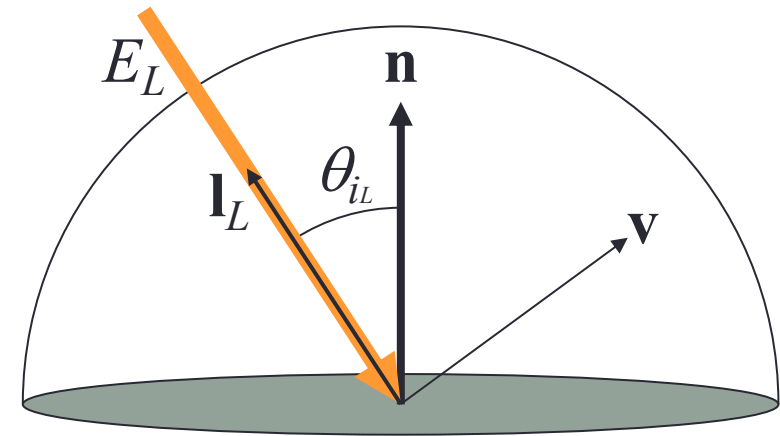


面光源

定数とみなす

$$L_o(\mathbf{v}) = \int_{\omega_L} f(\mathbf{l}, \mathbf{v}) \otimes L_L \cos \theta_i d\omega_i \approx E_L \cos \theta_{iL}$$

面光源の面積



点光源 / 平行光線

この近似誤差は光源が小さいか
物体表面が粗ければ小さくなる

完全拡散反射面

- 完全拡散反射面の放射輝度は放射照度に比例する

$$L_o(\mathbf{v}) = \frac{c_{diff}}{\pi} E \quad c_{diff} \text{ は面の拡散反射色}$$

定数

- したがって、この放射照度は

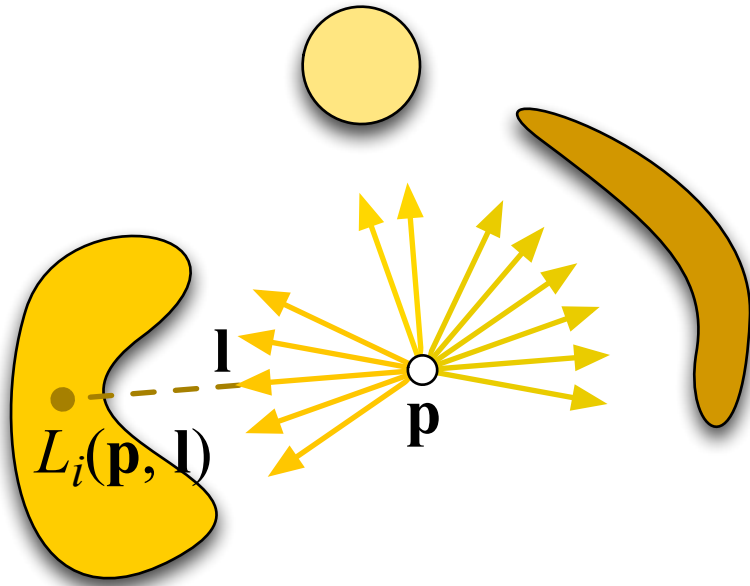
$$E = \int_{\omega_L} L_L \overline{\cos\theta_i} d\omega_i$$

$$\approx E_L \overline{\cos\theta_{iL}}$$

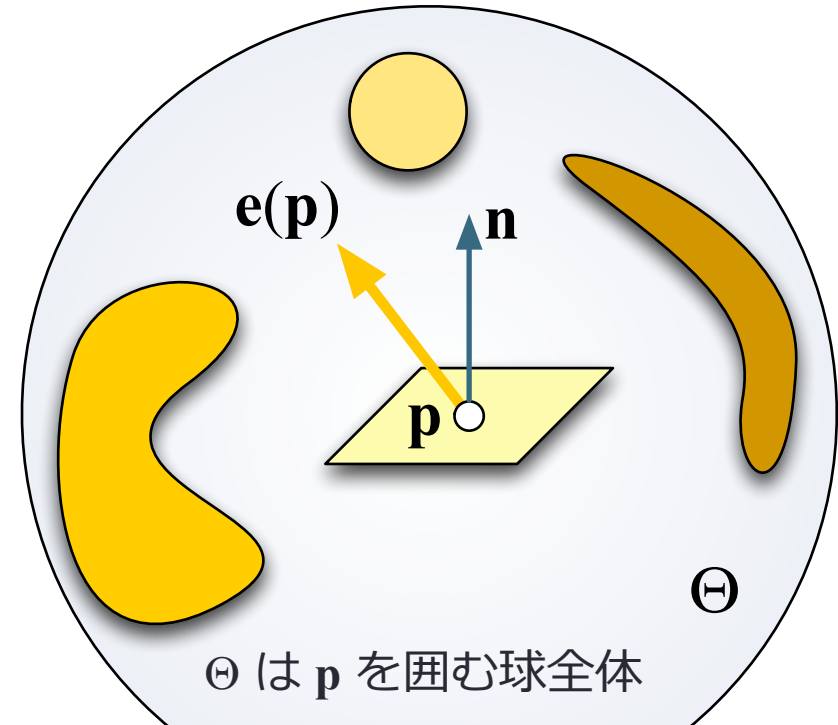
完全拡散反射面の放射照度は
入射光の放射照度に入射角の
余弦をかけたもの

Lambert の余弦法則

ベクトル放射照度



点 p は様々な形や大きさや放射輝度分布の光源に囲まれている



Θ は p を囲む球全体

これを e の方向から到来する光に集約する (ベクトル放射照度)

$L_i(\mathbf{p}, \mathbf{l})$: p から \mathbf{l} 方向の放射輝度

ベクトル

$$\mathbf{e}(\mathbf{p}) = \int_{\Theta} L_i(\mathbf{p}, \mathbf{l}) d\omega_i$$

ベクトル放射照度による放射照度



正味の放射照度

$$E(\mathbf{p}, \mathbf{n}) - E(\mathbf{p}, -\mathbf{n}) = \mathbf{n} \cdot \mathbf{e}(\mathbf{p})$$

表面 (\mathbf{n} の方向) の放射照度から裏面の放射照度を引いたもの

裏面に光が当たらなければ $E(\mathbf{p}, -\mathbf{n}) = 0$

$$E(\mathbf{p}, \mathbf{n}) = \mathbf{n} \cdot \mathbf{e}(\mathbf{p})$$

ベクトル放射照度と波長

- ベクトル放射照度は単一の波長が対象
 - 異なる色の光源が与えられたとき
 - 集約したベクトルは波長ごとに異なる方向を向く
 - 全ての光源が同じスペクトル分布のとき
 - ベクトルは全て同じ方向を向く
- したがって

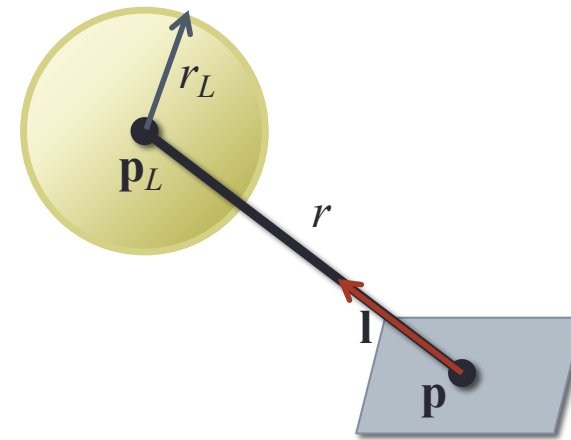
$$\mathbf{l} = \frac{\mathbf{e}(\mathbf{p})}{|\mathbf{e}(\mathbf{p})|}, \quad E_L = c_L |\mathbf{e}(\mathbf{p})|$$

c_L は光源色

- 中心 \mathbf{p}_L 半径 r_L の球体の光源 (表面の放射輝度 L_L)

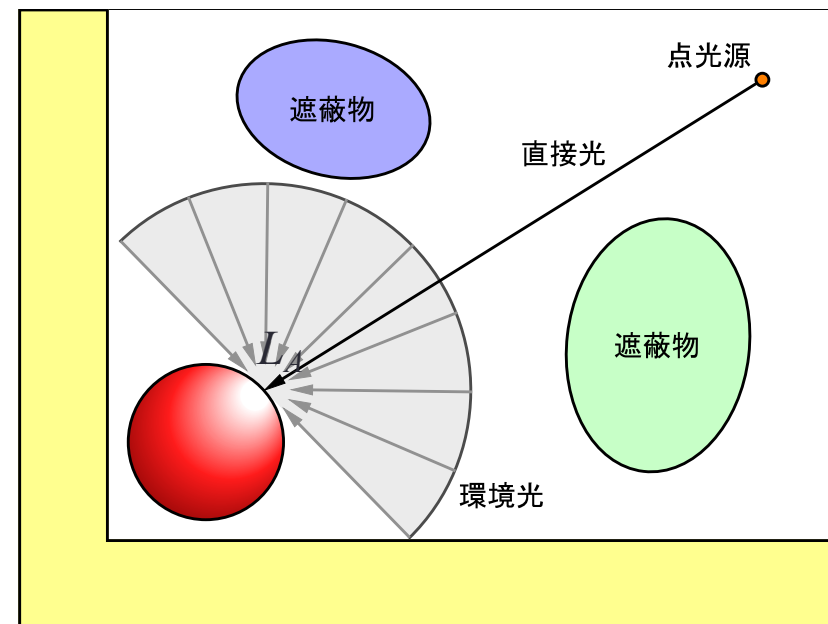
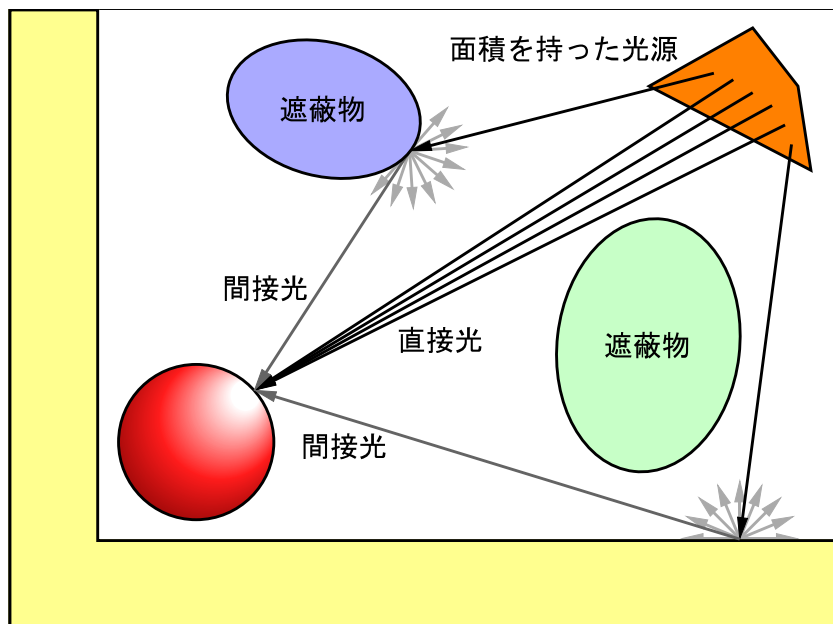
$$\mathbf{l} = \frac{\mathbf{p}_L - \mathbf{p}}{|\mathbf{p}_L - \mathbf{p}|}, \quad E_L = \frac{\pi r_L^2}{r^2} L_L$$

放射照度は光源の半径の二乗に比例し距離の二乗に反比例する



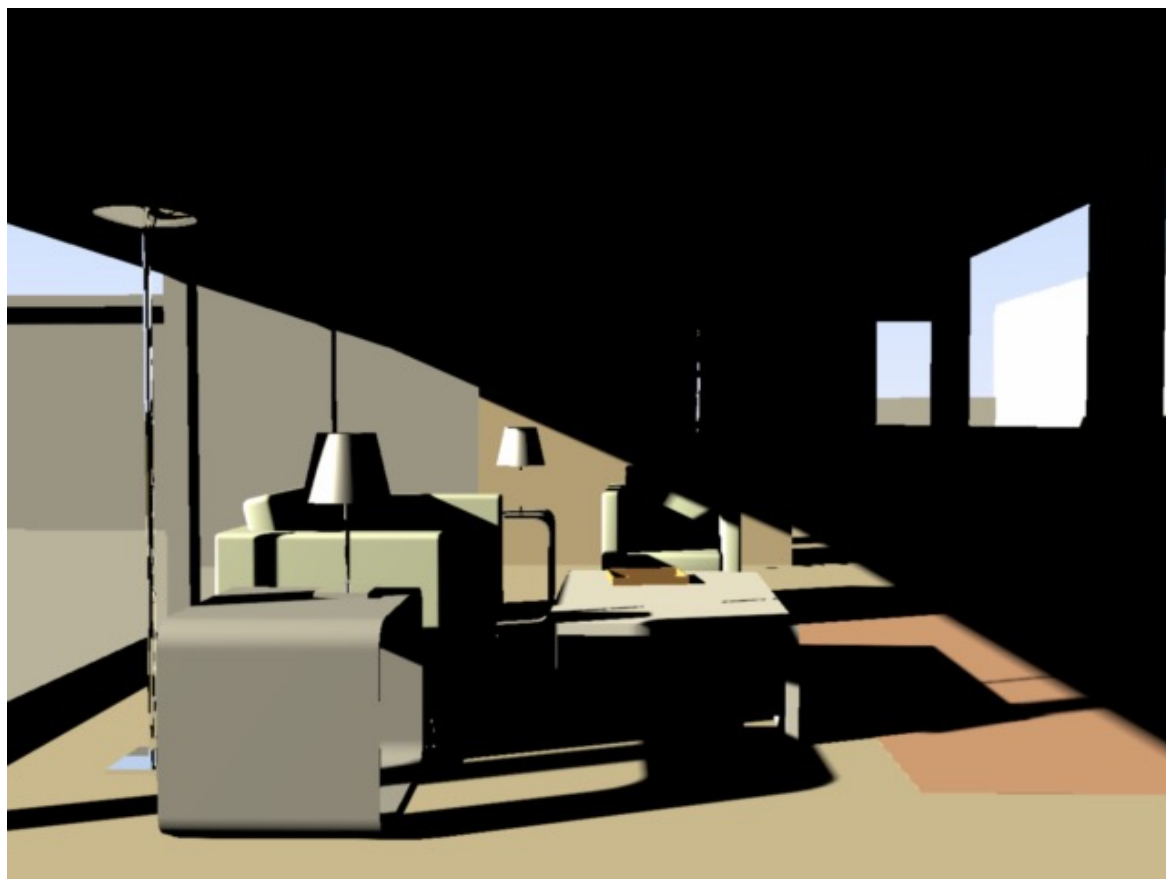
環境光

- 間接光の放射輝度が到来方向に対して不変であるとみなす
 - 放射輝度は定数で表される → L_A
- 間接光がないシーンは全然リアルではない
 - 物体の光が当たらない部分は完全に黒



環境光

環境光なし



環境光あり



環境光による放射輝度

- 放射輝度は面の法線 \mathbf{n} や視線 \mathbf{v} に無関係

$$\begin{aligned}
 L_o(\mathbf{v}) &= \int_{\Omega} f(\mathbf{l}, \mathbf{v}) \otimes L_i(\mathbf{l}) \cos \theta_i d\omega_i \\
 &= \frac{c_{diff}}{\pi} \otimes L_A \int_{\Omega} \cos \theta_i d\omega_i \\
 &= c_{diff} \otimes L_A
 \end{aligned}$$

π

- シェーディングの際はこれを直接光の成分に加える

$$L_o(\mathbf{v}) = \frac{c_{diff}}{\pi} \otimes \left(\pi L_A + \sum_{k=1}^n E_{L_k} \overline{\cos \theta_{i_k}} \right)$$

環境光による放射輝度

- 任意の BRDF に対して

$$L_o(\mathbf{v}) = L_A \int_{\Omega} f(\mathbf{l}, \mathbf{v}) \cos \theta_i d\omega_i + \sum_{k=1}^n f(\mathbf{l}_k, \mathbf{v}) E_{L_k} \overline{\cos \theta_{i_k}}$$

環境光
光源

- 環境光反射率 $R_A(\mathbf{v})$

$$R_A(\mathbf{v}) = \int_{\Omega} f(\mathbf{l}, \mathbf{v}) \cos \theta_i d\omega_i$$

環境光も BRDF
の影響を受ける

環境光の反射率は
視点に依存する

環境光による放射輝度

- 環境光の反射係数 c_{amb}



$$L_o(\mathbf{v}) = c_{amb} \otimes L_A + \sum_{k=1}^n f(\mathbf{l}_k, \mathbf{v}) E_{L_k} \overline{\cos\theta}_{i_k}$$

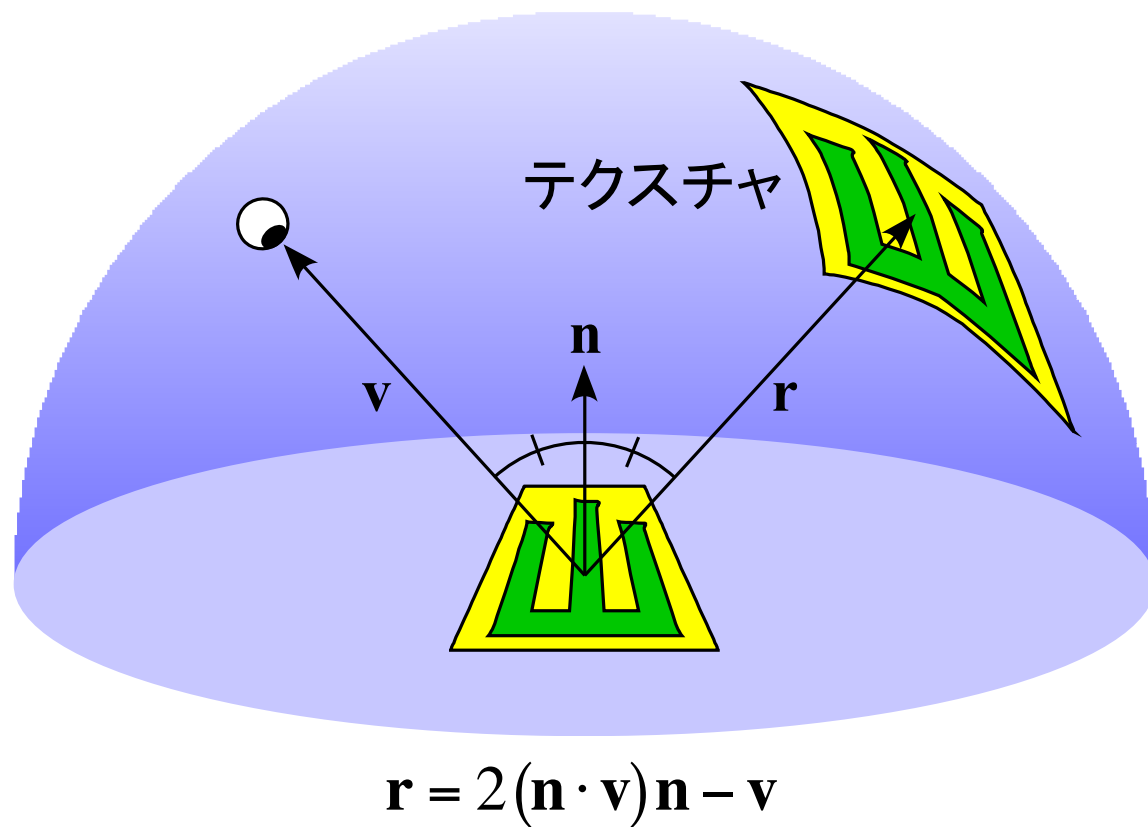
環境マッピング

周囲の光源のテクスチャによる表現

環境マッピング (Environment Mapping)

- 環境（周囲の情景）をテクスチャとしてマッピングする
 - 曲面への映り込みを擬似的に表現する
 - 反射マッピング (Reflection Mapping)
- 手順の概略
 - 視点から発射された視線が物体表面上で反射した方向を求める
 - 反射方向に置かれたテクスチャを標本化して、物体表面上の反射点の陰影計算に使う
- 手法
 - Blinn と Newell の方法
 - Sphere Mapping
 - Cube Mapping
 - Dual Paraboloid Mapping

Environment Mapping の例

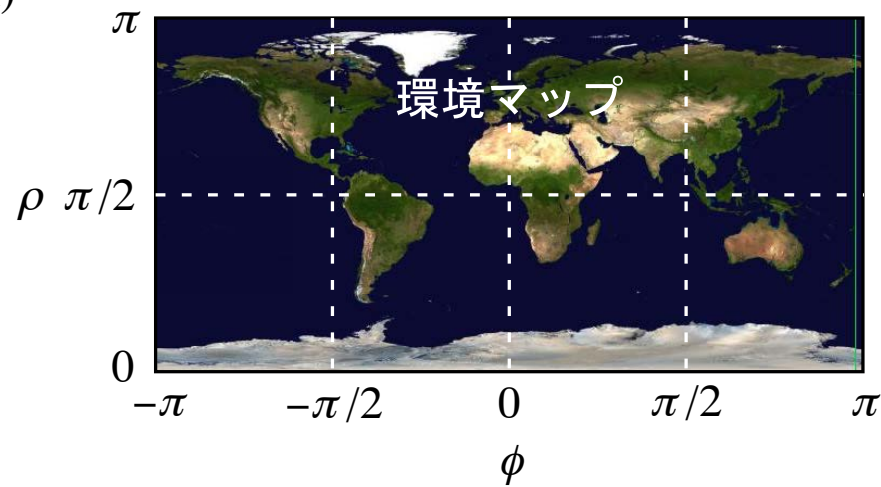
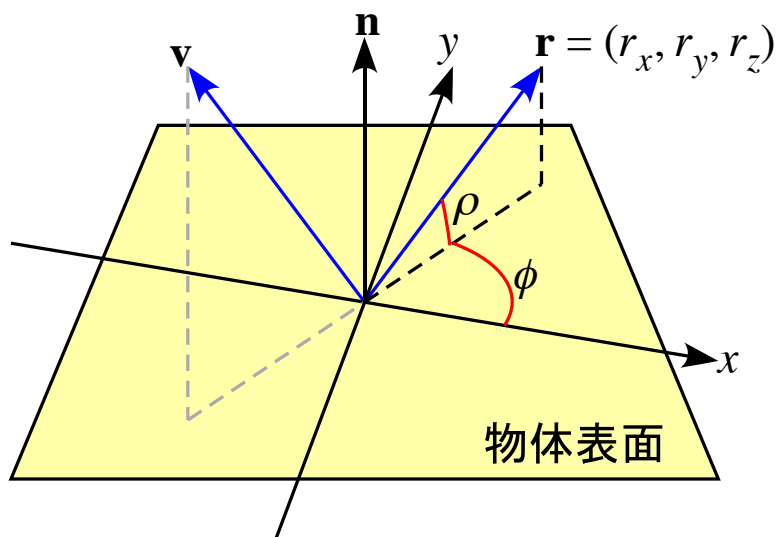


ポットの注ぎ口が写り込んでいない

Blinn と Newell の方法

- 反射ベクトルを極座標に変換する
 - 極座標のパラメータを使ってテクスチャを標本化
 - テクスチャ画像は「正距円筒図法」

$$\mathbf{r} = (r_x, r_y, r_z) = 2(\mathbf{n} \cdot \mathbf{v})\mathbf{n} - \mathbf{v} \longrightarrow \begin{cases} \rho = \text{acos}(-r_z) \\ \phi = \text{atan2}(r_y, r_x) \end{cases}$$



バーテックスシェーダ

```
#version 410
in vec4 pv;           // 頂点位置
in vec3 nv;           // 頂点の法線ベクトル
uniform mat4 mw;      // モデルビュー変換行列
uniform mat3 mg;      // 法線変換行列
out vec2 t;           // テクスチャ座標
const float PI = 3.141593;

void main(void)
{
    vec3 v = -normalize(mw * pv).xyz;           // 視線ベクトル
    vec3 r = reflect(v, normalize(mg * nv)); // 視線の正反射方向
    float s = -acos(r.z) / PI;
    float t = atan(r.y, r.x) * 0.5 / PI + 0.5;
    t = vec2(s, t);
    ...
}
```

reflect() はベクトル v の法線 n に対する
正反射方向を求める組み込み関数

GLSL の atan() は C 言語の atan2() に相当する

Sphere Mapping

- 球体への映り込みを環境マップに使う
 - テクスチャは球状の鏡を撮影して得られる
 - カメラが映りこんでしまう
 - 視線方向から見た映り込みのテクスチャを使用する
 - 視線方向を変更できない



テクスチャ画像

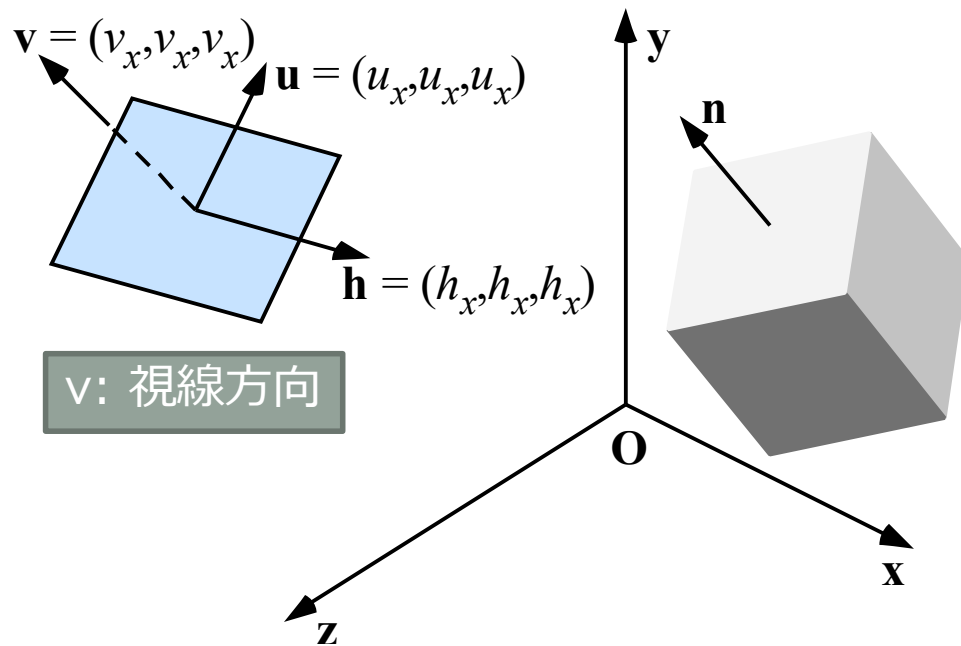


球に環境マッピング



ポットに環境マッピング

Sphere Mapping の基底行列



v: 視線方向

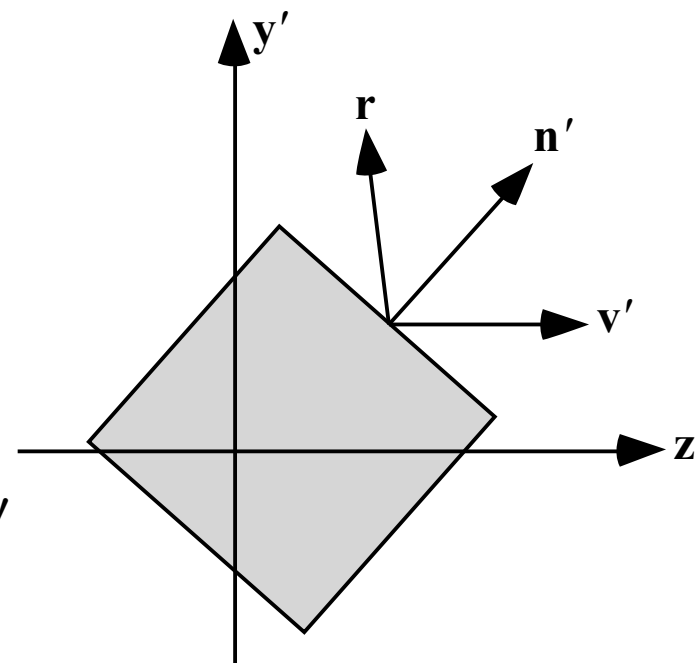


Sphere Mapping の基底行列

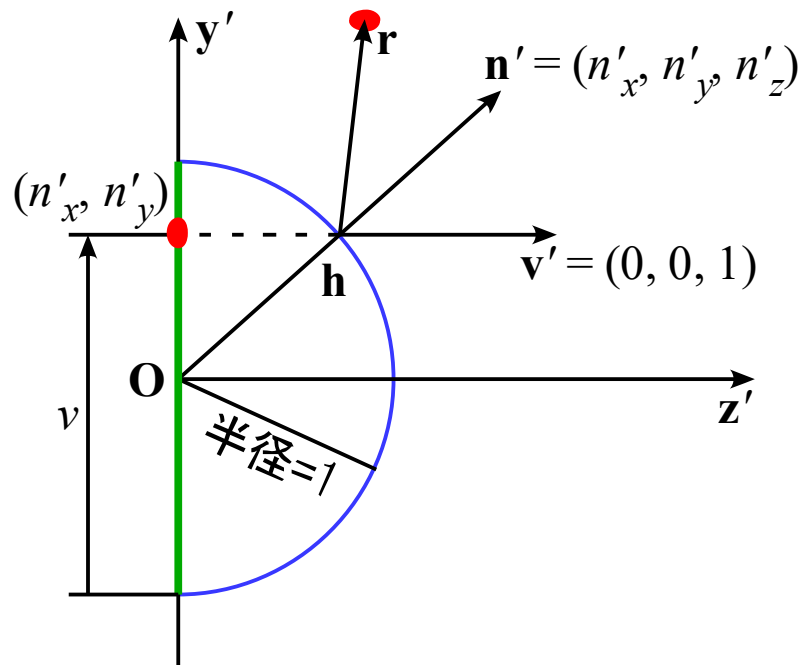
$$\mathbf{M}_s = \begin{pmatrix} h_x & h_y & h_z & 0 \\ u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

z 軸が視線 v 方向を
向くよう物体を回転
したときの法線ベク
トルを求める

$$\begin{aligned} \mathbf{v}' &= (0, 0, 1) \\ \mathbf{n}' &= \mathbf{M}_s \mathbf{n} \\ \mathbf{r} &= 2(\mathbf{n}' \cdot \mathbf{v}') \mathbf{n}' - \mathbf{v}' \end{aligned}$$



反射方向とテクスチャ座標



反射ベクトル

$$\mathbf{r} = (r_x, r_y, r_z)$$

視線ベクトル

$$\mathbf{v}' = (0, 0, 1)$$

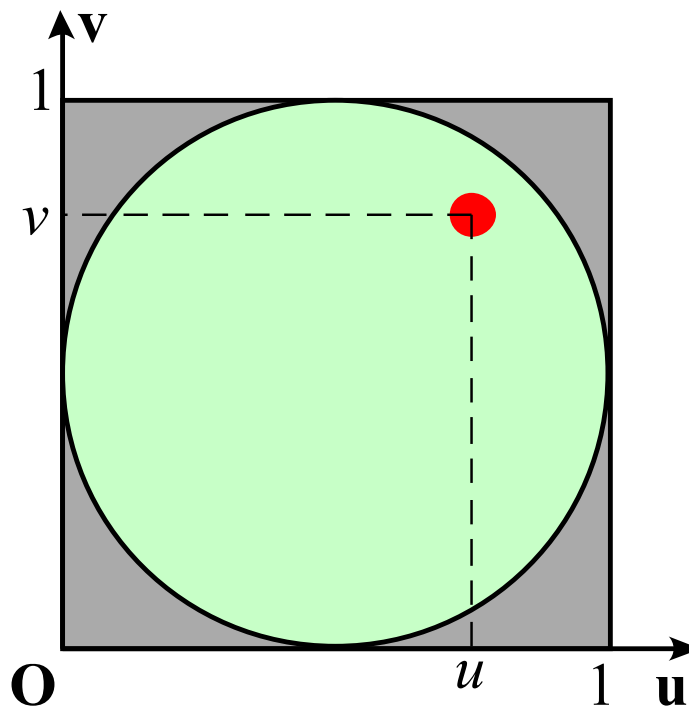
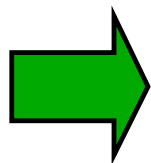
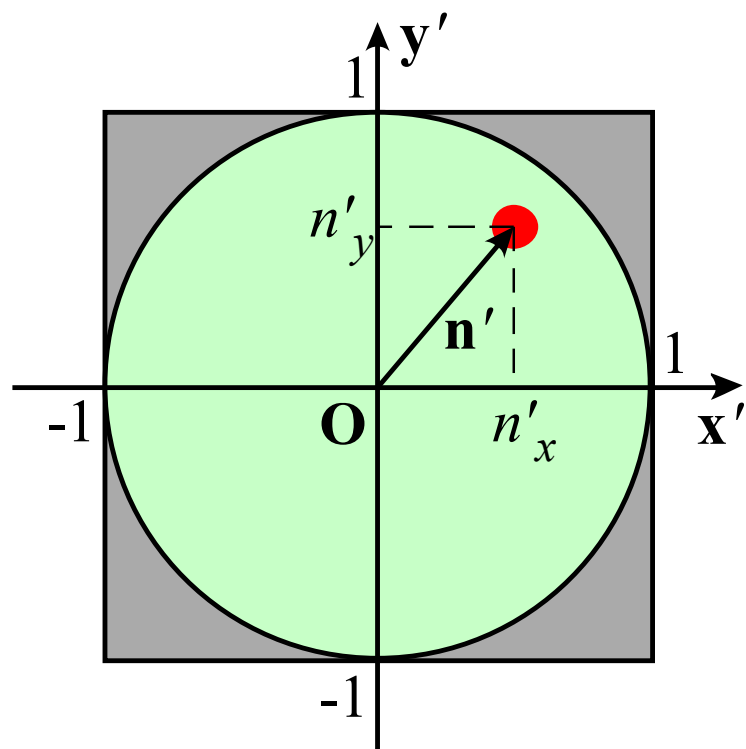
したがって中間ベクトルは

$$\mathbf{h}' = \frac{(r_x, r_y, r_z + 1)}{\sqrt{r_x^2 + r_y^2 + (r_z + 1)^2}} = \mathbf{n}'$$

$$m = \sqrt{r_x^2 + r_y^2 + (r_z + 1)^2} \Rightarrow \mathbf{n}' = \left(\frac{r_x}{m}, \frac{r_y}{m}, \frac{r_z + 1}{m} \right)$$

r 方向に見えるテクスチャが (n'_x, n'_y) の位置に映りこんで見える

法線ベクトルとテクスチャ座標の対応



$$m = \sqrt{r_x^2 + r_y^2 + (r_z + 1)^2}$$

$$u = \frac{n'_x}{2} + \frac{1}{2} = \frac{r_x}{2m} + \frac{1}{2}$$

$$v = \frac{n'_y}{2} + \frac{1}{2} = \frac{r_y}{2m} + \frac{1}{2}$$

法線ベクトルの xy 座標 (n'_x, n'_y)
をテクスチャ座標に使えばよい

バーテックスシェーダ

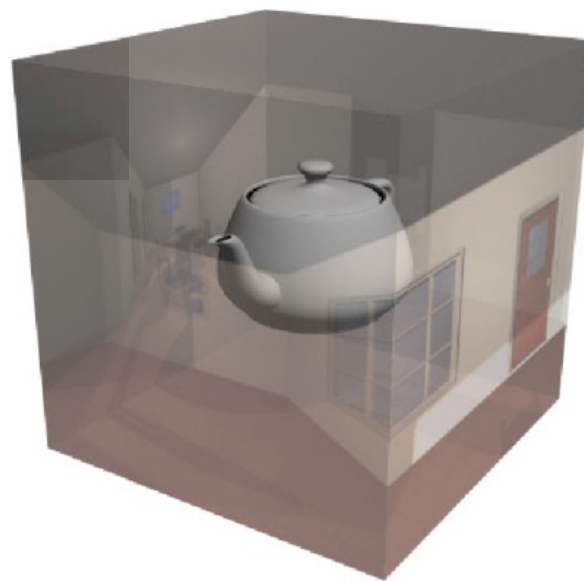
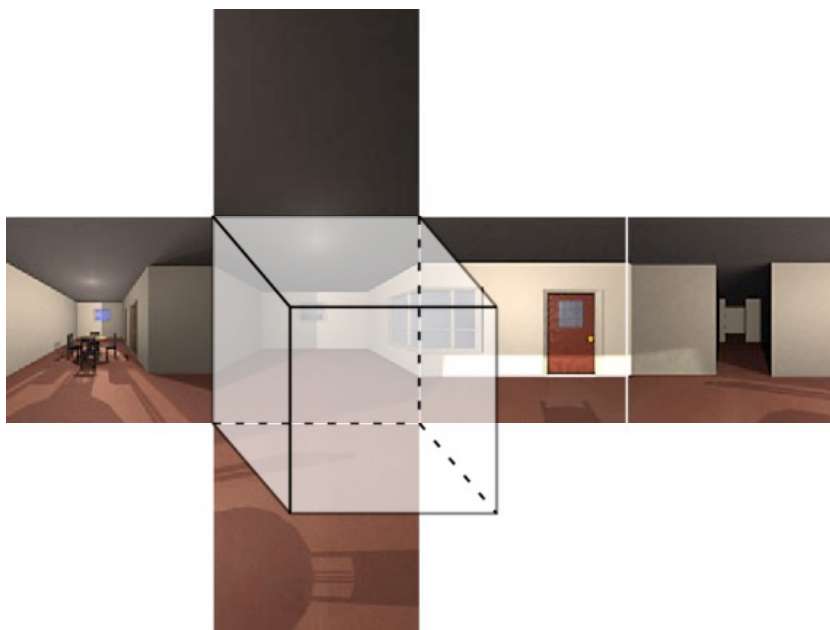
```
#version 410
in vec4 pv;           // 頂点位置
in vec3 nv;           // 頂点の法線ベクトル
uniform mat4 mw;      // モデルビュー変換行列
uniform mat3 mg;      // 法線変換行列
uniform mat3 ms;      // Sphere Mapping の基底行列
out vec2 t;           // テクスチャ座標

void main(void)
{
    t = normalize(ms * nv).xy * 0.5 + 0.5;
    ...
}
```

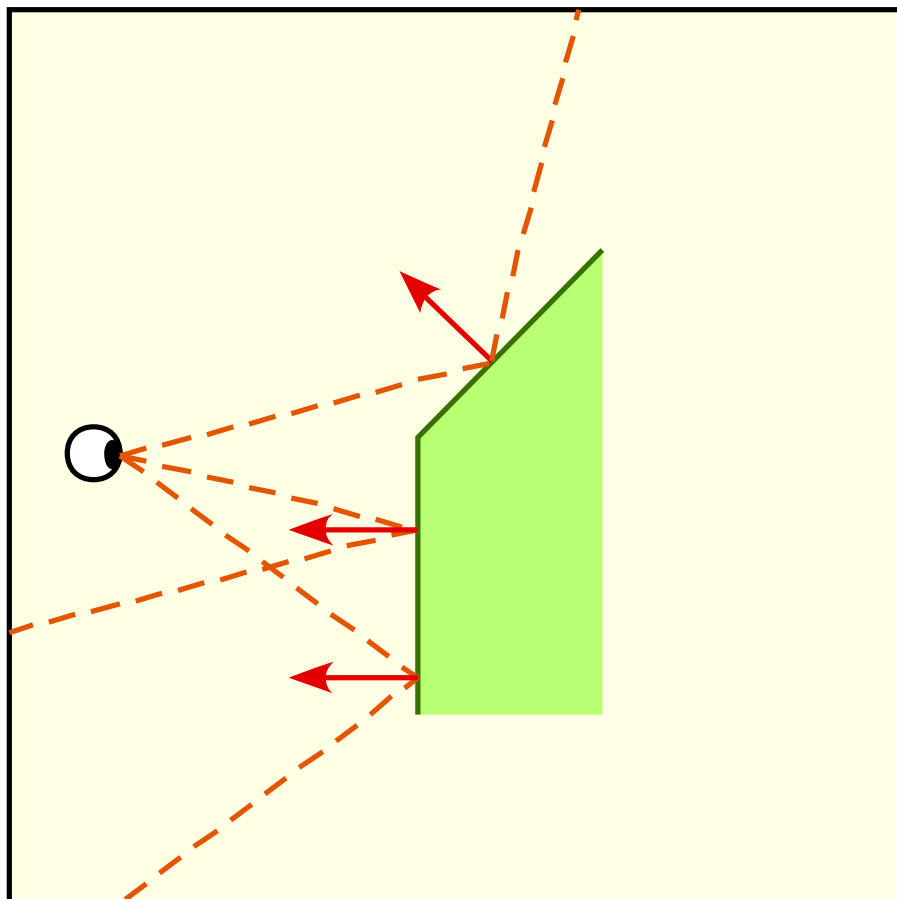
Sphere Mapping の基底行列 ms は
法線変換行列 mg と同じなので $ms \rightarrow mg$ として構わない

Cubic Environment Mapping

- 立方体に環境マップを貼り付ける
 - テクスチャ画像は6枚使う
 - 反射方向にどのテクスチャがあるか判別する



テクスチャの選択



どのテクスチャを標本化するかを反射ベクトルの方向で決定する

メインプログラム

```
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, 0, internal,  
  WIDTH, HEIGHT, 0, GL_RGB, GL_UNSIGNED_BYTE, textureNX);  
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, internal,  
  WIDTH, HEIGHT, 0, GL_RGB, GL_UNSIGNED_BYTE, texturePX);  
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, internal,  
  WIDTH, HEIGHT, 0, GL_RGB, GL_UNSIGNED_BYTE, textureNY);  
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, 0, internal,  
  WIDTH, HEIGHT, 0, GL_RGB, GL_UNSIGNED_BYTE, texturePY);  
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, internal,  
  WIDTH, HEIGHT, 0, GL_RGB, GL_UNSIGNED_BYTE, textureNZ);  
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, 0, internal,  
  WIDTH, HEIGHT, 0, GL_RGB, GL_UNSIGNED_BYTE, texturePZ);
```

internal は内部フォーマット, WIDTH, HEIGHT は読み込むテクスチャの幅と高さ,
GL_RGB と GL_UNSIGNED_BYTE は読み込みテクスチャのデータ形式

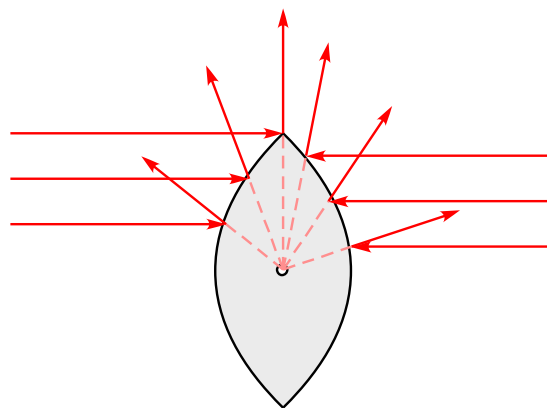
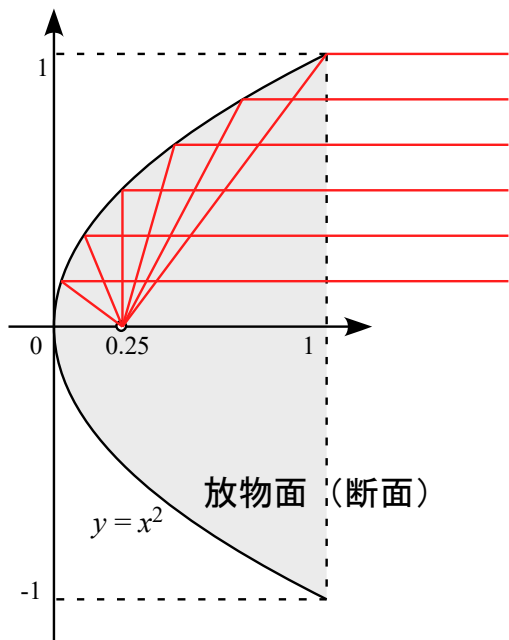
フラグメントシェーダ

```
#version 410
...
uniform samplerCube cubemap; // Cube Map の sampler
...
in vec3 r; // 反射方向ベクトルはバーテックスシェーダで計算しておく
...
void main(void)
{
    vec4 rcolor = texture(cubemap, r); // 反射方向の色
    ...
}
```

r を正規化する必要はない

Dual Paraboloid Mapping

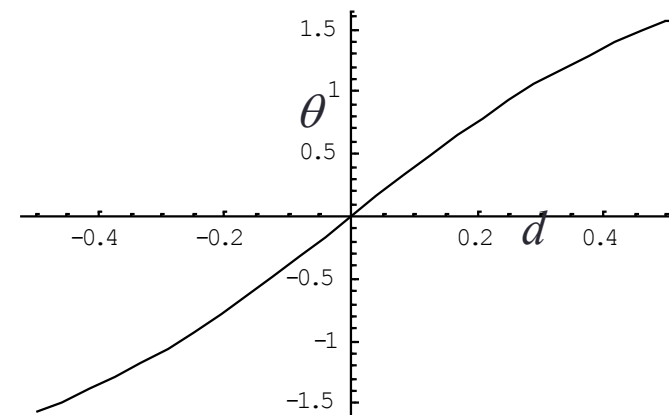
- 放物面への映り込みを環境マップに使う
 - 2枚のテクスチャを使う
 - 近似的に魚眼レンズで撮影した画像が使用できる
 - 魚眼レンズを使えばカメラが映り込むことはない



2つの放物面鏡への
環境の映り込み

$$d = \sqrt{u^2 + v^2}$$

$$\theta = \arctan\left(\frac{d}{0.25 - d^2}\right)$$



Paraboloid Mapping のテクスチャ座標

• 反射ベクトル

$$\mathbf{r} = (r_x, r_y, r_z)$$

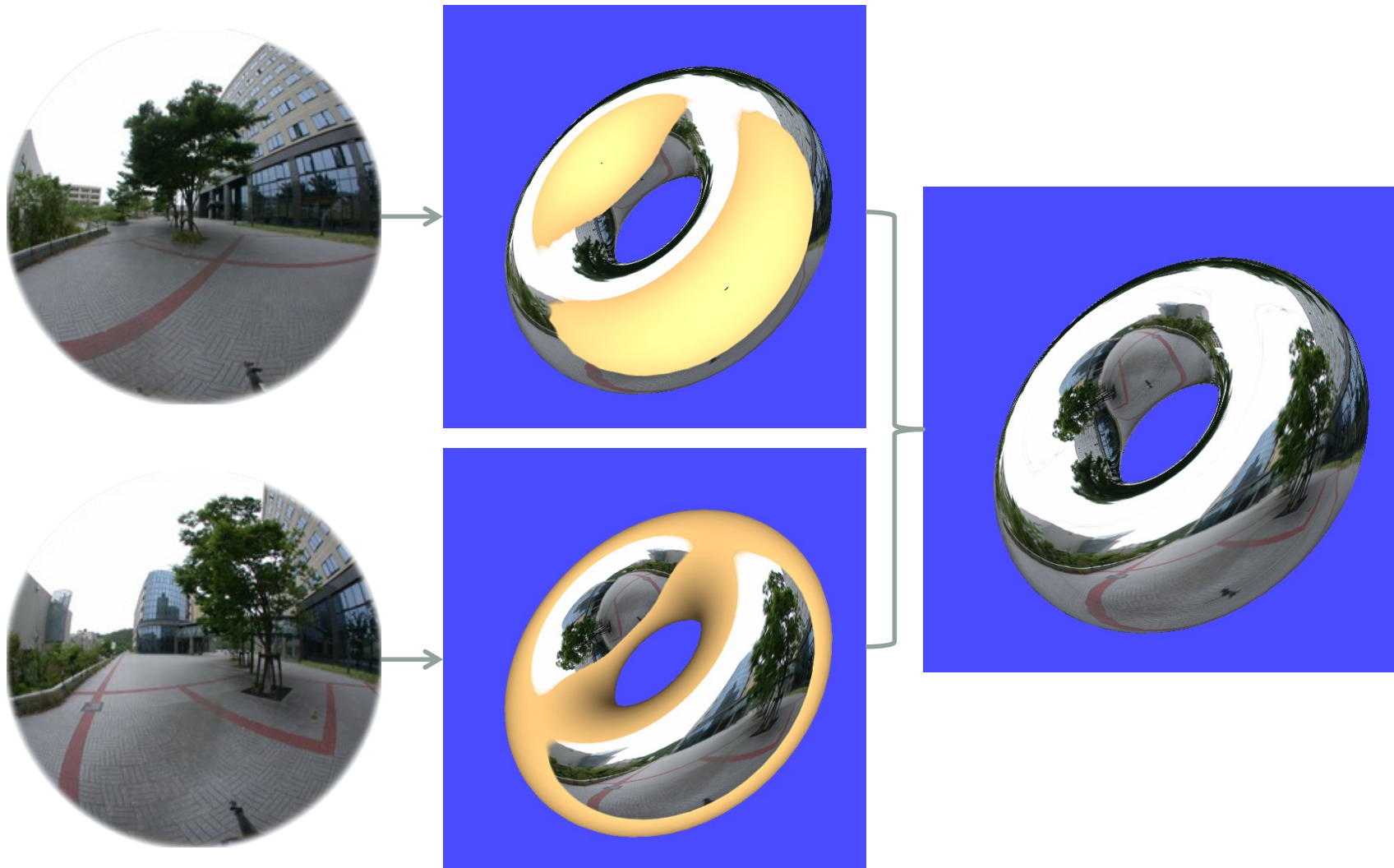
$$r_z \geq 0 \text{ のとき} \begin{cases} u = \frac{r_x}{2(1+r_z)} + 0.5 \\ v = \frac{r_y}{2(1+r_z)} + 0.5 \end{cases} \Rightarrow \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 \end{pmatrix} \begin{pmatrix} r_x \\ r_y \\ r_z \\ 1 \end{pmatrix}$$

で表側のテクスチャを標本化する

$$r_z < 0 \text{ のとき} \begin{cases} u = \frac{r_x}{2(1-r_z)} + 0.5 \\ v = \frac{r_y}{2(1-r_z)} + 0.5 \end{cases} \Rightarrow \begin{pmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 \end{pmatrix} \begin{pmatrix} r_x \\ r_y \\ r_z \\ 1 \end{pmatrix}$$

で裏側のテクスチャを標本化する

2枚のテクスチャの合成



OpenGL の固定機能を使った実装

- 頂点のテクスチャ座標として $(r_x \ r_y \ r_z \ 1)^T$ を設定する
- テクスチャ変換行列に前記の行列を設定する
- 表と裏の両方のテクスチャに対して標本値を求める
- マルチテクスチャを使って、これらの標本値の合計をマッピングする
 - 一方のテクスチャの標本点が範囲内であれば、もう一方のテクスチャは範囲外が標本化される
 - 範囲外の標本値を 0（黒）にしておけば、常にどちらか一方だけのテクスチャの標本値が得られる
 - 実はこれはうまくいかない ⇒ アルファ値によるデカールなどを用いる
 - これにより多角形のテクスチャ座標が表と裏の両方のテクスチャにまたがっていても正しくマッピングされる

GLSL による実装 (フラグメントシェーダ)

```
#version 410
...

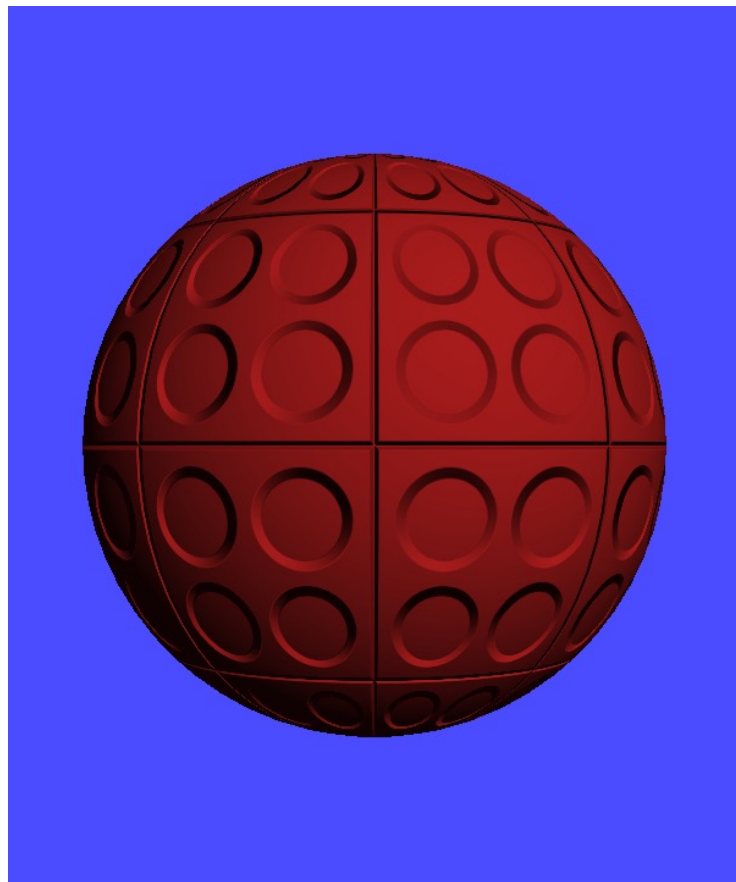
uniform sampler2D front; // 二重放物面マップの表のテクスチャ
uniform sampler2D back;  // 二重放物面マップの裏のテクスチャ
...

in vec3 r; // 反射方向ベクトルはバーテックスシェーダで計算しておく
...

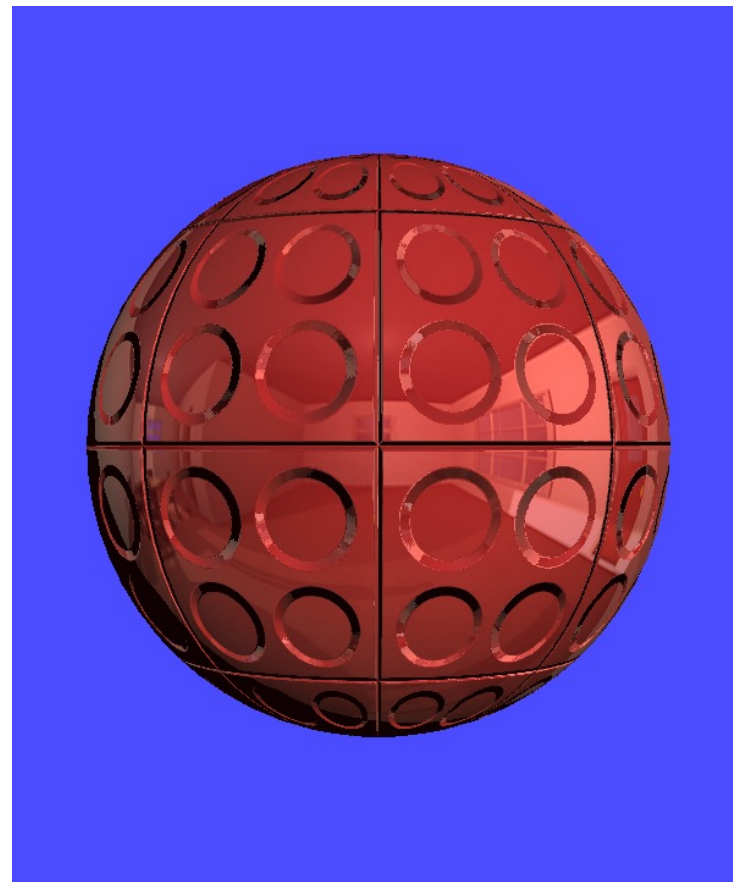
main()
{
    vec2 uv = r.xy * 0.5 / (1.0 + abs(r.z)) + 0.5;
    vec4 rcolor = texture(r.z >= 0.0 ? front : back, uv);
    ...
}
```

Environment Map Bump Mapping

バンプマッピング



環境マップバンプマッピング

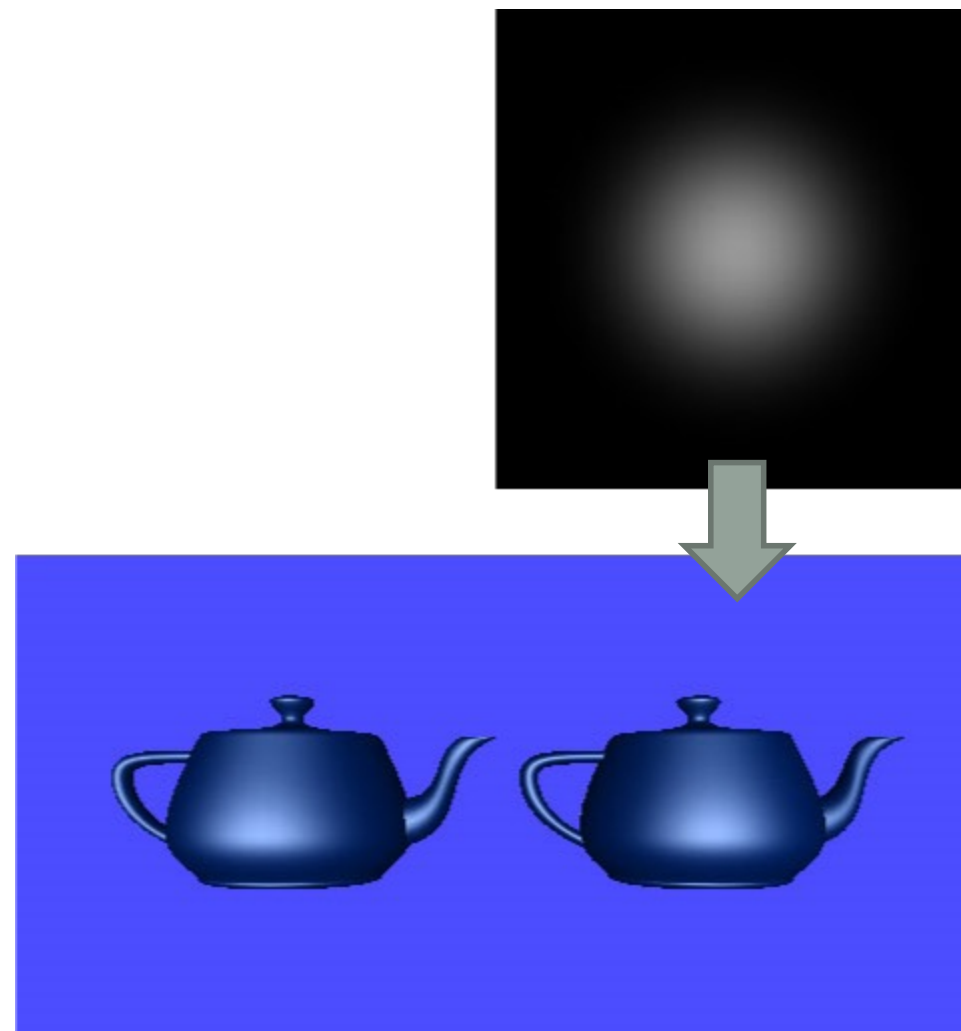


環境マッピングによる陰影計算

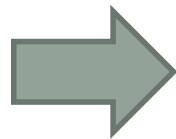
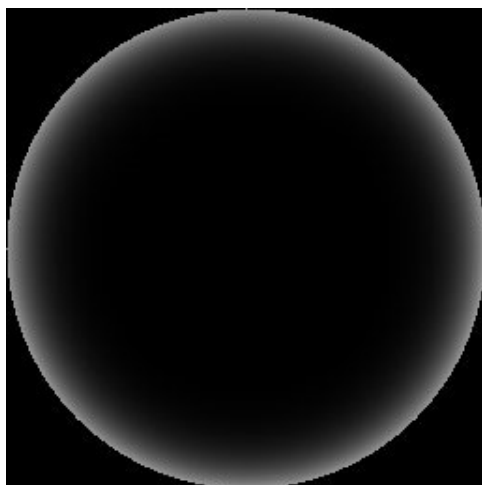
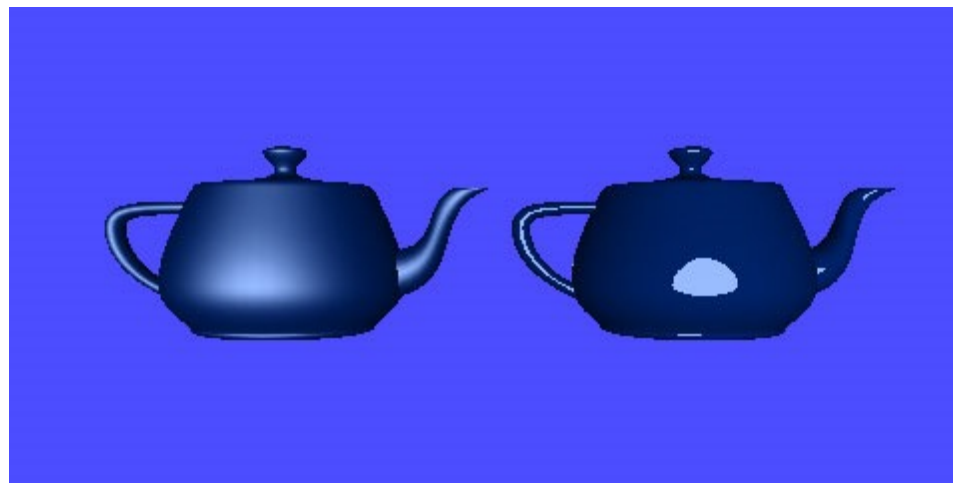
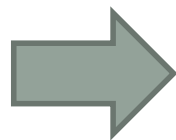
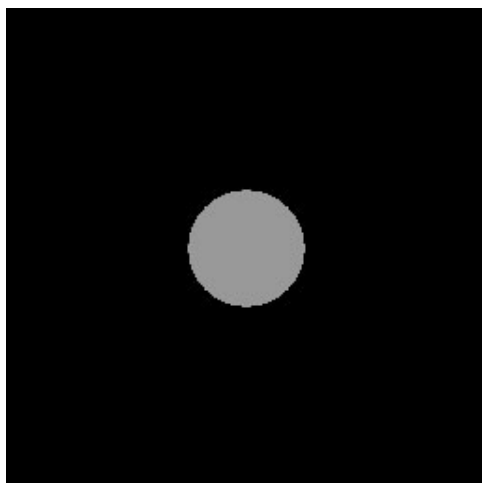
光源領域のテクスチャによる表現

ハイライトマッピング

- 環境マッピングを使ってハイライトを表現する
 - 視線をZ軸 $(0,0,1)$ と一致させた時の光線ベクトルを求める
 - $-1 \leq x, y \leq 1$ の領域上で $z^2 = 1 - x^2 - y^2$ を求める
 - $z > 0$ なら (x, y, z) を法線ベクトルとして鏡面反射光強度を求める
 - それをテクスチャの $(0.5x + 0.5, 0.5y + 0.5)$ の位置に格納する
 - Sphere Mapping によりこのテクスチャを拡散反射光強度に**加算**

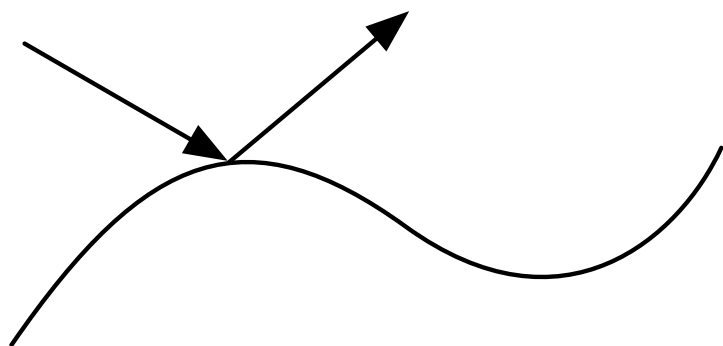


テクスチャなのでハイライトの形は自由に設定できる

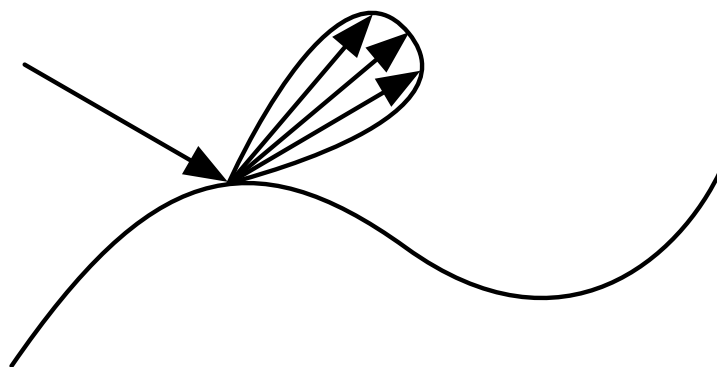


環境マップフィルタリング

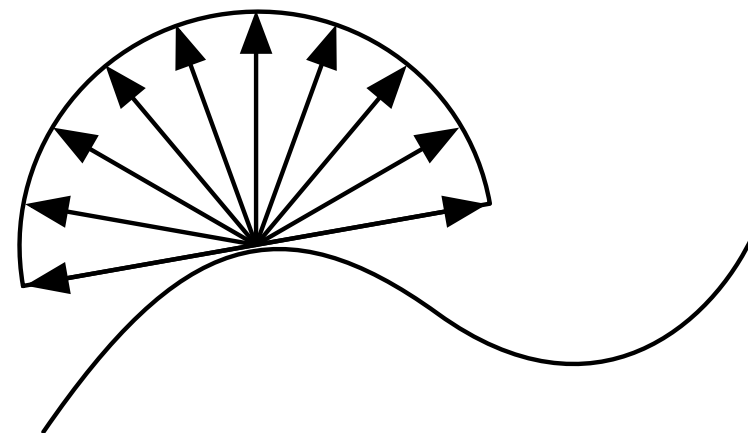
- 環境マップは映り込みの表現が可能
 - つややかな面や拡散面の陰影に応用する
- 反射光は天空光の分布とBRDFの畳み込み



完全な鏡面反射
(映り込み)



鏡面反射光の分布



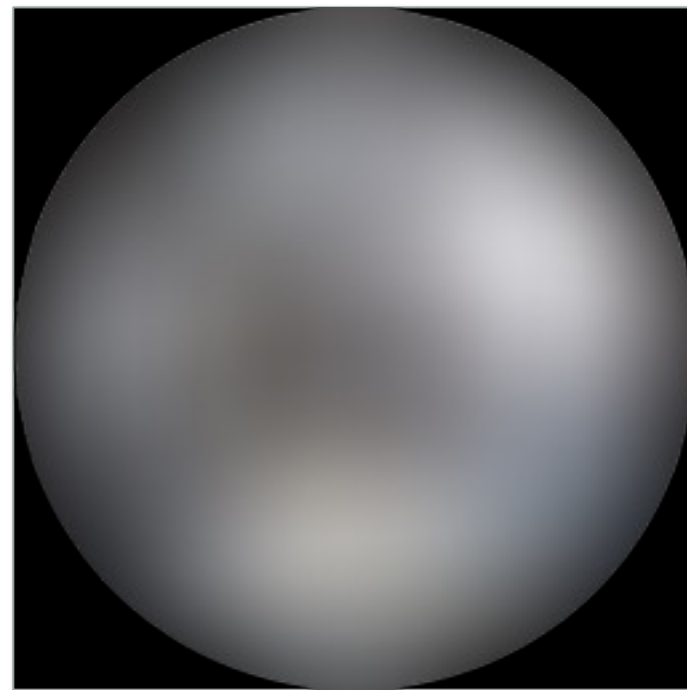
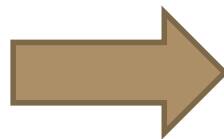
拡散反射光の分布

Shininess を
映り込みに反映する

環境マップ
フィルタリング

フィルタリングの方法

- 環境マップを均一にぼかす
 - 本当は場所によってぼけ方が異なる
 - 目は寛容なので全体的な効果の方が重要



環境マップフィルタリングの結果

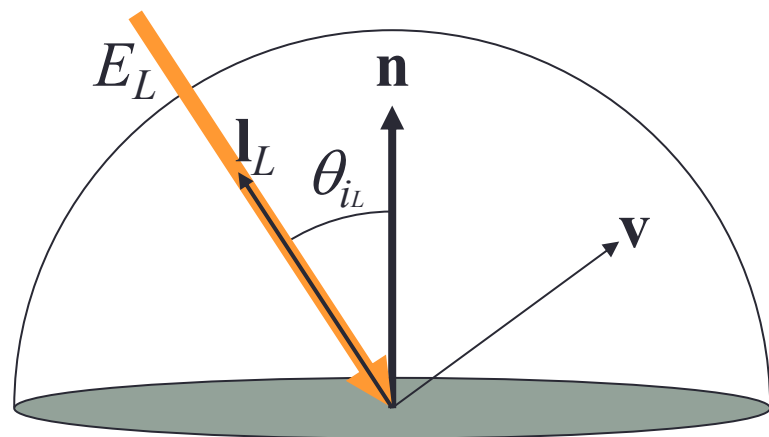
元の環境マップ



環境マップをぼかした場合

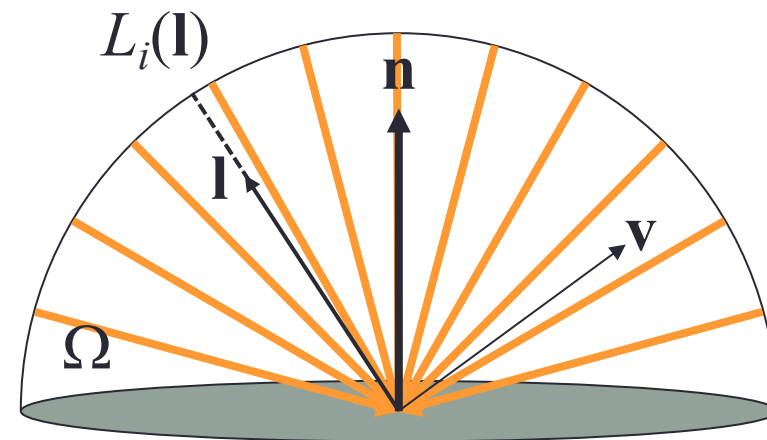


完全拡散反射面の天空光による陰影



点光源 / 平行光線

$$E = E_L \overline{\cos \theta_{i_L}}$$

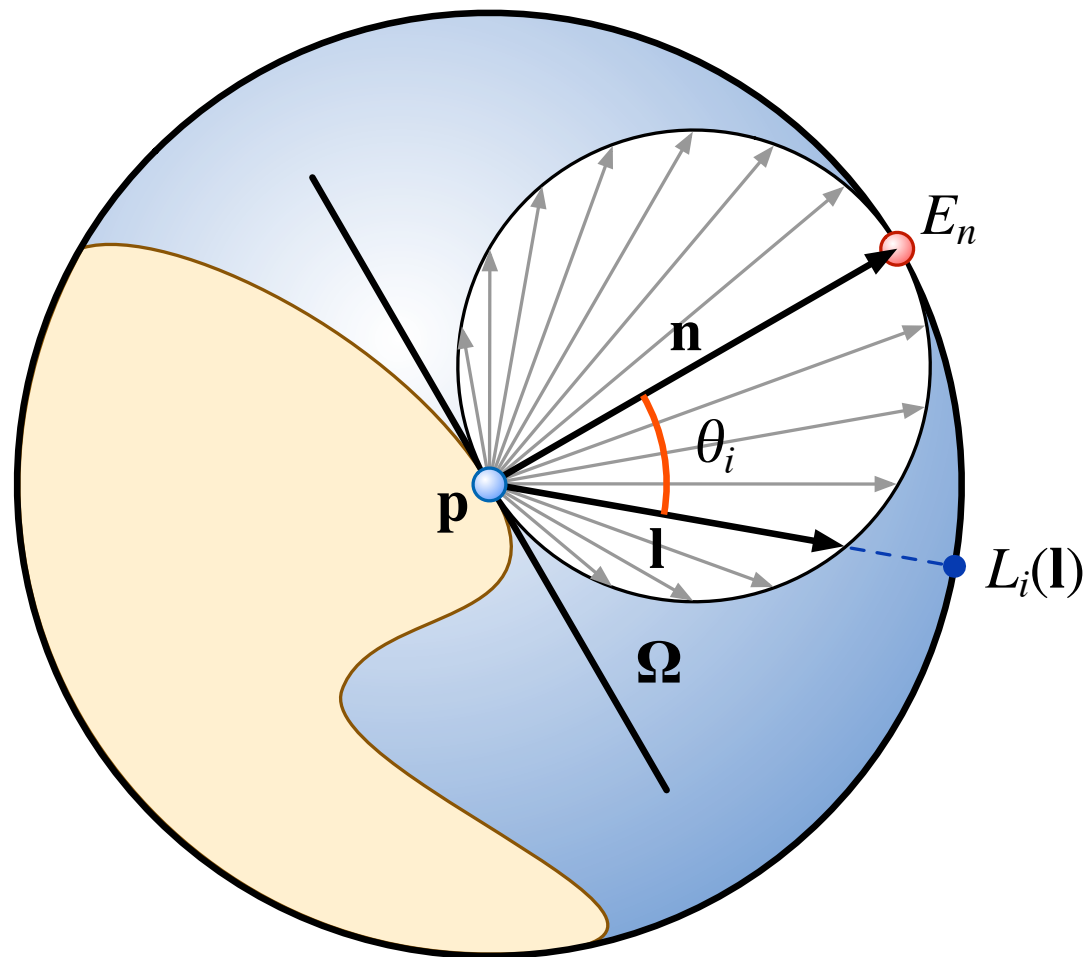


天空光

$$E = \int_{\Omega} L_i(\mathbf{l}) \cos \theta_i d\omega_i$$

放射照度マップ

放射照度マップの作成



Ω : p の接空間の天空
 \mathbf{n} : p の法線ベクトル
 \mathbf{l} : \mathbf{n} に対して θ_i 回転した方向
 $L_i(\mathbf{l})$: 天空の \mathbf{l} 方向の放射輝度

天空は p から十分遠いものとする

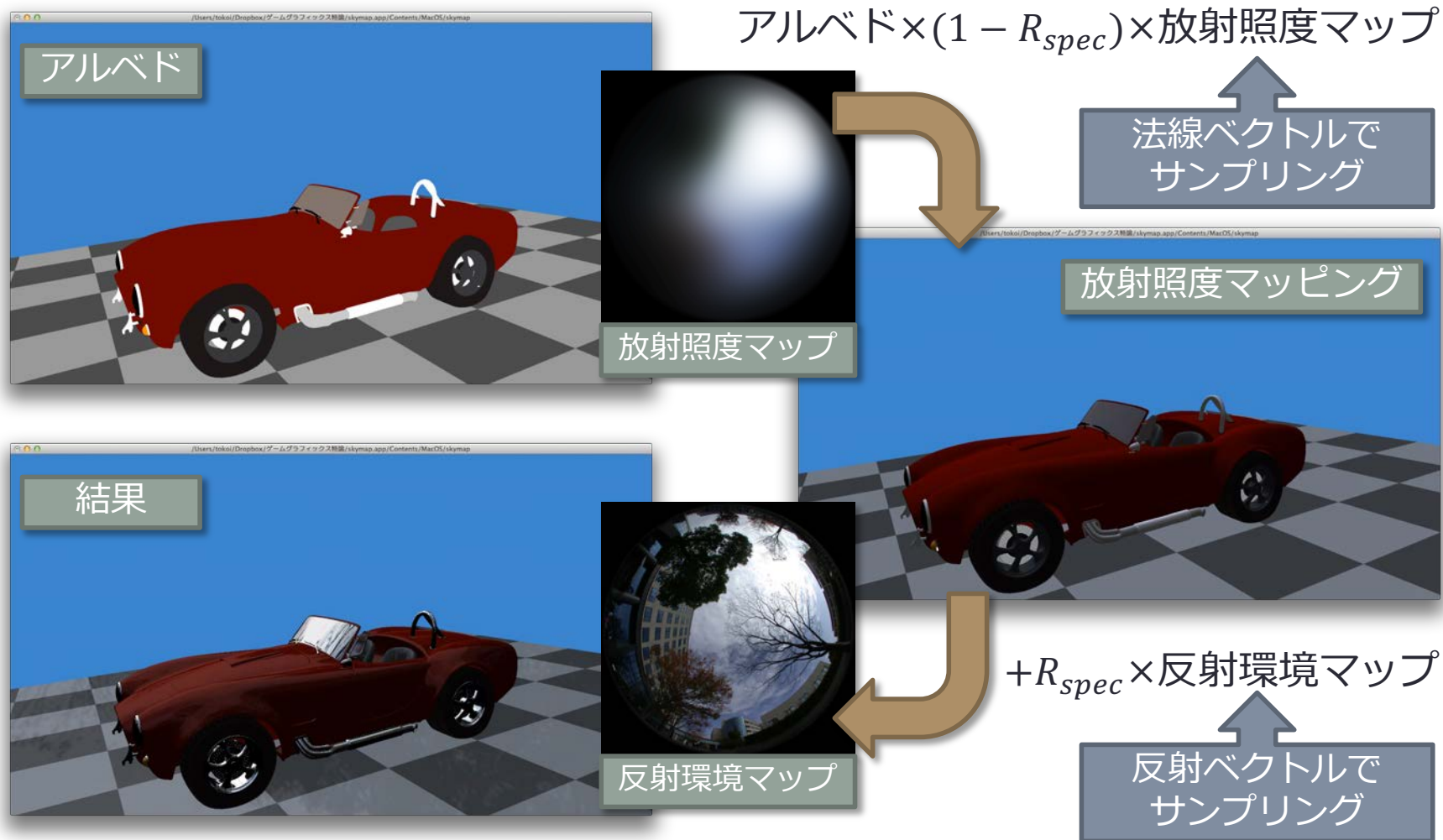


\mathbf{n} の方向を向いた面の
放射照度 E_n を次式で求める

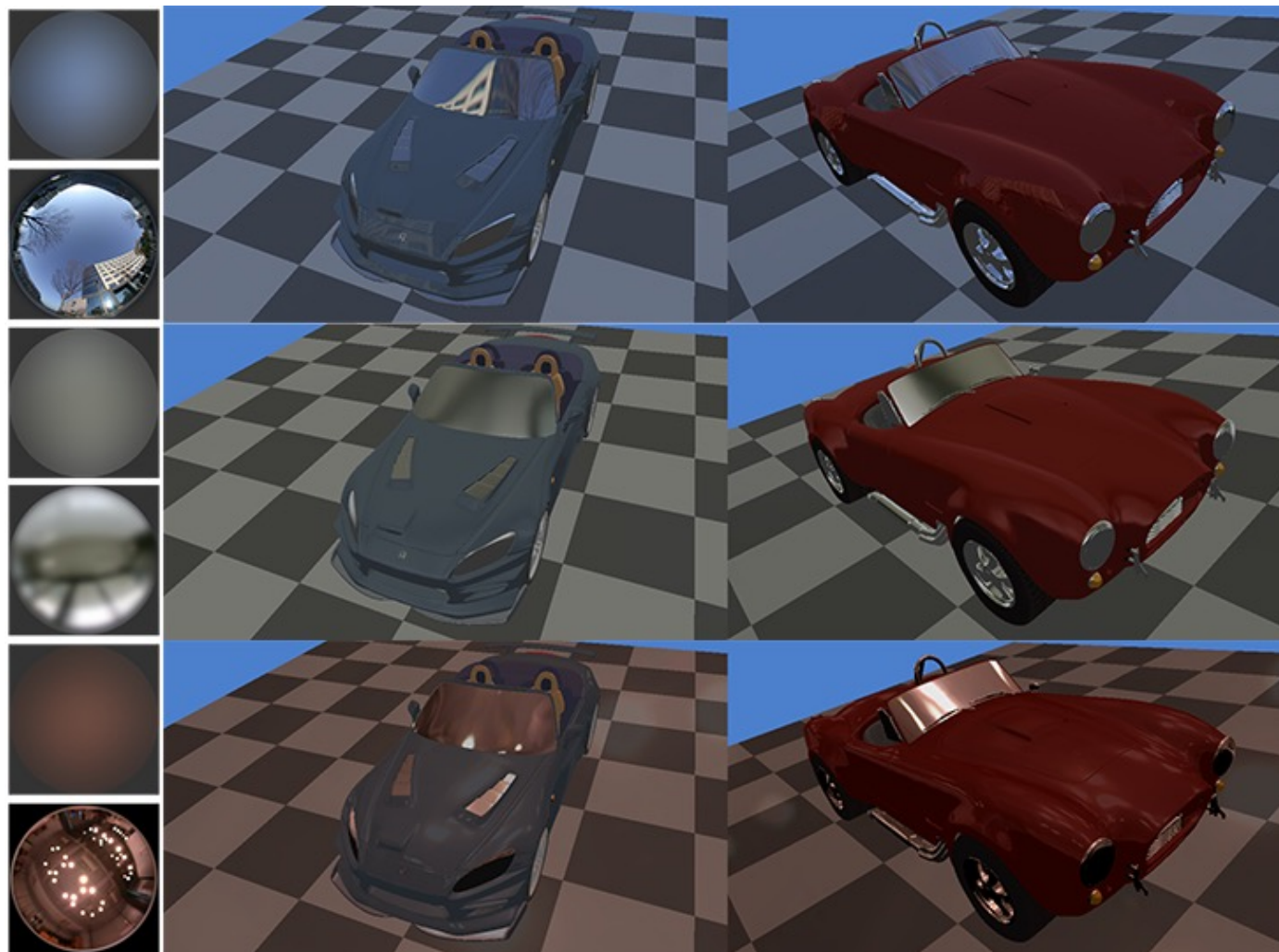
$$E_n = \int_{\Omega} L_i(\mathbf{l}) \cos \theta_i d\omega_i$$

この E_n を放射照度マップの
 \mathbf{n} の方向に格納する

放射照度マッピング



放射照度マップと陰影の変化



Cube Map による放射照度マップ

環境マップ



放射照度マップ



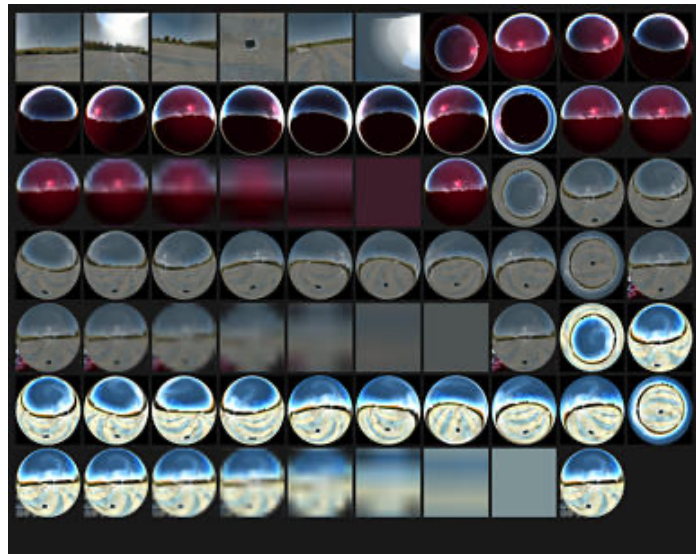
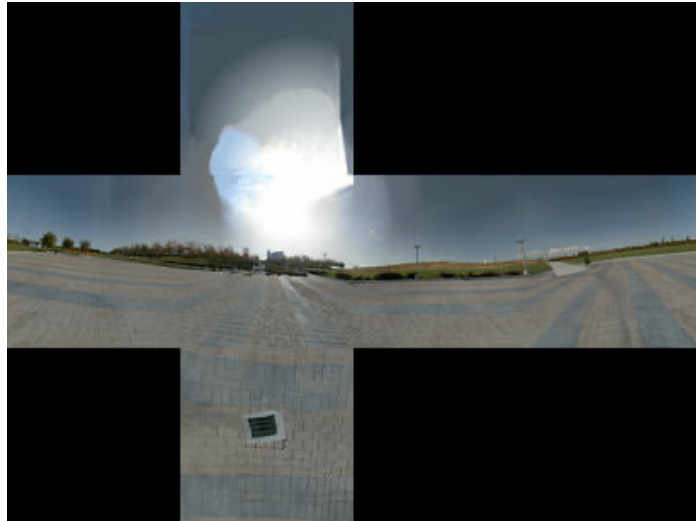
本当はもっとボケて
いないといけない

放射照度環境マッピングによる陰影



DEAD OR ALIVE 3 (c) TECMO, LTD. Team NINJA 2001

Reflection Space Image Based Rendering



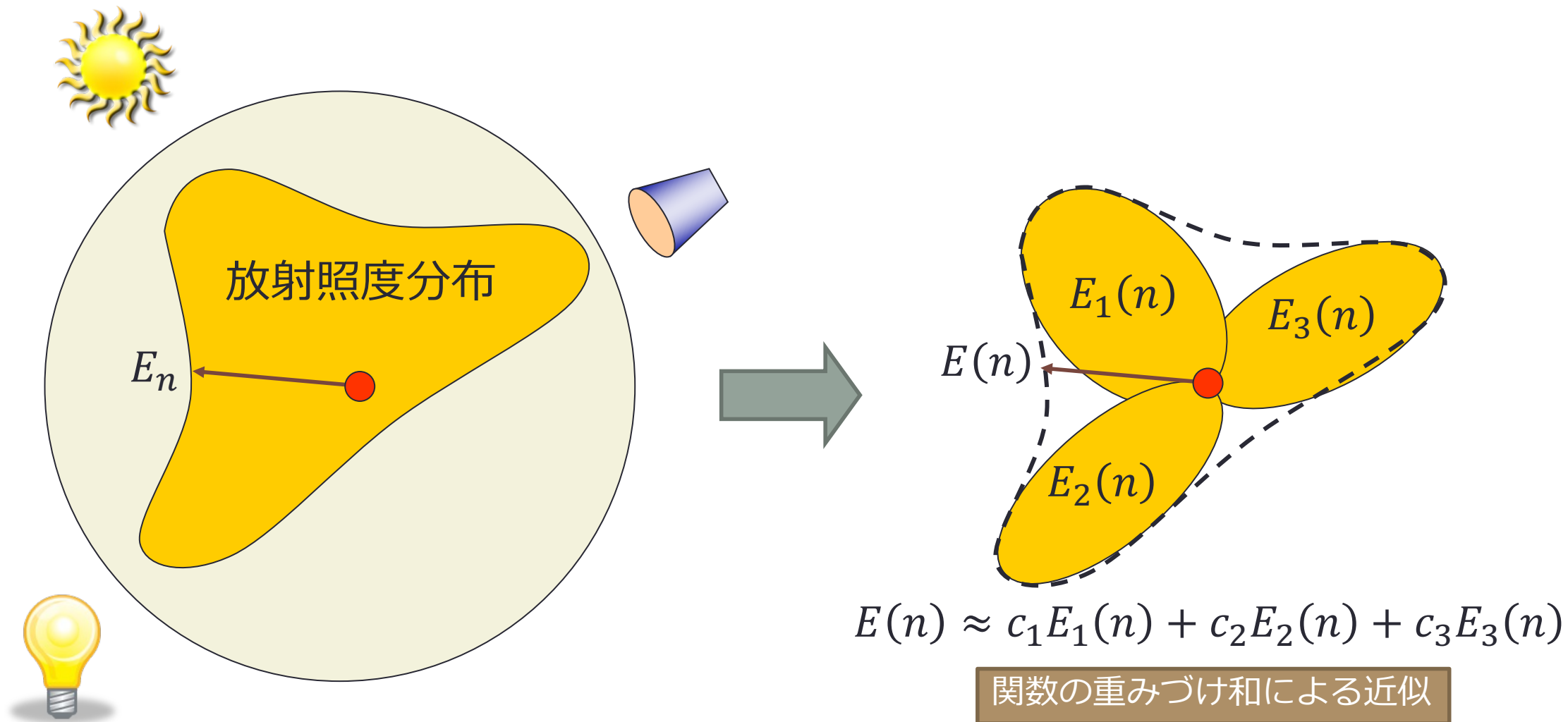
Cabral, Olano, Nemec (1999)



球面調和解析

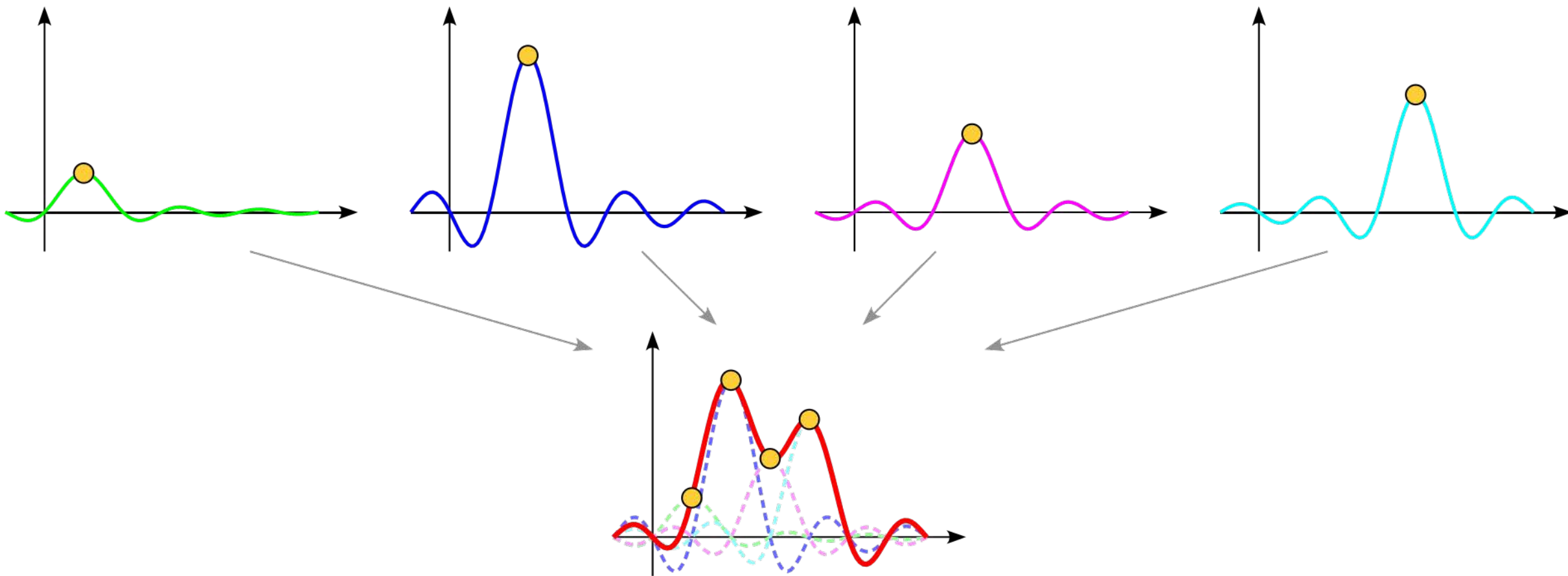
放射照度マップの関数近似

放射照度マップの近似



直交基底関数の重み付け和による近似

重み付けされた直交基底関数

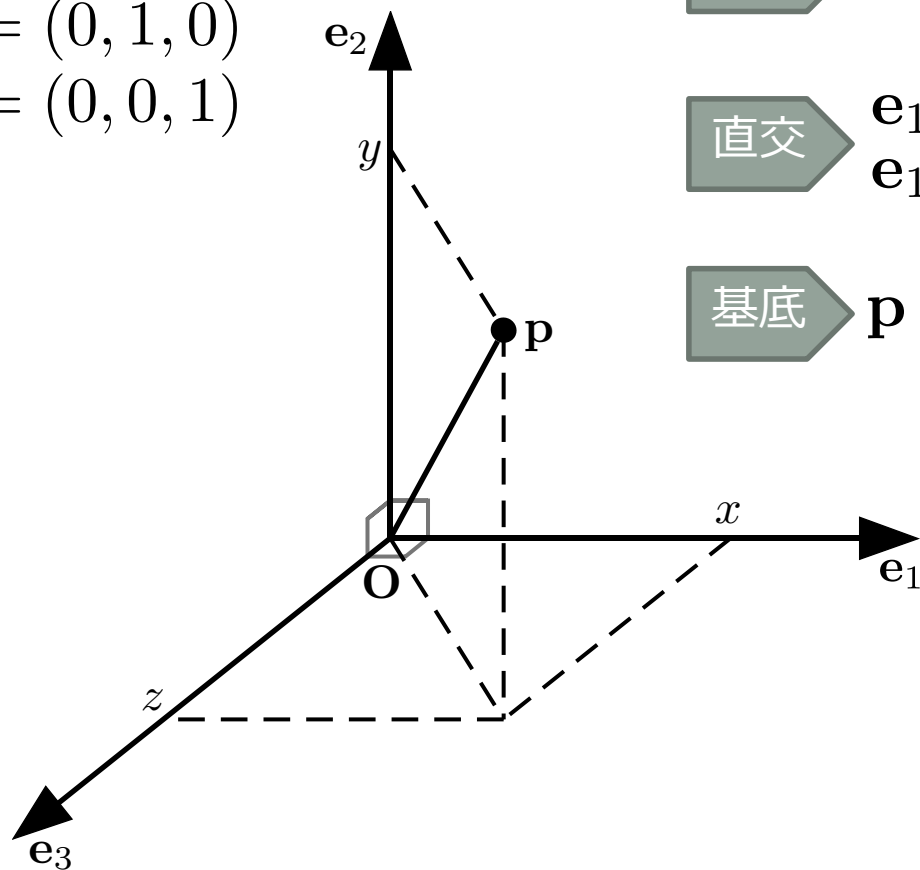


正規直交基底

$$\mathbf{e}_1 = (1, 0, 0)$$

$$\mathbf{e}_2 = (0, 1, 0)$$

$$\mathbf{e}_3 = (0, 0, 1)$$



正規 $|\mathbf{e}_1| = |\mathbf{e}_2| = |\mathbf{e}_3| = 1$

直交 $\mathbf{e}_1 \cdot \mathbf{e}_1 = \mathbf{e}_2 \cdot \mathbf{e}_2 = \mathbf{e}_3 \cdot \mathbf{e}_3 = 1$
 $\mathbf{e}_1 \cdot \mathbf{e}_2 = \mathbf{e}_2 \cdot \mathbf{e}_3 = \mathbf{e}_3 \cdot \mathbf{e}_1 = 0$

基底 $\mathbf{p} = x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3$

直交基底関数

- 内積

$$\langle f_i(x), f_j(x) \rangle = \int f_i(x) f_j(x) dx$$

- 空間的な関数では

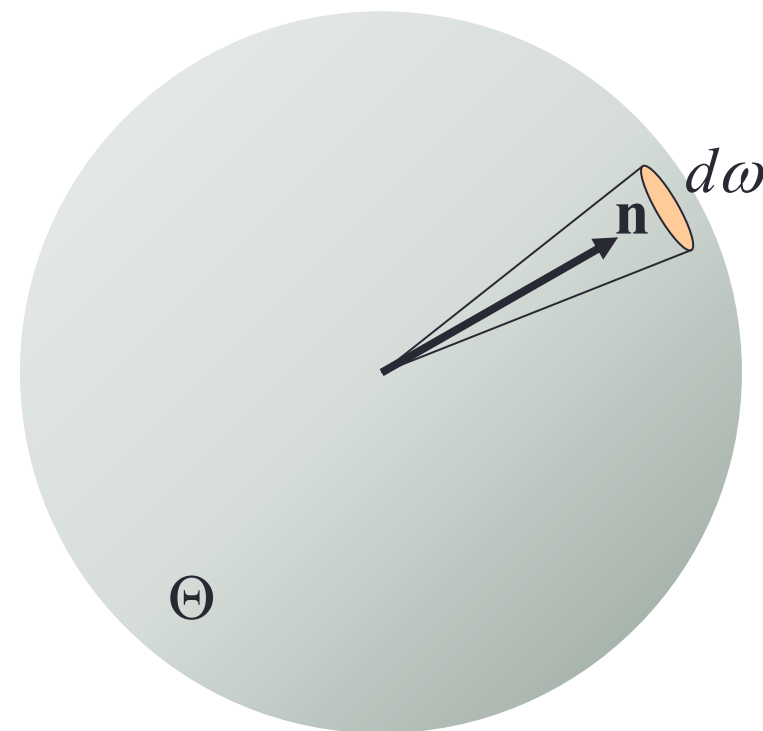
$$\langle f_i(\mathbf{n}), f_j(\mathbf{n}) \rangle = \int_{\Theta} f_i(\mathbf{n}) f_j(\mathbf{n}) d\omega$$

- 直交基底関数の性質

$$\langle f_i(), f_j() \rangle = \begin{cases} 0 & (i \neq j) \\ 1 & (i = j) \end{cases}$$

自分自身との内積は1, それ以外は 0 ⇒ **直交**

\mathbf{n} : 中心から向かう方向
 $d\omega$: 単位球上の微小面積
 Θ : 単位球全体



直交関数基底関数による関数の近似

- 近似しようとする関数 $f_{target}()$ と直交基底関数列 $f_j()$ との内積

$$k_j = \langle f_{target}(), f_j() \rangle$$

- より $f_{target}()$ は次式で近似できる

$$f_{target}() \approx \sum_{j=1}^n k_j f_j()$$

ルジャンドル陪関数 (ルジャンドル陪多項式)

- ルジャンドル多項式 (ロドリゲスの公式)

$$P_n(x) = \frac{1}{2^n n!} \cdot \frac{d^n}{dx^n} (x^2 - 1)^n$$

n : 次数 (degree)
 m : 位数 (order)
 $|x| \leq 1, |m| \leq n, n = 0, 1, 2, \dots$

- ルジャンドル陪関数

$$P_n^m(x) = (-1)^m (1 - x^2)^{\frac{m}{2}} \frac{d^m}{dx^m} P_n(x)$$

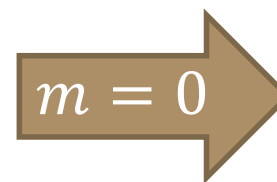
!!: 二重階乗
 (一つ飛ばしの階乗)
 $5!! = 5 \cdot 3 \cdot 1$
 $6!! = 6 \cdot 4 \cdot 2$

- 漸化式 ([ルジャンドル陪関数の漸化式 \(fc2.com\)](http://fc2.com))

$$(n - m)P_n^m(x) = x(2n - 1)P_{n-1}^m(x) - (n + m - 1)P_{n-2}^m(x)$$

$$P_m^m(x) = (-1)^m (2m - 1)!! (1 - x^2)^{\frac{m}{2}}$$

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



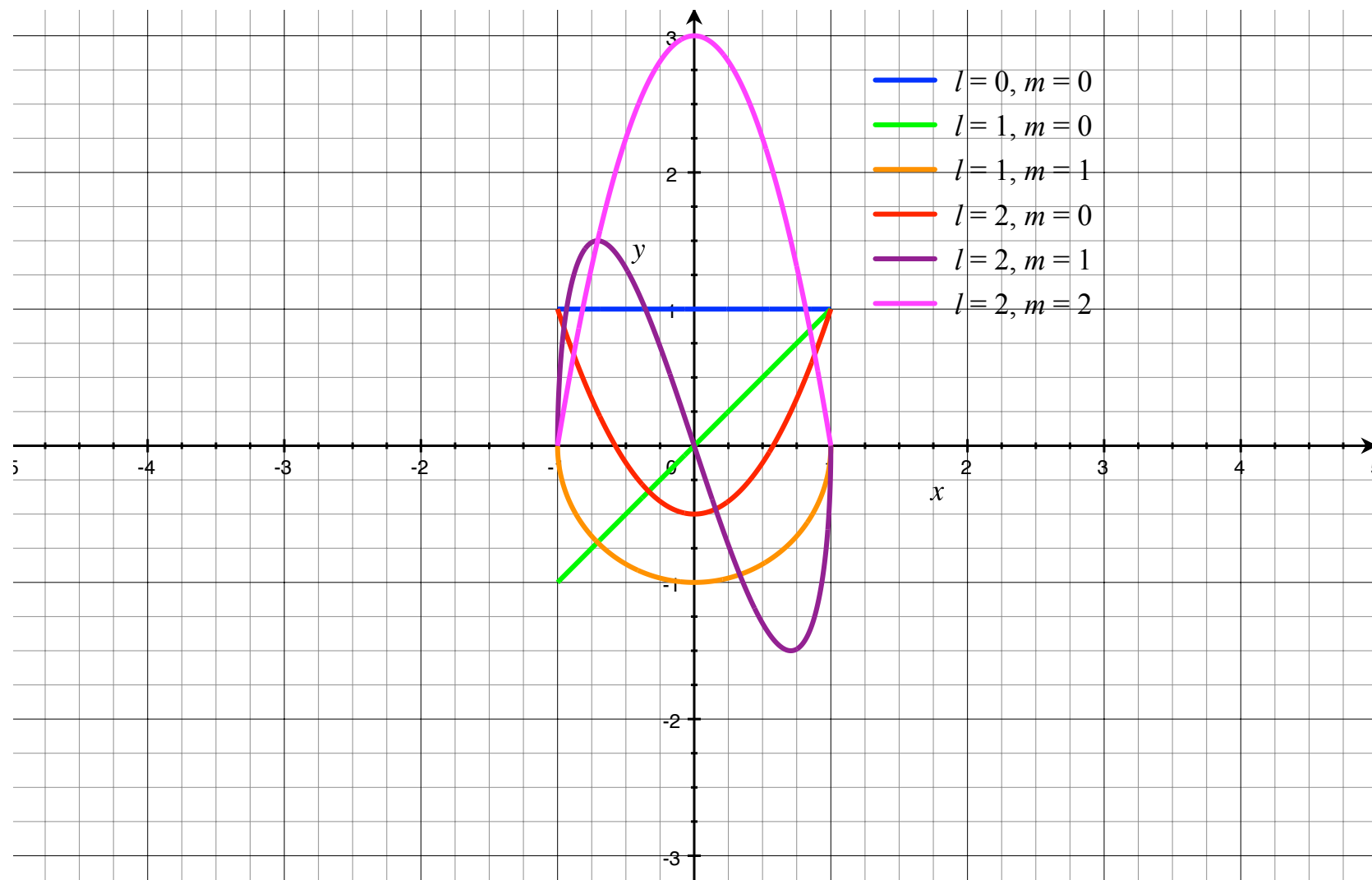
$$P_0^0(x) = 1$$

$$P_1^0(x) = x$$

ルジャンドル陪関数 $P_n^m(x)$

	$m = 0$	$m = 1$	$m = 2$
$l = 0$	$P_0^0(x) = 1$		
$l = 1$	$P_1^0(x) = x$	$P_1^1(x) = -\sqrt{1-x^2}$	
$l = 2$	$P_2^0(x) = \frac{3x^2 - 1}{2}$	$P_2^1(x) = -3x\sqrt{1-x^2}$	$P_2^2(x) = 3(1-x^2)$

$P_l^m(x)$ の形



ルジャンドル陪関数の性質

$$\int_{-1}^{+1} P_i^m(x) P_j^m(x) dx = \begin{cases} 0 & (i \neq j) \\ \frac{2}{2n+1} \cdot \frac{(n+m)!}{(n-m)!} & (i = j = n) \end{cases}$$

直交

[ルジャンドル多項式の直交性 \(fc2.com\)](http://fc2.com)

球面調和関数

$$Y_l^m(\theta, \phi) = (-1)^{\frac{m+|m|}{2}} K_l^m e^{im\phi} P_l^{|m|}(\cos \theta)$$

緯度方向

経度方向

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

$$\mathbf{n} = (x, y, z) = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta) \quad \xrightarrow{\text{より}} \quad \begin{cases} \cos \theta = z \\ \phi = \tan^{-1} \frac{y}{x} \end{cases}$$

球面調和関数表 (Wikipedia に載ってた)

$$\begin{aligned}x &= r \sin \theta \cos \varphi \\y &= r \sin \theta \sin \varphi \\z &= r \cos \theta\end{aligned}$$

$$Y_1^{-1}(x) = \frac{1}{2} \sqrt{\frac{3}{2\pi}} \cdot e^{-i\varphi} \cdot \sin \theta = \frac{1}{2} \sqrt{\frac{3}{2\pi}} \cdot \frac{x - iy}{r}$$

$$Y_0^0(x) = \frac{1}{2} \sqrt{\frac{1}{\pi}}$$

$$Y_1^0(x) = \frac{1}{2} \sqrt{\frac{3}{\pi}} \cdot \cos \theta = \frac{1}{2} \sqrt{\frac{3}{\pi}} \cdot \frac{z}{r}$$

$$Y_1^1(x) = -\frac{1}{2} \sqrt{\frac{3}{2\pi}} \cdot e^{i\varphi} \cdot \sin \theta = -\frac{1}{2} \sqrt{\frac{3}{2\pi}} \cdot \frac{x + iy}{r}$$

$$Y_2^{-2}(x) = \frac{1}{4} \sqrt{\frac{15}{2\pi}} \cdot e^{-2i\varphi} \cdot \sin^2 \theta = \frac{1}{4} \sqrt{\frac{15}{2\pi}} \cdot \frac{x^2 - 2ixy - y^2}{r^2}$$

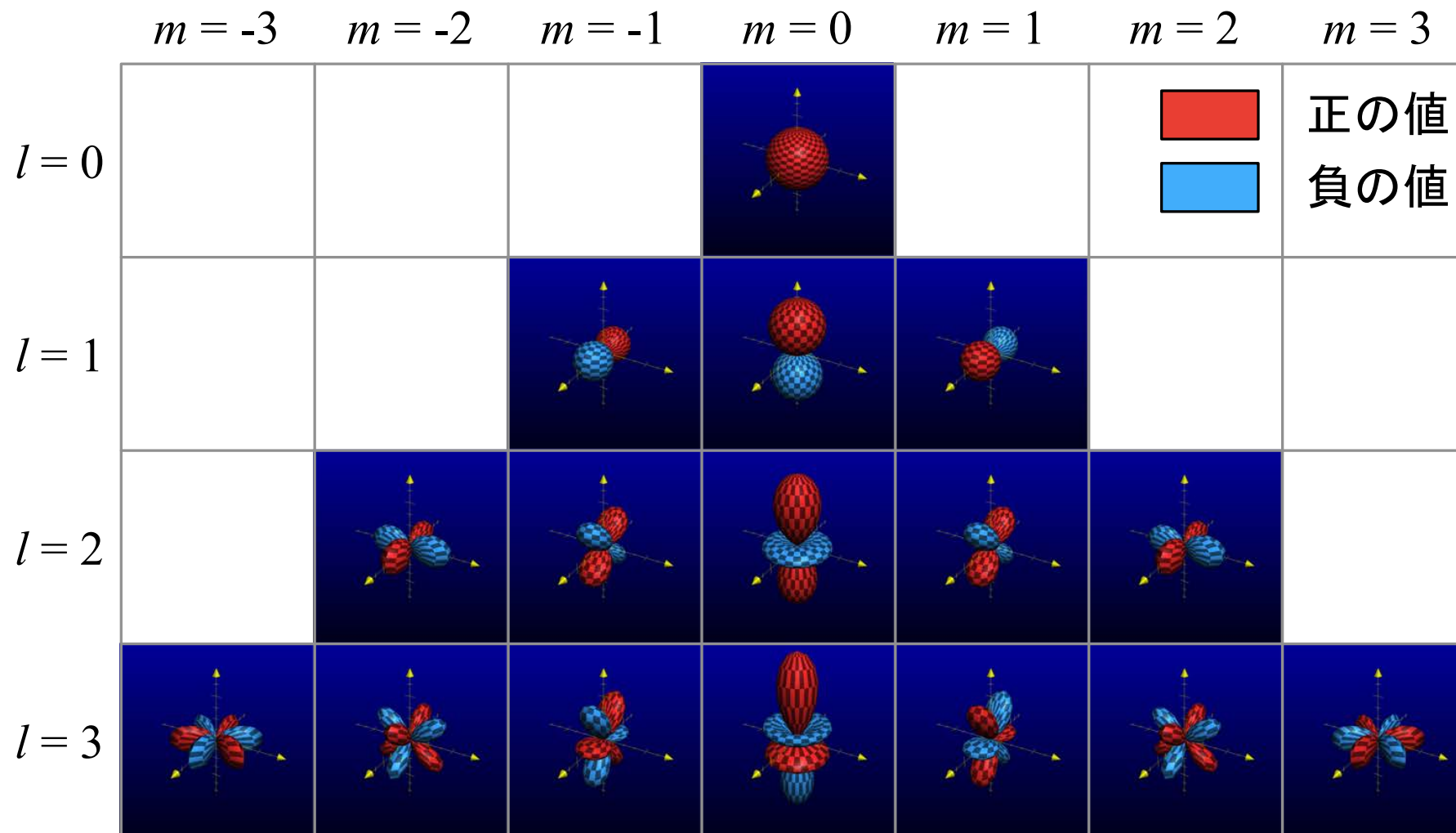
$$Y_2^{-1}(x) = \frac{1}{2} \sqrt{\frac{15}{2\pi}} \cdot e^{-i\varphi} \cdot \sin \theta \cdot \cos \theta = \frac{1}{2} \sqrt{\frac{15}{2\pi}} \cdot \frac{xz - iyz}{r^2}$$

$$Y_2^0(x) = \frac{1}{4} \sqrt{\frac{5}{\pi}} \cdot (3 \cos^2 \theta - 1) = \frac{1}{4} \sqrt{\frac{5}{\pi}} \cdot \frac{-x^2 - y^2 + 2z^2}{r^2}$$

$$Y_2^1(x) = -\frac{1}{2} \sqrt{\frac{15}{2\pi}} \cdot e^{i\varphi} \cdot \sin \theta \cdot \cos \theta = -\frac{1}{2} \sqrt{\frac{15}{2\pi}} \cdot \frac{xz + iyz}{r^2}$$

$$Y_2^2(x) = \frac{1}{4} \sqrt{\frac{15}{2\pi}} \cdot e^{2i\varphi} \cdot \sin^2 \theta = \frac{1}{4} \sqrt{\frac{15}{2\pi}} \cdot \frac{x^2 + 2ixy - y^2}{r^2}$$

$Y_l^m(\theta, \phi)$ の実部



(macOS 標準添付の Grapher で描いた)

球面調和関数の性質

$$\int_{-\pi}^{+\pi} \int_{-1}^{+1} Y_l^m(\theta, \phi) Y_{l'}^{m'}(\theta, \phi) d(\cos \theta) d\phi$$
$$= \int_{-\pi}^{+\pi} \int_0^{\pi} Y_l^m(\theta, \phi) Y_{l'}^{m'}(\theta, \phi) \sin \theta d\theta d\phi = \begin{cases} 1 & (l = l', m = m') \\ 0 & (\text{それ以外}) \end{cases}$$

直交

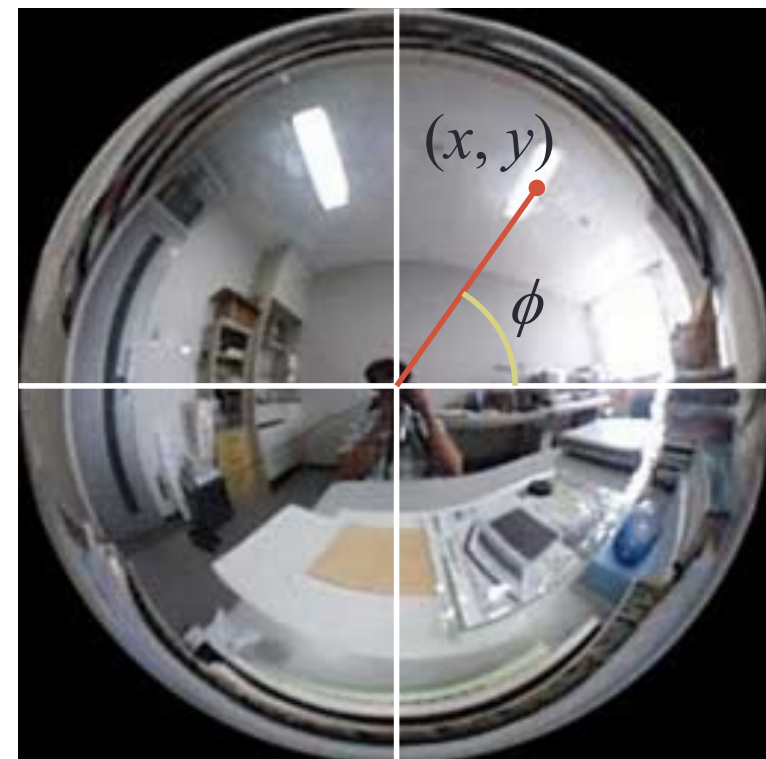
球面調和関数展開

$$f(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} c_l^m Y_l^m(\theta, \phi)$$

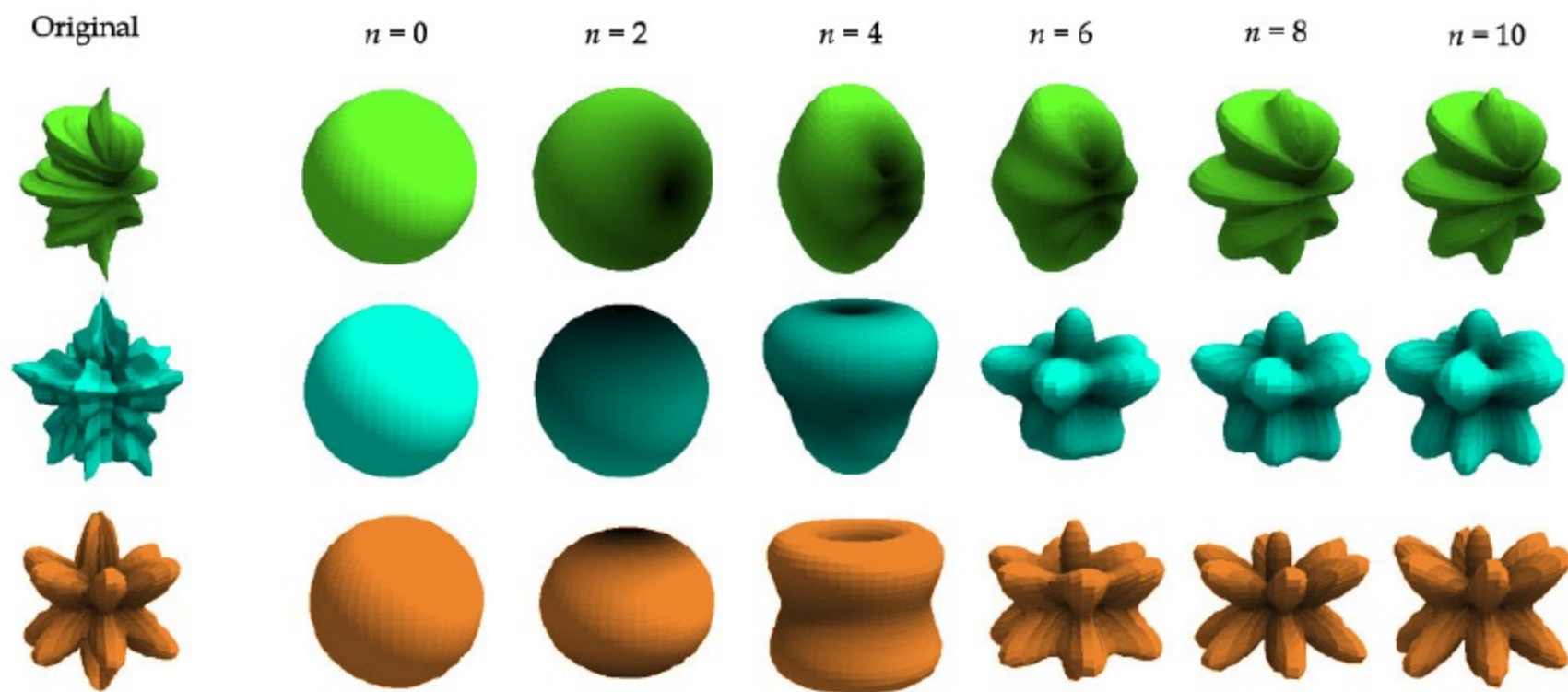
$$\begin{aligned} c_l^m &= \int_{-\pi}^{+\pi} \int_{-1}^{+1} f(\theta, \phi) Y_l^m(\theta, \phi) \underline{d(\cos \theta) d\phi} \\ &= \int_{-\pi}^{+\pi} \int_0^{\pi} f(\theta, \phi) Y_l^m(\theta, \phi) \underline{\sin \theta d\theta d\phi} \end{aligned}$$

light probe 画像の場合

$$\begin{aligned} \sin \theta &= \sqrt{x^2 + y^2} \\ \cos \theta &= z = \sqrt{1 - x^2 - y^2} \end{aligned}$$

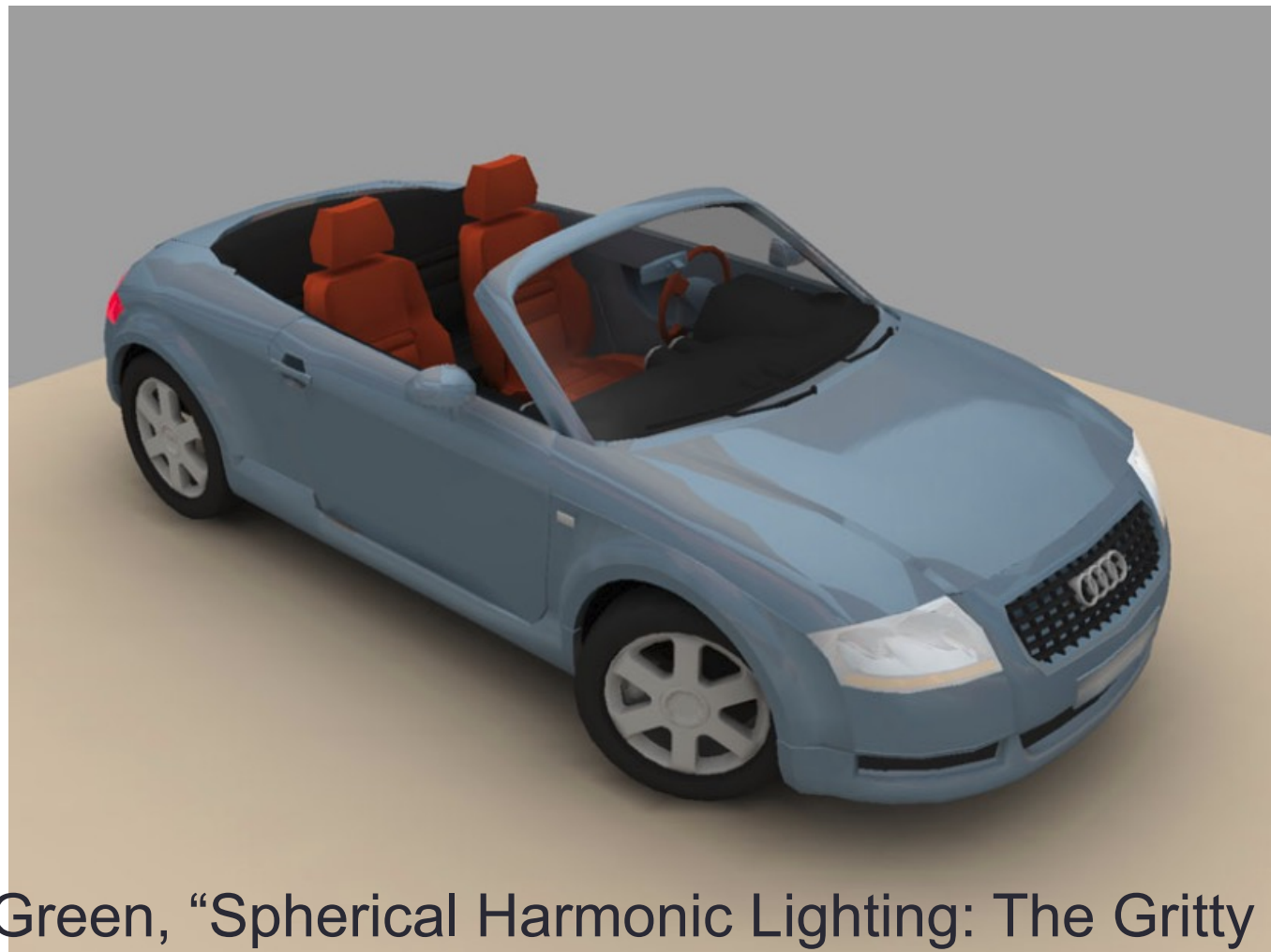


球面調和関数写像



Robin Green, "Spherical Harmonic Lighting: The Gritty Details"

作例



Robin Green, “Spherical Harmonic Lighting: The Gritty Details”

放射照度マップの球面調和関数近似

- 環境からの放射照度の総和 E

$$E = k_1 c_2^2 (x^2 - y^2) + k_3 c_2^0 z^2 + k_4 c_0^0 - k_5 c_2^0 \\ + 2k_1 (c_2^{-2} xy + c_2^1 xz + c_2^{-1} yz) \\ + 2k_2 (c_1^1 x + c_1^{-1} y + c_1^0 z)$$

- ここで (x, y, z) は法線ベクトル \mathbf{n}

$$\mathbf{n} = (x, y, z) = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta)$$

- 拡散反射光強度 I_{diff}

$$I_{diff} = (K_{diff} \otimes E) f_{scale}$$

E は非常に大きな値になることがあるため、HDR (ダイナミックレンジ) レンダリングが行えない (明度値に上限がある) 場合は、適当なスケールファクタ f_{scale} をかける。

$$k_1 = 0.429043$$

$$k_2 = 0.511664$$

$$k_3 = 0.743125$$

$$k_4 = 0.886227$$

$$k_5 = 0.247708$$

小テストー環境による照明

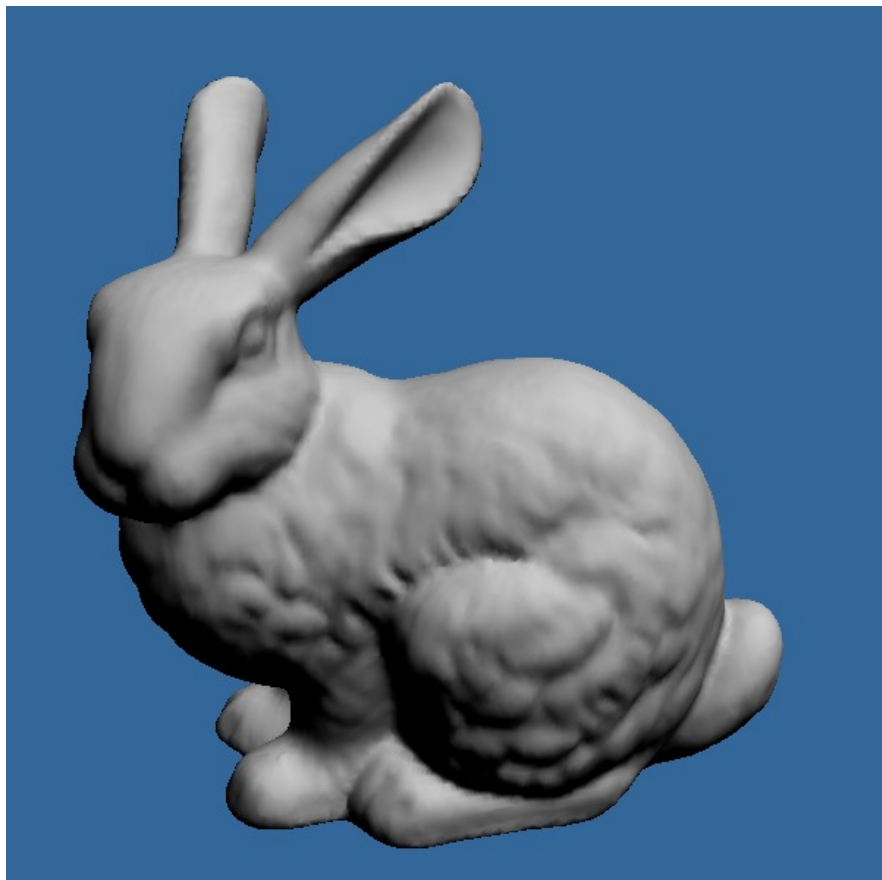
Moodle の小テストに解答してください

宿題

- 環境の光源による放射照度の球面調和解析による陰影付けを実装してください。
 - 次のプログラムは完全拡散反射面の Lambert の余弦法則による拡散反射光による陰影を付けた図形を表示します。
 - <https://github.com/tokoik/ggsample10>
 - これを環境の放射照度マップの球面調和関数による近似を用いた陰影付けに変更してください。
 - 球面調和係数は uniform 変数の配列 sh に格納されています。
 - $sh[0] = c_0^0$, $sh[1] = c_1^{-1}$, $sh[2] = c_1^0$, $sh[3] = c_1^1$,
 $sh[4] = c_2^{-2}$, $sh[5] = c_2^{-1}$, $sh[6] = c_2^0$, $sh[7] = c_2^1$, $sh[8] = c_2^2$
 - 係数は1秒ごとに自動的に切り替わります。
- ggsample10.vert をアップロードしてください

宿題プログラムの生成画像

Lambertの余弦法則による拡散反射光



環境の放射照度の球面調和関数近似

