シェーディング

シェーディング

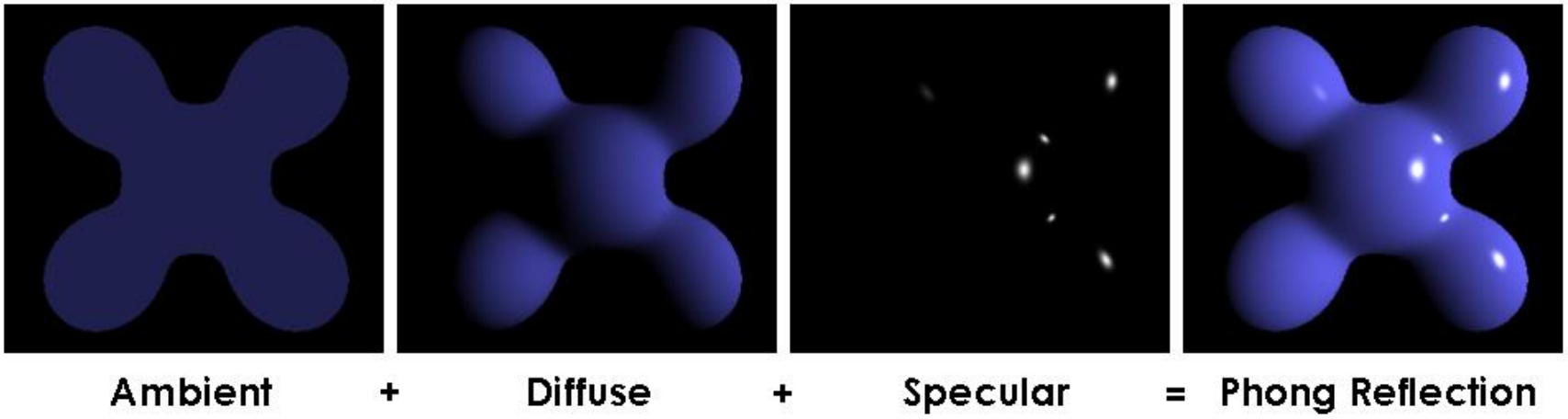
- 方程式（モデル）を使用して、材料特性と光源に基づいて、ビューレイに沿った放射輝度を計算するプロセス

- 放射輝度：領域ごとおよび入射方向ごとの光の流れの密度

Blinn-Phongモデル

- **アンビエントライトモデル-** すべての表面を均等に照らします。
- **拡散光モデル-** 反射強度は視線方向とは無関係ですが、光源に対する表面の向きに依存します。

- **スペキュラーライトモデル-** 鏡面を照らす光は輝点を引き起こします。鏡面反射率はビューに依存します。



画像クレジット：ウィキペディア

周囲光

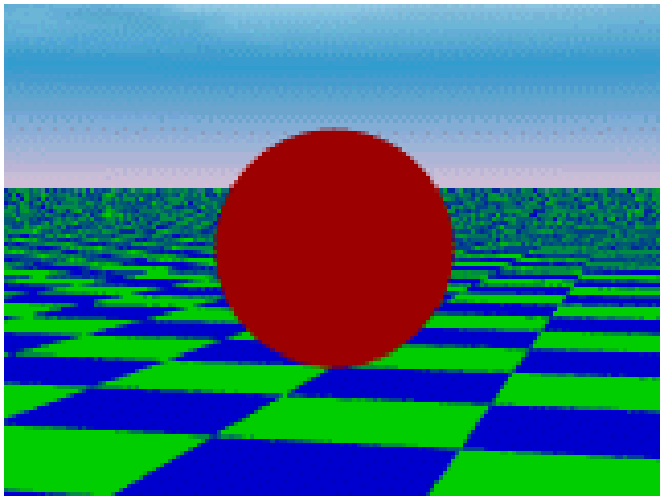
単純な照明モデルは、二次光源が無視されることを意味します。

直接照らされていないオブジェクトは黒く表示されます。あまりない*現実的です。*

と呼ばれる無指向性ライトの導入 **周囲光I_a**

結果として生じる反射光は、各表面で一定であり、
に依存します **周囲反射** 係数 **k_a**

$$私_a = k_a 私$$



表記

前の式では 、 どちらも **k** そして **私** RGBAです *ベクトル。*

k は材料特性（周囲、拡散、…）です。

私 光強度特性。

各成分は、赤、緑、青、アルファの強度に対応しています。この番号は[0、1]にあります。

掛け算は *コンポーネントごと。*

例

仮定：

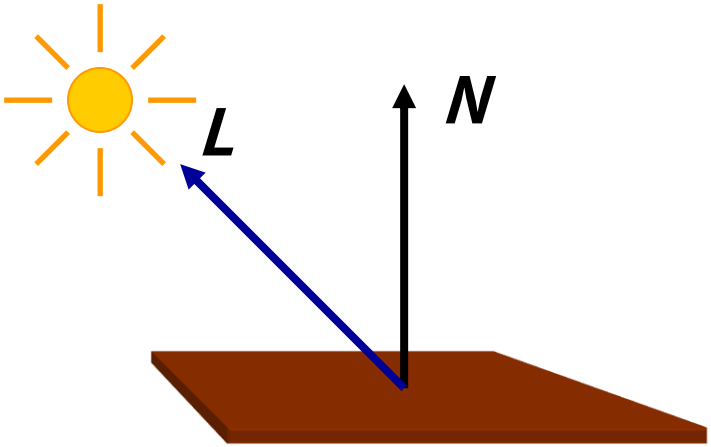
- 白色光： **I** = [1、1、1、1]
- 赤いオブジェクト： **k** = [1、0、0、1]次

に、

= 1 * 1[、0 * 1,0 * 1,1 * 1 = [1,0,0,1]

拡散反射

理想的な拡散反射（ランバート反射）



= cos = 最大。、0 （→ →）

k_d **拡散反射** 係数

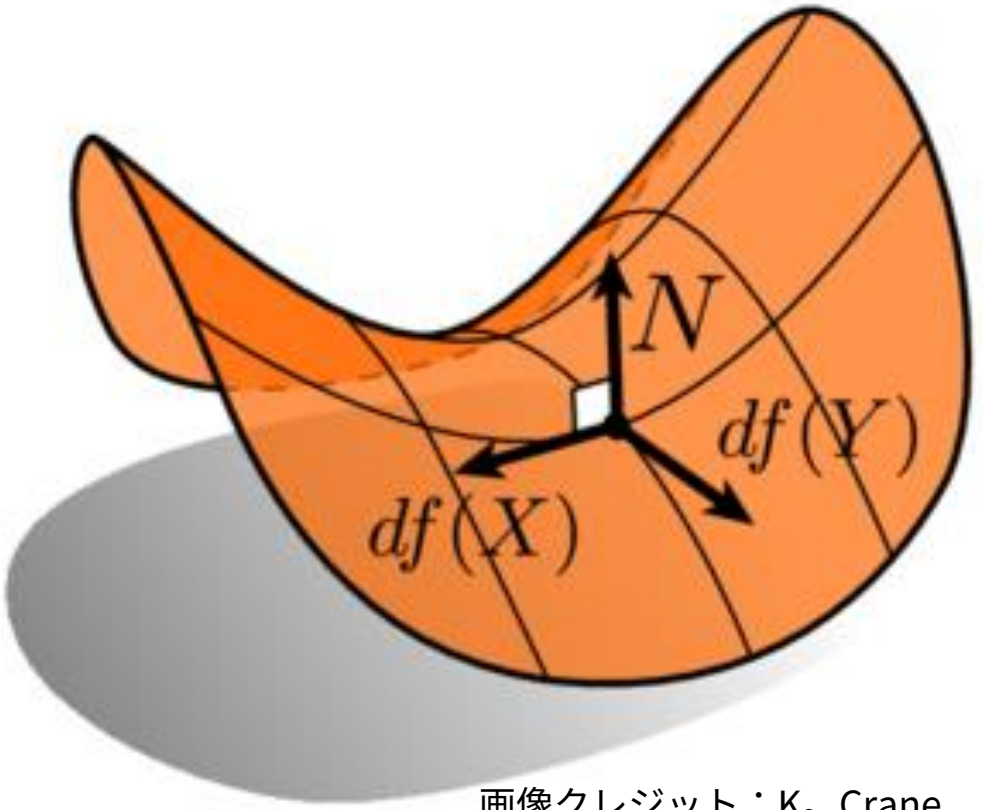
→：表面に対する単位法線ベクトル

→：現在の点から光源までの単位ベクトル

さまざまな照明角度で見られる球

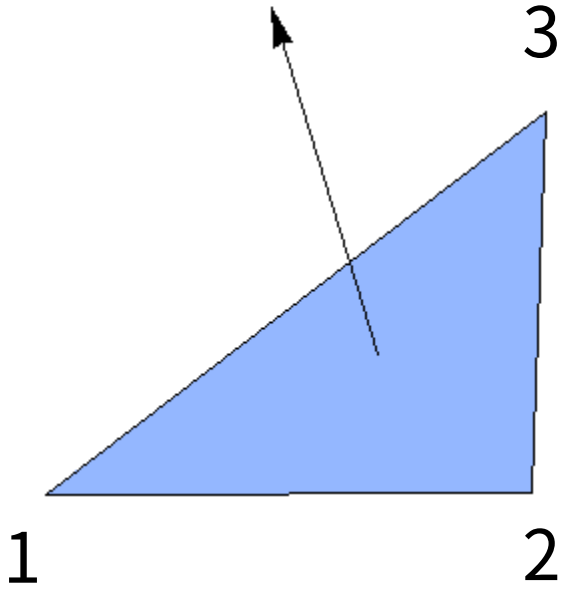
表面法線

法線は、特定の点ですべての接線に直交するベクトルです。



画像クレジット：K。Crane

三角形の法線



$$= \frac{1 \times 2}{\| 1 \times 2 \|}$$

$$1 = 2 - 1$$

$$2 = 3 - 1$$

三角形の法線

```
ボイド クロス（浮く* u、 浮く* v、 浮く* w）{
  w[0] = u[1] * v[2]- u[2] * v[1];
  w[1] = u[2] * v[0]- u[0] * v[2];
  w[2] = u[0] * v[1]- u[1] * v[0];
}

ボイド ノーマライズ（浮く* u）{
  浮く umag = sqrt（u[0] *u[0] +u[1] *u[1] +u[2] *u[2]）;
  もし （fabs（umag）<1e-5f）umag = 1.f;
  u[0] = u[0] / umag; u[1] = u[1] / umag; u[2] = u[2] / umag;
}

ボイド サブ（浮く* p1、 浮く* p2、 浮く* v21）{
  v21[0] = p1[0]- p2[0];
  v21[1] = p1[1]- p2[1];
  v21[2] = p1[2]- p2[2];
}
```

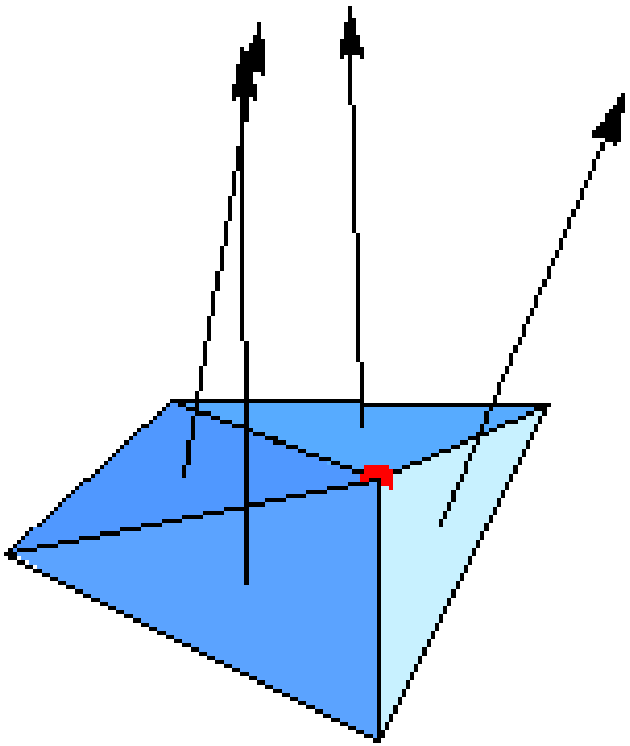
三角形の法線

```
ボイド computeNormal（浮く* p1、 浮く* p2、
                        浮く* p3、 浮く* 正常）。
{{
  浮く v1 [3]; サブ（p2、 p1、v1）;
  浮く v2 [3]; サブ（p3、 p1、v2）;
  cross（v1、v2、正常）; ノーマライ
  ズ（正常）;
}
```

OpenGLの法線を指定します

```
ボイド
drawTriangle（浮く* p0、 浮く* p1、 浮く* p2、
                浮く* 正常）。
{{
  glBegin（GL_TRIANGLES）;
  glNormal3fv（正常）;
  glVertex3fv（p0）;
  glVertex3fv（p1）;
  glVertex3fv（p2）;
  glEnd（）;
}
```

頂点法線



赤い頂点で法線を計算するには：

1.隣接する各面の法線を計算します

2.これらの法線の加重和を計算します

可能な重み：

- 1
- 三角形の領域
- …

OpenGLに垂直な（頂点ごとの）指定

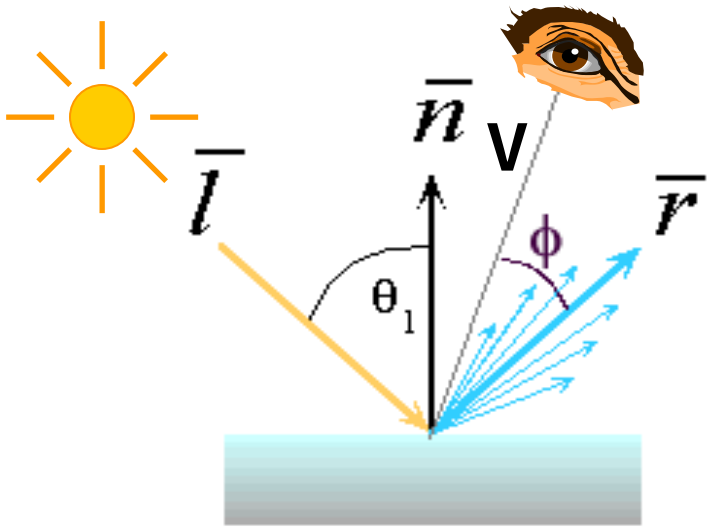
```
ボイド
drawTriangle（浮く* p0、浮く* p1、浮く* p2、
                浮く* n0、浮く* n1、浮く* n2）。
{{
  glBegin（GL_TRIANGLES）;
  glNormal3fv（n0）;
  glVertex3fv（p0）;
  glNormal3fv（n1）;
  glVertex3fv（p1）;
  glNormal3fv（n2）;
  glVertex3fv（p2）;
  glEnd（）;
}
```

鏡面反射

フォン鏡面反射モデル

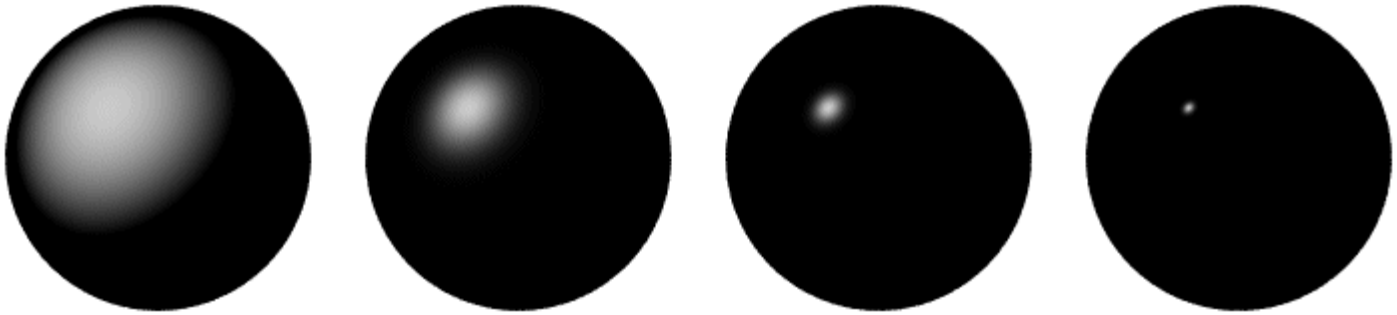
= cos（　　）

= 最大。（　　）



k_s **鏡面反射** 係数

光沢係数（m）が異なる球

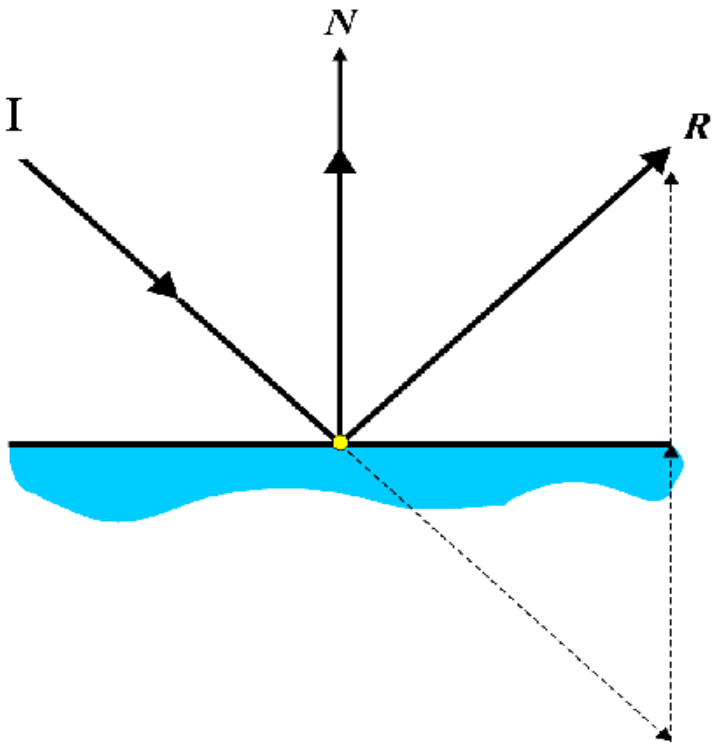


反射ベクトルの計算

I：インシデントベクトル

R：反射ベクトル（純粋反射）

IとNの両方が単位である場合、Rも単位です。



$$R = I + 2N\,(-私\ \ N)$$

$$= 私 - 2N\,(I\ \ N)$$

ハーフベクトルバリアント（ブリン）

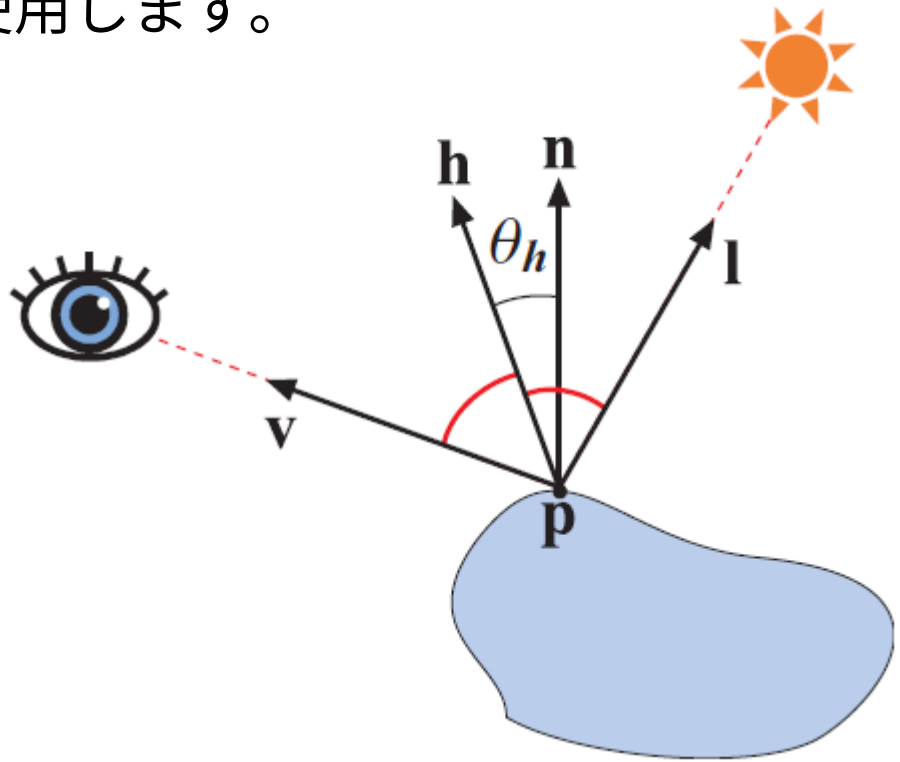
代わりにスペキュラー用語を使用します。

= cos *h*( )

= 最大。( )

どこ

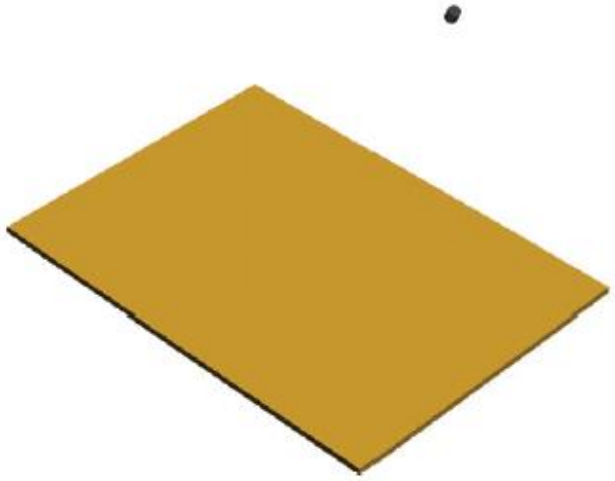
=



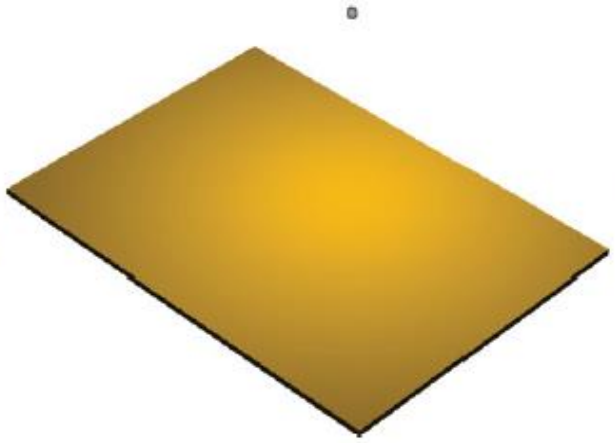
リアルタイムレンダリング、Haines etal。

光源

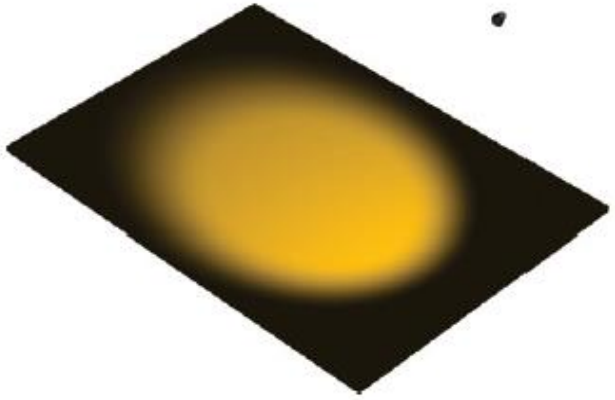
- 指向性ライト
- ポイントライト
- スポットライト



方向性



ポイントライト

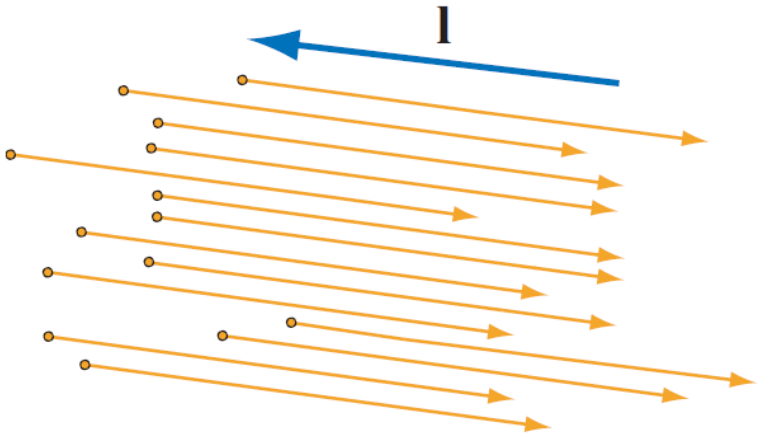


スポットライト

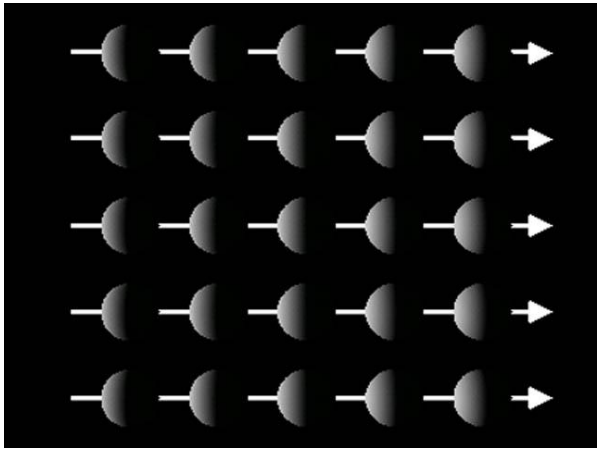
リアルタイムレンダリング、Haines etal。

指向性ライト

- 無限に遠く離れた光源から来る平行な光線

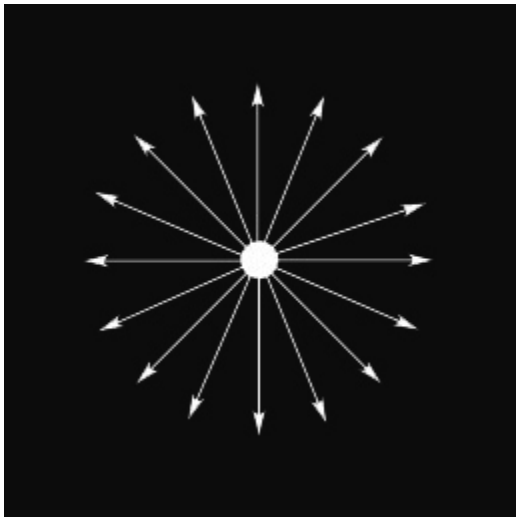


- 太陽のような光源をモデル化するために使用されます

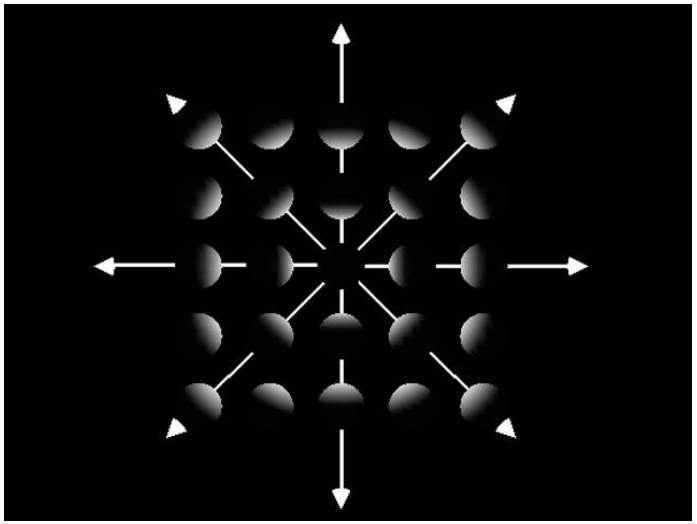


ポイントライト

- 光は一点から全方向に放射されます
- 強度は、光源からの距離dとともに減少します。



$$f_{att}(d) = \frac{1}{k_0 + k_1 d + k_2 d^2}$$

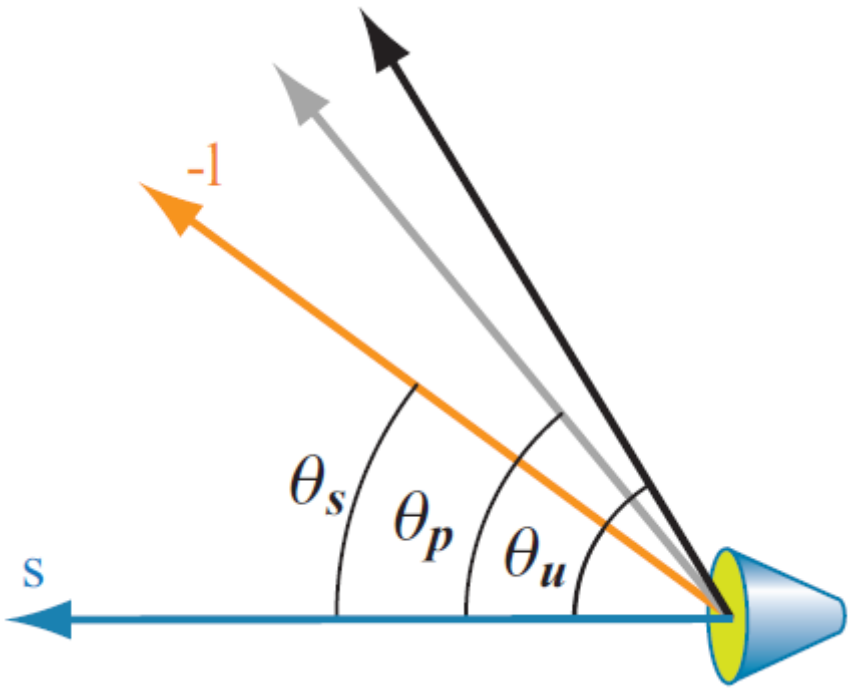


スポットライト

- 強度の方向性の変動

（cos ）、 ≦

0、 >



照明は加法です：個々の光源の寄与を合計する必要があります

$$h = (\text{最大。、0} + \max(x(.,)0)) +$$

ポリゴンのレンダリング方法

フラットシェーディング、 単一の強度は、三角形ごとに計算されます。フラットシェーディングは、隣接する三角形の共通のエッジに沿って強度の間に不連続性を生成します。
グーローシェーディング、 頂点で計算された強度は、三角形全体に補間されます。
フォンシェーディング、 頂点法線は三角形全体に補間され、強度は補間されたものを使用してフラグメントごとに計算されます
正常。

平らな　　グーロー　　フォン

リアルタイムレンダリング、Haines etal。

OpenGL照明モデル

- OpenGLの固定レンダリングパイプラインは、グローバルアンビエントタームとエミッションタームが追加されたBlinn-Phongと非常によく似たモデルを使用します
- 放射項：光るオブジェクト（電球など）をシミュレートします

OpenGL：ライトプロパティ

glLight {if}（GLenum *光*、グレナム *pname*、タイプ *param*）；
glLight {if} v（GLenum *光*、グレナム *pname*、タイプ*param*）；

IDでライトのパラメータを指定します ***光***、 これは **GL_LIGHT0**、…、**GL_LIGHT7**

（少なくとも8つのライトが利用可能です）

pname	デフォルト	意味
GL_AMBIENT	（0.0,0.0,0.0,1.0）	周囲光強度拡散光強
GL_DIFFUSE	ライト0の場合は（1.0,1.0,1.0,1.0） その他の場合は（0.0,0.0,0.0,1.0）	度
GL_SPECULAR	ライト0の場合は（1.0,1.0,1.0,1.0） その他の場合は（0.0,0.0,0.0,1.0）	鏡面光強度
GL_POSITION	（0.0,0.0,0.0,1.0）	（x、y、z、w）光の位置（x、
GL_SPOT_DIRECTION	（0.0,0.0,0.0,1.0）	y、z）スポットライトの方向
GL_SPOT_EXPONENT	0.0	スポットライト指数
GL_SPOT_CUTOFF	180.0	スポットライトカットオフ角度
GL_CONSTANT_ATTENUATION	1.0	一定の減衰係数
GL_LINEAR_ATTENUATION	0.0	線形減衰係数二次減衰係
GL_QUADRATIC_ATTENUATION	0.0	数

OpenGL：指向性ライト

例：方向（3、3、3）で指向性ライトを作成します

GLfloat l_dir [] = {3.0f、3.0f、3.0f、0.0f};

glLightfv（GL_LIGHT0、GL_POSITION、l_dir）;

同次座標では、　、　、　、0　（　　）
3Dベクトルに対応します 、(、 。)

スポットライト

（c os ）、 ≦

0、 >

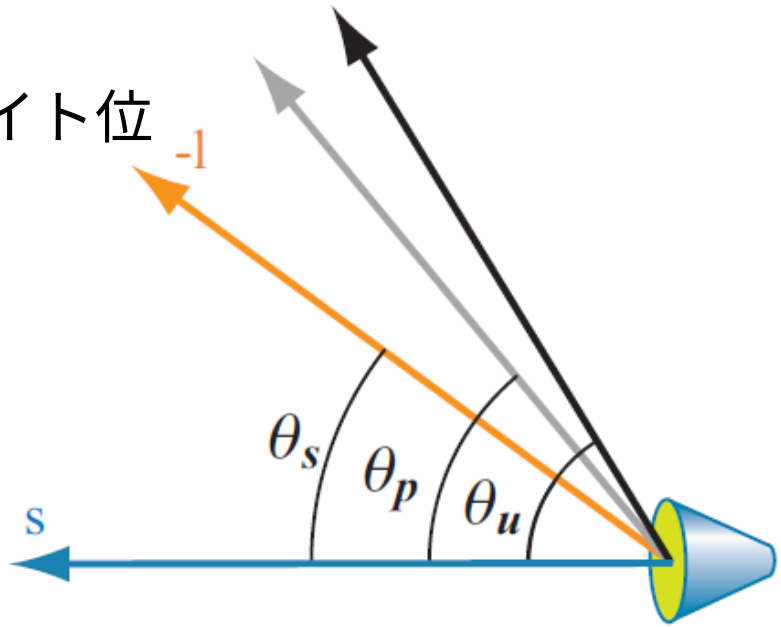
cos = 。 (–)

s： GL_SPOT_DIRECTION

l： サーフェスポイントからスポットライト位置までの光の方向（GL_POSITION）

：GL_SPOT_CUTOFF

m：GL_SPOT_EXPONENT



材料特性

glMaterial {if}（GLenum *面*、グレナム *pname*、タイプ *param*）;
glMaterial {if} v（GLenum *面*、グレナム *pname*、タイプparm*）;**

照明計算で使用する現在の材料特性を指定します。

面= GL_FRONT、GL_BACK または GL_FRONT_AND_BACK

pname	デフォルト	意味
GL_AMBIENT	(0.2,0.2,0.2,1.0)	マテリアルのアンビエントカ
GL_DIFFUSE	(0.8,0.8,0.8,1.0)	ラーマテリアルの拡散カラーア
GL_AMBIENT_AND_DIFFUSE		ンビエントカラーと拡散カラー
GL_SPECULAR	(0.0,0.0,0.0,1.0)	マテリアルのスペキュラカラー
GL_SHININESS	0.0	スペキュラ指数
GL_EMISSION	(0.0,0.0,0.0,1.0)	素材の発光色
GL_COLOR_INDEXES	(0,1,1)	アンビエント、ディフューズ、スペキュラーカラーインデックス

モデルのシェーディングと照明の有効化

glShadeModel（GL_FLAT）;

ポリゴンのフラットシェーディングモデル

glShadeModel（GL_SMOOTH）;

ポリゴンのスムーズ（グーロー）シェーディングモデル

glEnable（GL_LIGHTING）;

照明を有効にします

glDisable（GL_LIGHTING）;

照明を無効にします

glEnable（GL_LIGHT0）;

光源0を有効にします

glEnable（GL_LIGHT1）;

光源1を有効にします…

GL_FLAT

GL_SMOOTH

OpenGLでの照明/マテリアルの指定

1）材料特性を指定します。

void glMaterialfv（GLenum face、GLenumparam、GLfloat * value）;

顔= {GL_FRONT | GL_BACK | GL_FRONT_AND_BACK}
param = {GL_AMBIENT | GL_DIFFUSE | GL_EMISSIVE | GL_SPECULAR} value
= float [4] // RGBA

void glMaterialf（GLenum face、GL_SHININESS、GLfloat value）;

2）周囲光を指定します。

void glLightModelfv（GLenumparam、GLfloat * value）;

param = GL_LIGHT_MODEL_AMBIENT
value = float [4] // RGBA

3）ライトの色を指定します。

void glLightfv（GLenum light、GLenumparam、GLfloat * value）;

ライト= {GL_LIGHT0 | GL_LIGHT1 |…}
param = {GL_AMBIENT | GL_DIFFUSE | GL_SPECULAR}
value = float [4] // RGBA

照明のプログラミング（続き）

4）ライトの位置/方向の位置を指定します。

void glLightfv（GLenum light、GL_POSITION、GLfloat * value）;
light = {GL_LIGHT0 | GL_LIGHT1 |…}
value = float [4] // x、y、z、w

光源の座標は、現在の変換行列によって変換されます

5）照明を有効にする

void glEnable（GLenum type）; タイプ= GL_LIGHTING;

6）個々のライトを有効にする

void glEnable（GLenum type）; タイプ= GL_LIGHT0、GL_LIGHT1、…

7）シェーディングモデルを指定します

void glShadeModel（GLenum type）;

タイプ= GL_FLAT – フラットシェーディング

タイプ= GL_SMOOTH – スムーズシェーディング（グーローシェーディング）