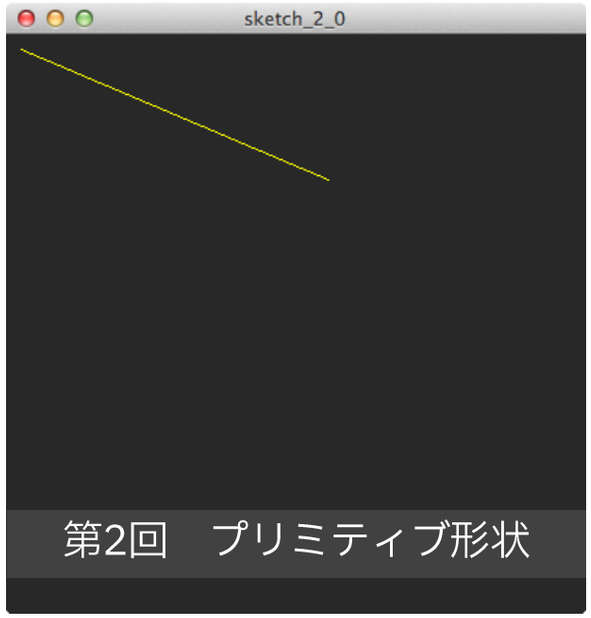
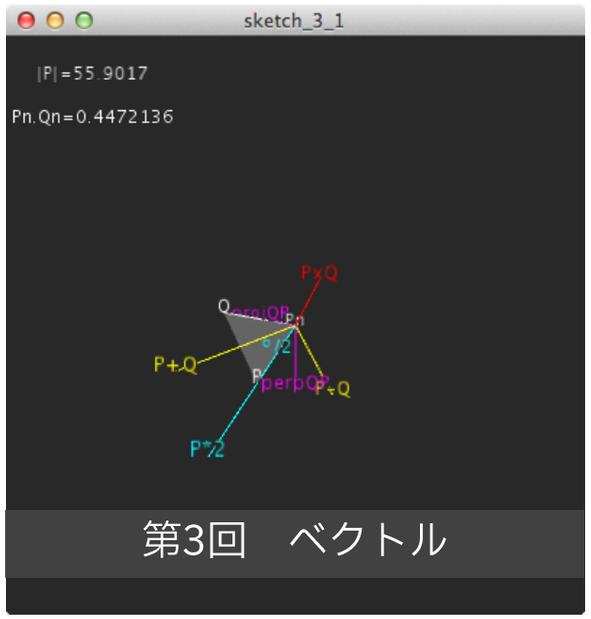
CGとCADの数理

GEOMETRIC MODELING AND COMPUTER GRAPHICS

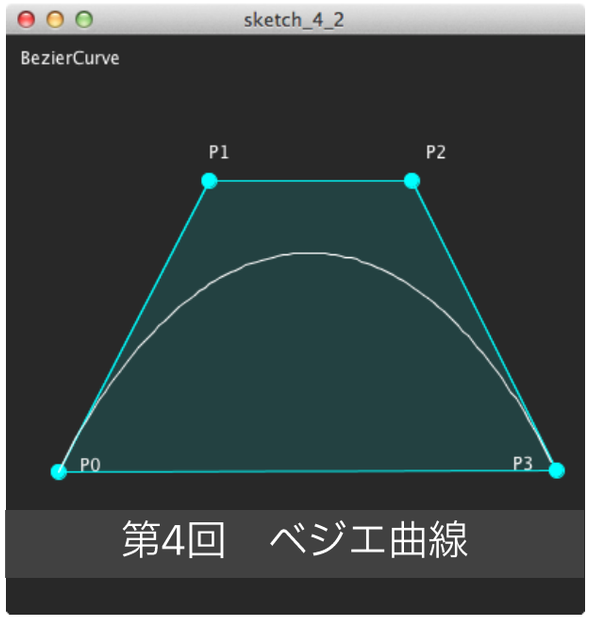
第11回 レイトレーシング



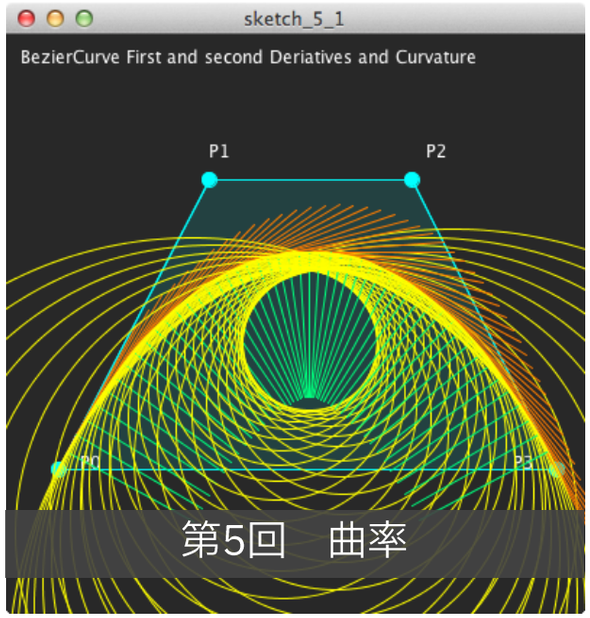
第2回　プリミティブ形状



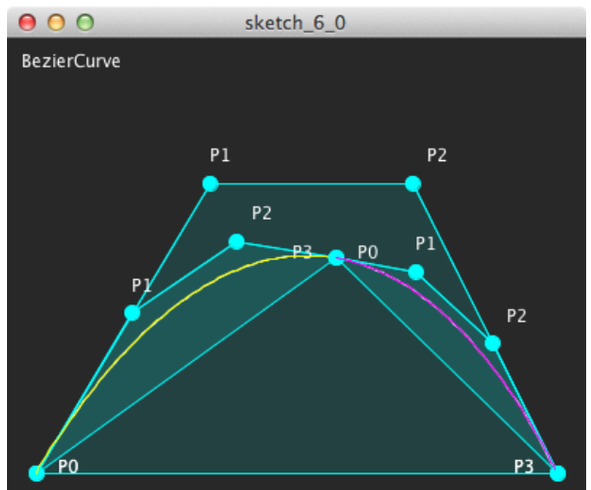
第3回　ベクトル



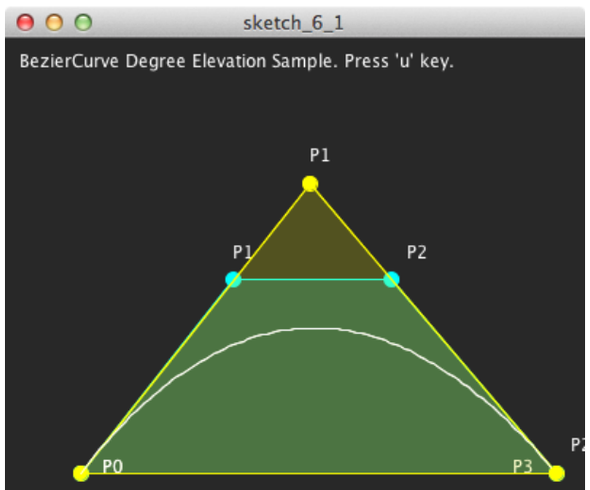
第4回　ベジエ曲線

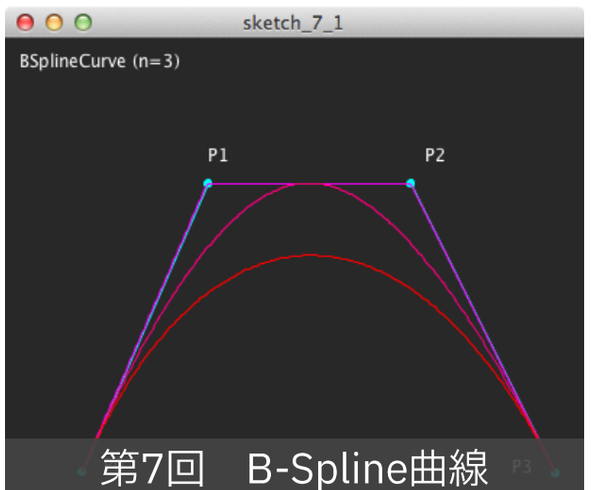


第5回　曲率



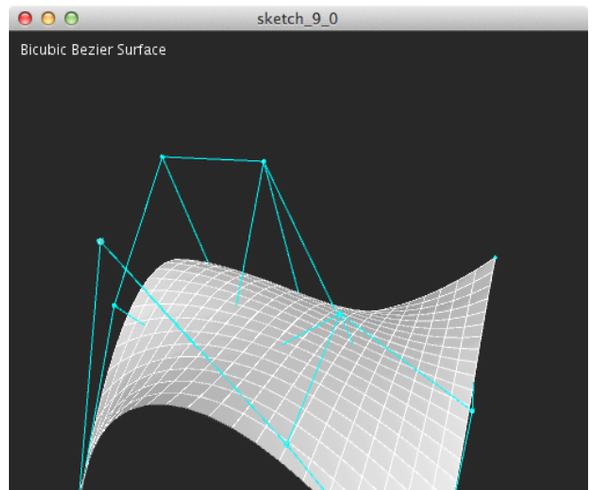
第6回　次数上げ、分割と接続





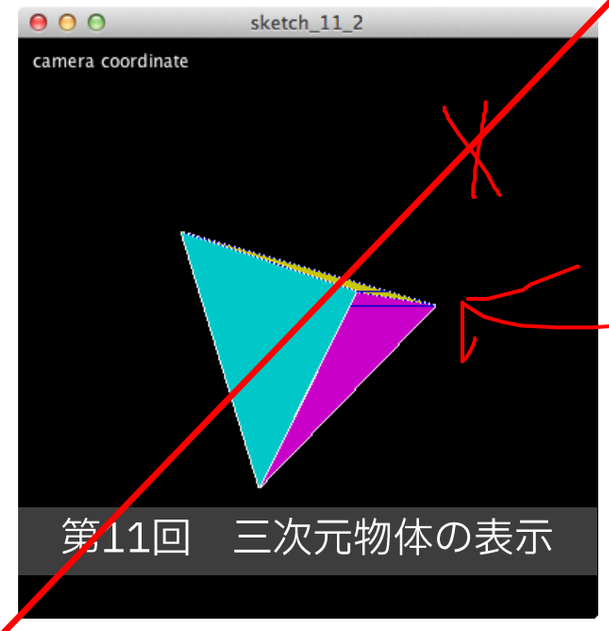
第7回　B-Spline曲線

第8回　NURBS曲線



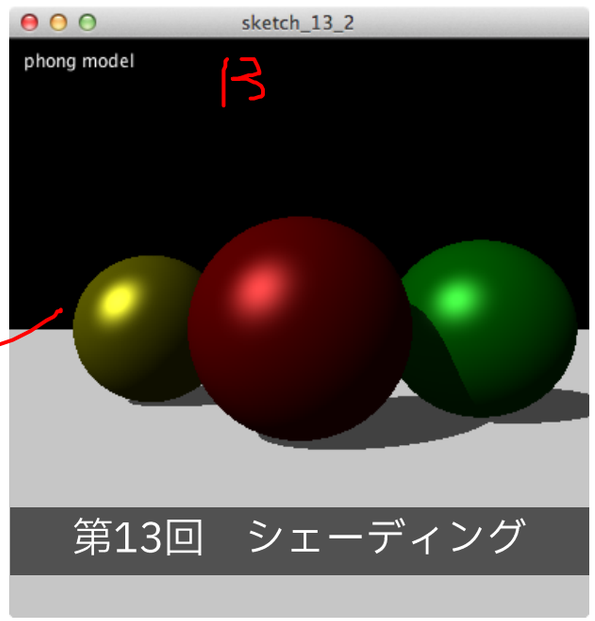
第9回　パラメトリックな曲面

第10回　行列



第11回　三次元物体の表示

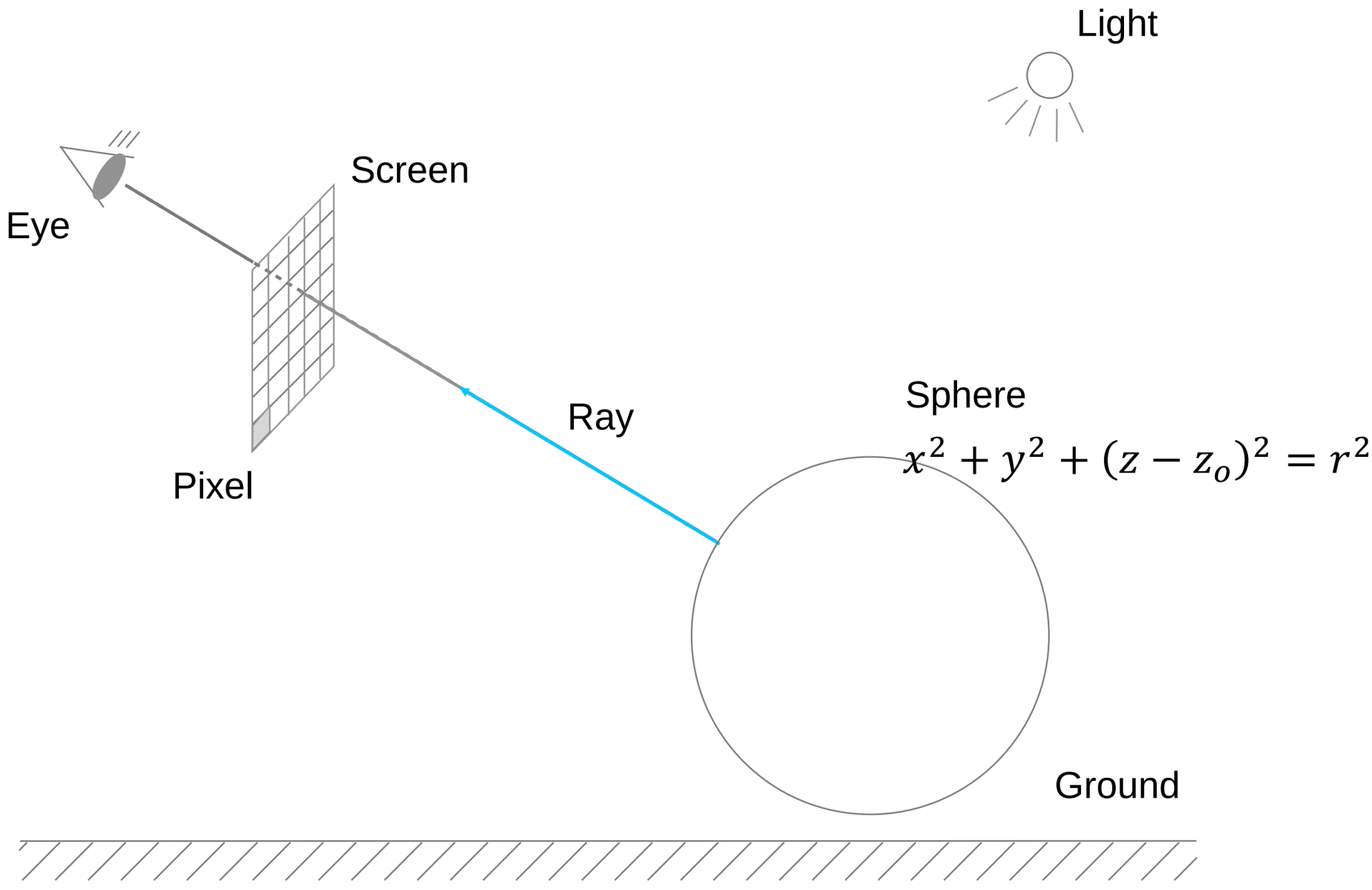
第12回　レイトレーシング



第13回　シェーディング

sketch_11_0.pde をダウンロードして下さい

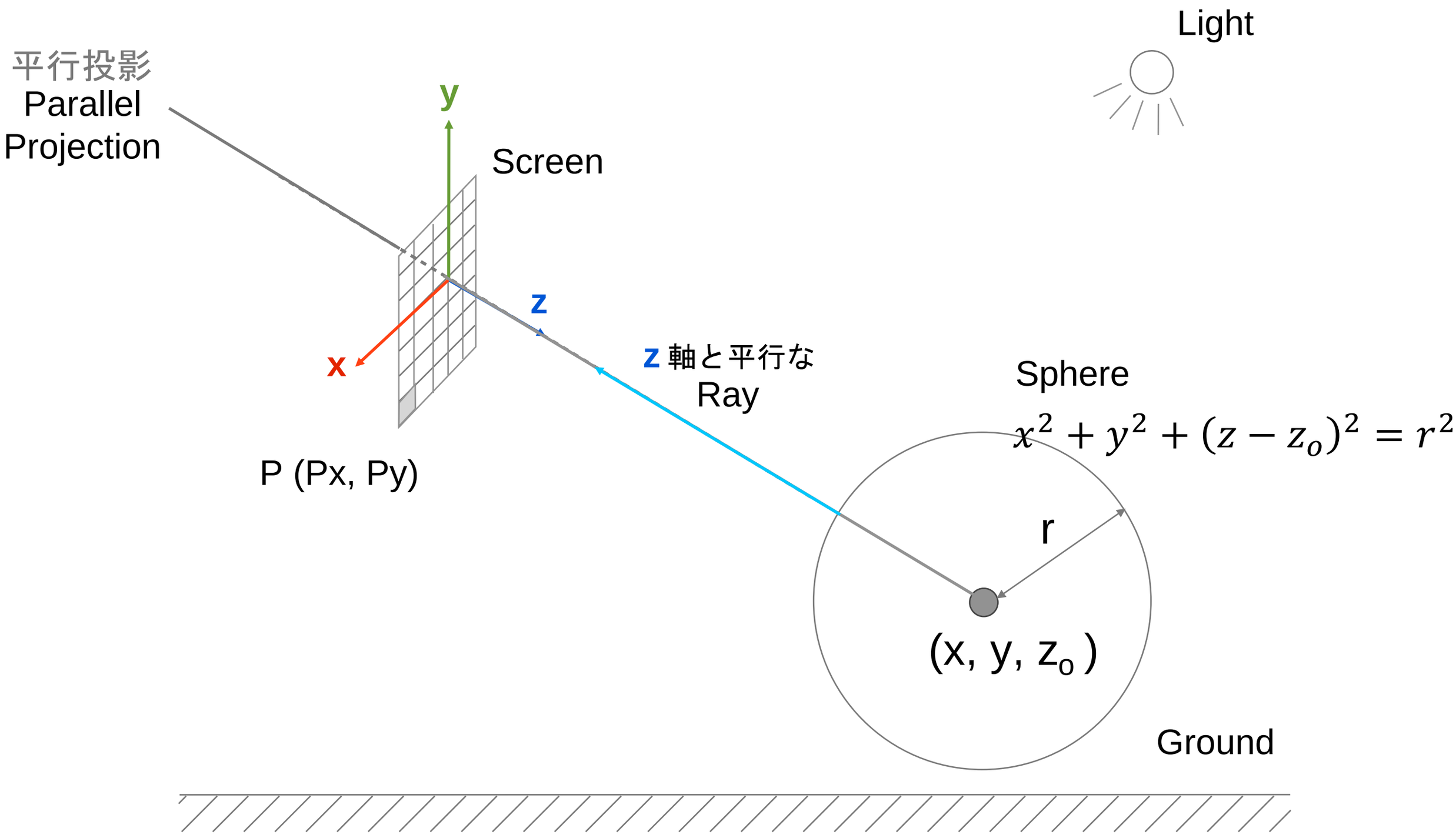
Ray Tracing （光線追跡法）



スクリーン上の Pixel から視点へ入射してくる光線（ Ray ）を **逆方向に** 追跡（ Trace ）する手法

Ray Tracing （光線追跡法）

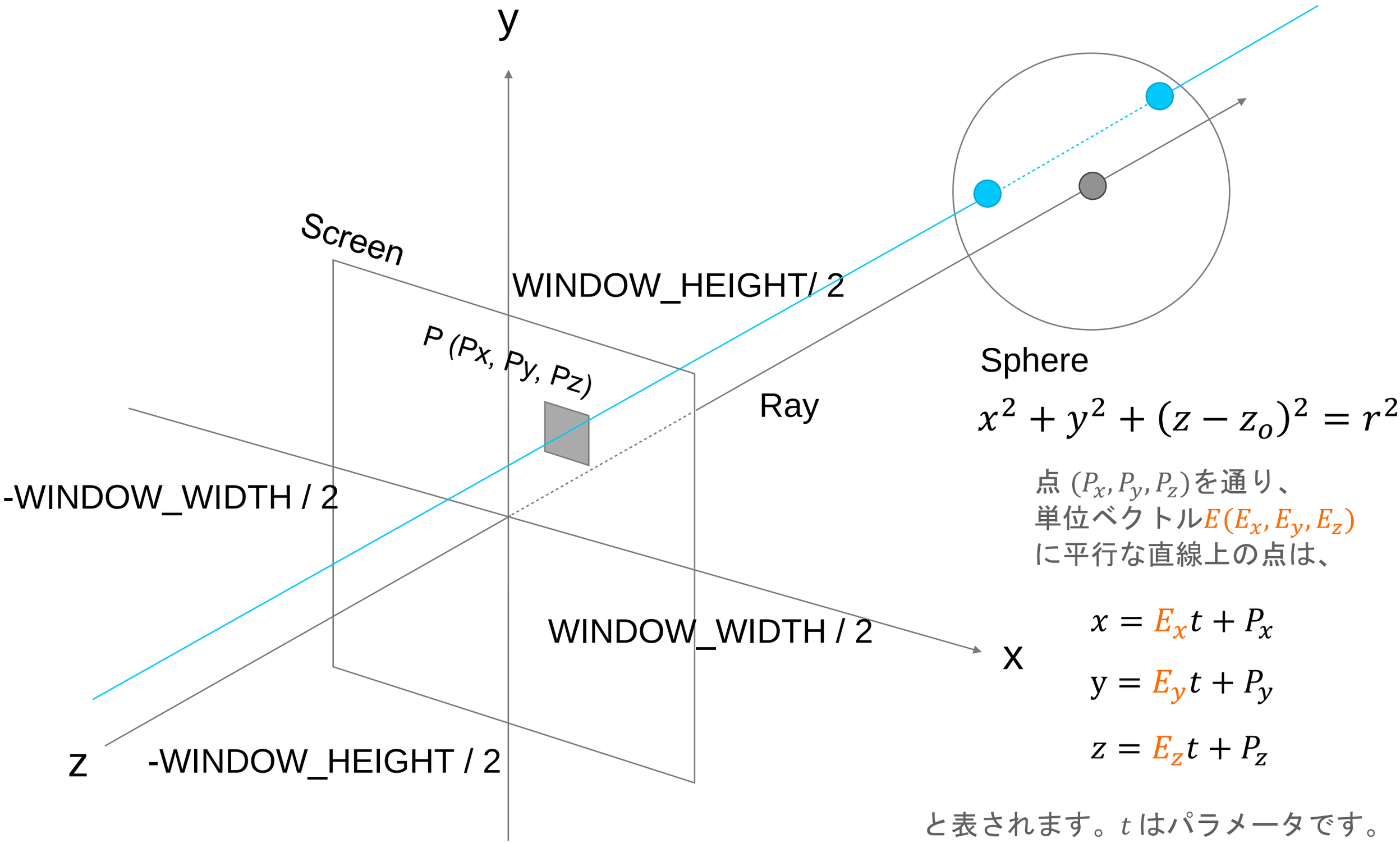
Super Simple Ray Tracing （超シンプル光線追跡法）



P (Px, Py) を通して視えるものを決定する

Super Simple Ray Tracing （超シンプル光線追跡法）

Ray を考えよう



$$x^2 + y^2 + (z - z_o)^2 = r^2$$

点 (P_x, P_y, P_z)を通り、
単位ベクトル$E(E_x, E_y, E_z)$
に平行な直線上の点は、

$$x = E_x t + P_x$$

$$\mathrm{y} = E_y t + P_y$$

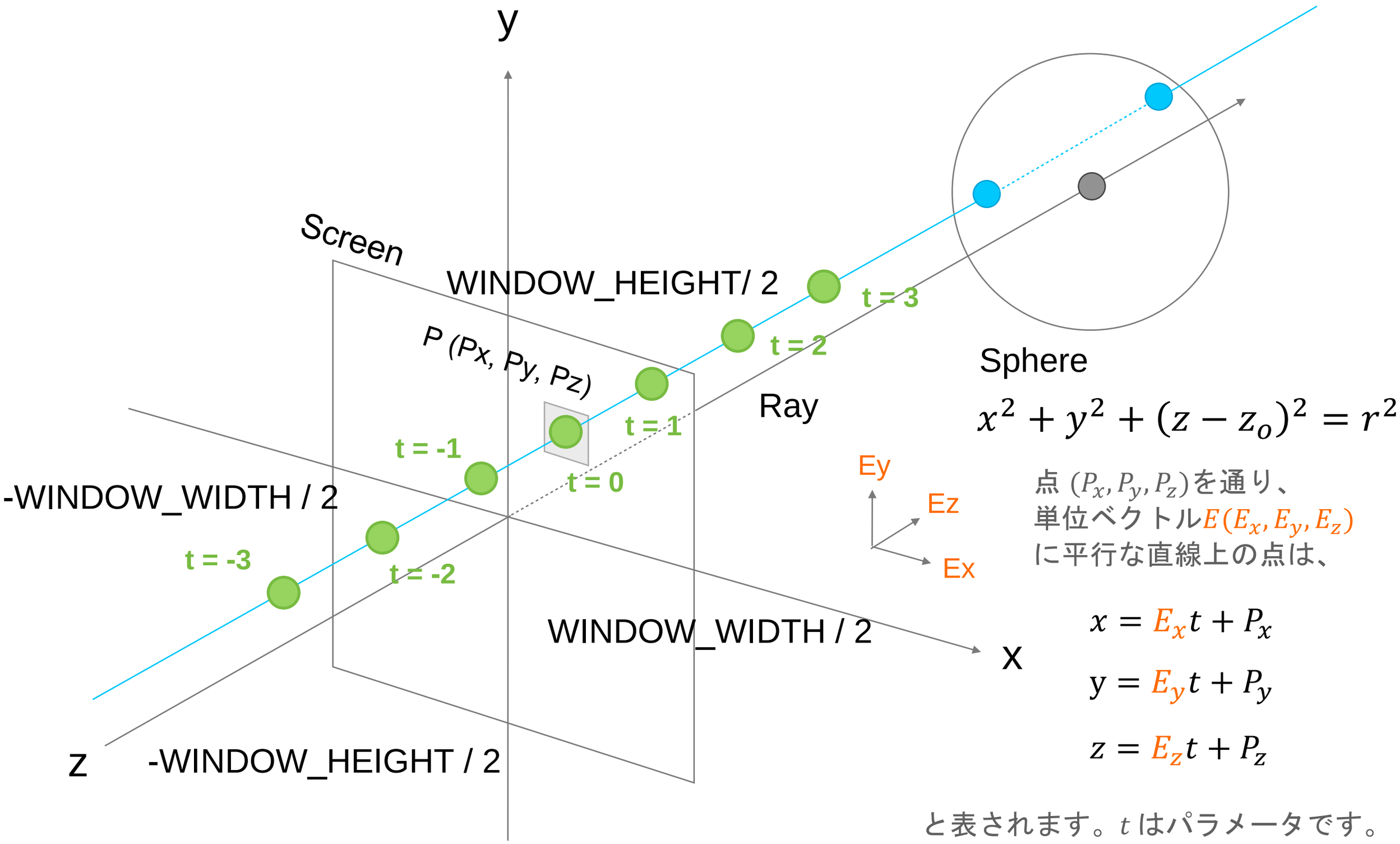
$$z = E_z t + P_z$$

と表されます。t はパラメータです。

$P(P_x, P_y)$ を通り Z軸に平行な直線と球 (Sphere)との交点を求め、手前の交点の色が見える。

Super Simple Ray Tracing （超シンプル光線追跡法）

Ray を考えよう



点 (P_x, P_y, P_z)を通り、
単位ベクトル$E(E_x, E_y, E_z)$
に平行な直線上の点は、

$$x = E_x t + P_x$$

$$y = E_y t + P_y$$

$$z = E_z t + P_z$$

と表されます。t はパラメータです。

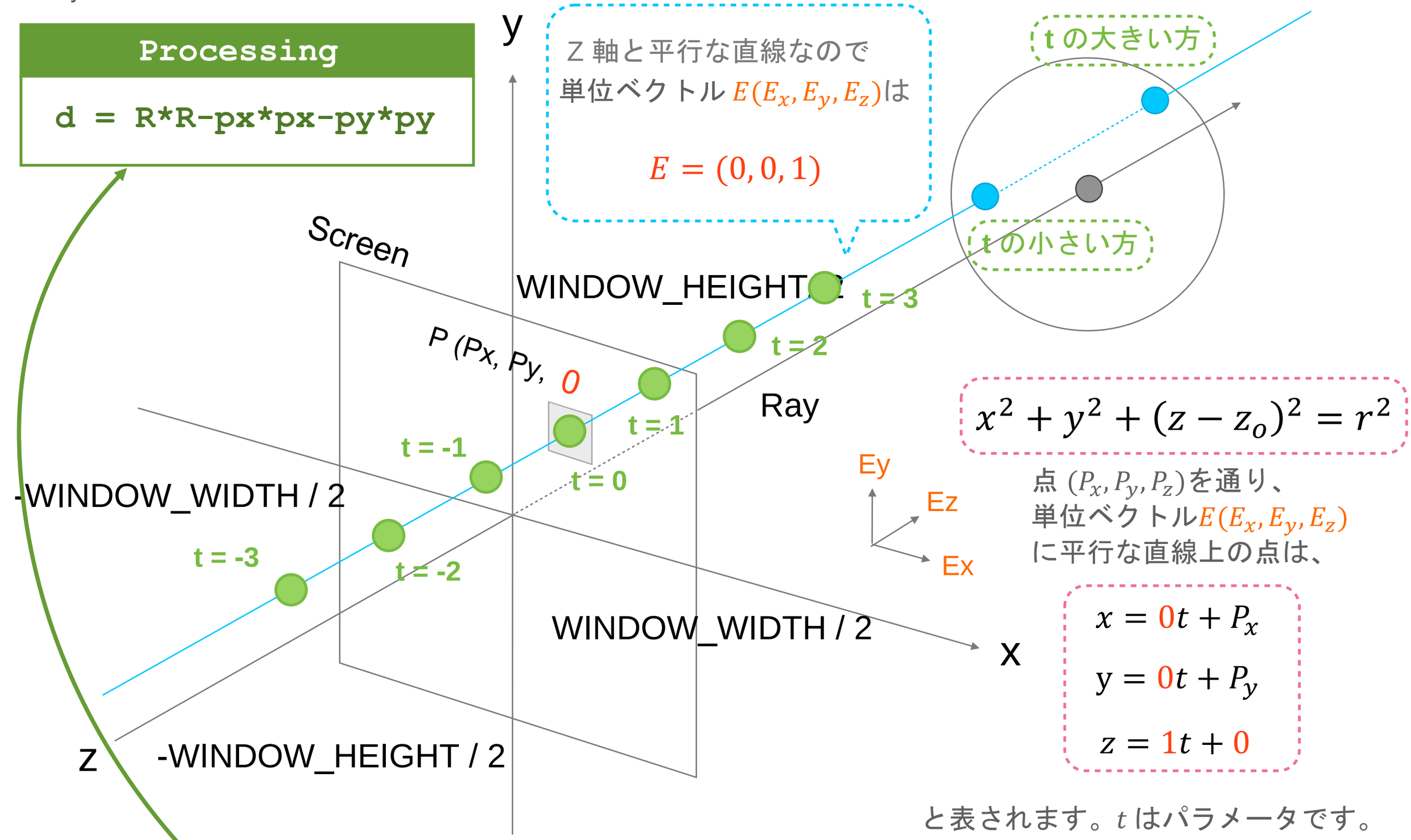
$E(E_x, E_y, E_z)$のように方向を示す単位ベクトルを **方向余弦** と言うことがあります。

Super Simple Ray Tracing （超シンプル光線追跡法）

Rayと球との交点 を考えよう

Processing

```
d = R*R-px*px-py*py
```



$$x^2 + y^2 + (z - z_o)^2 = r^2$$

点 (P_x, P_y, P_z)を通り、
単位ベクトル$E(E_x, E_y, E_z)$
に平行な直線上の点は、

$$x = 0t + P_x$$
$$y = 0t + P_y$$
$$z = 1t + 0$$

と表されます。t はパラメータです。

$$P_x^2 + P_y^2 + (t - Z_o)^2 = r^2 \leftrightarrow t^2 - 2Z_o t + P_x^2 + P_y^2 + Z_o^2 - r^2 = 0$$

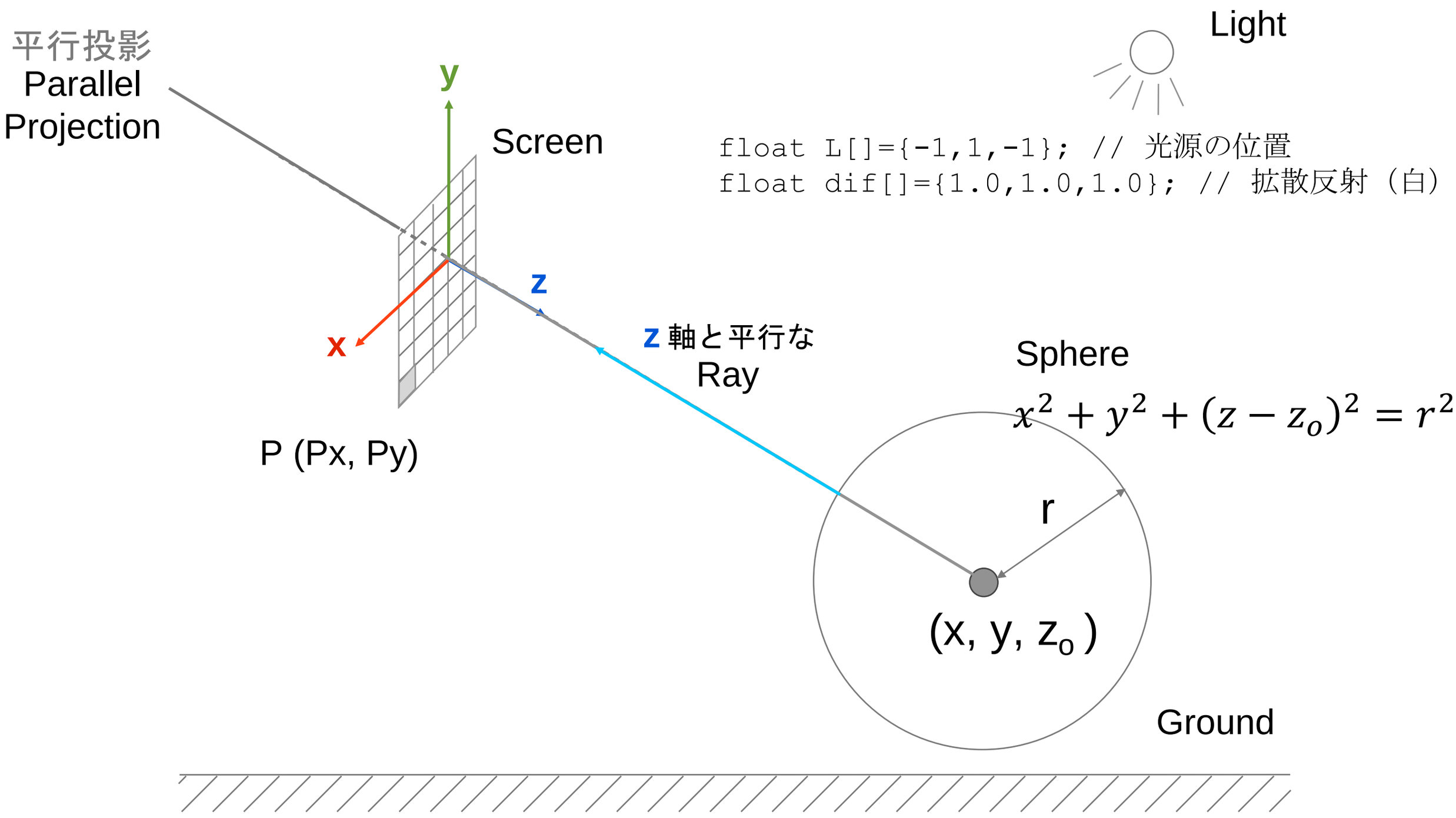
$$t = Z_o \pm \sqrt{r^2 - P_x^2 - P_y^2}$$

$r^2 - P_x^2 - P_y^2 < 0$ のとき、交点が存在しない

Ray Tracing

sketch_12_1.zip をダウンロードして下さい

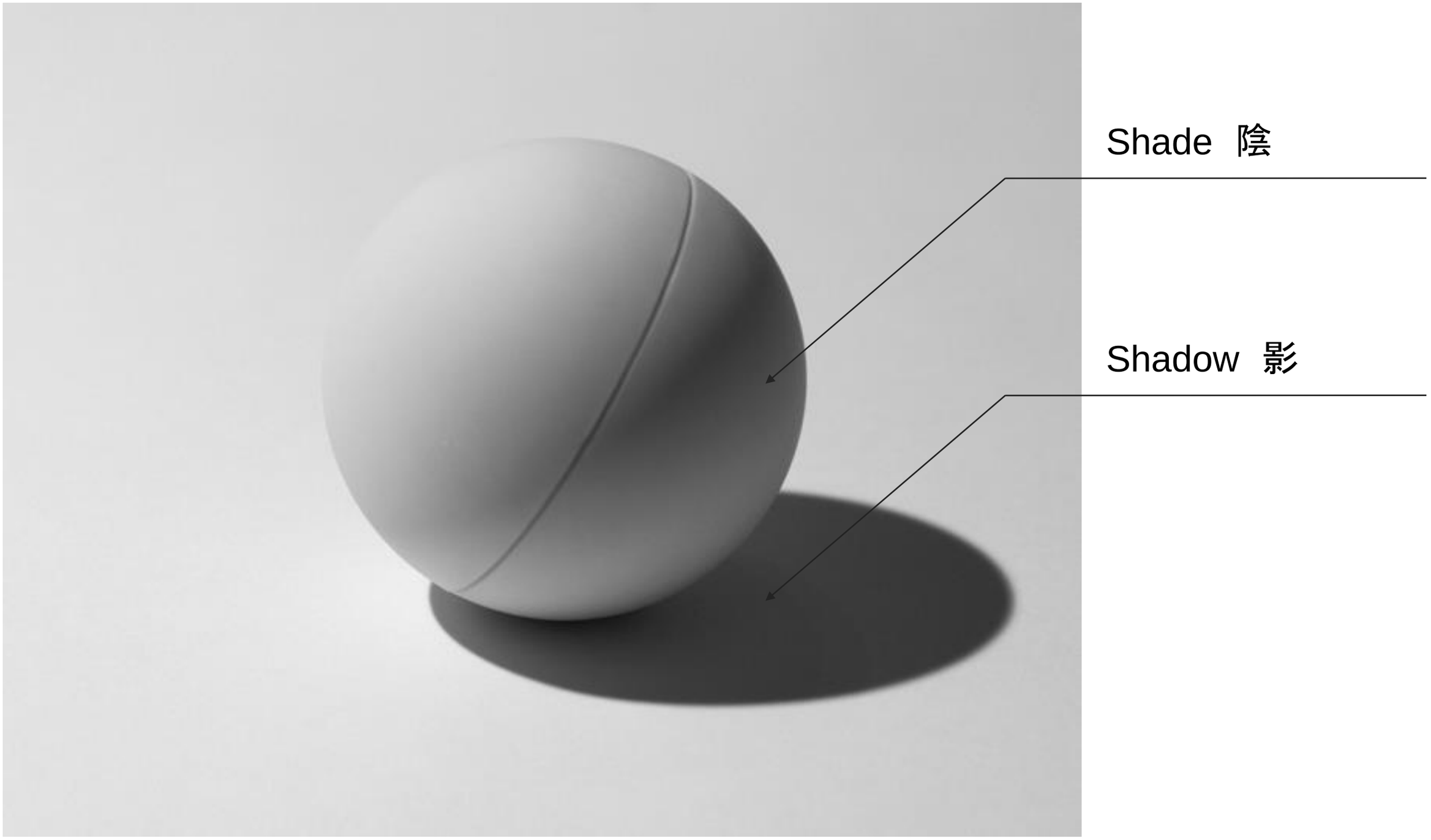
Super Simple Ray Tracing （超シンプル光線追跡法）




```
float L[]={-1,1,-1}; // 光源の位置
float dif[]={1.0,1.0,1.0}; // 拡散反射（白）
```

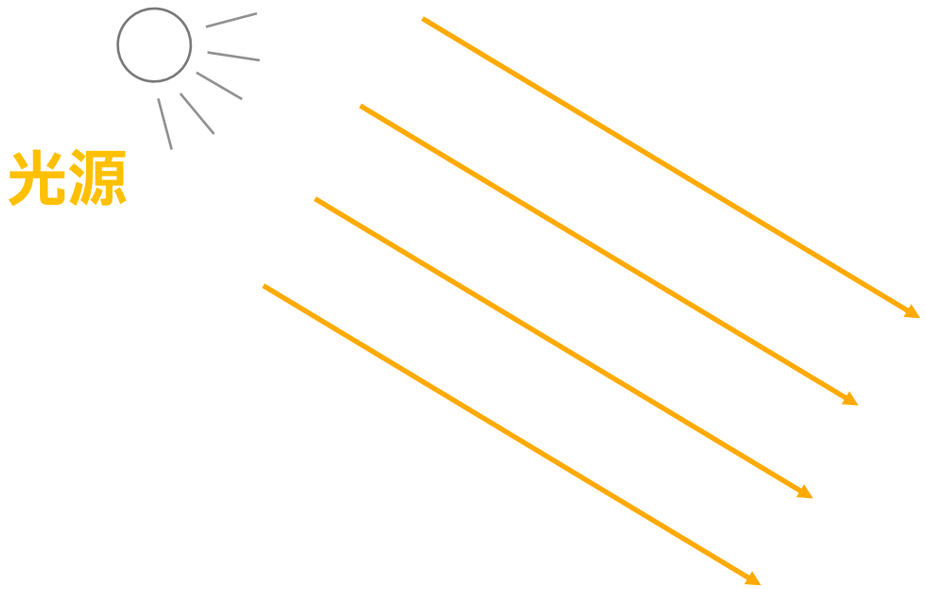
光源を計算に加えてみましょう。

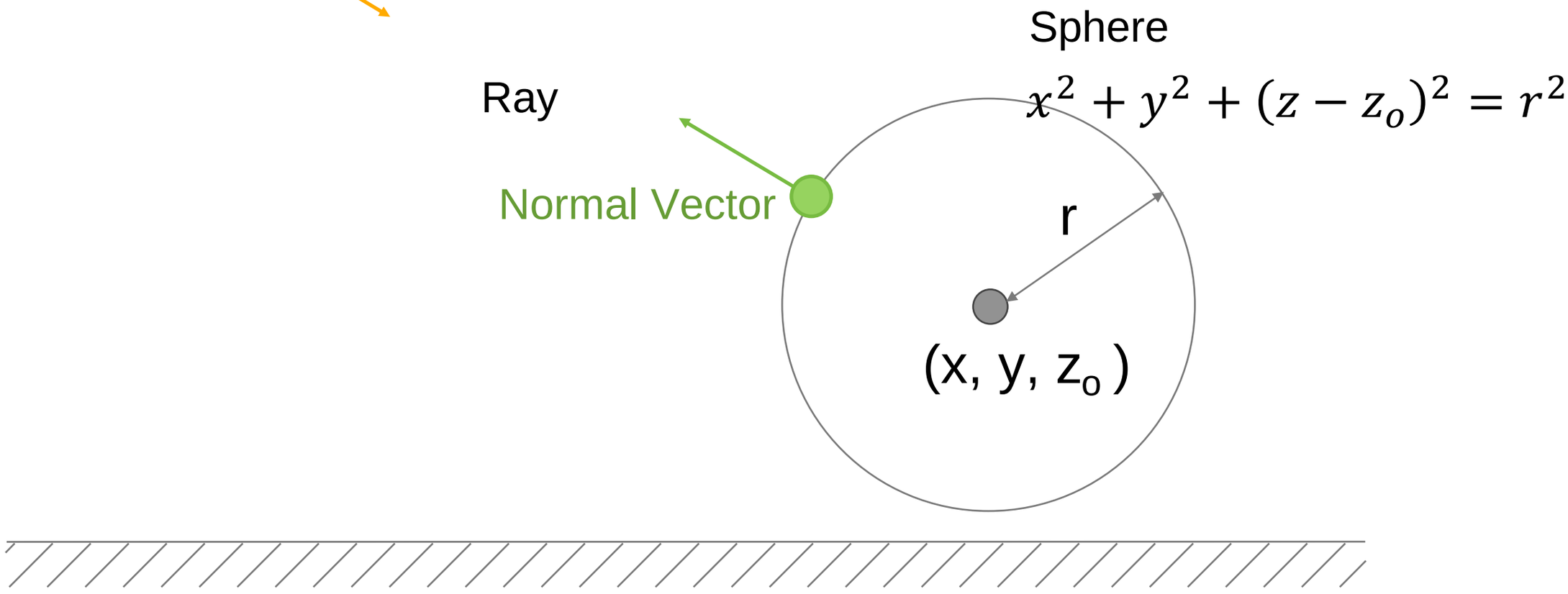
Shade and Shadow （陰影）



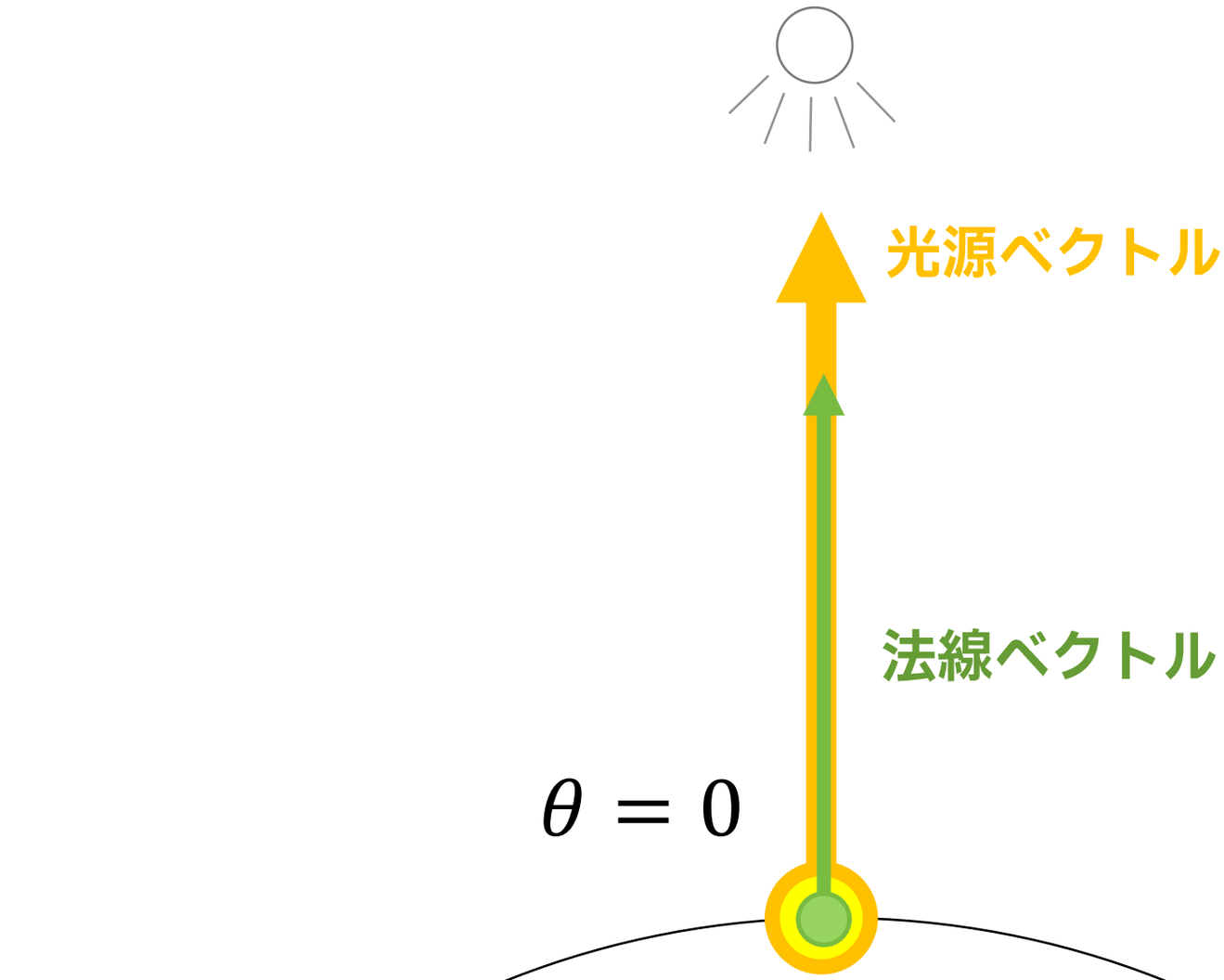
Shade（陰）をつけることを **Shading（シェーディング）** といいます。

Lambert’s cosine law （ランバートの余弦則）の概説





Lambert’s cosine law （ランバートの余弦則）の概説



光源ベクトル

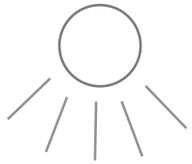
法線ベクトル

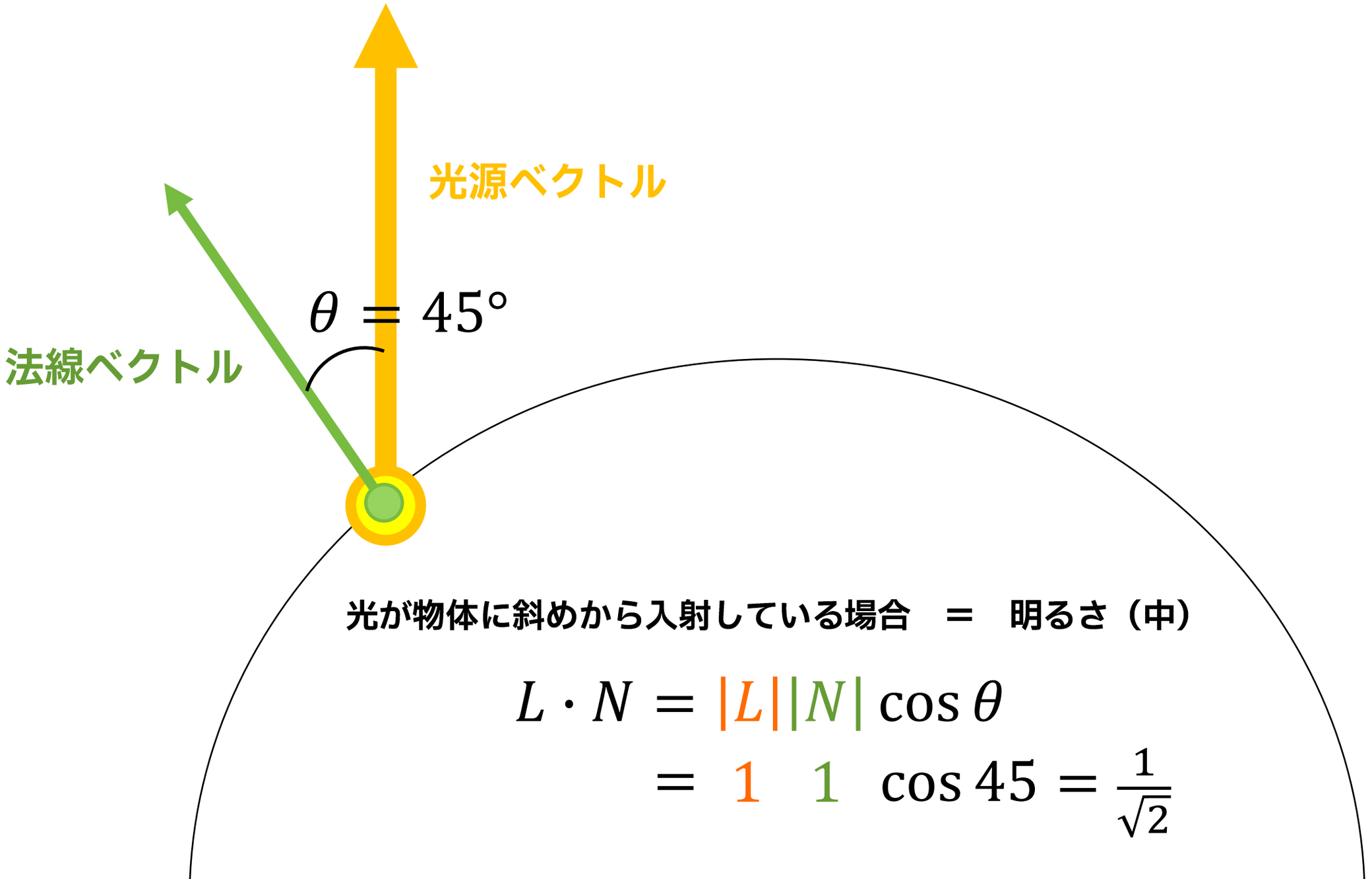
$\theta = 0$

光が物体に垂直に入射している場合　＝　明るさ（大）

$$L \cdot N = |L||N| \cos\theta$$
$$= 1 \quad 1 \quad \cos 0 = 1$$

Lambert’s cosine law （ランバートの余弦則）の概説



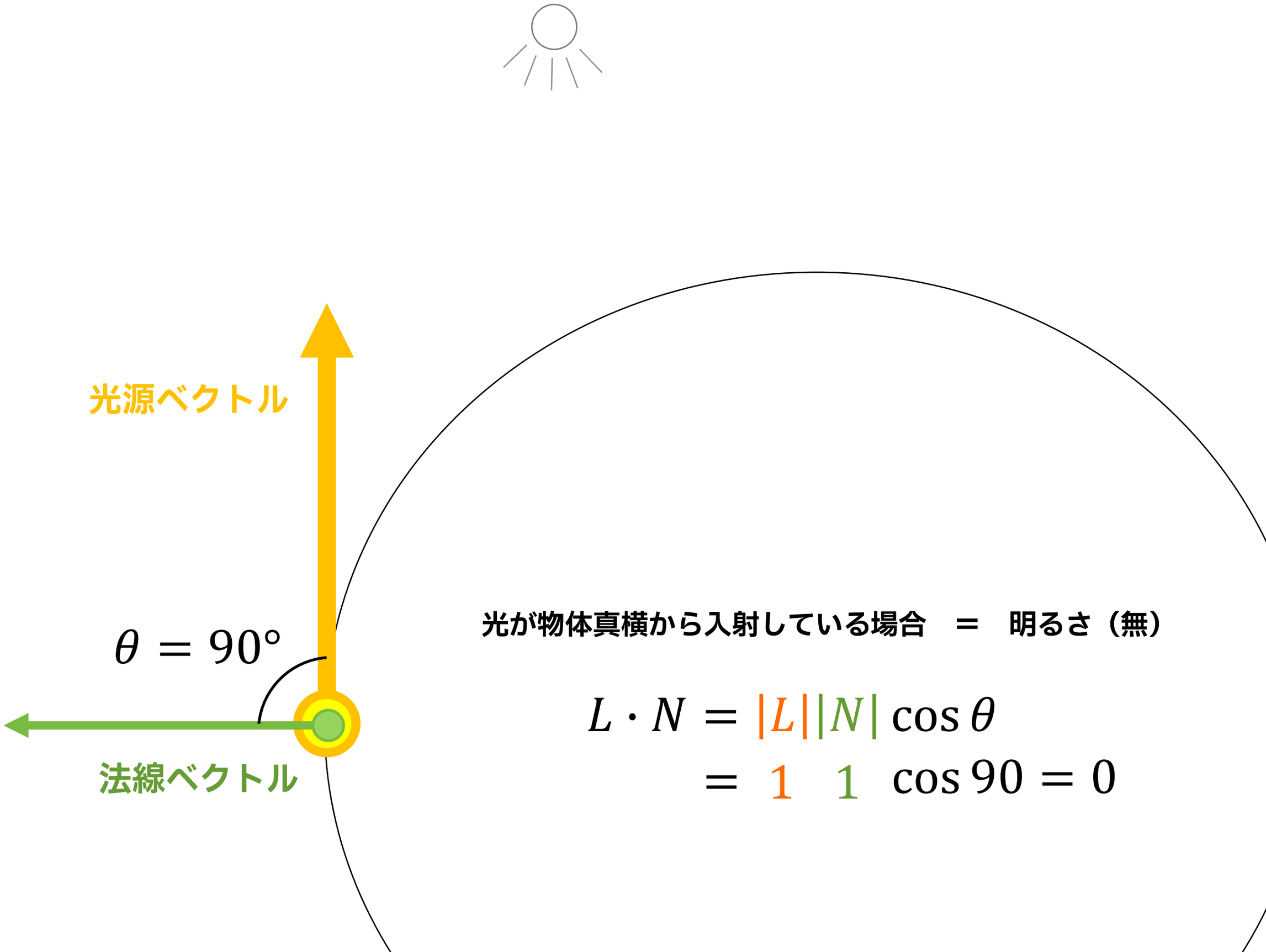


光が物体に斜めから入射している場合　＝　明るさ（中）

$$L \cdot N = |L||N| \cos\theta$$

$$= 1 \quad 1 \quad \cos 45 = \frac{1}{\sqrt{2}}$$

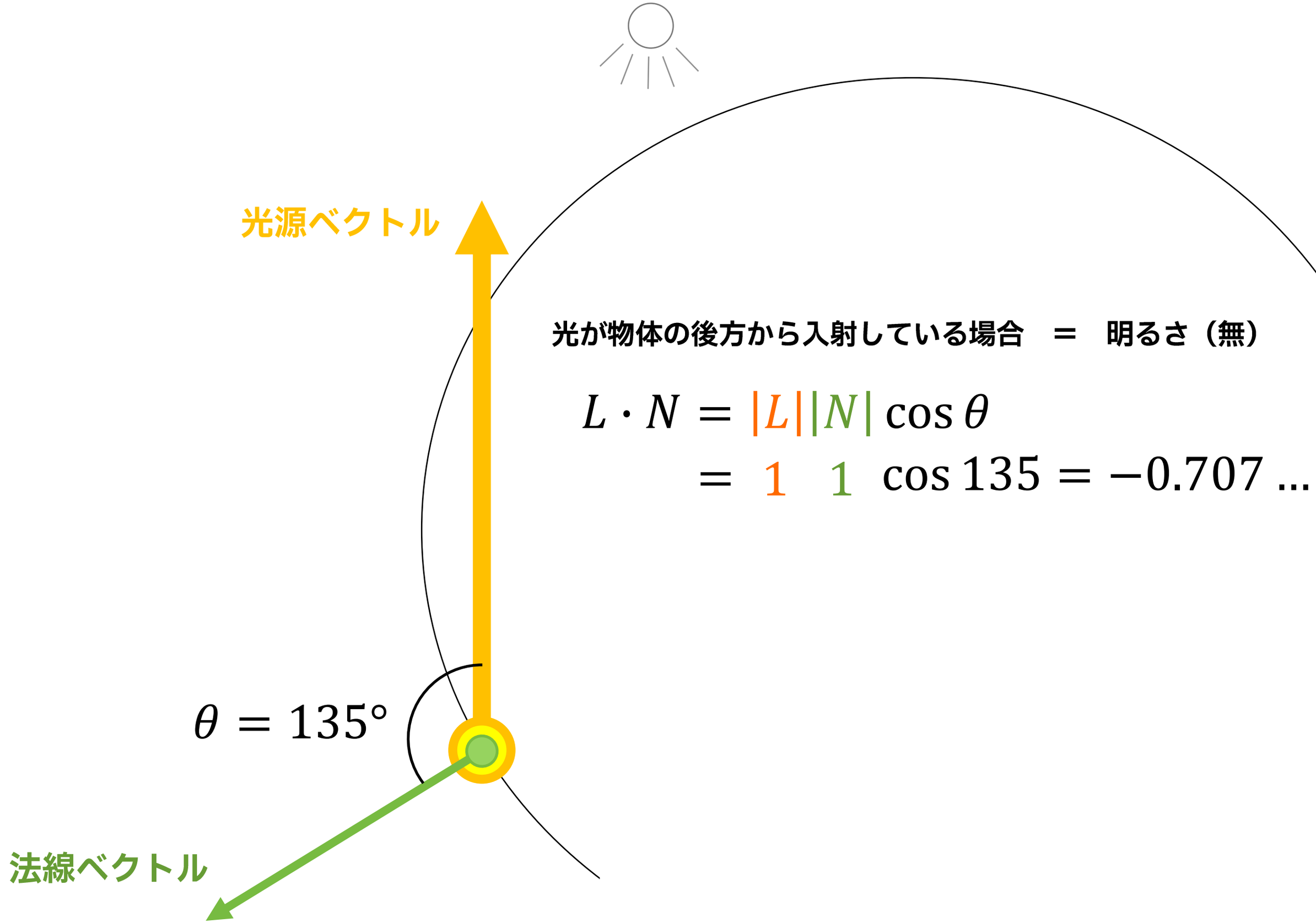
Lambert’s cosine law （ランバートの余弦則）の概説



光が物体真横から入射している場合　＝　明るさ（無）

$$L \cdot N = |L||N| \cos\theta$$
$$= 1 \quad 1 \quad \cos 90 = 0$$

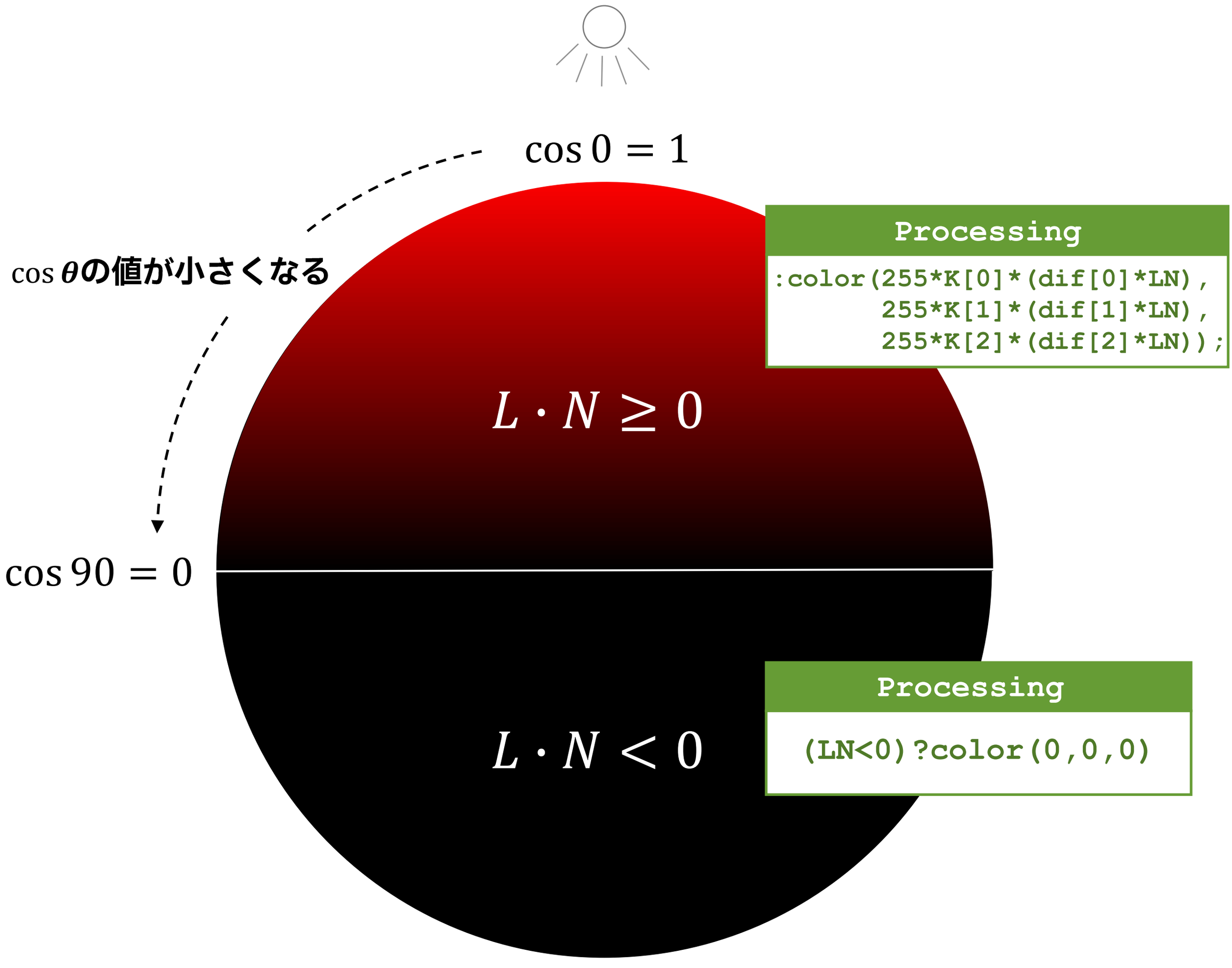
Lambert’s cosine law （ランバートの余弦則）の概説



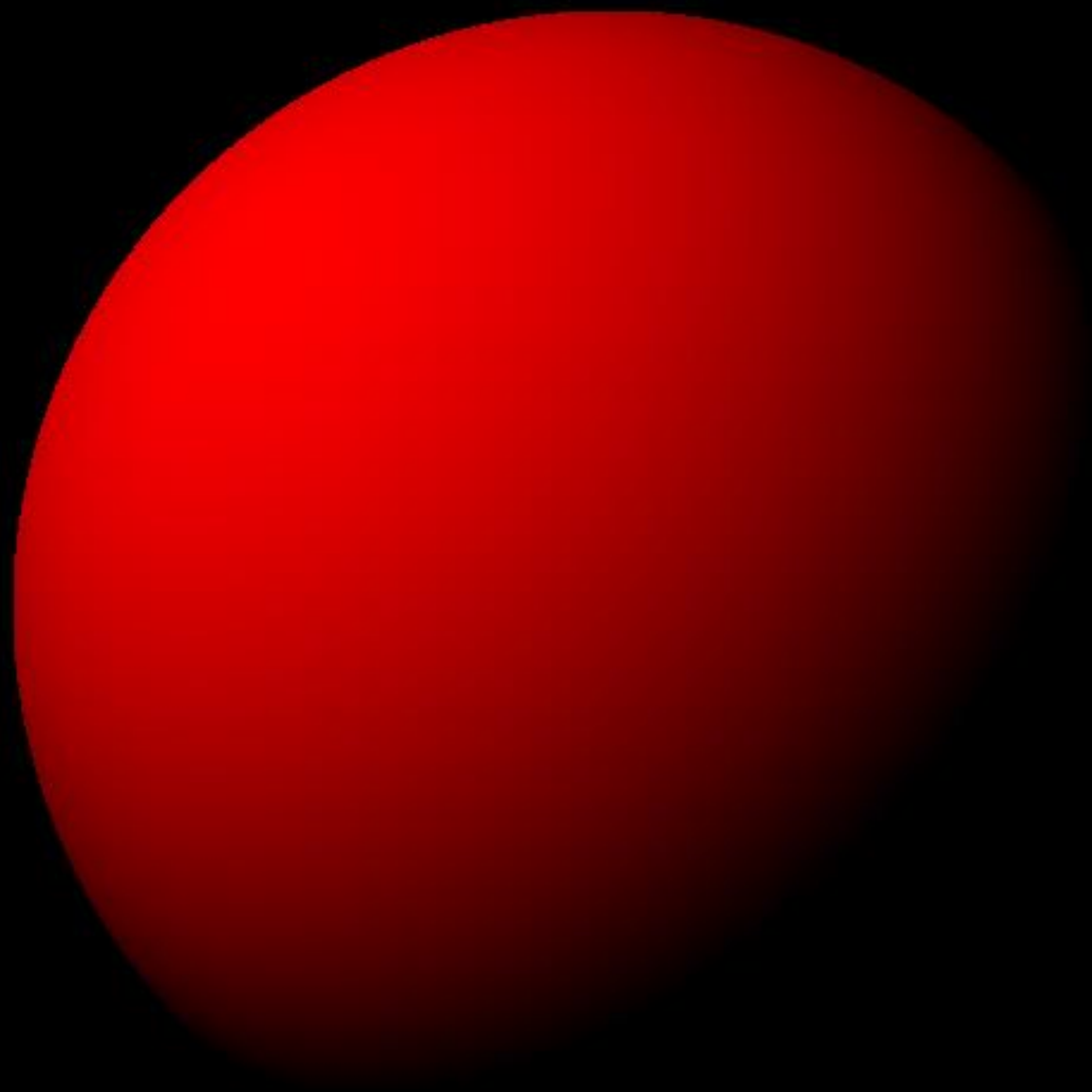
光が物体の後方から入射している場合　＝　明るさ（無）

$$L \cdot N = |L||N| \cos\theta$$

$$= 1 \quad 1 \quad \cos 135 = -0.707 \ldots$$


$\cos 0 = 1$
$\cos\theta$の値が小さくなる
$\cos 90 = 0$
$L \cdot N \geq 0$
$L \cdot N < 0$
Processing
:color(255*K[0]*(dif[0]*LN),
255*K[1]*(dif[1]*LN),
255*K[2]*(dif[2]*LN));
Processing
(LN<0)?color(0,0,0)

Lambert’s cosine law （ランバートの余弦則）の概説



sketch_12_2.zip をダウンロードして下さい

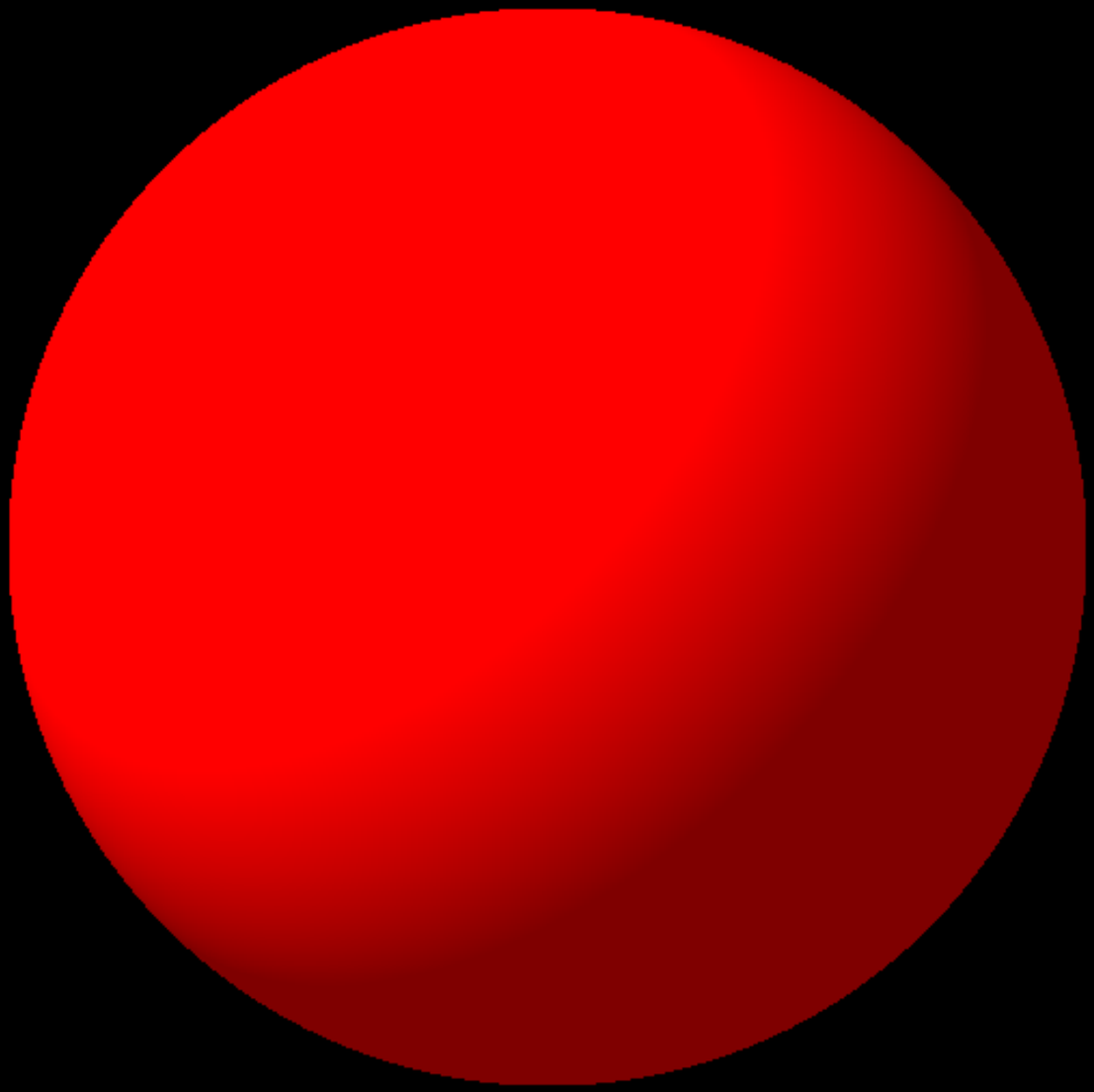
```
color Shading(float[] N) {
   float LN=innerProduct(L,N);
   return (LN<0)?color(0,0,0)  // 光源が反対側にある時
                :color(255*K[0]*(dif[0]*LN),
                       255*K[1]*(dif[1]*LN),
                       255*K[2]*(dif[2]*LN));
}
```

しかし、実際の世界ではたとえ光源が反対側にある場合でも、
環境光でやんわりと照らされています。
それを再現してみましょう。

```
color Shading(float[] N) {
   float LN=innerProduct(L,N);
   return (LN<0)?color(255*K[0]*amb[0], // 光源が反対側にある時
                       255*K[1]*amb[1], // にも、環境光が反映
                       255*K[2]*amb[2]) // されるようにします。
                :color(255*K[0]*(dif[0]*LN+amb[0]),
                       255*K[1]*(dif[1]*LN+amb[1]),
                       255*K[2]*(dif[2]*LN+amb[2]));
}
```

Lambert’s cosine law （ランバートの余弦則）の概説

環境光無し



環境光有り