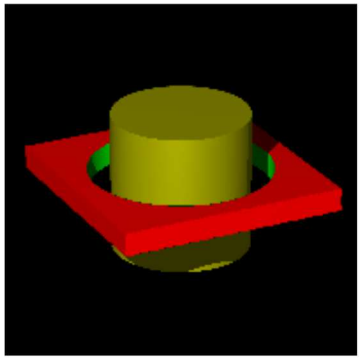
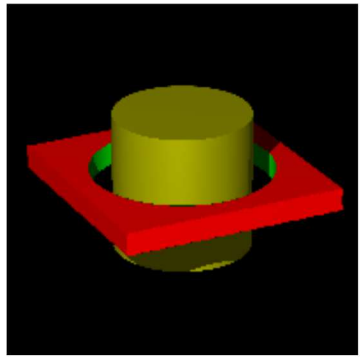
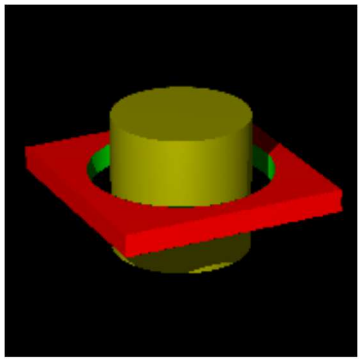
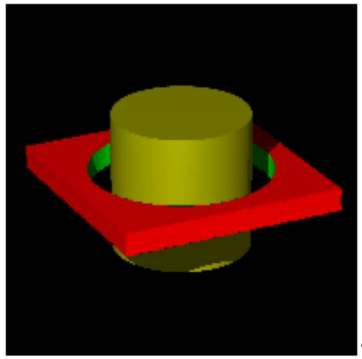
Cプログラミング III

簡易CSGモデル用の
ポリゴンレンダリングソフトの作成

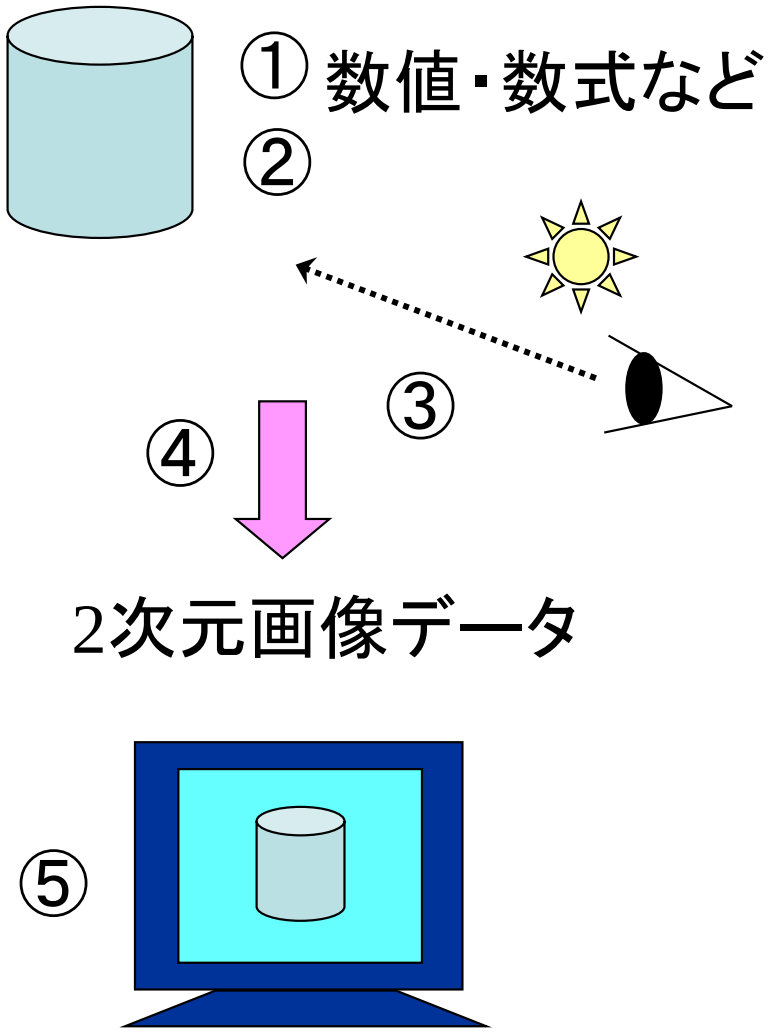



2章
CGの仕掛けを知る

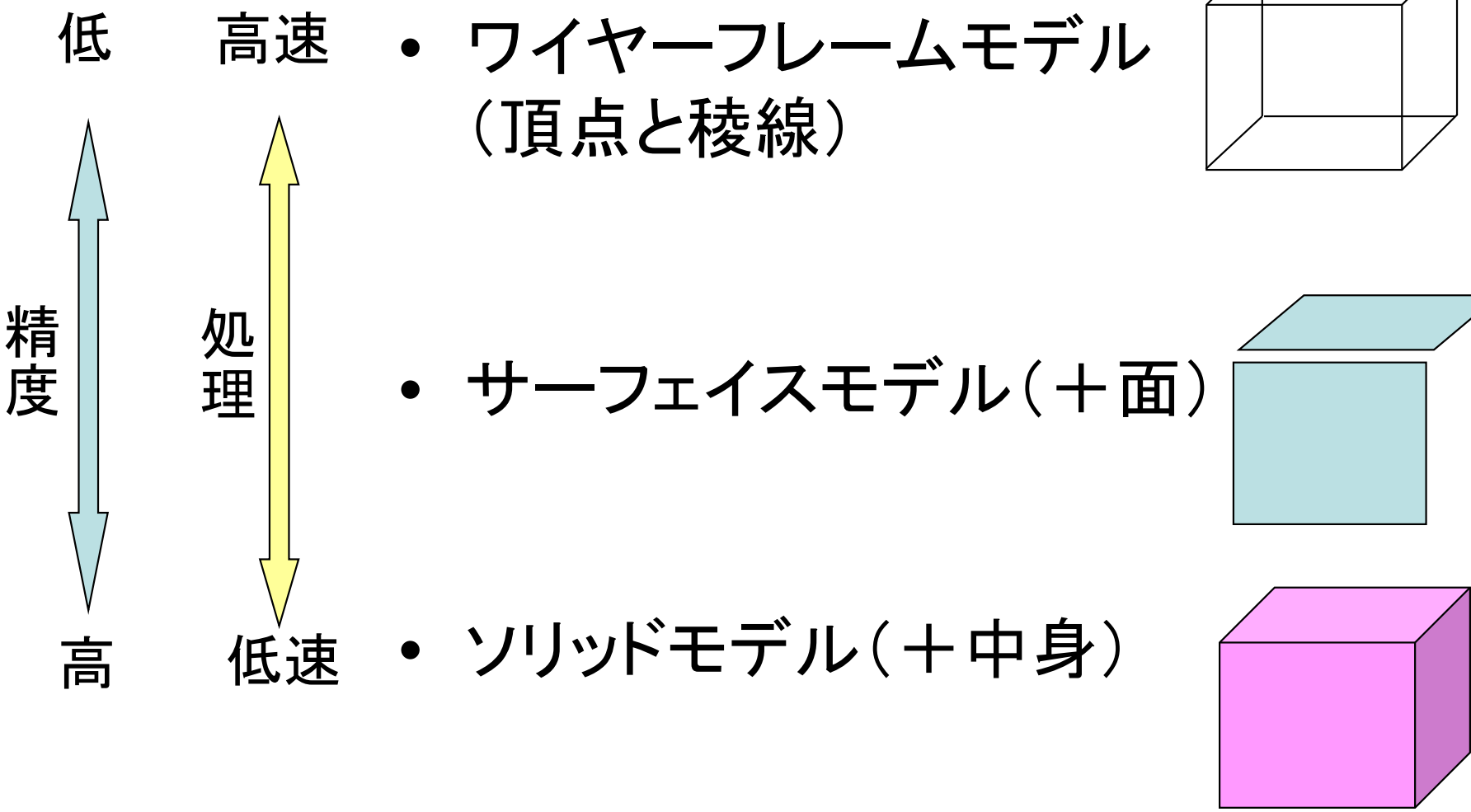



CGによる画像生成の流れ

①形状モデルの作成
　＜形状モデリング＞
②色・模様などの属性定義
③視点・視線（カメラ定義），
　光源の設定
　＜変換＞
④隠面消去とシェーディング
　＜レンダリング＞
⑤ディスプレイへの表示
　＜ディジタル画像表現＞



立体の形状モデル



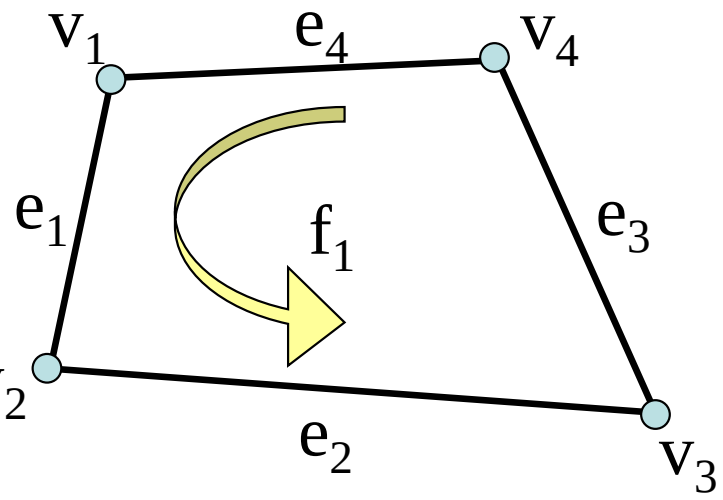
- ワイヤーフレームモデル（頂点と稜線）

- サーフェイスモデル（＋面）

- ソリッドモデル（＋中身）

ポリゴンモデル（サーフェイス）

- 頂点 $v_i=(x_i, y_i, z_i)$
- 辺（稜線）＝直線分 $e_k=(v_i, v_j)$
- 面（面分）＝平面（分）
  $f_i=(e_i, e_j, \ldots, e_n)=(v_i, v_j, \ldots, v_n)$



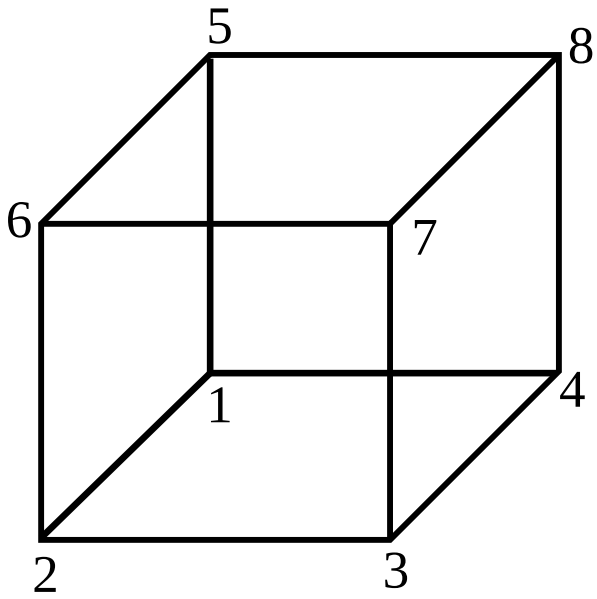
面の向き

- 頂点列を面の表側から見たとき反時計回りに統一する.
- 頂点の並びの順に右ねじを回したとき，ねじが進む方向が形状の外側.

表：$f=(v_1, v_2, v_3, v_4)$

裏：$f=(v_1, v_4, v_3, v_2)$

ポリゴンモデルの例



頂点の幾何情報

頂点	座標		
v1	x1	y1	z1
v2	x2	y2	z2
v3	x3	y3	z3
v4	x4	y4	z4
v5	x5	y5	z5
v6	x6	y6	z6
v7	x7	y7	z7
v8	x8	y8	z8

面分の位相情報
（＝幾何情報）

面分	頂点			
f1	v1	v5	v8	v4
f2	v2	v3	v7	v6
f3	v1	v2	v6	v5
f4	v4	v8	v7	v3
f5	v1	v4	v3	v2
f6	v5	v6	v7	v8

変換

CGでは様々な変換を用いる.
- 投影変換：3次元形状を2次元平面に投影する
- 幾何変換：3次元形状を拡大縮小・移動・回転する

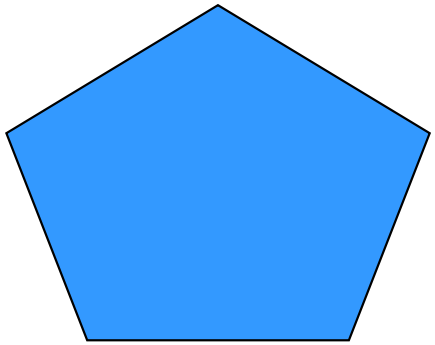
これらの変換は，おもに行列計算により実現される.

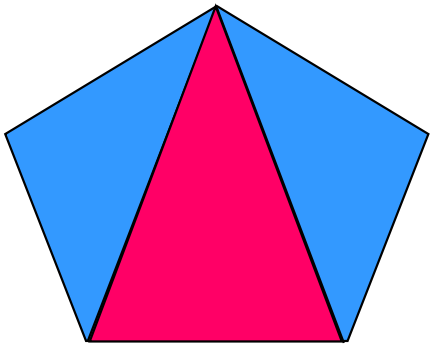
レンダリング

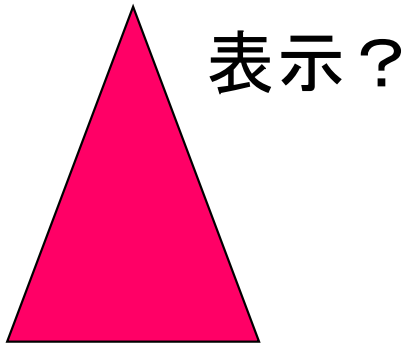
３次元形状モデルから2次元画像データを求める処理
- （隠れ面消去）
- シェーディング：陰影表現
- テクスチャマッピング：模様や凸凹
- レイトレーシング
- ボリュームレンダリング

３角形面を描く

- ポリゴンモデルは3角形面の集合に変換可能

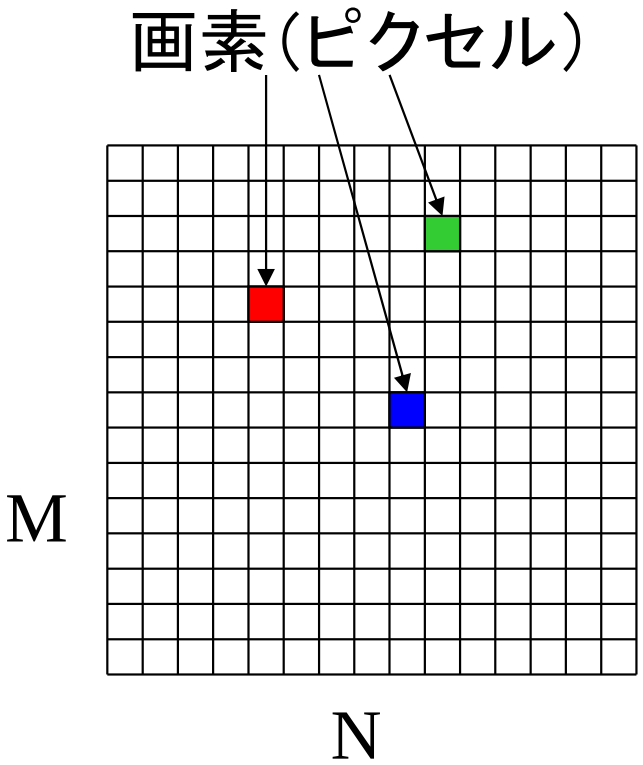






点を描く

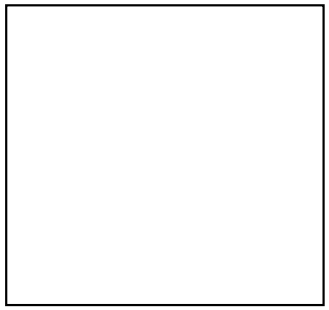
- カラーディスプレイ＝画像＝M×N個の画素(pixel)の集合（2次元配列）



赤　緑　青

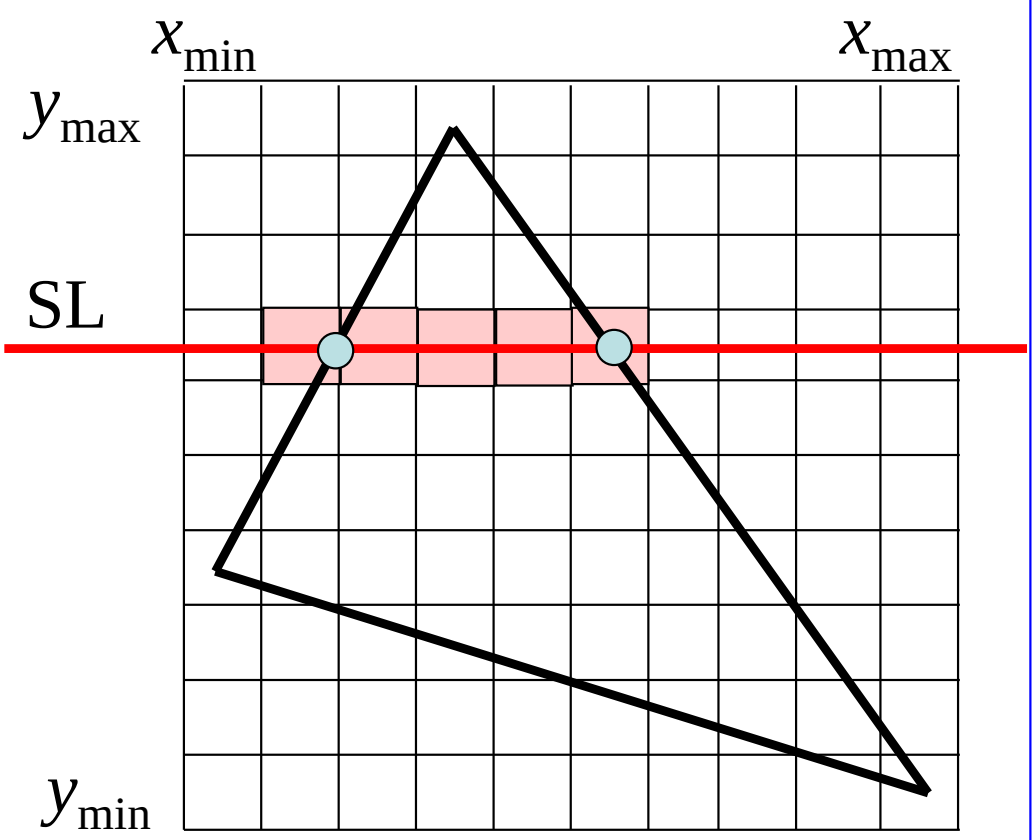
ピクセル＝ | R | G | B |

R,G,Bをそれぞれ1バイト（2^8=256:0～255）で表すと，2^{24}=約1600万色表示できる.



黒＝(0, 0, 0)
赤＝(255, 0, 0)
緑＝(0, 255, 0)
黄＝(255, 255, 0)
白＝(255, 255, 255)

３角形面を描く（２次元）



1. ３角形を囲む長方形
   $(x_{min},x_{max},y_{min},y_{max})$
2. Y座標をy_{max}～y_{min}まで，
   １ずつ減らしながら以下を
   繰り返す
   (a)スキャンラインSL($Y=y$)発生
   (b)SLと３角形との交点
   （線形補間）
   (c)交点間の画素を
   表示色(r,g,b)
   で塗る

＊端点や水平線に注意

3章
3次元形状をモデリングする

平面の式

$$ax+by+cz+d=0$$

球面の式

$$(x-a)^2+(y-b)^2+(z-c)^2-r^2=0$$

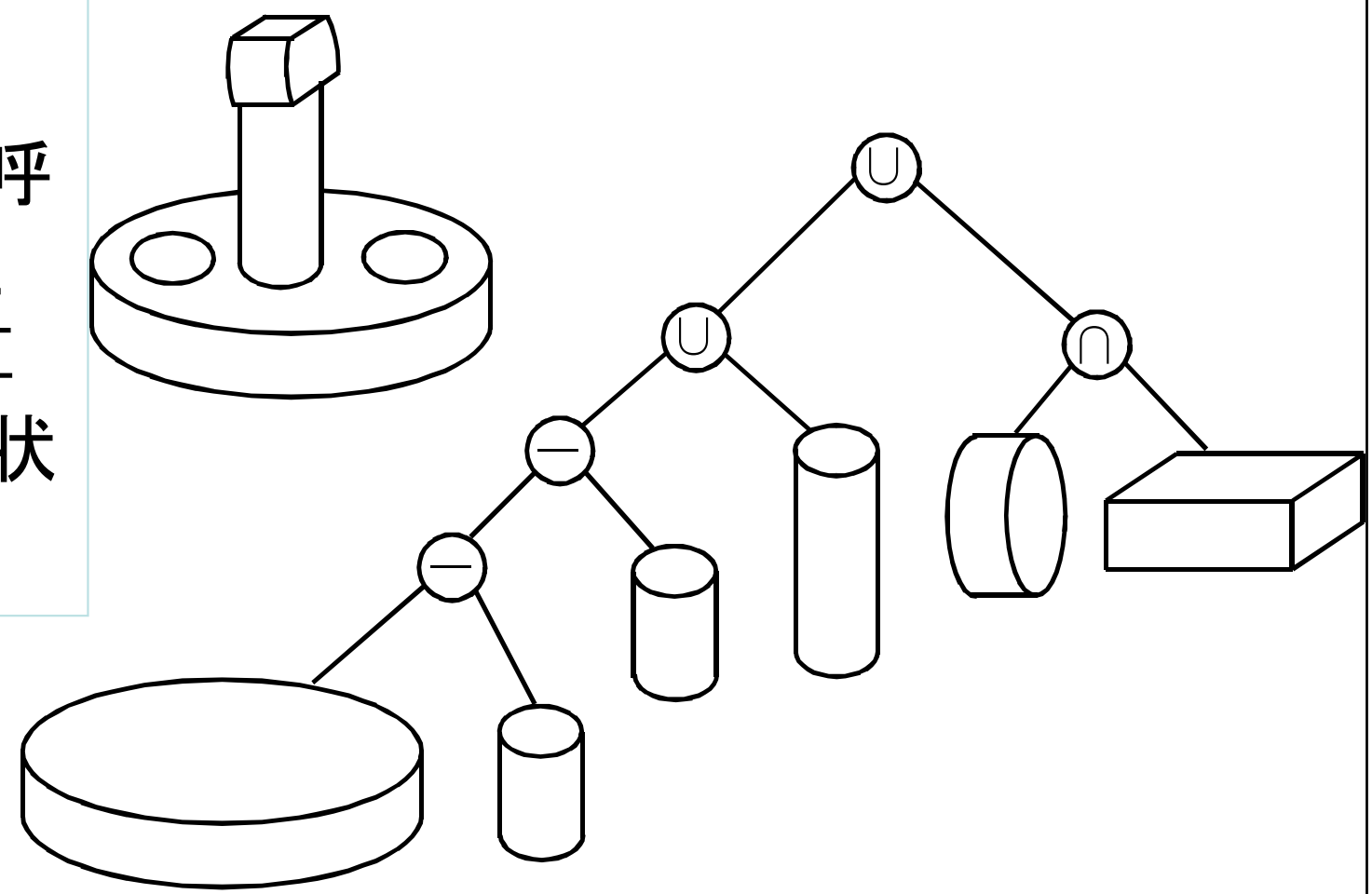
一般の2次曲面の式

$$ax^2+by^2+cz^2+dxy+eyz+fzx+px+qy+rz+s=0$$

球面，楕円体面，円柱面，楕円柱面，円錐面，惰円錐面，
放物面，双曲面など

CSG（集合演算）

- Constructive Solid Geometry
- プリミティブと呼ばれる基本立体の集合演算により立体形状を表現する.



立体形状＝3次元空間内の点の集合

陰関数表現：$f(x,y,z)$

$f(x,y,z)=0$　・・・曲面表面上の点の集合

$f(x,y,z)\le 0$　・・・曲面$f(x,y,z)=0$を境界とする領域（半空間）

プリミティブ$P=\bigcap_{i=1}^{n}(f_i(x,y,z)\le 0)$　・・・半空間の積集合
（空間的に閉じている）

プリミティブの例：円柱プリミティブ

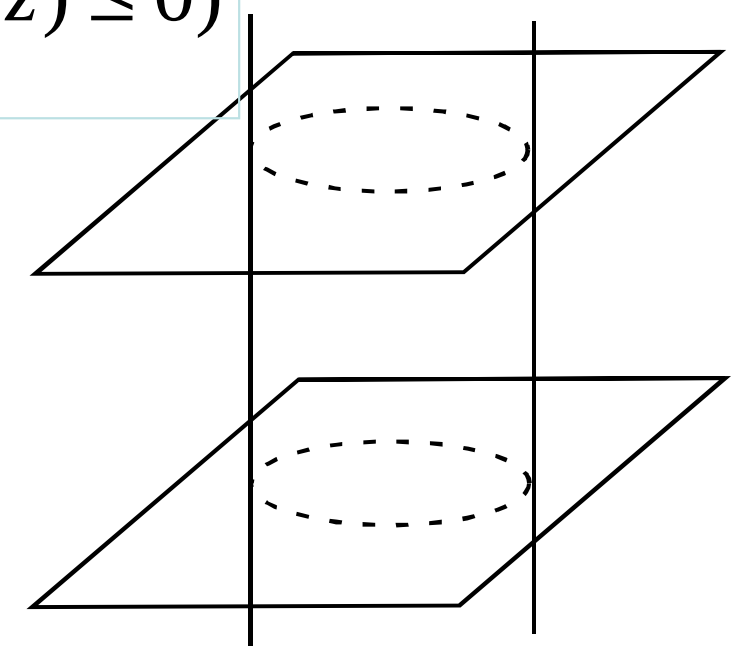
$$f_1(x,y,z) = x^2 + y^2 - r^2$$

$$f_2(x,y,z) = z - h$$

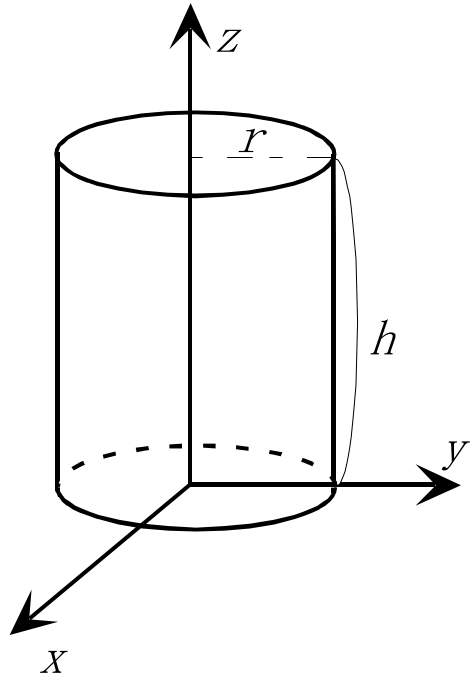
$$f_3(x,y,z) = -z$$

$$P_{cyl} = \bigcap_{i=1}^{3} (f_i(x,y,z) \le 0)$$

半径r，高さh



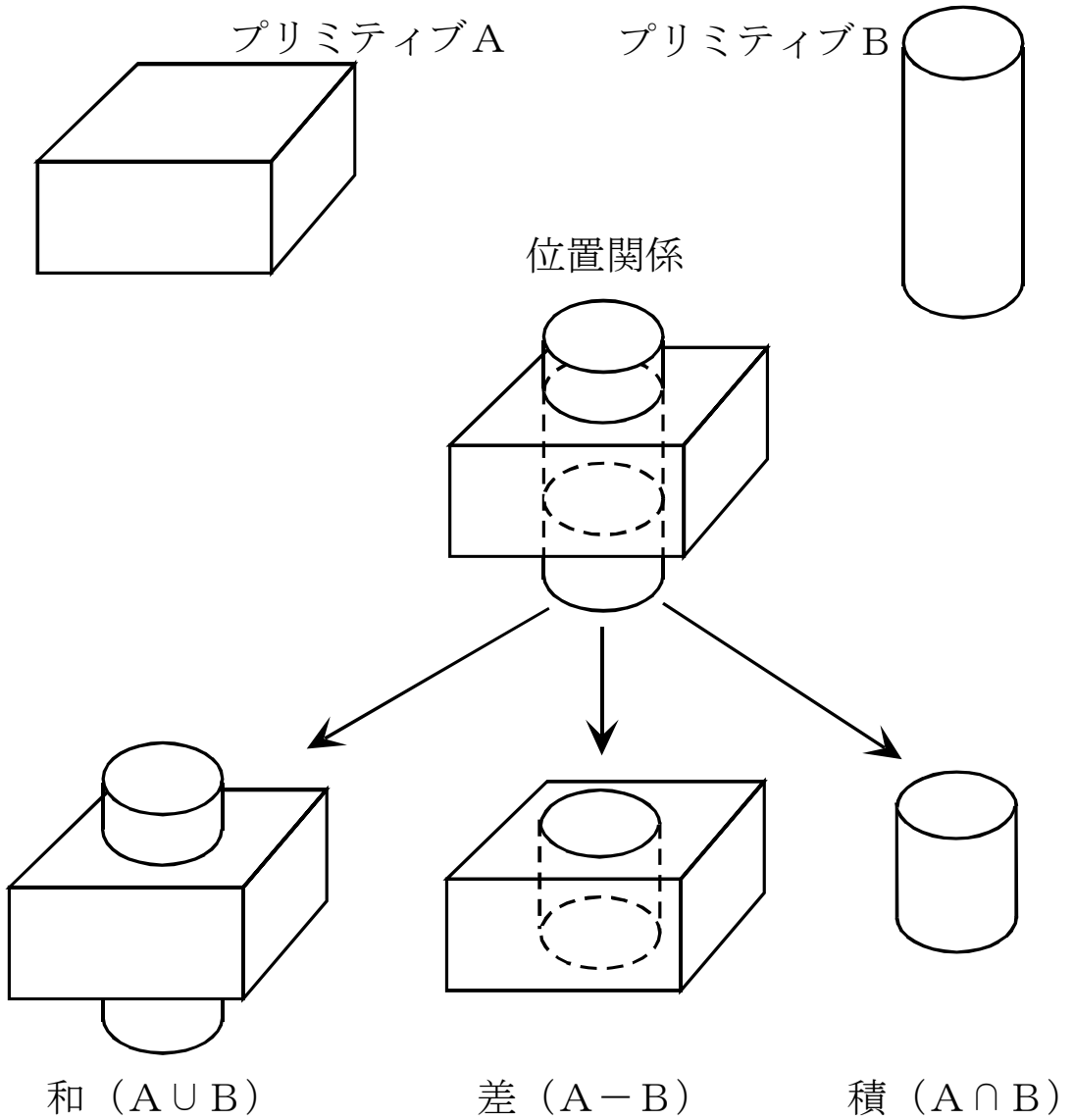
３つの半空間



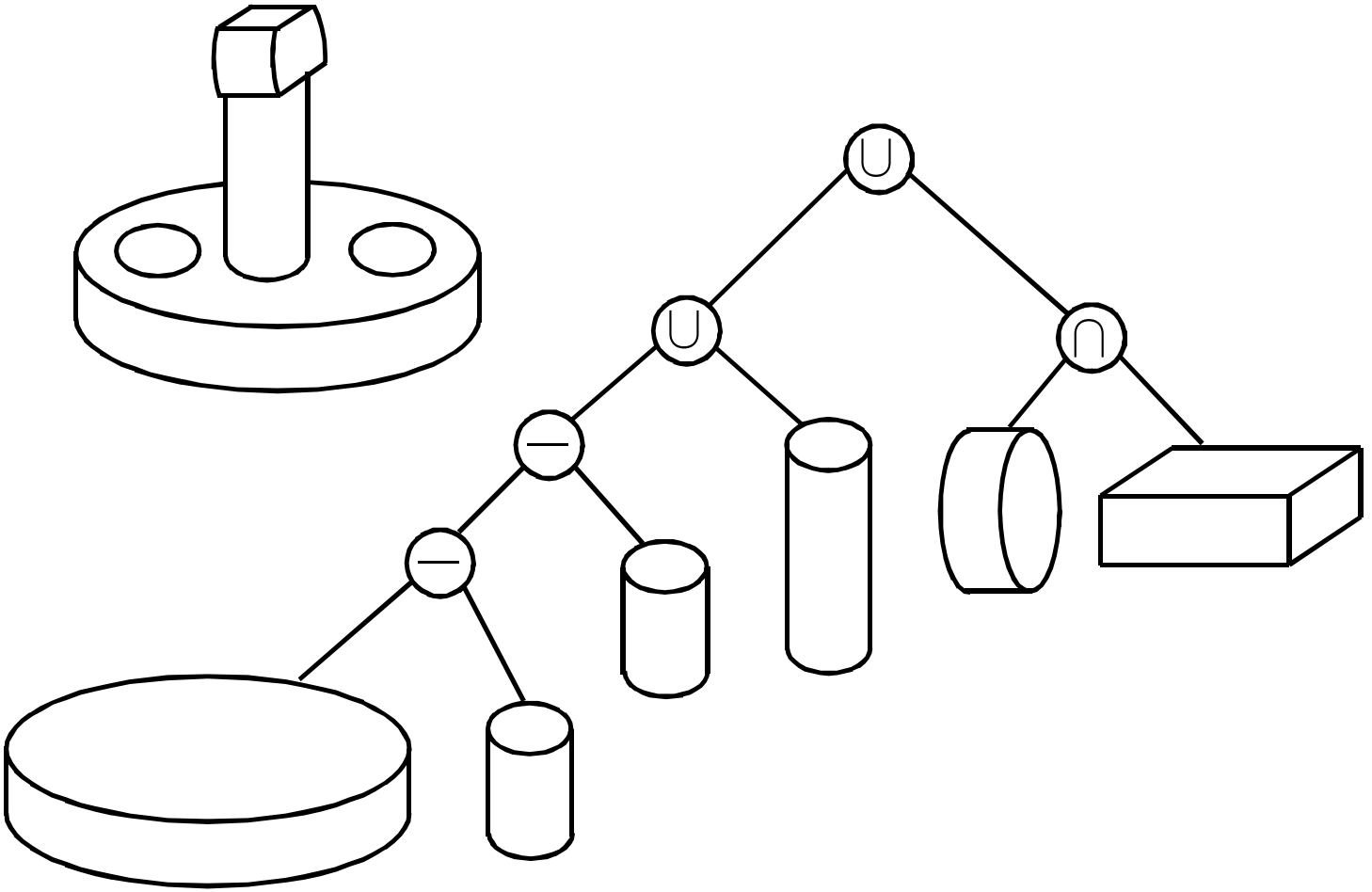
円柱プリミティブ

集合演算

和集合＋(∪)
差集合ー
積集合＊(∩)



複雑な立体形状
＝集合演算を階層的に繰り返す
＝CSGトリー構造（木構造）



第５回
投影変換と幾何変換を行う

５章　３次元形状を2次元面に投影する

６章　３次元形状を変形・移動させる

同次座標

通常座標$(x, y, z) \Leftrightarrow$同次座標$(X, Y, Z, W)$

$$x = X / W$$

$$y = Y / W$$

$$z = Z / W$$

同次座標$(X, Y, Z, 0)$は,(X, Y, Z)方向の無限遠点

同次座標を用いることで，様々な変換を統一的に扱える.

４×４の行列の積

射影変換

射影変換 $$\begin{bmatrix} W' & X' & Y' & Z' \end{bmatrix} = \begin{bmatrix} W & X & Y & Z \end{bmatrix} A$$

または

$$\begin{bmatrix} X' & Y' & Z' & W' \end{bmatrix} = \begin{bmatrix} X & Y & Z & W \end{bmatrix} A$$

アフィン変換 $$\begin{bmatrix} 1 & x' & y' & z' \end{bmatrix} = \begin{bmatrix} 1 & x & y & z \end{bmatrix} A$$

または

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} A$$

Aは４×４の行列

幾何変換

- 物体の移動，拡大/縮小，回転は，物体の頂点座標*P*に変換行列を掛けることで実現する.
- 変換行列は，４×4行列（同次座標系）を用いる.
- 同次座標系では，頂点座標*P*(*x*,*y*,*z*)を*P*(*X*,*Y*,*Z*,*W*)で表現し，実際の座標は，
  $x=X/W$，$y=Y/W$，$z=Z/W$
  で計算する.
- 同次座標系を用いることにより，すべての変換（幾何変換・投影変換）を行列の積の形で表現できる.

平行移動(点Pを平行移動TによりP'に変換)

$$P' = P + t$$

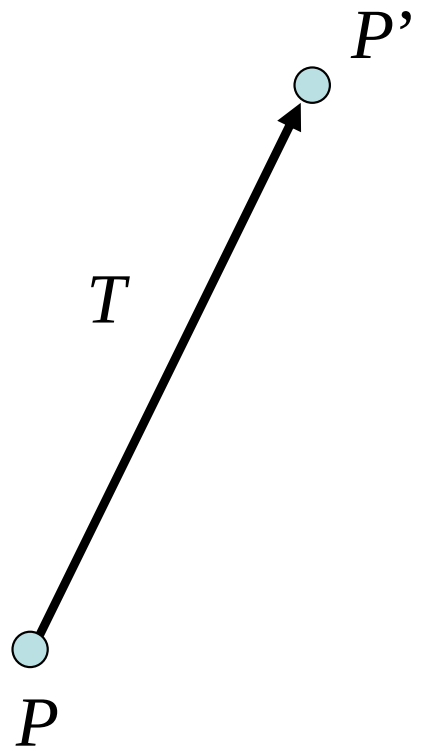
$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} + \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}$$

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x + t_x & y + t_y & z + t_z & 1 \end{bmatrix}$$

$$= \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= \begin{bmatrix} x & y & z & 1 \end{bmatrix} T$$

$$P' = PT$$



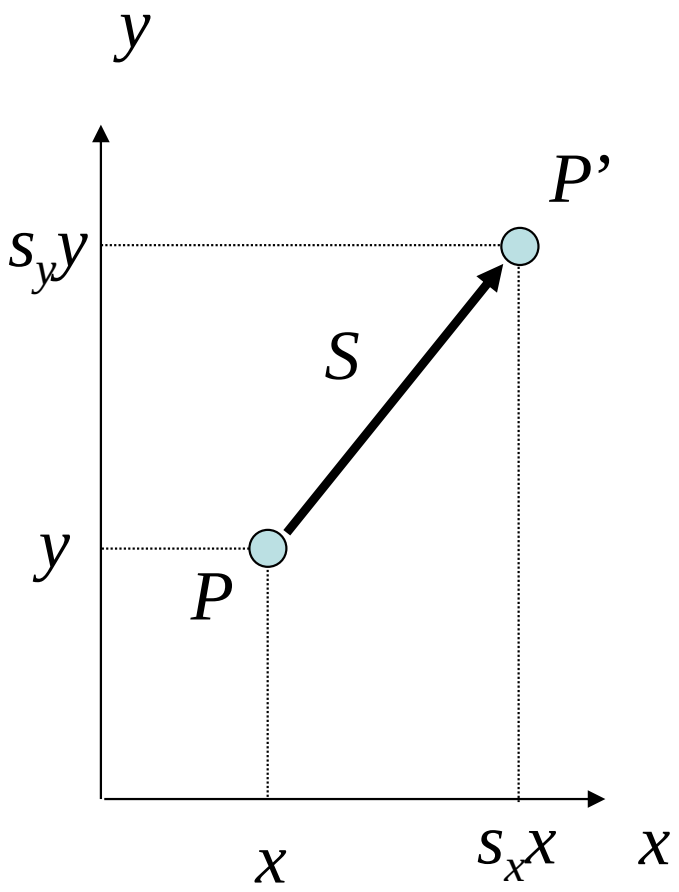
拡大/縮小（点Pを拡大/縮小SによりP'に変換）

$$\begin{cases} x' = s_x x \\ y' = s_y y \\ z' = s_z z \end{cases}$$

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix}$$

$$= \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

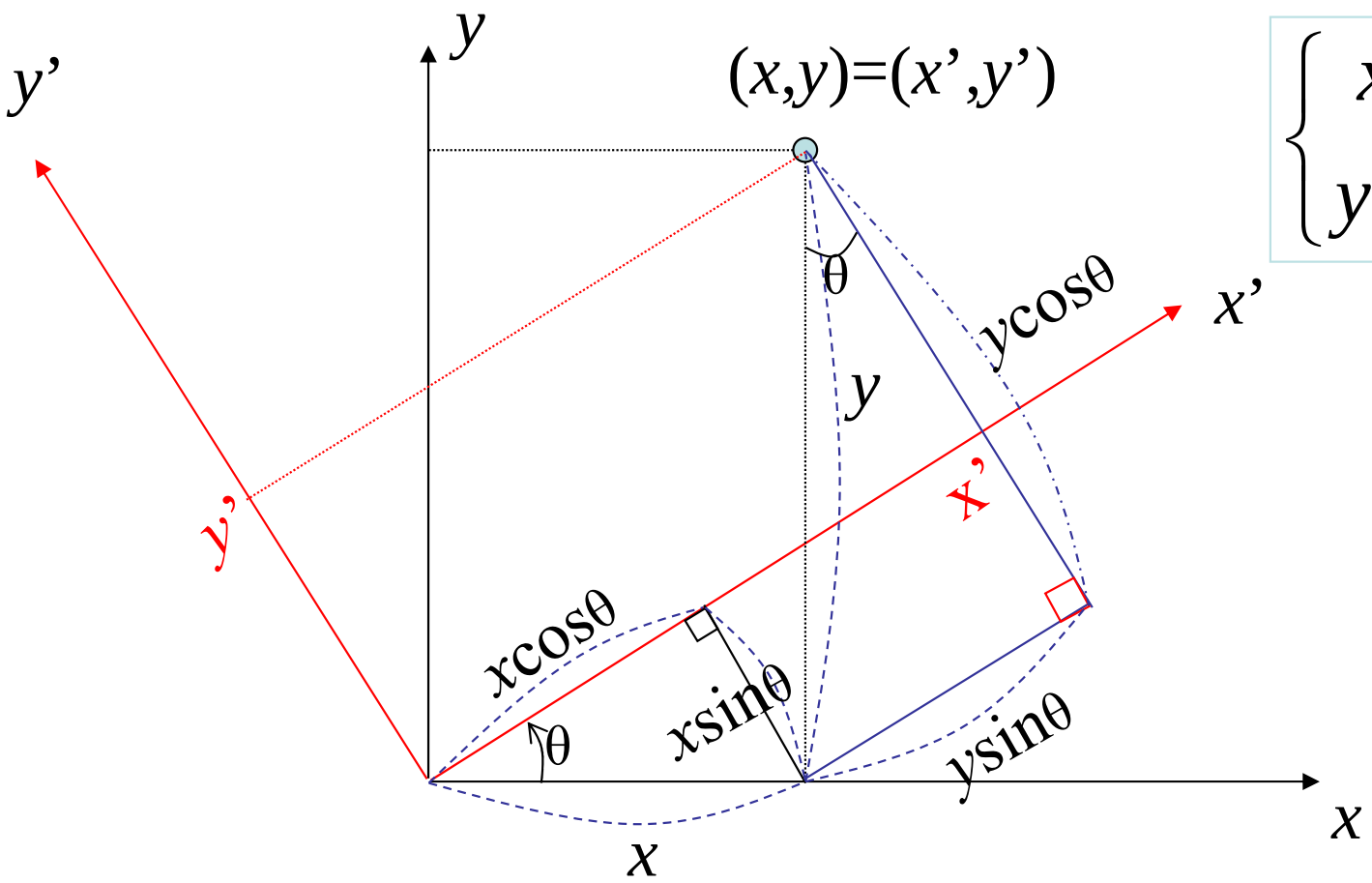
$$P' = PS$$



回転（点Pをz軸回りの回転R_zにより点P'に変換）

2次元（x-y空間）での回転は，

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

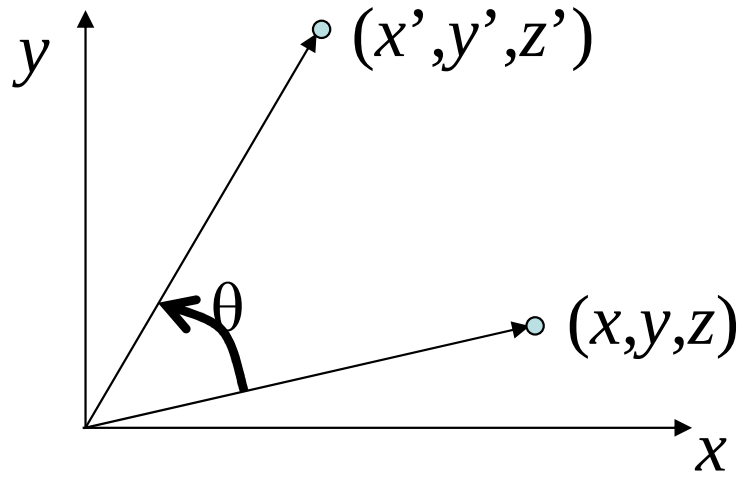


$$\begin{cases} x' = x\cos\theta + y\sin\theta \\ y' = -x\sin\theta + y\cos\theta \end{cases}$$

これは座標系の回転なので，点の回転は，
$\theta \leftarrow -\theta$
と置き換える．

2次元の回転にzを加える.

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \\ z' = z \end{cases}$$



すると，3次元空間での z軸回りの回転に相当する.
同次座標系になおすと,

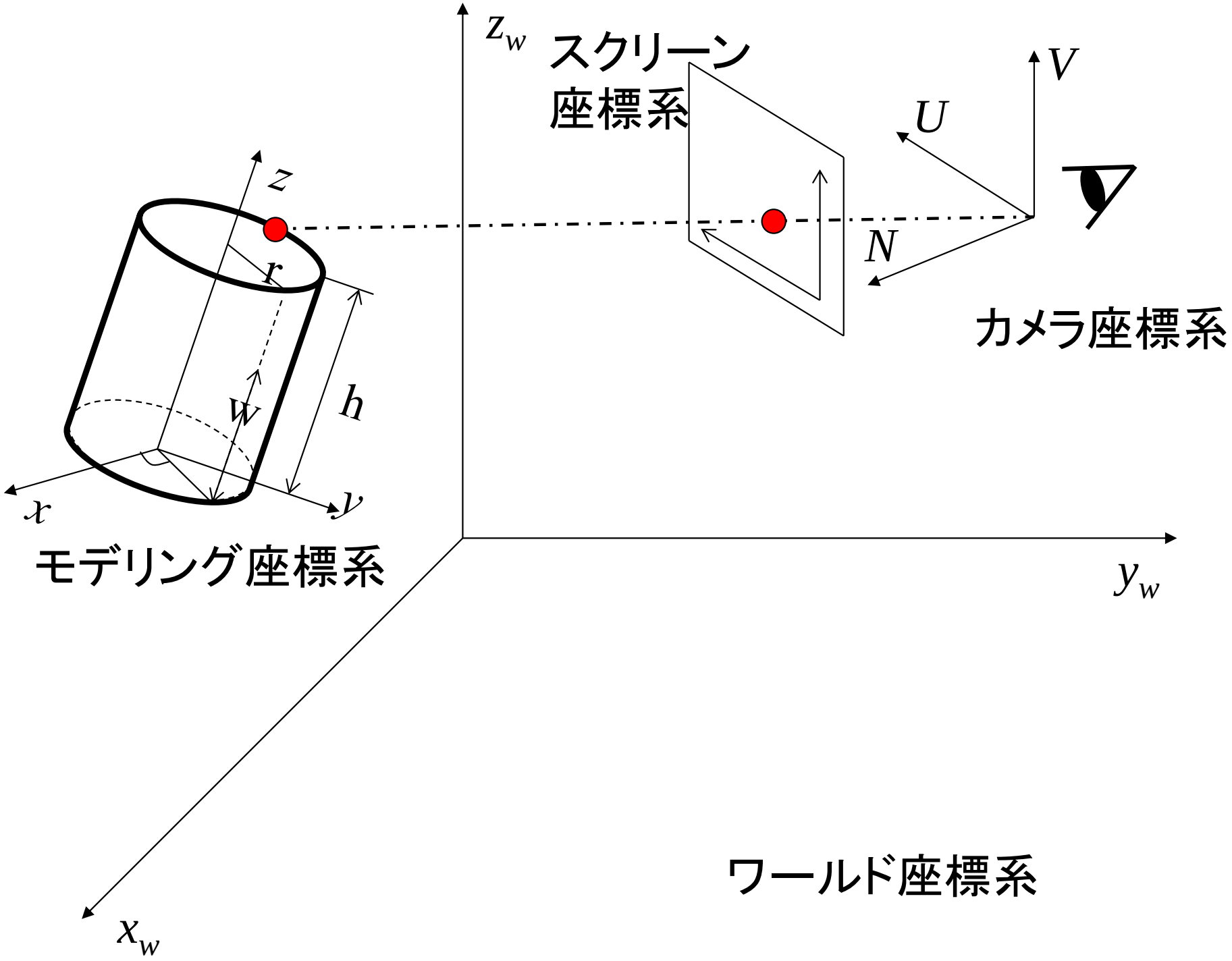
$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = PR_z$$

同様に，x軸回りの回転R_x，および，y軸回りの回転R_yは，

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$


z_w
スクリーン
座標系
V
U
N
カメラ座標系
z
r
w
h
x
y
モデリング座標系
y_w
x_w
ワールド座標系

投影変換

ここでは，一般の場合，すなわち，
「視点CからベクトルN方向に見る場合」
の透視投影について述べる.

カメラ座標系

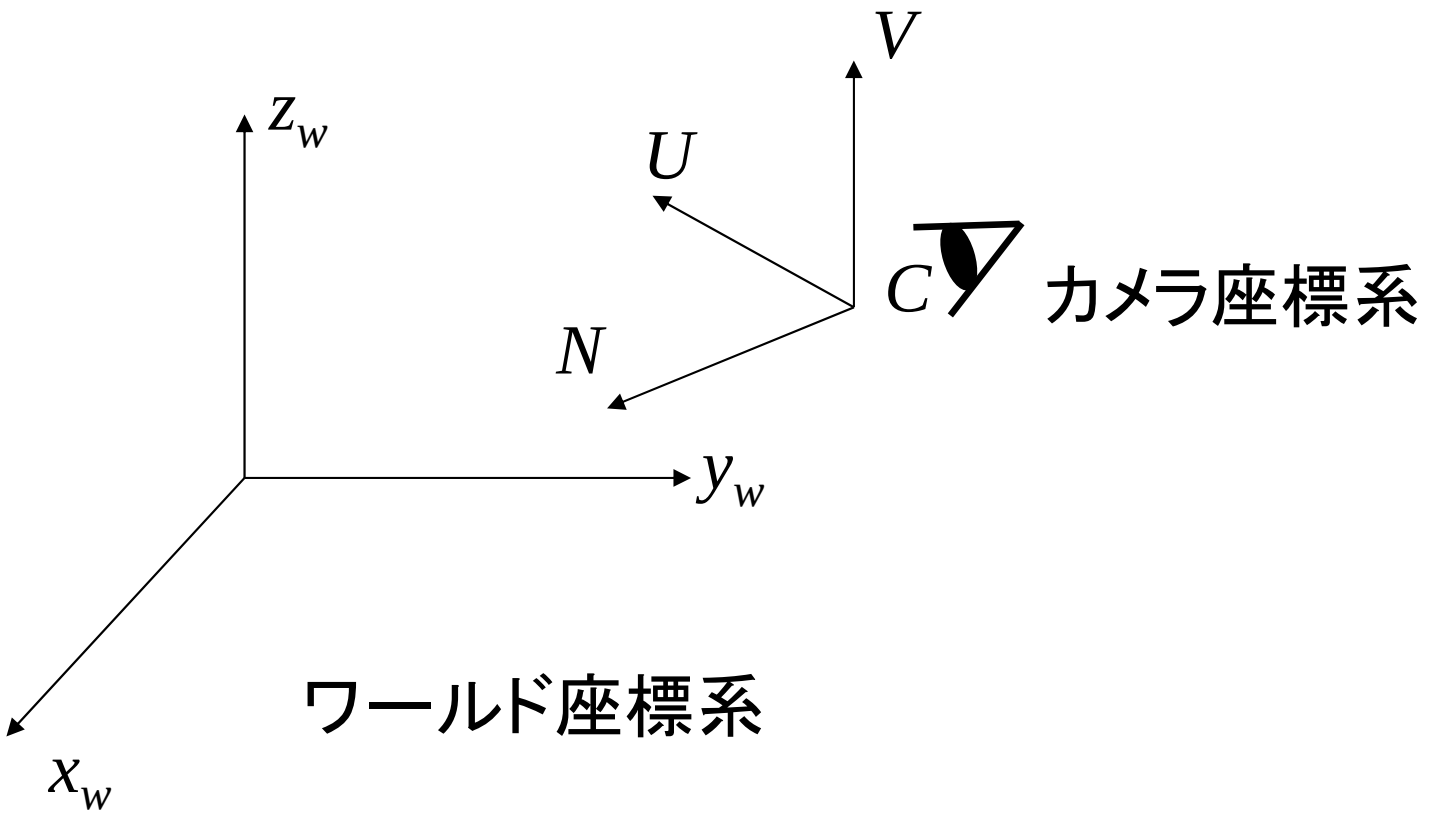
ワールド座標系の点$P_w(x_w,y_w,z_w)$をカメラ座標系の点$P_c(x_c,y_c,z_c)$へ変換する変換行列をAとすれば,

$$\text{（カメラ座標）} \overset{A}{\leftarrow} \text{（ワールド座標）}$$

$$\begin{bmatrix} x_c & y_c & z_c & 1 \end{bmatrix} = \begin{bmatrix} x_w & y_w & z_w & 1 \end{bmatrix} A$$

ワールド座標系では，
カメラ座標系の原点$C(C_x,C_y,C_z)$
視線方向の単位ベクトル$N(N_x,N_y,N_z)$
Nに垂直で互いに垂直な2つの単位ベクトル
$U(U_x,U_y,U_z)$：スクリーン上のx軸に相当
$V(V_x,V_y,V_z)$：スクリーン上のy軸に相当

カメラ座標系では，
$C \rightarrow (0,0,0)$
$N \rightarrow (0,0,1)$
$U \rightarrow (1,0,0)$
$V \rightarrow (0,1,0)$



変換行列Aを求める

$$A = TR$$

T. . . 平行移動

R. . . 回転

Tは(C_x, C_y, C_z)を$(0,0,0)$に平行移動する変換なので,

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -C_x & -C_y & -C_z & 1 \end{bmatrix}$$

行列Rを求める

行列Rが満たすべき条件式

$$\begin{bmatrix} U_x & U_y & U_z & 1 \end{bmatrix} R = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} V_x & V_y & V_z & 1 \end{bmatrix} R = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} N_x & N_y & N_z & 1 \end{bmatrix} R = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$$

$$U_x^2 + U_y^2 + U_z^2 = 1$$
$$V_x^2 + V_y^2 + V_z^2 = 1$$
$$N_x^2 + N_y^2 + N_z^2 = 1$$

大きさ1
（単位ベクトル）

$$U_x V_x + U_y V_y + U_z V_z = 0$$
$$V_x N_x + V_y N_y + V_z N_z = 0$$
$$N_x U_x + N_y U_y + N_z U_z = 0$$

内積0
（直交）

$$R = \begin{bmatrix} U_x & V_x & N_x & 0 \\ U_y & V_y & N_y & 0 \\ U_z & V_z & N_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ベクトルU,V,Nの決め方

N:視線方向の単位ベクトル

視点Cから点Dを見ることにすれば,

$N=(D-C)/|D-C|$

V:画面の上方向(y軸)に相当

適当なV'を指定し,

($V'=(0,0,1)$たとえば)

$V' \leftarrow V'/|V'|$

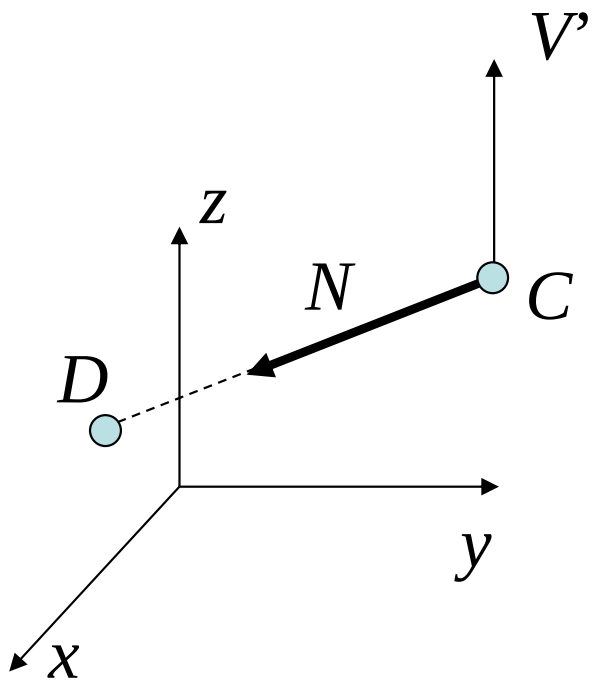
$V=V'-(V' \cdot N)N$

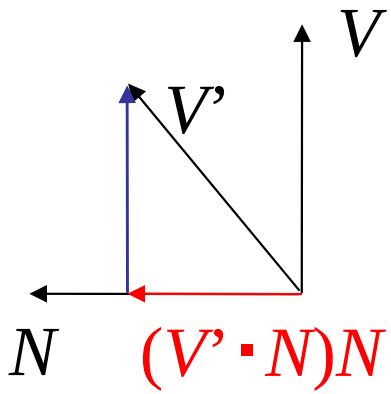
$V \leftarrow V/|V|$

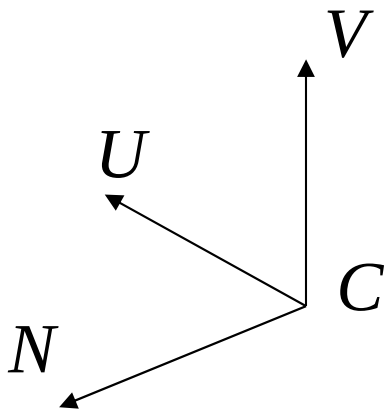
U:画面の右方向(x軸)に相当

$U=N \times V$

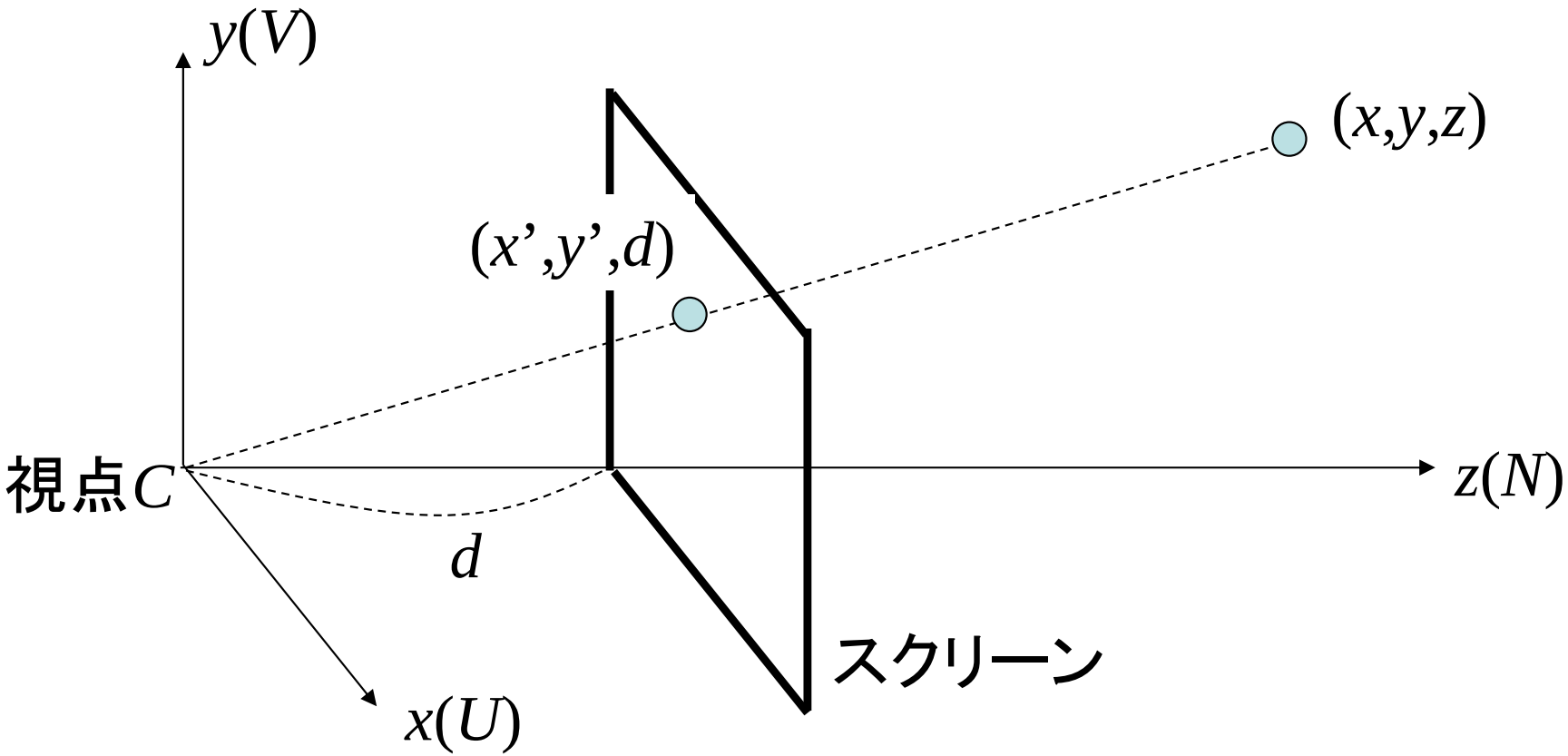
左手座標系







スクリーン座標系



カメラ座標系において透視投影を行う

透視投影 $x'=(d/z)x,\ y'=(d/z)y,\ z'=d$ ただし, $z>0$

透視投影 $x'=(d/z)x,\ y'=(d/z)y,\ z'=d$ ただし, $z>0$

この透視投影を射影変換で表すと以下のようになる.

$$\begin{cases} x' = xd/z \\ y' = yd/z \\ z' = d \end{cases}$$

$$[x' \quad y' \quad z' \quad W']$$

$$= [x \quad y \quad z \quad 1] \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P' = PB$$

ワールド座標系で定義されている点Pをスクリーン上の点に変換するまでの流れ

視点C，見る点D，視点からスクリーンまでの
距離d，および，アップベクトルV'を指定する．

1．カメラ座標系U-V-Nを求める
2．変換行列$A=TR$を求める
3．ワールド座標系の点PをAでカメラ座標に変換する
4．透視投影し，スクリーン上での(x', y')座標を得る

７章
隠れ面の消去を行う

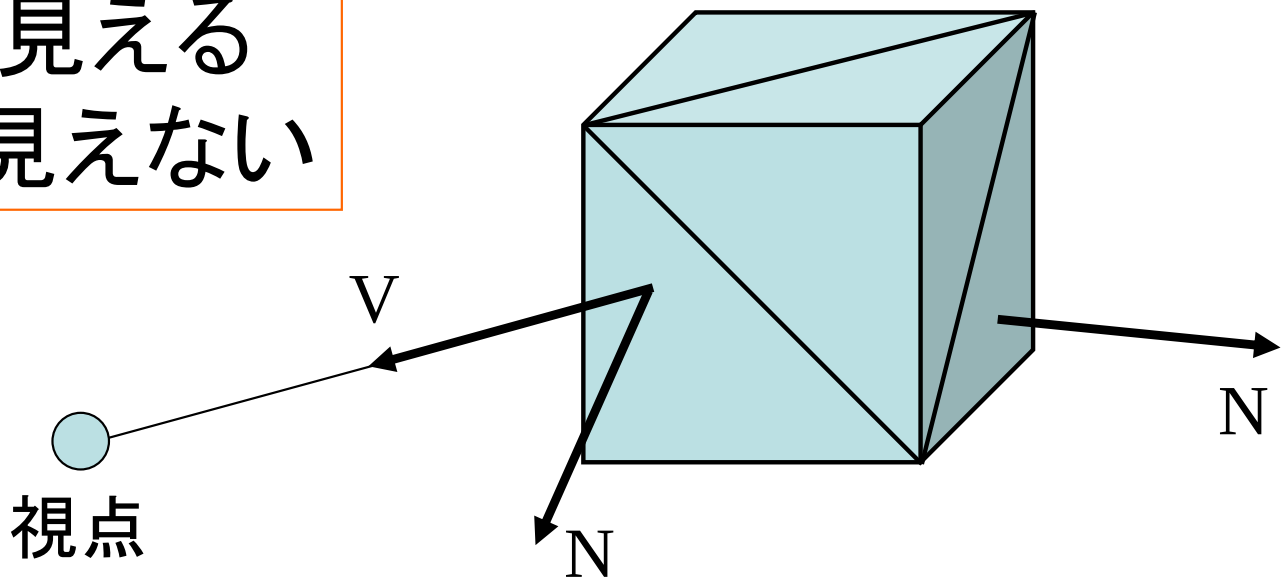
Z−バッファ法

スキャンライン法

後面除去

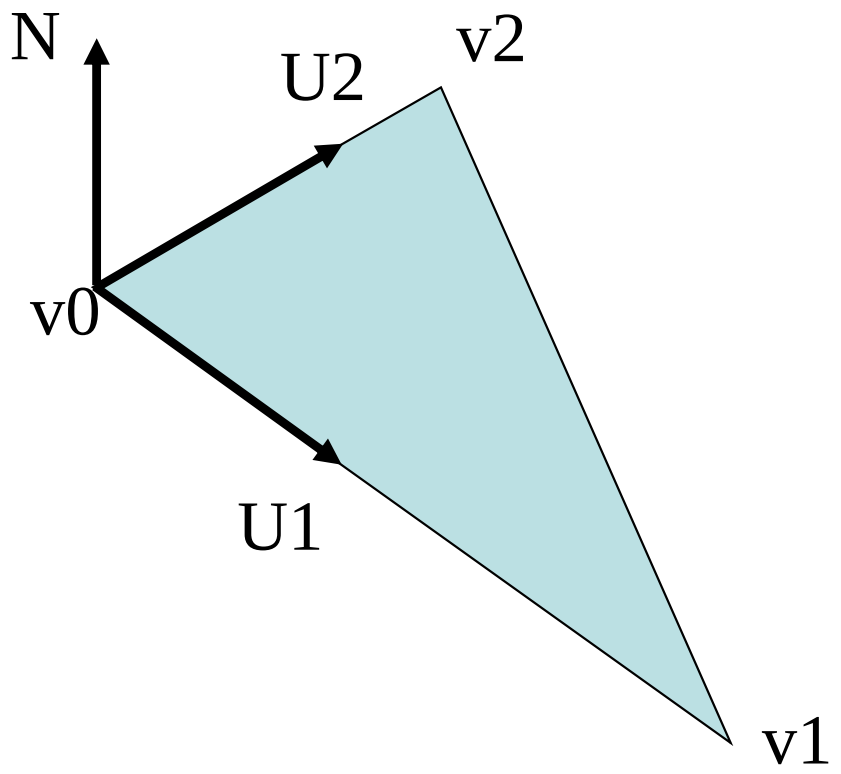
- 凸多面体1個に対してのみ有効
- 面の法線ベクトルと視線ベクトルのみで判定

N･V ≧ 0 なら見える
N･V ＜ 0なら見えない



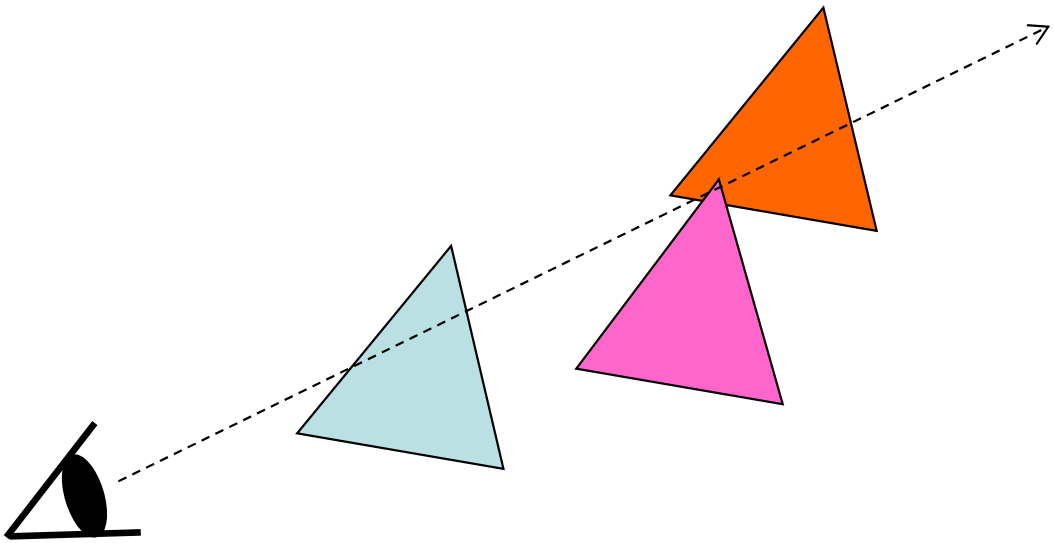
３角形面の法線ベクトル

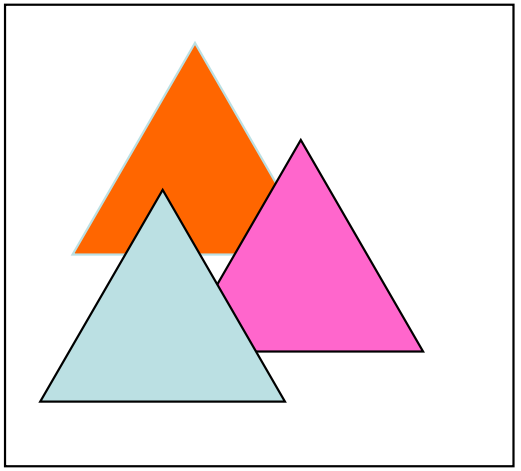
３角形面f = (v0, v1, v2)
 に対して,
ベクトルU1=v1 − v0
ベクトルU2=v2 − v0
とすると,
法線ベクトルNは,
N = U1 × U2



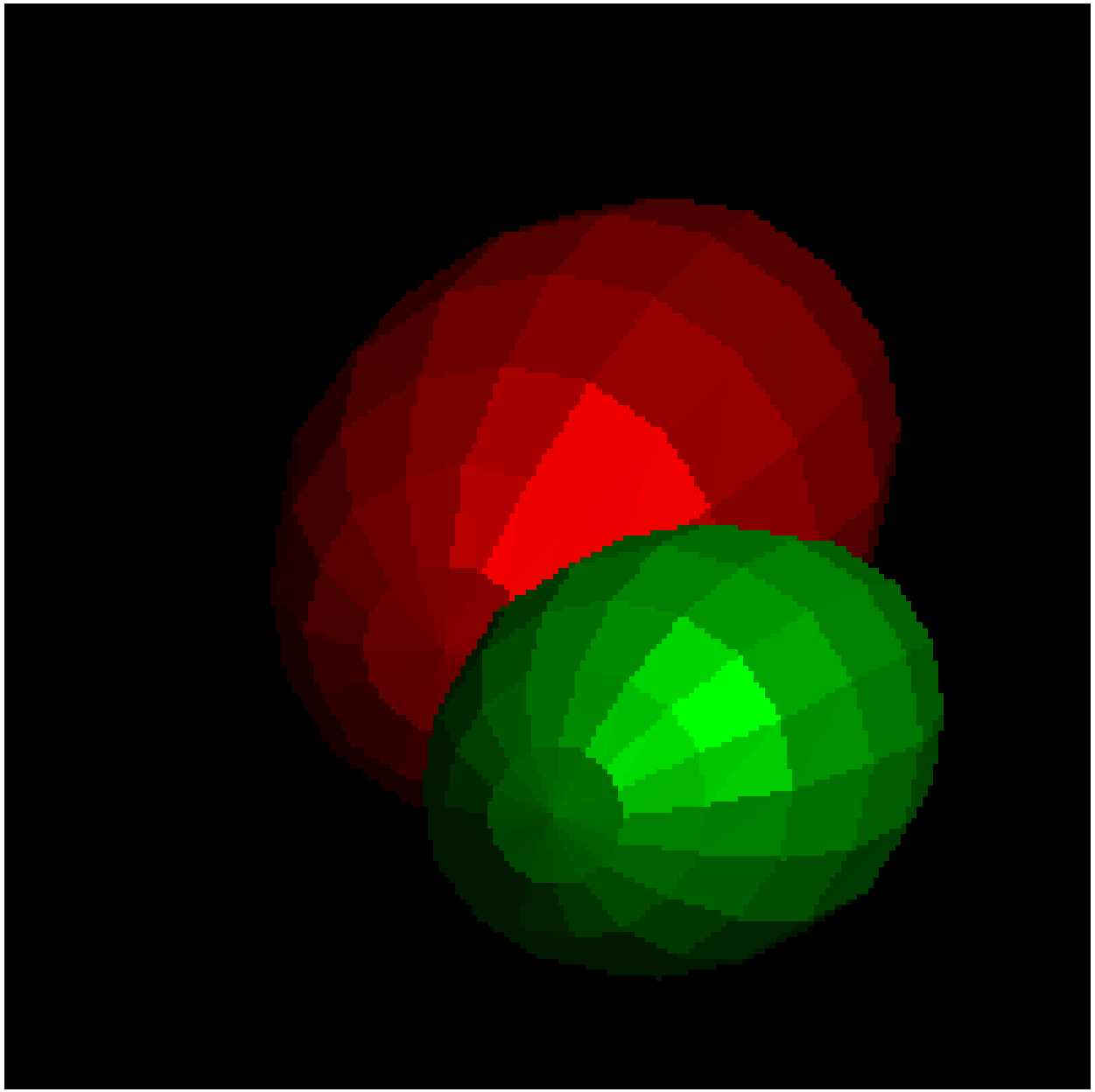
ペインタアルゴリズム（奥行き優先度法）

- 視点から見て，奥（遠く）にある面から順に塗る.
- 失敗することがある.

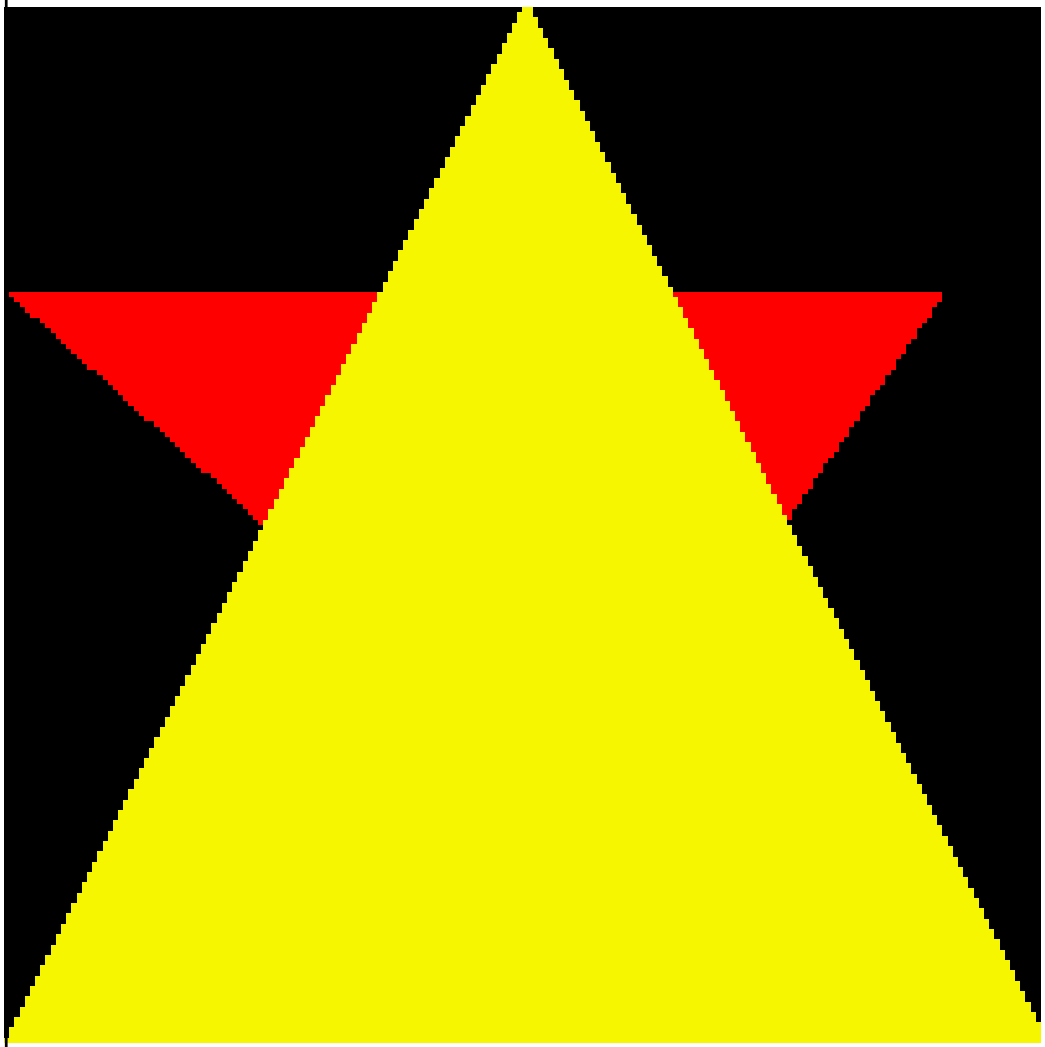


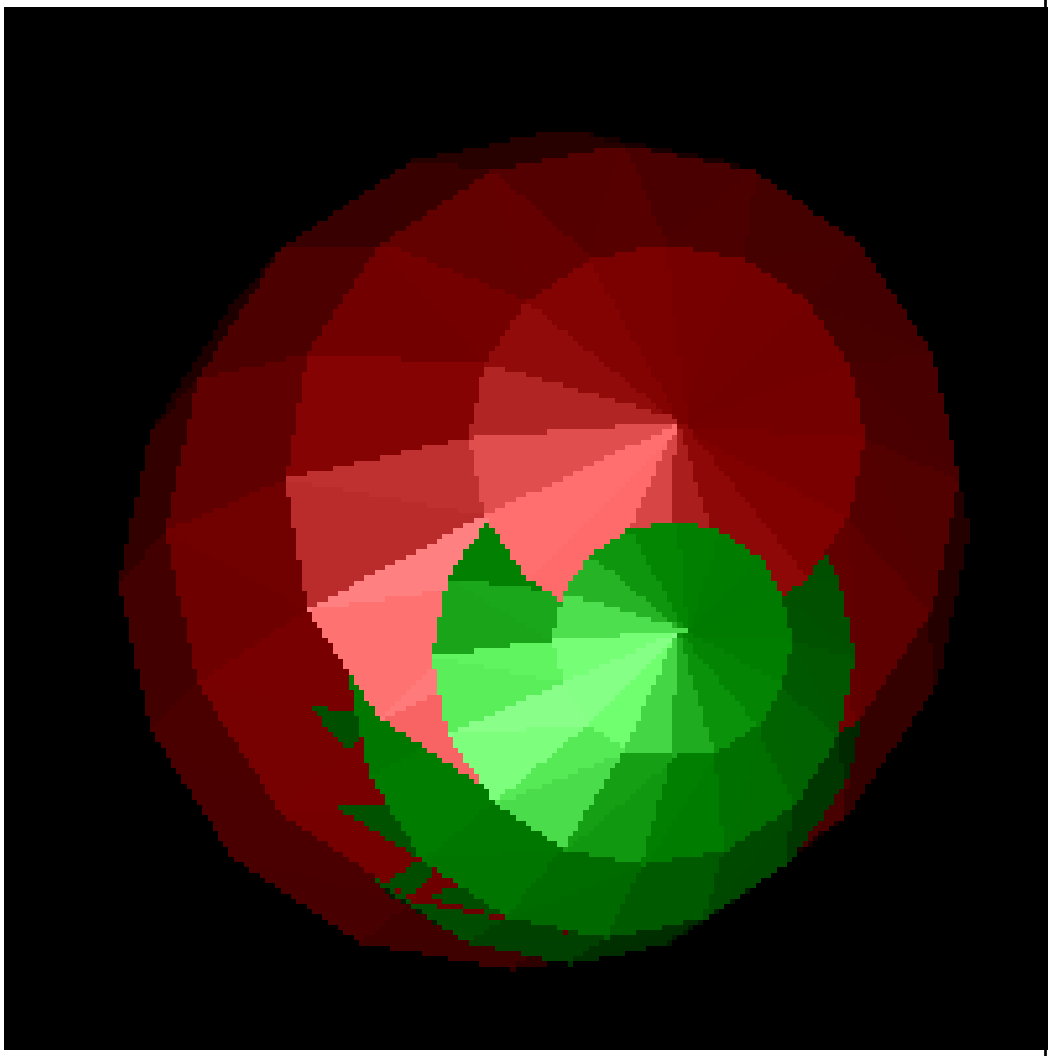


ペインタアルゴリズムによる描画例



ペインタアルゴリズムでは失敗することがある





互いに交差している物体

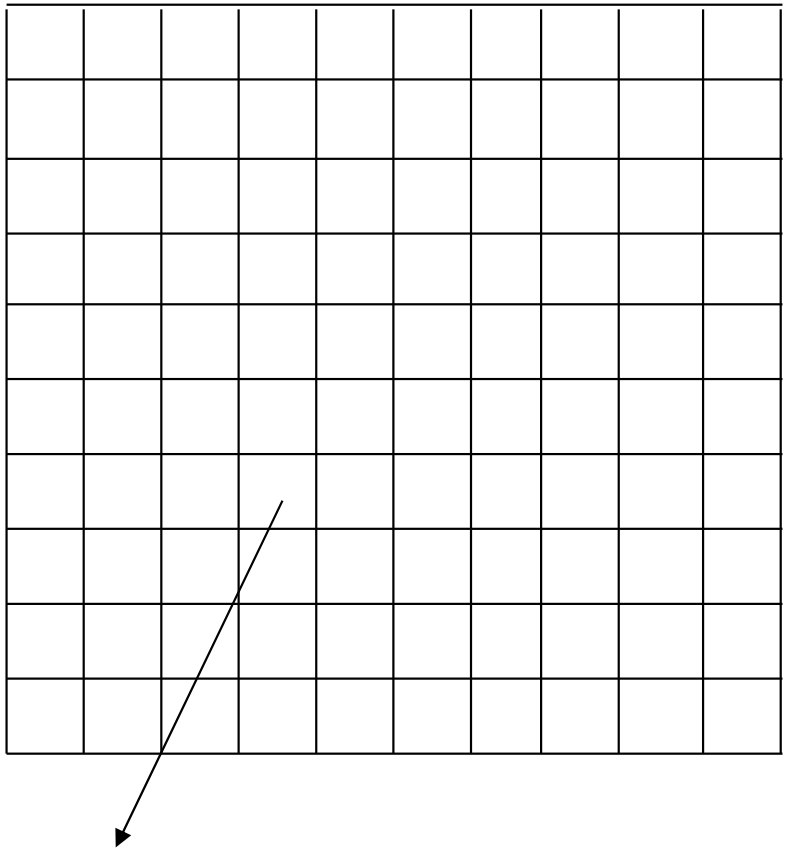
Z-バッファ法

- ３次元の隠面消去問題を１次元の「隠点」消去問題に帰着させたもの.
- Z-バッファと呼ぶ2次元配列を用いる.
- Z-バッファは，Depth（奥行き値）バッファとも呼ばれ，表示用フレームバッファ（各画素の色を保持する）と同じサイズで，各画素に対応した奥行き方向の距離（Z値）を保持する.

Z-バッファ法のアルゴリズム

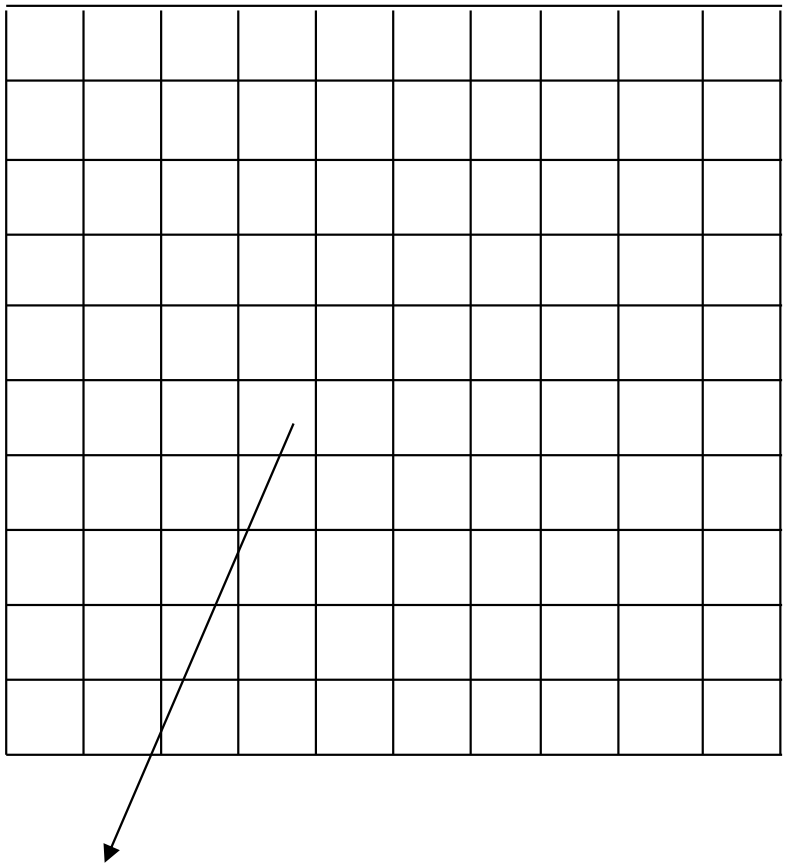
- Z−バッファの初期化（最大値を代入）
  （テキストでは最小値）
- 各３角形面毎に以下を繰り返す.
  - スクリーンに投影する.
  - スクリーンの画素毎にZ値を計算する.
  - そのZ値（Z*）とZ−バッファのZ値（Z）を比較し，
    もし，Z* < Z ならば，
    - フレームバッファにその３角形面の色を代入する.
    - Z=Z*（Z−バッファにZ*を代入する）

フレームバッファ

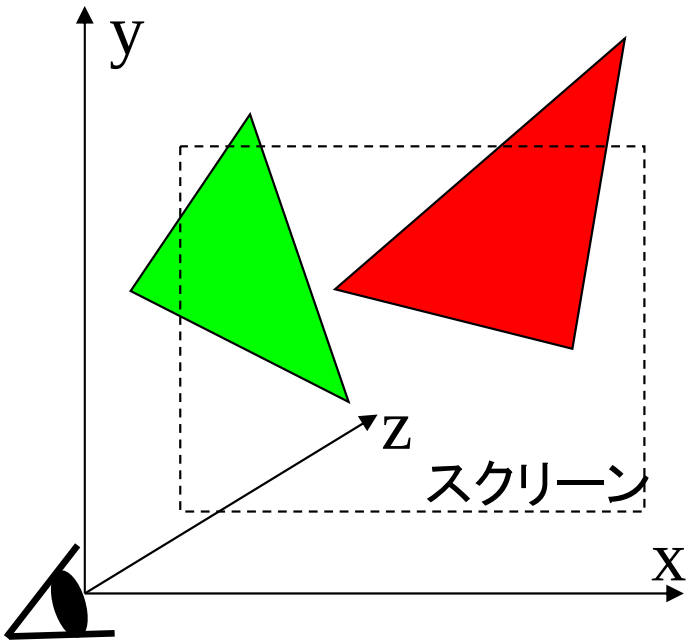


色情報（R,G,B）
24ビットならR,G,Bは
それぞれ，0～255の
整数

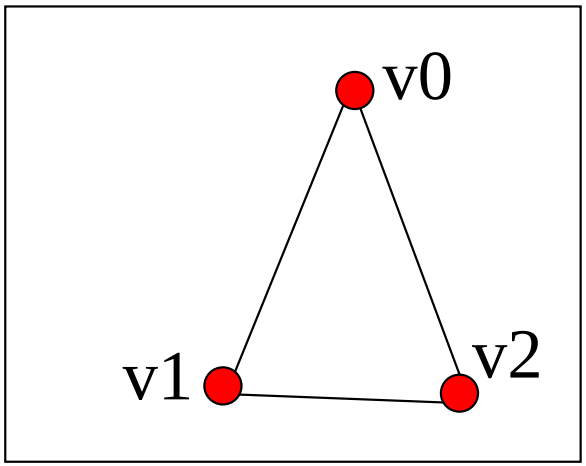
Z-バッファ



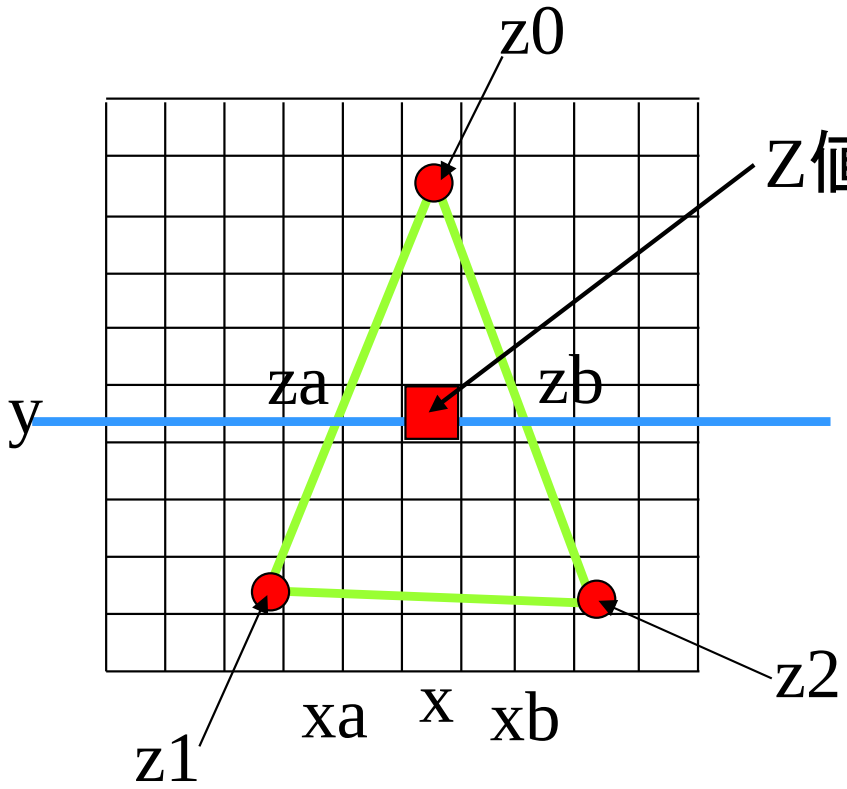
Z値（視点からの奥行き
方向の距離）



スクリーンに投影（投影変換）



v0(x0,y0) z0
v1(x1,y1) z1
v2(x2,y2) z2



Z値を線形補間で求める

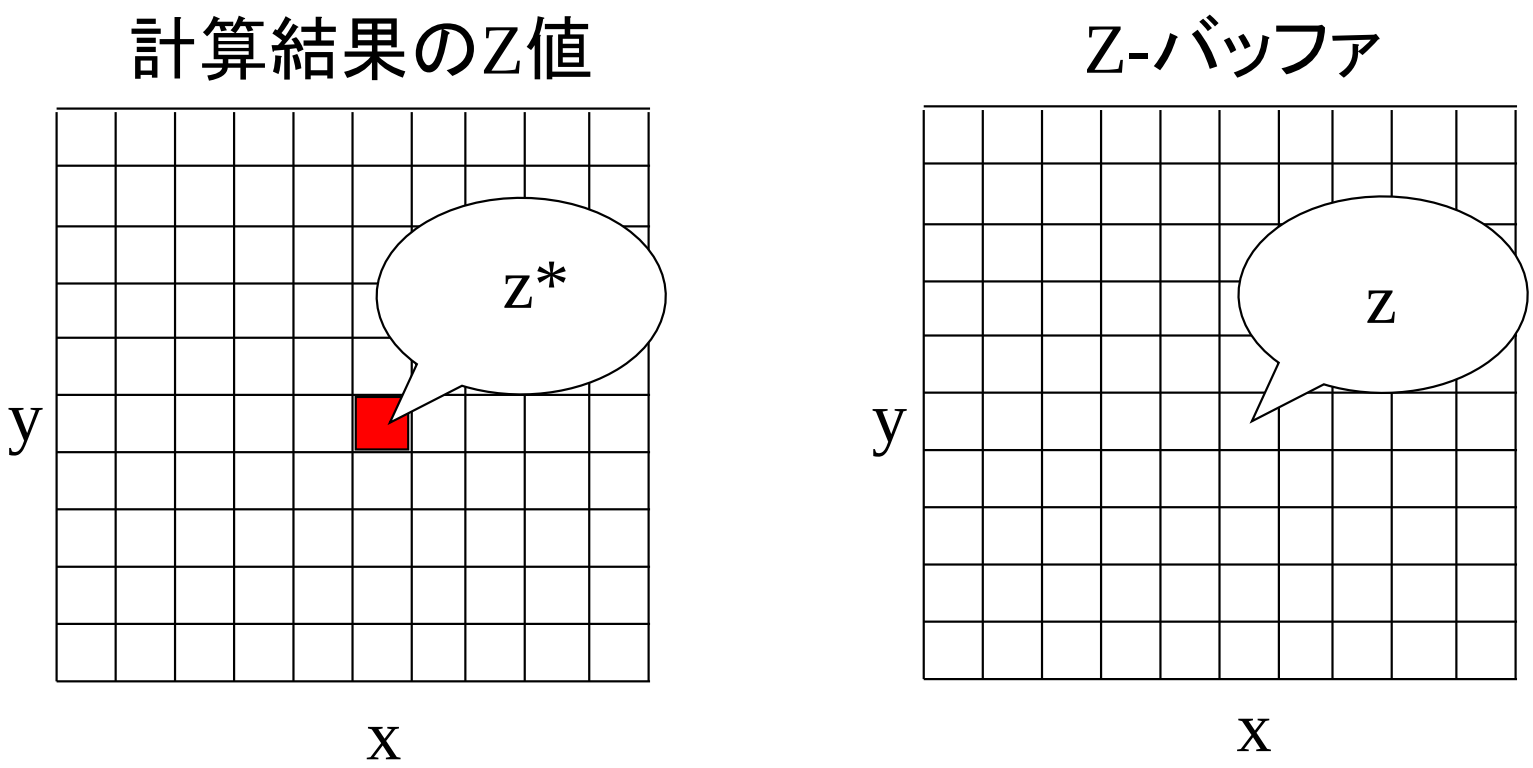
za=(z0-z1)(y-y1)/(y0-y1)+z1

zb=(z0-z2)(y-y2)/(y0-y2)+z2

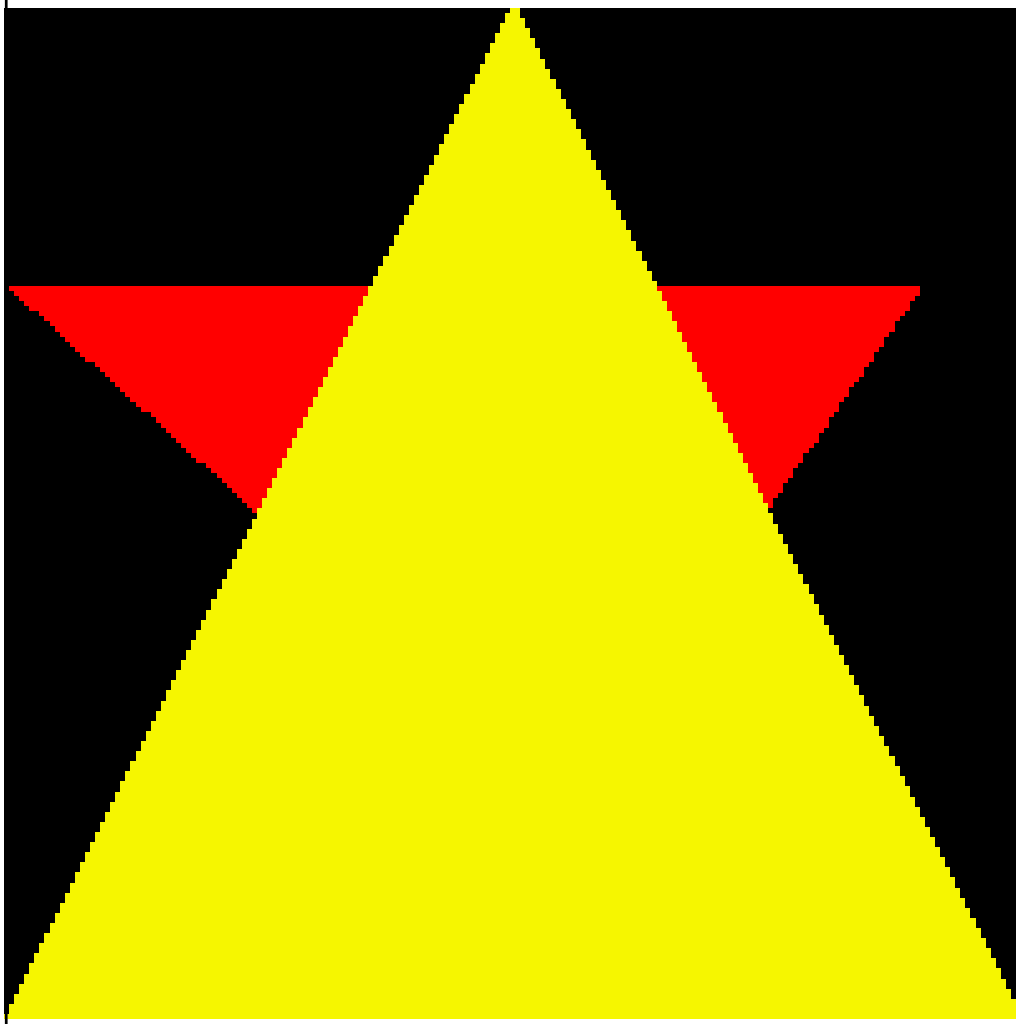
xa=(x0-x1)(y-y1)/(y0-y1)+x1

xb=(x0-x2)(y-y2)/(y0-y2)+x2

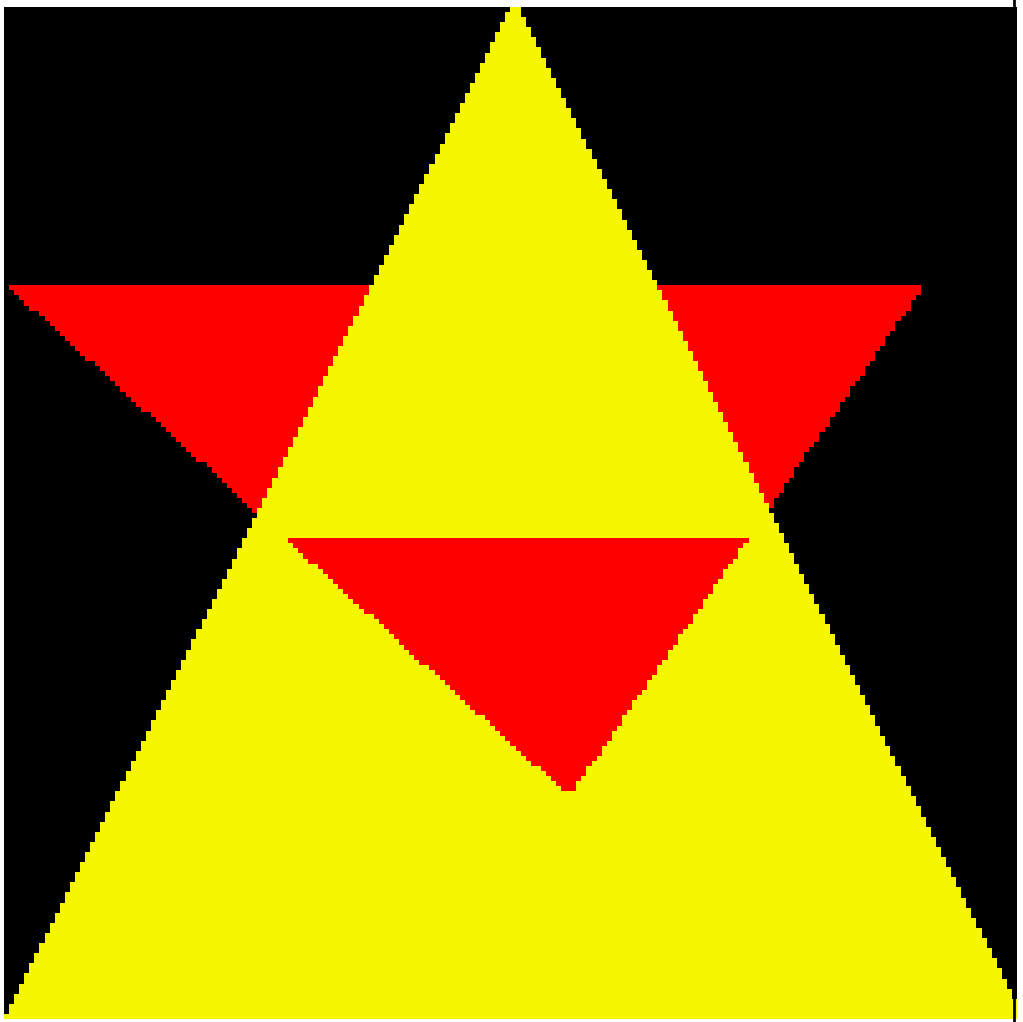
z*=(zb-za)(x-xa)/(xb-xa)+za



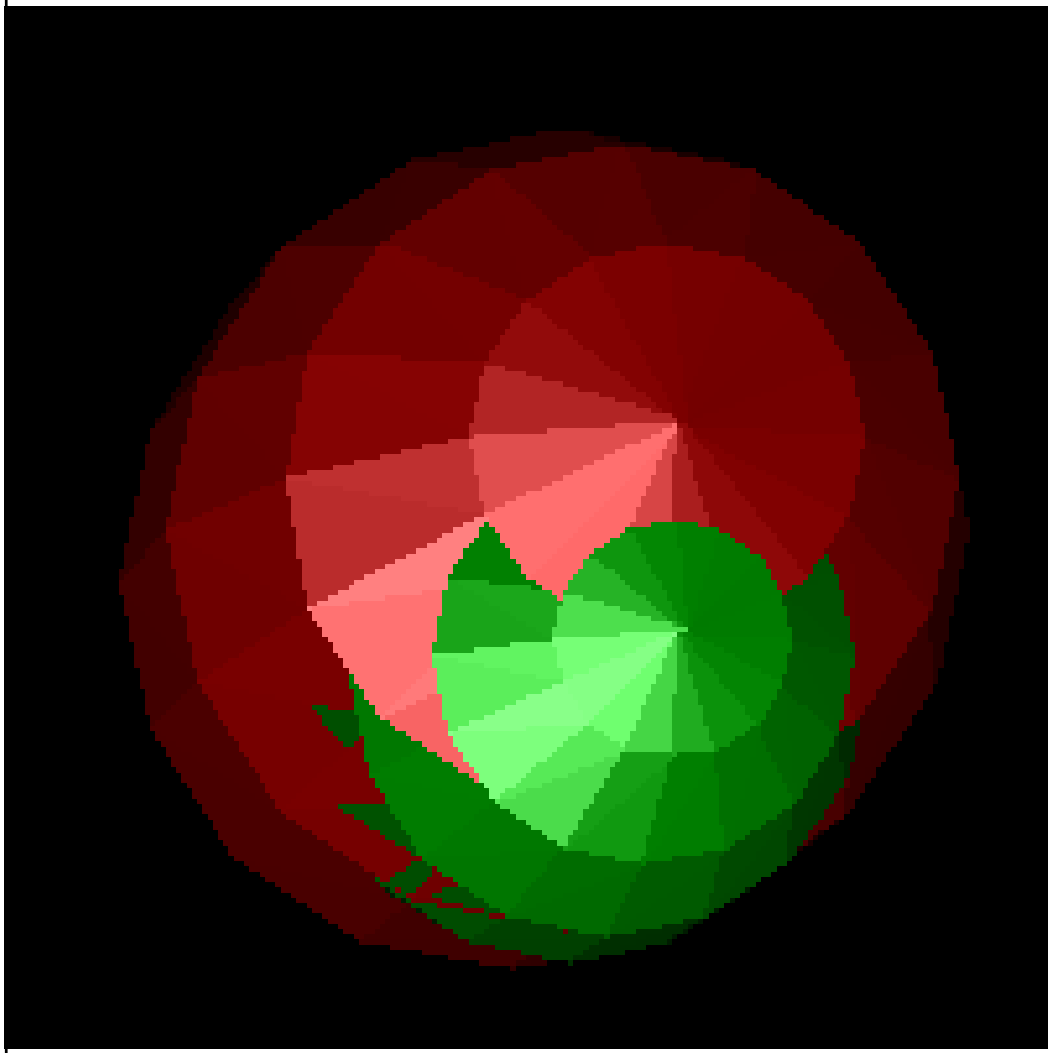
z* < zならば, (視点に近い)
フレームバッファにその3角形面の色を代入する.
z=z*(Z-バッファにz*を代入する)



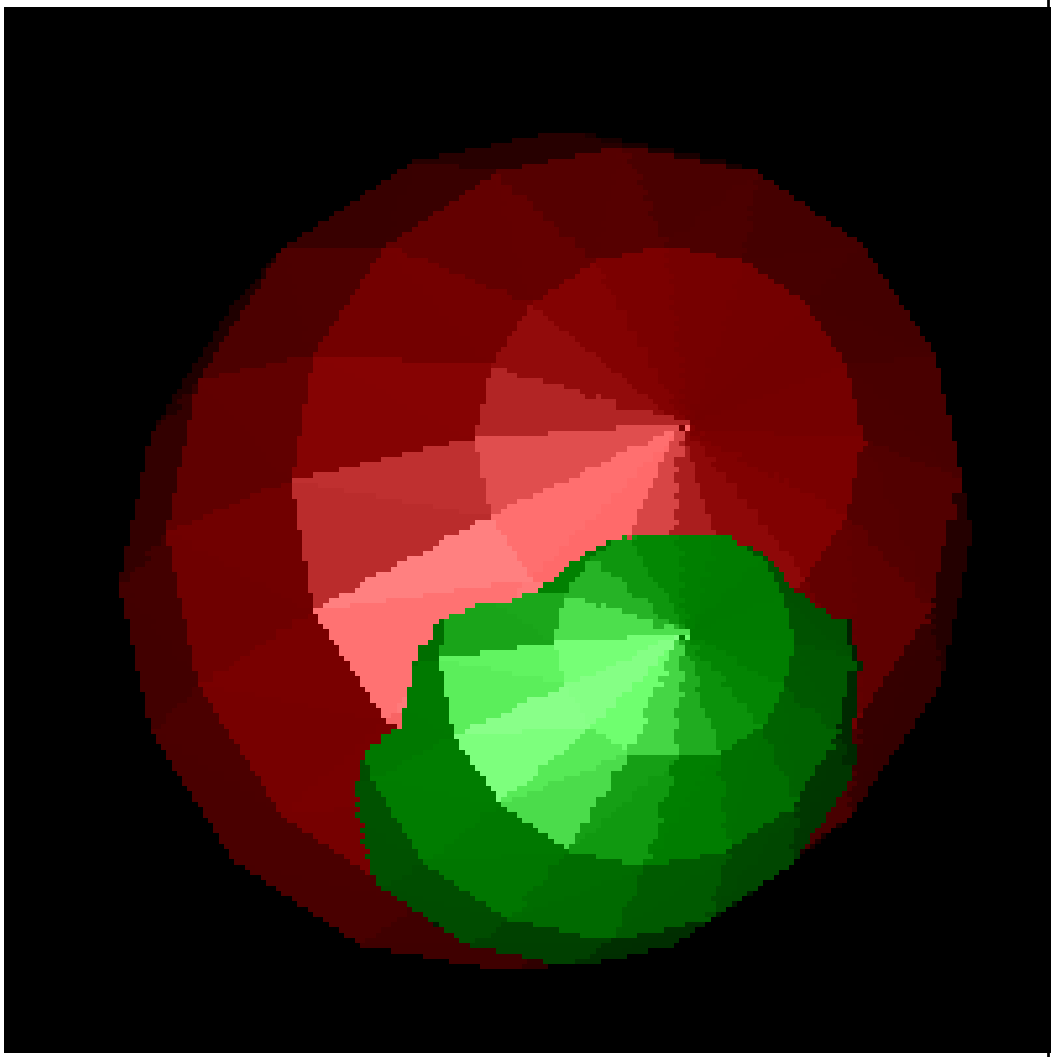
ペインタアルゴリズム



Z-バッファ法



ペインタアルゴリズム



Z-バッファ法

8章
輝度計算と
スムーズシェーディング

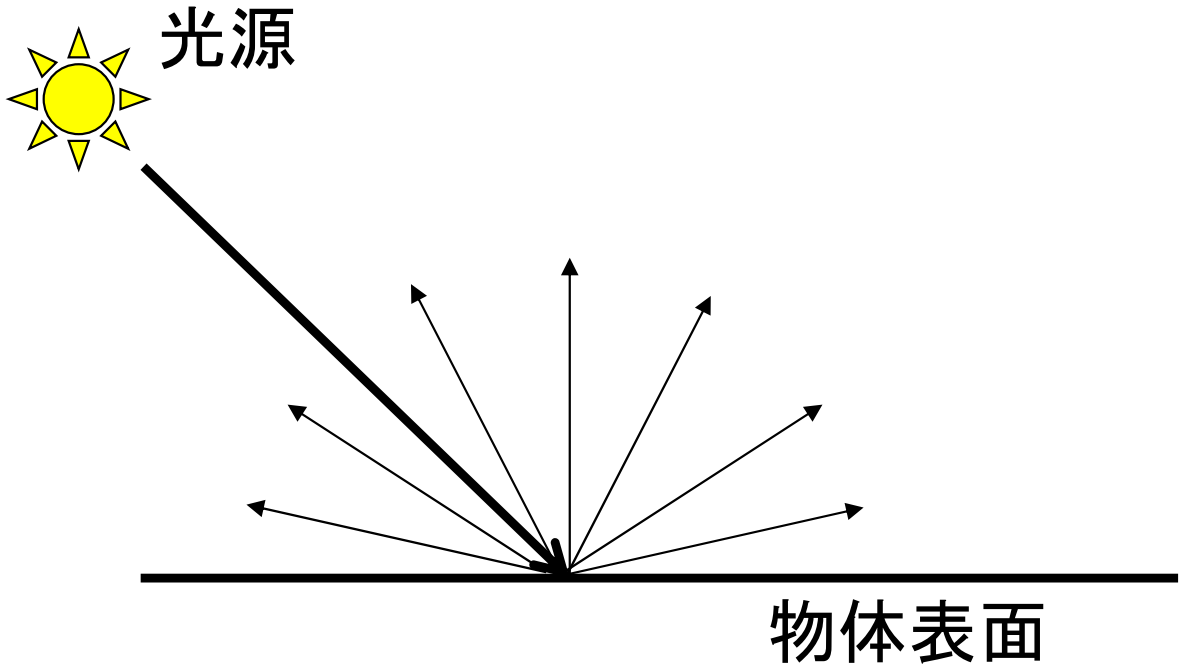
シェーディング

スムーズシェーディング

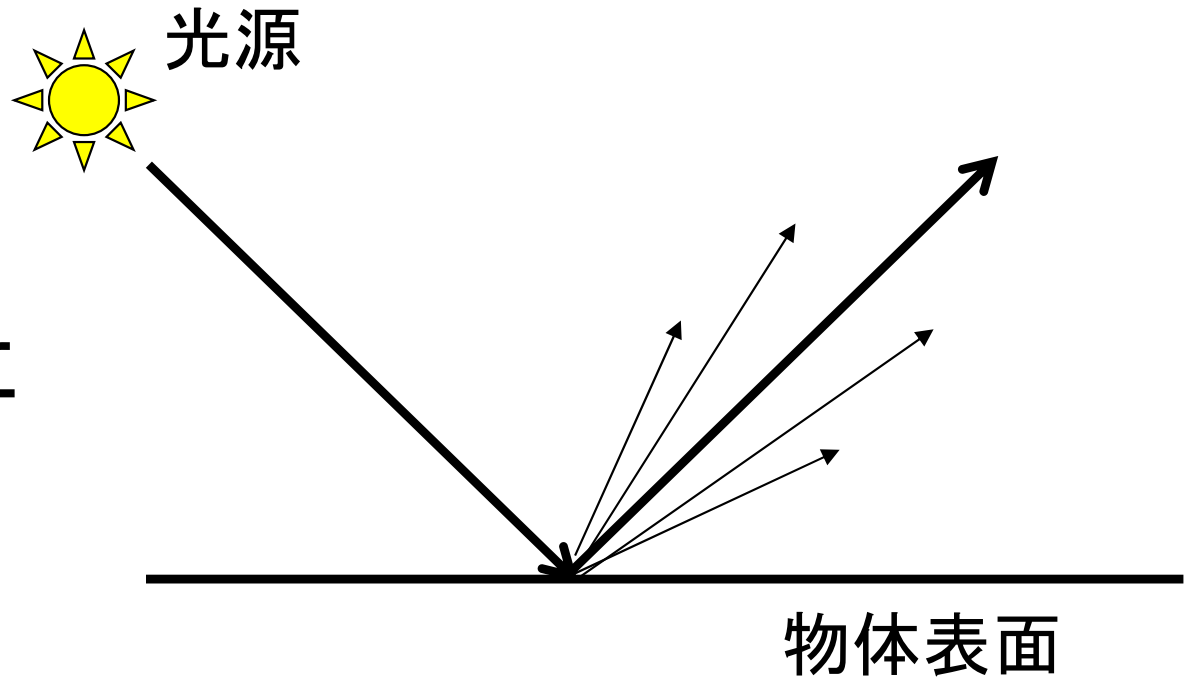
光の種類

- 環境光(ambient light)
  計算の難しい光を環境光とする（間接光による拡散反射や大気による散乱光など）
- 拡散反射光(diffusely reflected light)
  全方向に均等に反射する光（布地などでの反射）
- 鏡面反射光(specularly reflected light)
  ある方向（正反射方向）に強く反射する光（鏡での反射）
- 透過光(transmitted light)
  透明な物体を通過してきた光（屈折率を考慮）

拡散反射
（全方向に
一様に反射）

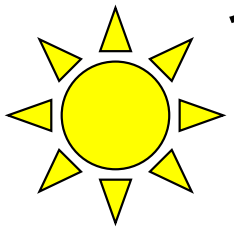


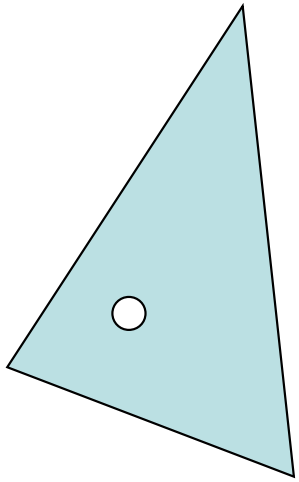
鏡面反射
（正反射方向に
強く反射）



シェーディングモデル
１点の輝度計算

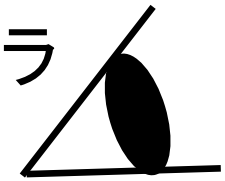
光源





この面のある点の
明るさ（輝度）は？

カメラ



- 環境光 $I_a = K_a * I_{amb}$

  K_aは面の反射係数，I_{amb}環境光の強さ
  一般に I_a=一定 とする

- 拡散反射光（ランバートの余弦則）
  $I_d = K_d * I_{in} * \cos\theta$

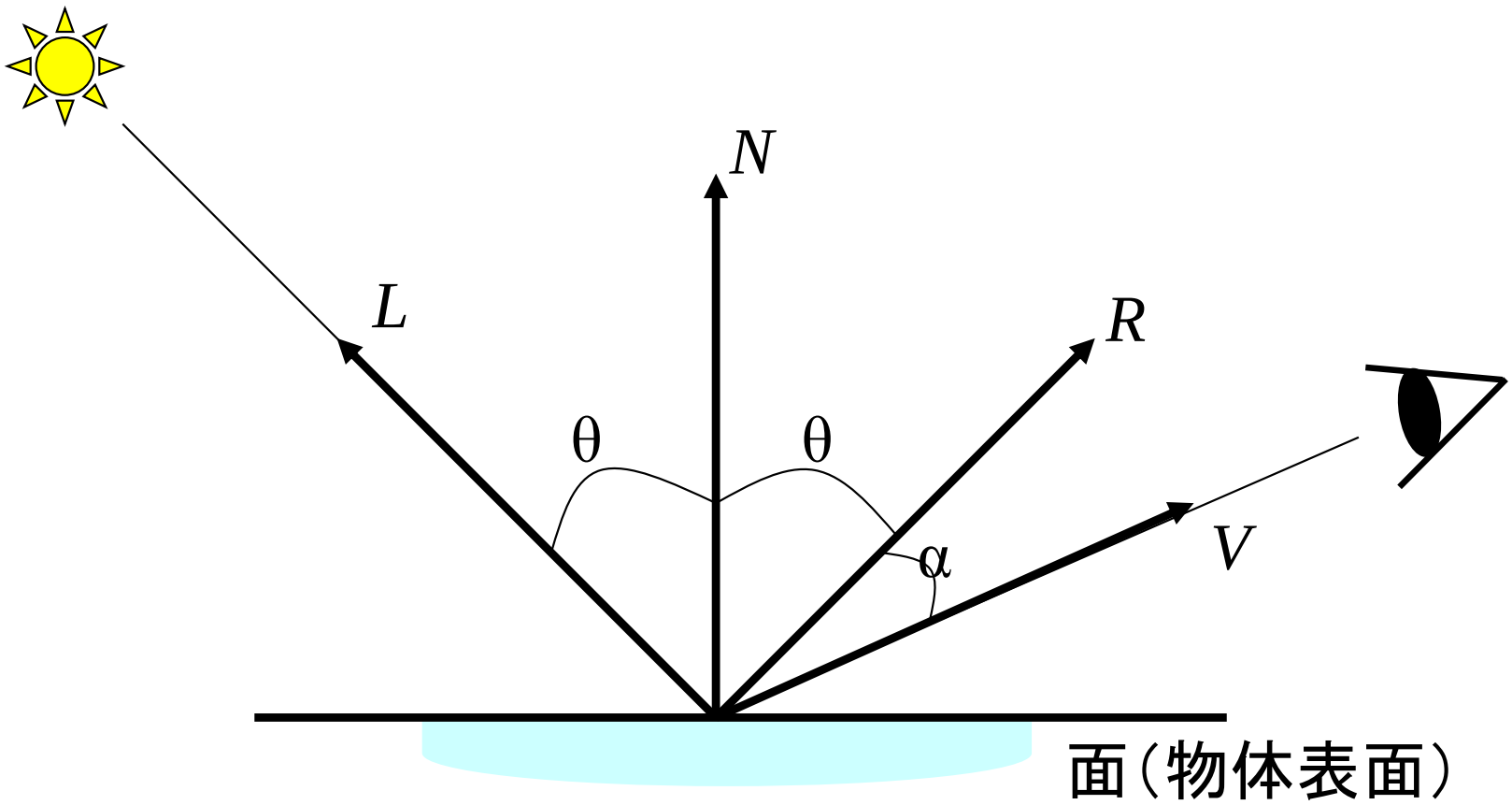
  K_dは拡散反射係数，I_{in}は入射光の強さ，
  θは光線方向と面の法線方向とのなす角度

  $(0 \leqq \theta \leqq \pi/2)$

  $\cos\theta = L \cdot N$（ベクトルの内積）

- 鏡面反射光（フォンの反射モデル）
  $I_s = K_s * I_{in} * \cos^n\alpha$

  K_sは鏡面反射係数，I_{in}は入射光の強さ，
  nはハイライト係数(反射光の鋭さを規定する定数)，
  αは正反射方向と視線方向とのなす角度
  $\cos\alpha = R \cdot V$



L：光源への方向ベクトル
N：面の法線ベクトル
R：正反射方向ベクトル
V：視点への方向ベクトル

ベクトルL,N,R,Vは単位ベクトル

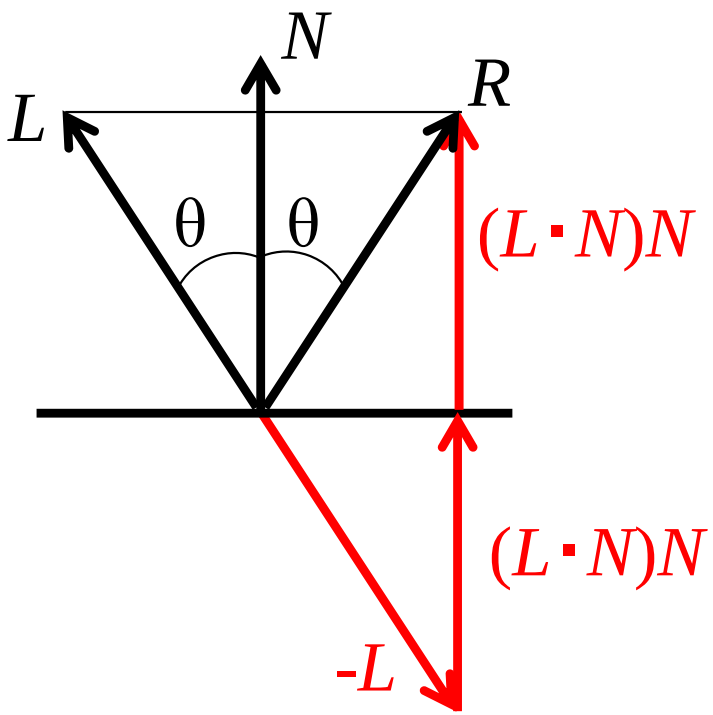
光源が1つの場合

輝度$I = I_a + I_d + I_s$

$= K_a * I_{amb} + k_d * I_{in} * \cos\theta + K_s * I_{in} * \cos^n\alpha$

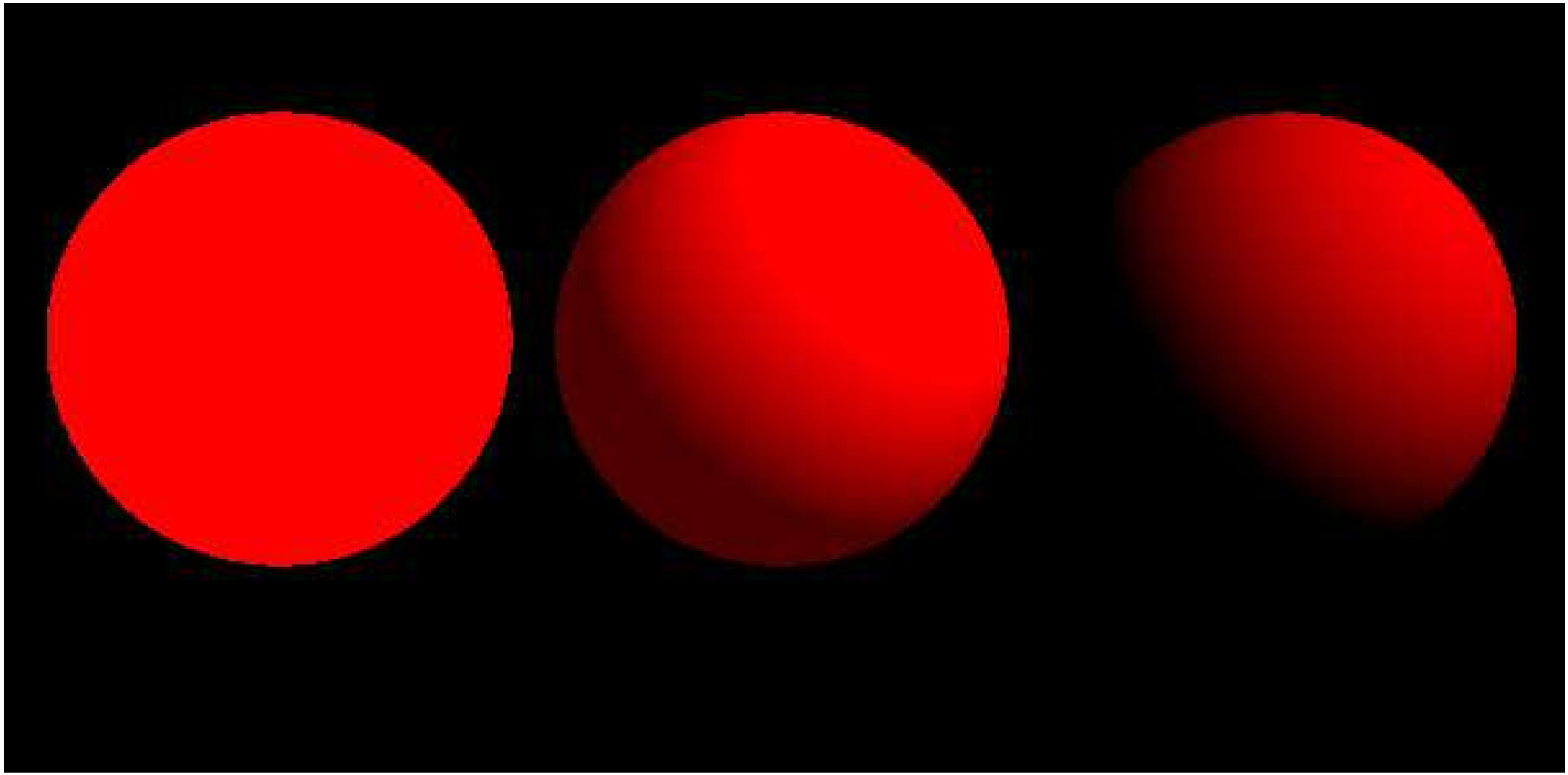
$= K_a * I_{amb} + K_d * I_{in} * (L \cdot N) + K_s * I_{in} * (R \cdot V)^n$

光源が複数の場合は2，3項目を光源数分たし合わせる.



正反射ベクトル

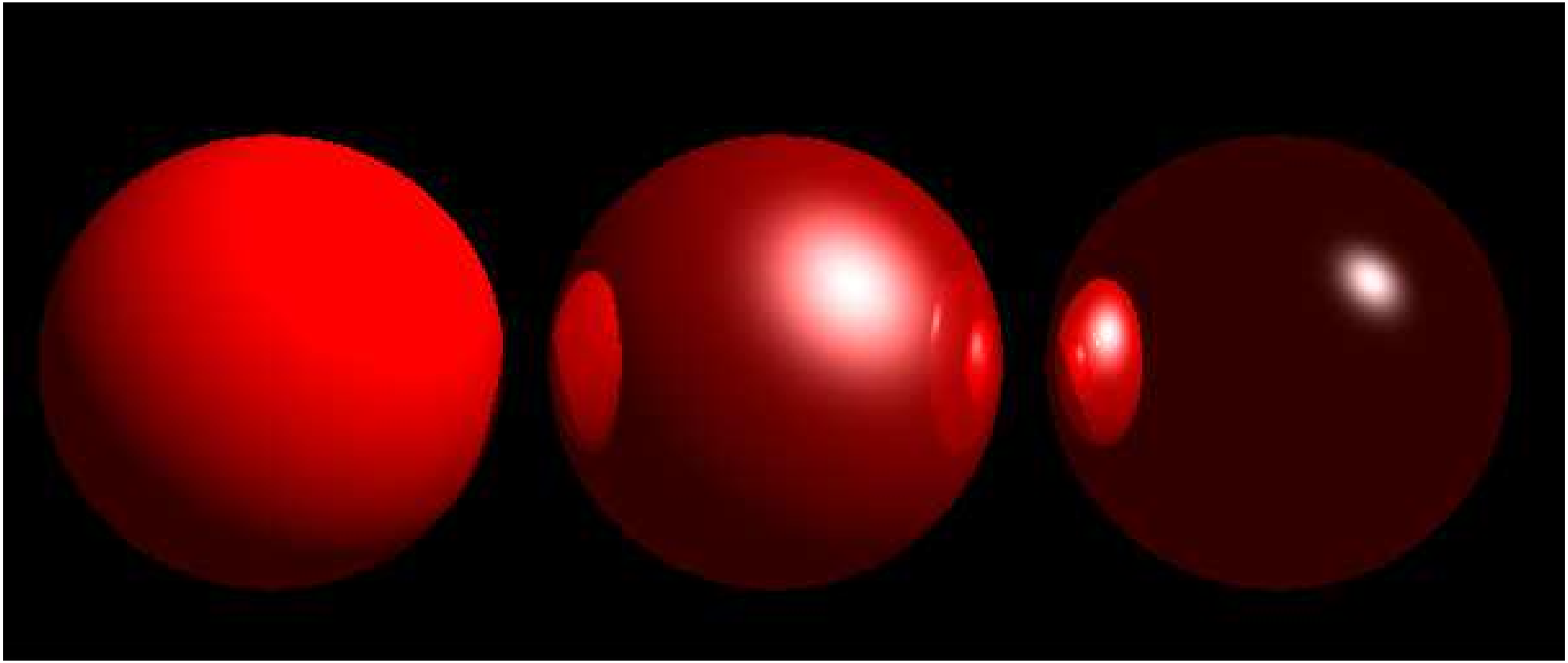
$R = 2(L \cdot N)N - L$



環境光のみ

環境光と
拡散反射光

拡散反射光のみ



拡散反射のみ

拡散反射と
鏡面反射

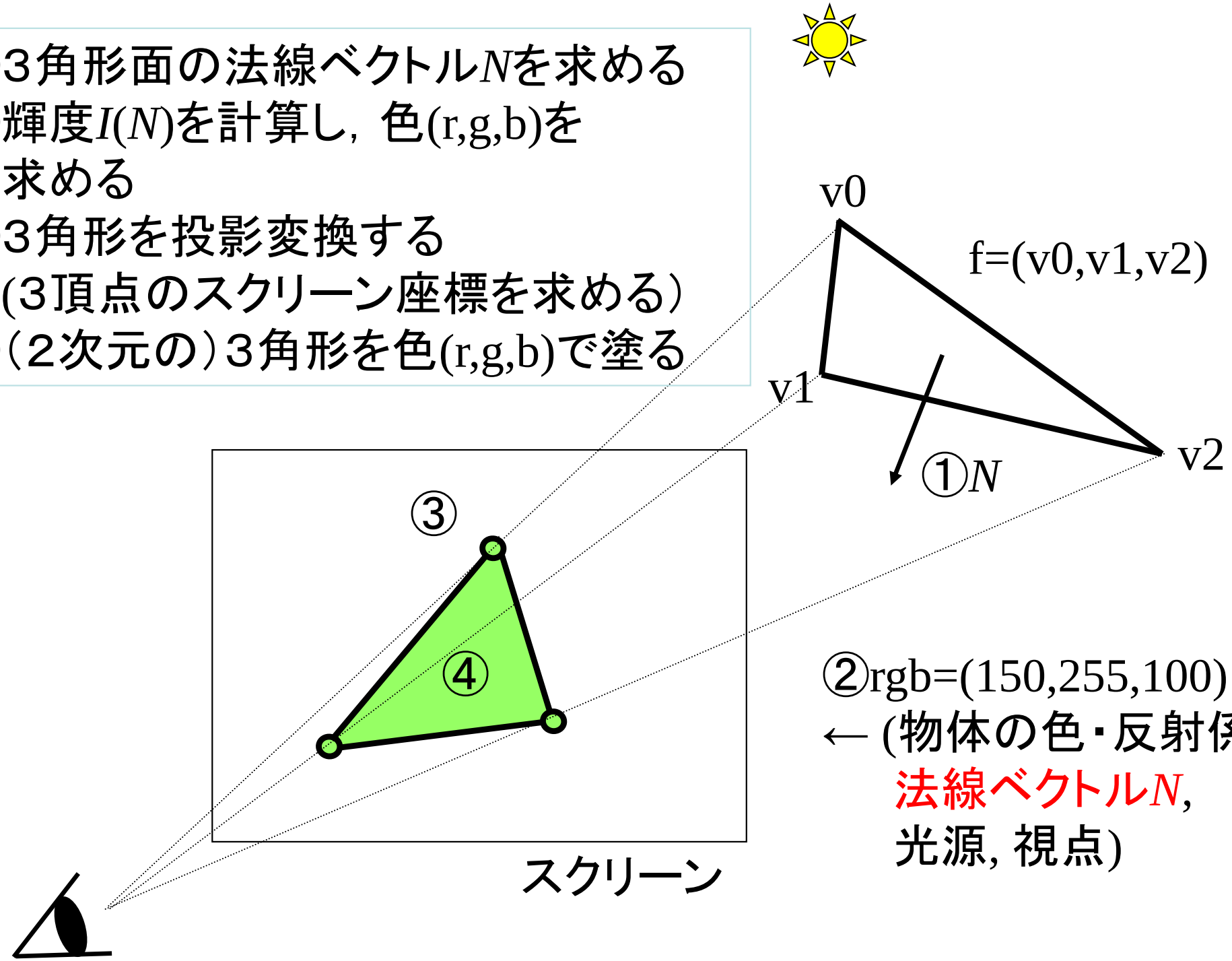
鏡面反射のみ

（環境光はすべてあり）

３角形面一枚の描画手順
(単純な描画法＝フラットシェーディング)

1. ３角形面の法線ベクトルNを求める.
2. 代表点（重心など）の輝度$I(N)$を計算し，その点の表示色(r,g,b)を求める.
3. ３角形を投影変換する.
   (3頂点のスクリーン座標を求める. ）
4. （2次元の）３角形を表示色(r,g,b)で塗る.

①3角形面の法線ベクトルNを求める
②輝度$I(N)$を計算し，色(r,g,b)を
　求める
③3角形を投影変換する
　(3頂点のスクリーン座標を求める)
④(2次元の)3角形を色(r,g,b)で塗る



②rgb=(150,255,100)
← (物体の色・反射係数,
　法線ベクトルN,
　光源, 視点)

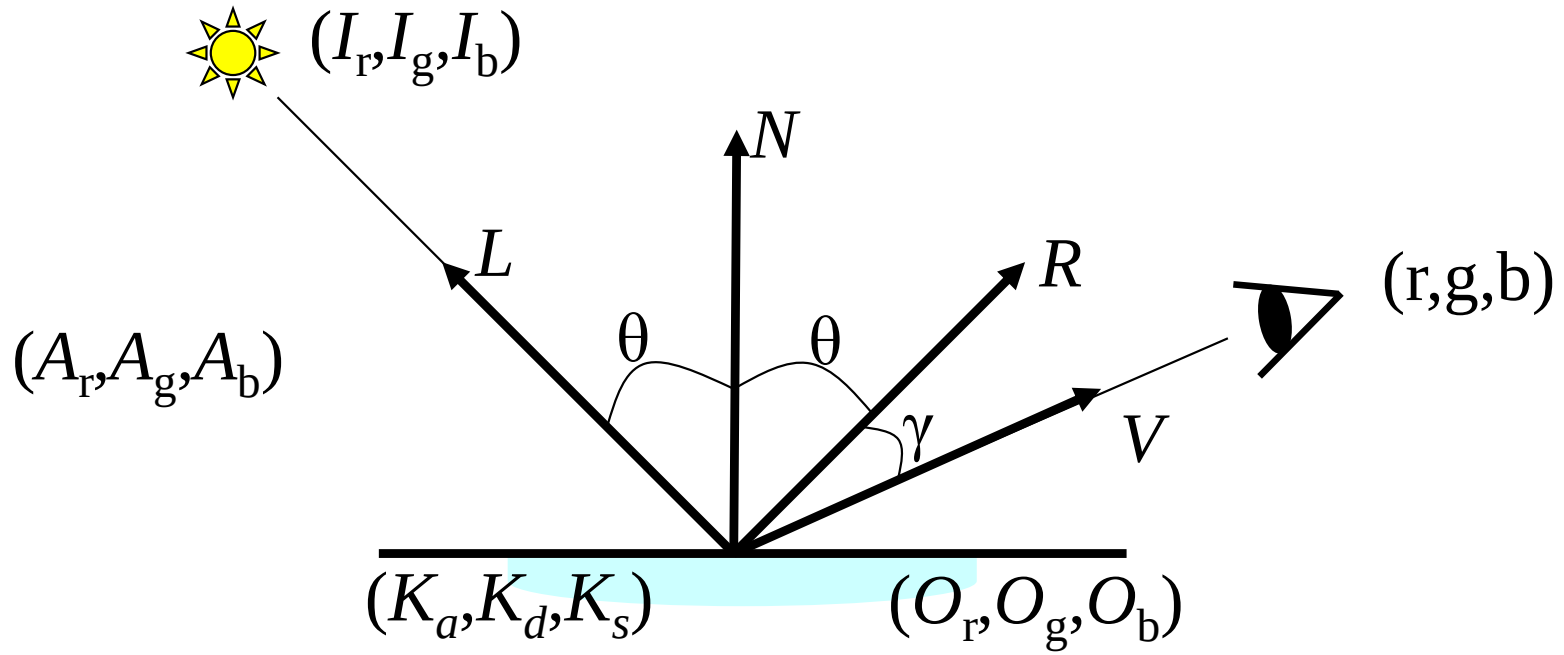
輝度$I = K_a * I_{amb} + K_d * I_{in} * (L \cdot N) + K_s * I_{in} * (R \cdot V)^n$

物体の元の色を(O_r, O_g, O_b)，光源の色を(I_r, I_g, I_b)，環境光の色を(A_r, A_g, A_b)とすれば，表示色(r,g,b)は，

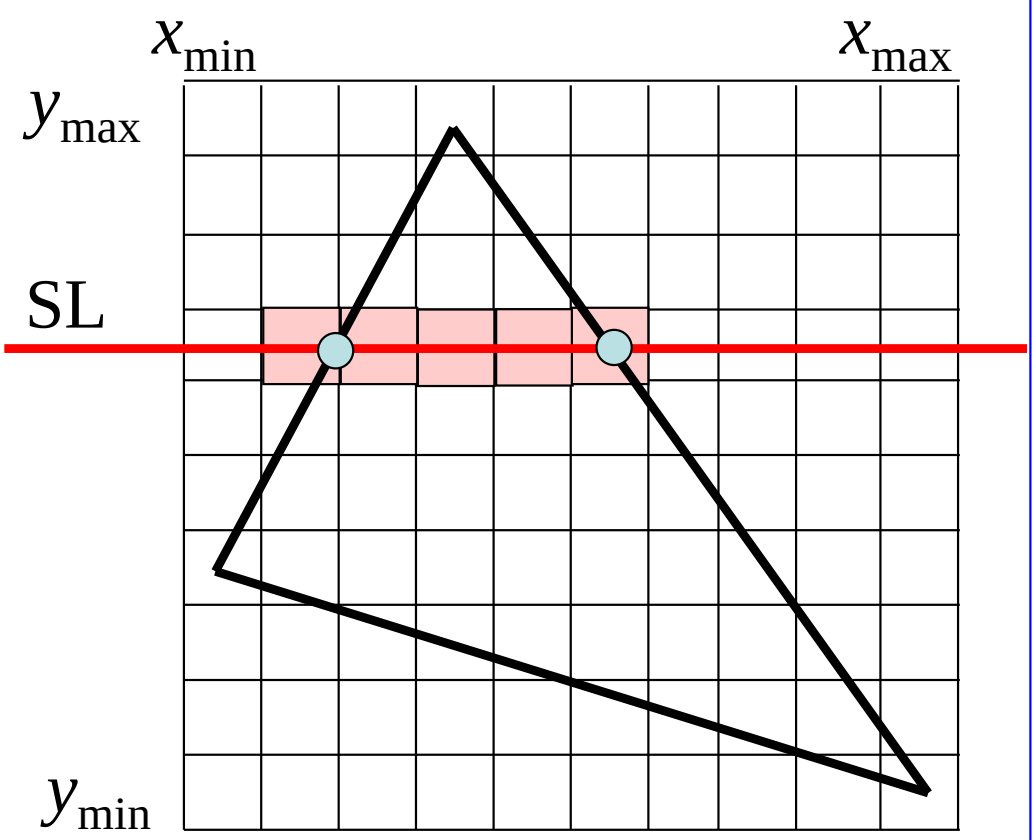
$$r = K_a * O_r * A_r + K_d * O_r * I_r * (L \cdot N) + K_s * I_r * (R \cdot V)^n$$

$$g = K_a * O_g * A_g + K_d * O_g * I_g * (L \cdot N) + K_s * I_g * (R \cdot V)^n$$

$$b = K_a * O_b * A_b + K_d * O_b * I_b * (L \cdot N) + K_s * I_b * (R \cdot V)^n$$



３角形面を描く（２次元）

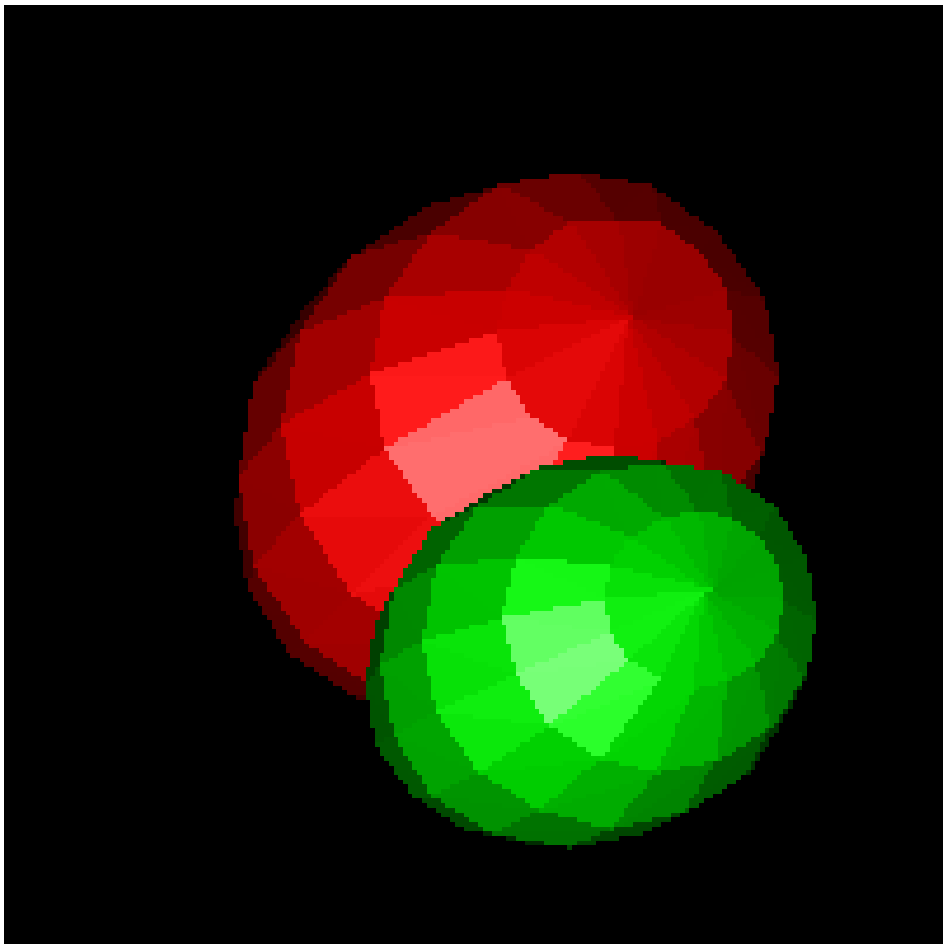


1. ３角形を囲む長方形
   $(x_{min},x_{max},y_{min},y_{max})$
2. Y座標をy_{max}～y_{min}まで，
   １ずつ減らしながら以下を
   繰り返す
   (a)スキャンラインSL($Y=y$)発生
   (b)SLと３角形との交点
   （線形補間）
   (c)交点間の画素を
   表示色(r,g,b)
   で塗る

＊端点や水平線に注意

スムーズシェーディング

- 各3角形面毎に輝度一定で表示すると，滑らかに見えない．
  - フラットシェーディング
- 3角形面の内部の輝度を変化させる．
  - グローシェーディング
  - フォンシェーディング



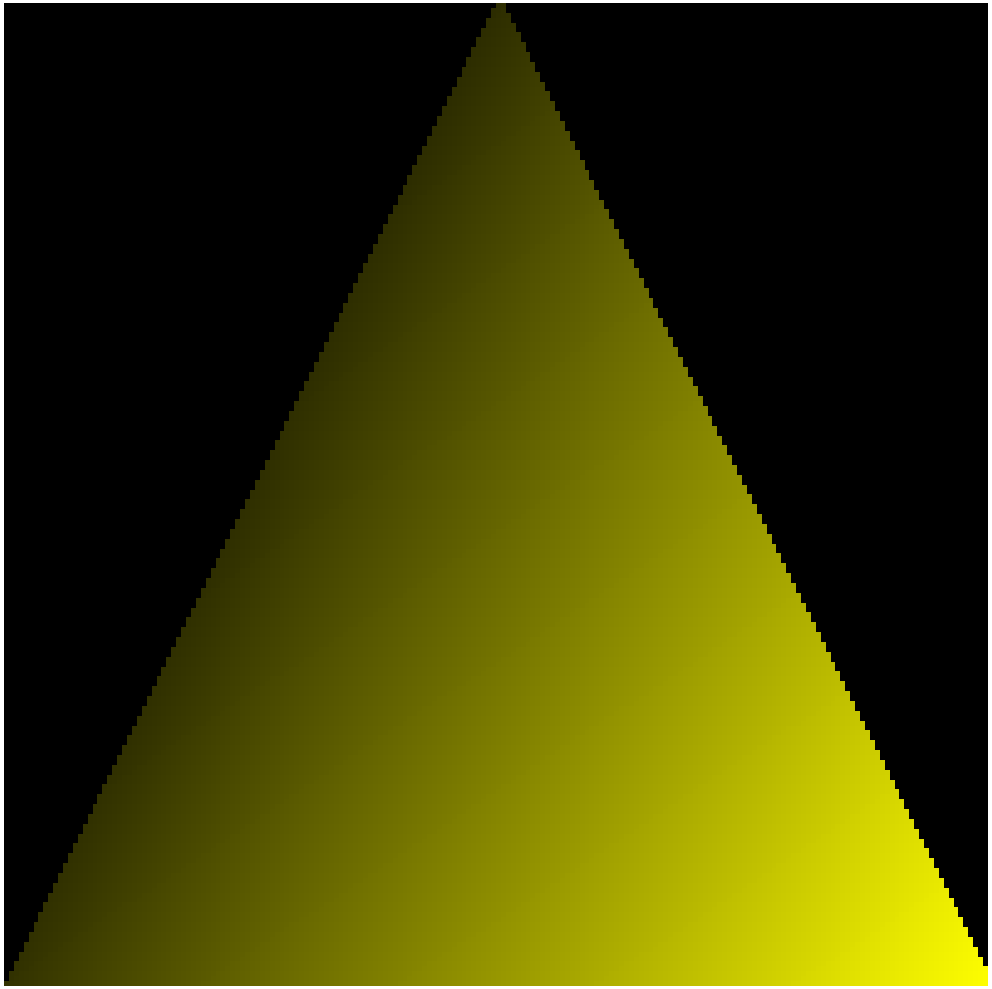
フラットシェーディング

グローシェーディング
（輝度補間法）

- 曲面を3角形分割する.
- 各頂点の法線ベクトルを求める.
  - 各頂点のまわりの3角形面の法線ベクトルの平均
  - 元の曲面から求める
- 各頂点の法線ベクトルから，各頂点の輝度を計算し，表示色(r,g,b)を求める.
- 3角形面内部の点の表示色は各頂点の表示色から(r,g,b毎に)線形補間により求める.

輝度（色）の線形補間

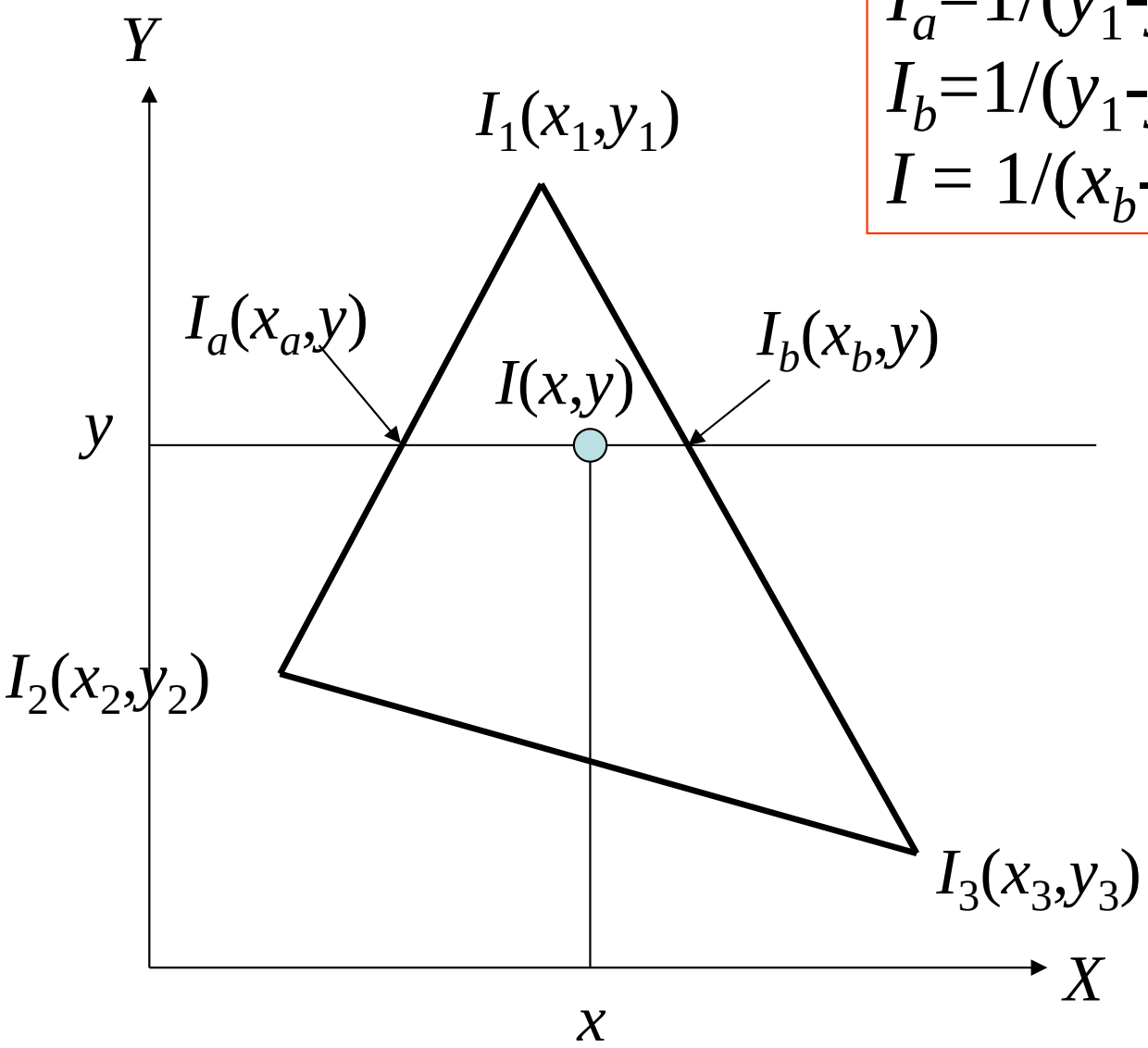
(60,60,0)



(60,60,0)　　(255,255,0)

３角形内部の点(x,y)の輝度$I(x,y)$を線形補間で求める

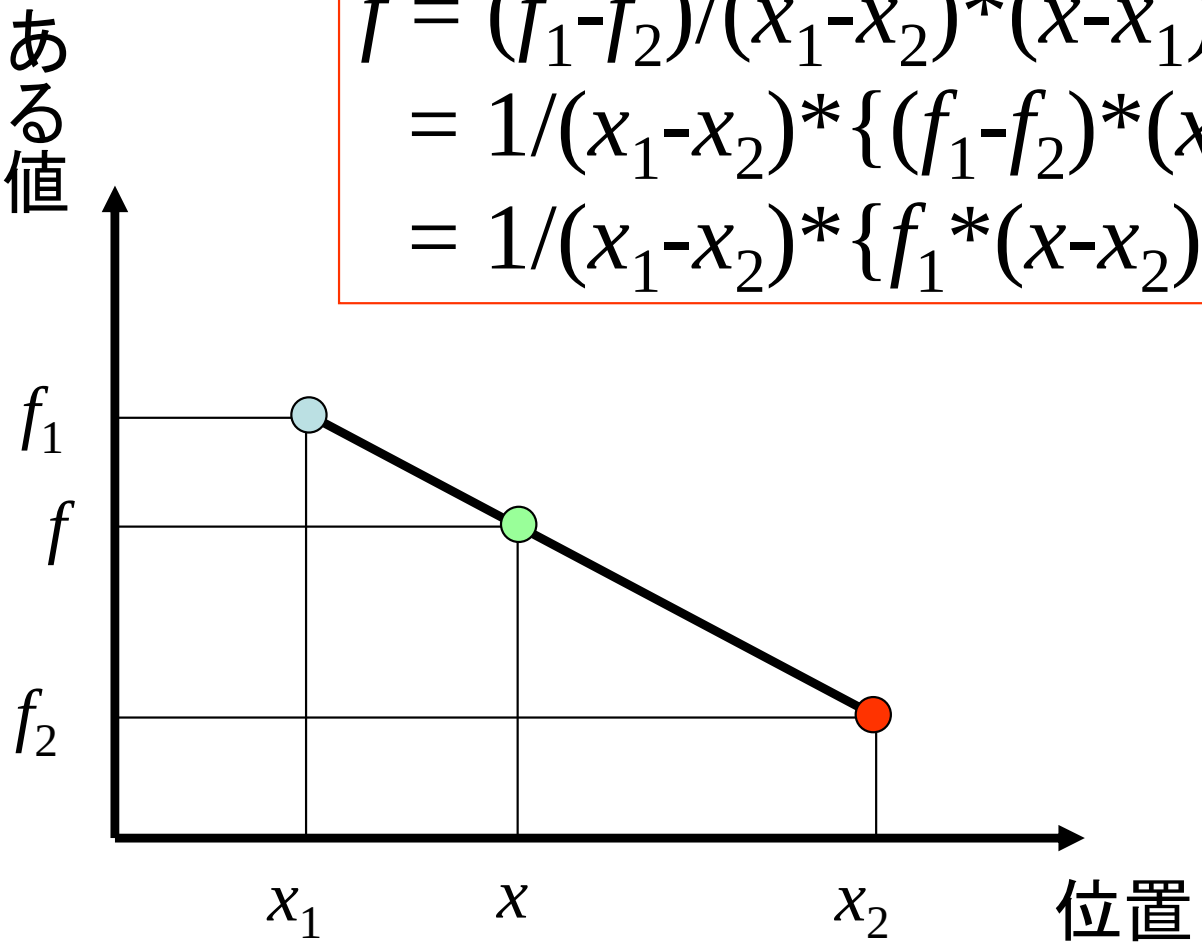
$$I_a=1/(y_1-y_2)*(I_1*(y-y_2)+I_2*(y_1-y))$$
$$I_b=1/(y_1-y_3)*(I_1*(y-y_3)+I_3*(y_1-y))$$
$$I = 1/(x_b-x_a)*(I_b*(x-x_a)+I_a*(x_b-x))$$



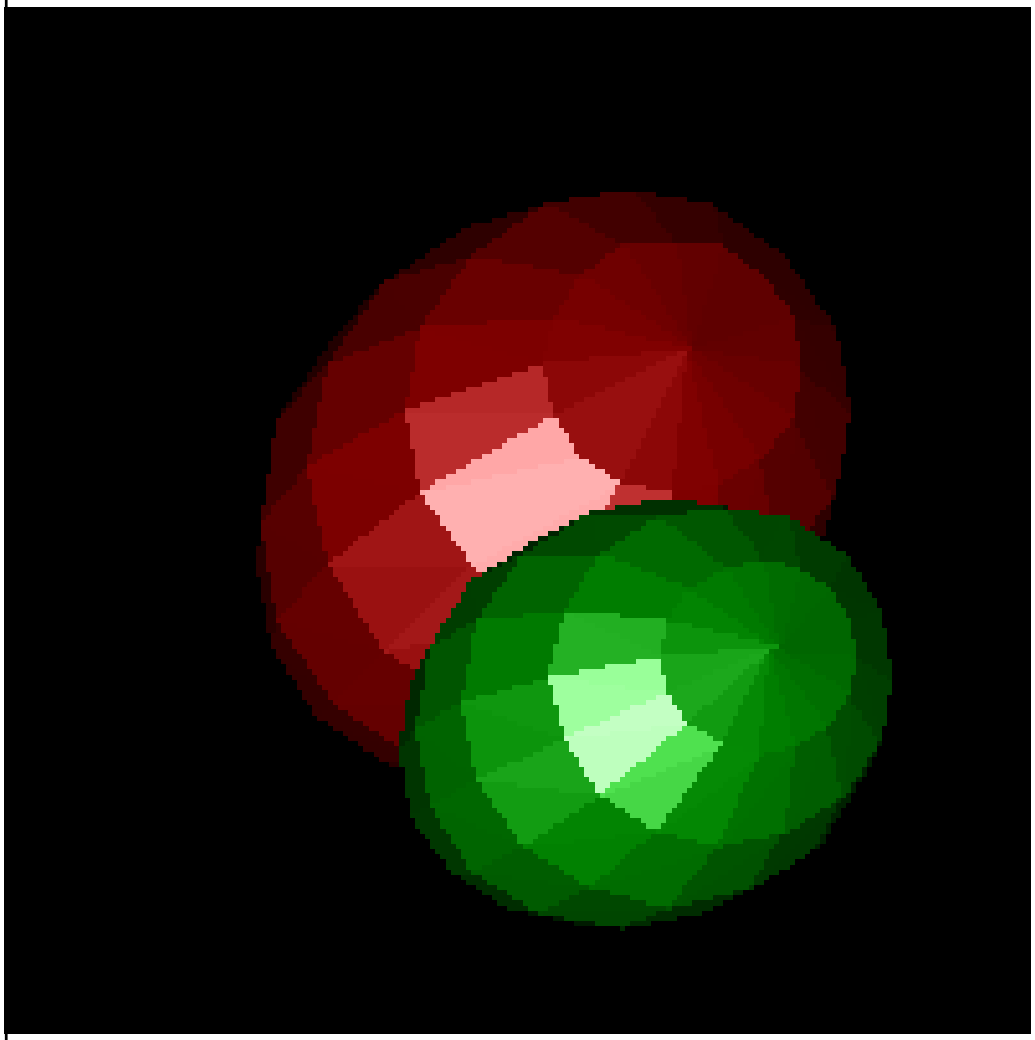
線形補間の式

２点$(x_1,f_1),(x_2,f_2)$を通る直線:

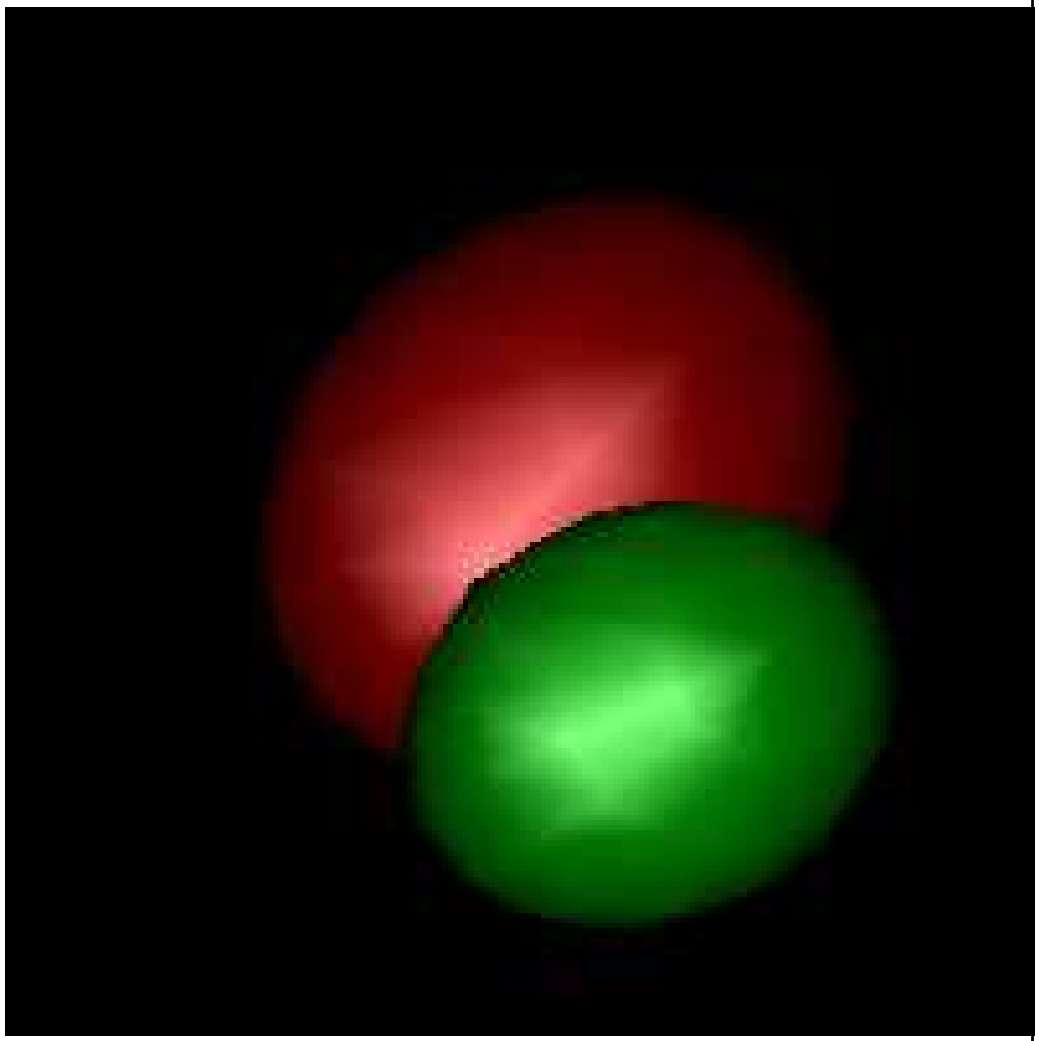
$$f = (f_1-f_2)/(x_1-x_2)*(x-x_1)+f_1$$
$$= 1/(x_1-x_2)*\{(f_1-f_2)*(x-x_1)+f_1*(x_1-x_2)\}$$
$$= 1/(x_1-x_2)*\{f_1*(x-x_2)+f_2*(x_1-x)\}$$



フラットシェーディング
(スムーズシェーディングなし)



グロー(Gouraud)
シェーディング



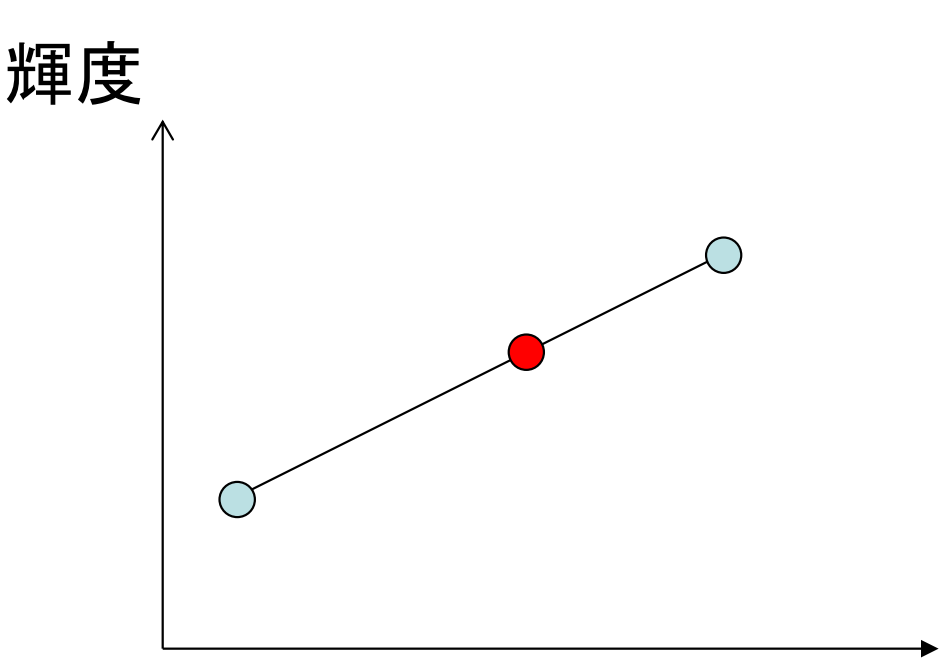
$K_a=K_d=0.5$, $K_s=0.8$, $n=4$

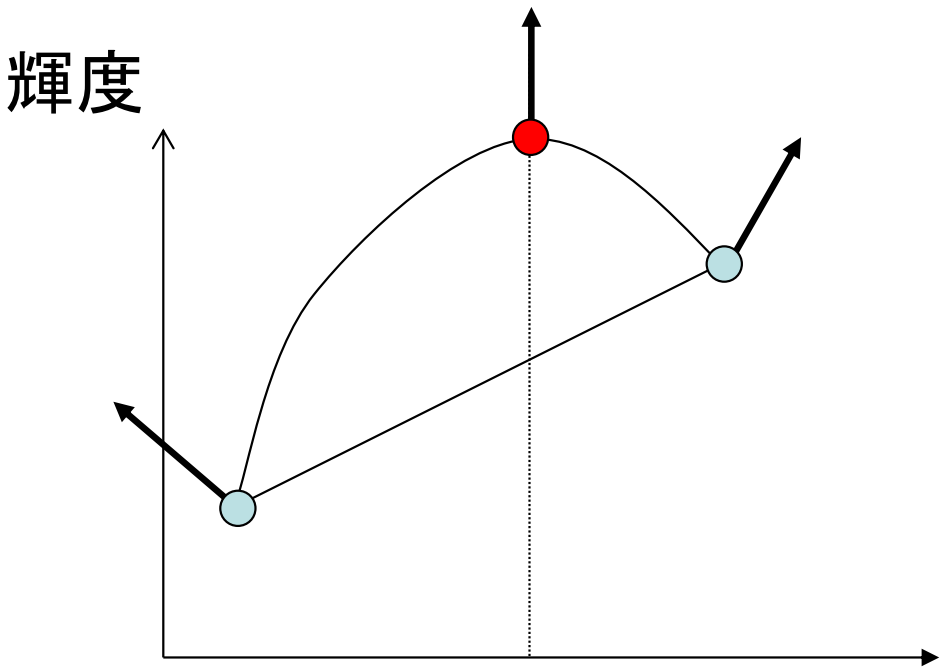
フォンシェーディング
（法線補間法）

- 曲面を3角形分割する.
- 各頂点の法線ベクトルを求める.
  - 各頂点のまわりの3角形面の法線ベクトルの平均
  - 元の曲面から求める
- 3角形面内部の点の法線ベクトルを各頂点の法線ベクトルから線形補間により求め，その点の表示色(r,g,b)を計算する.

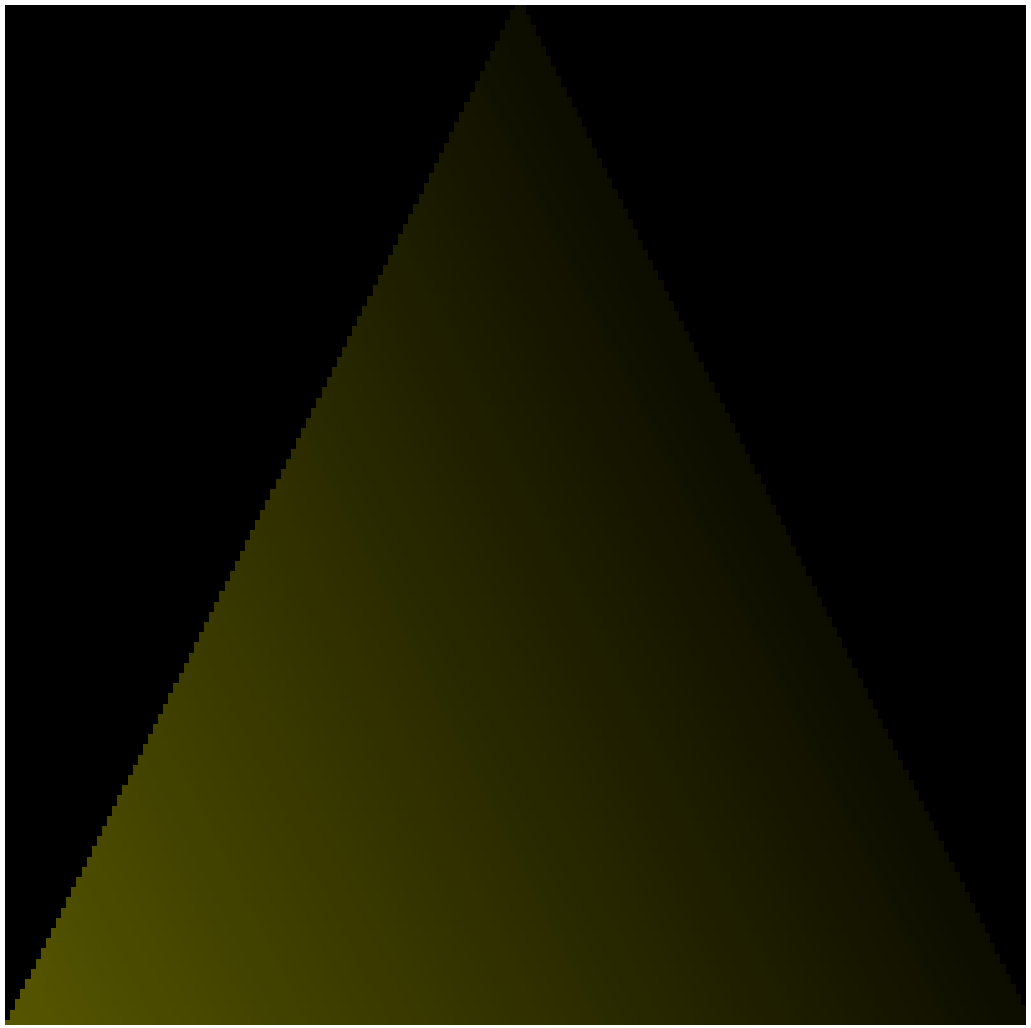
グローシェーディング
・輝度(色)を補間する
・ハイライトを見逃しやすい
(精度が低い)
・計算が速い

フォンシェーディング
・法線ベクトルを補間する
・ハイライトもOK
(精度が高い)
・計算量が多い

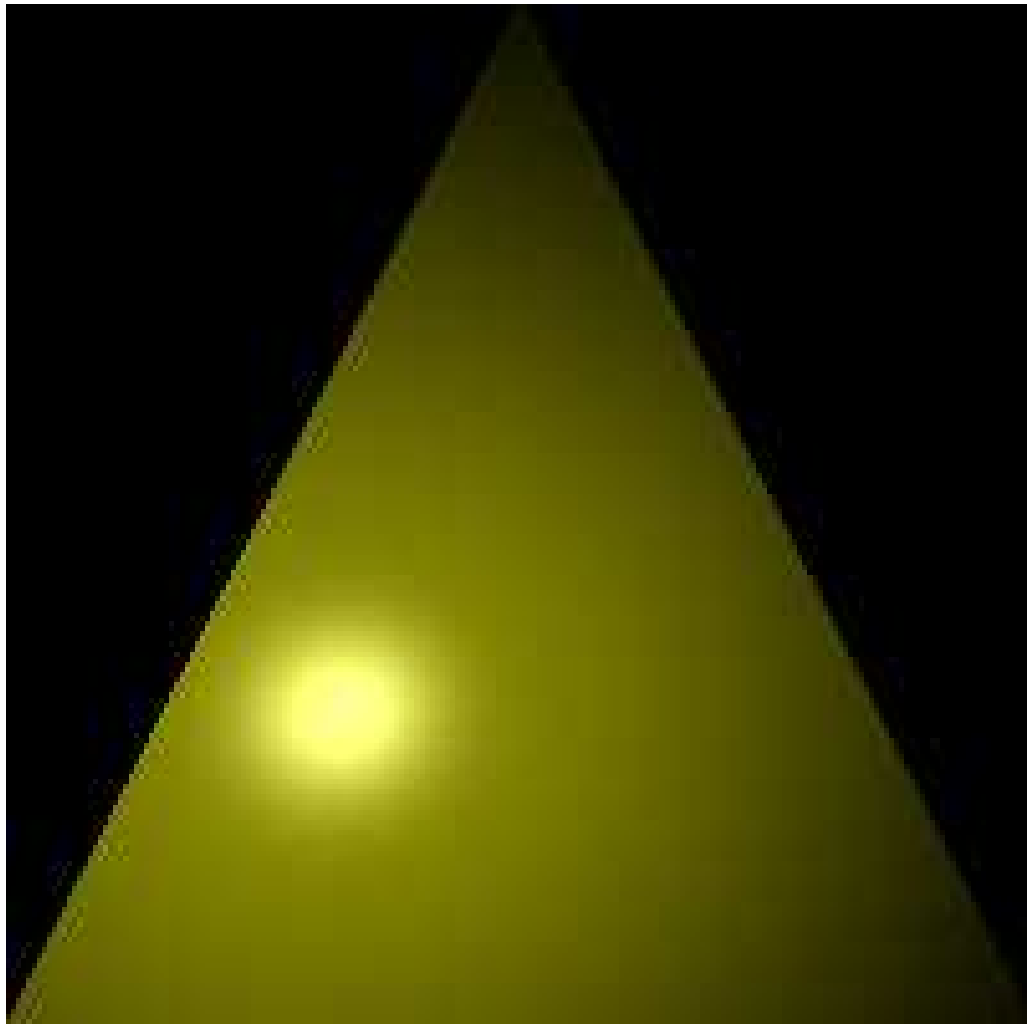




グローシェーディング

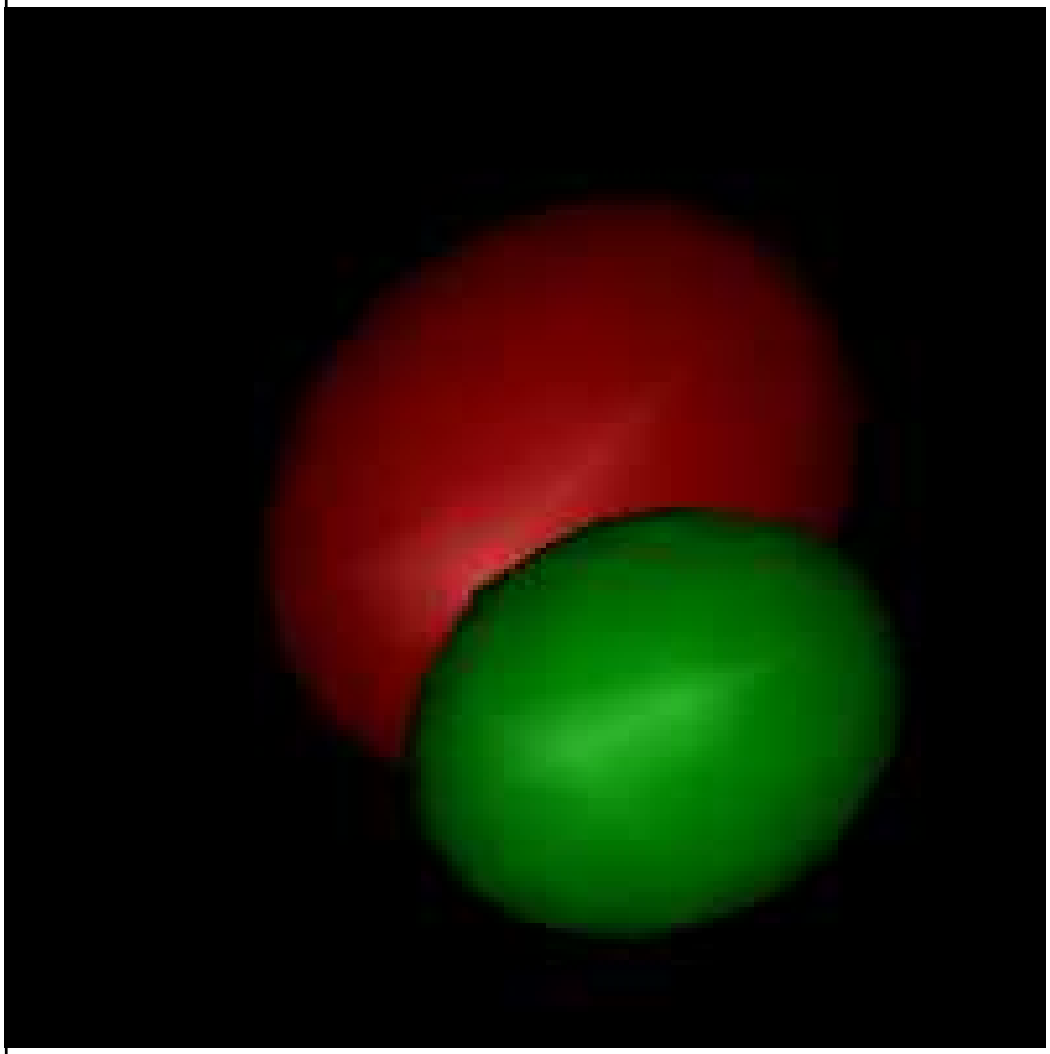


フォンシェーディング

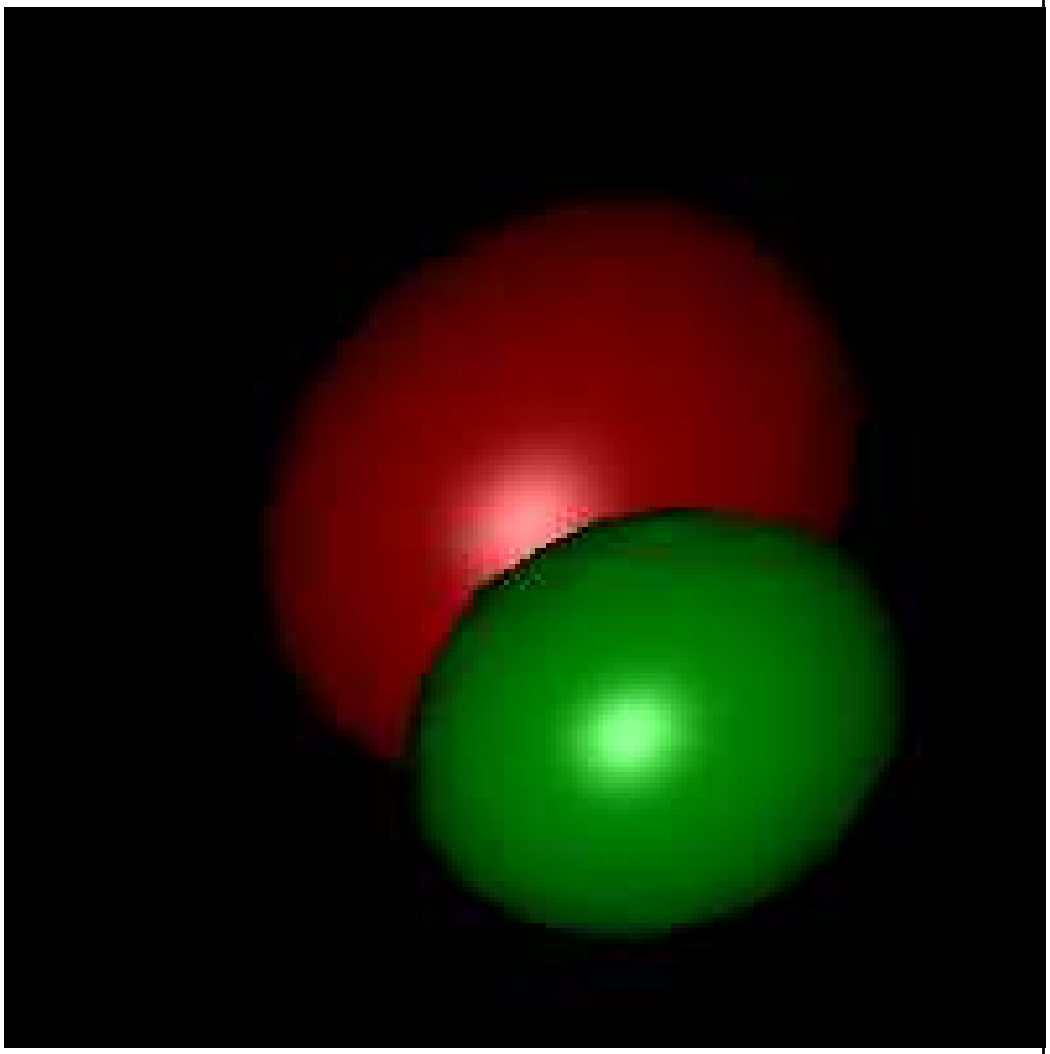


$K_a=K_d=0.5$, $K_s=0.6$, $n=10$

グローシェーディング



フォンシェーディング



K_a=K_d=0.5, K_s=0.6, n=10