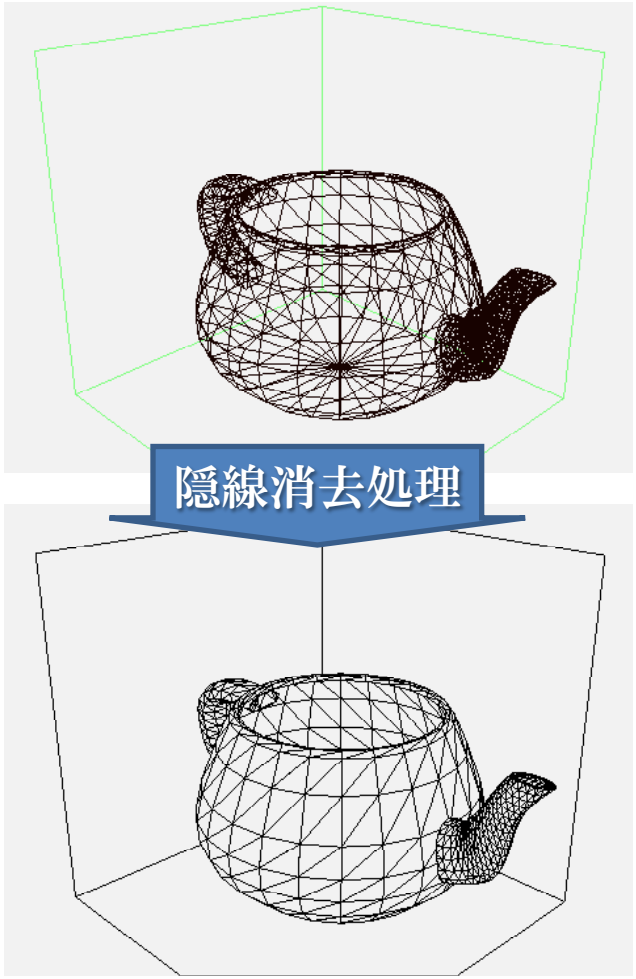
コンピュータグラフィックス

第9回：見えない面を隠す

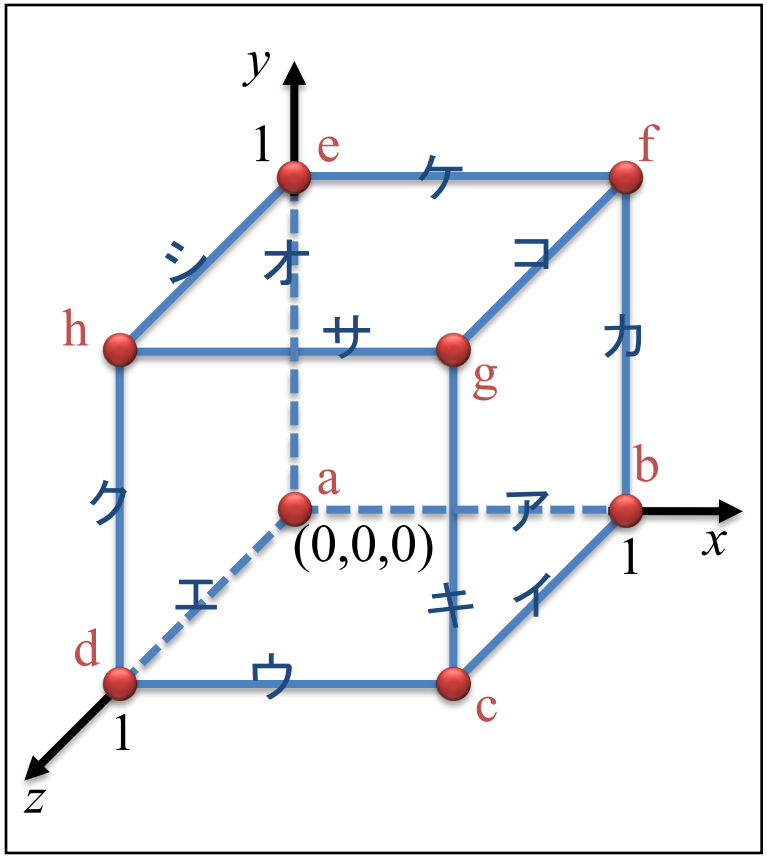
隠線消去処理・隠面消去処理



隠線消去処理

隠面消去処理（＋陰影付け）

サーフェースモデルの形状データ



頂点表

頂点	座標値		
	x	*y*	*z*
a	0	0	0
b	1	0	0
c	1	0	1
d	0	0	1
e	0	1	0
f	1	1	0
g	1	1	1
h	0	1	1

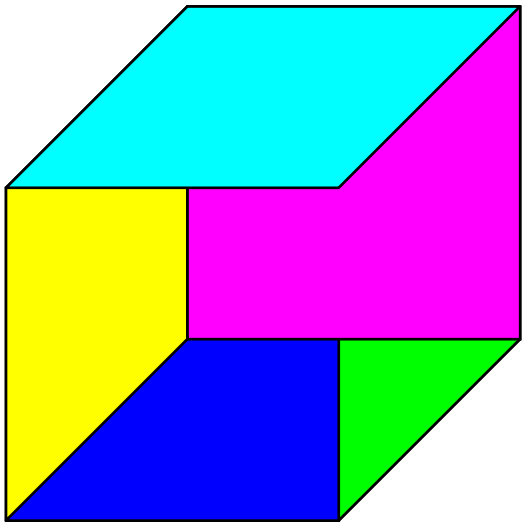
面データ

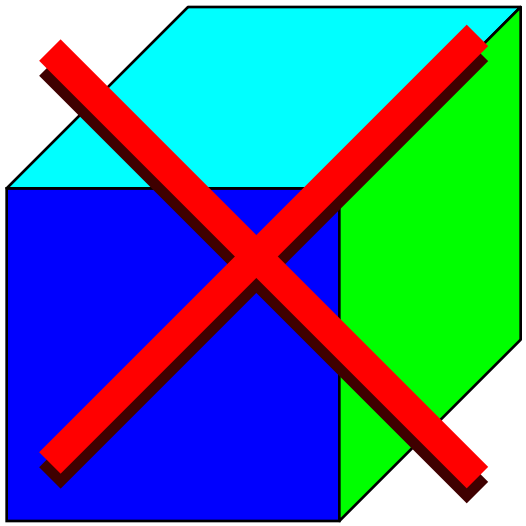
面	頂点				色
下	a	b	c	d	
右	b	f	g	c	
前	c	g	h	d	
左	d	h	e	a	
後	a	e	f	b	
上	h	g	f	e	

（頂点リスト形式）

面データの順に面を描く

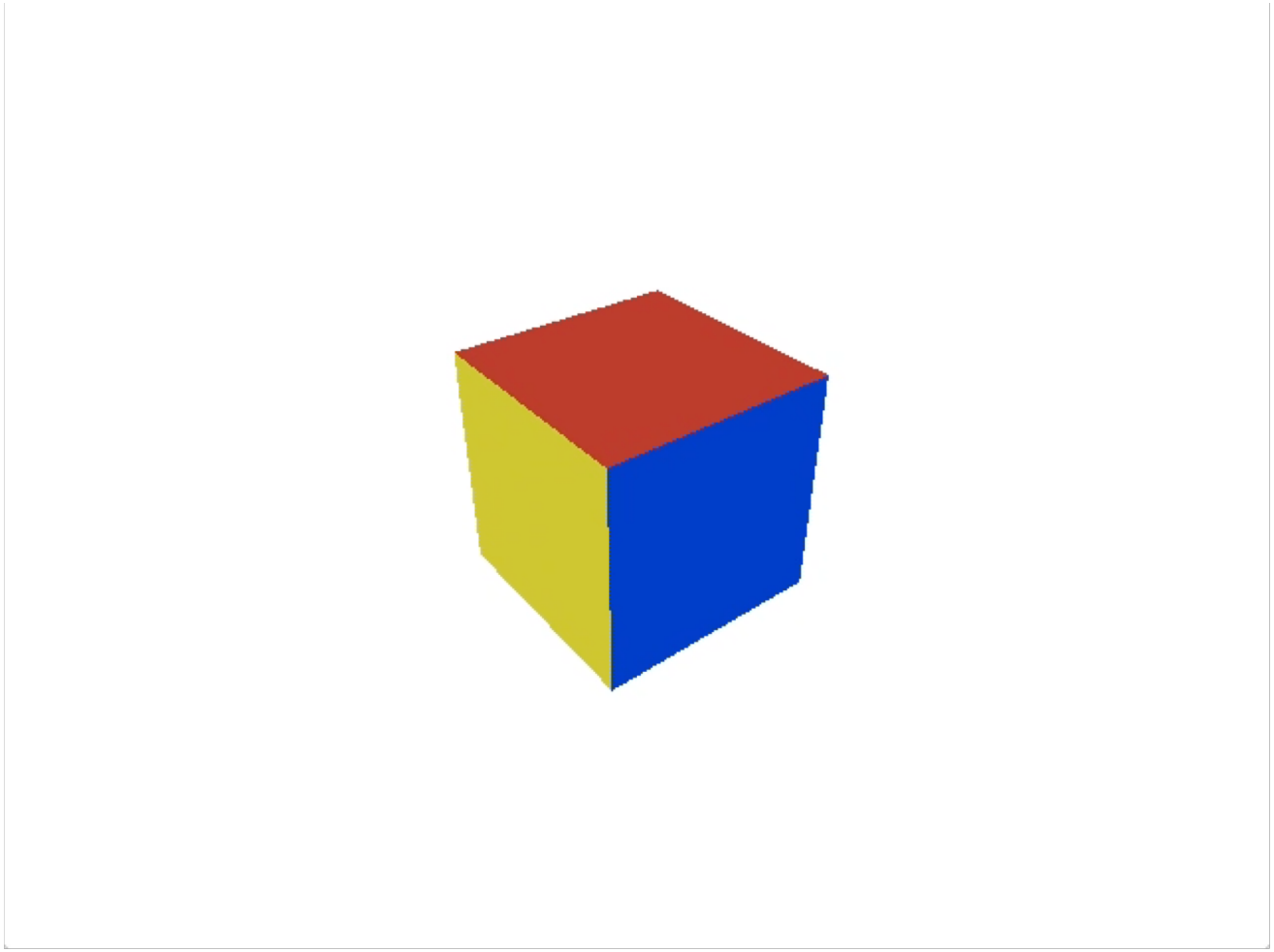
下 → 右 → 前 → 左 → 後 → 上 の順に描く



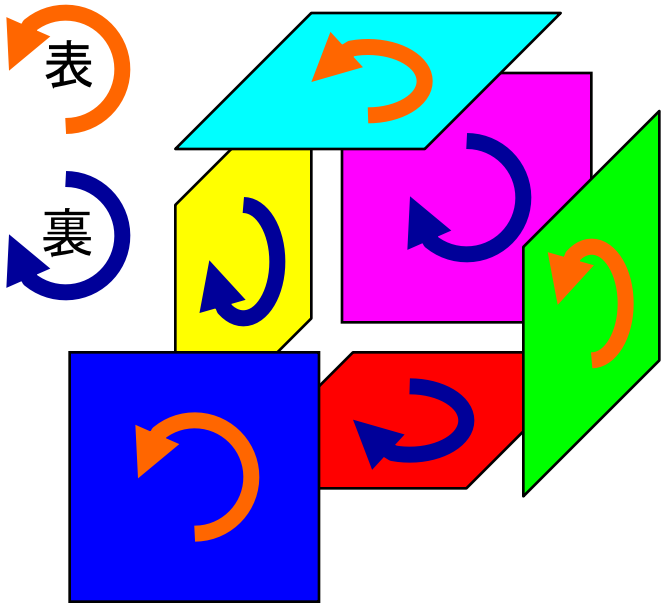


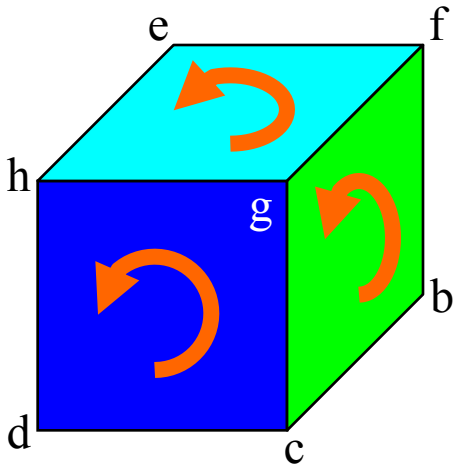
こうはならない

面データの順に面を描いた場合



背面ポリゴンの除去



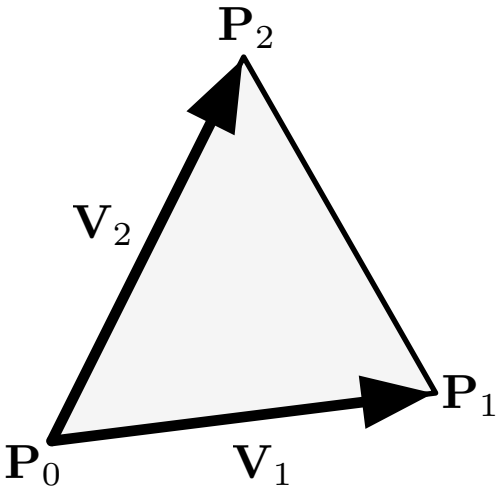


面データ

面	頂点				色
下	a	b	c	d	
右	b	f	g	c	
前	c	g	h	d	
左	d	h	e	a	
後	a	e	f	b	
上	h	g	f	e	

- 閉じた図形では背面（左，下，後）のポリゴンは見えない
  - したがって，このようなポリゴンは描く必要が無い
- 背面のポリゴンは視点に対して裏側を向けている
- ポリゴンの表裏は面データの頂点をたどる順序で与える
  - たとえば，左回りのときに「表」というように

三角形の表裏判定



$$\mathbf{V}_1 = \mathbf{P}_1 - \mathbf{P}_0 = (dx_1, dy_1, 0)$$

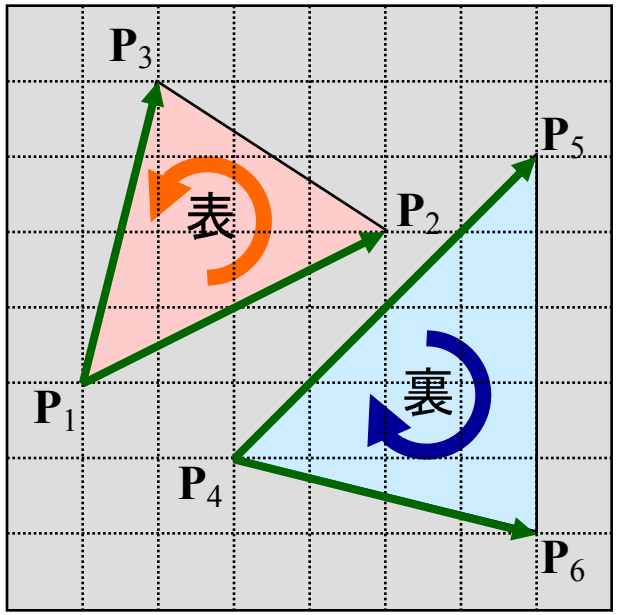
$$\mathbf{V}_2 = \mathbf{P}_2 - \mathbf{P}_0 = (dx_2, dy_2, 0)$$

$$\mathbf{V}_1 \times \mathbf{V}_2 = (dx_1, dy_1, 0) \times (dx_2, dy_2, 0) = \left(0, 0, \underline{dx_1 dy_2 - dx_2 dy_1}\right)$$

左手系なら

$$dx_1 dy_2 - dx_2 dy_1 \begin{cases} > 0 : \text{表を向いている} \\ = 0 : \text{つぶれている（見えない）} \\ < 0 : \text{裏を向いている} \end{cases}$$

三角形の表裏判定の例



スクリーン

頂点	座標値	
	x	y
P_1	1	3
P_2	5	5
P_3	2	7
P_4	3	2
P_5	7	6
P_6	7	1

透視変換後の
頂点座標値

面	頂点			色
左	P_1	P_2	P_3	
右	P_4	P_5	P_6	

表から見たとき左回りに
頂点をたどった面データ

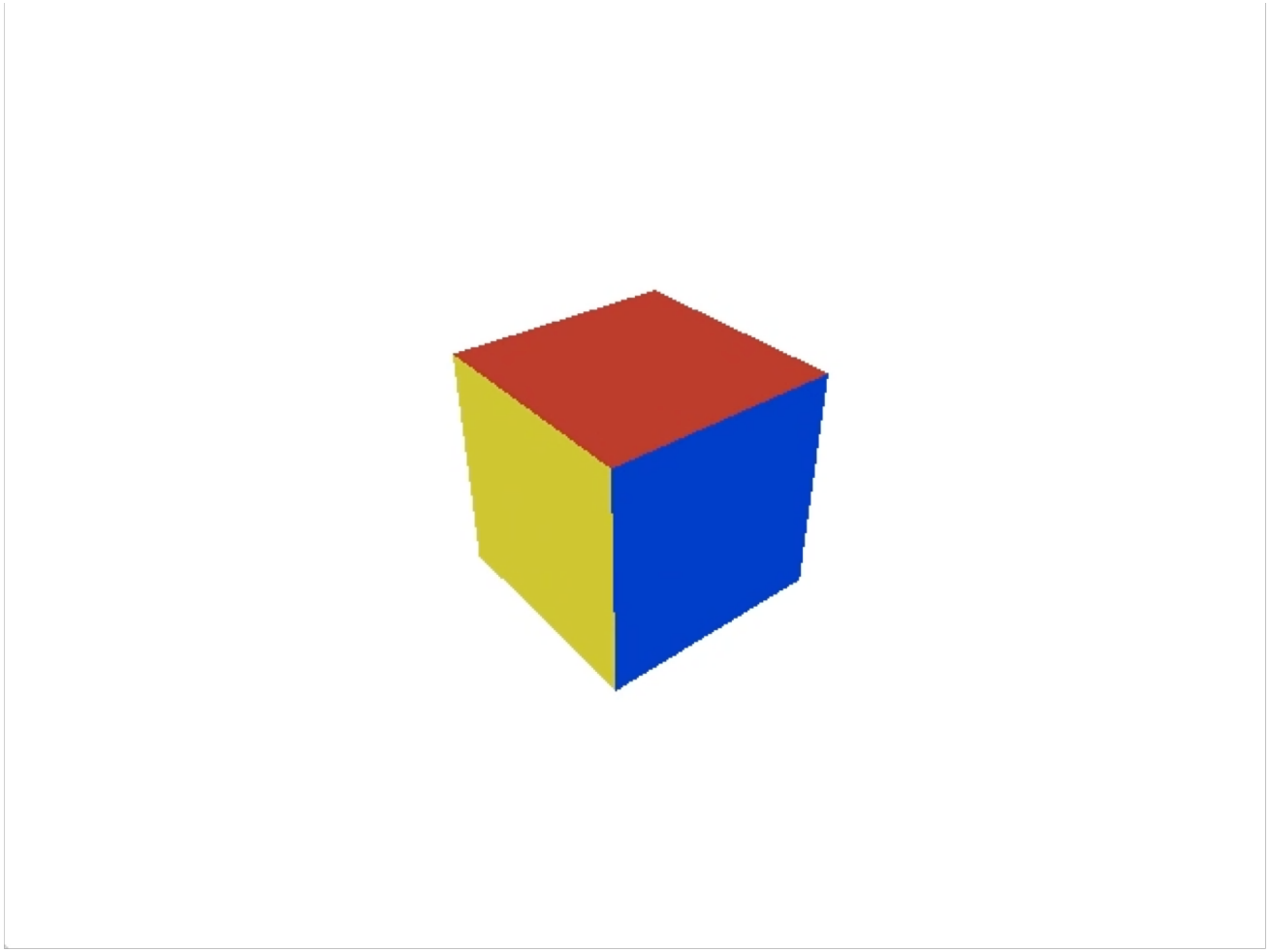
$$(P_2 - P_1) \times (P_3 - P_1)$$
$$= (5-1, 5-3) \times (2-1, 7-3)$$
$$= (4, 2) \times (1, 4)$$
$$= (0, 0, 4 \cdot 4 - 2 \cdot 1)$$
$$= (0, 0, \underline{14})$$

表

$$(P_5 - P_4) \times (P_6 - P_4)$$
$$= (7-3, 6-2) \times (7-3, 1-2)$$
$$= (4, 4) \times (4, -1)$$
$$= (0, 0, 4 \cdot (-1) - 4 \cdot 4)$$
$$= (0, 0, \underline{-20})$$

裏

背面ポリゴンを除去した場合



背面ポリゴン除去の限界

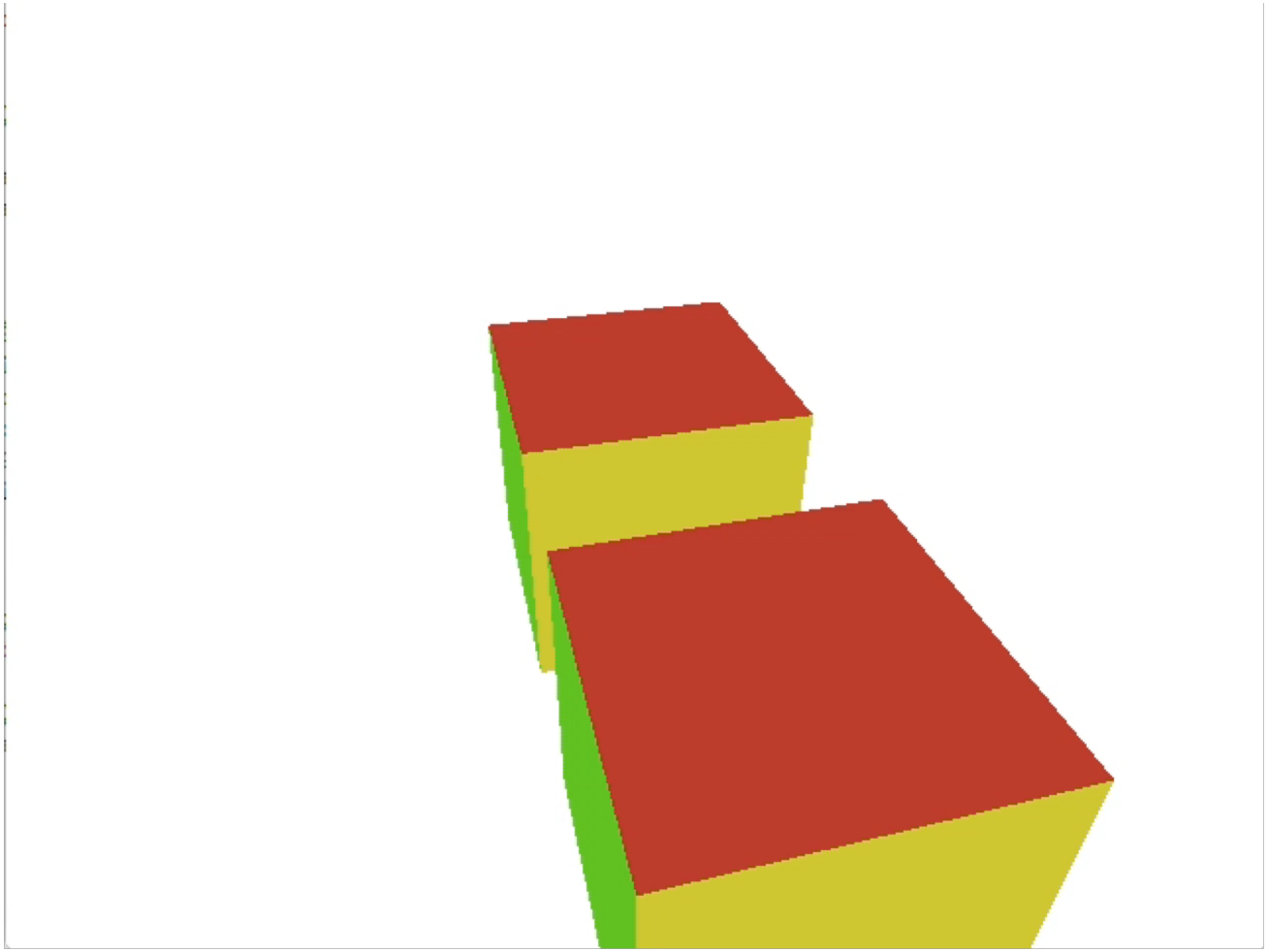
- 複数の物体の重なり合いに対応できない

こうなる可能性がある

- 凹部を含む多面体に対応できない

こうなる可能性がある

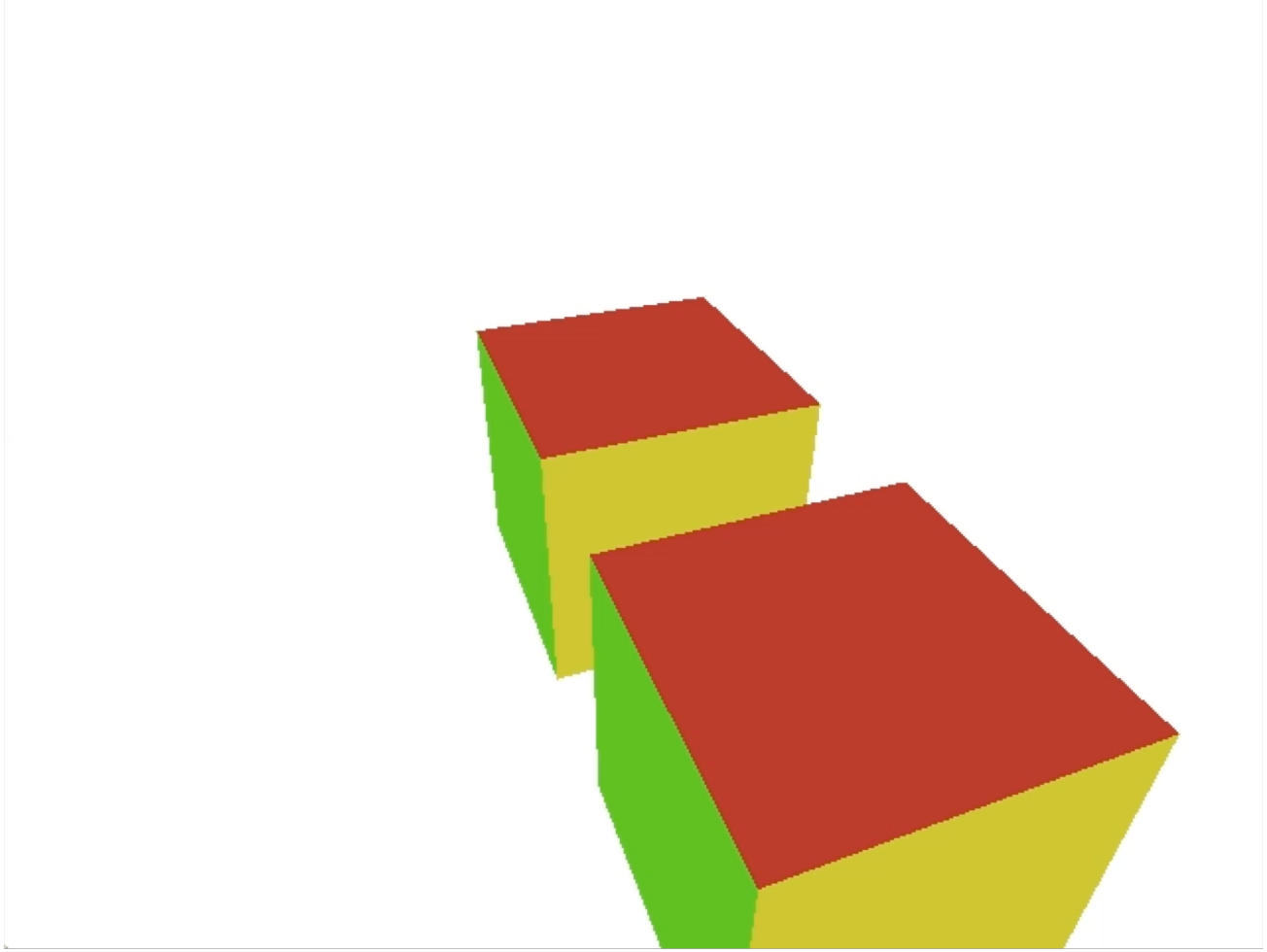
複数の物体がある場合



隠面消去処理

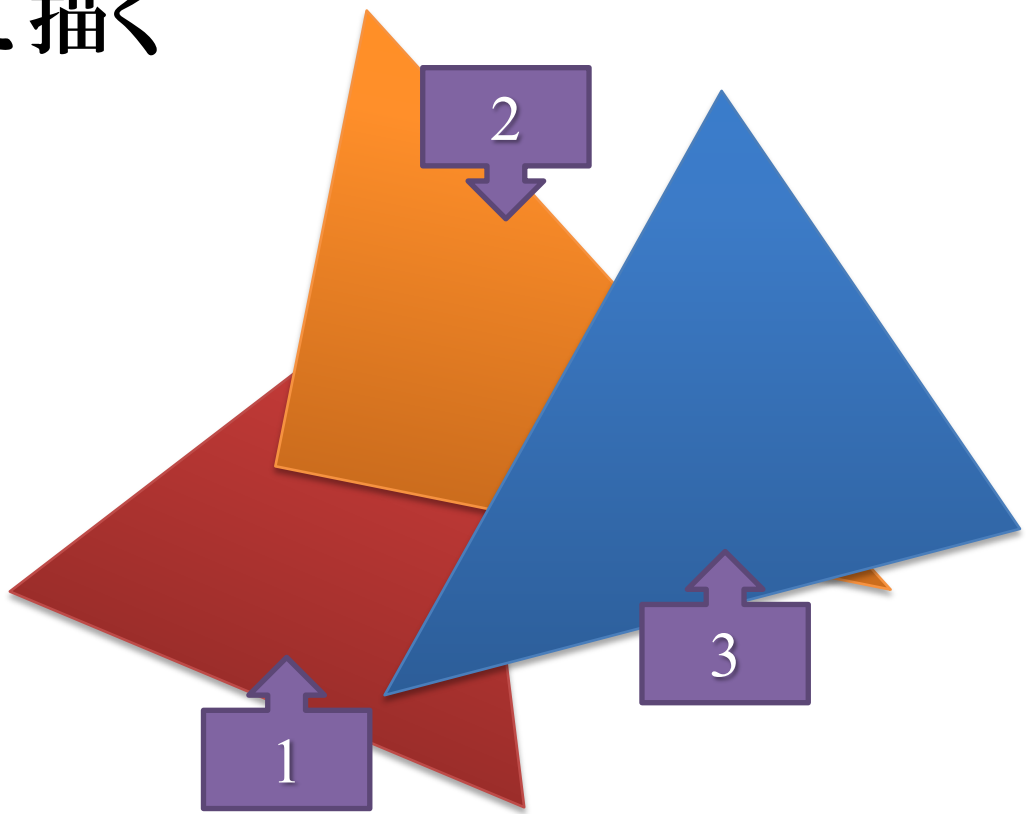
- デプスソート法
  - Zソート法，ペインタアルゴリズム
- 分割統治法
  - Warnock のアルゴリズム
- スキャンライン法
  - Watkins のアルゴリズム
- デプスバッファ法
  - Edwin Catmull

隠面消去処理を行った場合



デプスソート法

- ポリゴンを深度（デプス，奥行き，Z値）の順に並べ替えて描く



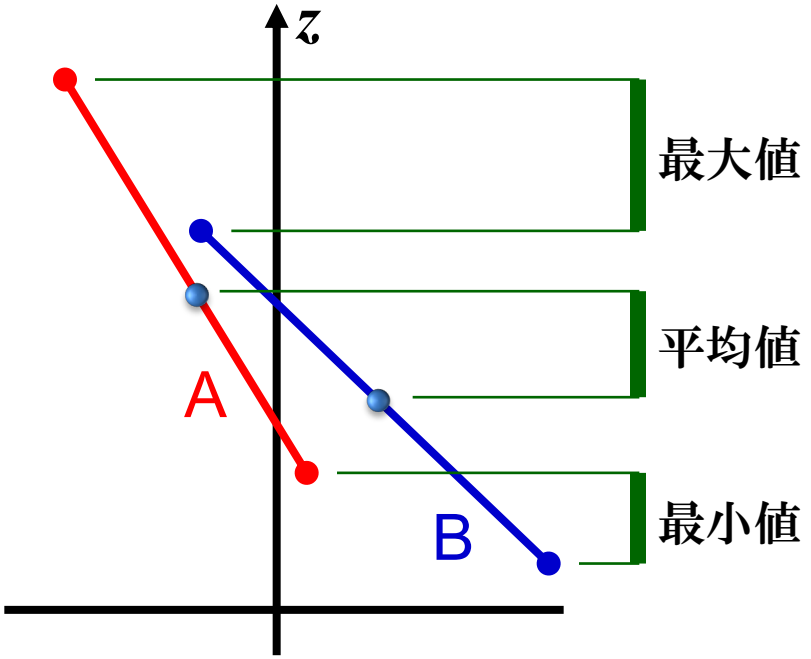
デプスソート法の問題点

- ポリゴンの順序を一意に決められない
  - 深度を比較するポリゴン上の点はいくつもある
    - 最小値・最大値・平均値，あるいは点と面の比較
  - 3次元空間内において順序は一意に決まらない
    - 3すくみ，凹部を含むポリゴンなど
  - 交差しているポリゴンを処理できない
    - 交差線を算出する必要がある
- 適当な位置でポリゴンを分割する必要がある
  - 分割する場所を決めるのは結構難しい問題

深度の代表値

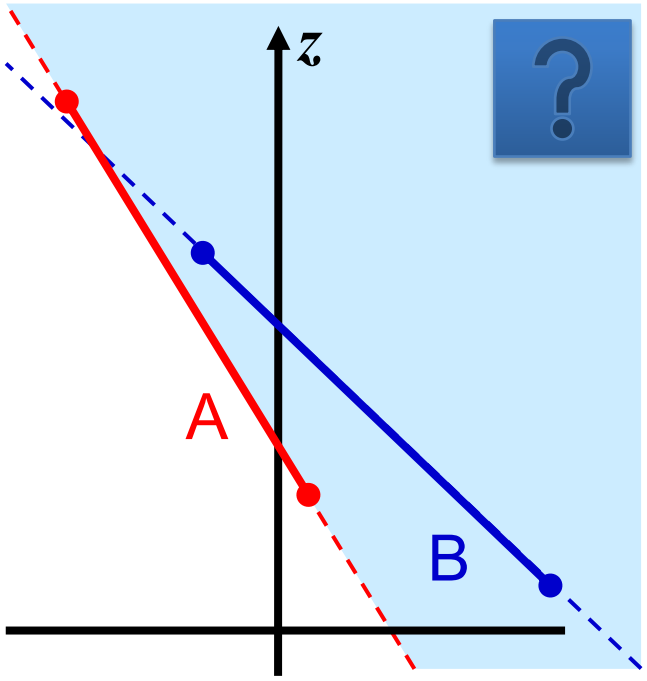
深度の比較では判定できない場合（AよりBを先に描く必要がある）

頂点の深度による比較



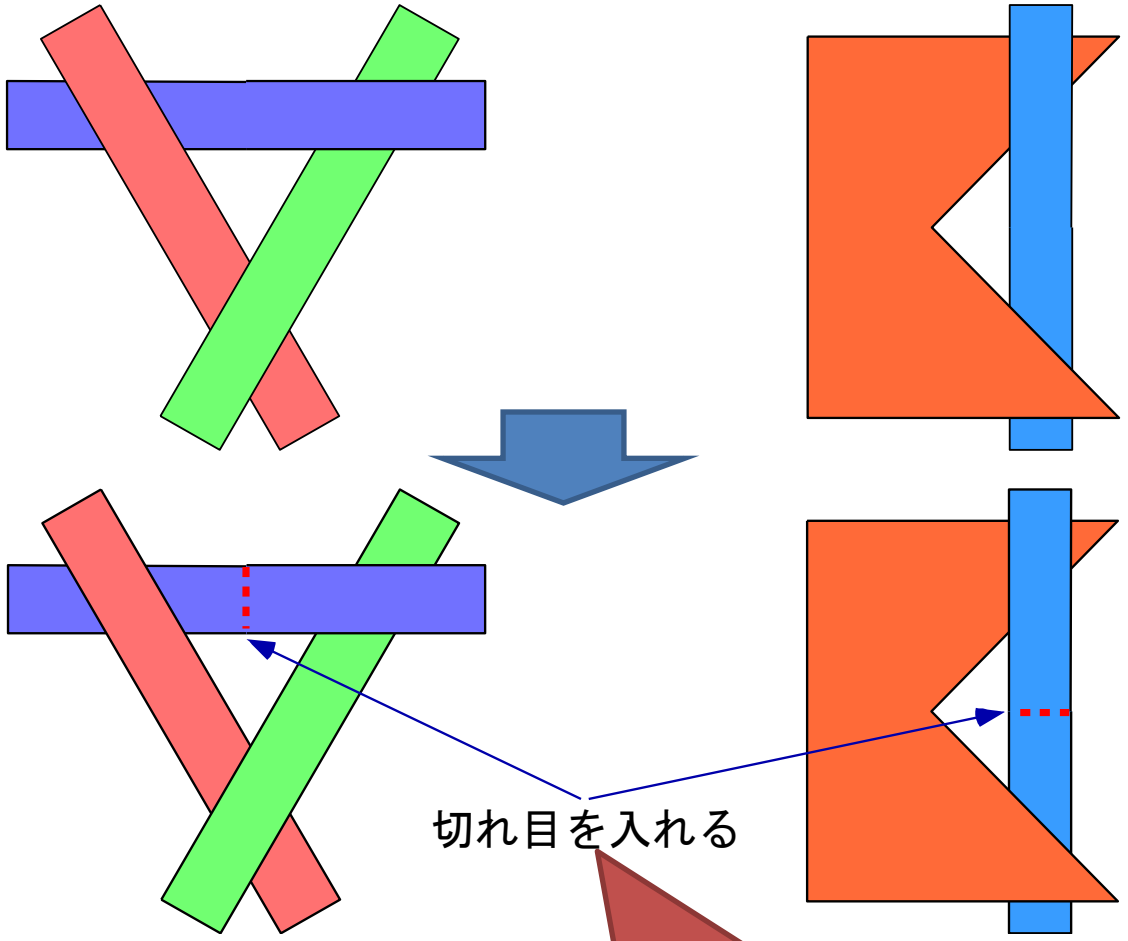
いずれで比較してもBの方が小さい
（Aを先に描くことになってしまう）

点と平面の関係による比較



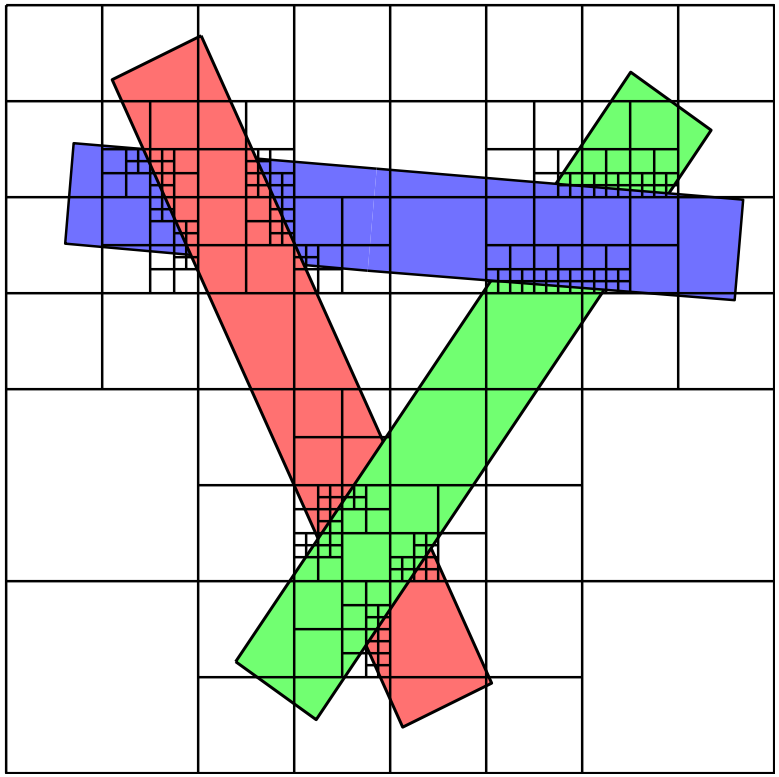
Bの頂点はすべてAを含む平面の
背後にある（Bを先に描く）

順序が一意に決まらない場合



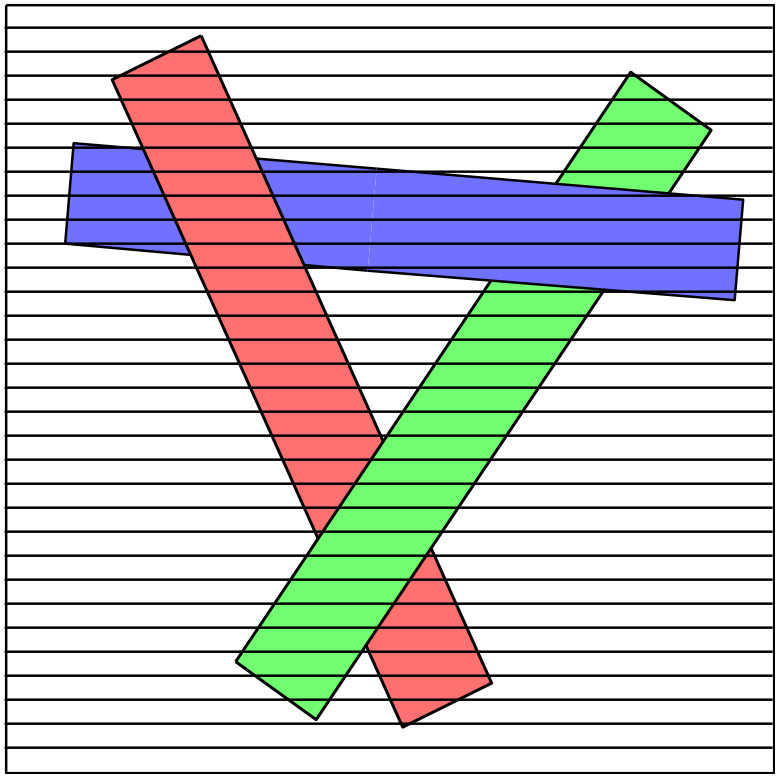
分割統治法

- 領域内で見えるポリゴンが容易に判定できるようになるまで領域を再帰的に分割
  - Warnock のアルゴリズム
    - Dr. John E. Warnock – 1982 に Adobe 設立

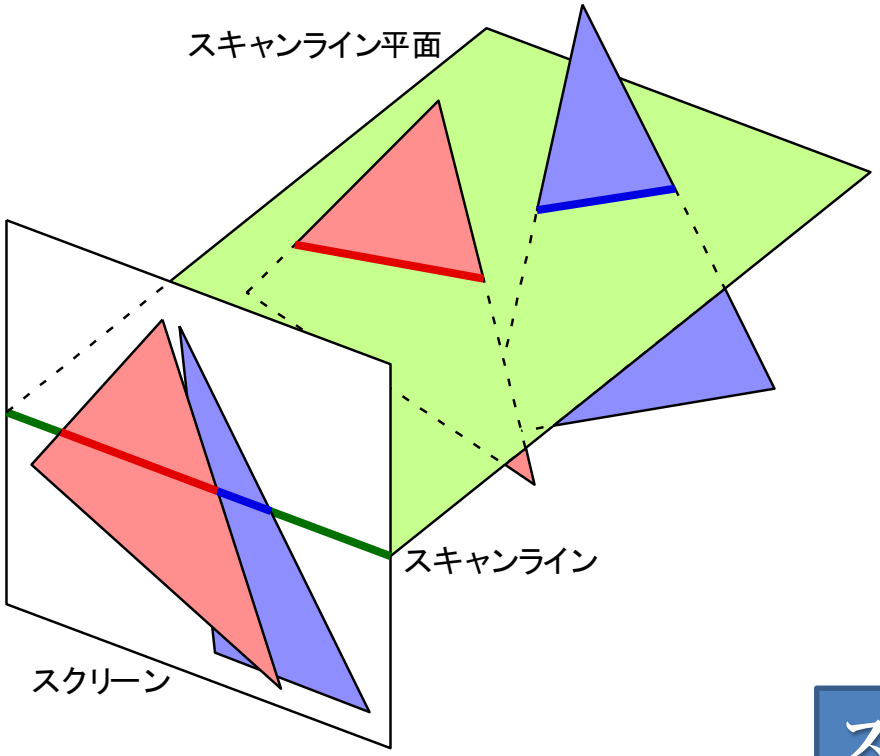


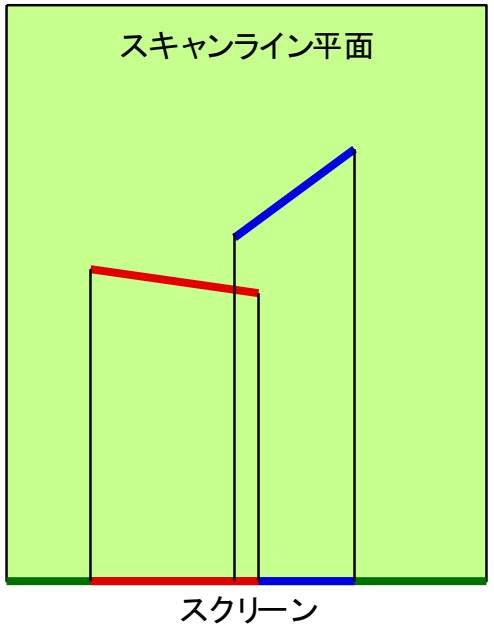
スキャンライン法

- 画面をスキャンライン（走査線）単位に分割
  - 可視判定をスキャンライン平面上の問題に帰着する
  - Watkins のアルゴリズム
    - スキャンライン上で分割統治法を用いる



スキャンライン平面上の処理

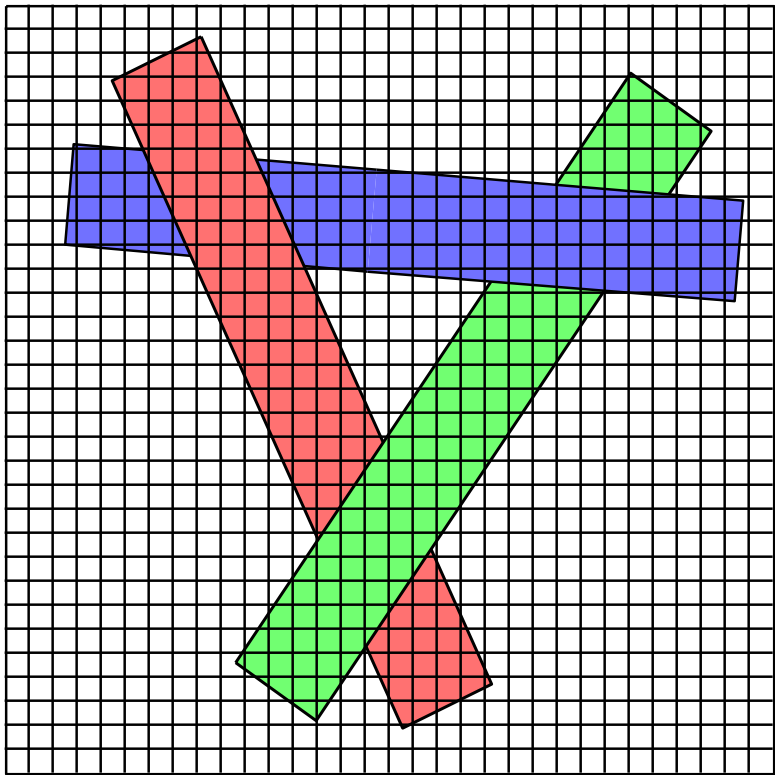




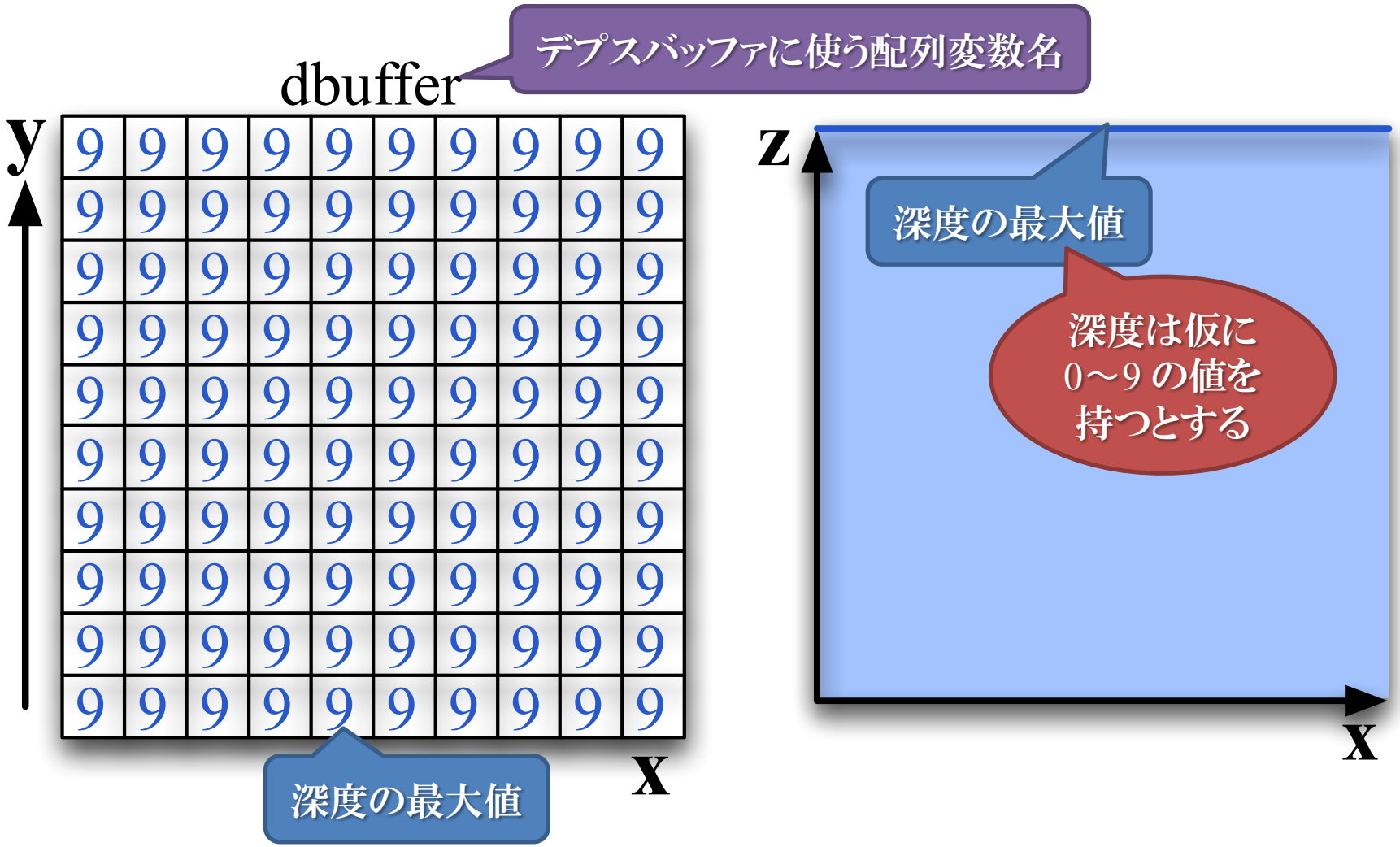
スキャンライン平面上で線分（ポリゴンとの交差線）を比較

デプスバッファ法

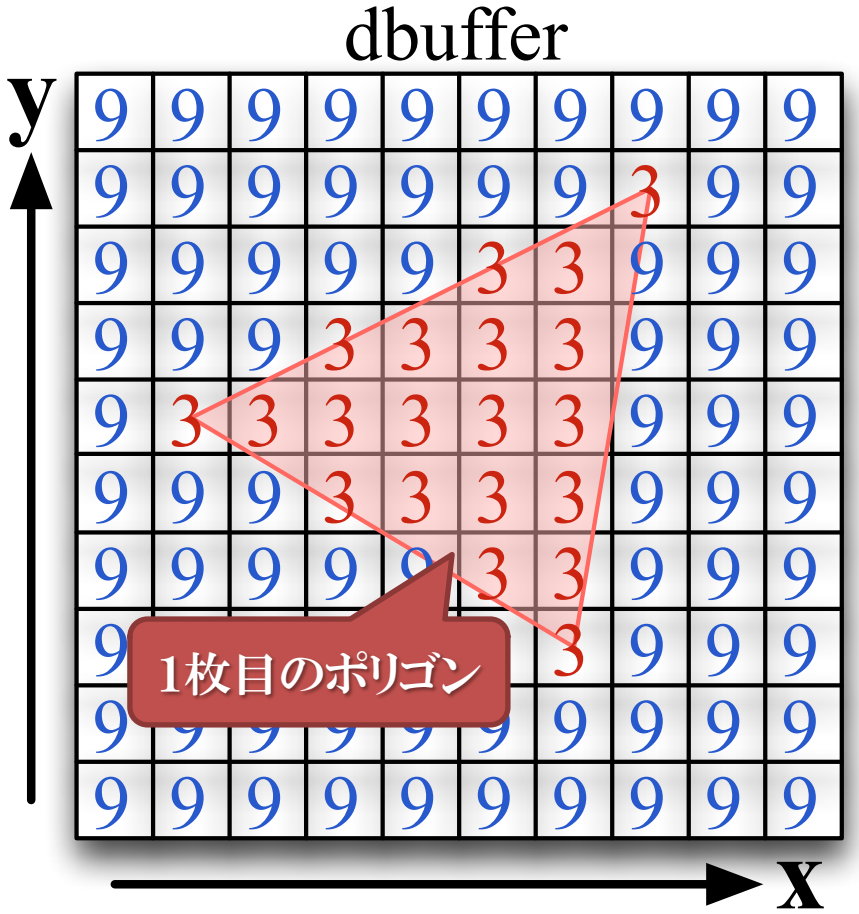
- 画面をピクセル（画素）単位に分割
  - ポリゴンの描画時に画素単位に深度を比較する
  - Edwin Catmull
    - ピクサーを設立，現在はウォルト・ディズニー・アニメーション・スタジオ及びピクサー・アニメーション・スタジオ社長

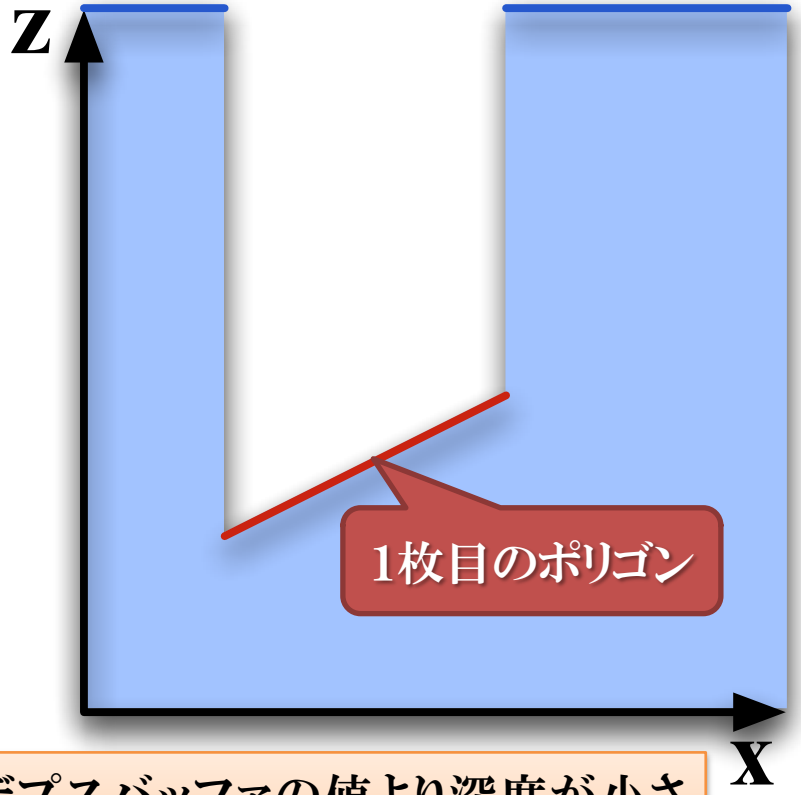


デプスバッファの初期化



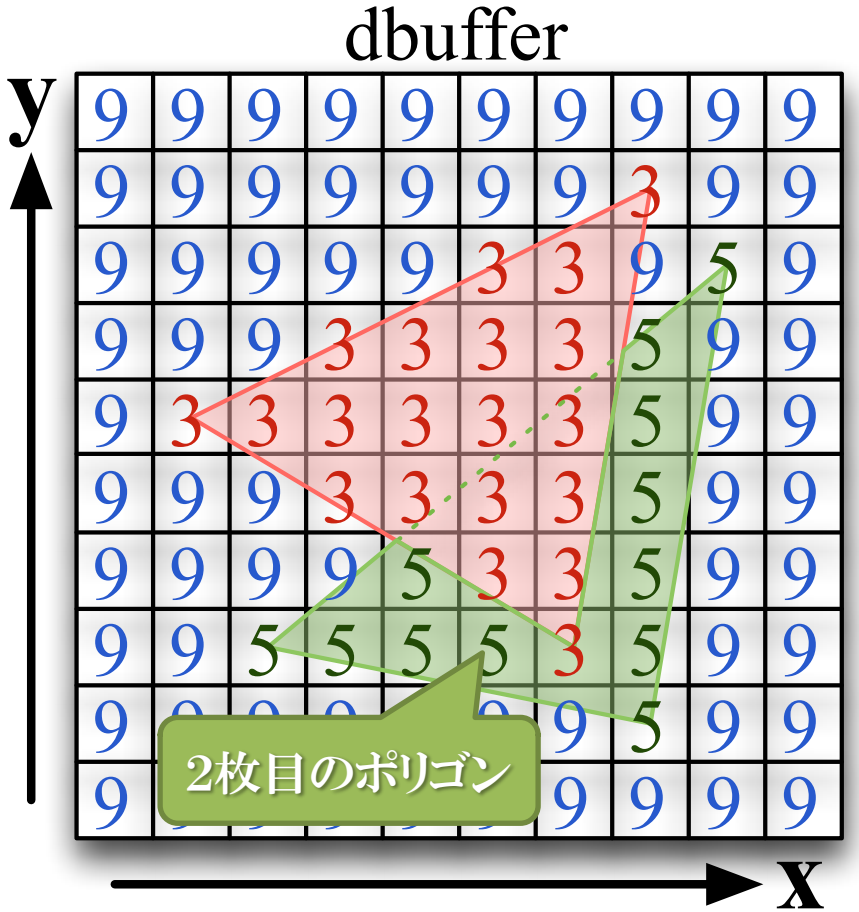
1枚目のポリゴンの描画

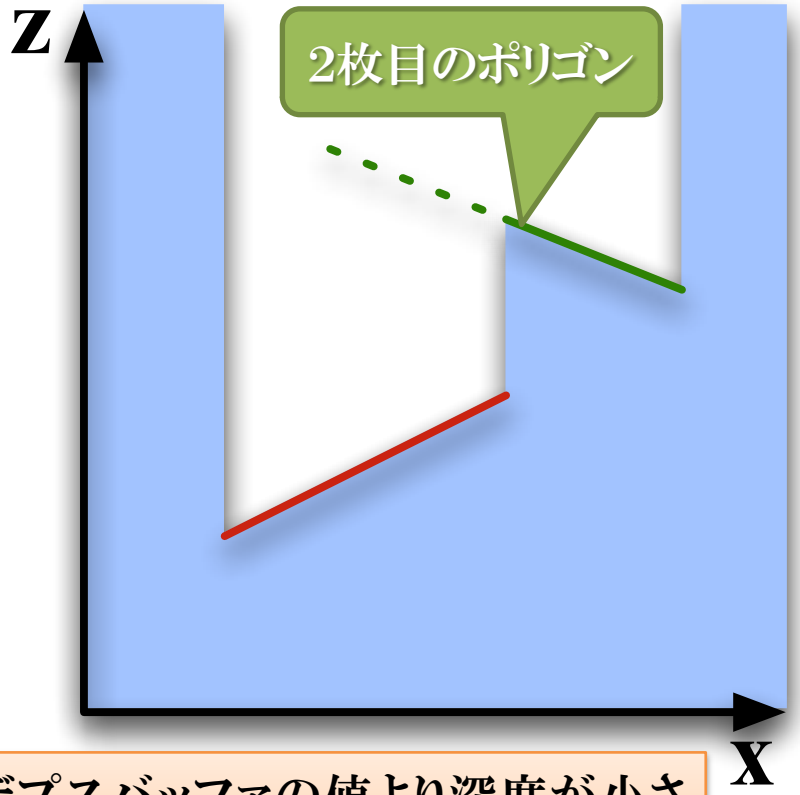




デプスバッファの値より深度が小さければその画素の色を表示し，デプスバッファの内容をその深度にする

2枚目のポリゴンの描画

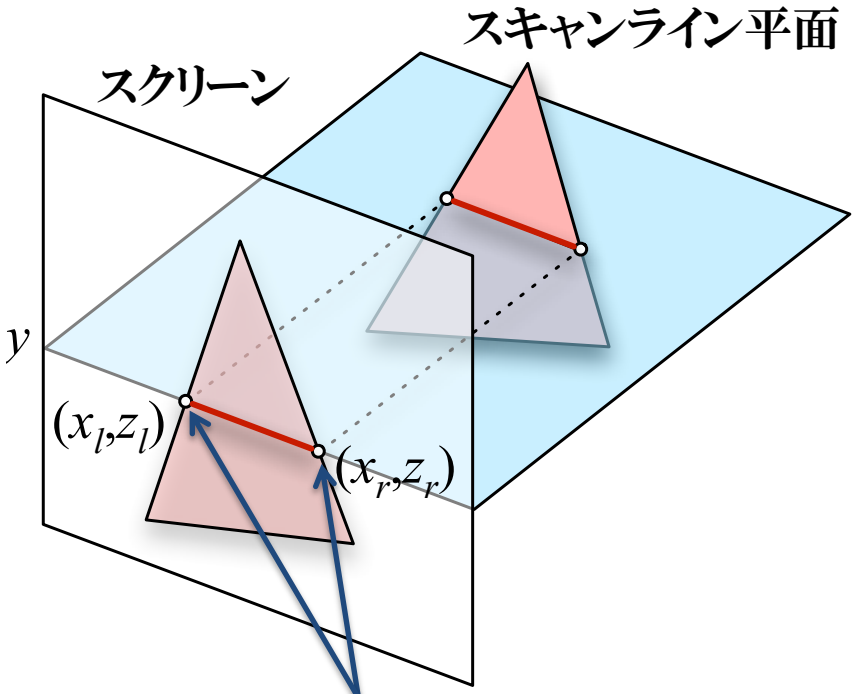




デプスバッファの値より深度が小さければその画素の色を表示し，デプスバッファの内容をその深度にする

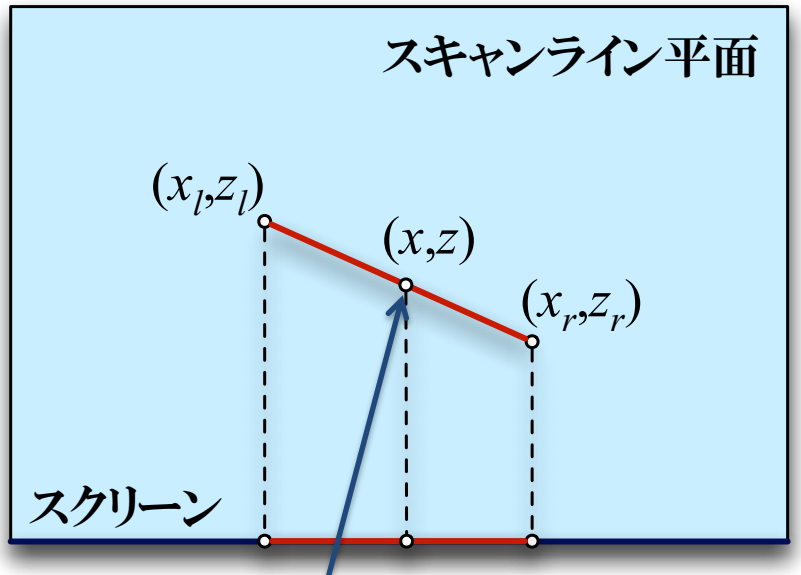
三角形描画プログラムの3次元化

三角形の描画



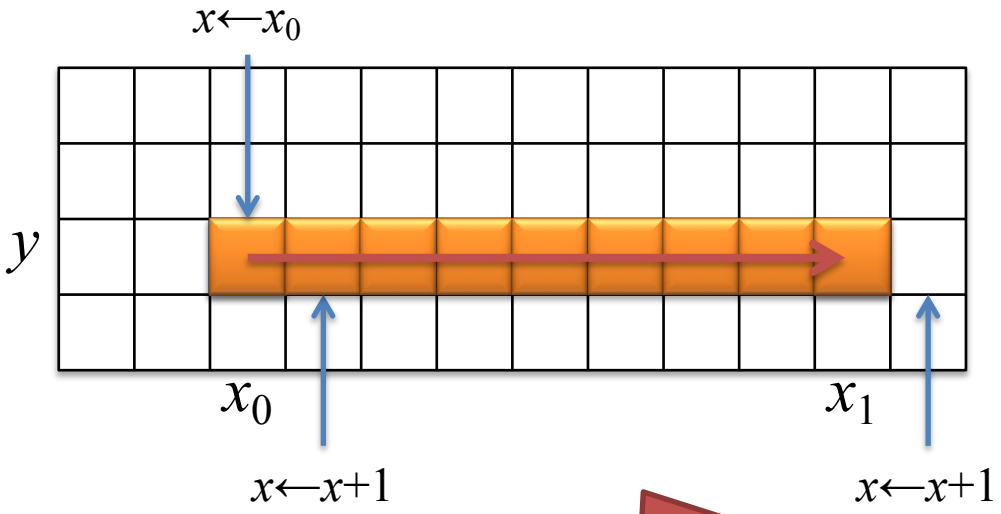
yに対してx値と同時にz値も求める

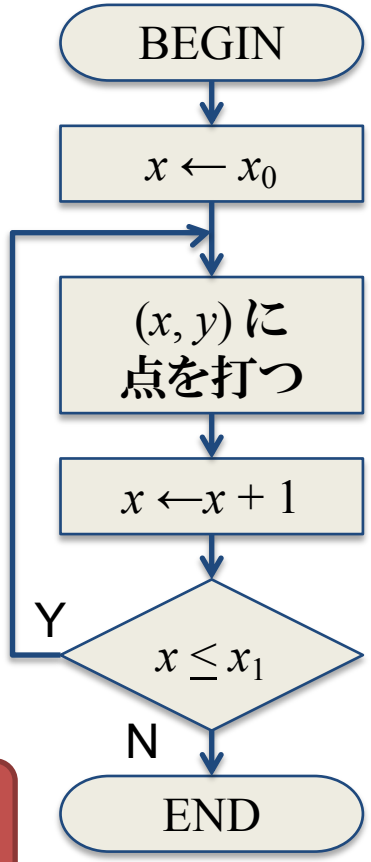
水平線の描画



xに対するz値を求める

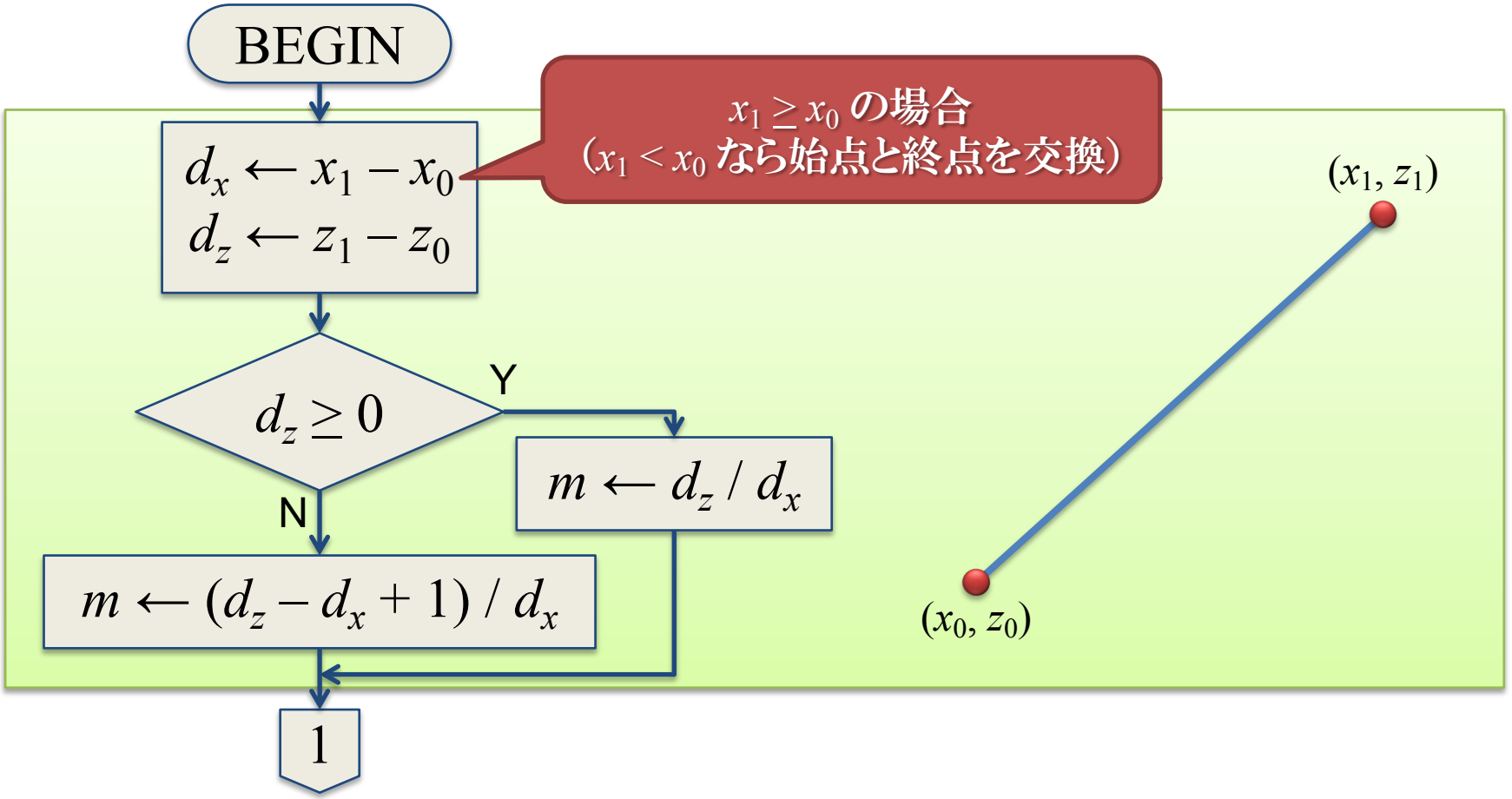
水平線の描画の3次元化



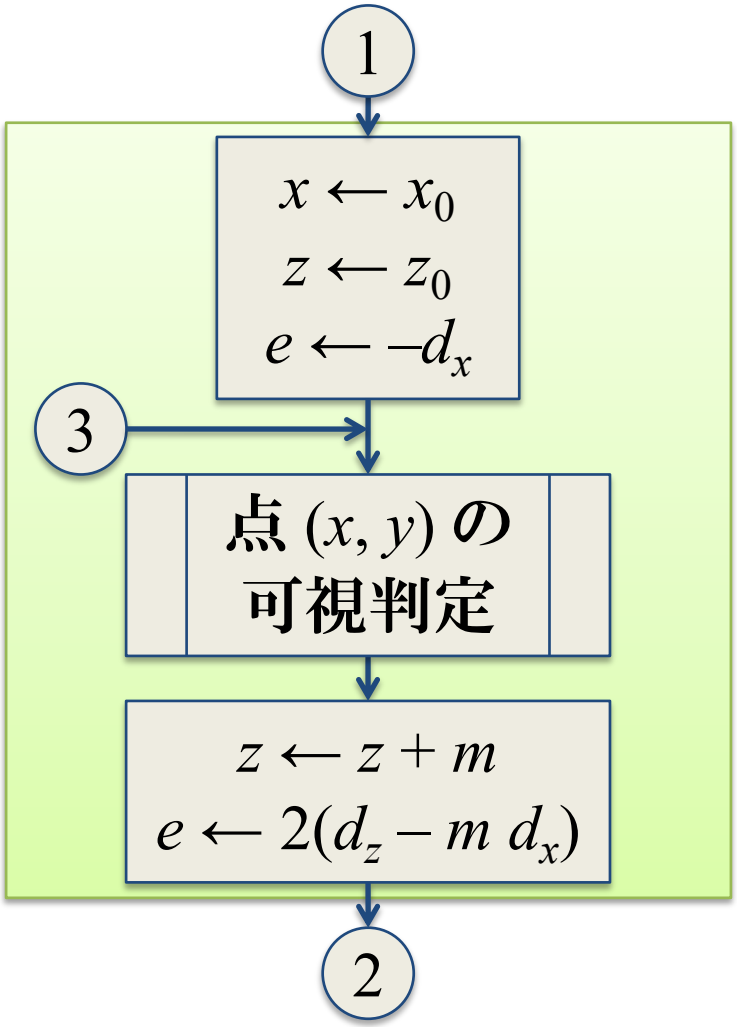


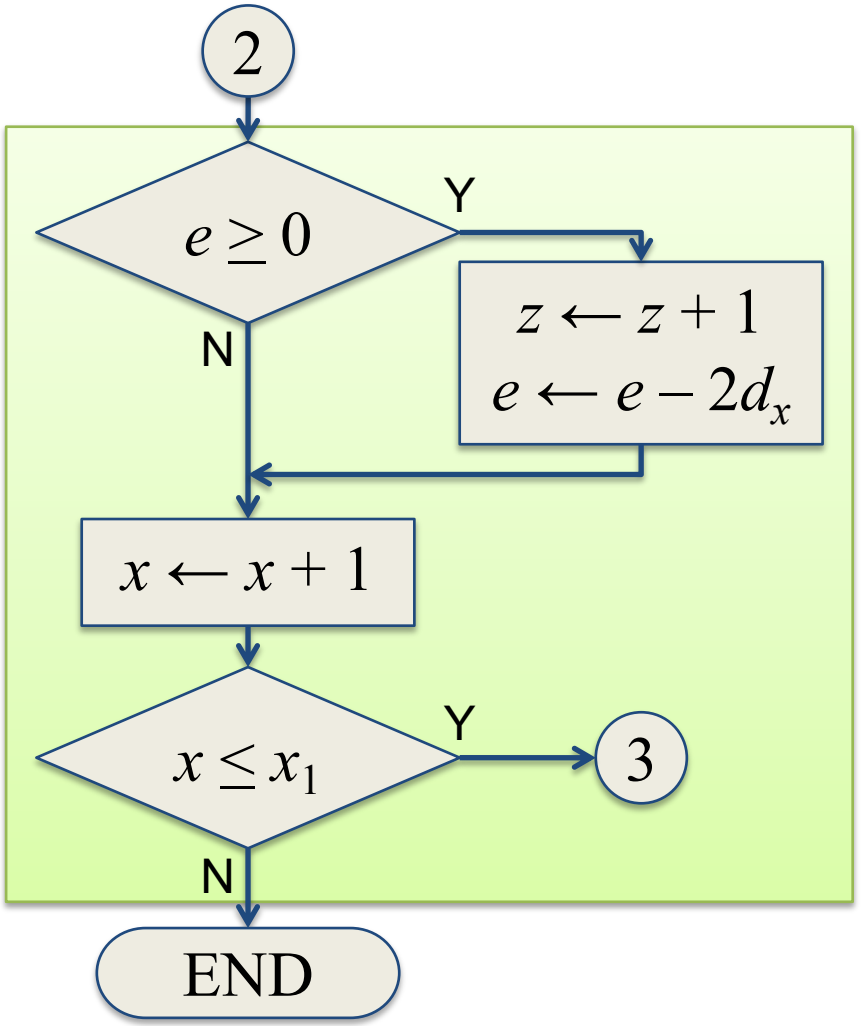
この処理の際にx値に対するz値を求める

xに対するz値の算出 (1)

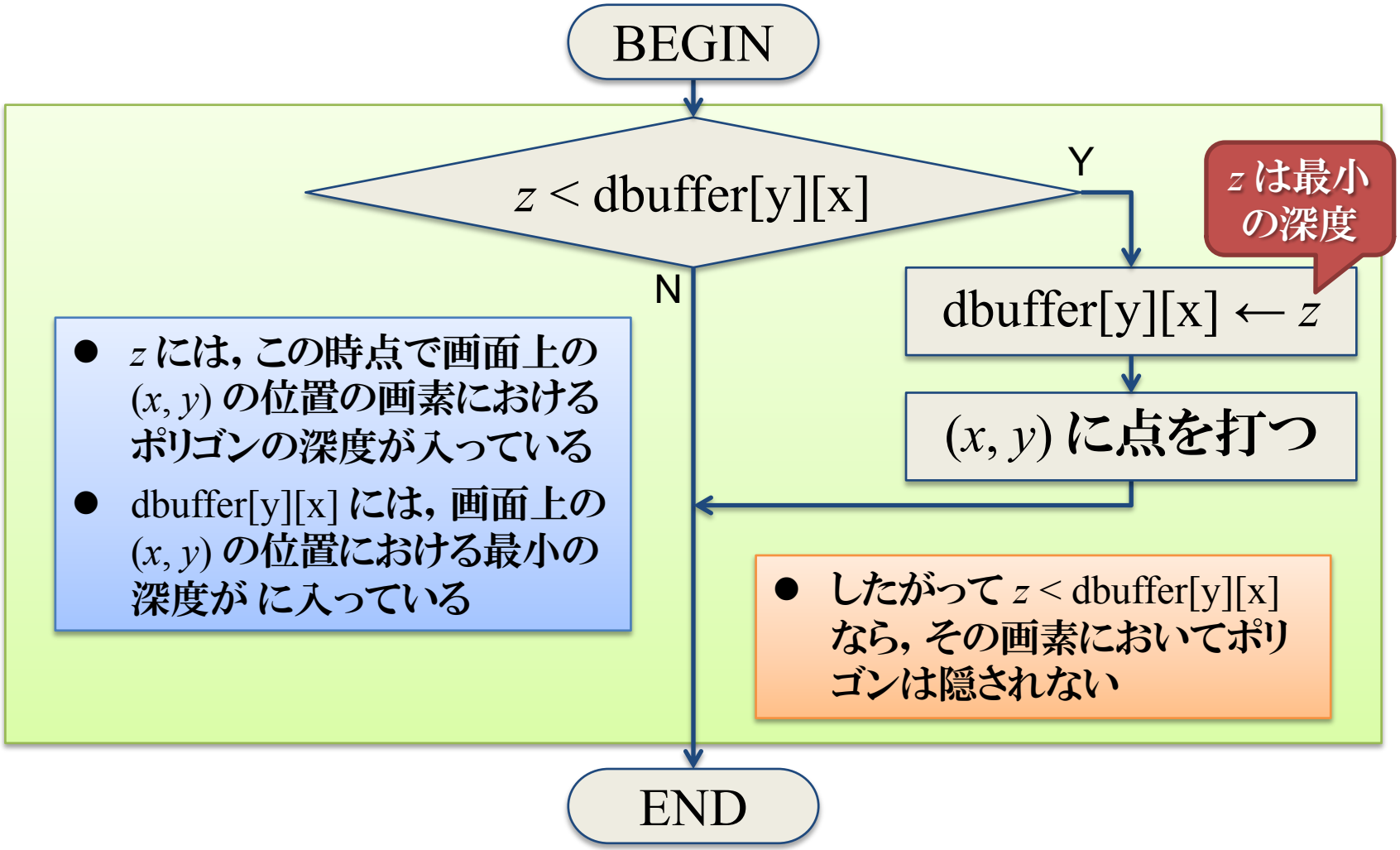


xに対するz値の算出 (2)

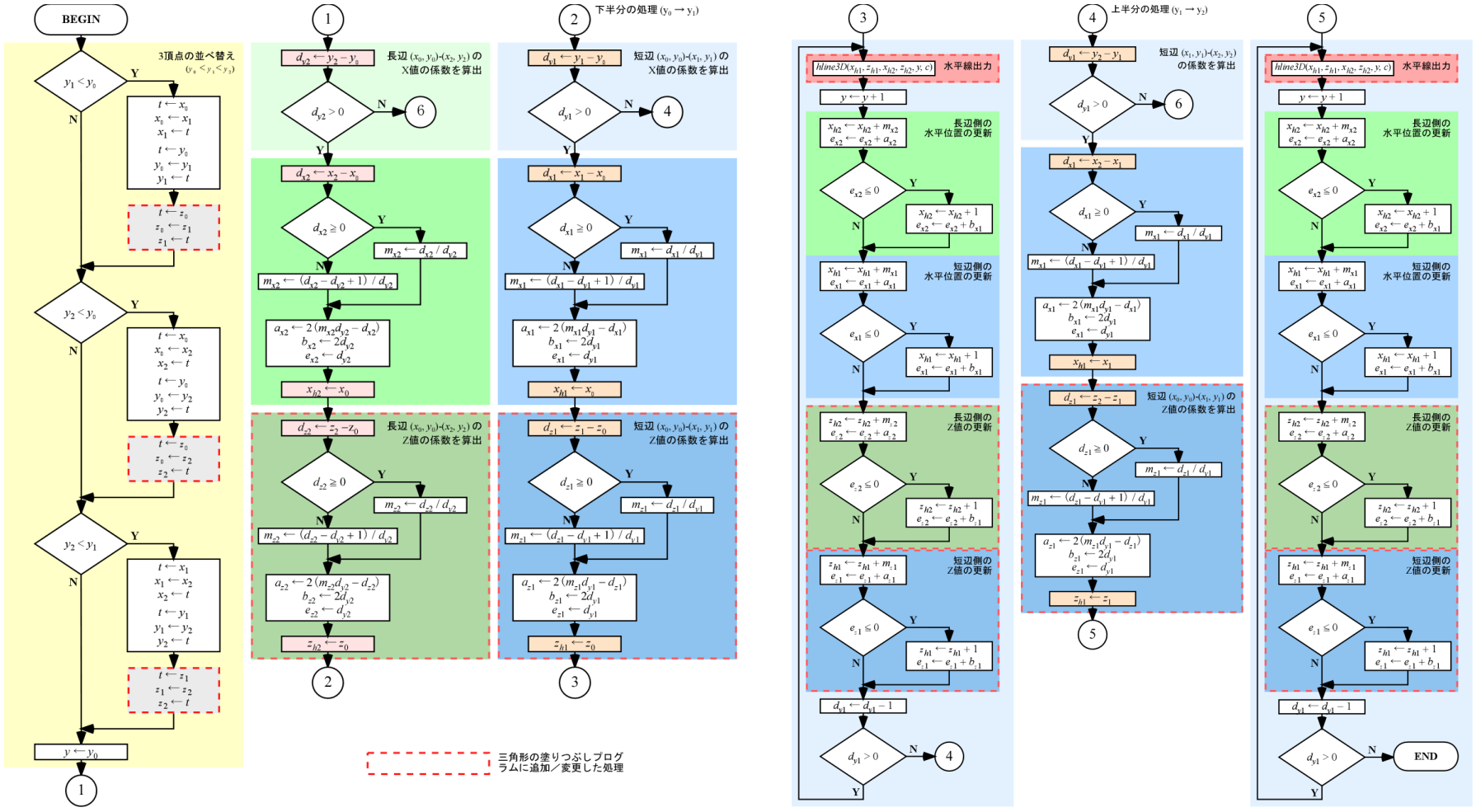




点 (x, y) の可視判定


BEGIN
z < dbuffer[y][x]
Y
N
z は最小の深度
dbuffer[y][x] ← z
(x, y) に点を打つ
z には，この時点で画面上の (x, y) の位置の画素におけるポリゴンの深度が入っている
dbuffer[y][x] には，画面上の (x, y) の位置における最小の深度が に入っている
したがって z < dbuffer[y][x] なら，その画素においてポリゴンは隠されない
END

三角形描画アルゴリズムの変更点



おわり