CG 基礎、数学的準備

情報可視化論 第 03 回

陰山 聡

神戸大学 システム情報学研究科 計算科学専攻
［情報基盤センター分館 第 1 演習室］

2015.04.28

環境設定

- ログイン確認
- WebGL 環境オン
  - Safari 立ち上げ
  - 環境設定
  - → メニューバーに開発メニューを表示 にチェック
  - （開発メニューから “WebGL を有効”）
- 確認
  - この講義のウェブページ
  - `http://tinyurl.com/kageyama2015a`
  - サンプルをクリック

前回の復習

WebGL とは

WebBL = シェーダを使い、HTML5 の canvas に、JavaScript で 3D CG を書くための API

WebGL の特徴

- スタンドアロンアプリからウェブアプリへの流れ
- クロスプラットフォーム
- オープンスタンダード
- Web で GPU を使ったレンダリングが可能
- 開発・利用が容易： プラグイン不要
- ソースコードが見える
- グラフィックス（OpenGL）と UI（ウィンドウ管理やイベント処理）の分離が明白

コンピュータグラフィックス（CG）

CG と並列計算

並列計算については学んできたはず。

空間を分割して、通信用のメモリを確保して、MPI で通信して ···

並列プログラム作りは結構大変。← 結局同期が大変だから。

一方、同期について気にしなくてもよい問題もたくさんある。もともとのアルゴリズムが並列化可能なもの。→ “Embarrassingly” parallel problems

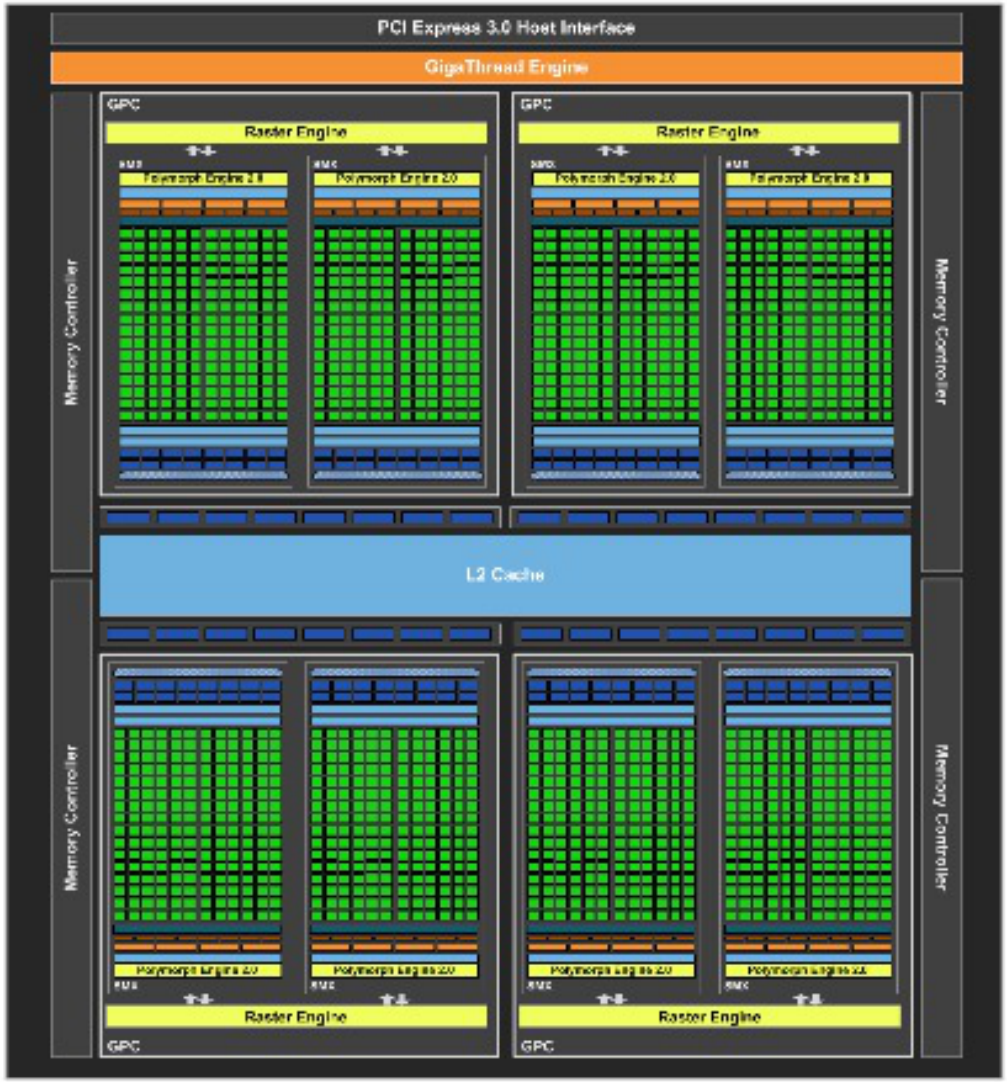
CG の処理の多くが embarrassingly parallel

座標変換、テクスチャマップ、シェーディング、ラスタ化、etc.

CG 専用並列計算機 → GPU (Graphics Processing Unit)

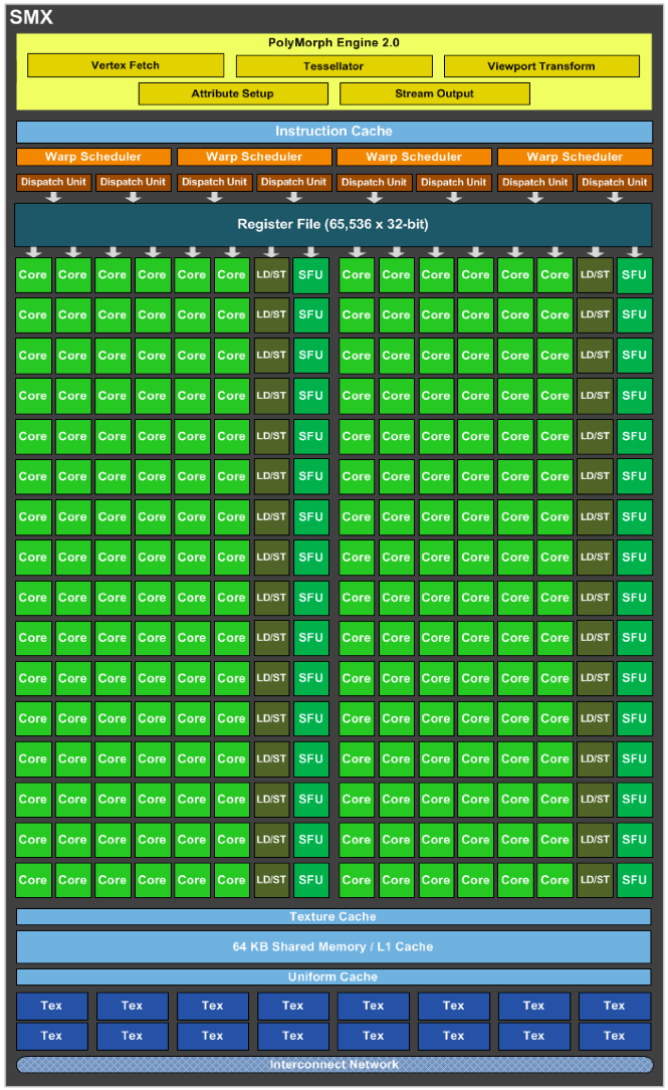
Kepler GeForce の構造

Nvidia white paper より引用



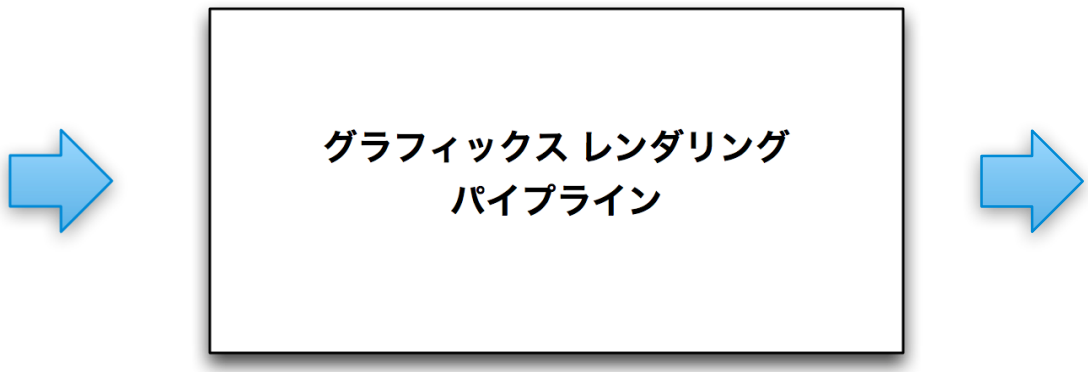
Kepler GeForce の構造

Nvidia white paper より引用



グラフィックス レンダリング パイプライン

Graphics rendering pipeline 「パイプライン」

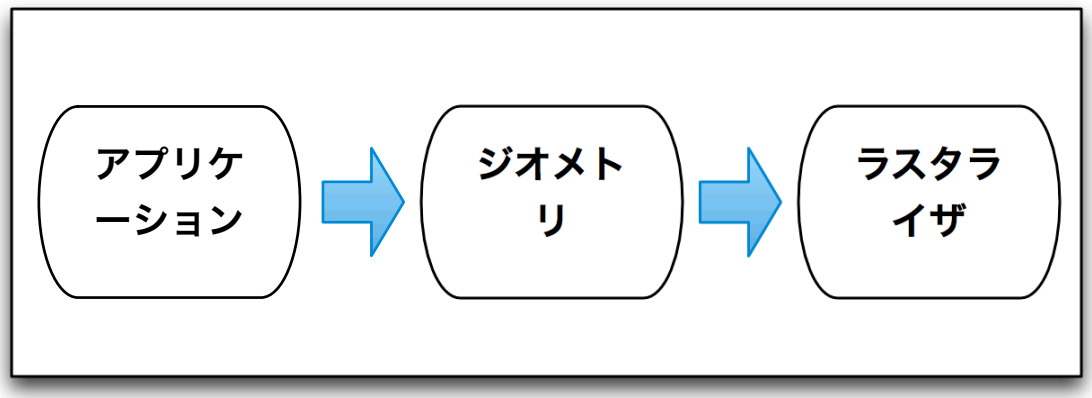


input: カメラ設定、3 次元オブジェクト、光源、ライティングモデル、テクスチャ

output: 2 次元画像

パイプラインの構造

アプリケーション ステージ → ジオメトリ ステージ→ ラスタライザ ステージ



アプリケーション ステージ

ソフトウェア

オブジェクトの生成、アニメーション、衝突検出、カリング、等々

ジオメトリ ステージ

ハードウェア

ビューボリューム

canonical view volume （標準ビューボリューム）

辺の長さ 2 の立方体

$$(-1,-1,-1) \leq (x,y,z) \leq (1,1,1)$$

標準ビューボリューム内の座標を正規化デバイス座標 （Normalized Device Coordinates, NDC）と呼ぶ。

ジオメトリステージの役割は、アプリケーションが設定した CG 世界をビューボリュームにマップすること。

射影

2 種類：

・正射影 orthographic projection, 平行投影（parallel projection）

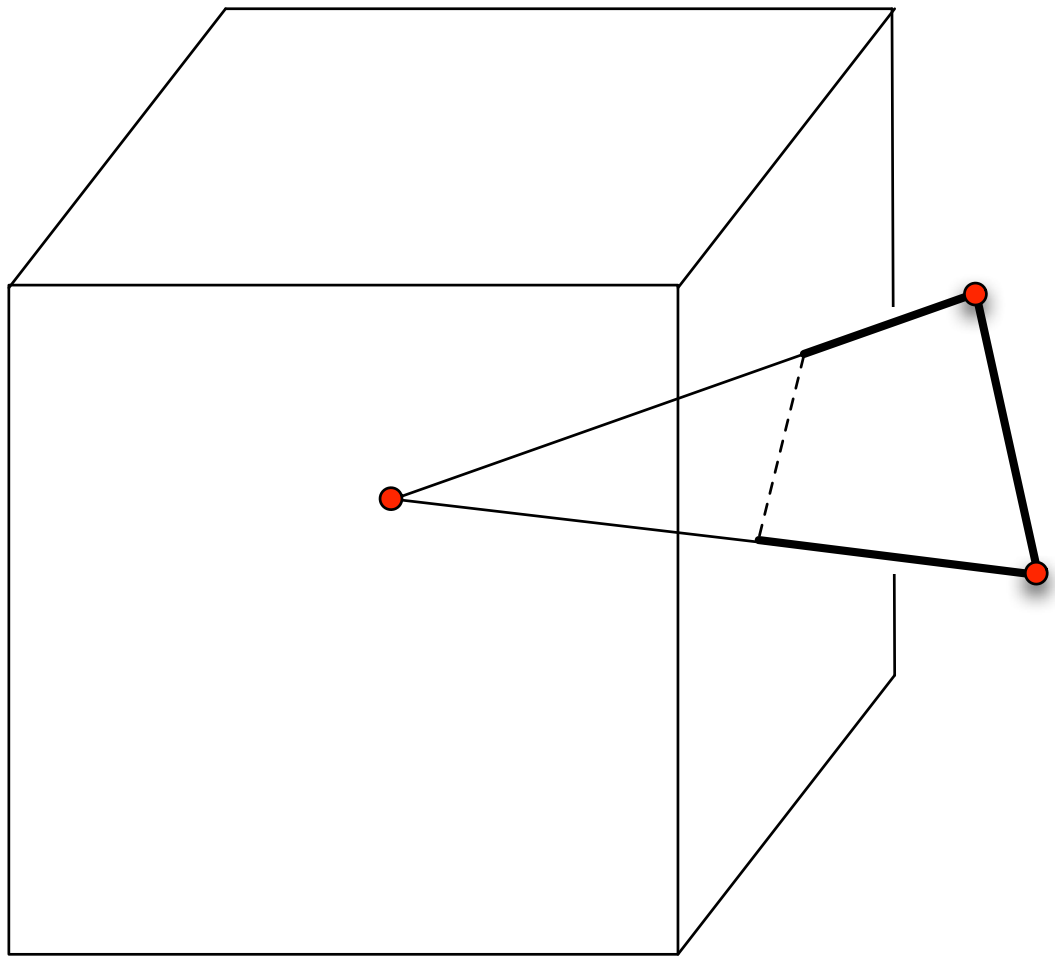
・透視射影 perspective projection, 遠近投影

視錐台 view frustum

どちらの射影も 4 行 4 列の行列演算で書ける（後述）。

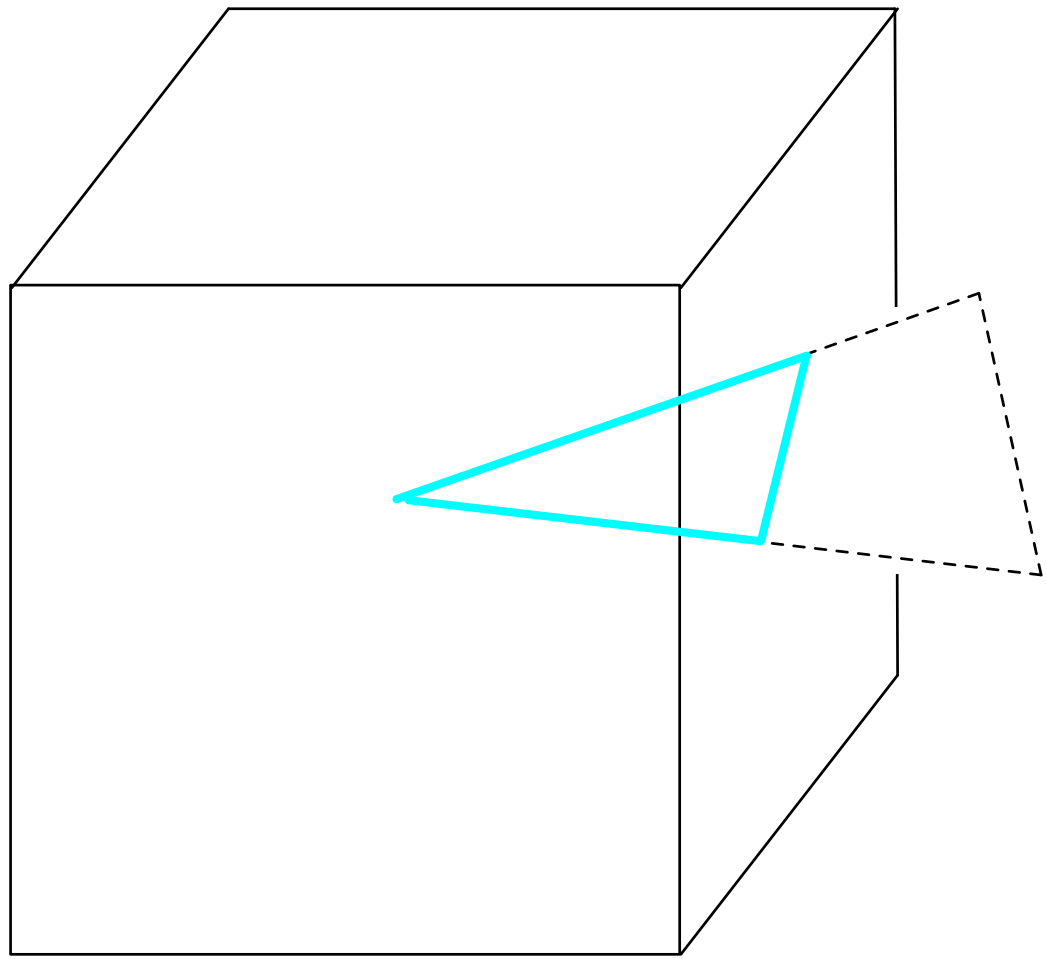
クリップ処理

ビューボリュームの外部は描かない。



クリップ処理

ビューボリュームの表面に新しい辺が自動的にできる。

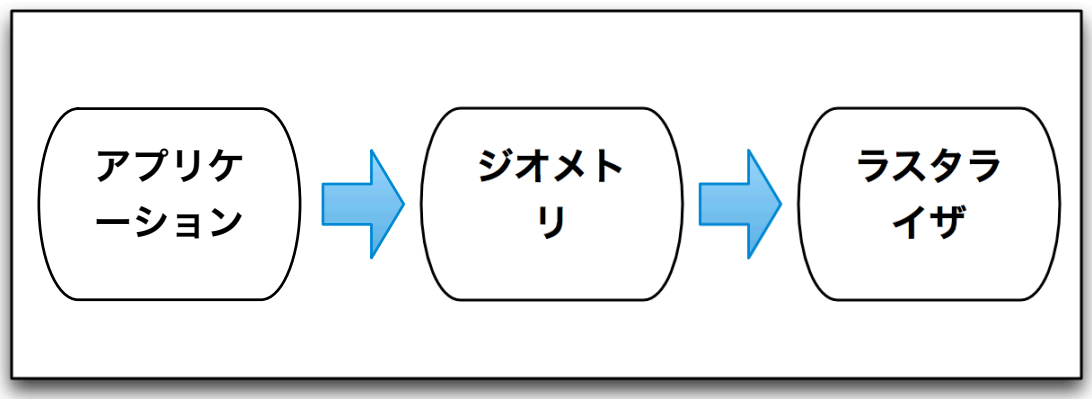


スクリーンマッピング

最終的な画像の大きさ（width と height）に合わせて、立方体のビューボリュームを x 方向と y 方向にスケール変換＋平行移動させる。z 方向には何もしない（$-1 \leq z \leq 1$）

パイプラインの構造

再掲

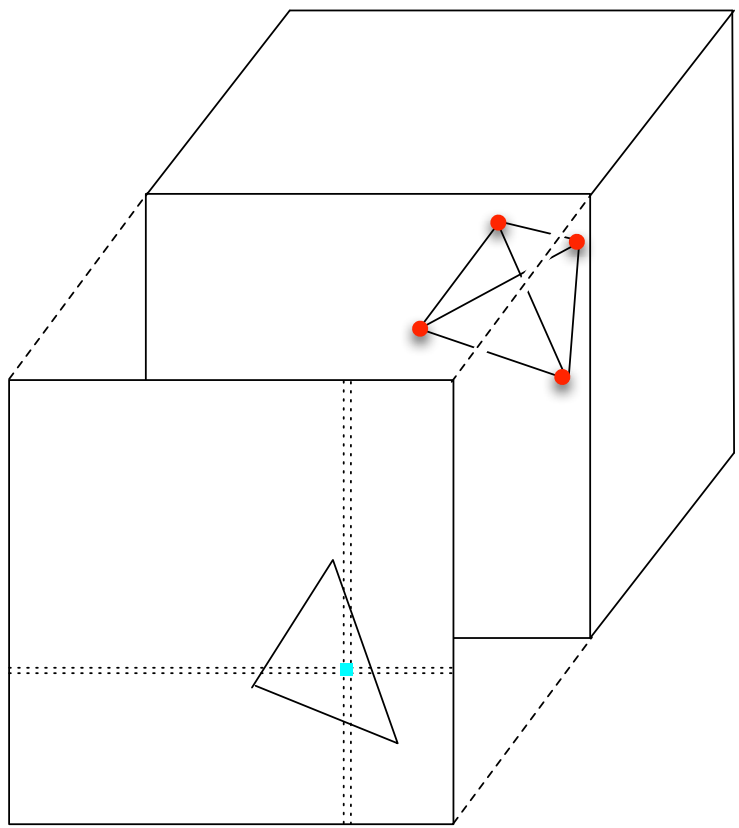


ラスタライザステージ

ビューボリューム内のデータ（3D）＋テクスチャデータ（2D） → 画像（2D）

頂点のデータ → ピクセルの色

簡単のため、スクリーンマッピングを省略すると



ラスタライザステージ

デプステスト。後述。

アルファテスト

ステンシルテスト

テクスチャマッピング

グラフィックスハードウェア

グラフィックスハードウェアの概念図

→ 【図】

フレームバッファ

カラーバッファ

Z バッファ

ステンシルバッファ

カラーバッファ

ピクセル RGBA

16 bits, 23 bits, 32 bits

RGBA32 = R8 + G8 + B8 + A8

Z バッファ

Z-buffer

デプステスト

Edwin Catmull が考案。→ 写真

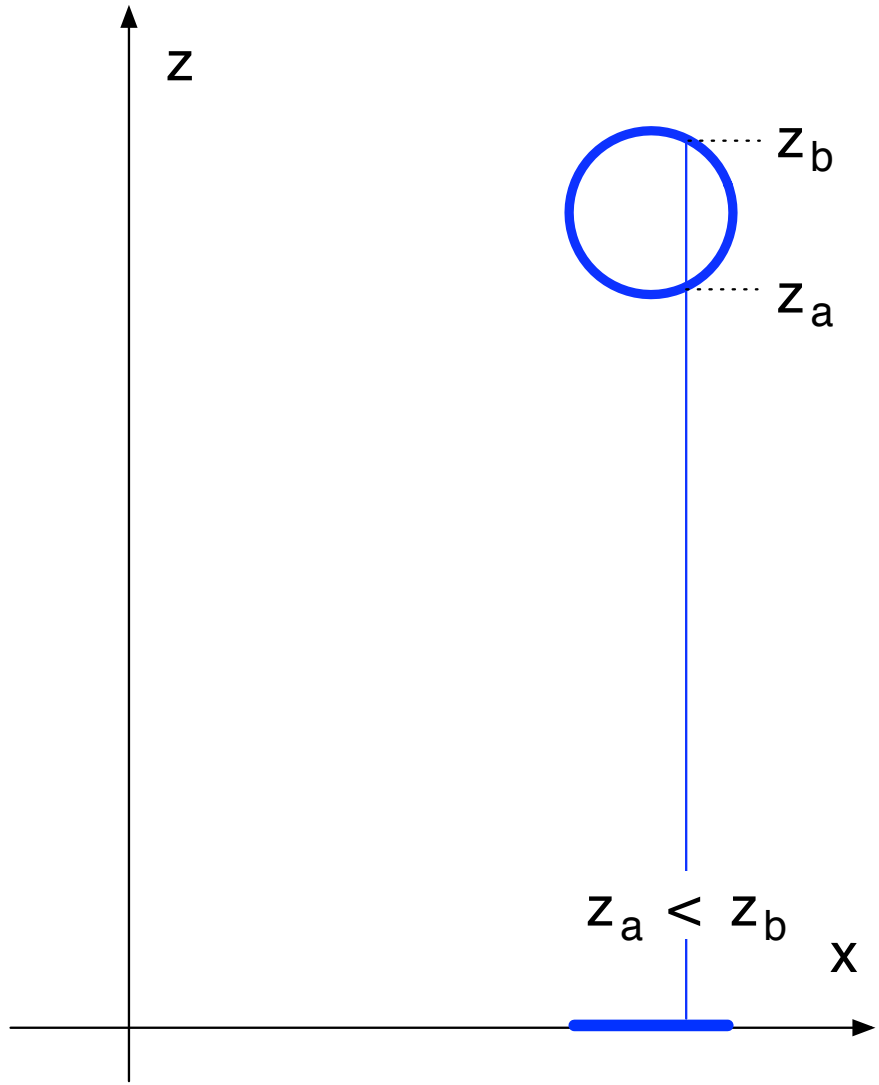
博士論文（1974 年）。

RenderMan 開発（アカデミー賞）

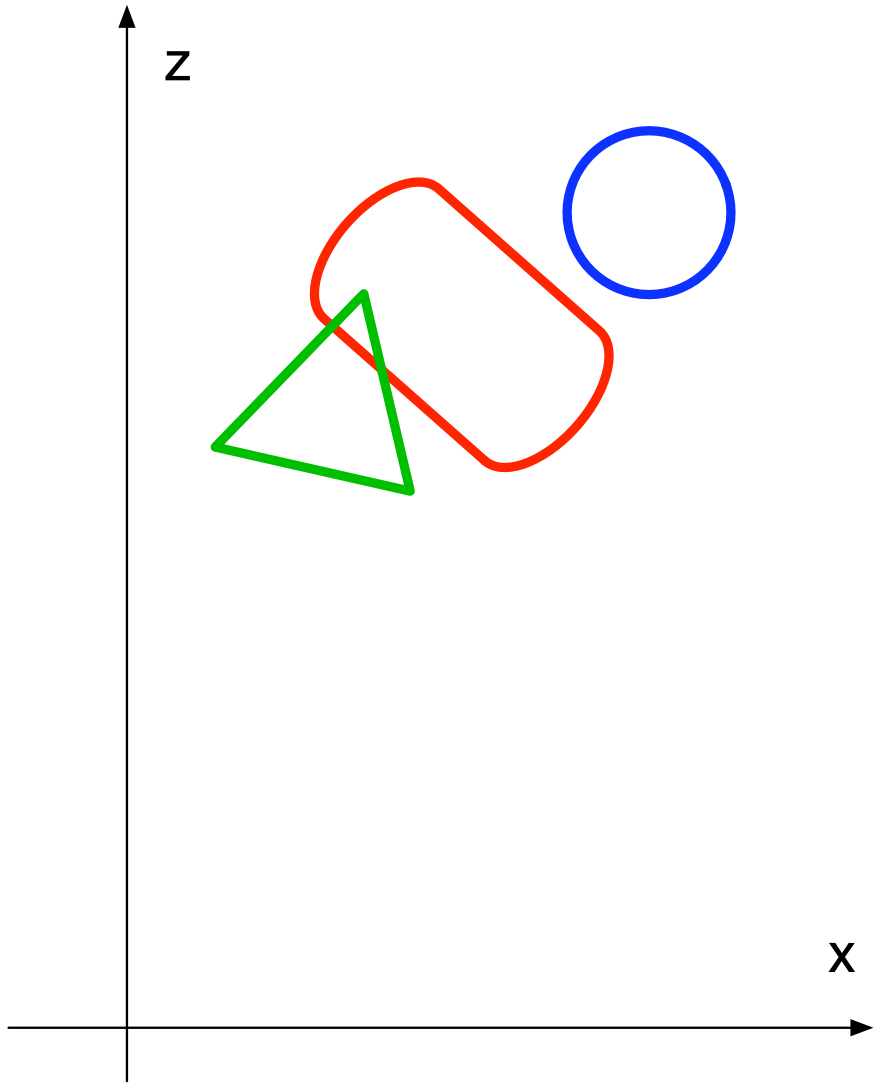
ルーカスフィルム → ピクサー設立。

ウォルト・ディズニー・アニメーション・スタジオ社長

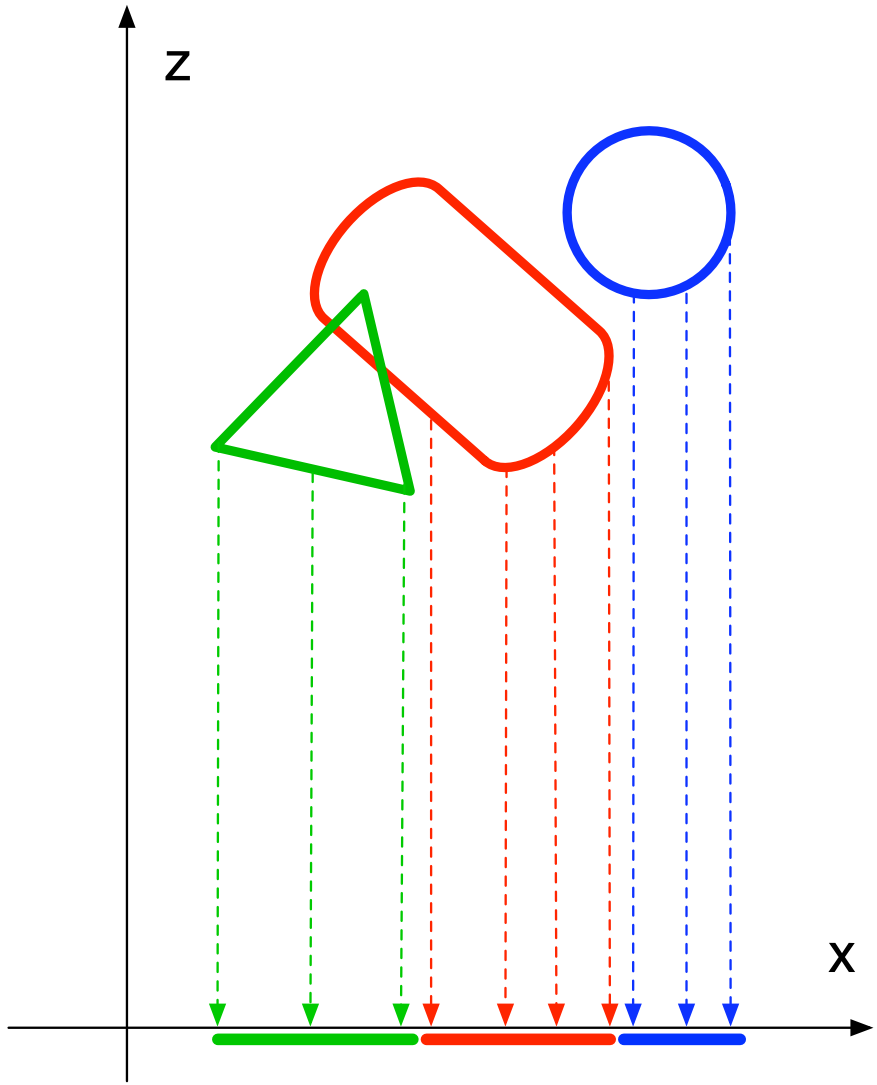
Z-buffer



Z-buffer

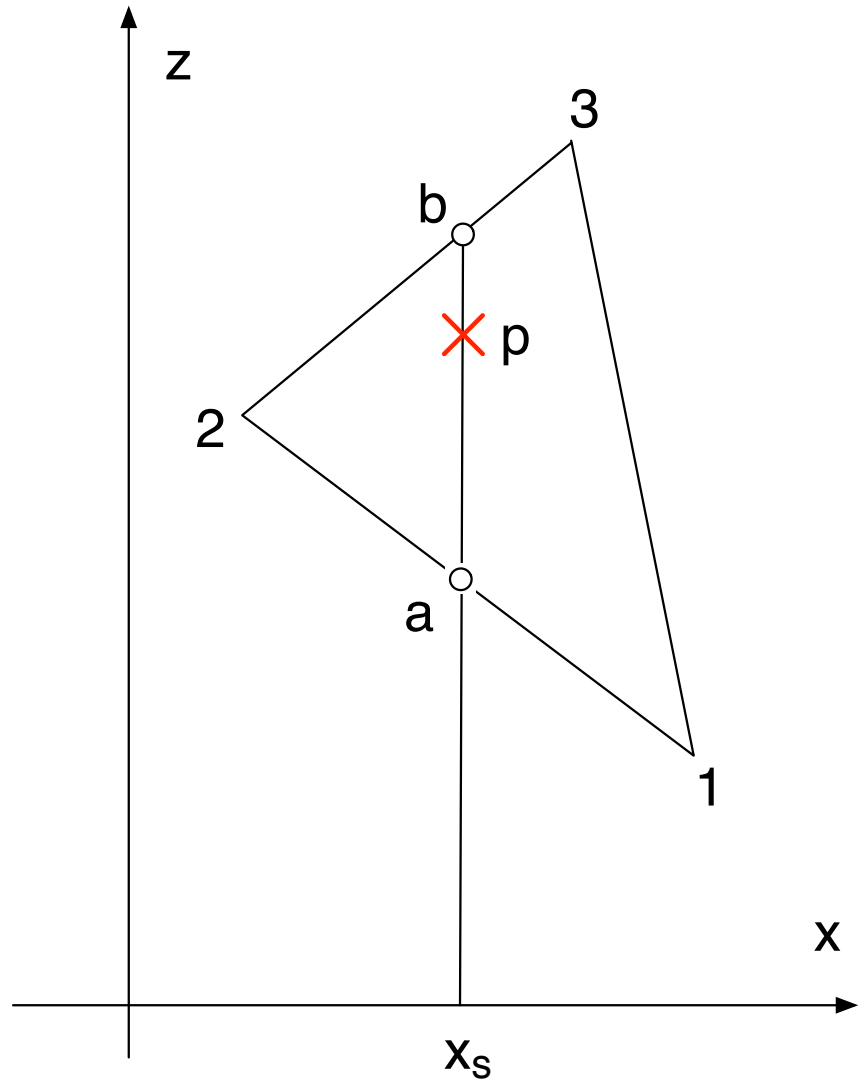


Z-buffer



Z-buffer

$$
\begin{aligned}
z_a &= z_1 + (z_2 - z_1)\frac{x_s - x_1}{x_2 - x_1} \\
z_b &= z_2 + (z_3 - z_2)\frac{x_s - x_2}{x_3 - x_2} \\
z_p &= z_a + (z_b - z_a)\frac{y_p - y_b}{y_b - x_a}
\end{aligned}
$$

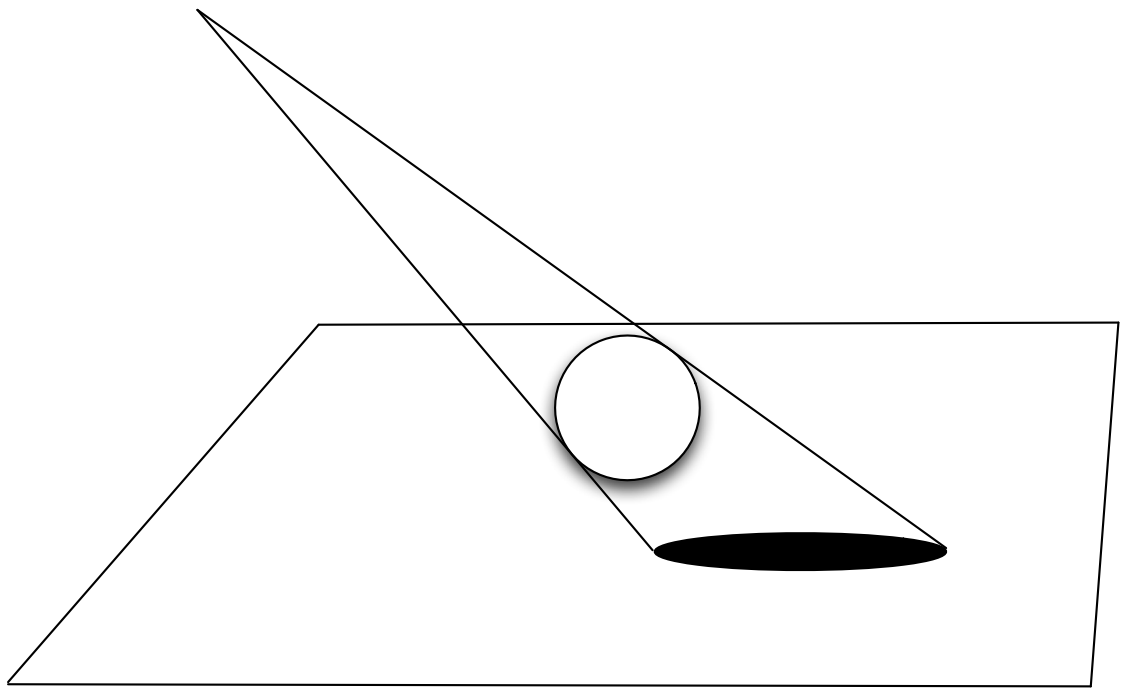


デプスマップアルゴリズム

シャドウマップアルゴリズムともいう (Williams1978)

光源に視点をおいてレンダリング

光源からオブジェクトの各点までの距離 → z バッファ → 2 次元データ → テクスチャ→ テクスチャマッピング



ステンシルバッファ

カラーバッファの値を実際に描画するかどうかをピクセル単位で制御

「型紙」

数学的準備

同次座標

同次座標（homogeneous coordinates）とは 3 次元空間中の位置座標 $\boldsymbol{x}$ と、任意のベクトル $\boldsymbol{v}$ をあえて 4 成分で表現したもの。

3 次元空間中の位置座標 $\boldsymbol{x}$ を

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1}$$

ベクトル $\boldsymbol{v}$ は

$$\begin{pmatrix} v_x \\ v_y \\ v_z \\ 0 \end{pmatrix} \tag{2}$$

アフィン変換

3 次元空間の位置座標 $\boldsymbol{x}$ や、ベクトル $\boldsymbol{v}$ の変換を考える。

$$\boldsymbol{x} \longrightarrow \boldsymbol{y} \equiv F(\boldsymbol{x}). \tag{3}$$

線形変換＝スケール変換＋回転＋剪断。

アフィン変換＝線形変換＋平行移動。

平行移動は 3 行 3 列の行列では書けない。

同次座標と 4 行 4 列の行列を使えば書ける。

線形代数の復習：内積

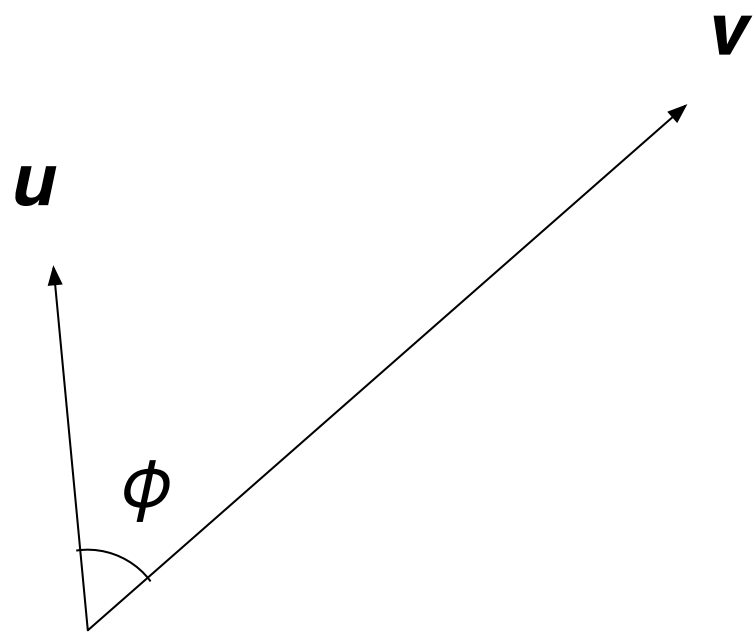
n 次元空間中のベクトルと正方行列

ベクトル $\boldsymbol{u}$ の大きさ

$$u = |\boldsymbol{u}|$$

内積

$$\boldsymbol{u} \cdot \boldsymbol{v} = u_i v_j = u\, v\, \cos\phi$$



線形代数の復習：正規直交系

$$\boldsymbol{e}_i \cdot \boldsymbol{e}_j = \delta_{ij} \quad （クロネッカーのデルタ）$$

一般のベクトル $\boldsymbol{v}$ と正規直交系 $\{\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{n-1}\}$

$\boldsymbol{v}$ の i 成分

$$v_i = \boldsymbol{v} \cdot \boldsymbol{e}_i$$

線形代数の復習：外積

3 次元空間のベクトル

$$\boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v} \qquad w_i = \epsilon_{ijk} u_j \, v_k \quad \text{(エディントンのイプシロン)}$$

$\boldsymbol{w}$ は $\boldsymbol{u}$ と $\boldsymbol{v}$ の両方に垂直

$$w = u \, v \, \sin\phi$$

$$\boldsymbol{u} \times \boldsymbol{v} = -\boldsymbol{v} \times \boldsymbol{u}$$

$$\boldsymbol{u} \cdot (\boldsymbol{v} \times \boldsymbol{w}) = (\boldsymbol{u} \cdot \boldsymbol{w}) \, \boldsymbol{v} - (\boldsymbol{u} \cdot \boldsymbol{v}) \, \boldsymbol{w}$$

線形代数の復習：行列のかけ算

$\boldsymbol{M}$ と $\boldsymbol{N}$ は行列

行列 $\boldsymbol{M}$ の成分を $M_{ij} \quad (i, j = 0, 1, \ldots, n-1)$

行列 $\boldsymbol{N}$ の成分を $N_{ij} \quad (i, j = 0, 1, \ldots, n-1)$

とすると

$$\boldsymbol{L} = \boldsymbol{M}\boldsymbol{N}$$

の成分は

$$L_{ij} = \sum_{k=0}^{n-1} M_{ik} N_{kj} = M_{ik} N_{kj}$$

線形代数の復習：行列のかけ算

$$(\boldsymbol{LM})\,\boldsymbol{N} = \boldsymbol{L}\,(\boldsymbol{MN})$$

$$(\boldsymbol{L}+\boldsymbol{M})\,\boldsymbol{N} = \boldsymbol{LN}+\boldsymbol{MN}$$

$$\boldsymbol{MI} = \boldsymbol{IM} = \boldsymbol{M} \quad (\boldsymbol{I}：単位行列)$$

一般には非可換：

$$\boldsymbol{MN} \neq \boldsymbol{NM}$$

線形代数の復習：逆行列

正方行列 $\boldsymbol{M}$ に対して

$$\boldsymbol{M}\boldsymbol{N} = \boldsymbol{N}\boldsymbol{M} = \boldsymbol{I}$$

という行列 $\boldsymbol{N}$ を逆行列という。

逆行列を

$$\boldsymbol{M}^{-1}$$

と書く。

一般には逆行列を求めるのは大変（計算量が多い）

glMatrix.js（後述）では 4 行 4 列の逆行列を求める関数が組み込まれている。

線形代数の復習：行列式

正方行列に対して

$$\det(\boldsymbol{M})$$

$$\det(\boldsymbol{I}) = 1$$

$$\det(\boldsymbol{M}\boldsymbol{N}) = \det(\boldsymbol{M})\,\det(\boldsymbol{N})$$

$$\det\left(\boldsymbol{M}^t\right) = \det(\boldsymbol{M})$$

線形代数の復習：行列の転置

行列 $\boldsymbol{M}$ の成分を $M_{ij} \quad (i, j = 0, 1, \ldots, n-1)$

転置行列を $\boldsymbol{M}^t$ と書く

a を数、$\boldsymbol{M}$ と $\boldsymbol{N}$ を行列として

$$(a\boldsymbol{M})^t = a\boldsymbol{M}^t$$

$$(\boldsymbol{M} + \boldsymbol{N})^t = \boldsymbol{M}^t + \boldsymbol{N}^t$$

$$\left(\boldsymbol{M}^t\right)^t = \boldsymbol{M}$$

$$(\boldsymbol{M}\boldsymbol{N})^t = \boldsymbol{N}^t\boldsymbol{M}^t$$

線形代数の復習：行列のトレース

$$tr(M) = \sum_{i=0}^{n-1} M_{ii}$$

線形代数の復習：直交行列

$$\boldsymbol{M}\boldsymbol{M}^t = \boldsymbol{M}^t\boldsymbol{M} = \boldsymbol{I}$$

を満たす正方行列 $\boldsymbol{M}$ を直交行列という。

$$\boldsymbol{M}^t = \boldsymbol{M}^{-1}$$

$$\det(\boldsymbol{M}) = \pm 1$$

$\boldsymbol{M}^t$ も直交行列。

直交行列はベクトルの長さを変えない：

$$|\boldsymbol{M}\boldsymbol{u}| = |\boldsymbol{u}|$$

直交する二つのベクトルを直交行列で変換しても直交したまま。

$$\boldsymbol{u}\cdot\boldsymbol{v} = 0 \Longleftrightarrow (\boldsymbol{M}\boldsymbol{u})\cdot(\boldsymbol{M}\boldsymbol{v}) = 0$$

3次元空間中の平面

点 $\boldsymbol{p}$ を通り、ベクトル $\boldsymbol{u}$ とベクトル $\boldsymbol{v}$ で張られる平面の式：

$$\boldsymbol{x} = \boldsymbol{p} + s\,\boldsymbol{u} + t\,\boldsymbol{v}$$

単位ベクトル $\boldsymbol{n} \equiv \boldsymbol{u} \times \boldsymbol{v}/|\boldsymbol{u} \times \boldsymbol{v}|$ を使えば、

$$\boldsymbol{n} \cdot \boldsymbol{x} + d = 0$$

$\boldsymbol{n}$ を法線ベクトルという。$f(\boldsymbol{x}) = \boldsymbol{n} \cdot \boldsymbol{x} + d$ とすると

$f(\boldsymbol{x}_0) = 0 \Longleftrightarrow$ 点 $\boldsymbol{x}_0$ はこの平面の上

$f(\boldsymbol{x}_0) > 0 \Longleftrightarrow$ 点 $\boldsymbol{x}_0$ は $\boldsymbol{p} + \boldsymbol{n}$ 側にある

$f(\boldsymbol{x}_0) < 0 \Longleftrightarrow$ 点 $\boldsymbol{x}_0$ は $\boldsymbol{p} - \boldsymbol{n}$ 側にある

面積

3 点 $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$ を頂点とする 3 角形の面積

$$S = \frac{1}{2} \left| (\boldsymbol{p} - \boldsymbol{r}) \times (\boldsymbol{q} - \boldsymbol{r}) \right|$$

x-y 平面上におかれた n 角形の面積

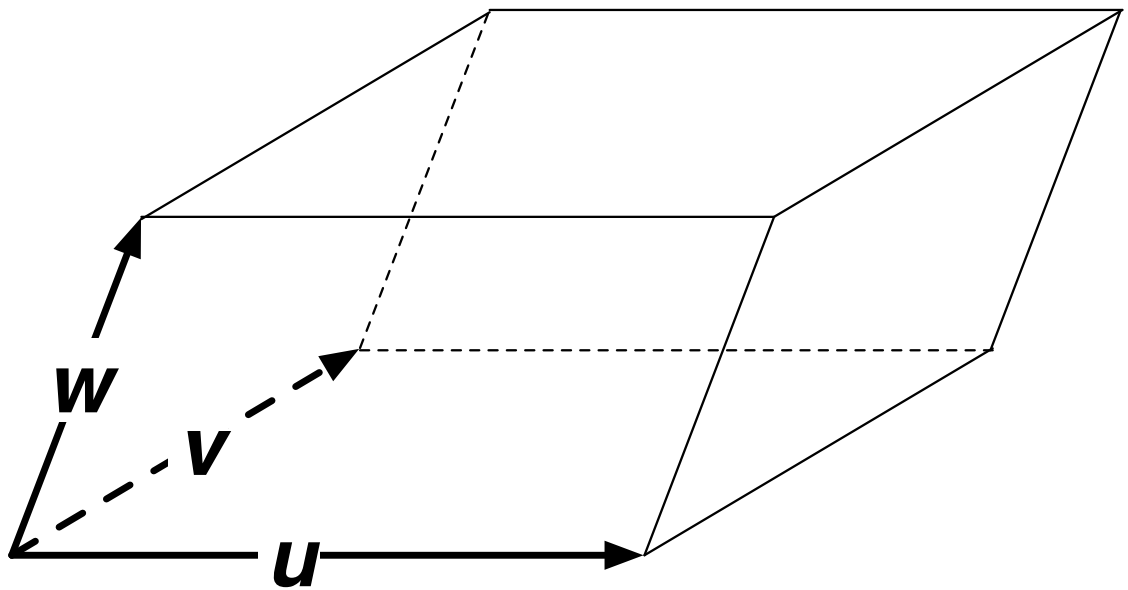
$$S = \frac{1}{2} \sum_{i=0}^{n-1} (x_i \, y_{i+1} - y_i \, x_{i+1}) = \frac{1}{2} \sum_{i=0}^{n-1} \{x_i \, (y_{i+1} - y_{i-1})\}$$

添字は　$\mathrm{mod}\ (n)$ をとる。

体積

原点を基点とする 3 つのベクトル $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ が張る平行 6 面体の体積

$$V = \boldsymbol{u} \cdot (\boldsymbol{v} \times \boldsymbol{w}) = \boldsymbol{v} \cdot (\boldsymbol{w} \times \boldsymbol{u}) = \boldsymbol{w} \cdot (\boldsymbol{u} \times \boldsymbol{v})$$



同次座標

3 次元 $\Longrightarrow$ 4 次元

3 次元空間の位置座標

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \Longrightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3 次元空間のベクトル

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \Longrightarrow \begin{pmatrix} v_x \\ v_y \\ v_z \\ 0 \end{pmatrix}$$

同次座標

行列

$$\boldsymbol{M} = \begin{pmatrix} M_{00} & M_{01} & M_{02} & 0 \\ M_{10} & M_{11} & M_{12} & 0 \\ M_{20} & M_{21} & M_{22} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

平行移動

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \end{pmatrix} \tag{4}$$

これを

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = T(t_x, t_y, t_z) \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{5}$$

と書けるような 3 行 3 列の行列 T は存在しない。$\Rightarrow$ 4 行 4 列にすれば OK.

平行移動

平行移動行列

$$T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6}$$

回転

z 軸の周りの回転

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7}$$

スケール変換

$$S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{8}$$

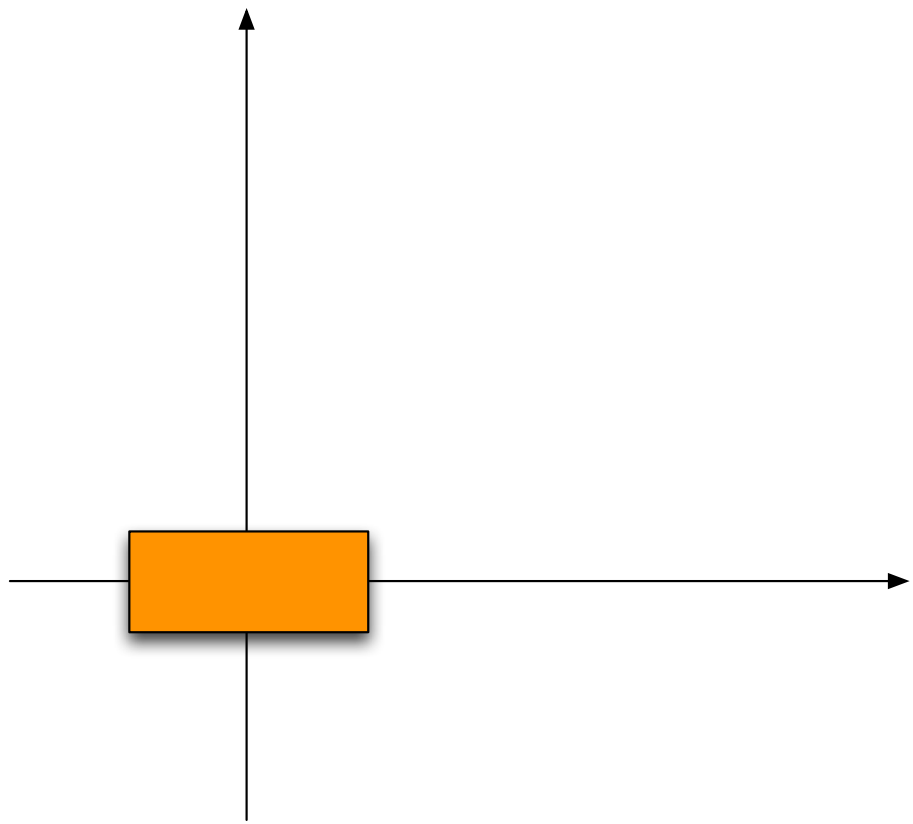
剪断

$$H_{xy}(\beta) = \begin{pmatrix} 1 & \beta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9}$$

座標変換の合成

アフィン変換は非可換。一般に

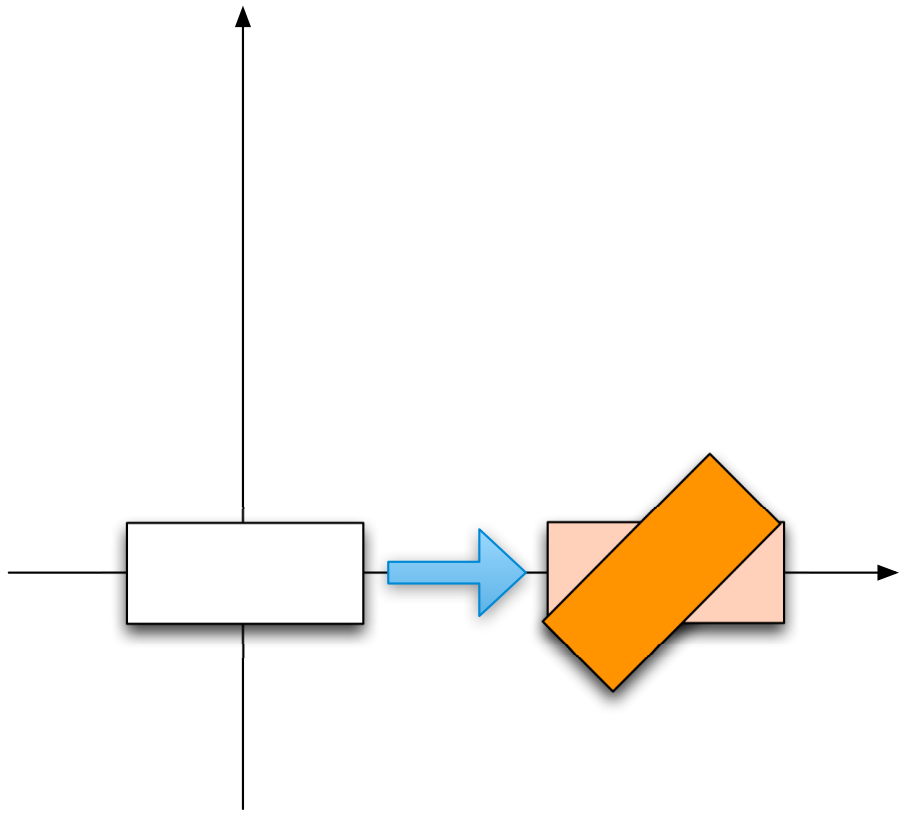
$$M_1 M_2 \neq M_2 M_1$$



座標変換の合成

アフィン変換は非可換。一般に

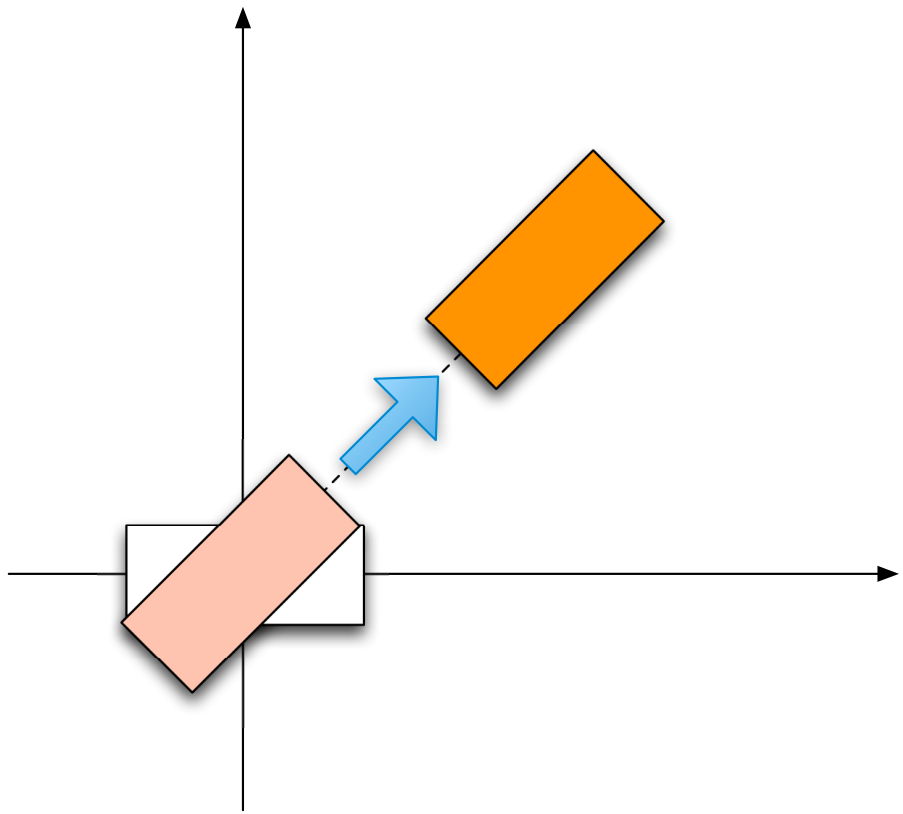
$$RT \neq TR$$



座標変換の合成

アフィン変換は非可換

$$RT \neq TR$$



演習：機能の確認

WebGL 公式ページ

`http://www.khronos.org/webgl/`

デモ集`http://www.khronos.org/webgl/wiki/Demo_Repository`

Safari でのソースコード表示方法：

- 開発メニュー
- → ページのソースを表示