WebGL によるデータ可視化入門[*1]

簡単な WebGL プログラム

陰山 聡

神戸大学 システム情報学研究科 計算科学専攻

2013.04.30

[*1] 情報可視化論 X021（2013 年前期） LR301

事務連絡

前回の復習

線形代数の復習

WebGL グラフィックスパイプライン

事務連絡

- 次週より教室変更
- 3 号館 1 階演習室
- 自分の（ネットにつながる）ノート PC を持ち込んでもよい
- ID とパスワード：

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

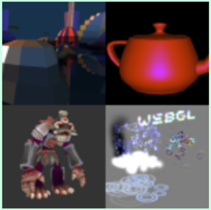
公式Web Siteとサポートしているブラウザ

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

Implementations

- Desktop
  - Apple Safari (WebKit)
  - Google Chrome
  - Mozilla Firefox
  - Opera
- Mobile
  - Chrome for Android
  - Firefox for Android

Demos



Demo Repository

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

線形代数の復習

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。

位置座標 $\boldsymbol{x} = (x, y, z)^T$ は

$$(x, y, z, 1)^t \tag{1}$$

とする。

ベクトル $\boldsymbol{v} = (v_x, v_y, v_z)^T$ は

$$(v_x, v_y, v_z, 0)^t \tag{2}$$

とする。

アフィン変換

3 次元空間の位置座標 $\boldsymbol{x}$ や一般のベクトル $\boldsymbol{v}$ 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

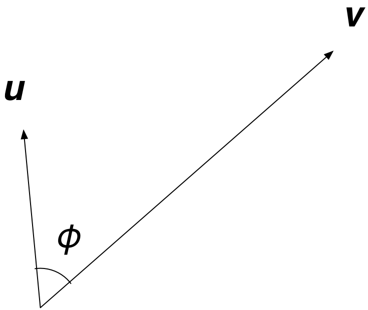
n 次元空間中のベクトルと正方行列

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

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

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

$$\boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v} \qquad w_i = \epsilon_{ijk} u_j \, v_k$$ （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

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

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

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

一般には非可換：

$$\boldsymbol{M}\boldsymbol{N} \neq \boldsymbol{N}\boldsymbol{M}$$

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

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

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

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

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

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すると直交する

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

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

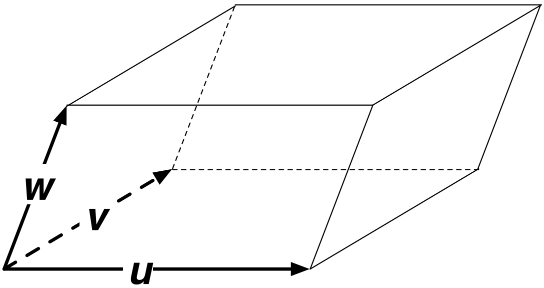
$$S = \frac{1}{2} \sum_{i=0}^{n-1} (x_i \, y_{i+1} - y_i \, x_{i+1}) = \frac{1}{2} \sum_{i=0}^{n-1} \{x_i \, (y_{i+1} - y_{i-1})\}$$

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 $\rightarrow$ 4 次元

3 次元空間の位置座標

$$(x, y, z) \Longrightarrow (x, y, z, 1)$$

3 次元空間のベクトル

$$(v_x, v_y, v_z) \Longrightarrow (v_x, v_y, v_z, 0)$$

3 次元空間の行列

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

平行移動

平行移動行列

$$T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4}$$

回転

z 軸の周りの回転

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5}$$

スケール変換

$$S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6}$$

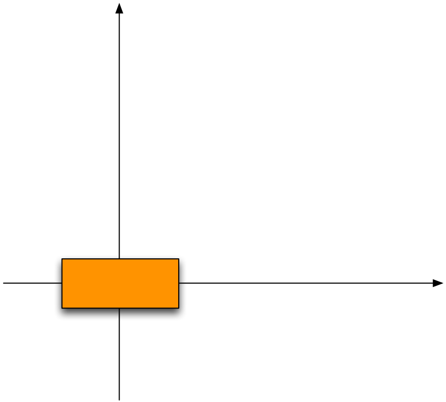
剪断

$$H_{xy}\beta = \begin{pmatrix} 1 & \beta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7}$$

座標変換の合成

アフィン変換は非可換。一般に

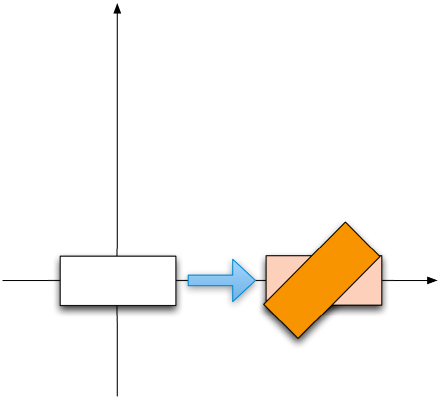
$$M_1 M_2 \neq M_2 M_1$$



座標変換の合成

アフィン変換は非可換。一般に

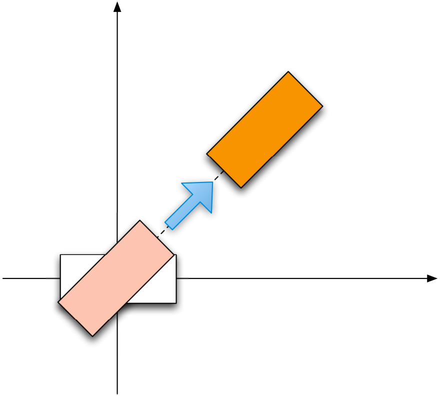
$$RT \neq TR$$



座標変換の合成

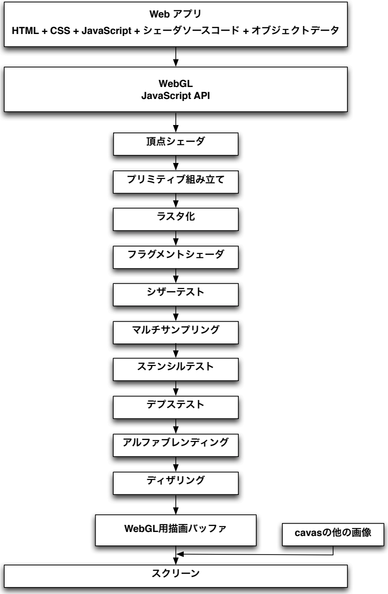
アフィン変換は非可換

$$RT \neq TR$$



WebGL グラフィックスパイプライン

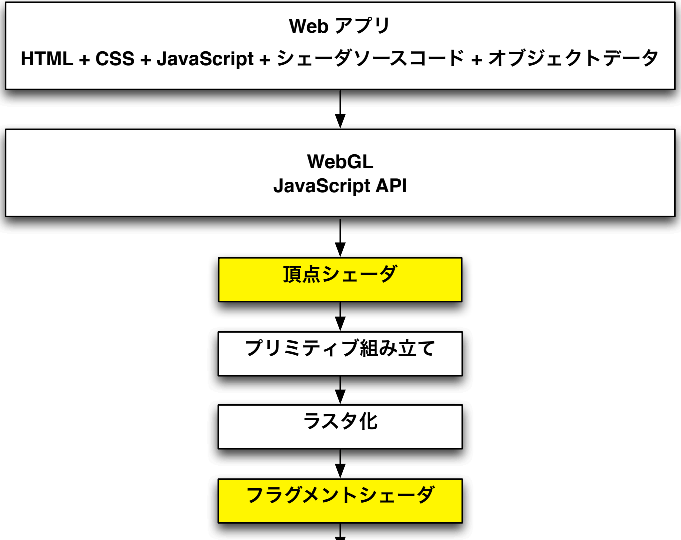
WebGL のグラフィックスパイプライン



シェーダ（拡大図は次のページ）

シェーダ

頂点シェーダ（バーテックスシェーダ）とフラグメントシェーダ



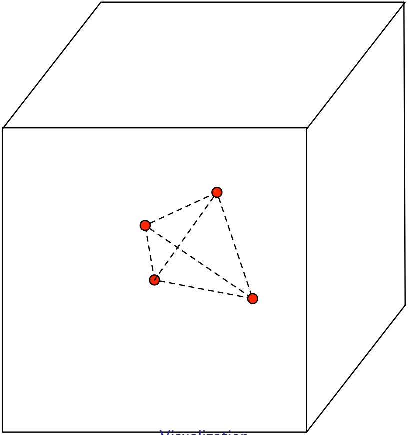
WebGL アプリケーション

Web アプリ = HTML + CSS + JavaScript

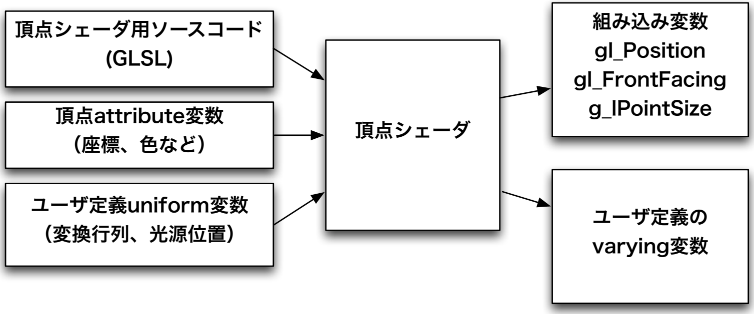
WebGL アプリ = HTML + CSS + JavaScript + シェーダ言語（OpenGL SL）

頂点シェーダ

- 各頂点に対して処理を行う
- 並列処理
- n 個の頂点があれば n 個の頂点シェーダプロセッサを同時に実行させる



頂点シェーダの入出力データ



頂点シェーダプログラム

- C 言語に似ている。
- OpenGL SL (Shading Language)
- 4 行 4 列の行列ベクトル演算が組み込み関数

```
attribute vec3 aVertexPos;
atrribute vec4 aVertexColor;

uniform mat4 uMVMatrix;
uniform mat4 uPMatrix;

varying vec4 vColor;

void main() {
  gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPos, 1.0);
  vColor = aVertexColor;
}
```