

# コンピュータグラフィックスS

## 第8回 座標変換(1)

システム創成情報工学科 尾下 真樹

### 今日の内容

- 前回の復習
  - 演習(2):ポリゴンモデルの描画
- 変換行列
  - 座標変換の概要
  - 座標系
  - 視野変換(アフィン変換)
  - 透視変換
  - 座標変換のまとめ
  - 演習問題

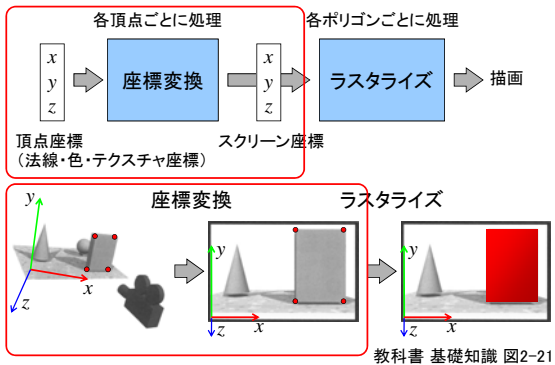
### 教科書(参考書)

- 「コンピュータグラフィックス」  
CG-ARTS協会 編集・出版  
- 2章
- 「ビジュアル情報処理 -CG・画像処理入門-」  
CG-ARTS協会 編集・出版  
- 1章(1-2~1-3節)  
- 3次元ではなく2次元の例で説明

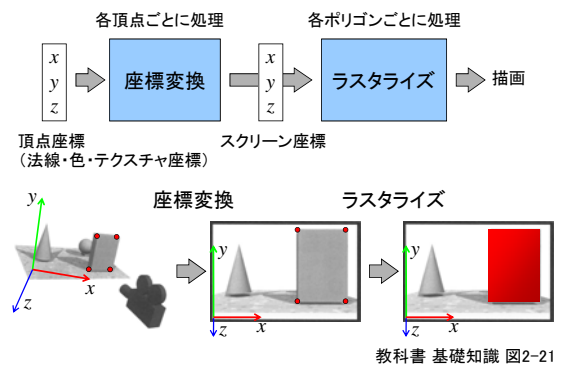


### 前回の復習

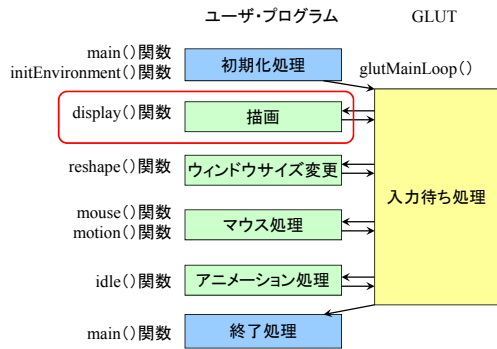
### レンダリング・パイプライン



### 処理の流れ



### サンプルプログラムの構成



### 描画関数の流れ

```
//
// ウィンドウ再描画時に呼ばれるコールバック関数
//
void display( void )
{
    // 画面をクリア(ピクセルデータとZバッファの両方をクリア)
    // 変換行列を設定(ワールド座標系→カメラ座標系)
    // 光源位置を設定(モデルビュー行列の変更にあわせて再設定)
    // 地面を描画
    // 変換行列を設定(物体のモデル座標系→カメラ座標系)
    // 物体(1枚のポリゴン)を描画
    // バックバッファに描画した画面をフロントバッファに表示
}
```

### ポリゴンモデルの描画方法

- 方法1: `glVertex()` 関数に直接頂点座標を記述
  - 頂点データ(直接記述)、頂点ごとに渡す
- 方法2: 頂点データの配列を使用
  - 頂点データ、頂点ごとに渡す
- 方法3: 頂点データと面インデックスの配列を使用
  - 頂点データ+面インデックス、頂点ごとに渡す

### 配列を使った四角すいの描画(1)

#### • 配列データの定義

```
const int num_pyramid_vertices = 5; // 頂点数
const int num_pyramid_triangles = 6; // 三角面数
// 角すいの頂点座標の配列
float pyramid_vertices[ num_pyramid_vertices ][ 3 ] = {
    { 0.0, 1.0, 0.0 }, { 1.0,-0.8, 1.0 }, { 1.0,-0.8,-1.0 }, .....
};
// 三角面インデックス(各三角面を構成する頂点の頂点番号)の配列
int pyramid_tri_index[ num_pyramid_triangles ][ 3 ] = {
    { 0,3,1 }, { 0,2,4 }, { 0,1,2 }, { 0,4,3 }, { 1,3,2 }, { 4,2,3 }
};
// 三角面の法線ベクトルの配列(三角面を構成する頂点座標から計算)
float pyramid_tri_normals[ num_pyramid_triangles ][ 3 ] = {
    { 0.00, 0.53, 0.85 }, // +Z方向の面
    .....
};
```

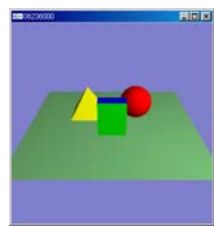
### 配列を使った四角すいの描画(2)

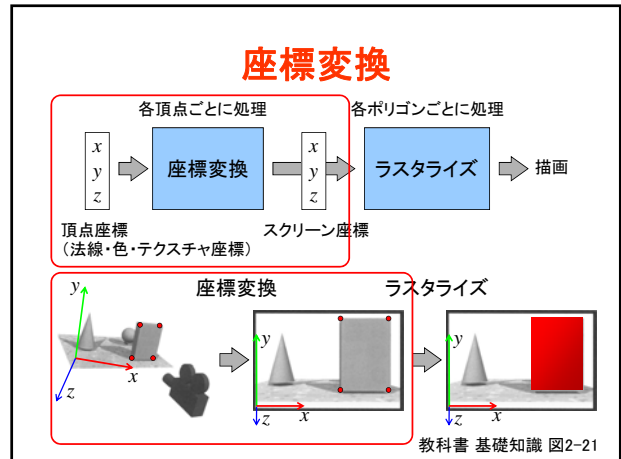
- 配列データを参照しながら三角面を描画

```
void renderPyramid3D()
{
    // 各三角面ごとに繰り返し
    for ( i=0; i<num_pyramid_triangles; i++ )
    {
        // 面の法線を指定 (i番目の面のデータを指定)
        glNormal3f( pyramid_tri_normals[i][0], .. [i][1], .. [i][2] );
        // 三角面の各頂点ごとに繰り返し
        for ( j=0; j<3; j++ )
        {
            // 頂点番号を取得 (i番目の面のj番目の頂点が、何番目の頂点を使うかを取得)
            int v_no = pyramid_tri_index[ i ][ j ];
            // 頂点座標を指定 (v_no番目の頂点のデータを指定)
            glVertex3f( pyramid_vertices[ v_no ][0], ... [ v_no ][1], ... );
        }
        glEnd();
    }
}
```

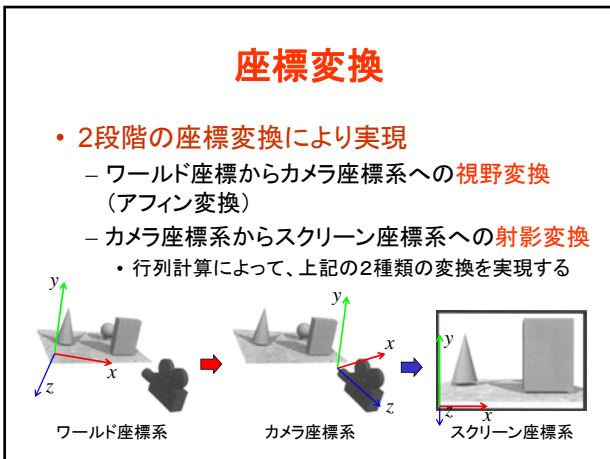
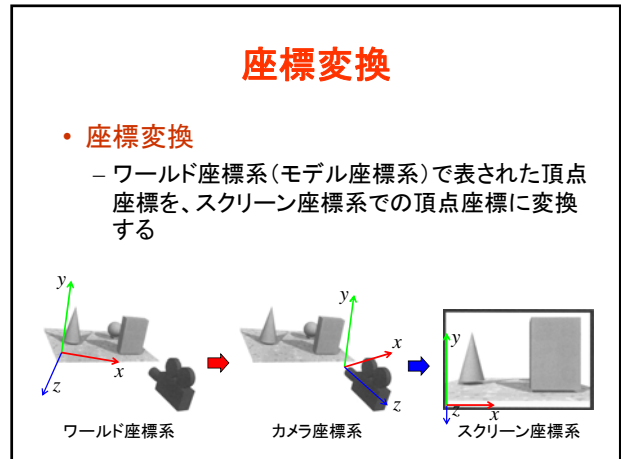
### 前回の演習

- ポリゴンの描画方法(復習)
- 基本オブジェクトの描画
- ポリゴンモデルの描画
- 演習課題





- ### 座標変換
- 座標変換の概要
  - 座標系
  - 視野変換(アフィン変換)
  - 透視変換
  - 座標変換のまとめ
  - 演習問題



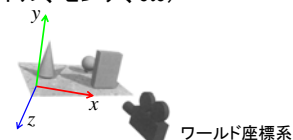
- ### 座標変換
- 座標変換の概要
  - 座標系
  - 視野変換(アフィン変換)
  - 射影変換
  - 座標変換のまとめ
  - 演習問題

### 座標系の種類

- 原点と座標軸の取り方により、さまざまな座標系がある
  - モデル座標系
  - ワールド座標系
  - カメラ座標系
  - スクリーン座標系
- 座標系の軸の取り方に違いがある
  - 右手座標系
  - 左手座標系

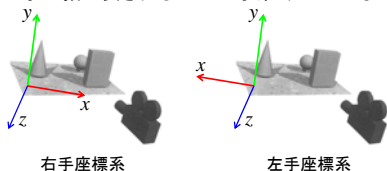
### ワールド座標系

- 3次元空間の座標系
  - 物体や光源やカメラなどを配置する座標系
  - 原点や軸方向は適当にとつて構わない
    - カメラと描画対象の相対位置・向きのみが重要
  - 単位も統一さえされていれば自由に設定して構わない(メートル、センチ、etc)



### 右手座標系と左手座標系

- 右手座標系と左手座標系
  - 座標系の軸の取り方の違い
  - 親指をX軸、人差し指をY軸、中指をZ軸とすると
    - 右手の指で表されるのが右手系 (OpenGLなど)
    - 左手の指で表されるのが左手系 (DirectXなど)

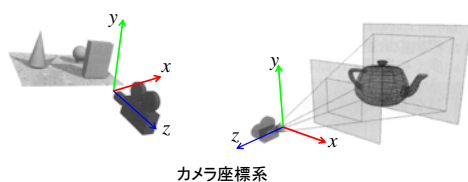


### 右手座標系と左手座標系(続き)

- 右手座標系と左手座標系の違い
  - 基本的にはほとんど同じ
  - 外積の定義が異なる
    - 外積の計算式は、右手座標系で定義されたもの
    - 左手座標系で外積を計算するときには、符号を反転する必要がある
      - 剛体の運動計算や電磁気などの物理計算では重要になる(この講義では扱わない)
  - 異なる座標系で定義されたモデルデータを利用する時には、変換が必要
    - 左右反転、面の方向を反転

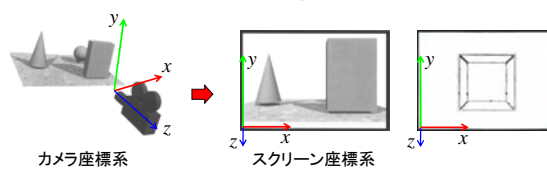
### カメラ座標系

- カメラを中心とする座標系
  - X軸・Y軸がスクリーンのX軸・Y軸に相当
  - 奥行きがZ軸に相当



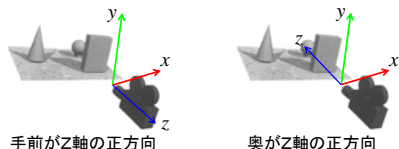
### スクリーン座標系

- スクリーン上の座標
  - 射影変換(透視変換)を適用した後の座標
    - 奥にあるものほど中央に描画されるように座標計算
  - スクリーン座標も奥行き値(Z座標)も持つことに注意 → Zバッファ法で使用



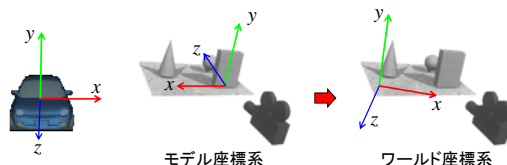
### 右手座標系と左手座標系

- カメラ座標系・スクリーン座標系も、軸の取り方によって、座標系は異なる
  - 手前がZ軸の正方向 (OpenGL)
  - 奥がZ軸の正方向 (DirectX)
- こちらも基本的にはどちらでも構わない



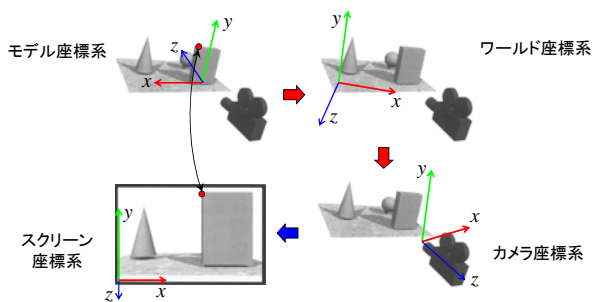
### モデル座標系

- 物体のローカル座標
  - ポリゴンモデルの頂点はモデル内部の原点を基準とするモデル座標系で定義される
  - 正面方向をZ軸にとる場合が多い
  - ワールド座標系にモデルを配置



### 座標変換の流れ(詳細)

- モデル座標系からスクリーン座標系に変換

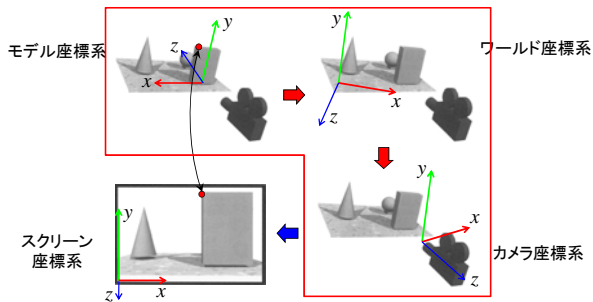


### 座標変換

- 座標変換の概要
- 座標系
- 視野変換(アフィン変換)
- 射影変換
- 座標変換のまとめ
- 演習問題

### 視野変換(アフィン変換)

- モデル座標系からカメラ座標系に変換



### アフィン変換

- アフィン変換(同次座標系変換)
  - 4x4行列の演算によって、3次元空間における平行移動・回転・拡大縮小などを実現
  - 同次座標系
    - (x, y, z, w)の4次元座標値によって扱う
    - 3次元座標値は(x/w, y/w, z/w)で計算(通常は w = 1)

$$\begin{pmatrix} R_{00}S_x & R_{01} & R_{02} & T_x \\ R_{10} & R_{11}S_y & R_{12} & T_y \\ R_{20} & R_{21} & R_{22}S_z & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix}$$

### 平行移動

• 平行移動

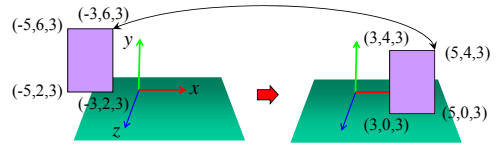
-  $(T_x, T_y, T_z)$  の平行移動

- $4 \times 4$ 行列を用いることで、平行移動を適用することができる

$$\begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+T_x \\ y+T_y \\ z+T_z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

### 平行移動の例

•  $(8, -2, 0)$  平行移動



$$\begin{pmatrix} 1 & 0 & 0 & 8 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+8 \\ y-2 \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

### 回転変換

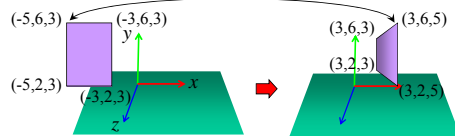
• 回転変換

- 原点を中心とする回転を表す

$$\begin{pmatrix} R_{00} & R_{01} & R_{02} & 0 \\ R_{10} & R_{11} & R_{12} & 0 \\ R_{20} & R_{21} & R_{22} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} R_{00}x + R_{01}y + R_{02}z \\ R_{10}x + R_{11}y + R_{12}z \\ R_{20}x + R_{21}y + R_{22}z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

### 回転変換の例

• Y軸を中心として90度回転



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

### 回転変換の行列

• 回転変換の行列の導出方法

- 各軸を中心として右ねじの方向の回転(軸の元から見て反時計回り方向の回転)を通常使用
- yz平面, xz平面, xy平面での回転を考えれば、2次元平面での回転変換と同様に求められる

• 2次元平面での回転行列は、高校の数学の内容

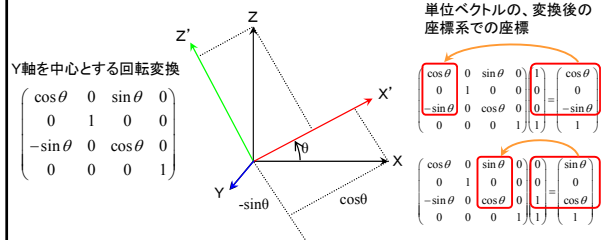
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

X軸を中心とする回転変換    Y軸を中心とする回転変換    Z軸を中心とする回転変換

### 回転変換の行列(続き)

• 回転変換の行列の導出方法の例

- 例えば、y軸周りの回転行列は、xz平面での回転を考えれば、導出できる



### 拡大縮小

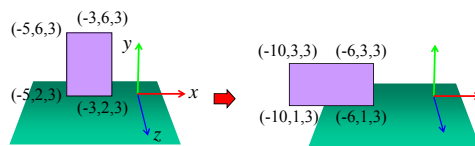
• 拡大縮小

- (S<sub>x</sub>, S<sub>y</sub>, S<sub>z</sub>) 倍のスケールリング

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S_x x \\ S_y y \\ S_z z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

### 拡大縮小の例

• (2, 0.5, 1) 倍に拡大縮小

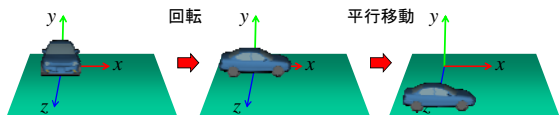


$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 2x \\ 0.5y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

### 行列演算の適用

- 1つの行列演算で各種の変換を適用可能
- 行列を次々にかけていくことで、変換を適用することができる

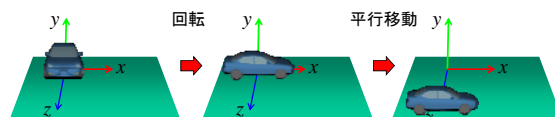
- 回転・移動の組み合わせの例



$$\begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

### 行列演算の適用

• 回転・移動の組み合わせの例



$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

先に適用する方が右側になることに注意!

### 行列計算の適用順序

- 行列演算では可換則は成り立たないことに注意!

$$AB \neq BA$$

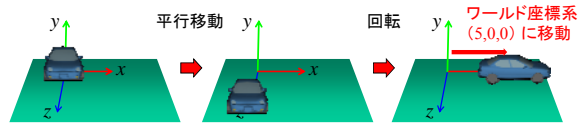
- 行列の適用順序によって結果が異なる

- 例:

- 回転 → 平行移動
- 平行移動 → 回転

### 行列演算の適用

• 移動→回転の順番で適用したときの例



$$\begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 5 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

この場合は平行移動成分にも回転がかかる

### 行列演算の適用

- 移動→回転の順番で適用したときの例

ワールド座標系 (5,0,0) に移動

モデル座標系 (0,0,5) に移動

平行移動はモデル座標系の向きで適用されていることになる

$$\begin{matrix} \text{回転} & \text{平行移動} \\ \begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

### 行列演算の適用

- 回転→移動の順番で適用(さきほどの例)

平行移動

回転

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

こちらの順番の方が普通に使う場合が多い

### アフィン変換の考え方

- アフィン変換の考え方
  - ある座標系内での回転・平行移動・拡大縮小の変換と考えることもできるし、
  - ある座標系から別の座標系への座標系の変換と考えることもできる

モデルをC1→C2に移動・回転 = C2→C1の変換行列を求める

変換行列を適用しない状態では、移動や回転はなし

回転→移動の変換を適用

### アフィン変換の逆変換

- 逆行列を計算すれば、反対方向の変換も求まる
- アフィン変換の行列は、常に正則であるため、逆行列が存在する

### アフィン変換のメリット

- 行列演算だけでさまざまな処理を行える
  - アフィン変換を使わずとも、回転・平行移動・拡大縮小など各処理に応じて計算することは可能
    - それぞれの処理だけをみればこちらの方が高速
  - 各種処理を同じ方法で扱えることに意味がある
- 複数の変換をまとめて一つの行列にできる
  - 最初に一度全行列を計算してしまえば、後は各頂点につき1回の行列演算だけで処理できる
- CG以外の分野でも広く用いられている

### アフィン変換の表記方法

- 2通りの書き方がある
  - どちらの書き方で考えても良い
    - 本講義では、左から行列を掛ける表記を使用
  - 使用するライブラリによって行列データの渡し方が異なるので注意

$$\begin{pmatrix} R_{00} & R_{01} & R_{02} & T_x \\ R_{10} & R_{11} & R_{12} & T_y \\ R_{20} & R_{21} & R_{22} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

左から行列を掛けていく表記 (OpenGL)

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \begin{pmatrix} R_{00} & R_{01} & R_{02} & 0 \\ R_{10} & R_{11} & R_{12} & 0 \\ R_{20} & R_{21} & R_{22} & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

右から行列を掛けていく表記 (DirectX)

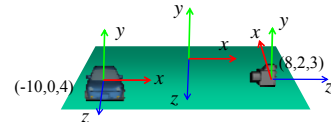


### 2次元空間でのアフィン変換

- 2次元空間(平面)でも、同様にアフィン変換は定義される
  - 3次元空間でのアフィン変換...4×4行列
  - 2次元空間でのアフィン変換...3×3行列
- 2次元空間のアフィン変換については、参考書を参照(本講義では扱わない)

### 座標変換の例

- 下記のシーンにおける、モデル座標系からカメラ座標系への変換行列を計算せよ
  - 物体の位置が (-10,0,4) にあり、ワールド座標系と同じ向き
  - カメラの位置が (8,2,3) にあり、ワールド座標系のY軸を中心として90度回転している



### 座標変換の例

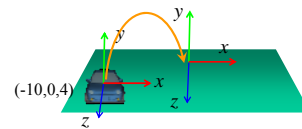
- 座標変換の考え方
  - モデル座標系 → ワールド座標系 への変換行列
  - ワールド座標系 → カメラ座標系 への変換行列
- この2つの変換を求めて、順に適用することで、モデル座標系 → カメラ座標系 への変換を実現
  - カメラやモデルの位置・向きは、ワールド座標系で表されているため、全体を一度に求めることは難しい

$$\begin{pmatrix} ? \\ ? \\ ? \\ 1 \end{pmatrix} \begin{pmatrix} ? \\ ? \\ ? \\ 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

ワールド → カメラ    モデル → ワールド

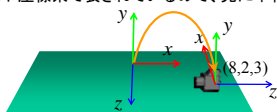
### 座標変換の例

- モデル座標系 → ワールド座標系
  - $$\begin{pmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
 平行移動のみ
  - 回転が必要であれば、平行移動行列の前に回転行列を適用する必要がある(今回は向きが同じなので不要)
  - モデル座標系の原点(0,0,0)はワールド座標系の(-10,0,4)に平行移動される



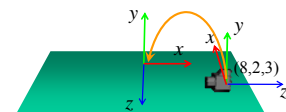
### 座標変換の例

- ワールド座標系 → カメラ座標系
  - $$\begin{pmatrix} \cos(-90^\circ) & 0 & \sin(-90^\circ) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-90^\circ) & 0 & \cos(-90^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -8 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
  - カメラの座標系から見てワールド座標系は、カメラの位置が(8,2,3)なので、ワールド座標系のY軸を中心として-90度回転    ワールド→カメラは(-8,-2,-3)
  - 位置はワールド座標系で表されているので、先に平行移動を適用



### 座標変換の例

- カメラ座標系 → ワールド座標系(参考)
  - $$\begin{pmatrix} 1 & 0 & 0 & 8 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(90^\circ) & 0 & \sin(90^\circ) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(90^\circ) & 0 & \cos(90^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
  - モデル座標系 → ワールド座標系と同様の行列になる
  - 回転 → 平行移動の順で適用、符号は反転の必要なし

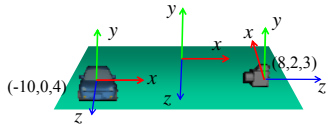


### 座標変換の例

- モデル座標系 → カメラ座標系

$$\begin{pmatrix} \cos(-90^\circ) & 0 & \sin(-90^\circ) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-90^\circ) & 0 & \cos(-90^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -8 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

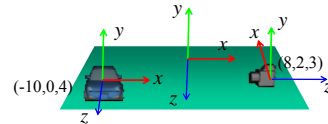
ワールド → カメラ                      モデル → ワールド



### 座標変換の順序に注意

- 回転と平行移動を適用する順序に注意！

$$\begin{pmatrix} \text{カメラ座標での平行移動} \\ \text{ワールド座標での回転} \end{pmatrix} \begin{pmatrix} \text{ワールド座標での平行移動} \\ \text{ワールド座標での回転} \end{pmatrix} \begin{pmatrix} \text{モデル座標での平行移動} \\ \text{モデル座標での回転} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

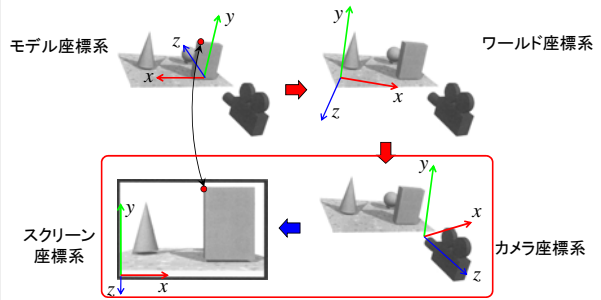


### 座標変換

- 座標変換の概要
- 座標系
- 視野変換(アフィン変換)
- 射影変換
- 座標変換のまとめ
- 演習問題

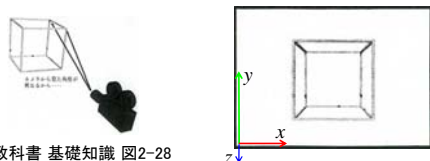
### 射影変換

- カメラ座標系からスクリーン座標系に変換



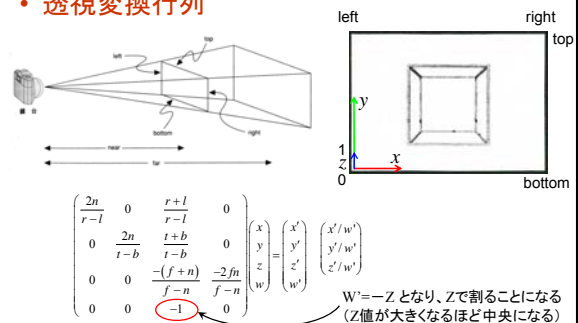
### 射影変換

- スクリーン座標を計算
- 透視変換
  - 一般的な射影変換の方法
  - 奥にあるものほど中央に描画されるように計算



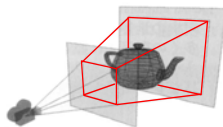
### 透視変換

- 透視変換行列



## クリッピング

- 視野内にないポリゴンは描画しない
- 背面除去も適用
  - カメラから後ろ向きのポリゴンは描画しない
- 背面除去・クリッピングの両方を適用

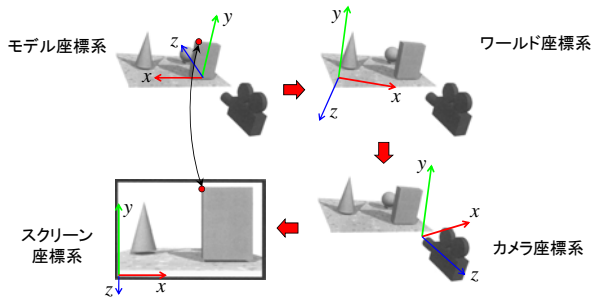


## 座標変換

- 座標変換の概要
- 座標系
- 視野変換(アフィン変換)
- 射影変換
- 座標変換のまとめ
- 演習問題

## 座標変換の流れのまとめ

- モデル座標系からスクリーン座標系に変換



## 変換行列による座標変換の実現

- アフィン変換+射影変換(透視変換)
  - 最終的なスクリーン座標は  $(x'/w' \ y'/w' \ z'/w')$

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} R_{00}S_x & R_{01} & R_{02} & T_x \\ R_{10} & R_{11}S_y & R_{12} & T_y \\ R_{20} & R_{21} & R_{22}S_z & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix}$$

透視変換 (カメラ→スクリーン)      アフィン変換 (モデル→カメラ)      モデル座標系での頂点座標  
 ↓      ↓      ↓  
 スクリーン座標系での頂点座標

## 座標変換の設定

- 自分のプログラムから OpenGLやDirectXに、2つの変換行列を設定する
  - ワールド座標からカメラ座標系への視野変換
    - カメラの位置・向きや、物体の位置向きに応じて、適切なアフィン変換行列を設定
    - さまざまな状況で、適切な変換行列を設定できるように、十分に理解しておく必要がある
  - カメラ座標系からスクリーン座標系への射影変換
    - 透視変換行列は、通常、固定なので、最初に一度だけ設定
    - 視野角やスクリーンサイズなどを適切に設定

## 射影変換の設定(サンプルプログラム)

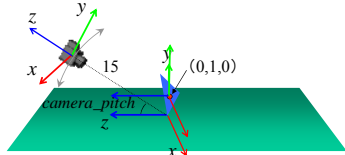
- ウィンドウサイズから変更された時に設定
  - 透視変換行列の設定(視野角を45度とする)
  - 各関数の詳細は、次回の演習で説明

```

void reshape( int w, int h )
{
    .....
    // カメラ座標系→スクリーン座標系への変換行列を設定
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluPerspective( 45, (double)w/h, 1, 500 );
}
    
```

### 変換行列の設定(サンプルプログラム)

- サンプルプログラムでのカメラ位置の設定



- 以下の変換行列により表せる(詳細は後日説明)

ポリゴンを基準とする座標系での頂点座標

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -15 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

カメラから見た頂点座標(描画に使う頂点座標)

### 変換行列の設定(サンプルプログラム)

- 描画処理の中で設定
  - 各関数の詳細は、次回の演習で説明

```
// 変換行列を設定(ワールド座標系→カメラ座標系)
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
glTranslatef( 0.0, 0.0, - 15.0 );
glRotatef( - camera_pitch, 1.0, 0.0, 0.0 );

// 地面を描画
.....

// 変換行列を設定(物体のモデル座標系→カメラ座標系)
glTranslatef( 0.0, 1.0, 0.0 );

// 物体(1枚のポリゴン)を描画
.....
```

### 座標変換

- 座標変換の概要
- 座標系
- 視野変換(アフィン変換)
- 射影変換
- 座標変換のまとめ
- 演習問題

### まとめ

- 変換行列
  - 座標変換の概要
  - 座標系
  - 視野変換(アフィン変換)
  - 透視変換
  - 座標変換のまとめ
  - 演習問題

### 次回予告

- 次回(講義)
  - 変換行列の復習・応用
  - OpenGLプログラミング
    - 変換行列の設定方法
- 次々回(演習)
  - 視点操作の拡張
  - 変換行列によるアニメーション

### 次回予定

- 来週(6月9日)は、文化講演会に振替
  - Moodleから感想を提出すること(毎回の演習問題と同様の扱い)
  - 締切 6/15(月)18:00
- 次回の授業は再来週(6月16日)