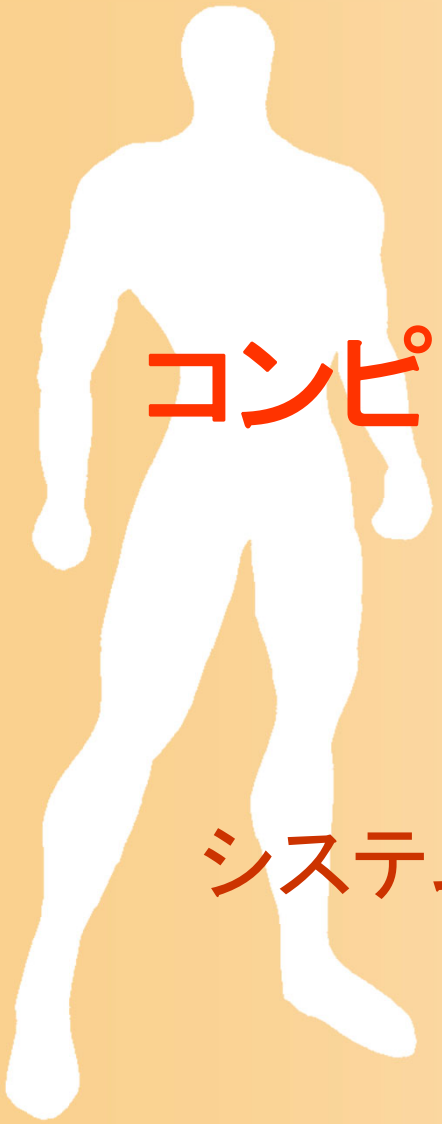


コンピューターグラフィックスS

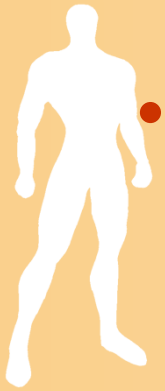
第8回　座標変換(1)

システム創成情報工学科　尾下 真樹

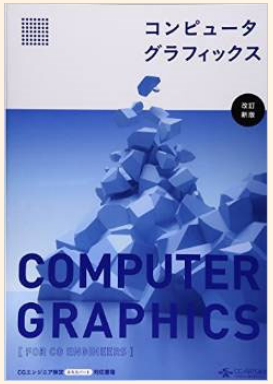
2019年度 Q2

今日の内容

- 前回の復習
- 座標変換の概要
- 座標系
- 視野変換
- 射影変換
- 座標変換のまとめ



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

講義資料

- 講義梗概
  - Moodleの本科目のコースで公開
  - 今回の講義のまとめ

コンピュータグラフィックスS 講義資料 第8回 座標変換（1）

九州工業大学 情報工学部 システム創成情報工学科 講義担当：尾下真樹

1. 座標変換

座標変換は、コンピュータグラフィックスにおける重要な基礎技術である。座標変換にはさまざまな応用があるが、特に、コンピュータグラフィックスにおいては、レンダリングの際に、モデル座標系で表された頂点座標を、スクリーン座標系での座標に変換するために用いられる。

座標変換において特に重要になるのが、モデル座標系からカメラ座標系への変換を実現するための視野変換であり、回転・平行移動などのアフィン変換を、同次座標変換と呼ばれる方法を用いて実現する。

同次座標変換では、3次元空間の頂点座標を (x, y, z, w) の4次元の同次座標ベクトル（通常は $w=1$）で表し、4×4 行列 $\mathbf{A}$ をかけることで座標変換を適用する。座標変換は、右のような式で書ける。

$$\mathbf{A}\begin{pmatrix}x\\y\\z\\w\end{pmatrix}=\begin{pmatrix}x'\\y'\\z'\\w'\end{pmatrix}$$

適切な行列 $\mathbf{A}$ を用いることでさまざまな変換が実現できるが、以下では、特に重要となる回転・平行移動とその組み合わせについてまとめる。（その他の変換については、講義での説明、講義スライド、教科書を参照のこと。）

1.1. 回転行列

以下のような行列（左上3×3成分に回転成分に適切な値を設定した行列）を用いることで、回転を適用できる。

$$\begin{pmatrix}R_{xx}&R_{xy}&R_{xz}&0\\R_{yx}&R_{yy}&R_{yz}&0\\R_{zx}&R_{zy}&R_{zz}&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}x'\\y'\\z'\\1\end{pmatrix}$$

具体的な行列は

x軸周りの回転変換 $\begin{pmatrix}1&0&0&0\\0&\cos\theta&-\sin\theta&0\\0&\sin\theta&\cos\theta&0\\0&0&0&1\end{pmatrix}$　y軸周りの回転変換 $\begin{pmatrix}\cos\theta&0&\sin\theta&0\\0&1&0&0\\-\sin\theta&0&\cos\theta&0\\0&0&0&1\end{pmatrix}$　z軸周りの回転変換 $\begin{pmatrix}\cos\theta&-\sin\theta&0&0\\\sin\theta&\cos\theta&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$ など。

回転行列の要素は、平面における回転後の座標を考えることで、求めることができる。例えばx軸周りの回転であれば、yz平面上における、(0,1,0)や(0,0,1)の回転後の座標を考えれば、行列を求めることができる。

1.2. 平行移動行列

以下のような行列（右端の列に適切な値を設定した行列）を用いることで、座標を (T_x, T_y, T_z) 動かす平行移動を適用できる。

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

左辺を計算すると $\begin{pmatrix}1&0&0&T_x\\0&1&0&T_y\\0&0&1&T_z\\0&0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}x+T_x\\y+T_y\\z+T_z\\1\end{pmatrix}$ となり、平行移動が適用されていることが分かる。

1.3. 複数の行列の組み合わせ

以下のように、複数の行列をかけることで、複数の平行移動や回転を組み合わせて適用することができる。このとき、変換行列は、右に書かれた行列から左に書かれた行列の順番（$\mathbf{A}_n$ から $\mathbf{A}_1$ の順番）で適用されることに注意する。

$$\mathbf{A}_n\cdots\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1\begin{pmatrix}x\\y\\z\\w\end{pmatrix}=\begin{pmatrix}x'\\y'\\z'\\w'\end{pmatrix}$$

このとき、行列の積を最初に計算して1つの行列を求めておけば、その行列をかけるだけで複数の変換をまとめて適用できる。また、ある座標系 C1 から別の座標系 C2 への変換行列 $\mathbf{A}$ があるとき、逆行列 $\mathbf{A}^{-1}$ を求めることで、座標系 C2 から座標系 C1 への逆方向の座標変換を得ることができる。

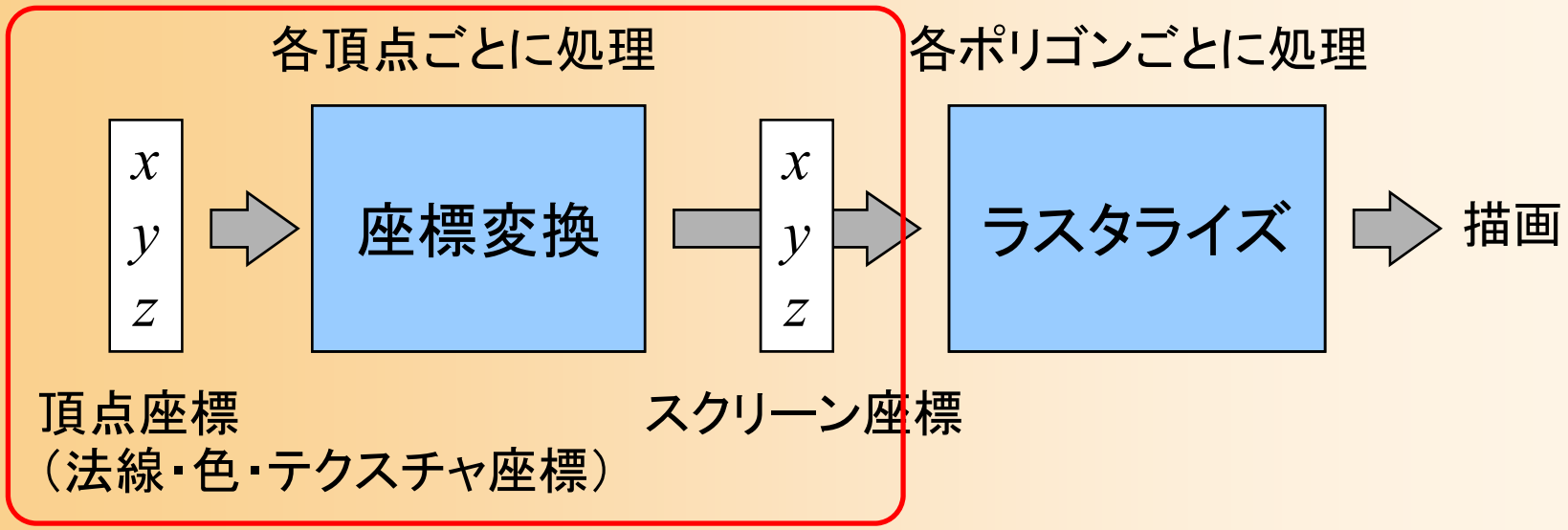
例えば、モデルが (0,0,5) の位置にあり、ワールド座標系の y 軸を中心として 90 度回転している向きにあるとき、モデル座標系からワールド座標系への変換行列（モデルを適切な位置に回転・移動する変換行列）による座標変換は、以下のように書ける。

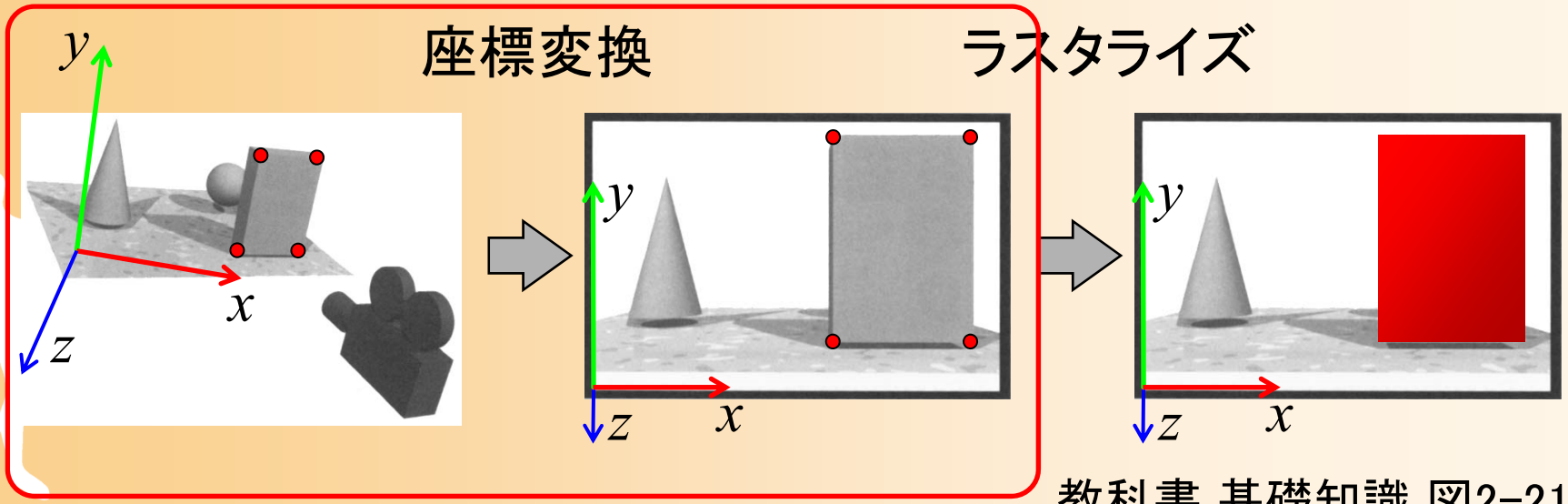
$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&5\\0&0&0&1\end{pmatrix}\begin{pmatrix}\cos90^\circ&0&\sin90^\circ&0\\0&1&0&0\\-\sin90^\circ&0&\cos90^\circ&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}x'\\y'\\z'\\1\end{pmatrix}$$

この場合、回転変換を先に適用することに注意する。（先に平行移動を適用すると、平行移動にまで次の回転が適用され、(0,0,5) ではなく (5,0,0) の位置にモデルが移動することになるため。）また、2 つの行列の順番、及び、平行移動成分・回転成分の符号を逆にすると、もとの行列の逆行列になるため、逆方向の変換であるワールド座標系からモデル座標系への座標変換列が求まる。

前回の復習

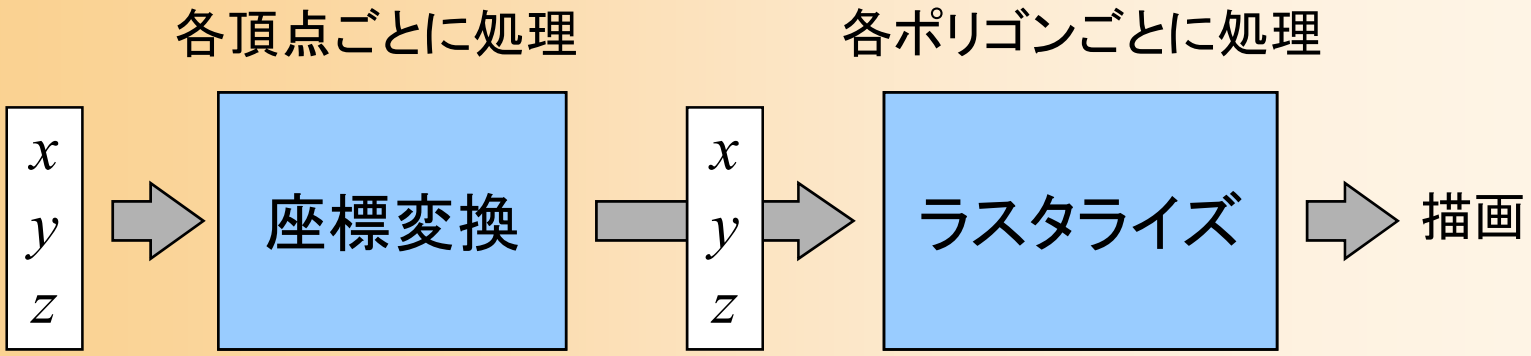
レンダリング・パイプライン

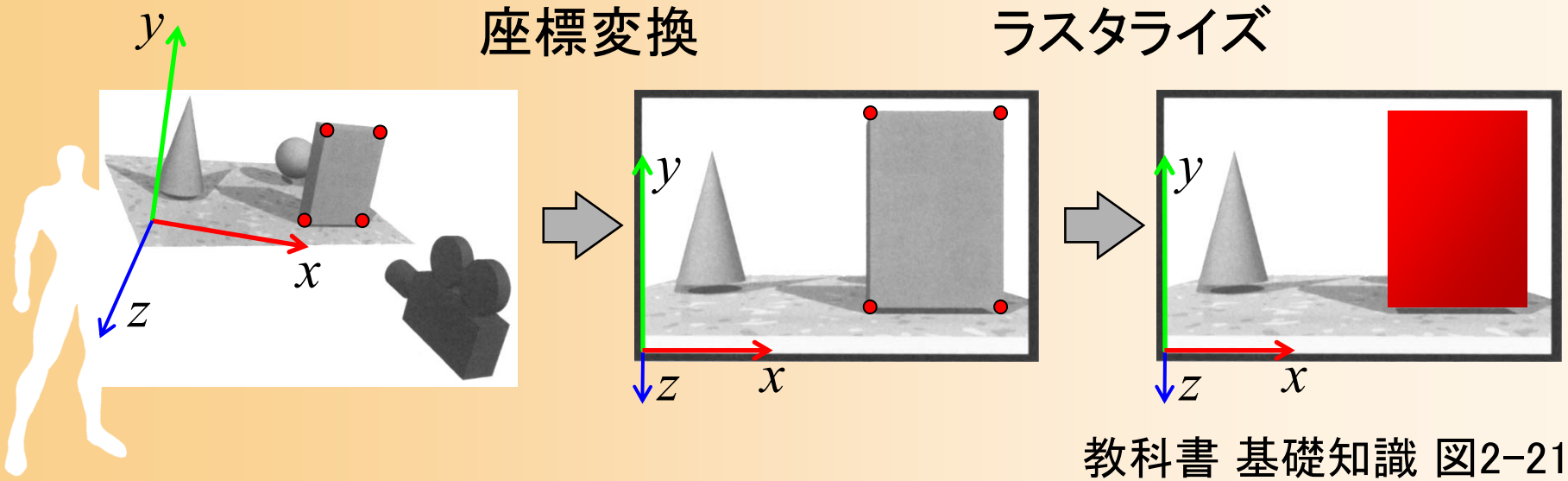




教科書 基礎知識 図2-21

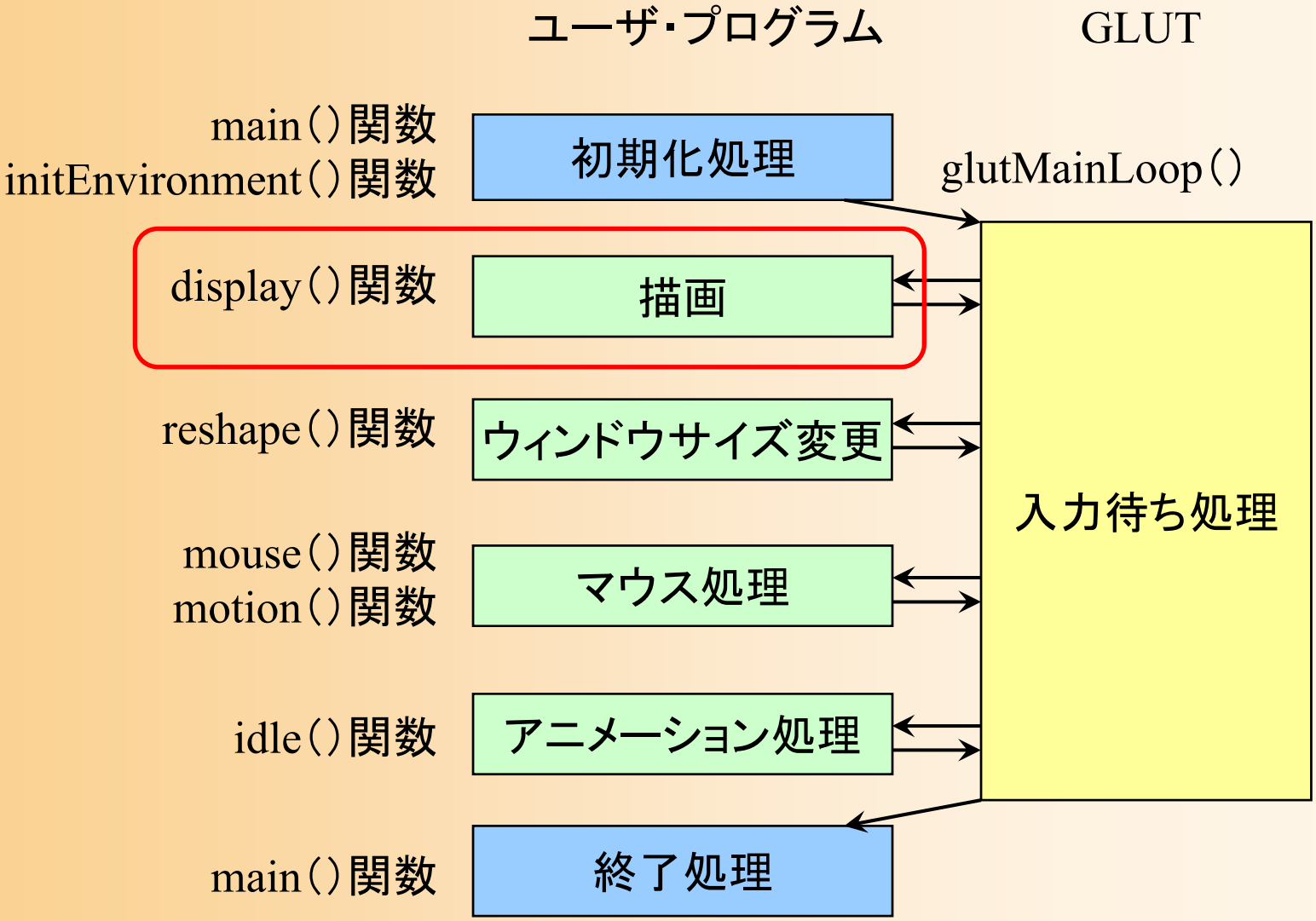
処理の流れ





教科書 基礎知識 図2-21

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  renderPyramid3()
{
    int  i, j, v_no;
    glBegin( GL_TRIANGLES );
    for ( i=0; i<num_pyramid_triangles; i++ )
    {
        glNormal3f( pyramid_tri_normals[i][0], [i][1], [i][2] );
        for ( j=0; j<3; j++ )
        {
            v_no = pyramid_tri_index[ i ][ j ];
            glVertex3f( pyramid_vertices[ v_no ][0], [ v_no ][1], 
        }
    }
    glEnd();
}
```

各三角面ごとに繰り返し

三角面の各頂点ごとに繰り返し

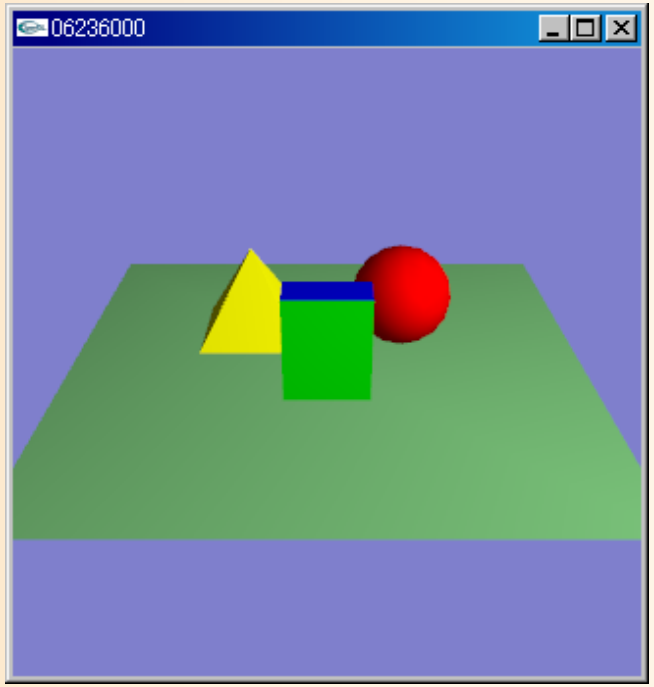
面の法線を指定（i番目の面のデータを指定）

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）

頂点座標を指定（v_no番目の頂点のデータを指定）

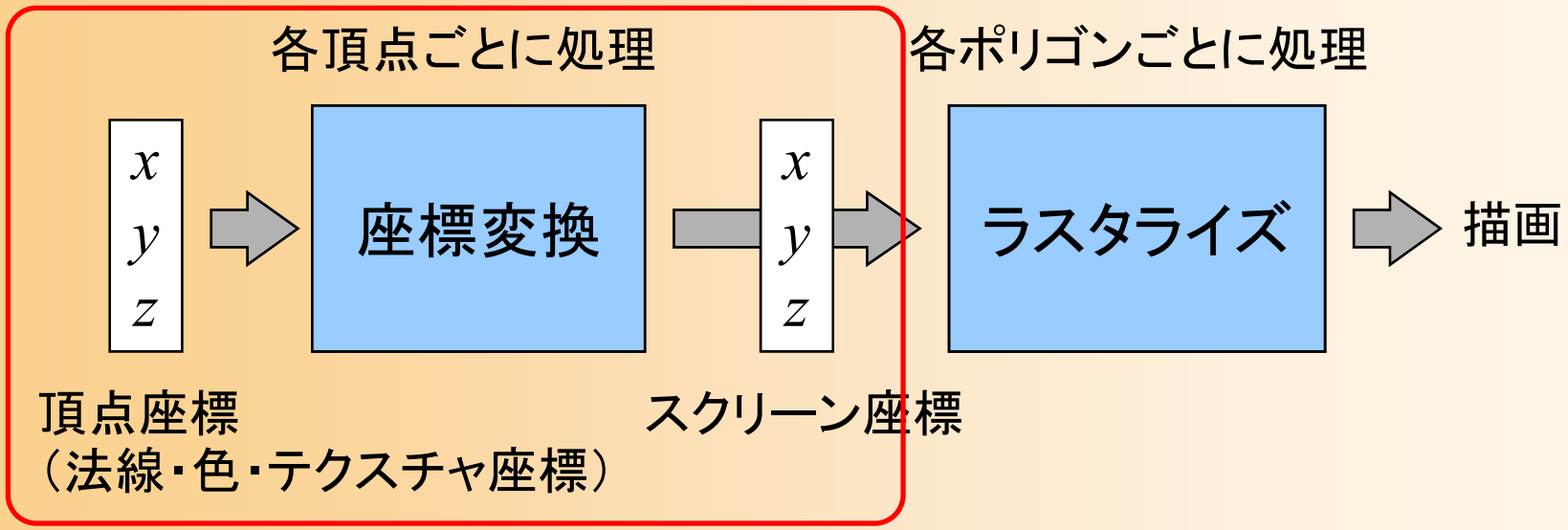
前回の演習

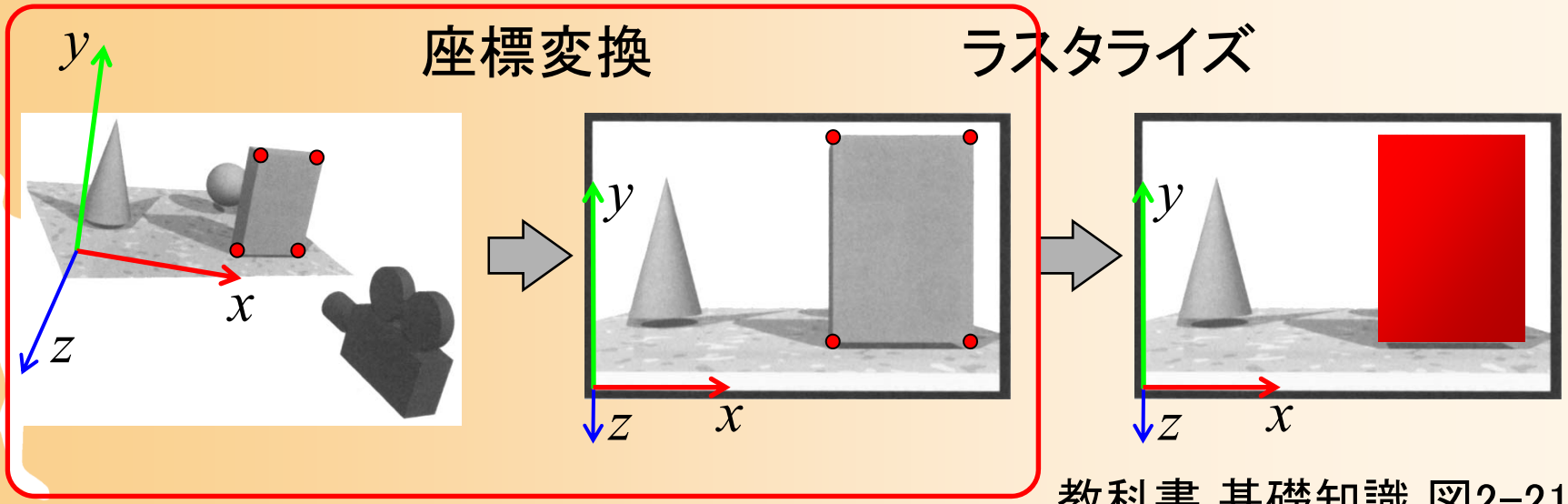
- ポリゴンの描画方法（復習）
- 基本オブジェクトの描画
- ポリゴンモデルの描画
- 演習課題



座標変換の概要

座標変換

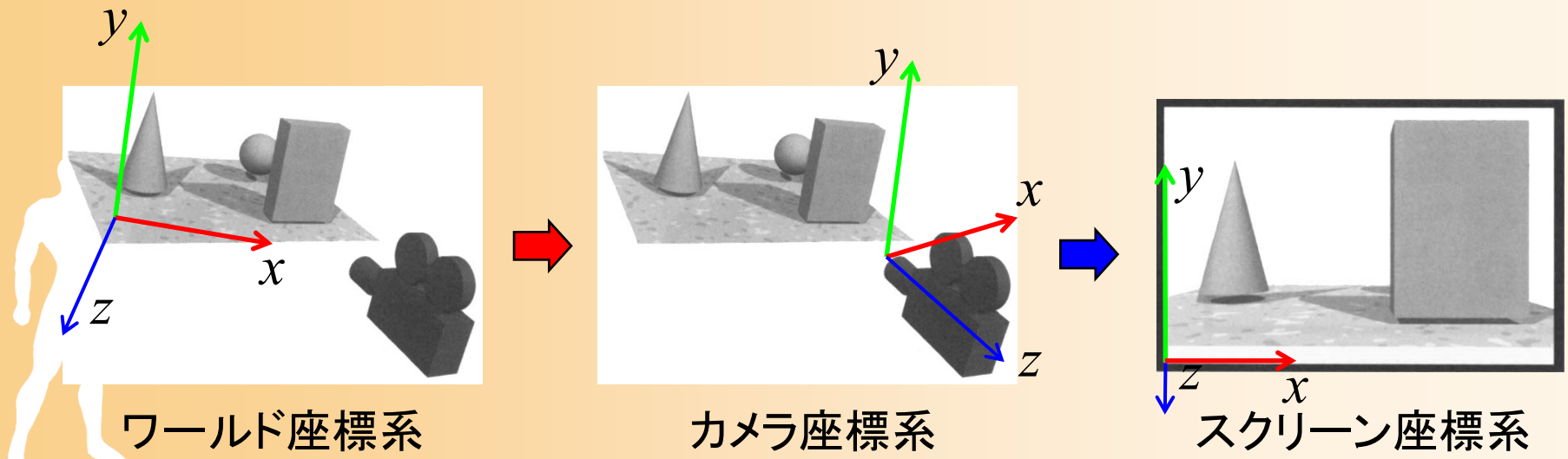




教科書 基礎知識 図2-21

座標変換

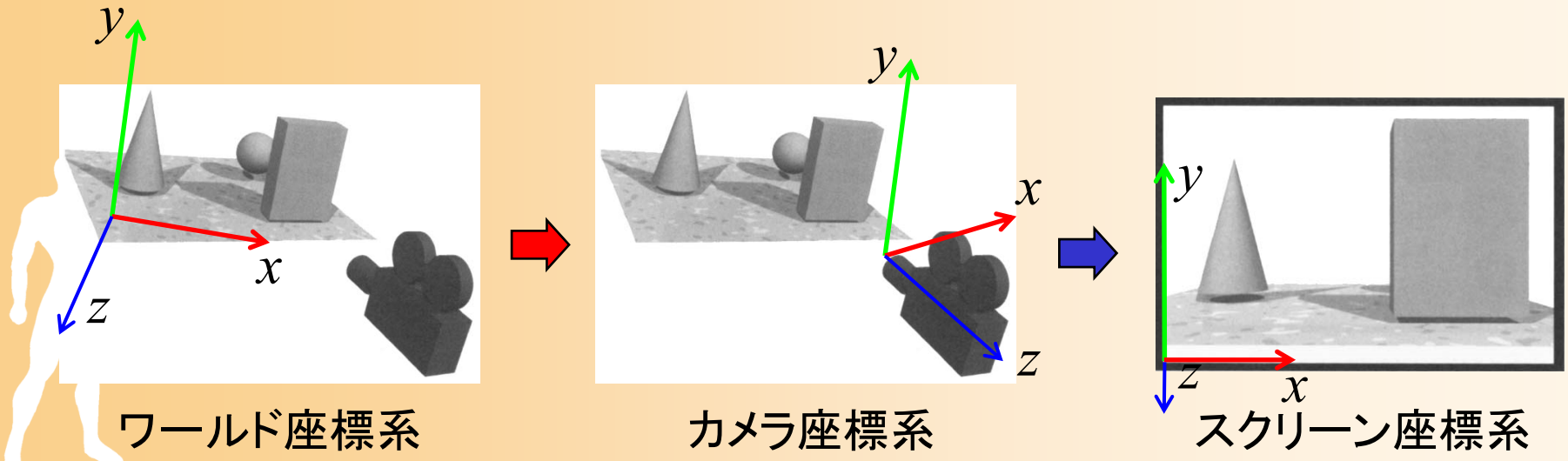
- 座標変換
  - ワールド座標系（モデル座標系）で表された頂点座標を、スクリーン座標系での頂点座標に変換する



ワールド座標系　　カメラ座標系　　スクリーン座標系

座標変換

- 2段階の座標変換により実現
  - ワールド座標からカメラ座標系への視野変換
  - カメラ座標系からスクリーン座標系への射影変換
    - 行列計算（同次座標変換）によって、上記の2種類の変換を実現する



ワールド座標系　　カメラ座標系　　スクリーン座標系

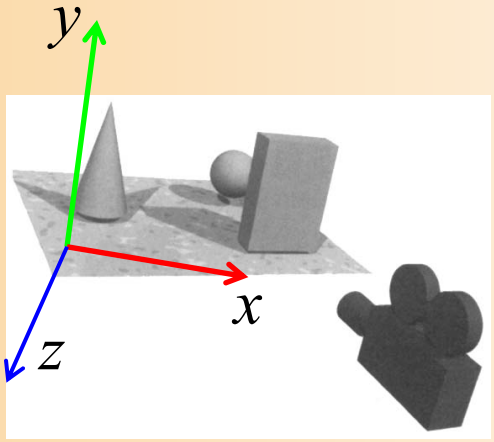
座標系

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

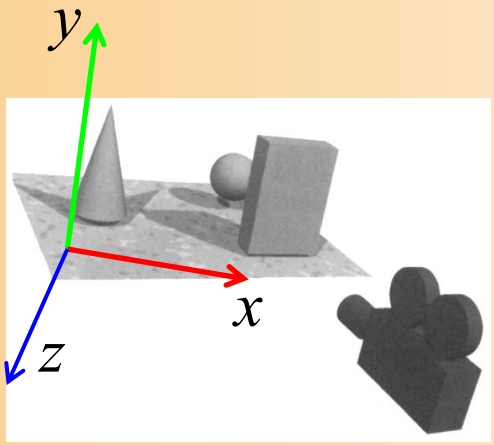
- 3次元空間の座標系
  - 物体や光源やカメラなどを配置する座標系
  - 原点や軸方向は適当にとって構わない
    - カメラと描画対象の相対位置・向きのみが重要
  - 単位も統一さえされていれば自由に設定して構わない（メートル、センチ、etc）



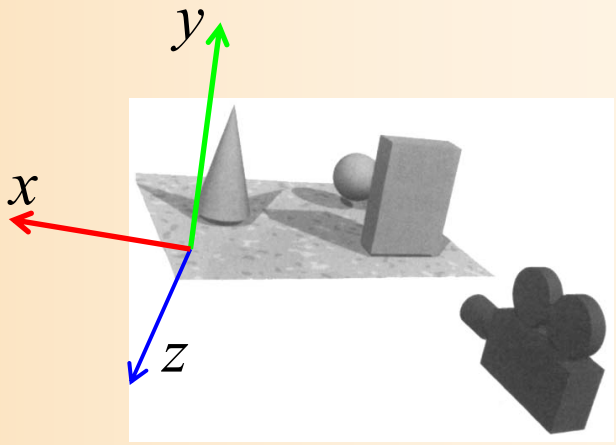
ワールド座標系

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



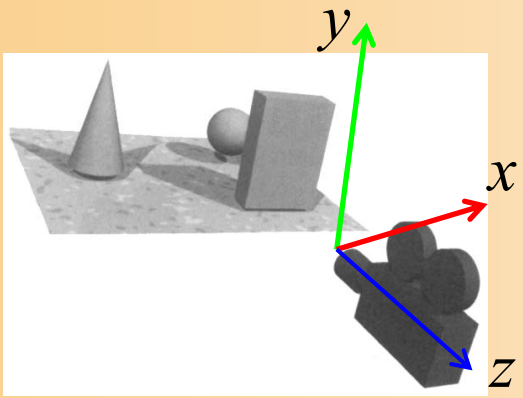
左手座標系

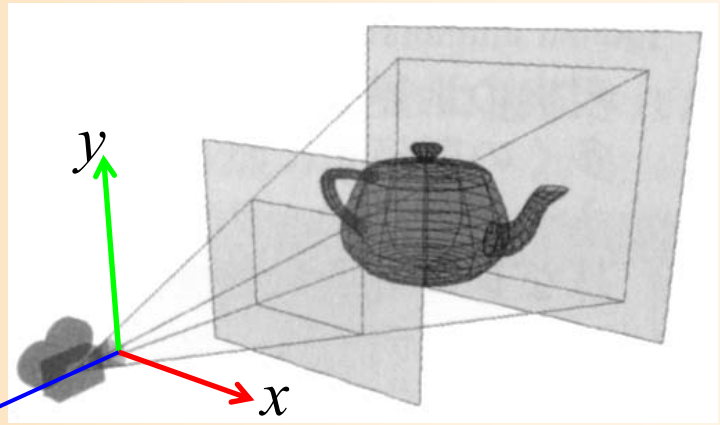
右手座標系と左手座標系（続き）

- 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

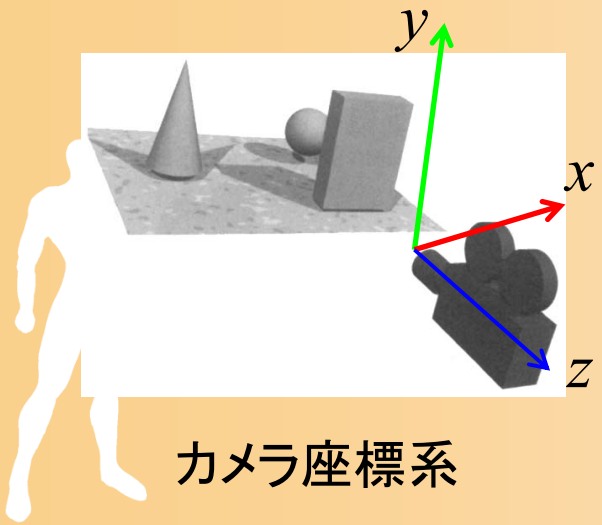




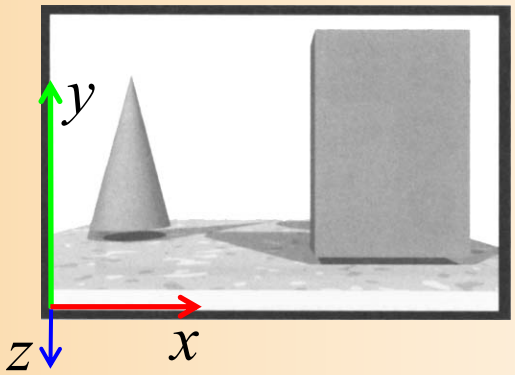
カメラ座標系

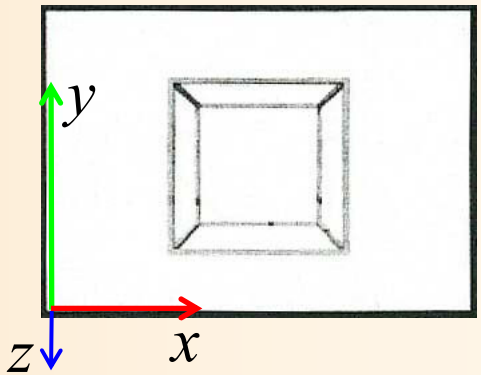
スクリーン座標系

- 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



カメラ座標系

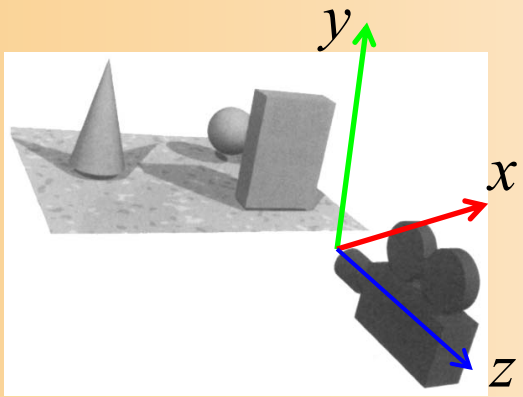




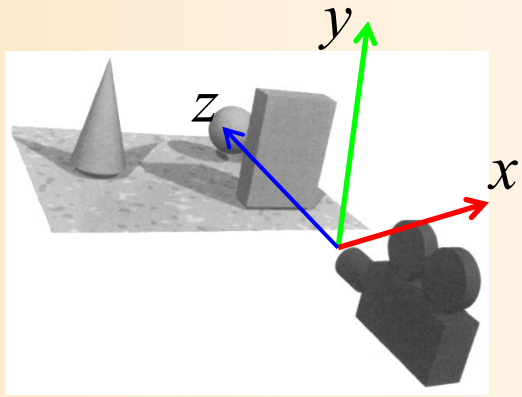
スクリーン座標系

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



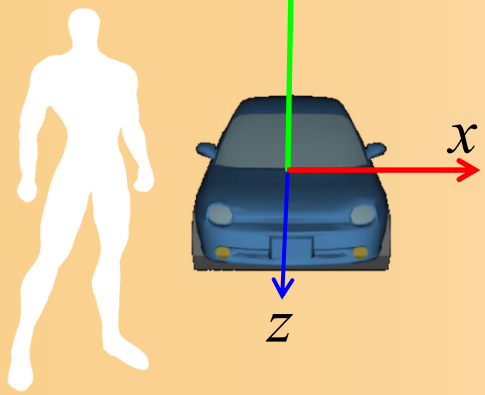
手前がZ軸の正方向

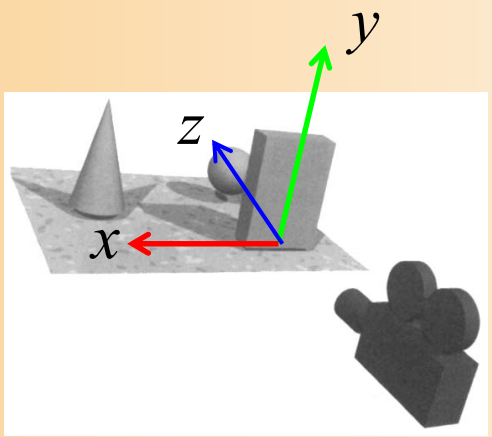


奥がZ軸の正方向

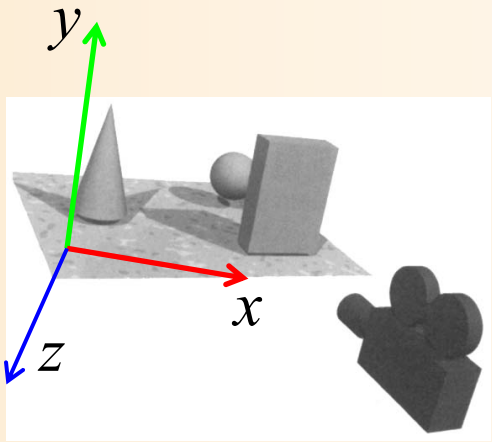
モデル座標系

- 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





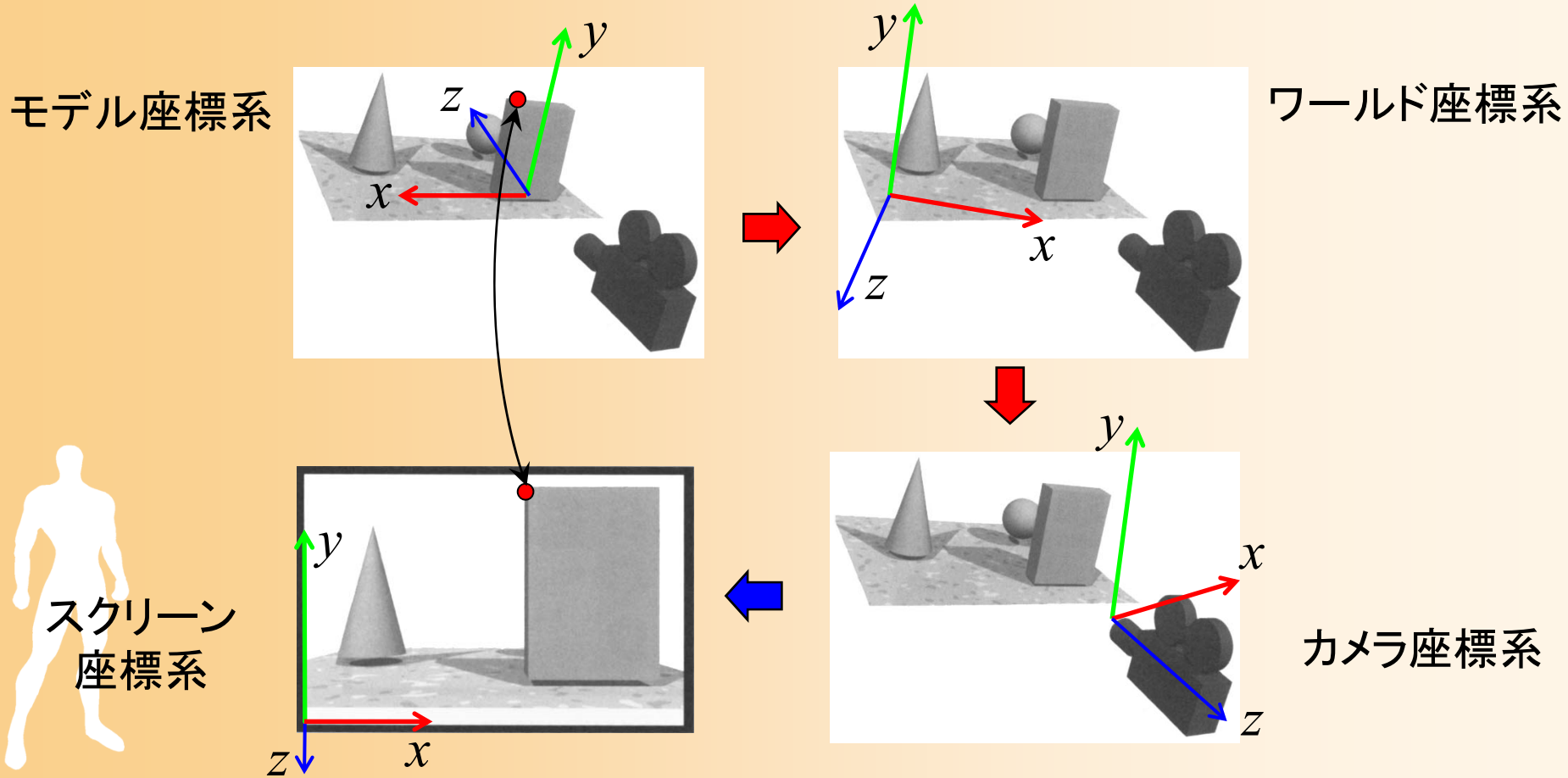
モデル座標系



ワールド座標系

座標変換の流れ（詳細）

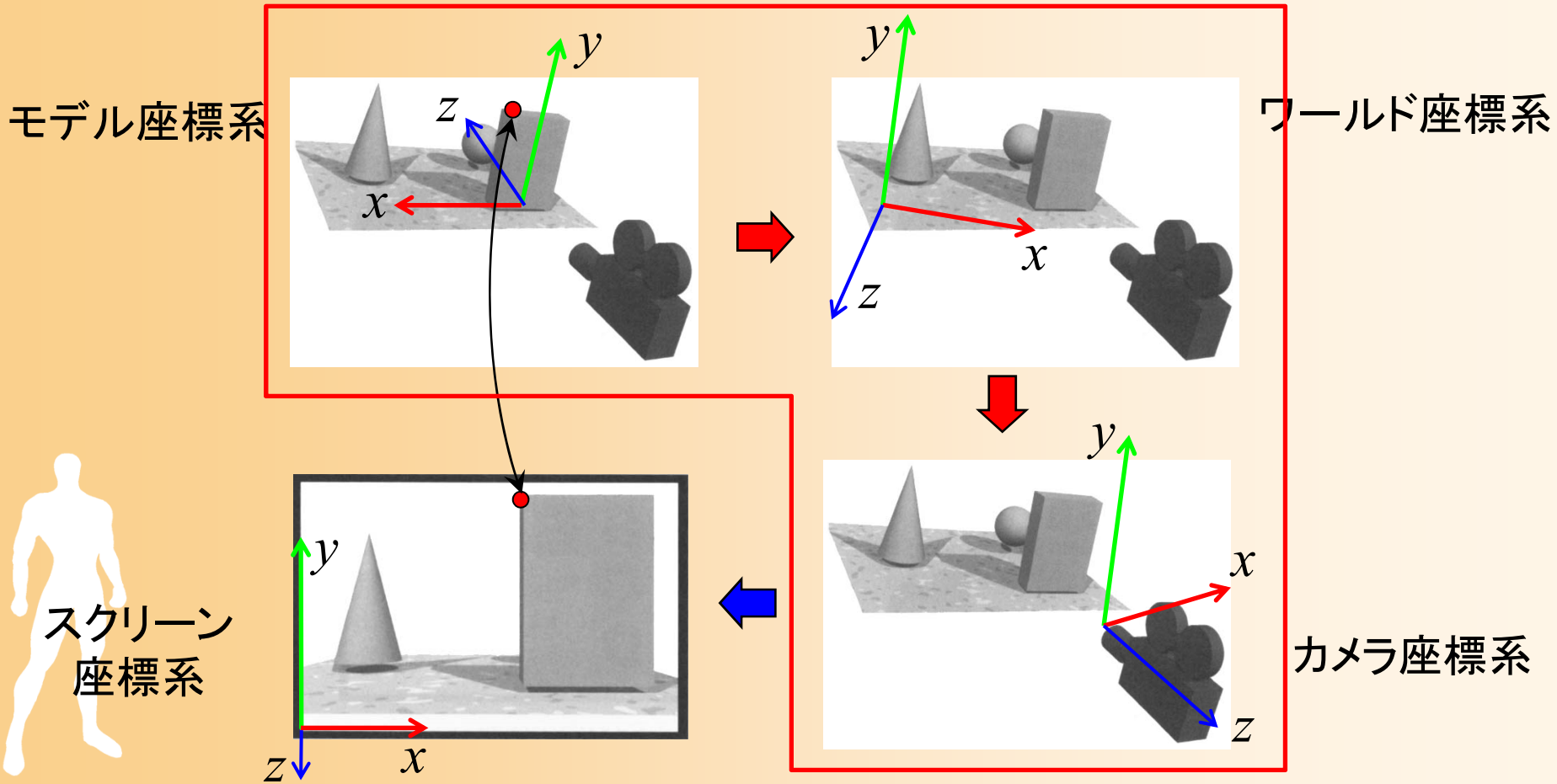
- モデル座標系からスクリーン座標系に変換



視野変換

視野変換

- モデル座標系からカメラ座標系に変換



同次座標変換

- 同次座標変換
  - ４×４行列の演算により、３次元空間における平行移動・回転・拡大縮小（アフィン変換）などの操作を統一的に実現
    - (x, y, z, w)の４次元座標値（同次座標）を扱う
    - ３次元座標値は$(x/w, y/w, z/w)$で計算（通常は $w = 1$）

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

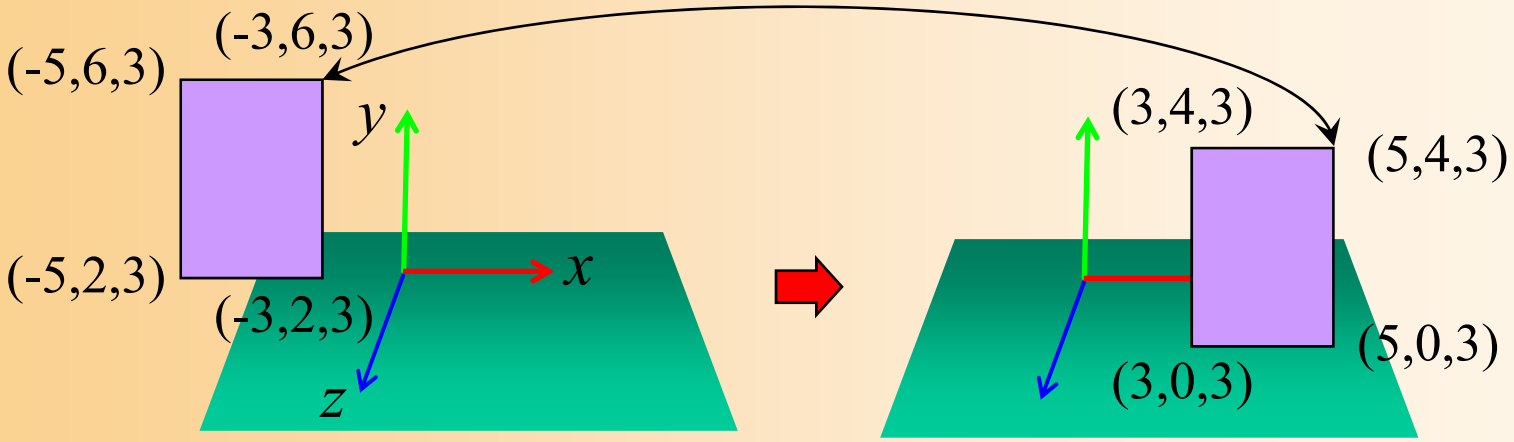
平行移動

- 平行移動
  - (Tx,Ty,Tz)の平行移動
    - 4×4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



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

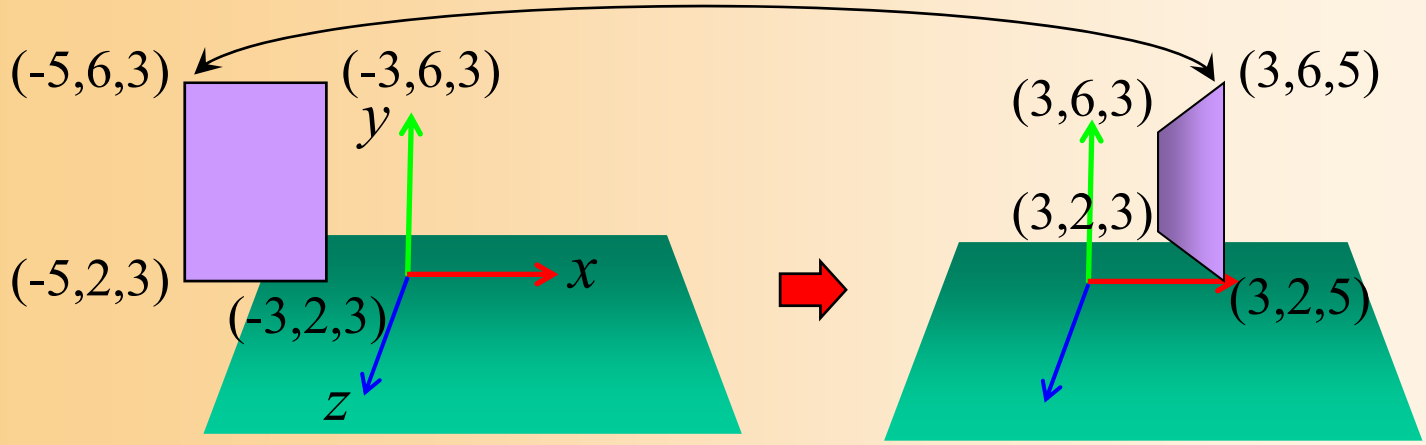
回転

- 回転
  - 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

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

X軸を中心とする回転変換

$$\begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Y軸を中心とする回転変換

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

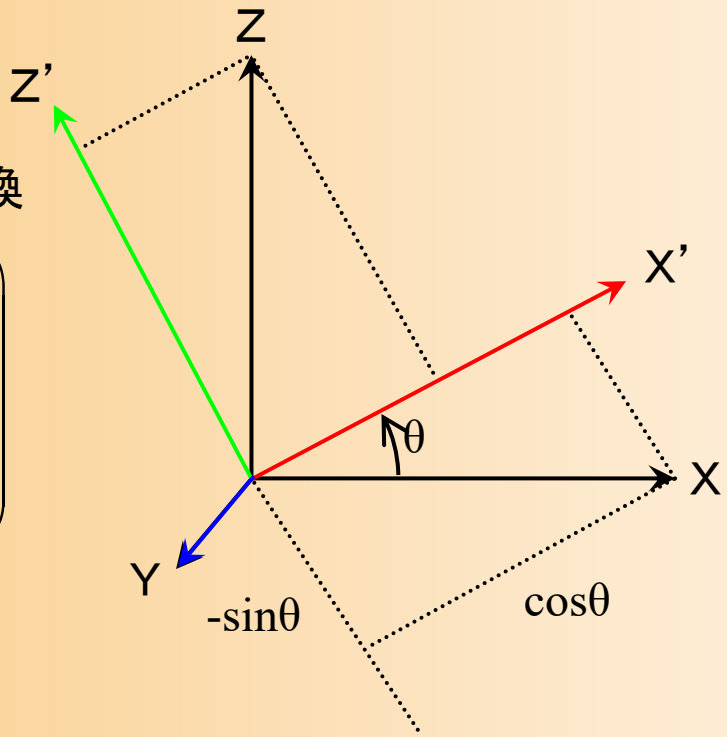
Z軸を中心とする回転変換

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

Y軸を中心とする回転変換

$$\begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



変換前のX軸・Z軸方向の単位ベクトルの、変換後の座標系での座標

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

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

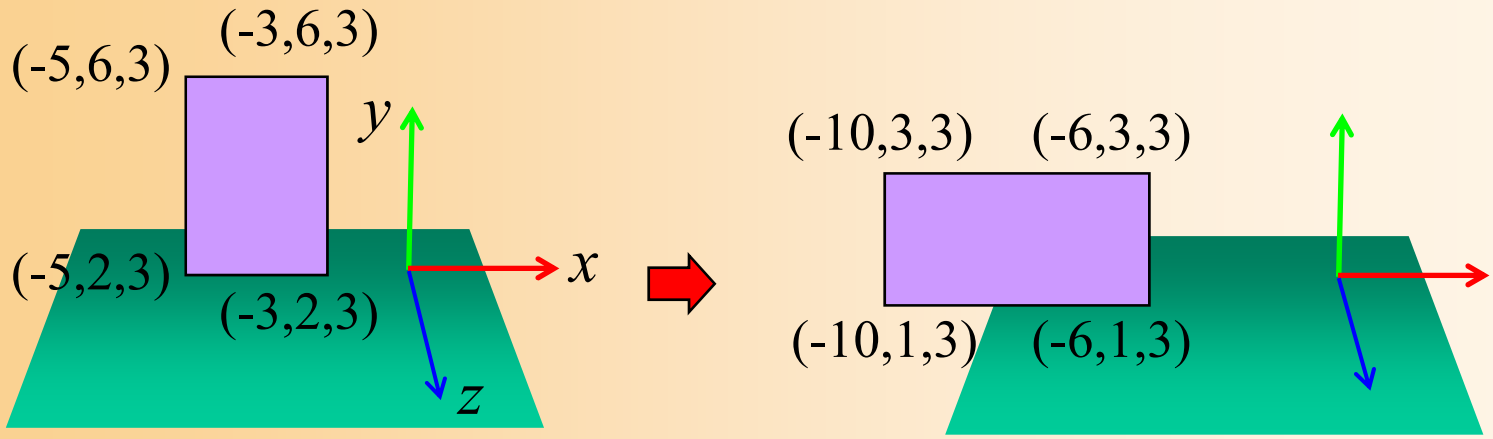
拡大縮小

- 拡大縮小
  - (Sx,Sy,Sz)倍のスケーリング

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

拡大縮小の例

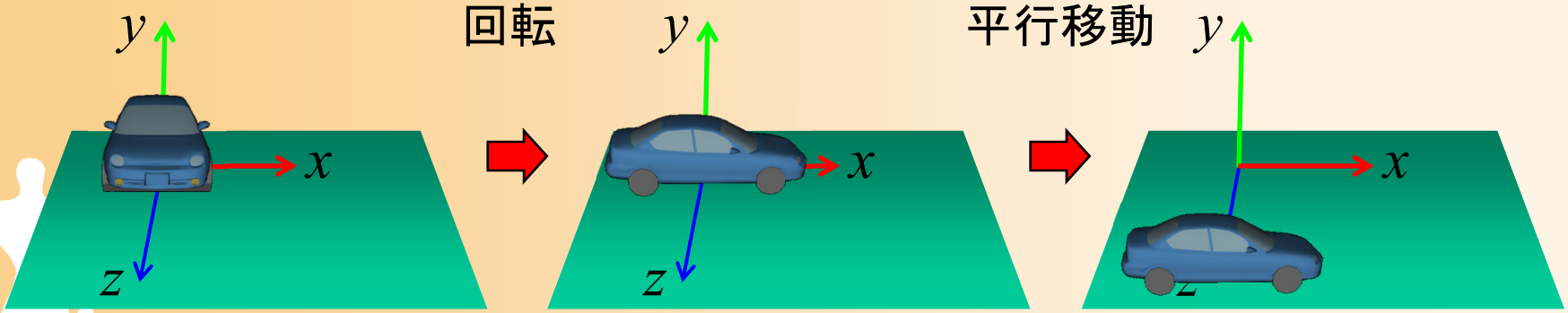
- (2, 0.5, 1)倍に拡大縮小



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

変換行列の適用

- 1つの行列かけ算で各種の変換を適用可能
- 複数の行列を順番にかけていくことで、複数の変換を連続して適用できる
  - 回転･移動の組み合わせの例

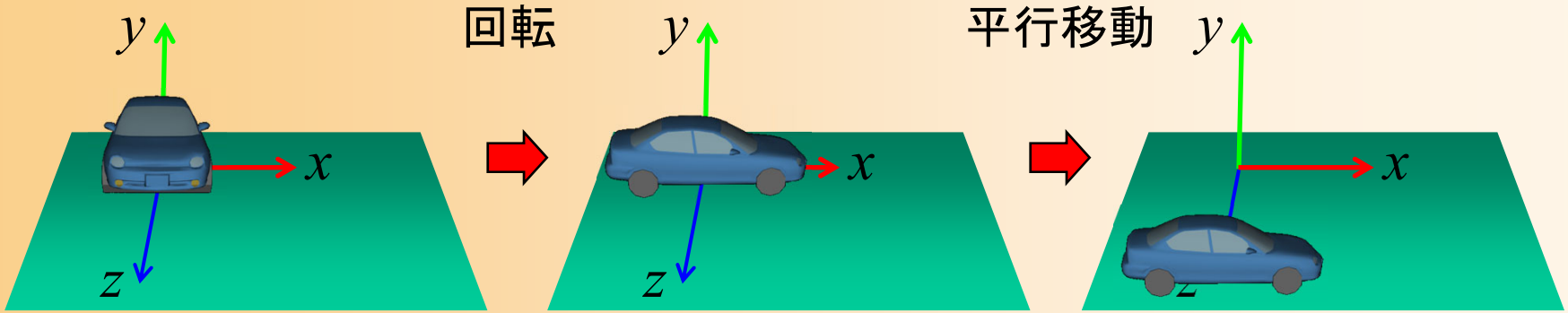


$$\begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

複数の変換行列の適用例

- 回転･移動の組み合わせの例



平行移動　　回転

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

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

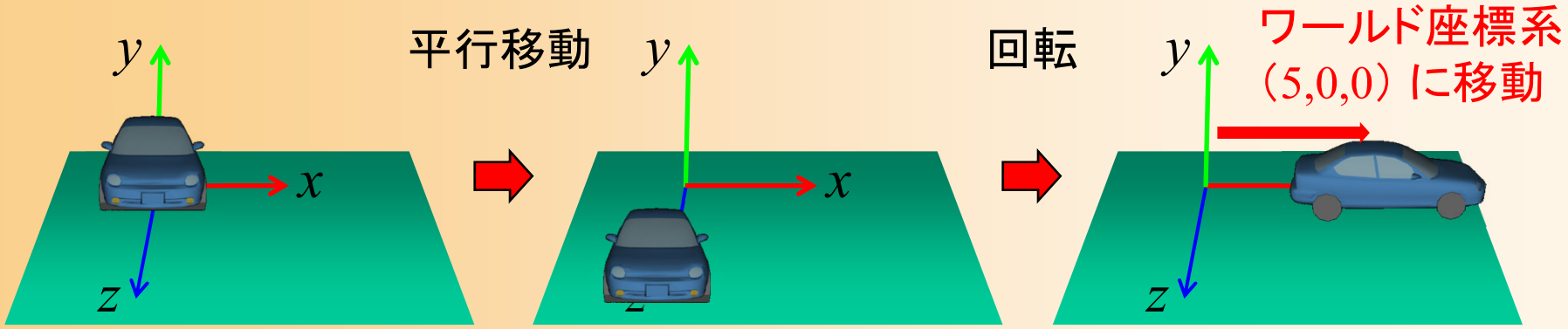
- 行列演算では可換則は成り立たないことに注意！

$$AB \neq BA$$

- 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

複数の変換行列の適用例(2)

- 移動→回転の順番で適用したときの例



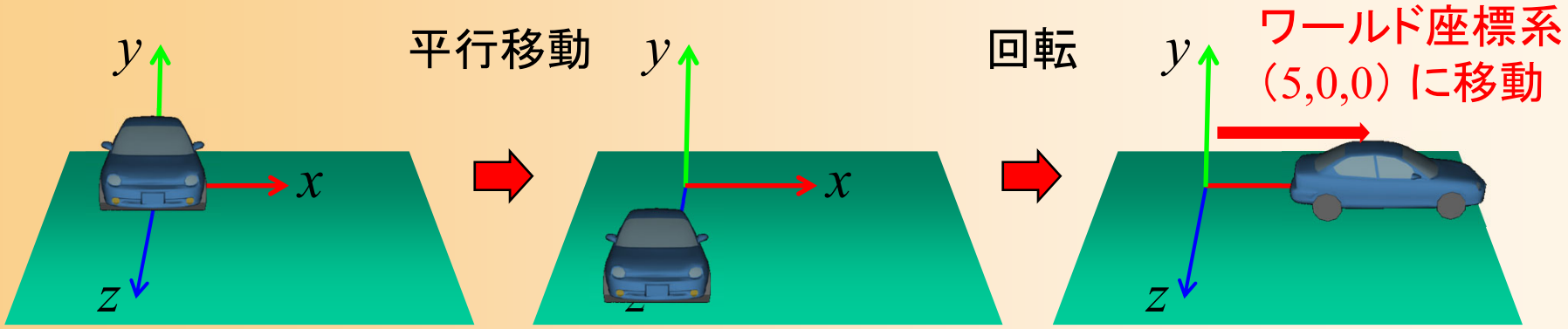
回転　　平行移動

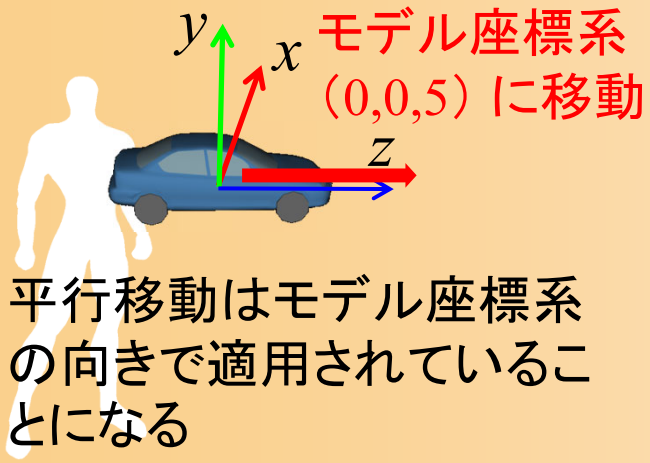
$$\begin{pmatrix} \cos 90^\circ & 0 & \sin 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix}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

この場合は平行移動成分にも回転がかかる

複数の変換行列の適用例(2)

- 移動→回転の順番で適用したときの例





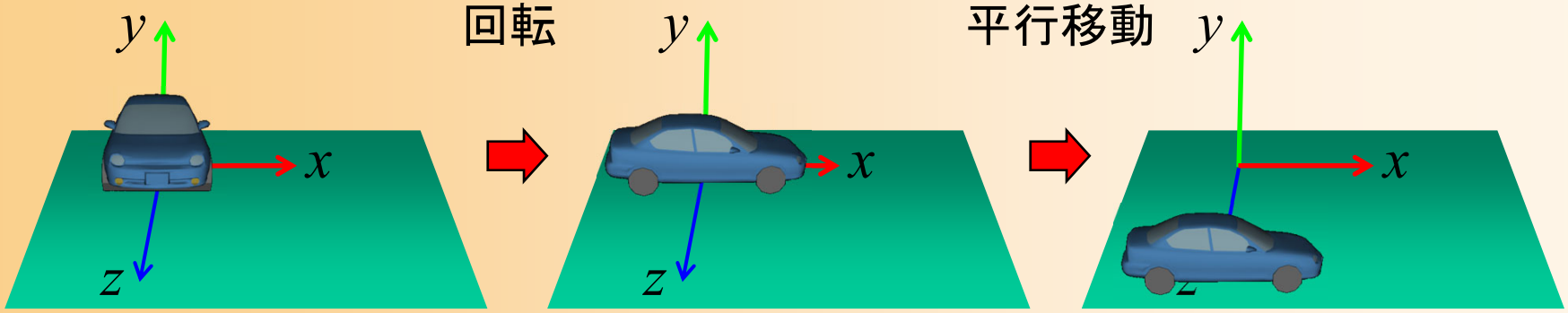
平行移動はモデル座標系の向きで適用されていることになる

回転　　平行移動

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

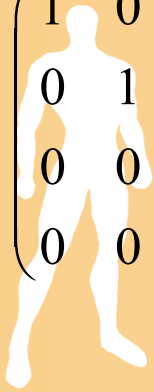
複数の変換行列の適用例(1)

- 回転→移動の順番で適用(さきほどの例)



平行移動　　回転

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



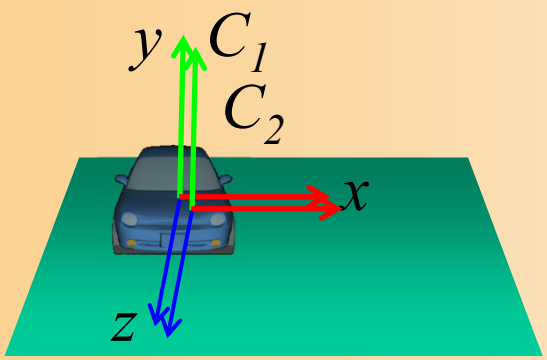
こちらの順番の方が普通に使う場合が多い

座標変換の考え方

- 座標変換の考え方
  - 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

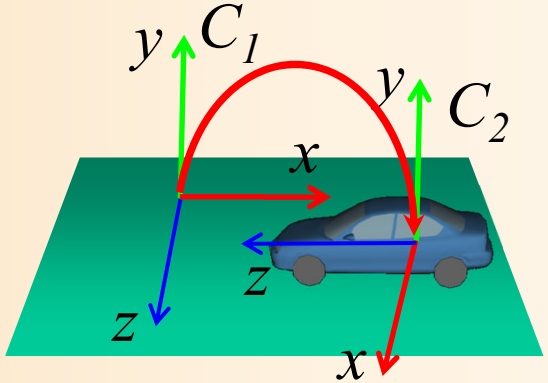
モデルをC1→C2に移動・回転 ＝
C2→C1の変換行列を求める

変換行列を適用しない状態では、
移動や回転はなし



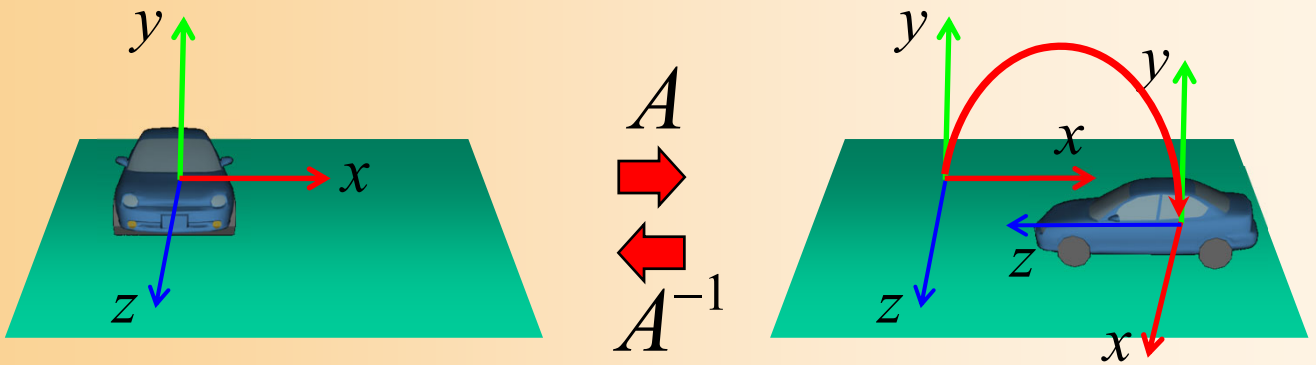
回転→移動の
変換を適用

A



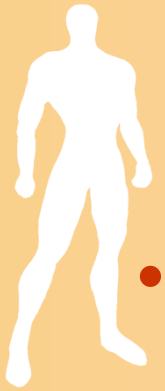
座標変換の逆変換

- 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（回転・平行移動・拡大縮小）の行列は、正則であるため、常に逆行列が存在する



同次座標変換のメリット

- 行列演算だけでさまざまな処理を行える
  - 同次座標変換を使わずとも、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の方が高速
  - 各種処理を統一的に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



同次座標変換の表記方法

- 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）

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}^t \begin{pmatrix} R_{00} & R_{10} & R_{20} & 0 \\ R_{01} & R_{11} & R_{21} & 0 \\ R_{02} & R_{12} & R_{22} & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

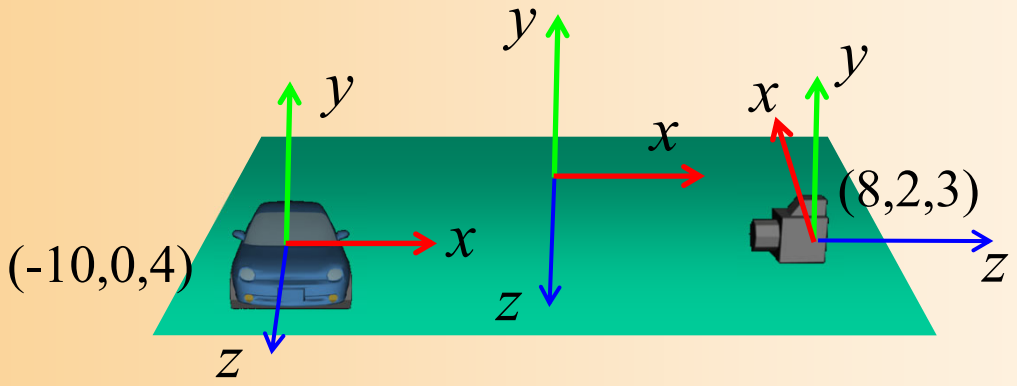
右から行列を掛けていく表記（DirectX）

2次元空間での同次座標変換

- 2次元空間（平面）でも、同様に同次座標変換は定義される
  - 3次元空間での同次座標変換・・・4×4行列
  - 2次元空間での同次座標変換・・・3×3行列
- 2次元空間の同次座標変換については、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

  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

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

ワールド → カメラ　モデル → ワールド

座標変換の例

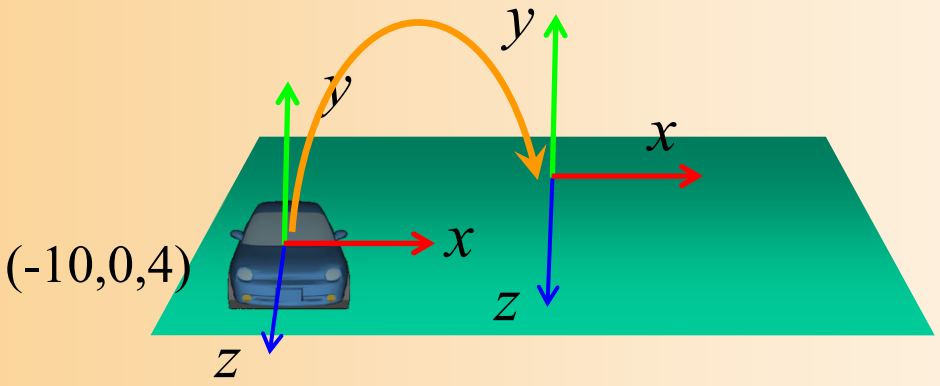
- モデル座標系 → ワールド座標系

$$\begin{pmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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

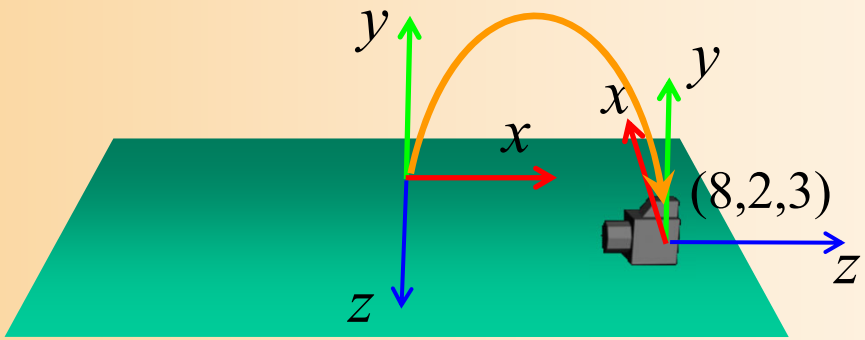
- ワールド座標系 → カメラ座標系

$$\begin{pmatrix} \cos(-90^\circ) & 0 & \sin(-90^\circ) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-90^\circ) & 0 & \cos(-90^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -8 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

カメラの座標系から見てワールド座標系は、
ワールド座標系のY軸を中心に -90 度回転

カメラの位置が(8,2,3)なので、
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



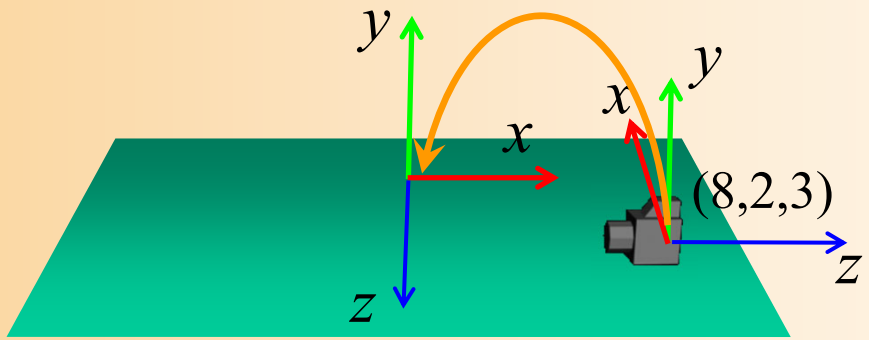
座標変換の例

- 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

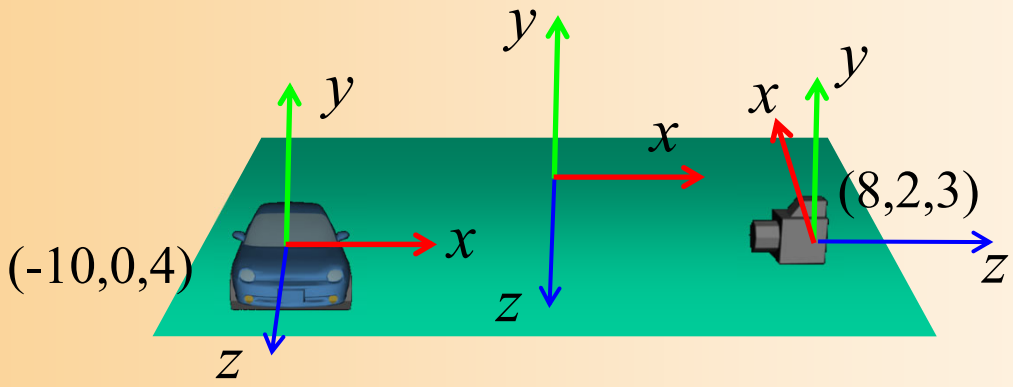


座標変換の例

- モデル座標系 → カメラ座標系

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

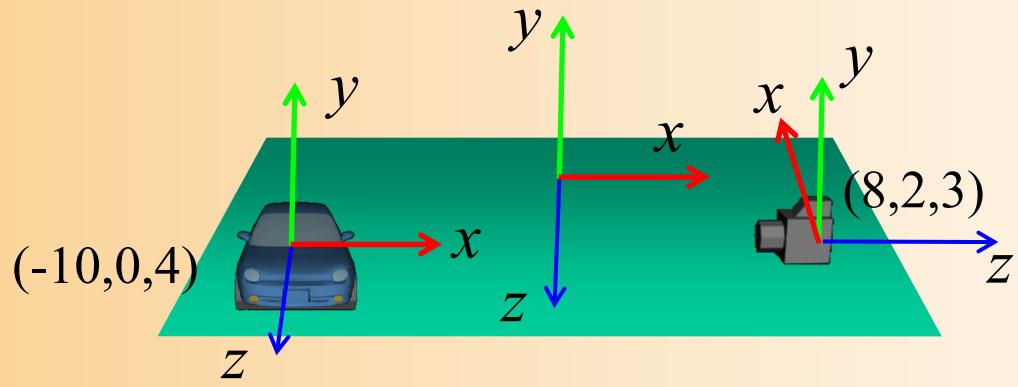
ワールド → カメラ　　モデル → ワールド



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！

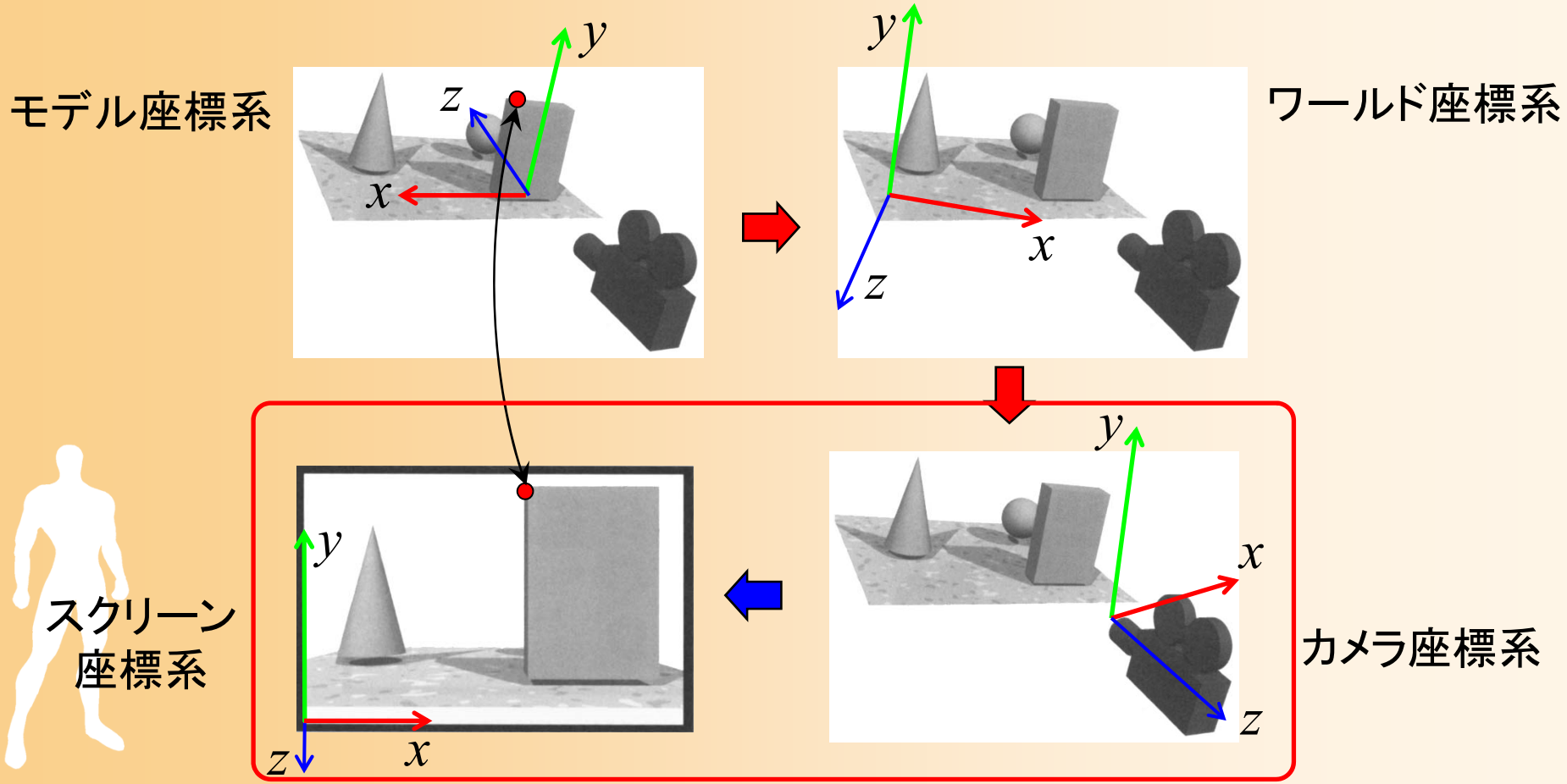
$$\left(\begin{array}{c}\text{カメラ}\\ \text{座標での}\\ \text{平行移動}\end{array}\right)\left(\begin{array}{c}\text{ワールド}\\ \rightarrow\text{カメラ}\\ \text{の回転}\end{array}\right)\left(\begin{array}{c}\text{ワールド}\\ \text{座標での}\\ \text{平行移動}\end{array}\right)\left(\begin{array}{c}\text{モデル}\rightarrow\\ \text{ワールド}\\ \text{の回転}\end{array}\right)\left(\begin{array}{c}\text{モデル}\\ \text{座標での}\\ \text{平行移動}\end{array}\right)\begin{pmatrix}x\\ y\\ z\\ 1\end{pmatrix}=\begin{pmatrix}x'\\ y'\\ z'\\ 1\end{pmatrix}$$



射影变换

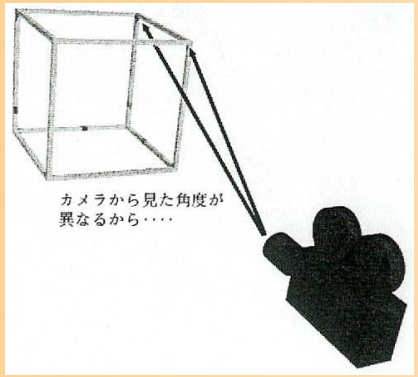
射影変換

- カメラ座標系からスクリーン座標系に変換

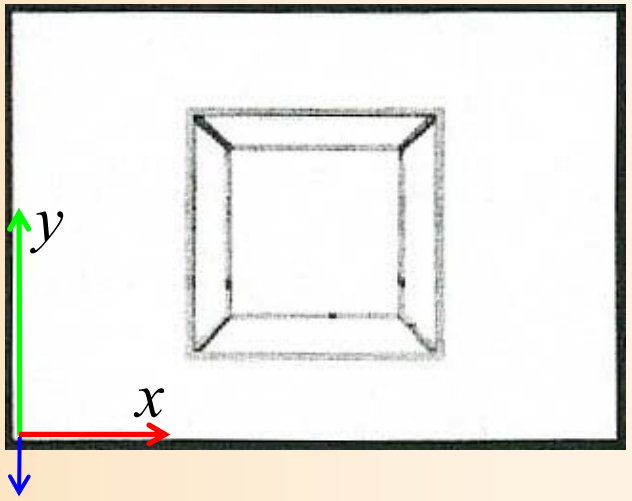


射影変換

- カメラ座標からスクリーン座標への射影変換
- 透視射影変換
  - 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

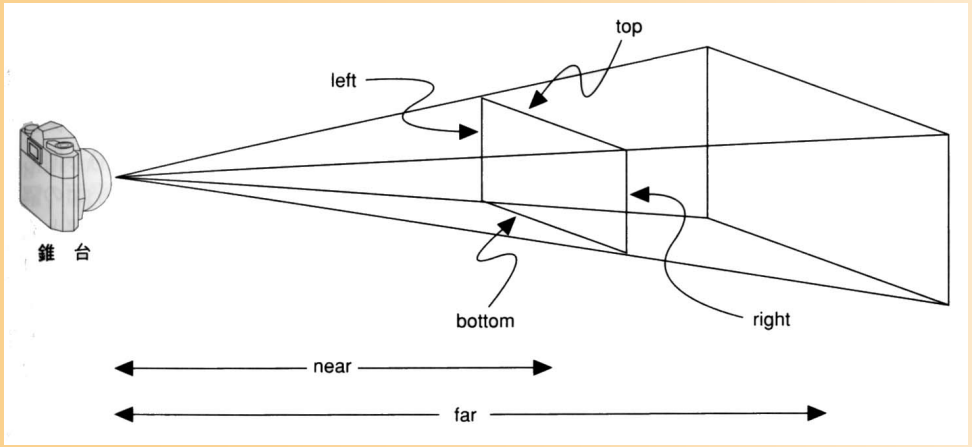


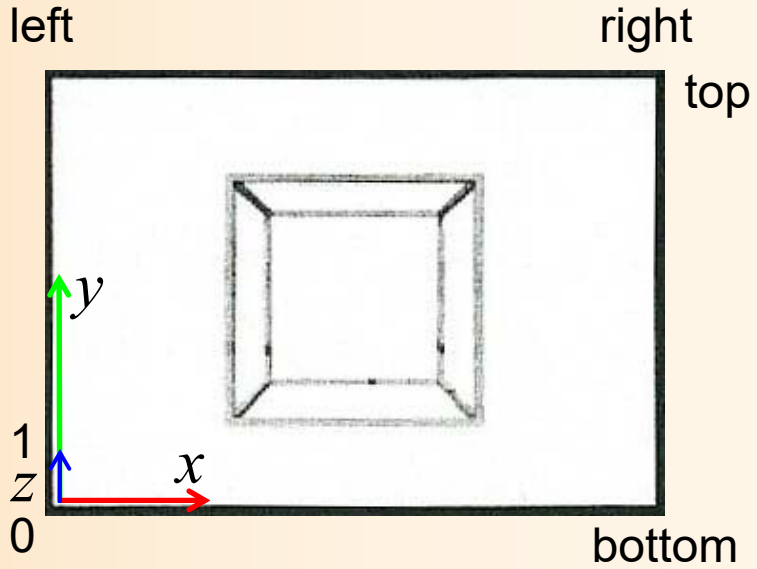
教科書 基礎知識 図2-28



透視射影変換

- 透視射影変換行列



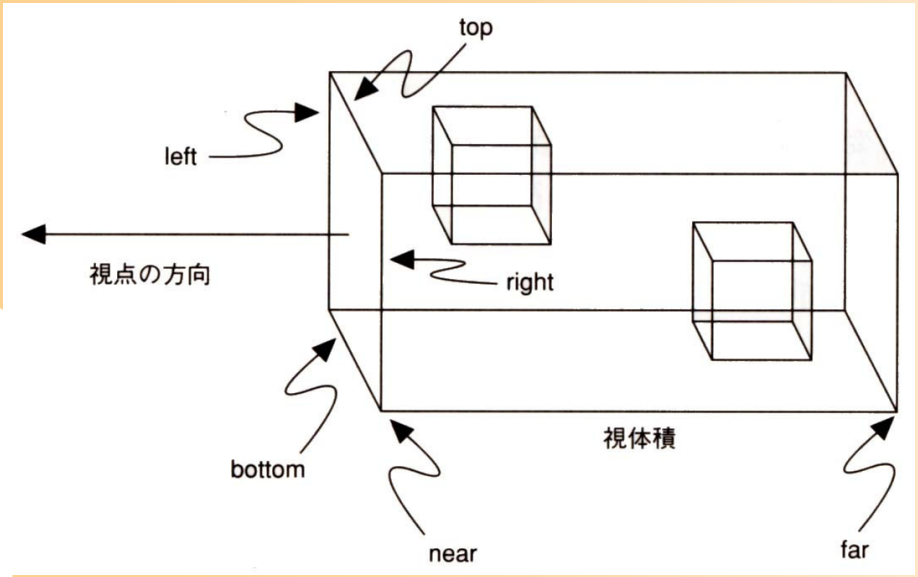


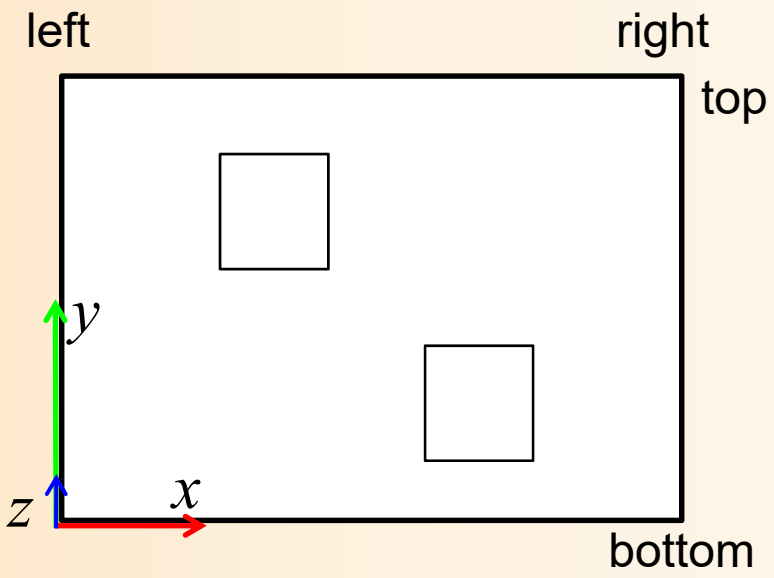
$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} \quad \begin{pmatrix} x'/w' \\ y'/w' \\ z'/w' \end{pmatrix}$$

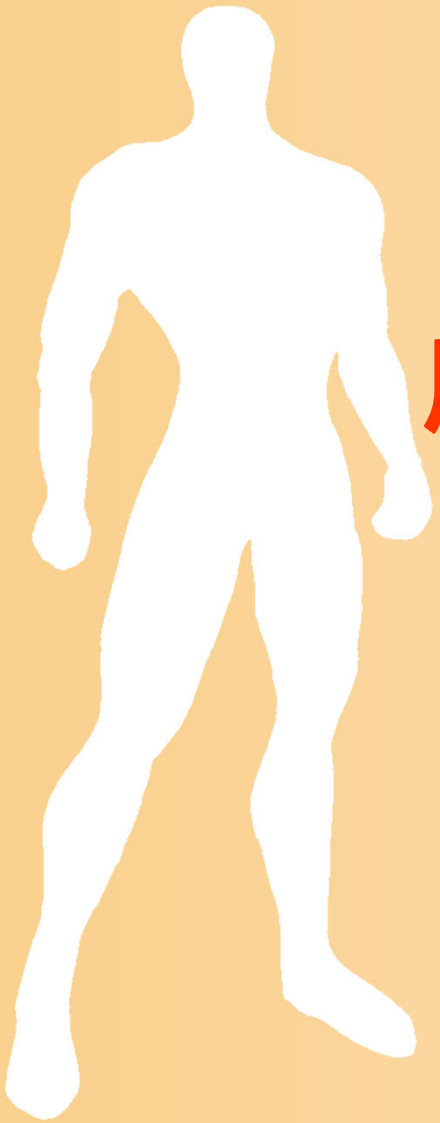
W'=－Z となり、Zで割ることになる
（Z値が大きくなるほど中央になる）

平行射影変換

- 平行射影変換
  - スクリーン対して平行に射影
  - 3面図などを描画するときに用いられる



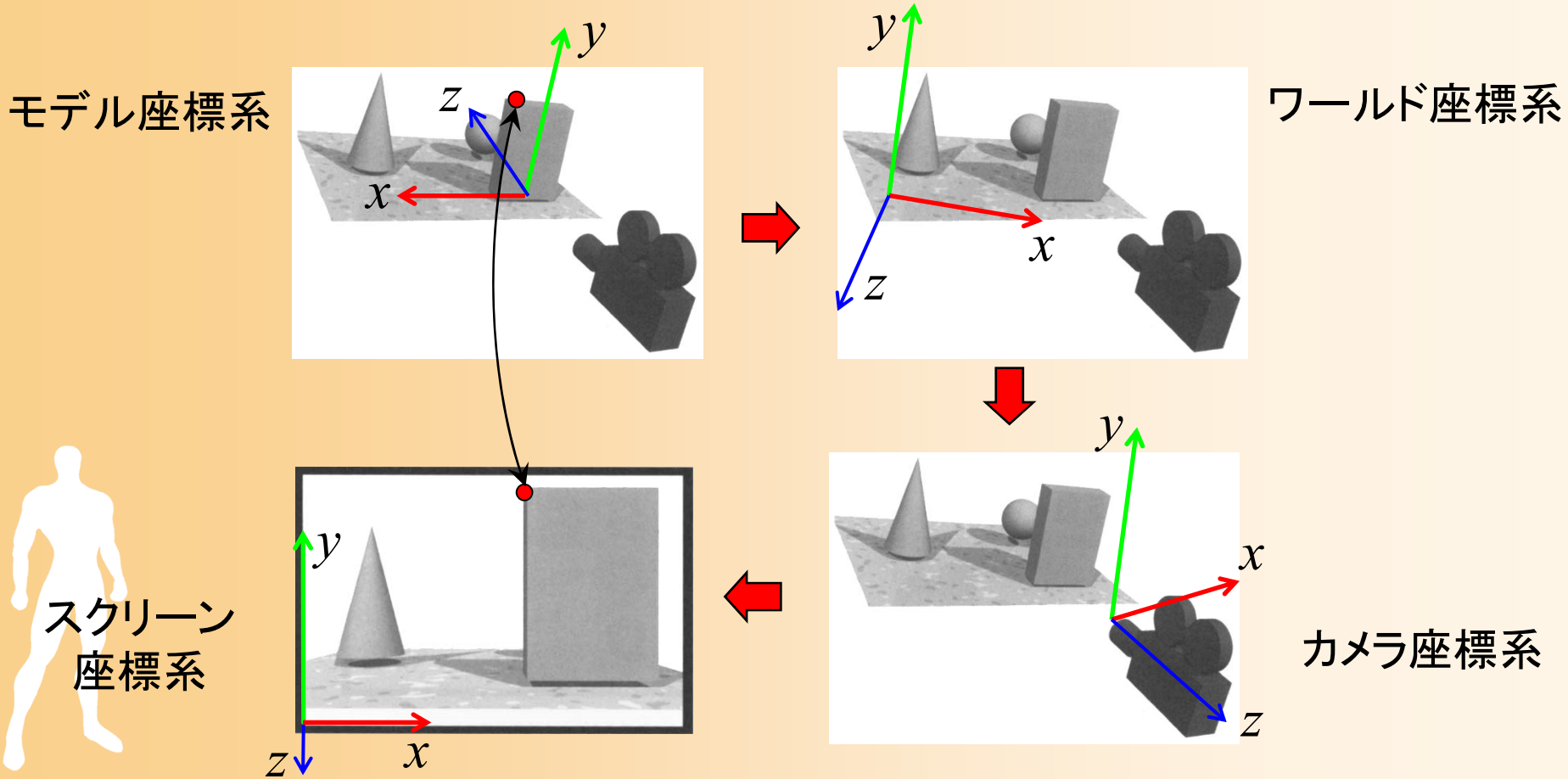




座標変換のまとめ

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

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

モデル座標系での頂点座標

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

射影変換（カメラ→スクリーン）

視野変換（モデル→カメラ）

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

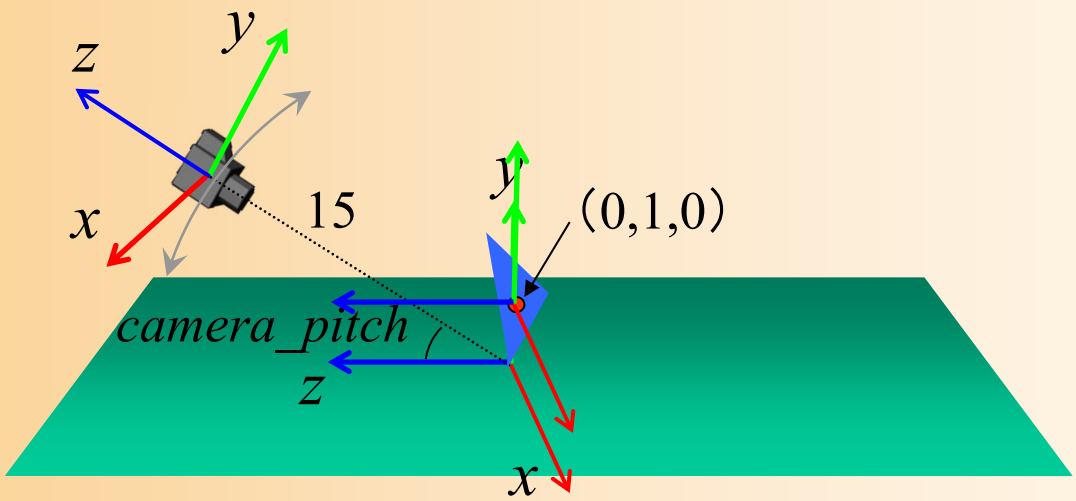
射影変換の設定（サンプルプログラム）

- 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

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -15 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-camera\_pitch) & -\sin(-camera\_pitch) & 0 \\ 0 & \sin(-camera\_pitch) & \cos(-camera\_pitch) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

カメラから見た頂点座標（描画に使う頂点座標）

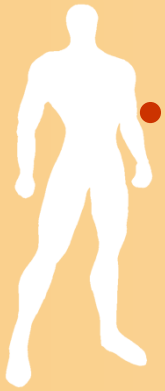
変換行列の設定（サンプルプログラム）

- 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
・・・・・・
```

まとめ

- 前回の復習
- 座標変換の概要
- 座標系
- 視野変換
- 射影変換
- 座標変換のまとめ



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