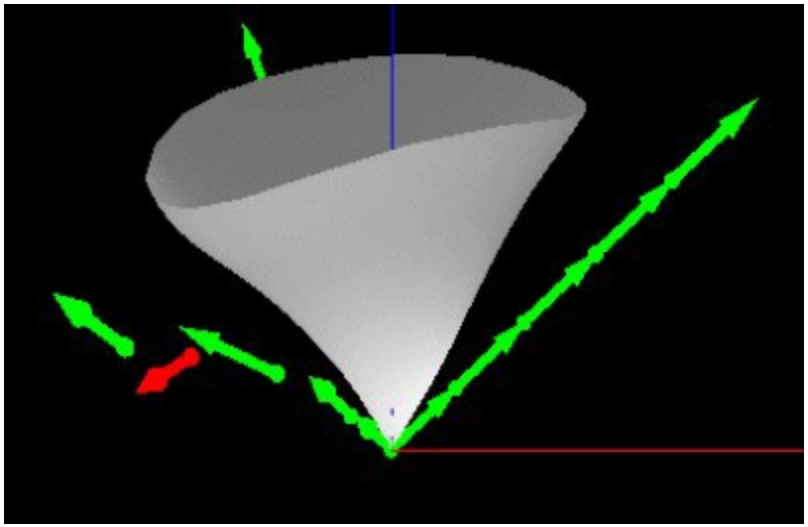
情報工学

2023年度後期　第3回 [11月1日]



静岡大学

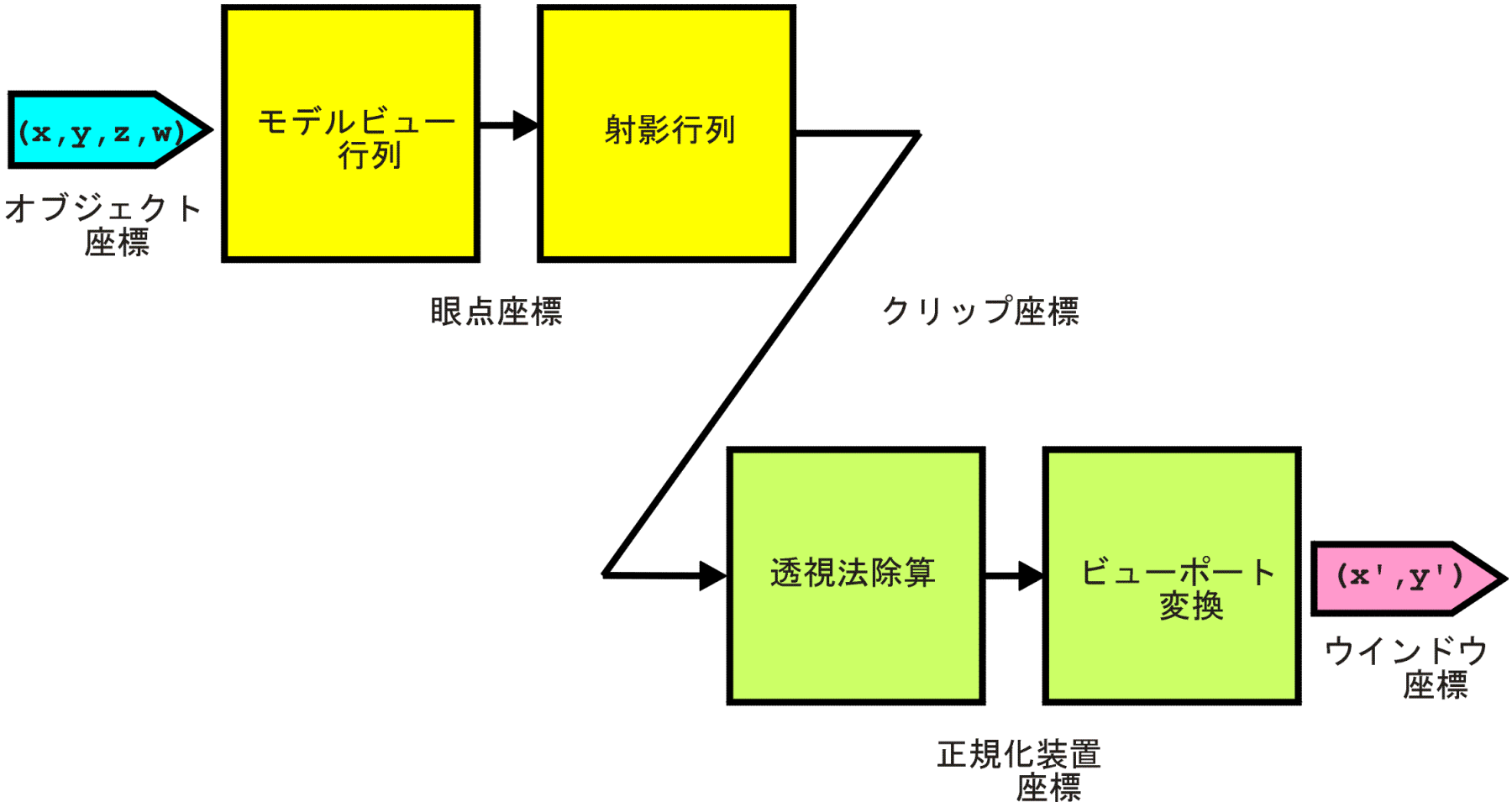
工学研究科機械工学専攻
ロボット・計測情報講座
創造科学技術大学院
情報科学専攻

三浦 憲二郎

講義アウトライン [11月1日]

- 平行移動, 回転移動, 拡大･縮小
- 同次座標系 (homogeneous coordinate system)
- ローカル座標系

頂点変換の手順


(x,y,z,w)
オブジェクト
座標
モデルビュー
行列
射影行列
眼点座標
クリップ座標
透視法除算
ビューポート
変換
(x',y')
ウインドウ
座標
正規化装置
座標

3D図形の移動と拡大･縮小

1. 平行に移動させる: 平行移動 (translation)
2. 回転させる: 回転移動 (rotation)
3. 大きさを変える: 拡大･縮小 (scaling)

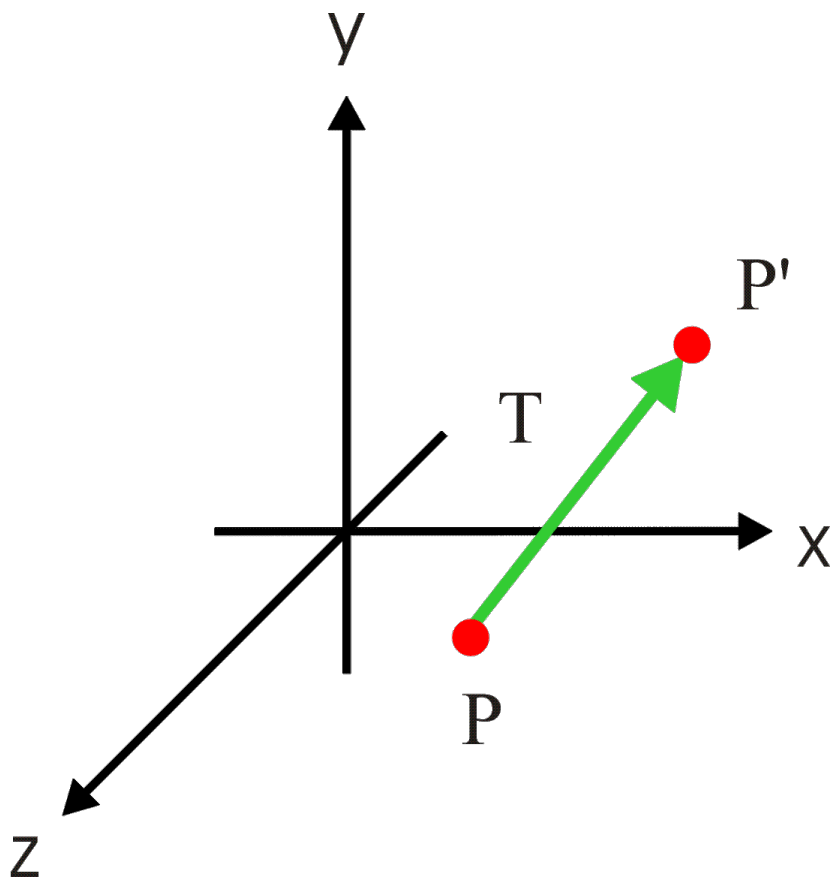
平行移動 (translation)

P(x, y, z) を $T(x_t, y_t, z_t)$だけ平行移動させる.

$$x' = x + x_t$$

$$y' = y + y_t$$

$$z' = z + z_t$$



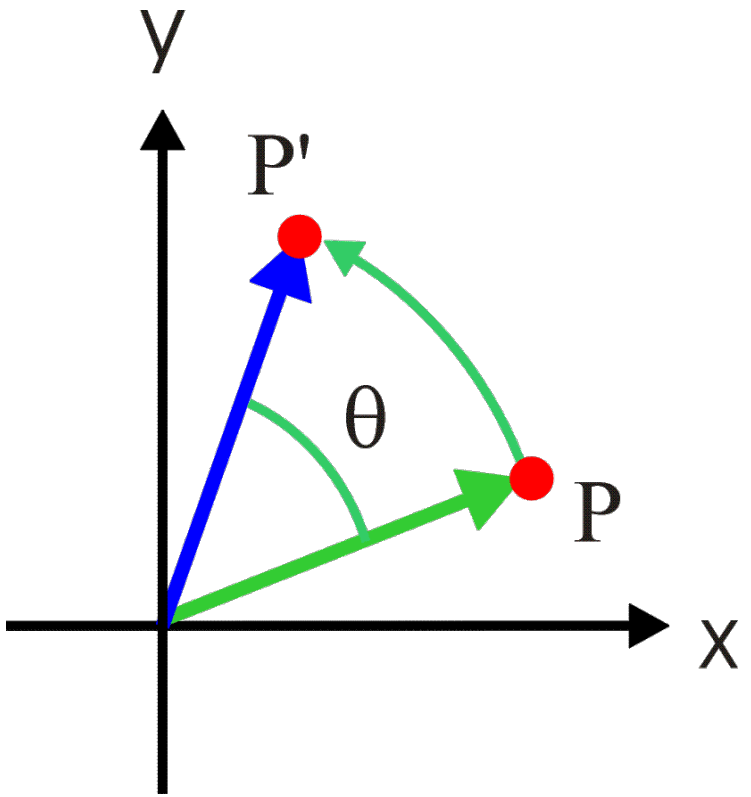
回転移動 (rotation)

例えば, P(x, y, z) を z軸回りにθ度回転移動させる.

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

$$z' = z$$



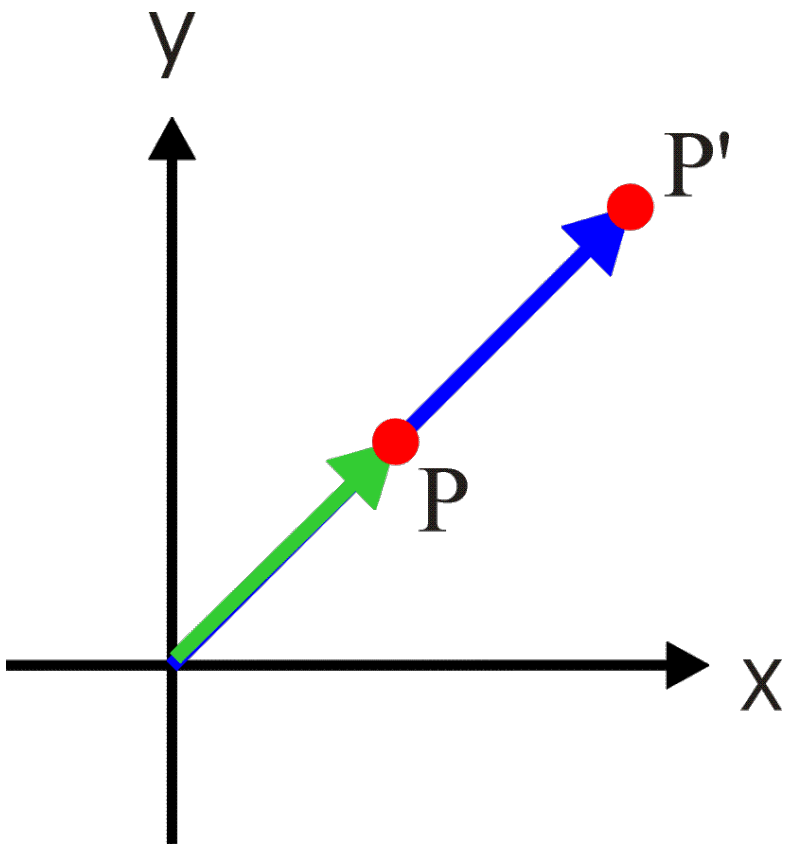
拡大･縮小(scaling)

原点を中心としたスケーリング.

$x' = s_x\, x$

$y' = s_y\, y$

$z' = s_z\, z$



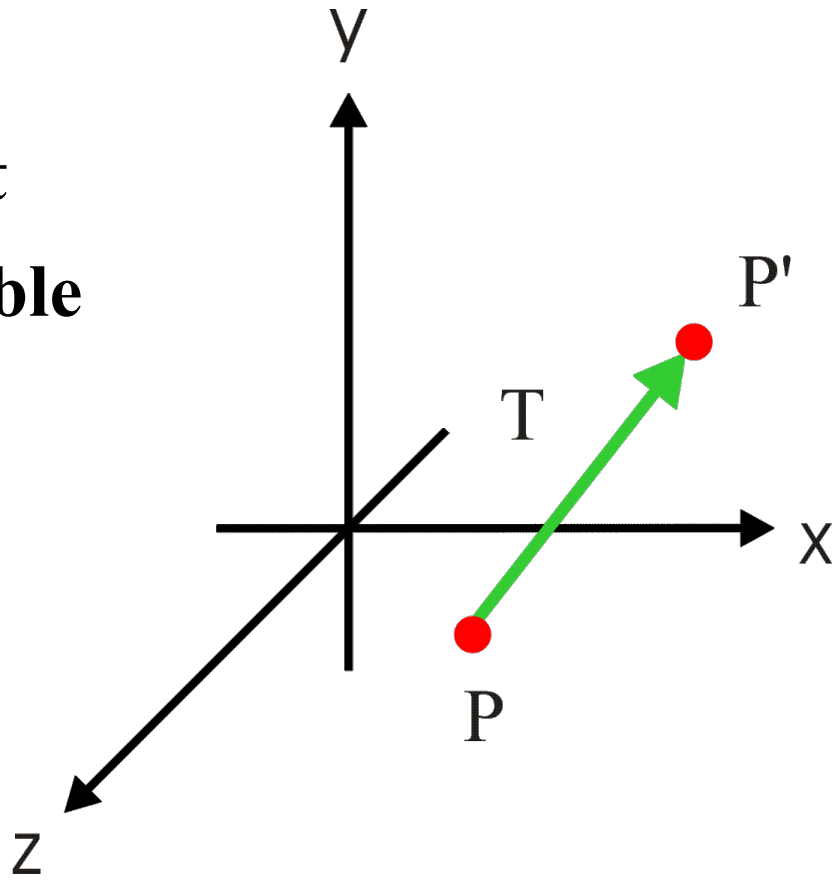
OpenGLのコマンド:平行移動

void glTranslate{fd}(TYPE *x*, TYPE *y*, TYPE *z*)

オブジェクトを(*x*, *y*, *z*)だけ平行移動させる.

TYPE f … GLfloat

d … GLdouble



OpenGLのコマンド:回転移動

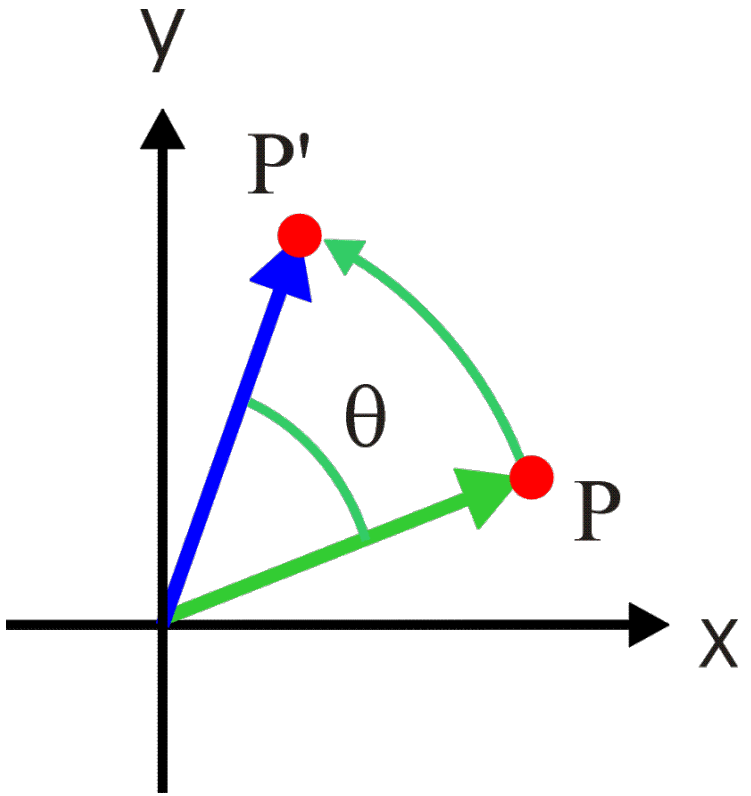
void glRotate{fd}(TYPE *angle*, TYPE *x*, TYPE *y*, TYPE *z*)

オブジェクトを軸(*x*, *y*, *z*)回りに *angle* 度回転移動させる.

TYPE　f … GLfloat

d … GLdouble

右図では(x,y,z)=(0,0,1)



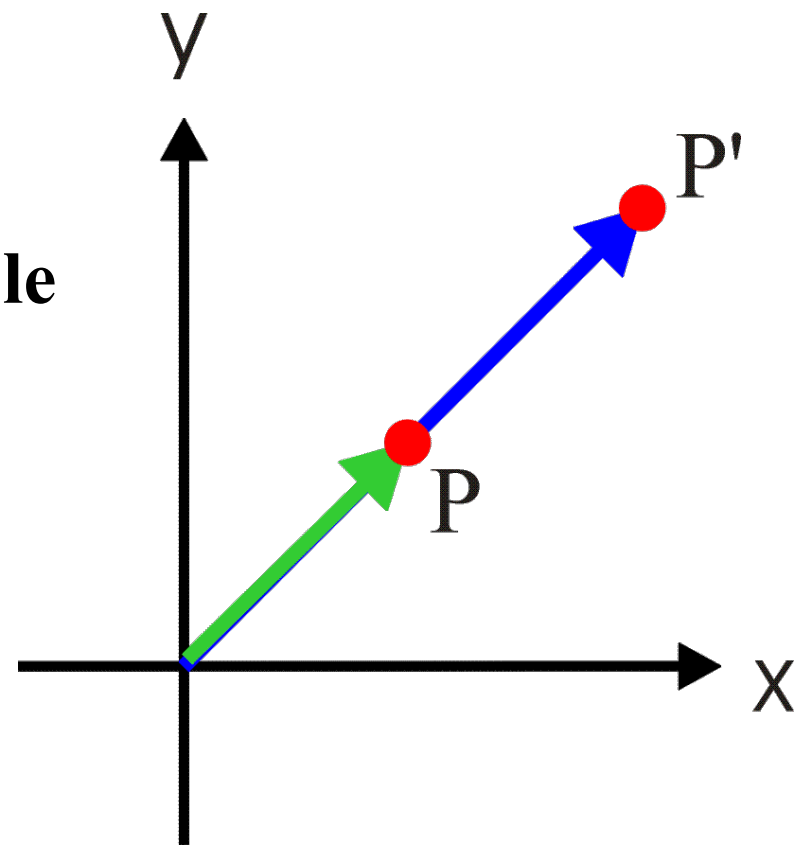
OpenGLのコマンド:拡大･縮小

void glScale{fd}(TYPE *x*, TYPE *y*, TYPE *z*)

オブジェクトを軸に沿って拡大･縮小する.

TYPE f … GLfloat

d … GLdouble



平行移動 (translation):同次座標

同次座標系 (homogeneous coordinate)

3次元 (x, y, z) -> 4次元 (x_h, y_h, z_h, w) として表す.

$$(x, y, z) = (\frac{x_h}{w}, \frac{y_h}{w}, \frac{z_h}{w})$$

平行移動を表す行列T

$$T = \begin{bmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

平行移動 (translation)：行列表現

$$P' = TP = \begin{bmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_t \\ y + y_t \\ z + z_t \\ 1 \end{bmatrix}$$

$$P' = (x + x_t, y + y_t, z + z_t, 1)$$

回転移動 (rotation):同次座標

Z軸回りの回転移動を表す行列R

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

回転移動 (rotation):行列表現

$$P' = RP = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \\ z \\ 1 \end{bmatrix}$$

$$P' = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta, z, 1)$$

拡大･縮小 (scaling):同次座標

拡大･縮小を表す行列S

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

拡大･縮小 (scaling):行列表現

$$P' = SP = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ s_z z \\ 1 \end{bmatrix}$$

$$P' = (s_x x, s_y y, s_z z, 1)$$

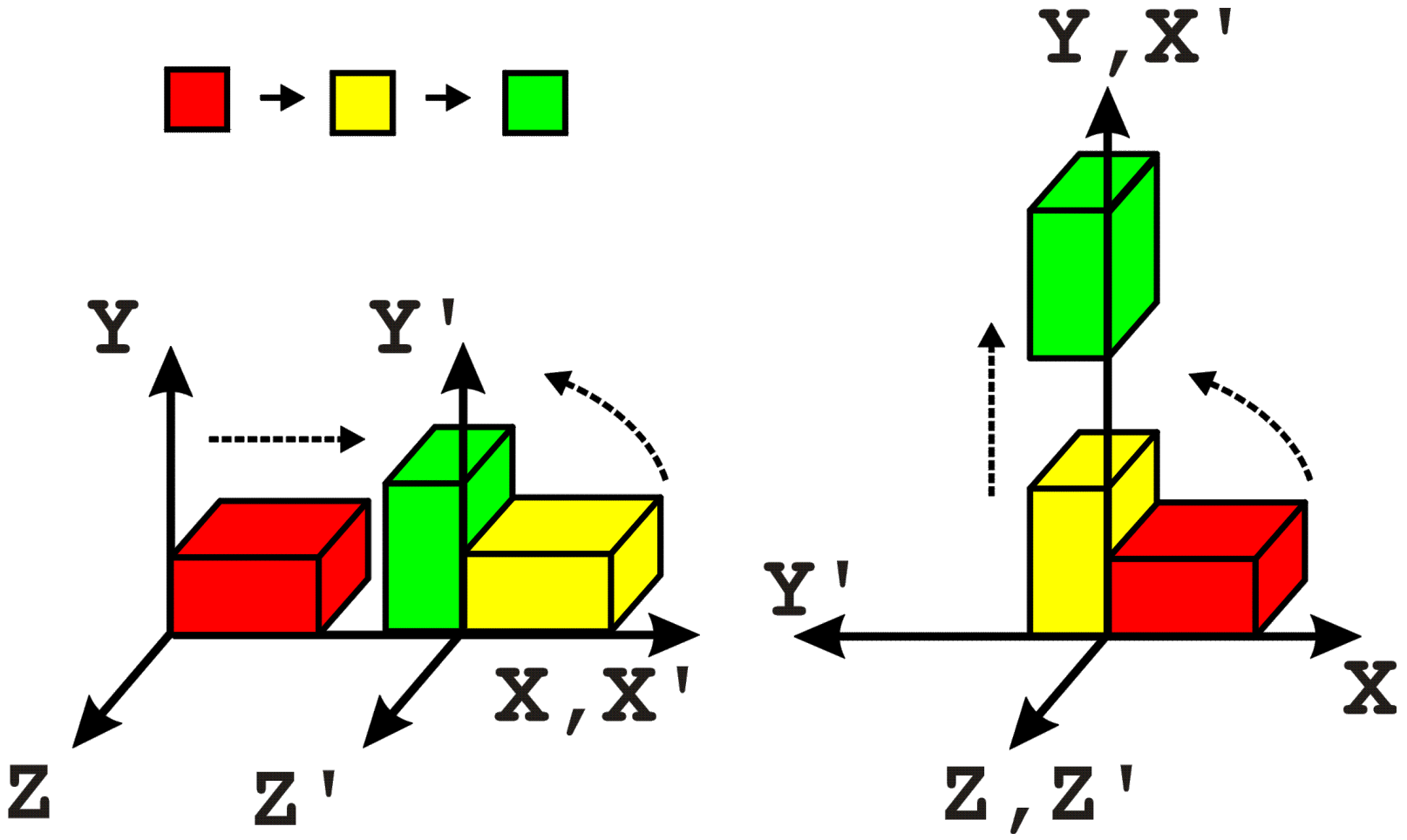
同次座標系のメリット

平行移動, 回転移動, 拡大･縮小の複数の幾何変換が4行4列の１つの行列で表現できる.

$$M_{total} = M_0 M_1 \cdot \cdot \cdot M_n$$

ローカル座標系

オブジェクトとともに座標系にも変換を施す．



ローカル座標系による行列の定義

各行列M_iはローカル座標系で定義される.

$$P' = M_{total} P = M_0 M_1 \cdot \cdot \cdot M_n P$$

課題3 回転，平行移動，拡大縮小

グラフィック

·OpenGLの利用

·線の描画(GL_LINES)

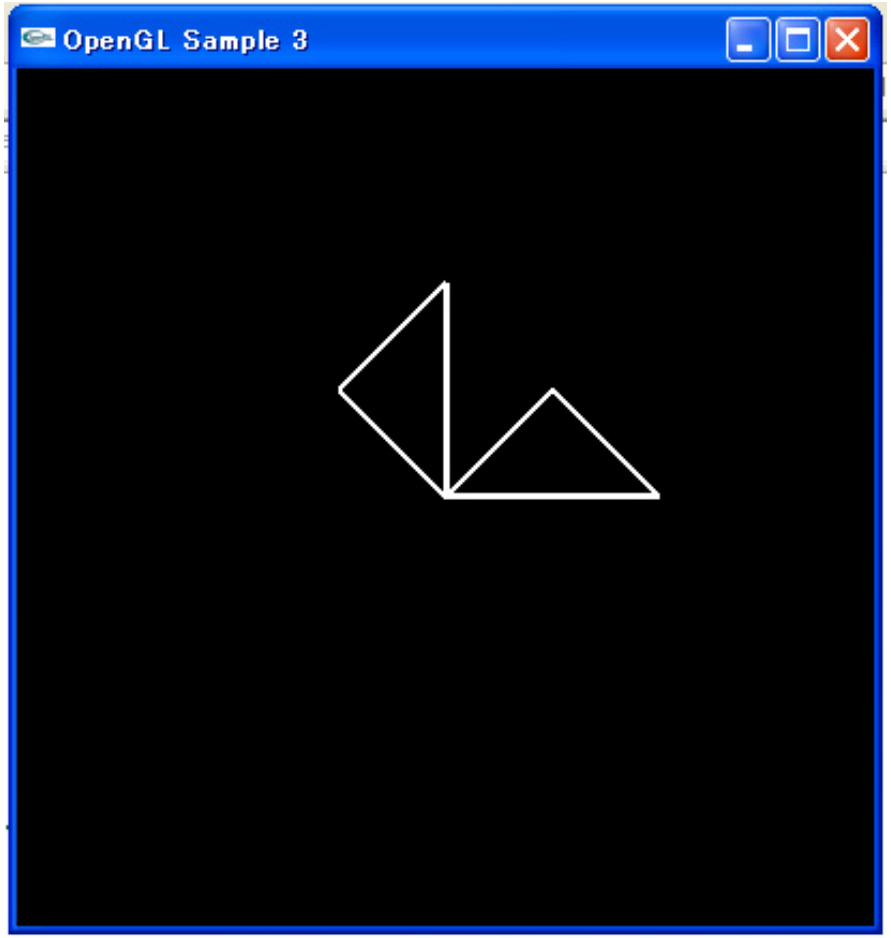
·ウィンドウ座標

·色の指定(glClearColor(), glColor())

·太さの指定(glLineWidth())

·四則演算
·和，差，積，商，余りの計算
·浮動小数型変数 (double 型)

課題3サンプルプログラム (実行結果)



exer3.c(main())

```
int main(int argc, char **argv){
  glutInit(&argc, argv);  /* GLUTの初期化  */
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
                          /* 表示モードの指定 */
  glutInitWindowSize(400, 400);
                          /* ウインドウサイズの指定  */
  glutInitWindowPosition(100, 100);
                          /* ウインドウの位置の指定  */
  glutCreateWindow("OpenGL Sample 2");
                          /* ウインドウのオープン    */
  init();                  /* 初期化処理 */
  glutDisplayFunc(display);
                          /* 表示の関数の指定 */
  glutMainLoop();         /* GLUTのメインループ    */
  return 0;
}
```

exer3.c(init())

```
void init(void)
{
  glClearColor ( 0.0, 0.0, 0.0, 0.0 ); /*背景色の指定*/
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
                                  /*描画のための投影法の指定*/
                                  /*正射影投影法          */
}
```

*初期状態ではカメラ位置は (0,0,0),

z軸の負の向きを向く. y軸が上方向.

exer3.c(drawTriangle(),display())

```
void drawTriangle(){                /* 三角形の描画          */
  glBegin(GL_LINE_LOOP);            /* 線分を描画する        */
  glVertex2f(0.0, 0.0);              /* 第1点の指定         */
  glVertex2f(0.5, 0.0);              /* 第2点の指定         */
  glVertex2f(0.25, 0.25);            /* 第3点の指定          */
  glEnd();
}

void display(void){
  glClear(GL_COLOR_BUFFER_BIT);/* 背景のクリア          */
  drawTriangle();
  glRotatef(90.0, 0.0, 0.0, 1.0);
  drawTriangle();
  glFlush();                    /* 画面を再描画する   */
}
```

三角形の描画

課題

課題A
for文を用いて1個の三角形を90度回転させ、4個の三角形を描画する。
　Hint. display()関数の中で、z軸を回転軸とし、回転角度を90度にして glRotatef()を呼び出しては，drawTriangle()を呼ぶことを繰り返す。

課題B
for文を用いて1個の三角形をx軸方向に0.1ずつ平行移動して4個の三角形を描画する.

課題C
for文を用いて1個の三角形をx軸, y軸方向ともに1.2倍ずつ拡大して4個の三角形を描画する．2つ目の三角形は1.2倍，3つ目の三角形は$1.2 \times 1.2 = (1.2)^2$倍となればよい。

課題D（時間に余裕のある人のために）
三角形に対して，回転，平行移動，およびスケーリングを組み合わせて意味のある形状や模様を作成せよ。

まとめ

- 平行移動, 回転移動, 拡大･縮小
- 同次座標系 (homogeneous coordinate system)
- ローカル座標系