基本課題7

復習

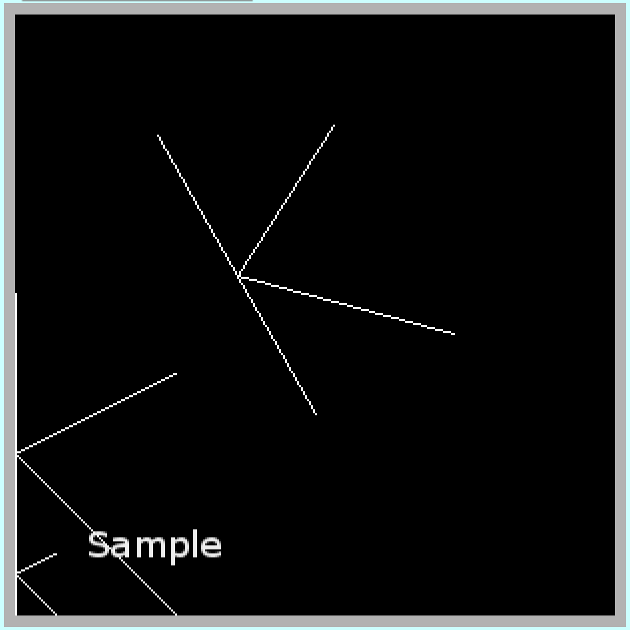
Report7-1

全点を原点の周りに角度 q [度]だけ回転する関数を作成し，以下のmain()関数を用いて実行せよ.

ソースと画像を提出

```
#include  <stdlib.h>
#include  <stdio.h>
#include  <math.h>
#include  "cglec.h"

// struct Point, CglDrawLine()は,"cglec.h"で定義されている
// DrawLines(), Kakudai(), Idou()は, これ以前の行で定義または宣言しておくこと

void Kaiten(Point point[], int n, double q)
{  /* この部分をプログラミング */  }

#define WIDTH 300
#define HEIGHT 300
int main(void)
{
    unsigned char data[WIDTH][HEIGHT];
    Image img = { (unsigned char*) data, WIDTH, HEIGHT };
    CglSetAll(img, 0);
    Point moji_k[] = {{0, 0}, {0, 40}, {20, 0},
                      {0, 20}, {0, 20}, {20, 30}};
    int N = 6;
    DrawLines(img, moji_k, N, 255);
    Kakudai(moji_k, N, 4);
    DrawLines(img, moji_k, N, 255);
    Kaiten(moji_k, N, 30);
    Idou(moji_k, N, 150, 100);
    DrawLines(img, moji_k, N, 255);
    CglSaveGrayBMP(img, "moji_k.bmp");
}
```

実行結果

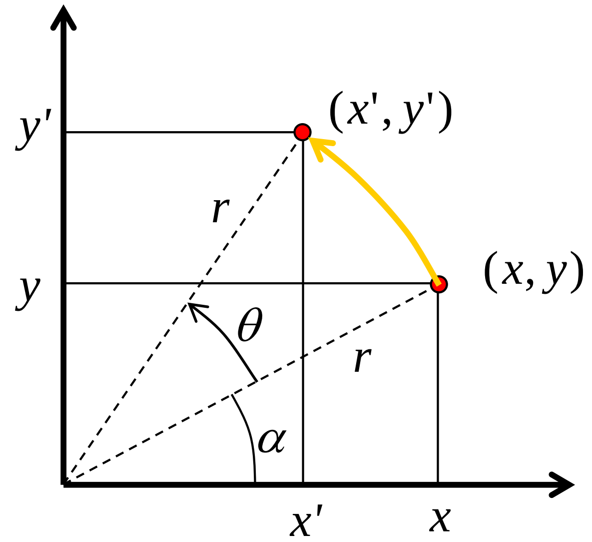


図形の回転

復習

座標位置を角度θ回転する

$$
\begin{aligned}
x' &= r\cos(\alpha+\theta) \\
&= r\cos\alpha\cos\theta - r\sin\alpha\sin\theta \\
&= x\cos\theta - y\sin\theta \\
y' &= r\sin(\alpha+\theta) \\
&= r\sin\alpha\cos\theta + r\cos\alpha\sin\theta \\
&= y\cos\theta + x\sin\theta
\end{aligned}
$$




```
void Kaiten(Point p[], int n, double q)
```

図形を角度 q [度]
回転する関数

基本課題7 解答例

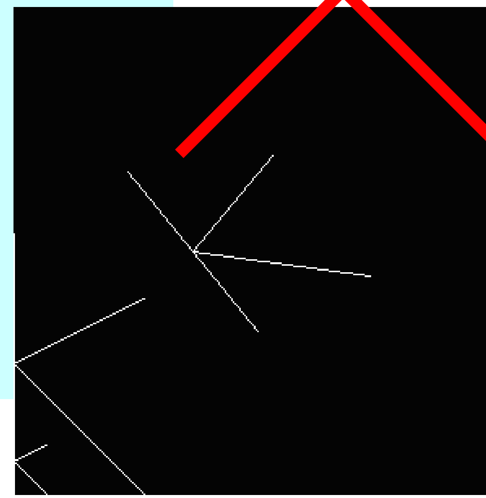
A君解答

```
void Kaiten(Point p[], int n, double q)
{
    int i;
    q = q / 180 * 3.1415;
        for (i = 0; i<n; i++)
        {
            p[i].x = (int)(p[i].x * cos(q) - p[i].y * sin(q));
            p[i].y = (int)(p[i].y * cos(q) + p[i].x * sin(q));

        }
}
```

誤った式

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

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



ライブラリ関数cos()やsin()の単位はラジアン！

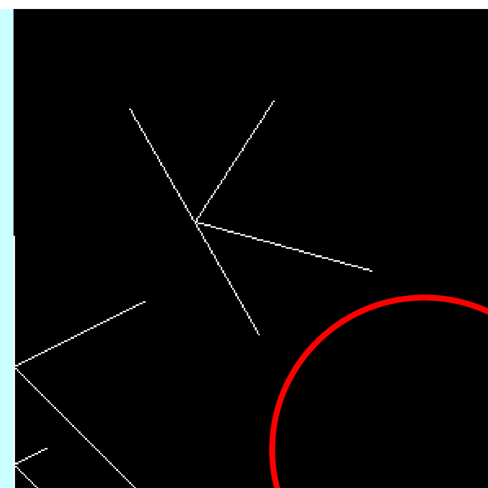
B君解答

```
void Kaiten(Point p[], int n, double q)
{ /* この部分をプログラミング*/
    int i, x, y;
    double si_ta = q/180*3.1415;
    for (i = 0; i < n; i++)
    {
        x = (int)(p[i].x * cos(si_ta) - p[i].y * sin(si_ta) );
        y = (int)(p[i].x * sin(si_ta) + p[i].y * cos(si_ta) );
        p[i].x = x;
        p[i].y = y;
    }

}
```

正しい式

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

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



基本課題7 解答例

C君解答

```
void Kaiten(Point p[], int n, double q)
{
    int i, x, y;
    double qq = q/180 * 3.1415;
    for (i = 0; i < n; i++)
    {
        x = (int)(p[i].x * cos(qq) - p[i].y * sin(qq) + 0.5);
        y = (int)(p[i].x * sin(qq) + p[i].y * cos(qq) + 0.5);
        p[i].x = x;
        p[i].y = y;
    }
}
```

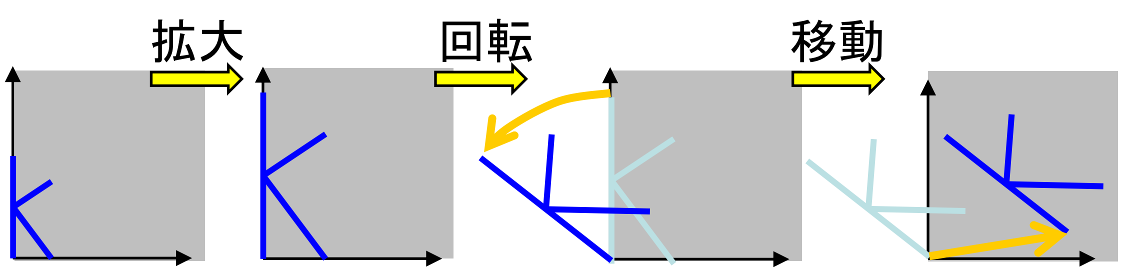
四捨五入のために
0.5を加算

10.499 + 0.5 = 10.999 → 10 ○
10.500 + 0.5 = 11.000 → 11 ○
−10.499 + 0.5 = −9.999 → −9 ×
−10.500 + 0.5 = −10.000→ −10 ×

−10.499 − 0.5 = −10.999 → −10 ○
−10.500 − 0.5 = −11.000 → −11 ○

負の数の四捨五入では0.5を減算しなければならない.

- 本課題の場合，負の数の四捨五入になるときがある
- 本課題の場合，正しく四捨五入しているかどうかはほとんど結果に影響しないが，場合によっては重要になることがある.



基本課題7 解答例

D君解答

```
void Kaiten(Point p[], int n, double q)
{
    int i, x, y;
    double qq = q / 180 * 3.1415;
    double c = cos(qq);
    double s = sin(qq);
    for (i = 0; i < n; i++)
    {
        x = (int)(p[i].x * c - p[i].y * s);
        y = (int)(p[i].x * s + p[i].y * c);
        p[i].x = x;
        p[i].y = y;
    }
}
```

cos()やsin()関数の値はループ内では変化しないのでループの外で計算しておく

良くできました!

アフィン変換の行列表示

復習

$$x' = ax + by + c$$
$$y' = dx + ey + f$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

アフィン変換の係数が一つにまとまる！

同次座標系

$$\boxed{\mathbf{v}' = \mathbf{M}\mathbf{v}}$$

$$\mathbf{v}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \ \mathbf{M} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{v} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

変換前の座標

変換後の座標

拡大/縮小(スケーリング)

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

移動

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

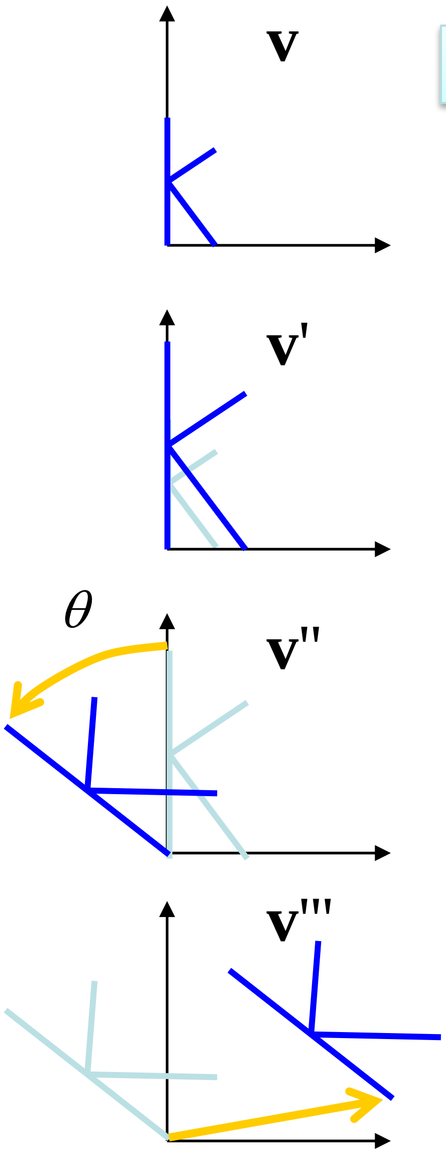
回転(反時計回り)

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

幾何変換の組合せとアフィン変換行列

復習

幾何変換操作



拡大 S

$$\mathbf{v}' = \mathbf{S}\mathbf{v}$$

回転 R

$$\mathbf{v}'' = \mathbf{R}\mathbf{v}'$$

移動 T

$$\mathbf{v}''' = \mathbf{T}\mathbf{v}''$$

$$\Rightarrow \mathbf{v}''' = \mathbf{TRS}\mathbf{v} \Rightarrow \mathbf{v}''' = \mathbf{M}\mathbf{v}$$

アフィン変換行列

$$\mathbf{M} = \mathbf{TRS}$$

移動 T

回転 R

拡大 S

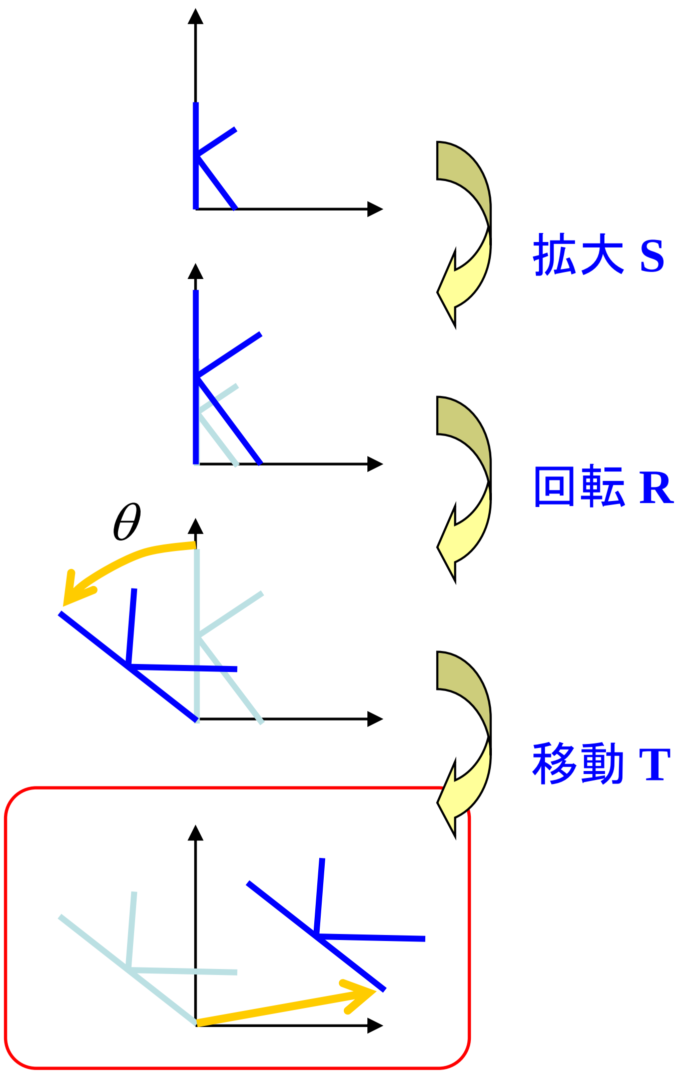
アフィン変換行列の並び順は
幾何変換操作と逆の順番！

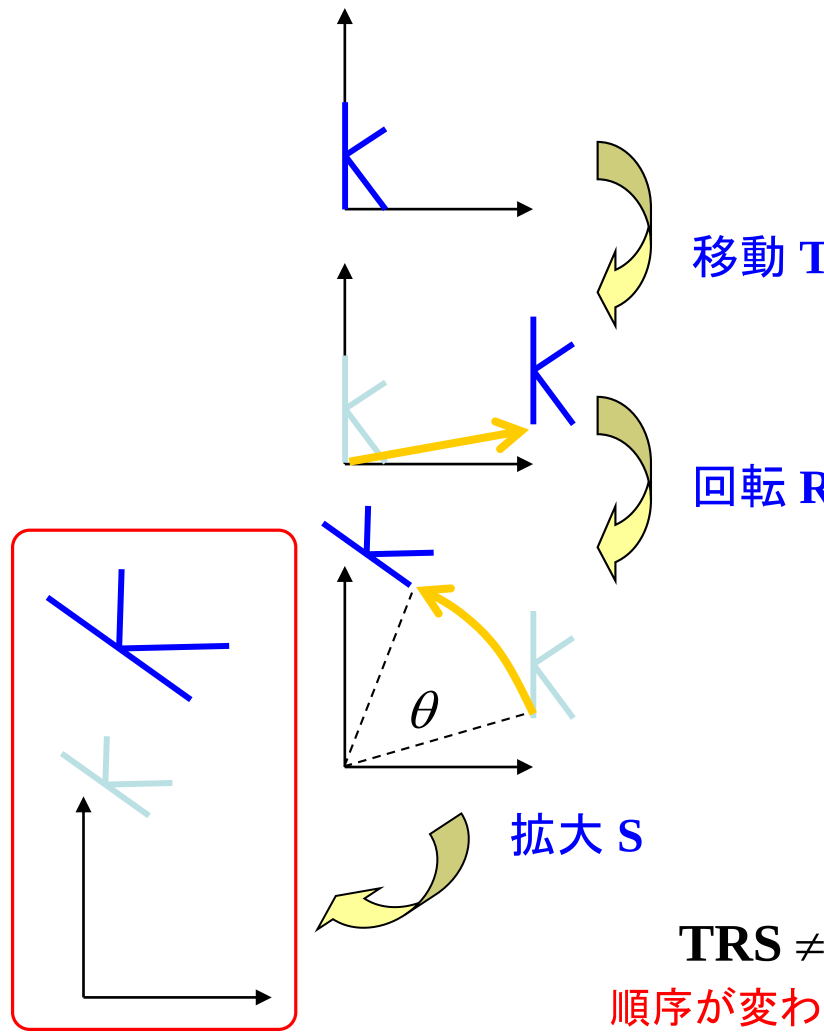
復習

幾何変換の順序

変換行列 $\mathbf{M} = \mathbf{TRS}$

変換行列 $\mathbf{M} = \mathbf{SRT}$

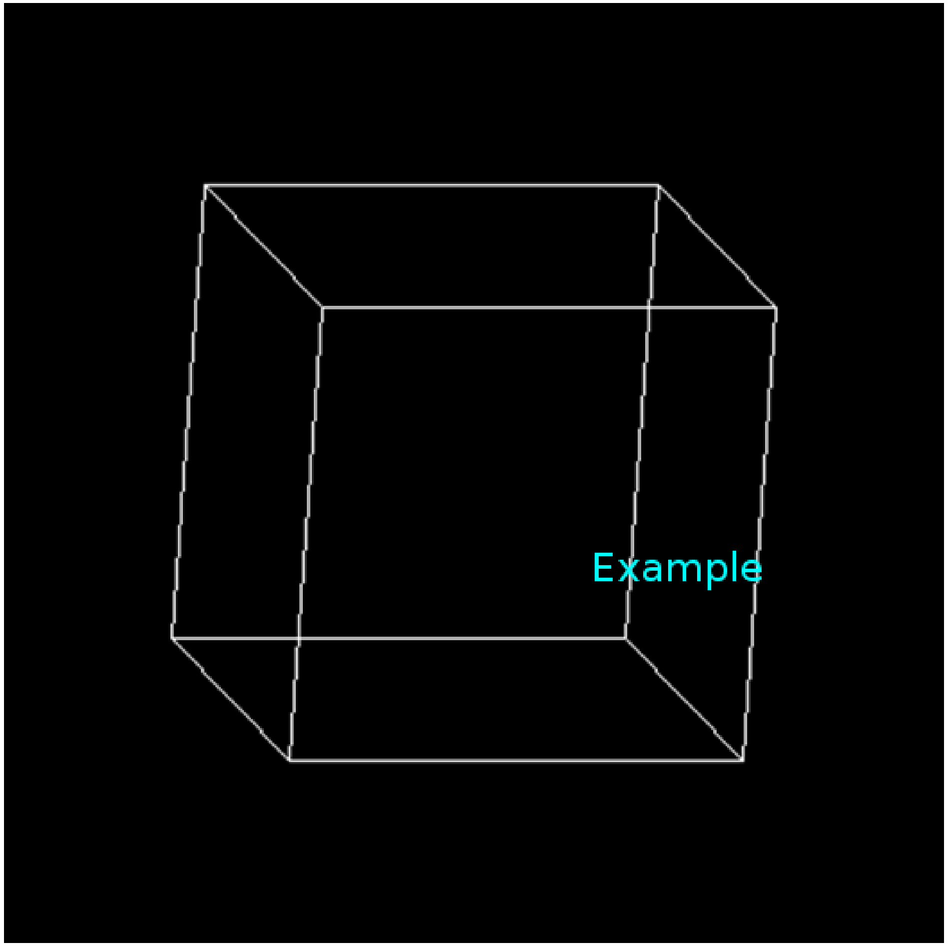




$\mathbf{TRS} \neq \mathbf{SRT}$

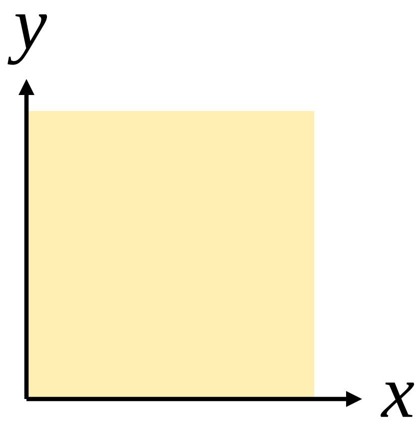
順序が変わると異なった変換になることがある

今日の目標



3次元座標系

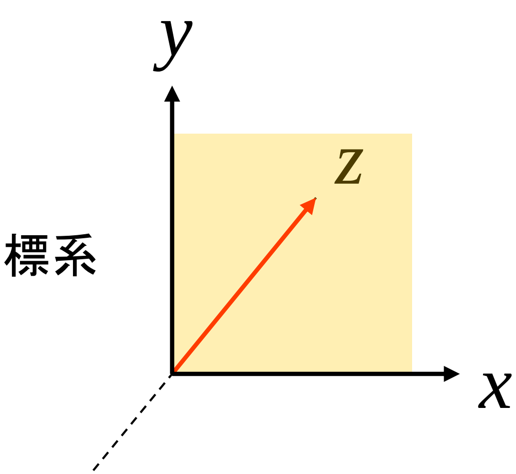
2次元座標系



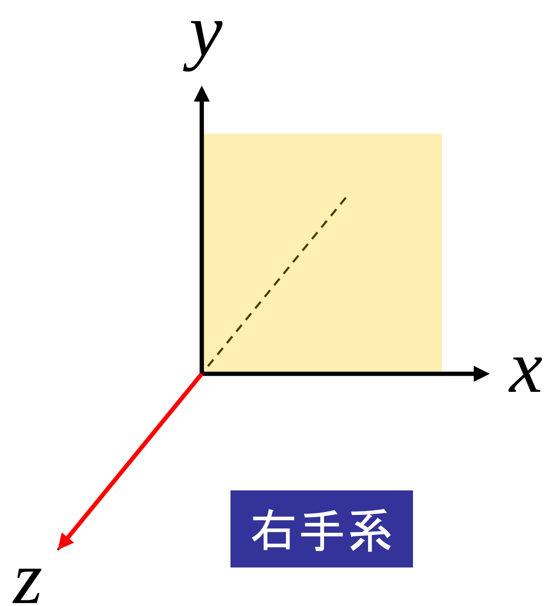
x軸：親指
y軸：人差し指
z軸：中指

右手でこの方向が指差せたら右手系！
左手でこの方向が指差せたら左手系！

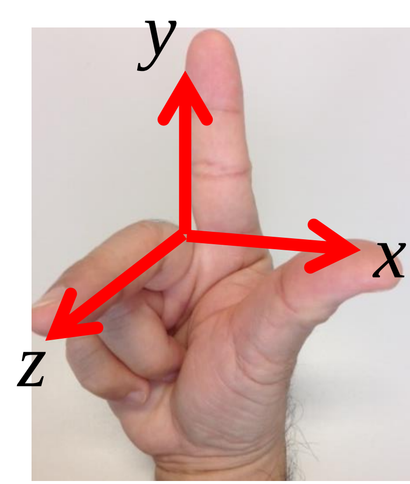
3次元座標系



左手系



右手系



3次元アフィン変換

$$x' = ax + by + cz + s$$
$$y' = dx + ey + fz + t$$
$$z' = gx + hy + iz + u$$

$\Leftarrow$

$$x' = ax + by + c$$
$$y' = dx + ey + f$$

$\Downarrow$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} s \\ t \\ u \end{bmatrix}$$

$\Rightarrow$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & s \\ d & e & f & t \\ g & h & i & u \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

同次座標系

$$\boxed{\mathbf{v}' = \mathbf{M}\mathbf{v}}$$

$$\mathbf{v}' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} a & b & c & s \\ d & e & f & t \\ g & h & i & u \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

変換前の座標

変換後の座標

3次元幾何変換

拡大/縮小(スケーリング)

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x' = S_x x = S_x x + 0 \times y + 0 \times z + 0$$
$$y' = S_y y = 0 \times x + S_y y + 0 \times z + 0$$
$$z' = S_z z = 0 \times x + 0 \times y + S_z z + 0$$

移動

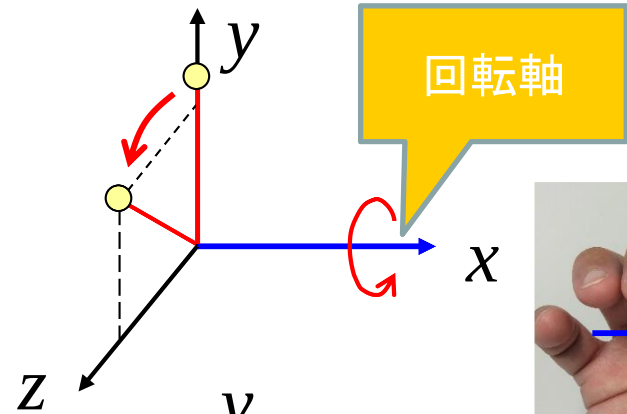
$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x' = x + t_x = 1 \times x + 0 \times y + 0 \times z + t_x$$
$$y' = y + t_y = 0 \times x + 1 \times y + 0 \times z + t_y$$
$$z' = z + t_z = 0 \times x + 0 \times y + 1 \times z + t_z$$

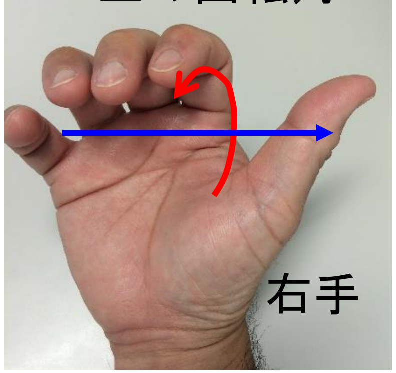
3次元の回転(典型的な例)

x軸周りの回転(反時計回り)

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

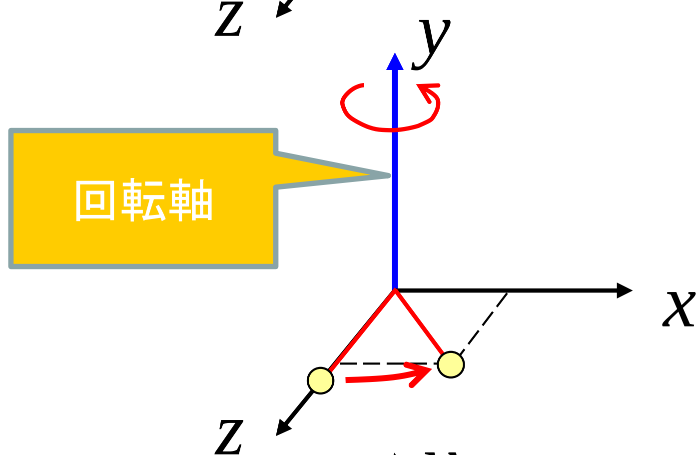


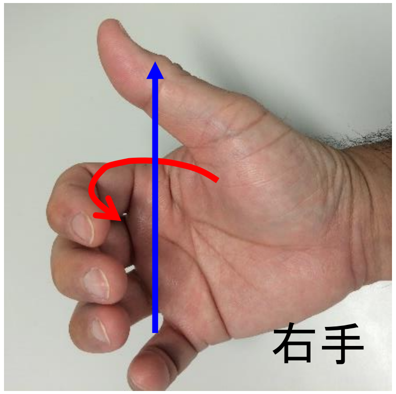
正の回転角



y軸周りの回転(反時計回り)

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

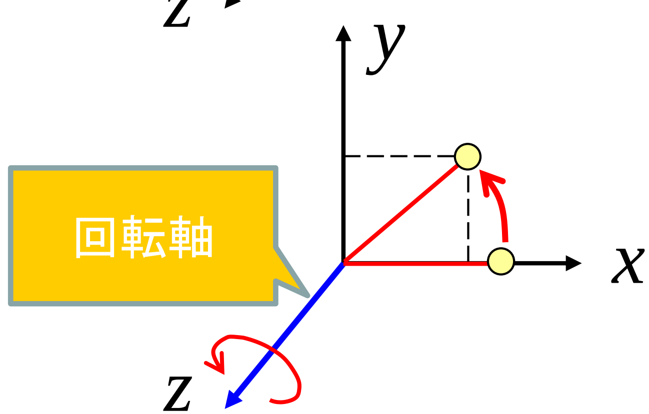




正の回転角

z軸周りの回転(反時計回り)

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



変換行列の組み合わせ

幾何変換操作　:　y 方向に A 倍拡大してから, x 方向に d 移動する.
①　②

① スケーリング

$$\mathbf{S}=\begin{bmatrix}1&0&0&0\\0&A&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

② 移動

$$\mathbf{T}=\begin{bmatrix}1&0&0&d\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$\mathbf{v}'=\mathbf{TSv}$$

アフィン変換行列

$$\mathbf{M}=\mathbf{TS}$$

$$=\begin{bmatrix}1&0&0&d\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&A&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$=\begin{bmatrix}1&0&0&d\\0&A&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

この例では$\mathbf{M} = \mathbf{ST}$としても結果は変わらないが,
回転操作が加わると順序が重要になる

基本課題8

[1] 次の順番どおりに3次元図形を幾何変換した時の総合アフィン変換行列を**M**として，

$$\mathbf{M} = \begin{bmatrix} \cdot & \cdots & & \\ \vdots & \ddots & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

の数式を示しなさい．

レポートでは，Wordの数式機能を用いてきれいに書くこと

途中経過があれば部分点がつくかも

例えば，こんな感じの解答

$$\mathbf{M} = \begin{bmatrix} M\cos\theta_z & \cdots & \cdots & \\ 0 & M\sin\theta_z\cos\theta_x & & \\ -M\cos\theta_y & M & \ddots & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

変換操作の順序と数式は逆順になることに注意

<u>幾何変換操作の順序</u>

(1) x軸の周りに角度 θ_x回転する
(2) y軸の周りに角度 θ_y回転する
(3) x, y, z全ての座標値をM倍する
(4) x方向にt_x, y方向にt_y, z方向にt_zだけ移動する

[2] 次の値を代入したときのアフィン変換行列の値を<u>電卓等を用いて計算し</u>，変換行列**M**を数値で示しなさい．

$\theta_x = \theta_y = 15$ [度]
$M = 100$
$t_x = t_y = t_z = 200$

例えば，こんな感じの解答

$$\mathbf{M} = \begin{bmatrix} 90.0 & 5.5 & \cdots & \\ 0 & -3.1 & & \\ -20.3 & 25.2 & \ddots & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

基本課題8[3]が3ページ後にあり

投影

3次元空間で与えられた図形を2次元平面の図形に変換すること
(3次元図形をコンピュータで表示するためにはこれが必要)

主な投影法

i. 正投影
ii. 斜投影
iii. 透視投影

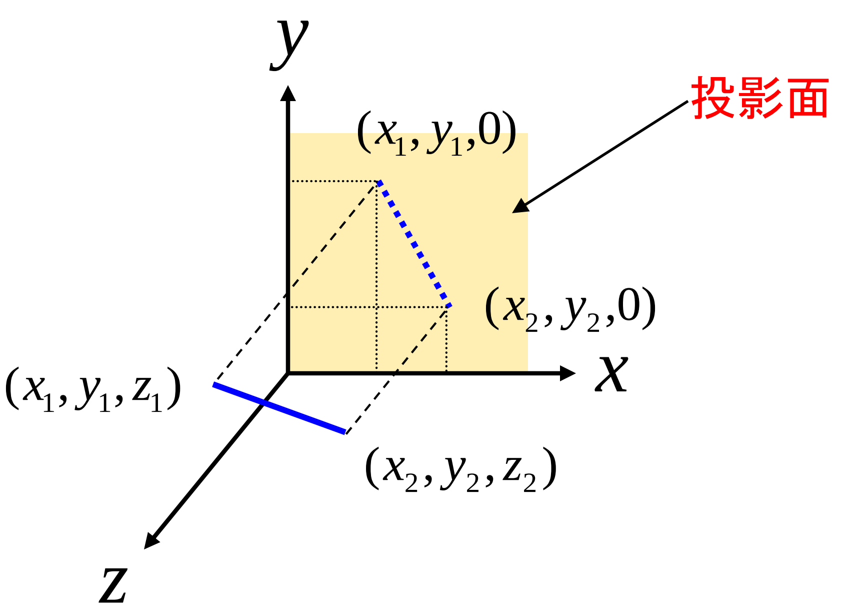
・・・他にもたくさんある

xy平面への正投影
(z座標値の除去)

もしも正投影を変換行列として書いたら. .

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

実際にはz値を無視するだけなので変換行列を用いる必要は無い.



アフィン変換を行う関数

Example8-1

$$\mathbf{M} = \begin{bmatrix} m[0][0] & m[0][1] & m[0][2] & m[0][3] \\ m[1][0] & m[1][1] & m[1][2] & m[1][3] \\ m[2][0] & m[2][1] & m[2][2] & m[2][3] \\ m[3][0] & m[3][1] & m[3][2] & m[3][3] \end{bmatrix}$$

```
struct Point3D  //3次元の座標点
{
    double x;
    double y;
    double z;
};

void AffineTransform(Point3D p[], int n, double m[][4]) //n個の座標点p[]をアフィン行列mで変換
{
    int i;
    for (i = 0; i < n; i++)
    {
        double x = p[i].x, y = p[i].y, z = p[i].z;
        p[i].x = m[0][0]*x + m[0][1]*y + m[0][2]*z + m[0][3];
        p[i].y = m[1][0]*x + m[1][1]*y + m[1][2]*z + m[1][3];
        p[i].z = m[2][0]*x + m[2][1]*y + m[2][2]*z + m[2][3];
    }
}

int main(void)      // 以下はAffineTransform()の使い方の例. 意味はない.
{
    Point3D p3[] = { {10, 20, 30}, {15, 30, 20}, {20, 20, 20} };  //3次元の三つの点(図形データ)
    double mat[][4] = { {1.0,  0.0, 0.0, 10.0},// y方向に20倍拡大し, x方向に10移動する変換行列
                        {0.0, 20.0, 0.0,  0.0},
                        {0.0,  0.0, 1.0,  0.0},
                        {0.0,  0.0, 0.0,  1.0}  }
    AffineTransform(p3, 3, mat);         // p3の各点を変換行列matで変換
}
```

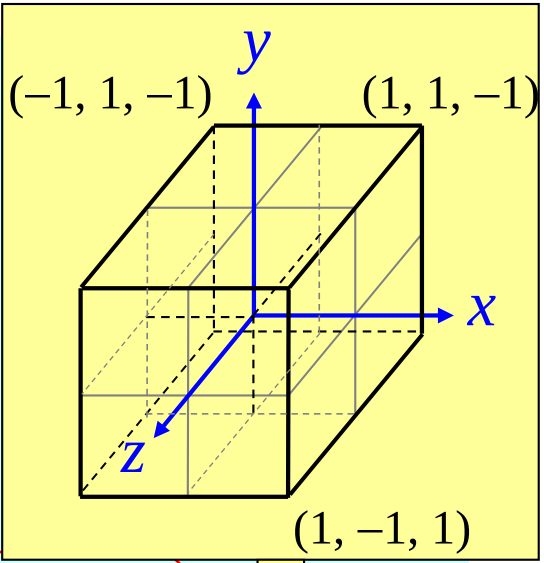
注) C言語では2次元配列を引数にする場合は二つ目のインデックスの要素数を明示しなければならない

基本課題8[3]

ソースと実行例を提出

Report8-1

以下は辺の長さが2である立方体の図形を，アフィン変換・正投影して描くプログラムである．このプログラムを完成せよ．




```
#include  <stdlib.h>
#include  <stdio.h>
#include  <math.h>
#include  "cglec.h"
// struct Point, CglDrawLine(), CglDrawLines()は"cglec.h"で定義済み
// struct Point3D, AffineTransform()はこれ以前で定義・宣言しておく
#define WIDTH 400
#define HEIGHT 400
int main(void)
{
    unsigned char data[WIDTH][HEIGHT];
    Image img = { (unsigned char*) data, WIDTH, HEIGHT };

    Point3D cube[] = {                                          //立方体の線図形データ
    {1, -1, 1}, {1, -1, -1}, {1, -1, -1}, {-1, -1, -1},  //底面
    {-1, -1, -1}, {-1, -1, 1}, {-1, -1, 1},{1, -1, 1},
    {1, 1, 1}, {1, 1, -1}, {1, 1, -1}, {-1, 1, -1},      //上面
    {-1, 1, -1}, {-1, 1, 1}, {-1, 1, 1},{1, 1, 1},
    {1, -1, 1}, {1, 1, 1}, {1, -1, -1}, {1, 1, -1},      //縦線
    {-1, -1, -1}, {-1, 1, -1}, {-1, -1, 1}, {-1, 1, 1} };
    int N = 24;    // 点データの数は24個

    // ============= ここで基本課題8[2]で求めたアフィン変換行列を定義する
    double mat[][4] = {   // 行列の数値を入れる
                          //  ・・・
                    };    //
```

モデル

各自でプログラムする内容

i) 基本課題8[2]で計算した数値をここに書き込む

ソースは次のページに続く

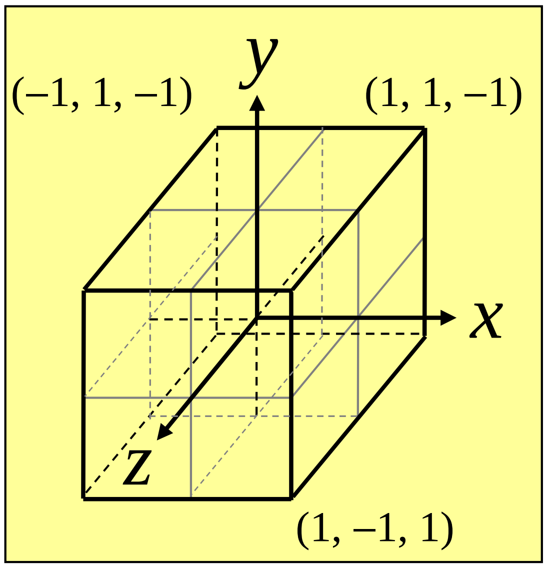
基本課題8[3]続き

```
    CglSetAll(img, 0);
    AffineTransform(cube, N, mat)    // アフィン変換を実行

    Point p[24];   // 正投影後の点データを入れる2次元の配列
    // ============== ここからプログラムする
    // アフィン変換したcubeを正投影する. cube[]→p[]
    // CglDrawLines()を用いて画像imgに2次元図形p[]を描く

    CglSaveGrayBMP(img, "cube.bmp");
}
```

各自でプログラムする内容

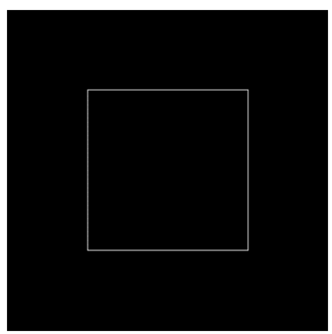
ii) Point3D型配列の3次元点データcubeを正投影(z値を無視)してPoint型配列の点データpに代入

iii) 変換した点データpとCglDrawLines()関数を用いて2次元画像を描画

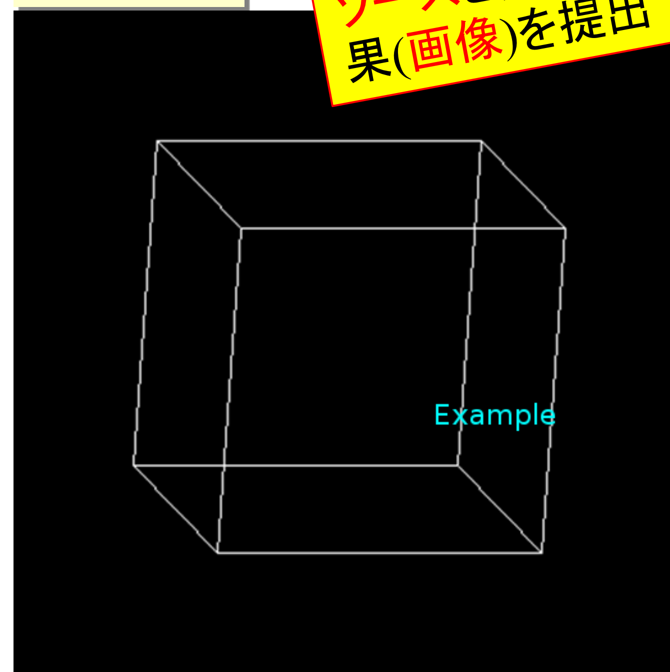


アフィン変換前の立方体データ

仮に回転せずに移動・拡大だけで正投影したら・・・



実行結果



ソースと実行結果(画像)を提出

発展課題8

基本課題8で行った3次元図形のアフィン変換と投影は，[1]〜[2]の行例の手計算がいささか面倒であるので，基本課題と同じ図形を次の方法で描画せよ.

1. 基本課題の図形座標データCube[]とAffineTransform()関数はそのまま利用する.
2. 基本課題の四つの個別のアフィン変換行列を数値で求める.
   - $\mathbf{R}_x$　x軸の周りに15度回転
   - $\mathbf{R}_\mathrm{y}$　y軸の周りに15度回転
   - $\mathbf{S}$　100倍に拡大
   - $\mathbf{T}$　x, y, z方向に200ピクセル移動
3. これらの四つのアフィン変換行列を用いてAffineTransform()関数を正しい順番で４回実行し，図形データCube[]の座標を変換する.
4. 基本課題と同様に，変換後の座標から正投影により２次元図形を描いて保存する.