第5章 アフィン変換やスキュー処理，モルフォロジ処理など

拡大/回転/移動/膨張

米田 幸生

拡大/縮小/回転/移動

● 準備…3×3の行列を扱うクラス

アフィン変換とは，画像の拡大/縮小や回転と平行移動を組み合わせて変換を行うことです．元画像を(x, y)，変換後の画像を(x', y')とすると，画像の拡大/縮小は，次式で表すことができます．

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

リスト1　3×3の行列を扱えるようにしたクラス（Matrix.h）

```
#ifndef _MATRIX_H
#define _MATRIX_H

class Matrix
{
public:
    static const int row = 3;
    static const int col = 3;

    //メンバ変数
    union {
        struct
        {
            double a, b, c,
                d, e, f,
                g, h, i;
        };
        double val[row][col];
    };

    //コンストラクタ
    Matrix()
    {
        int i, j;
        for (i = 0; i < row; i++)
        {
            for (j = 0; j < col; j++)
            {
                val[i][j] = 0;
            }
        }
    };

    Matrix(double x1, double x2, double x3,
           double x4, double x5, double x6,
           double x7, double x8, double x9)
    {
        val[0][0] = x1;
        val[0][1] = x2;
        val[0][2] = x3;
        val[1][0] = x4;
        val[1][1] = x5;
        val[1][2] = x6;
        val[2][0] = x7;
        val[2][1] = x8;
        val[2][2] = x9;
    };
: 略

//行列式を取得する
inline double determinant(const Matrix &cur)
{
    //Sarrusの方法
    double det = 0;
    det += cur.val[0][0] * cur.val[1][1] * cur.val[2][2];
    det += cur.val[0][1] * cur.val[1][2] * cur.val[2][0];
    det += cur.val[0][2] * cur.val[1][0] * cur.val[2][1];
    det -= cur.val[0][2] * cur.val[1][1] * cur.val[2][0];
    det -= cur.val[0][0] * cur.val[1][2] * cur.val[2][1];
    det -= cur.val[0][1] * cur.val[1][0] * cur.val[2][2];
    return det;
}

//逆行列を取得する
inline Matrix inverse(const Matrix &cur)
{
    Matrix New;
    double det = determinant(cur);

    if (det != 0)
    {
        //余因子展開,cramerの公式
    New.val[0][0] = (cur.val[1][1] * cur.val[2][2]
    - cur.val[1][2] * cur.val[2][1]) / det;
    New.val[0][1] = -(cur.val[0][1] * cur.val[2][2]
    - cur.val[0][2] * cur.val[2][1]) / det;
    New.val[0][2] = (cur.val[0][1] * cur.val[1][2]
    - cur.val[1][1] * cur.val[0][2]) / det;

    New.val[1][0] = -(cur.val[1][0] * cur.val[2][2]
    - cur.val[1][2] * cur.val[2][0]) / det;
    New.val[1][1] = (cur.val[0][0] * cur.val[2][2]
    - cur.val[0][2] * cur.val[2][0]) / det;
    New.val[1][2] = -(cur.val[0][0] * cur.val[1][2]
    - cur.val[1][0] * cur.val[0][2]) / det;

    New.val[2][0] = (cur.val[1][0] * cur.val[2][1]
    - cur.val[1][1] * cur.val[2][0]) / det;
    New.val[2][1] = -(cur.val[0][0] * cur.val[2][1]
    - cur.val[0][1] * cur.val[2][0]) / det;
    New.val[2][2] = (cur.val[0][0] * cur.val[1][1]
    - cur.val[0][1] * cur.val[1][0]) / det;
    }
    return New;
}

#endif /* _MATRIX_H */
```

行列を扱うためのクラスを作成．3×3の構成

構造体として値を設定することも，2次元配列で設定することもできるように共用体にしている

今回，代入演算子や比較演算子など作ってみたが，実際には画像処理にはほぼ使っていない