Excel による二次元図形処理
（第２報：合成マトリックス）

Two-dimensional Computer Graphics
by spreadsheet Excel
(Part2: matrix synthesis)

光成豊明

Toyoaki Mitsunari

要旨

前報では，二次元図形処理に関して代数式を基にしてプログラム言語を使用しないで，代数式の代わりに３×３の変換マトリックスを適用し，表計算ソフトウェアである Excel により計算処理から変換前と後の図形表示の処理方法を提案した．提案した処理方法は，Excel 内で定義した変換マトリックスの係数を変化させることで，各種の二次元図形処理が元の座標と変換された座標を同時に表示でき，それに伴って図形が表示されることから，二次元図形処理の変換プロセスの可視化が可能となり，図形処理教育への適用が期待できる．本報告では，前報では課題であった任意の点を考慮した図形処理，特に，合成マトリックスを適用した任意の点を中心とした回転について検討したものである．

［キーワード］ＣＧ，表計算ソフトウェア，図形処理教育

１．はじめに

前報[1)]では，平行移動，回転，スケーリング，反転，せん断の５つの二次元図形処理に関して代数式を基にしてプログラム言語を使用しないで，代数式の代わりに３×３の変換マトリックスを適用した，表計算ソフトウェアである Excel により計算処理から変換前と後の図形表示の処理方法を提案した．提案した処理方法は，Excel 内で定義した変換マトリックスの係数を変化させることで，各種の二次元図形処理が元の座標と変換された座標を同時に表示でき，それに伴って図形が表示されことから，二次元図形処理の変換プロセスの可視化が可能となり，図形処理教育への適用が期待できる．

しかしながら，前報では３×３の変換マトリックスを使用しておきながら，その計算処理では，Excel の mmult 関数（行列の積を求める関数）を使用せずに式(3.5) で定義された代数式を使用したことと任意の点を考慮した図形処理，例えば，任意の点を中心とした回転な

どについては検討していなかった．本報告では，前報で課題であった計算処理を Excel の mmult 関数を使用して，かつ，任意の点を考慮した図形処理を合成マトリックスの適用の可能性を検討したものである．

２．本研究の目的

本研究の目的は，前報と同様に二次元図形処理を代数式ではなく，３×３の変換マトリックスを適用し，平行移動処理の変換マトリックスと原点における回転処理の変換マトリックスの積，すなわち，合成マトリックスを任意の点を考慮した二次元図形処理に適用しようとするものである．

この結果，前報で取り扱った基本的な二次元図形処理よりも高度な二次元図形処理の座標変換プロセスが可視可できることから，図形処理教育への適用や貢献がより期待できる．

３．代数式と変換マトリックス

二次元の座標Ｐ（ｘ，ｙ）は，二次元列ベクトル［ｘ　ｙ］で表現することができるので，式(3.1) の２×２の変換マトリックスを導入する．

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \cdots\cdots \text{式(3.1)}$$

次に，二次元列ベクトルと式(3.1)で定義した変換マトリックスとの積を考えると，式(3.2) になる．

$$[x \quad y]\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (ax+cy) & \left(bx+dy\right) \end{bmatrix} \quad \cdots\cdots \text{式(3.2)}$$

ここで，式(3.3) のようにｘ′，ｙ′を定義する．

$$\begin{aligned} x' &= ax+cy \\ y' &= bx+dy \end{aligned} \quad \cdots\cdots \text{式(3.3)}$$

式(3.1)と式(3.3)より，式(3.4) が得られる．

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \cdots\cdots \text{式(3.4)}$$

式(3.4)が意味することは，二次元図形の任意の座標Ｐ（ｘ，ｙ）の幾何学的な変換は，２×２の変換マトリックスを導入することで，新しい座標Ｐ′（ｘ′，ｙ′）に変換されることを示している．この変換は，回転，スケーリング，反転およびせん断の各処理で利用できる．しかしながら，平行移動においては，このような平方行列とはならないので，２×２の行列では定義できない．

そこで，３×３の変換マトリックスの導入を考慮する．

つまり，二次元列ベクトル［ｘ　ｙ］で表現していた座標に対して，任意の定数wとして三次元列ベクトル［wｘ　wｙ　w］を使用する．

この座標のことを同時座標と呼び，二次元のコンピュータ・グラフィクスでは，この定数wを１とおいている．

したがって，式(3.4)は，式(3.5) のように定義される．

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

$$= \begin{bmatrix} (ax+cy+e) & (bx+dy+f) & 1 \end{bmatrix}$$

$\cdots\cdots$ 式(3.5)

４．二次元図形処理のアルゴリズムと３×３の変換マトリックス

二次元図形処理では，平行移動，回転，スケーリング，反転，せん断の５つの処理が挙げられる．次に，各々の代数式によるアルゴリズムとそれに対応する３×３の変換マトリックスを示す．

4.1 平行移動

二次元上の任意の点Ｐ（ｘ，ｙ）をＸ方向の dx，Ｙ方向に dy だけ平行移動した時の新しい点Ｐ′（ｘ′，ｙ′）は，式(4.1）のようになる．

$$\begin{aligned} x' &= x + dx \\ y' &= y + dy \end{aligned} \quad \cdots\cdots \text{式（4.1）}$$

また，３×３の変換マトリックスでは，式(4.2）のようになる．

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix} \cdots\cdots \text{式(4.2)}$$

4.2 原点を中心とした回転

二次元上の任意の点Ｐ（ｘ，ｙ）を，原点を中心として反時計方向に θ だけ回転した時の新しい点Ｐ′（ｘ′，ｙ′）は，式(4.3）のようになる．

$$\begin{aligned} x' &= x \cdot \cos\theta - y \cdot \sin\theta \\ y' &= x \cdot \sin\theta + y \cdot \cos\theta \end{aligned} \quad \cdots\cdots \text{式（4.3）}$$

また，３×３の変換マトリックスでは，式(4.4）のようになる．

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots\cdots \text{式(4.4)}$$

4.3 スケーリング

二次元上の任意の点Ｐ（ｘ，ｙ）を，Ｘ方向に$Ｓx$倍，Ｙ方向に$Ｓy$倍のスケーリングをした時の新しい点Ｐ′（ｘ′，ｙ′）は，式(4.5）のようになる．

なお，$Ｓx$，$Ｓy$をスケーリング係数と呼ぶ．

$$\begin{aligned} x' &= Sx \cdot x \\ y' &= Sy \cdot y \end{aligned} \quad \cdots\cdots \text{式（4.5）}$$

また，３×３の変換マトリックスでは，式(4.6）のようになる．

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots\cdots \text{式(4.6)}$$

4.4 反転

二次元上の任意の点Ｐ（ｘ，ｙ）を，Ｘ方向に$Ｈx$倍，Ｙ方向に$Ｈy$倍の反転をした時の新しい点Ｐ′（ｘ′，ｙ′）は，式(4.7）のようになる．

反転は，4.3のスケーリングにおいて，スケーリング係数が負の場合に相当する．

$$\begin{aligned} x' &= Hx \cdot x \\ y' &= Hy \cdot y \end{aligned} \quad \cdots\cdots \text{式（4.7）}$$

また，３×３の変換マトリックスでは，式(4.8）のようになる．

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} Hx & 0 & 0 \\ 0 & Hy & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots\cdots \text{式(4.8)}$$

4.5 せん断※

二次元上の任意の点Ｐ（ｘ，ｙ）を，Ｘ方向Dx，Ｙ方向にDyだけ力を作用させた時の新しい点Ｐ′（ｘ′，ｙ′）は，式(4.9) のようになる．

$$\begin{aligned} x' &= x + y \cdot Dx \\ y' &= x \cdot Dy + y \end{aligned} \quad \cdots\cdots \text{式 (4.9)}$$

また，３×３の変換マトリックスでは，式(4.10)のようになる．

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & Dx & 0 \\ Dy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots\cdots \text{式(4.10)}$$

※せん断：物体内部の任意の面に関して面に平行方向に力を作用させた状態を指す．それによる物体の変形をせん断変形と呼ぶ．

５．任意の点を中心とした回転

二次元上の任意の点Ｐ（ｘ，ｙ）を，任意の点Ｐａ（ｘａ，ｙａ）を中心として反時計方向にθだけ回転した時の新しい点Ｐ′（ｘ′，ｙ′）は，式(5.1) のようになる．

$$\begin{aligned} x' &= (x - xa)\cos\theta - (y - ya)\sin\theta + xa \\ y' &= (x - xa)\sin\theta + (y - ya)\cos\theta + ya \end{aligned} \quad \cdots\cdots \text{式 (5.1)}$$

また，３×３の変換マトリックスでは，次のような操作が必要になる．

1)任意の点Ｐａ（ｘａ，ｙａ）を原点に移動する．

これは，点Ｐを（－ｘａ，－ｙａ）だけ平行移動したことと同じである．

平行移動した点をＰ１（ｘ１，ｙ１）とすると式(5.2)のようになる．

$$\begin{bmatrix} x1 & y1 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xa & -ya & 1 \end{bmatrix} \cdots\cdots 式(5.2)$$

2)点Ｐ１（ｘ１，ｙ１）を，原点を中心に半時計方向に回転する．
回転した時の新しい点をＰ２（ｘ２，ｙ２）とすると式(5.3）のようになる．

$$\begin{bmatrix} x2 & y2 & 1 \end{bmatrix} = \begin{bmatrix} x1 & y1 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots\cdots 式(5.3)$$

3)回転した点Ｐ２（ｘ２，ｙ２）を（ｘａ，ｙａ）だけ元の座標へ移動する．
平行移動した点をＰ′（ｘ′，ｙ′）とすると式(5.4）のようになる．

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x2 & y2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xa & ya & 1 \end{bmatrix} \cdots\cdots 式(5.4)$$

これらのプロセスをまとめると，以下のような処理となり，変換マトリックスの積により，任意の点を中心とした回転が求められる．

1)負の平行移動処理
2)原点を中心とした回転処理
3)正の平行移動処理

そこで，式(5.2)～式(5.4)をまとめると式(5.5）が得られる．

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xa & -ya & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xa & ya & 1 \end{bmatrix}$$

$$= \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ xa(1-\cos) & ya(1-\cos\theta) & 1 \\ +ya\sin\theta & -xa\sin\theta & \end{bmatrix}$$

$$= \begin{bmatrix} (x-xa)\cos\theta & (x-xa)\sin\theta & \\ -(y-ya)\sin\theta+xa & +(y-ya)\cos\theta+ya & 1 \end{bmatrix}$$

・・・・式(5.5)

なお，式(5.5)の最終結果は，代数式で表現した式(5.1)と一致する．なお，式(5.5) によるマトリックスのことを合成マトリックスと呼ぶ.

６．実施例

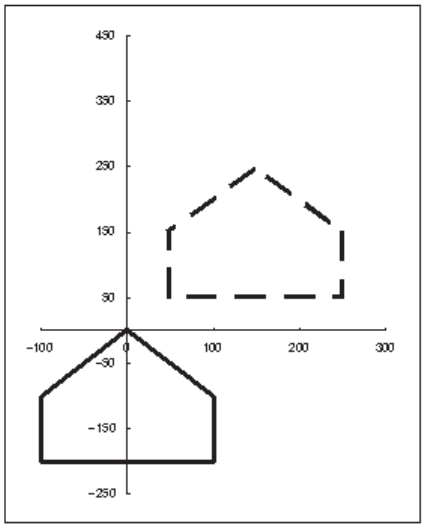
式(5.5) で表されるプロセスを，五角形の頂点Ｐ（150,250）を中心として反時計方向に 180 度回転させる処理の例を以下に示す．なお，五角形の元座標と変換された座標および元の図形は，破線で表示し，変換させた図形は，実線で表示してある．

6.1 原点への移動

図 6.1 は，五角形を原点まで移動した場合の実行例である．また，実行例の下はそれを数式表示したものである．なお，図形の表示は，Excel のグラフ機能の散布図を利用している．

	X座標	Y座標	w
元の座標	50	50	1
	50	150	1
	150	250	1
	250	150	1
	250	50	1
	50	50	1
変換した座標	-100	-200	1
	-100	-100	1
	0	0	1
	100	-100	1
	100	-200	1
	-100	-200	1

変換マトリックス	X方向	100	Y方向	250
	1	0	0	
	0	1	0	
	-150	-250	1	



	X座標	Y座標	w
元の座標	10	10	1
	10	30	1
	30	50	1
	50	30	1
	50	10	1
	10	10	1

変換した座標	=MMULT(C3:E3,$H$3:$	=MMULT(C3:E3,$H$3:$	=MMULT(C3:E3,$H$3:$
	=MMULT(C4:E4,$H$3:$	=MMULT(C4:E4,$H$3:$	=MMULT(C4:E4,$H$3:$
	=MMULT(C5:E5,$H$3:$	=MMULT(C5:E5,$H$3:$	=MMULT(C5:E5,$H$3:$
	=MMULT(C6:E6,$H$3:$	=MMULT(C6:E6,$H$3:$	=MMULT(C6:E6,$H$3:$
	=MMULT(C7:E7,$H$3:$	=MMULT(C7:E7,$H$3:$	=MMULT(C7:E7,$H$3:$
	=MMULT(C8:E8,$H$3:$	=MMULT(C8:E8,$H$3:$	=MMULT(C8:E8,$H$3:$

図 6.1 原点への移動の実行例

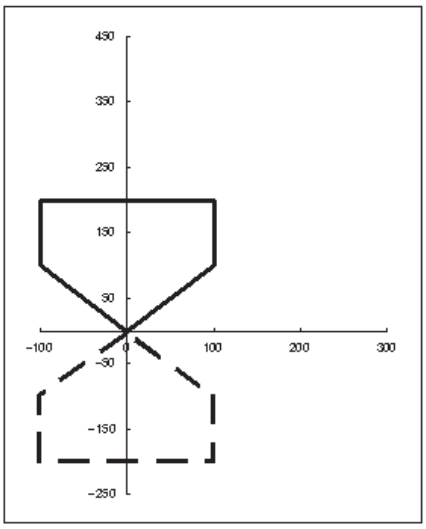
6.2 原点を中心に回転

図 6.2 は, 五角形を, 原点を中心に反時計方向に 180 度回転させた場合の実行例である. また, 実行例の下はそれを数式表示したものである.

	X座標	Y座標	w
元の座標	-100	-200	1
	-100	-100	1
	0	0	1
	100	-100	1
	100	-200	1
	-100	-200	1

変換したマトリクス	角度	[illegible]	ラジアン	3.141593
	-1	1.23E-16	0
	-1.2E-16	-1	0
	0	0	1

変換した座標	100	200	1
	100	100	1
	0	0	1
	-100	100	1
	-100	200	1
	100	200	1



	X座標	Y座標	w
元の座標	-20	-40	1
	-20	-20	1
	0	0	1
	20	-20	1
	20	-40	1
	-20	-40	1

変換した座標	=MMULT(C4:E4,$H$3:$	=MMULT(C4:E4,$H$3:$	=MMULT(C4:E4,$H$3:$
	=MMULT(C5:E5,$H$3:$	=MMULT(C5:E5,$H$3:$	=MMULT(C5:E5,$H$3:$
	=MMULT(C6:E6,$H$3:$	=MMULT(C6:E6,$H$3:$	=MMULT(C6:E6,$H$3:$
	=MMULT(C7:E7,$H$3:$	=MMULT(C7:E7,$H$3:$	=MMULT(C7:E7,$H$3:$
	=MMULT(C8:E8,$H$3:$	=MMULT(C8:E8,$H$3:$	=MMULT(C8:E8,$H$3:$
	=MMULT(C9:E9,$H$3:$	=MMULT(C9:E9,$H$3:$	=MMULT(C9:E9,$H$3:$

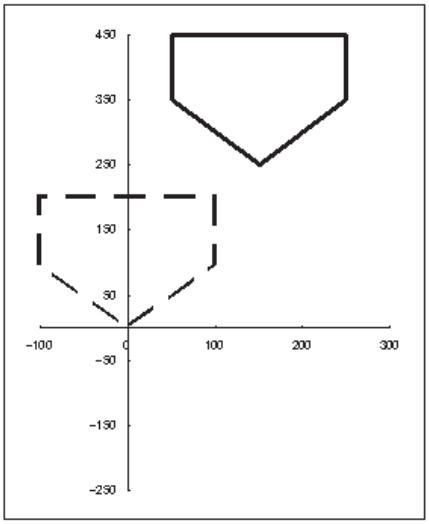
図 6.2 原点を中心とした回転の実行例

6.3 元の座標への移動

図 6.3 は，回転させた五角形を，元の座標まで移動した場合の実行例である．また，実行例の下はそれを数式表示したものである．

	X座標	Y座標	w
元の座標	100	200	1
	100	100	1
	0	0	1
	-100	100	1
	-100	200	1
	100	200	1
変換した座標	250	450	1
	250	350	1
	150	250	1
	50	350	1
	50	450	1
	250	450	1

変換マトリックス	X方向	[illegible]	Y方向	250
	1	0	0	
	0	1	0	
	150	250	1	



	X座標	Y座標	w
元の座標	20	40	1
	20	20	1
	0	0	1
	-20	20	1
	-20	40	1
	20	40	1

変換した座標	=MMULT(C3:E3,$H$3:$	=MMULT(C3:E3,$H$3:$	=MMULT(C3:E3,$H$3:$
	=MMULT(C4:E4,$H$3:$	=MMULT(C4:E4,$H$3:$	=MMULT(C4:E4,$H$3:$
	=MMULT(C5:E5,$H$3:$	=MMULT(C5:E5,$H$3:$	=MMULT(C5:E5,$H$3:$
	=MMULT(C6:E6,$H$3:$	=MMULT(C6:E6,$H$3:$	=MMULT(C6:E6,$H$3:$
	=MMULT(C7:E7,$H$3:$	=MMULT(C7:E7,$H$3:$	=MMULT(C7:E7,$H$3:$
	=MMULT(C8:E8,$H$3:$	=MMULT(C8:E8,$H$3:$	=MMULT(C8:E8,$H$3:$

図 6.3 元の座標への移動の実行例

6.4 任意の点を中心とした回転の結果

以上の結果から，最終的には図 6.4 のように任意の点（この例では，五角形の頂点）を中心とした回転（180 度）が表現できる．

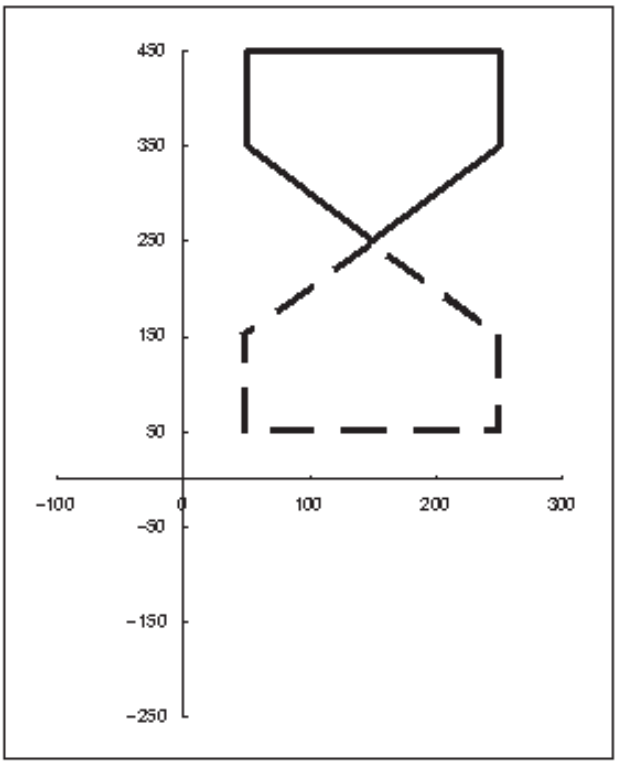


図 6.4 任意の点を中心とした回転の最終結果

7．考察

任意の点を中心とした図形処理の例として，任意の点を中心とした回転処理に提案した変換マトリックスの積，すなわち，合成マトリックスを適用した結果，原点への移動，原点を中心に回転，元の座標への移動の３つのプロセスでプログラム言語を必要とせずに容易に座標変換のプロセスとそれに伴う図形が表示できることが示された．

したがって，基本的な二次元図形処理よりも高度な二次元図形処理の座標変換プロセスが可視化できることから，図形処理教育への適用や貢献が期待できると考えられる．

参考に，この処理をプログラム言語(Visual Basic　6.0）で作成したコードが図 7.1 である．このように，プログラム言語では，図に示したように冗長なコードが必要になり，かつ，途中の計算プロセスは表示されない．もちろん，コードを更に追加すれば可能ではあるが，プログラム言語ではその処理は大変手間がかかるものである．

図 7.2 に，図 7.1 で示したコードの実行結果を示す．

```
Private Sub Form_Click()
’クリックすると実行を終了します.
    End
End Sub
Private Sub Ini(Title$)
’グラフィックの初期設定を行います.
’最大化の設定をします.
    WindowState = 2
’ピクセルの設定をします.
    ScaleMode = 3
’バックカラーを明るい白に設定します.
   BackColor = QBColor(15)
’グラフィック環境をONにします.
    AutoRedraw = "True"
’タイトルを指定します.
    Caption = Title$
End Sub
Private Sub Form_Load()
    Dim x_data(6), y_data(6)
’任意に点を中心とした回転の例
    Ini ("任意の点を中心とした回転の例")
    Cls ’画面をクリアにします.
’データの定義
    x_data(1) = 50: y_data(1) = 50
    x_data(2) = 50: y_data(2) = 150
    x_data(3) = 150: y_data(3) = 250
    x_data(4) = 250: y_data(4) = 150
    x_data(5) = 250: y_data(5) = 50
    x_data(6) = 50: y_data(6) = 50
’線分の太さを指定します.
    DrawWidth = 1
’線種を破線に指定します.
    DrawStyle = 1
    For i = 1 To 6
        xp = x_data(i): yp = y_data(i)
’iが1なら(xp,yp)へ移動します.
```

```
’そうでなければ(xp,yp)まで線分を描きます.
        If i = 1 Then
                    PSet (xp, Cd(yp))
                Else
                    Line -(xp, Cd(yp))
        End If
    Next i
’マチンの公式でπを定義します.
    PAI = 16 * Atn(1 / 5) - 4 * Atn(1 / 239)
’度をラジアンに変換する係数を求めます.
    rad = PAI / 180
’任意の点を定義します.
    xa = 150: ya = 250
’任意の点を円で表示します.
    Circle (xa, Cd(ya)), 2
’回転角度を定義します.
    the = 180
’度をラジアンへ変換します.
    the = (-the) * rad
’線分の太さを指定します.
    DrawWidth = 2
    For i = 1 To 6
        xp = x_data(i): yp = y_data(i)
’回転の計算処理をします.
        kcos = Cos(the): ksin = Sin(the)
        xpa = xp - xa: ypa = yp - ya
        x_tran = xpa * kcos - ypa * ksin + xa
        y_tran = xpa * ksin + ypa * kcos + ya
’ｉが１なら(x_tran,y_tran)へ移動します.
’そうでなければ(x_tran,y_tran)まで線分を描きます.
        If i = 1 Then
                    PSet (x_tran, Cd(y_tran))
                Else
                   Line-(x_tran, Cd(y_tran))
        End If
    Next i
```

```
End Sub
Private Function Cd(Y)
’座標変換処理をします.
    Cd = 500 - Y
End Function
```

図 7.1 プログラム言語(Visual Basic 6.0) によるコード例

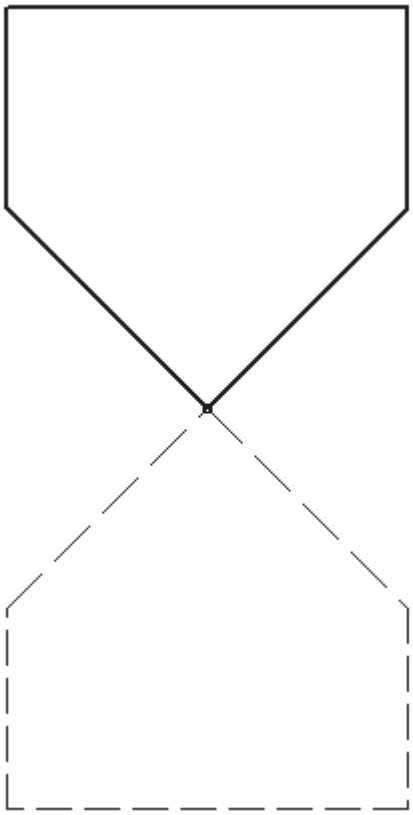


図 7.2 実行結果

８．まとめと今後の課題

本研究では，二次元図形処理を代数式ではなく，３×３の変換マトリックスを適用し，表計算ソフトウェアであるExcel の機能（関数機能およびグラフ機能）を使用して，平行移動処理の変換マトリックスと原点における回転処理の変換マトリックスの積である合成マトリックスを任意の点を考慮した図形処理（回転処理）に適用した.

その結果，基本的な図形処理よりも高度な二次元図形処理の座標変換プロセスが可視化できることが示され，より理解し易い図形処理教育への適用や貢献が期待できるので本研究の目的を達成することができた.

今後の課題としては，任意の点を中心としたスケーリング，任意の点を中心とした反転など任意の点の回転処理以外の二次元処理にも適用を考えたい.

さらには，Excelにおける定常的な処理をマクロ化することで，より効果的な図形処理教育への適用に関して検討したい.

参考文献

[1] 光成豊明(2012):「Excelによる二次元図形処理」，明星大学経営学部研究紀要，Vol.3
[2] 光成豊明(2001):「Visual BasicによるＣＧ」，産業図書
[3] 光成豊明(1995):「BasicによるＣＧ入門」，産業図書