易凡轉換在懷爾凡模式的應用

曹文瑜、楊惠貞、李雅如
國立勤益工商專校電算中心

摘 要

在電腦圖學最常用的轉換是易凡轉換[2]，因這種轉換有一個簡單的格式，共有4種基本的轉換 1.平移 2.放大縮小 3.旋轉 4.變形。任何一個轉換是由一個或數個基本轉換所組合成。在三度空間中先定一條線（圖1），對Z軸旋轉360度將得到像甜筒外殼一樣的圖形（圖2），叫做懷爾凡模式 (wireframe model)。依照所站的角度不同而顯示出看得見部份，將此看得見的部份再依易凡轉換轉成不同的圖形。

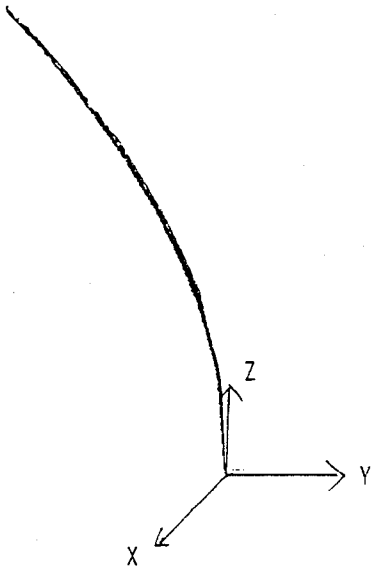


圖 (1)

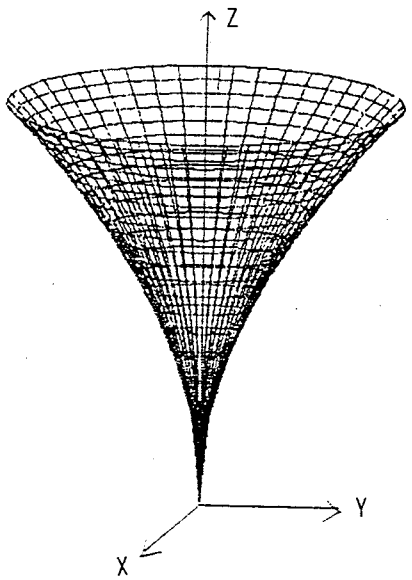


圖 (2)

關鍵字：易凡轉換、平移、放大縮小、旋轉、變形、線架構模式

ABSTRACT

The most common transformations used in Computer Graphics are the Affine Transformations, for they have a simple form. There are four elementary transformations: 1. translation 2.scaling 3.rotation 4.shearing. Any transformation is the combinations of one or more than one elemnetary transformations. In three dimensional space to generate wire and rotate 360° around Z axis, will get a shape like a crust of ice cream. We call it "Wireframe Model". Depending on the different site that you stand, you can see the visible part, and then using Affine Transformation to translate the visible part to different shape.

KEY WORD : Affine Transformation, translation, scaling, rotation, shearing, Wireframe Model

前　言

Affine Transformation 有個簡單的格式，例如：在二維平面上的一點 $P = (P_x, P_y)$由 Affine Transformation 映射至對應點 $Q = (Q_x, Q_y)$，P, Q 之間就有個對應關係爲：

$$Q_x = aP_x + cP_y + trx$$
$$Q_y = bP_x + dP_y + try \qquad 式(1)$$

其中 a,b,c,d,trx,try 爲常數且$ad \neq bc$

trx 爲 X 軸的位移

try 爲 Y 軸的位移

這樣的新組合是線性組合(linear combination)

改寫式(1) 爲有用的矩陣，如下：

$$(Q_x, Q_y) = (P_x, P_y) \begin{pmatrix} a & b \\ c & d \end{pmatrix} + (trx, try)$$

簡單表示爲$Q = PM + TR$，以圖(3)表示之

其中 $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

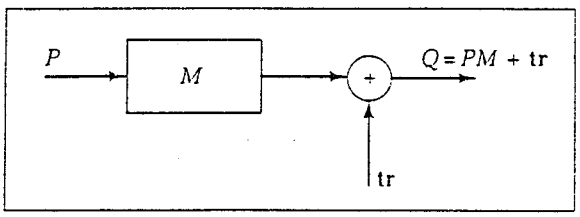


圖 (3)

基本上有 4 種轉換：1.平移　2.放大縮小

3.旋轉　4.變形

格式如下：

1.平移(translation)

$M = 0$　0 爲零矩陣

2.放大縮小(scaling)

$M = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix}$　S_x, S_y 爲常數

3.旋轉(rotation)

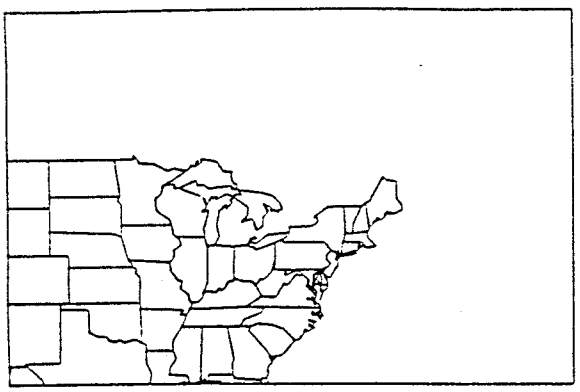
旋轉角度：θ

$$M = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

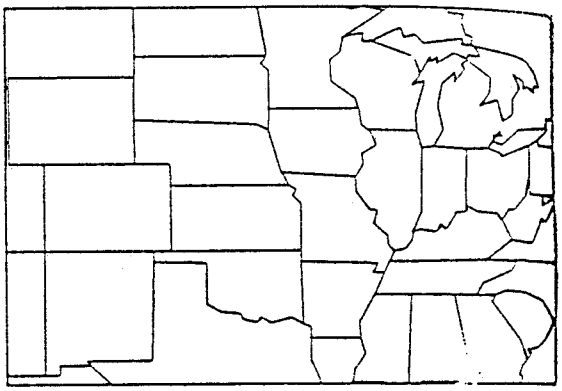
4.變形(shearing)

$M = \begin{pmatrix} k_1 & 1 \\ 1 & k_2 \end{pmatrix}$　k_1, k_2 爲常數

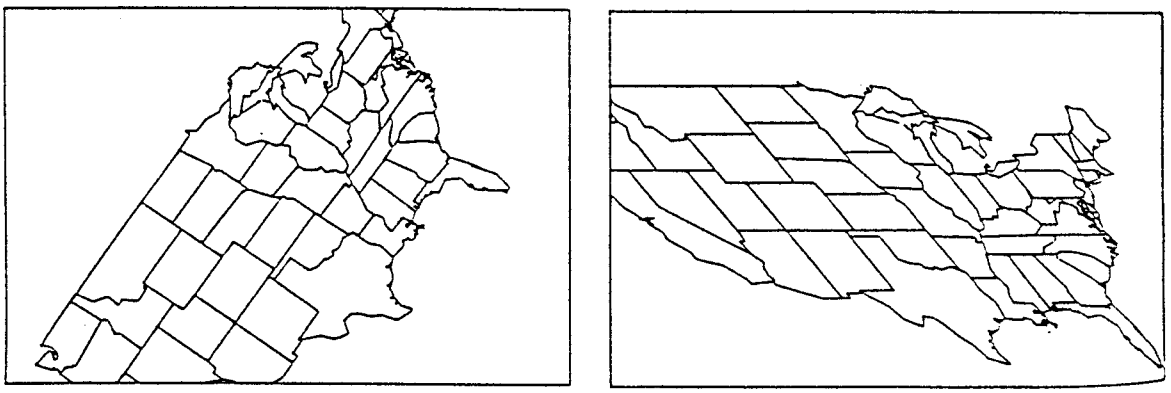
圖形(4)說明四種轉換圖形：



平移



放大縮小



旋轉　　變形

圖 (4)

理論說明

在 X,Z 平面上，給一個曲線X=f(v)=v*v，Z=g(v)=sqrt(80*v)−1 [4] 對 z 軸旋轉360度將得到一像甜筒外殼一樣的表層（如圖２），隨著X,Z 不同的函數可得到不同的圖形（圖５、６）。

此表層以式子表示之

$$S(v,\theta)=(f(v)\cos(\theta),f(v)\sin(\theta),g(v))$$

v 的範圍　$0 \leq v \leq k$　k爲大於零的任一常數

θ 的範圍　$0 \leq \theta \leq 360$

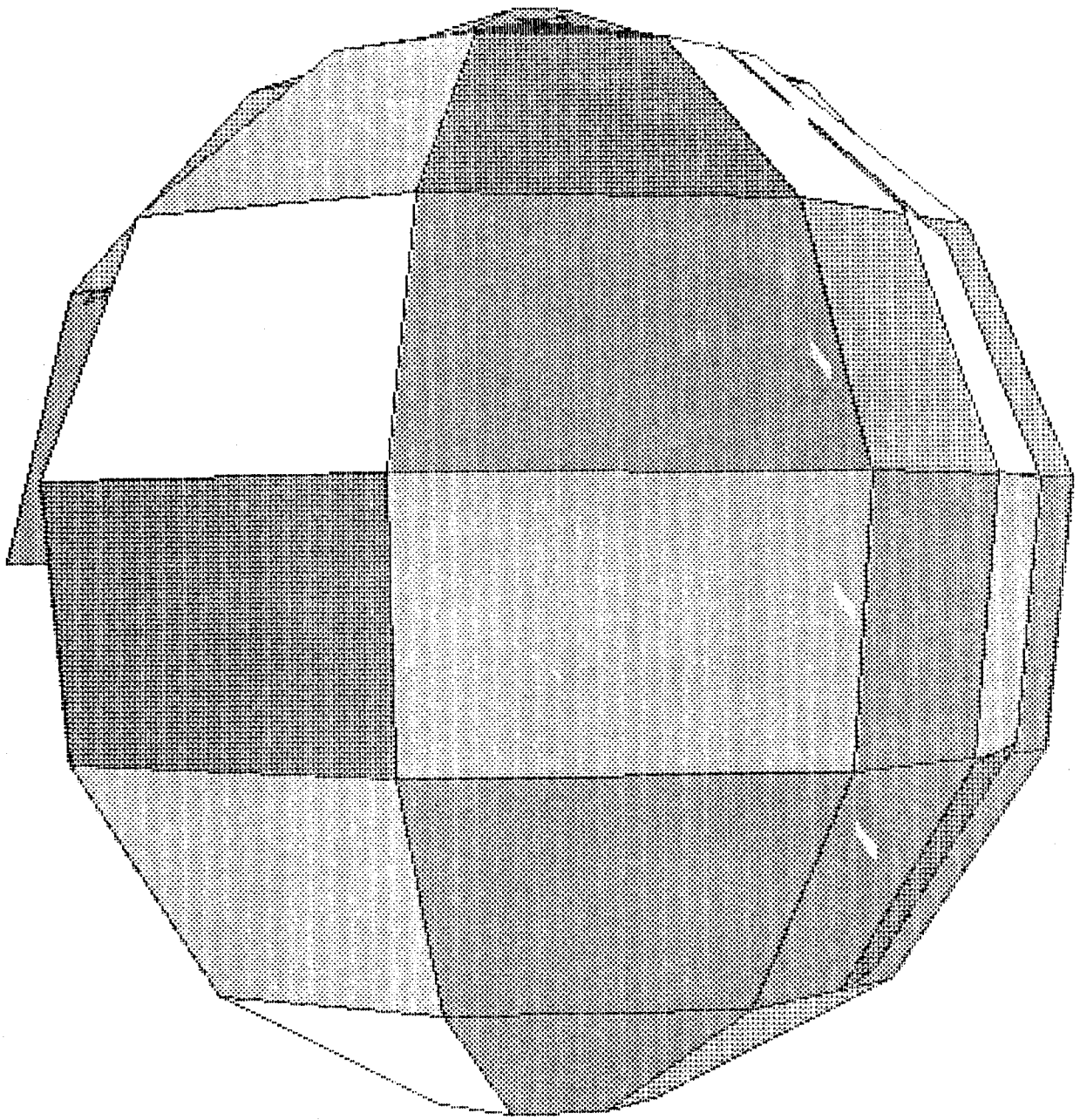
$$\text{其中 } M=\begin{pmatrix}\cos(\theta) & 0 & 0\\ 0 & \sin(\theta) & 0\\ 0 & 0 & 0\end{pmatrix}$$

使得$S(v,\theta) = (f(v),f(v),g(v)\cdot M$

將此表層細分成若干塊，每一塊稱爲一補丁，再將這些塊上的邊界點儲存在二維矩陣A(i,j)中，如何決定此塊補丁爲可見的(visible) 由每一個補丁的向外正交單位向量n 和眼睛所在的位置來決定。

The image picture on Y-Z plane

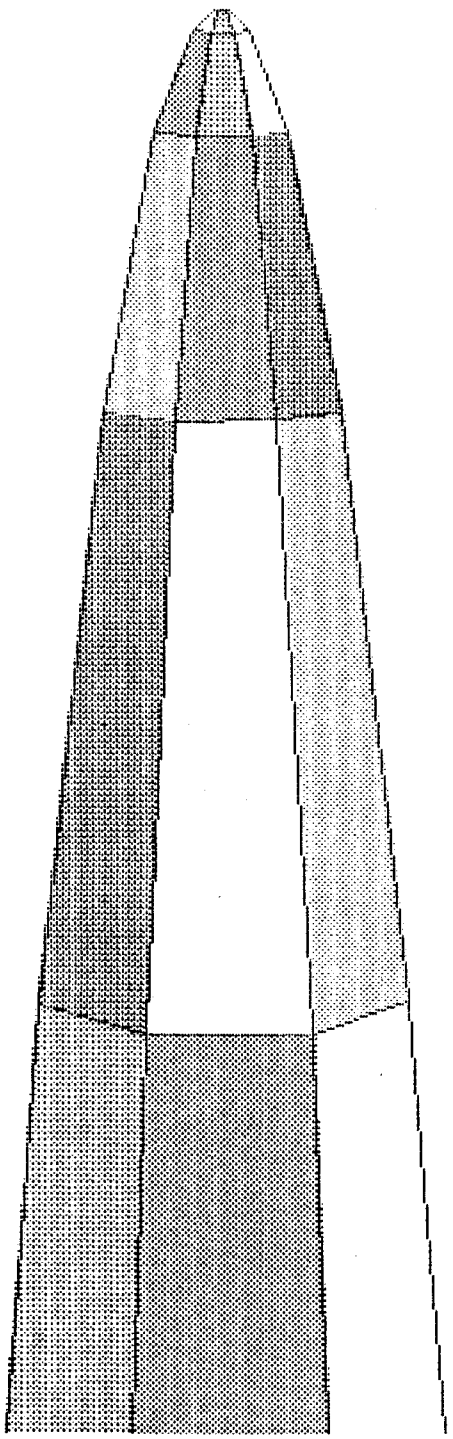
$$X = 8\cos(V)$$
$$Z = 8\sin(v)$$



圖(5)

The image picture on X=Z plane

$$X = v^2$$
$$Z = \sqrt{X}$$



圖(6)

假設一：

我們的眼睛在 X 座標軸上，垂直於顯示視窗(view plane)X=v上，眼睛的座標爲e(v,0,0)（圖7）。

步驟如下：

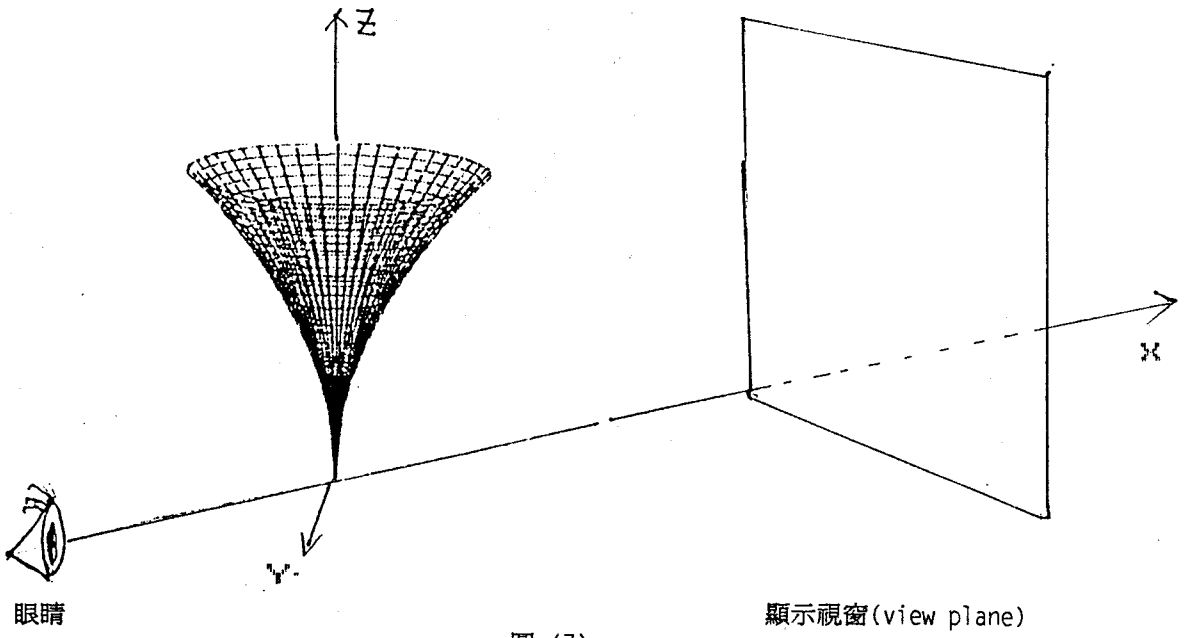


圖 (7)

步驟1.將此表層細分成若干塊（稱爲補丁），再將這些補丁的邊界四點儲存在二維矩陣A(i,j)中。

步驟2.將這些補丁映射至顯示視窗(view plane) X = v 上，並將這些映射點存在P(i,j)中。

步驟3.找出每一個補丁的向外正交單位向量n

$$n = [(A(I,J)-A(I,J+1)]X[A(I+1,J+1)-A(I,J+1)]$$

註：X 表示 cross product

在 Z 軸上自定表層內部一點 in_pt=(0,0,g(v))

由n 和in_pt_A[i1,j1+1]的內積來決定若大於 0，則 n 指向外，否則改變 n 的方向 n = −n

步驟4.決定向外正交單位量n 和e-a(i,j+1)之間的內積若大於 0，則此補丁爲可見的，否則爲不可見。

步驟5.將這些可見的補丁以彩色一塊一塊的映射至顯示視窗(view plane) X = v 上。見圖形(8)。

假設二：

再將我們的眼睛移在Y 座標軸上，這些可見的補丁以彩色映射至顯示視窗(view plane) Y = v 上，見圖形(9)。

假設三：

最後將我們的眼睛移至Z 座標軸上，這些可見的補丁以彩色映射至顯示視窗(view plane) Z=v 上，見圖形(10)。

再利用affine transformation 將view plane X=v 上補丁原形，旋轉，變形，縮小畫出圓形，見圖形(11,12,13,14)

The image picture on X=v plane

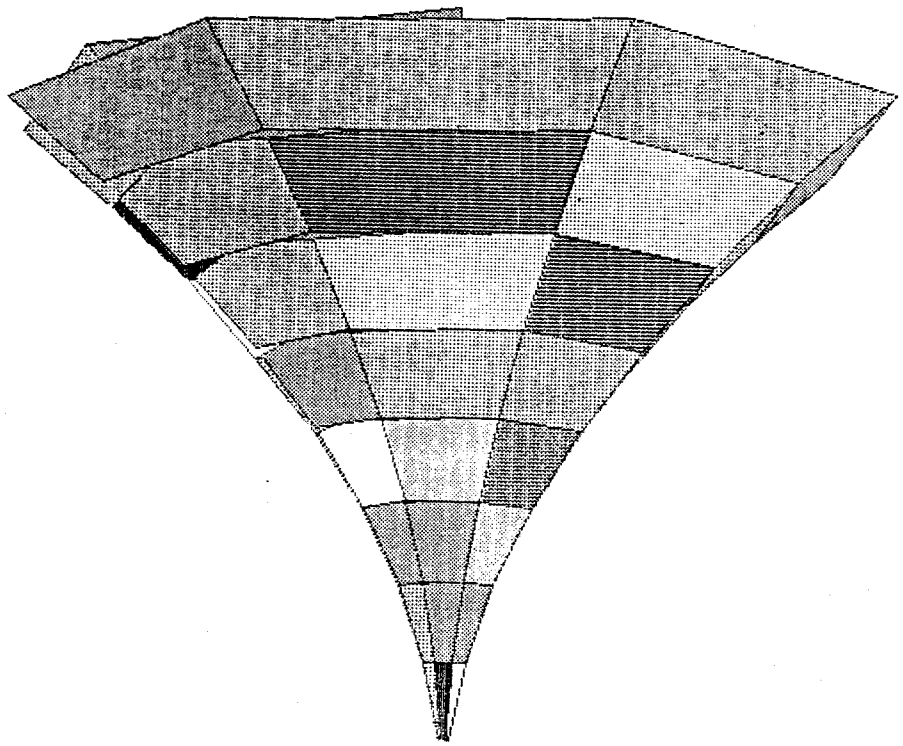


圖 (8)

The image picture on Y=v plane

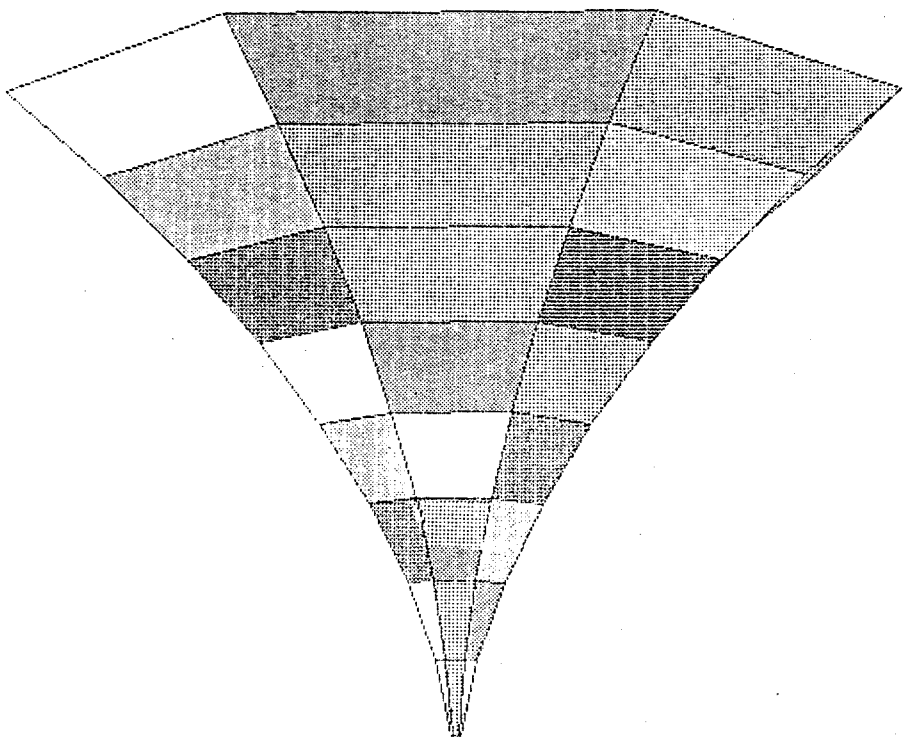


圖 (9)

The image picture on X-Y plane

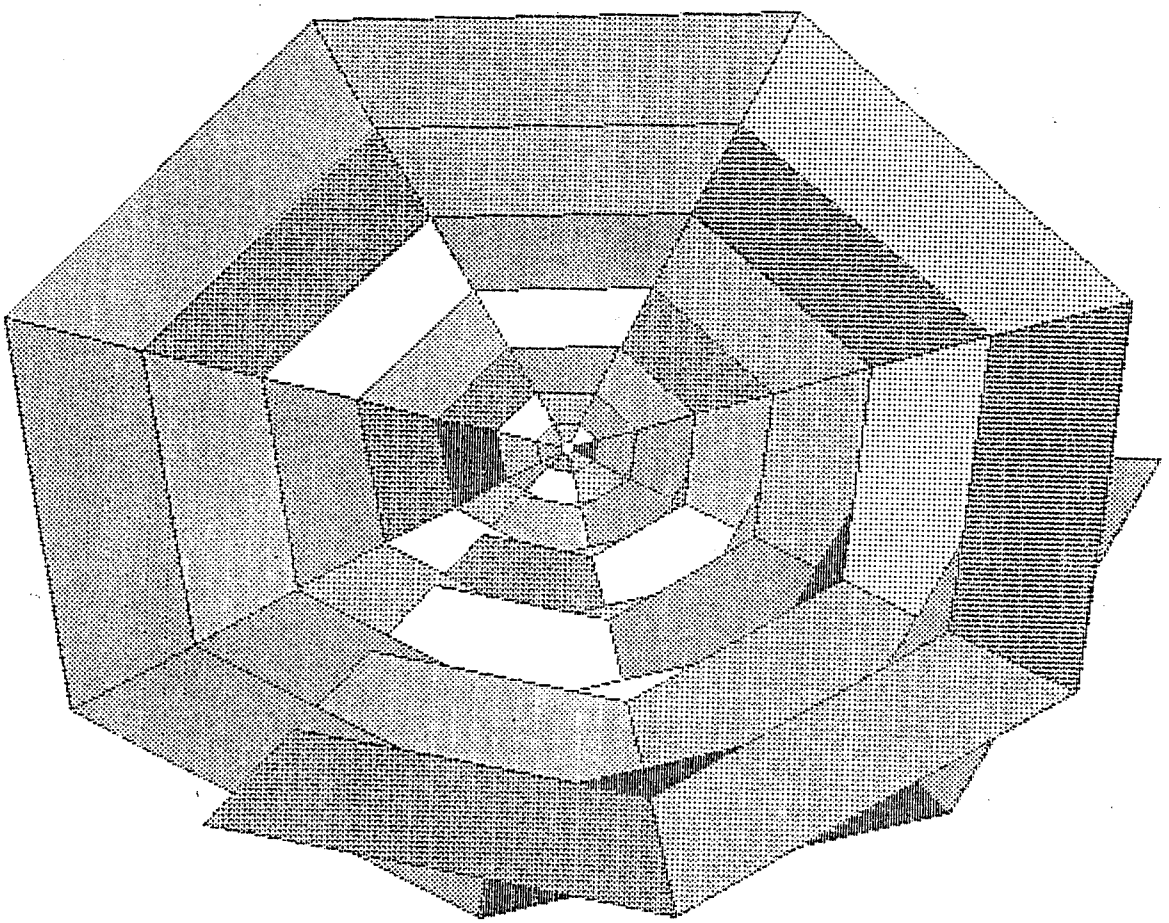


圖 (10)

The image picture on X=V plane

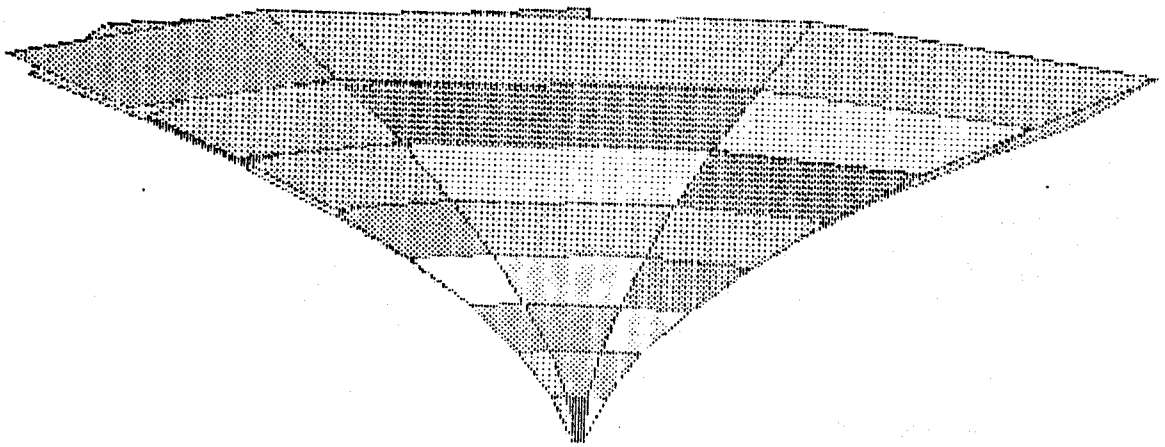


圖 (11)

The image picture of rotation on X=v plane

$$\text{matrix} = \begin{bmatrix} \cos(\pi) & \sin(\pi) \\ -\sin(\pi) & \cos(\pi) \end{bmatrix}$$

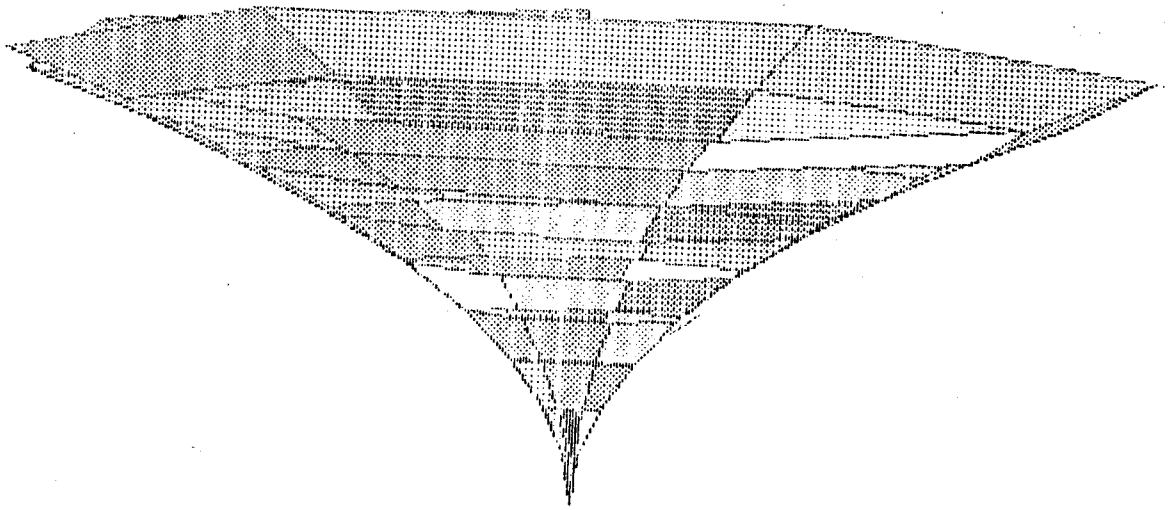


圖 (12)

The image picture of shearing on X=v plane

$$\text{matrix} = \begin{bmatrix} 1 & 0.5 \\ -0.1 & 1 \end{bmatrix}$$

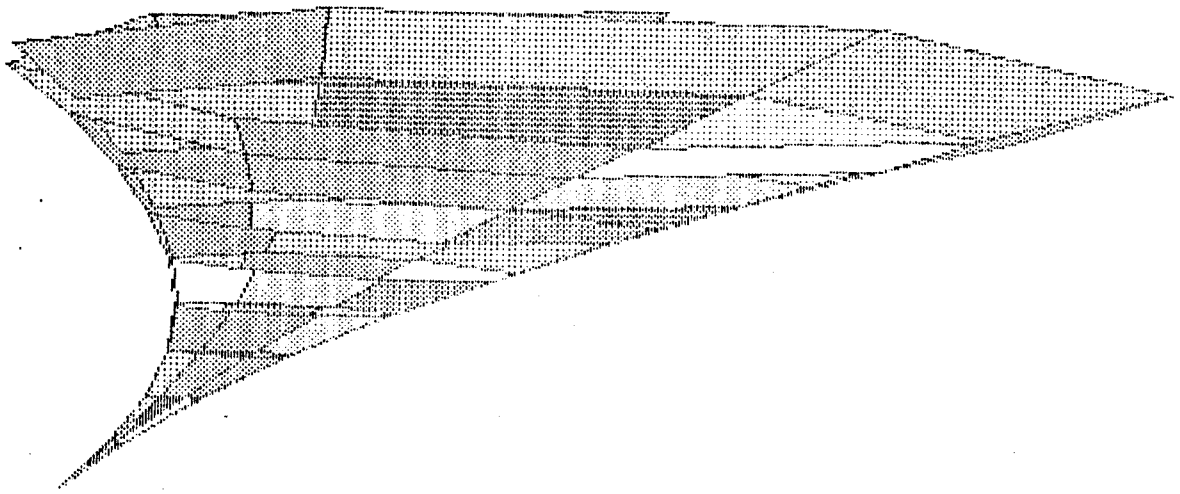


圖 (13)

The image picture of scaling on X=v plane

$$\text{matrix} = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}$$

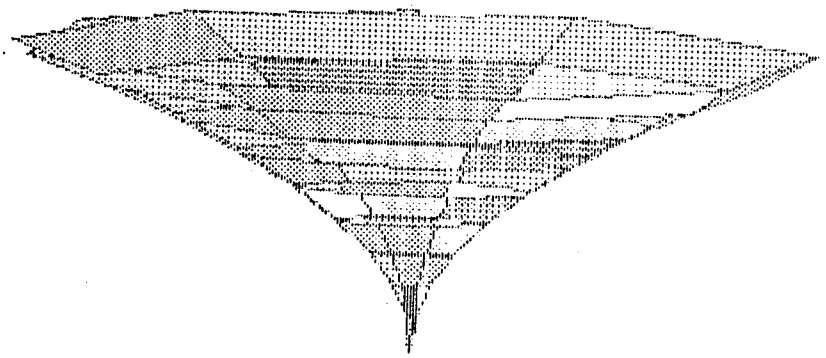


圖 (14)

注意事項：

1.在眞實世界轉換成輸出設備(viewport)所定的範圍影響圖形顯示的部份，就像開一視窗(window)，只能看見在視窗範圍內的圖形。

2.補丁取的大小可影響到圖形的精確度。

3.view plane距離原點的遠近影響圖形的大小。

結　論

在一Wireframe model 中的圖形若不管此補丁爲可見，圖形會顯得雜亂（見圖15），可由各角度來觀察，可得到較清楚的圖形（見圖９），用途甚廣。

Affine ransformation 中 M 矩陣就像一個黑箱子，在日常生活中有一事件發生，多多少少會有一些線索留下，再由黑箱子將模糊不清的線索轉換成清楚而有用的資訊，或由各角度來觀察，可得到更多不同的資訊以供參考。

參考文獻

1.書籍：TURBO PASCAL 5.5 OOP GUIDE

2.書籍：COMPUTER GRAPHICS
-F.S. Hill Jr. P330～p337.

3.書籍：An Introduction To Computer Science
An Algorithmic Approach
-JEAN_PAUL TREMEBLAY RICHARD B. BUNT

4.書籍：Calculus
-Saturnino L. Salas Einar Hille

The image picture on Y-Z plane

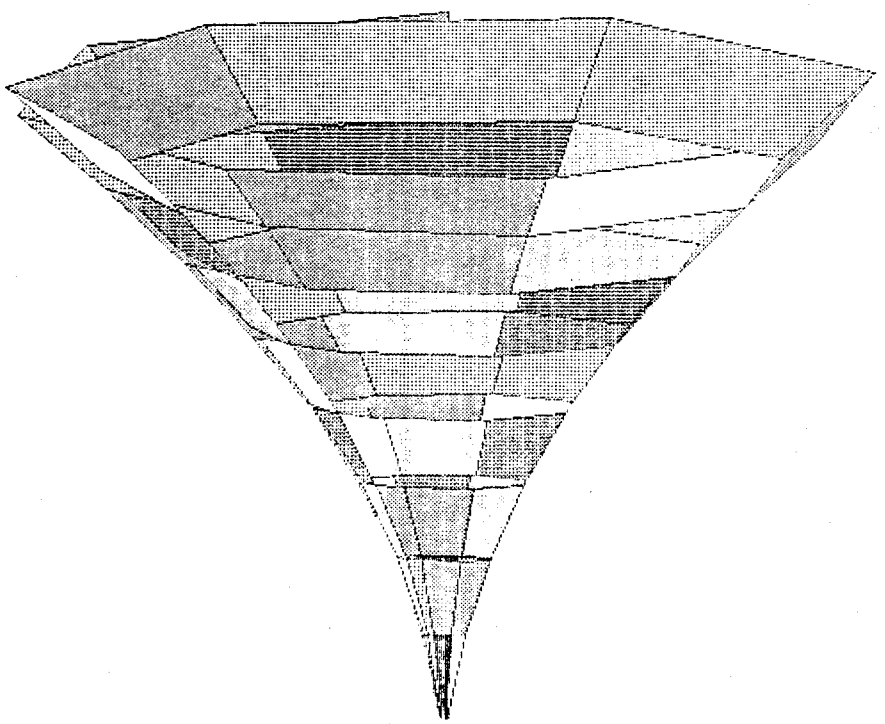


圖 (15)

```
{*********************************************************************
 *                    CS 745 COMPUTER GRAPHICS                       *
 *                    ------------------------                       *
 *                                                                   *
 * Instructor : Dr. Bein                        sid : 900-50-2810    *
 * Due        : April. 29, 1992                name : WEN_YU TSAO   *
 *********************************************************************
use       : turbo pascal 6.0
file name : g4.pas}

program main;
uses Graph;
const
        TWOPI  = 2*3.14159;
        maxnum = 20;
type    point = record
                x : real;
                y : real;
                z : real;
        end;

type    dev_point = record
                 iy : integer;
                 iz : integer;
        end;
type    image_point = record
                  y : real;
                  z : real;
        end;
var
        gmode,errorcode,gdriver : integer;
        i1,j1,i,j,i2            : integer;
        viewport                : viewporttype;
        a                       : array[1..maxnum,1..maxnum] of point:
        p,q,r,s                 : array[1..maxnum,1..maxnum] of image_point;
        in_pt,p1,p2,n,temp      : point;
        e,v,thelta,t1,t2        : real;
        center                  : dev_point;
        im1,im2                 : image_point;
        fig                     : array[1..8] of integer;
{-------------------------------------------------------------}
{                        WCtoDC point                           }
{-------------------------------------------------------------}
procedure WCtoDC(p:image_point;var dev_p:dev_point);
        { set Wb=-5, Wt=38, Wl=-25, Wr=25:
              Vb=349,Vt=0, Vl=0, Vr=639}
var
      wt,wb,wl,wr           : real;
      vt,vb,vl,vr           : integer;
begin
      wt := 65.0; wb := -45.0;    wl := -25.0; wr := 35.0;
      vt := 0;    vb := getmaxy; vl := 0;     vr := getmaxx;
      if (wl-wr <> 0) and (wt-wb <> 0) then
      begin
      dev_p.iy := round((vl-vr)/(wl-wr)*p.y+(wl*vr-wr*vl)/(wl-wr));
      dev_p.iz := round((vt-vb)/(wt-wb)*p.z+(vb*wt-wb*vt)/(wt-wb));
      end;
end;

function f(v:real):real;
begin
     f := v*v;
end;
```

```
function g(v:real):real;
begin
     g := 8*v-1 ;
end;

procedure cross(c,d:point;var temp:point);
begin
     temp.x := c.y*d.z - c.z*d.y;
     temp.y := d.x*c.z - c.x*d.z;
     temp.z := c.x*d.y - c.y*d.x;
end;

function dot(p1,p2:point):real;
begin
     dot := p1.x*p2.x + p1.y*p2.y + p1.z*p2.z;
end;

procedure rotate(var a1,a2:image_point);
var kk :real;
begin
      kk :=- 3.14159;
      a2.y:=a1.y*cos(kk)-a1.z*sin(kk);
      a2.z:=a1.y*sin(kk)+a1.z*cos(kk);
end;

procedure shear(var a1,a2:image_point);
begin
      a2.y:=a1.y+a1.z*0.5;
      a2.z:=a1.y*(-0.1)+a1.z;
end;

procedure scale(var a1,a2:image_point);
begin
      a2.y:=a1.y*0.7;
      a2.z:=a1.z*0.7;
end;

procedure draw_pic(t1,t2,t3,t4:image_point);
begin
      WCtoDC(t1,center);
      fig[1] := center.iy;
      fig[2] := center.iz;
      WCtoDC(t2,center);
      fig[3] := center.iy;
      fig[4] := center.iz;
      WCtoDC(t3,center);
      fig[5] := center.iy;
      fig[6] := center.iz;
      WCtoDC(t4,center);
      fig[7] := center.iy;
      fig[8] := center.iz;
      for i2 := 1 to 8 do
      begin
            SetFillStyle(SolidFill,(i1*j1 mod 15));
            DrawPoly(4,fig);
            FillPoly(4,fig);
      end;
end;
```

```
{-------------------------------------------------------------}
{                      main                                   }
{-------------------------------------------------------------}
begin {main}
        gdriver := Detect;

        { initialize graphics and local variables}
        InitGraph(gdriver,gmode,'C:\TP\DRIVERS');
        Getviewsettings(viewport);

        { read result of initialization }
        errorcode := graphresult;

        { check an error}
        if(errorcode <> grOk) then
                    { an error occurred}
        begin
                    writeln('Graphics error :  .grapherrormsg(errorcode));
                    writeln('Please press any key to halt ......');
                    readln;
                    exit;
        end
        else
        begin
             { just for check}
             outtextxy(120,80,'The image picture on X=v plane');
             { initialized}
             v :=0.; i := 1;
             { generate points on the surface and store in a[i,j]}
             while v <= 5. do
             begin
                  thelta := 0.0;
                  j := 1;
                  while thelta <= 180 do
                  begin
                       a[i,j].x := f(v) * cos(thelta);
                       a[i,j].y := f(v) * sin(thelta);
                       a[i,j].z := g(v);
                       j := j + 1;
                       thelta := thelta + 180/10.;
                  end;
                  i := i + 1;
                  v := v + 5./10;
             end;
             e := 130.;v := 15.;
             {save the projection in p[i,j]}
             i1 := 1;
             while i1 <=  i do
             begin
                 j1 := 1;
                 while j1 <= j do
                 begin
                      t1 := a[i1,j1].y*((e-v)/(e-a[i1,j1].x));
                      t2 := a[i1,j1].z*((e-v)/(e-a[i1,j1].x));
                      if(t1 <=30) or (t2 <= 20) then
                      begin
                           p[i1,j1].y := a[i1,j1].y*((e-v)/(e-a[i1,j1].x));
                           p[i1,j1].z := a[i1,j1].z*((e-v)/(e-a[i1,j1].x));
                           rotate(p[i1,j1],r[i1,j1]);
                           shear(p[i1,j1],s[i1,j1]);
                           scale(p[i1,j1],q[i1,j1]);
                           j1 := j1+1;
                      end;
```

```
    end;
    i1 := i1+1;
end;
{ find the normal}
for i1 := 1 to i-3 do
    for j1 := 1 to j-3 do
    begin
        p1.x := a[i1,j1].x-a[i1,j1+1].x;
        p1.y := a[i1,j1].y-a[i1,j1+1].y;
        p1.z := a[i1,j1].z-a[i1,j1+1].z;
        p2.x := a[i1+1,j1+1].x-a[i1,j1+1].x;
        p2.y := a[i1+1,j1+1].y-a[i1,j1+1].y;
        p2.z := a[i1+1,j1+1].z-a[i1,j1+1].z;
        cross(p1,p2,temp);
        n := temp;
        { find the angle between n and in+pt-a[i,j+1]}
        in_pt.x := 0.;in_pt.y := 0.;in_pt.z := g(v);
        p1 := n;
        p2.x := in_pt.x - a[i1,j1+1].x;
        p2.y := in_pt.y - a[i1,j1+1].y;
        p2.z := in_pt.z - a[i1,j1+1].z;
        thelta := dot(p1,p2);

        {check whether n point out the surface :}
        if thelta > 0. then
                    begin
                         n.x := -n.x;
                         n.y := -n.y;
                         n.z := -n.z;
                    end;
        p1 := n;
        p2.x := e - a[i1,j1+1].x;
        p2.y := - a[i1,j1+1].y;
        p2.z := - a[i1,j1+1].z;
        thelta := dot(p1,p2);
        if thelta > 0.  then
        begin
              rotate(p[i1,j1],r[i1,j1]);
              shear(p[i1,j1],s[i1,j1]);
              scale(p[i1,j1],q[i1,j1]);
              rotate(p[i1+1,j1],r[i1+1,j1]);
              shear(p[i1+1,j1],s[i1+1,j1]);
              scale(p[i1+1,j1+1],q[i1+1,j1+1]);
              rotate(p[i1+1,j1+1],r[i1+1,j1+1]);
              shear(p[i1+1,j1+1],s[i1+1,j1+1]);
              scale(p[i1+1,j1+1],q[i1+1,j1+1]);
              rotate(p[i1,j1+1],r[i1,j1+1]);
              shear(p[i1,j1+1],s[i1,j1+1]);
              scale(p[i1,j1+1],q[i1,j1+1]);
              draw_pic(p[i1,j1],p[i1+1,j1],p[i1+1,j1+1],p[i1,j1+1])

        end;
```

```
                end;
            readln;
            clearviewport;
            outtextxy(120,80,'The image picture of rotating on X=v plane');
            outtextxy(120,100,'matrix: |   cos(c)   sin(c)   |');
            outtextxy(120,110,'        |  -sin(c)   cos(c)   |');
            for i1 := 1 to i-3 do
                for j1 := 1 to j-3 do
                    draw_pic(r[i1,j1],r[i1+1,j1],r[i1+1,j1+1],r[i1,j1+1]);
            readln;
            clearviewport;
            outtextxy(220,80,'The image picture of shearing on X=v plane');
            outtextxy(220,100,'matrix: |   1     0.5   |');
            outtextxy(220,110,'        |  -0.1   1     |');
            for i1 := 1 to i-3 do
                for j1 := 1 to j-3 do
                    draw_pic(s[i1,j1],s[i1+1,j1],s[i1+1,j1+1],s[i1,j1+1]);
            readln;
            clearviewport;
            outtextxy(120,80,'The image picture of scaling on X=v plane');
            outtextxy(120,100,'matrix: |   0.7    0     |');
            outtextxy(120,110,'        |   0      0.7   |');
            for i1 := 1 to i-3 do
                for j1 := 1 to j-3 do
                    draw_pic(q[i1,j1],q[i1+1,j1],q[i1+1,j1+1],q[i1,j1+1]);
            readln;
            { clean up    }
            CloseGraph;
        end;
end.
```