Introduction

Graphito is a graphics programming system and tool kit for the generation and manipulation of images. It can be used:

1) as a picture generation language in its own right, enabling even novice programmers to use powerful tools to produce interesting results.
2) as a unique, visual and exciting way to learn structured programming
3) as a graphics tutor enabling a user to build up expertise in programming.
4) as a graphics package to design advertising material simple CAD facilities, educational software etc.

Graphito contains two types of material

1) pre-drawn motifs and alphabets
2) BASIC procedures that act upon these motifs and alphabets.

The material is grouped together in six modules, each of which produces images of a certain type (two-dimensional, colour, etc.)

The method for producing a *Graphito* image is as follows:

1) Select the module that contains the material you require.
2) Load this module from disc or tape.
3) Write a BASIC program making use of the material in the module. At its simplest this program will be a list of procedure calls, but any normal BASIC features (e.g. FOR ... NEXT loops, SOUND statements) can also be used.
4) Run the program.

Graphito resources

Modules

Graphito contains a large number of resources. Because of the limited memory available on the BBC Micro and Electron, they have been divided into six modules. There are four modules for producing two dimensional images, and two for three-dimensional images. Some of the resources appear in more than one module.

2DMOD1	A two-dimensional module that allows images to be generated that connsist of (transformed) text and motifs, and uses interaction.
2DMOD2	A limited version of 2DMOD1 that makes some of the two-dimensional resources available in MODES 0, 1 and 2. This allows interactive painting and colour mixing.
2DMOD3	A version of 2DMOD1 that concentrates on the generation of network or wallpaper groups using both the supplied resources and icons that are interactively designed.
2DMOD4	A two-dimensional module that includes mathematical and recursive patterns as resources as well as the resources of 2DMOD1. It also contains some of the network generations of 2DMOD3.
3DMOD1	A thee-dimensional module that allows the two-dimensional resources to be thickened into three-dimensional objects and manipulated in three-dimensional space. It also allows the user to decorate three-dimensional objects such as cubes and spheres with motifs and text.

3DMOD2 A three-dimensional module that allows users to set up their own three-dimensional objects and view these from any viewpoint Partial hidden surface removal is incorporated in this module

Each module starts with a program prelude which optimizes the memory resources. Details of how you can alter the preludes for particular purposes are given later on in the manual.

Motifs and alphabets

There are 34 predrawn motifs and six alphabets that can be used in the two-dimensional modules or expanded into three dimensions and used in 3DMOD1. In 3DMOD2 there are four predefined three-dimensional objects available. The motifs and alphabets are obtained by loading procedures in a module. See the descriptions of PROCIoad, PROCloadalpha, PROCalphaslice, PROCload3Ddata, and PROCmotif3D towards the end of this manual for a fuller treatment of the available material.

Procedures

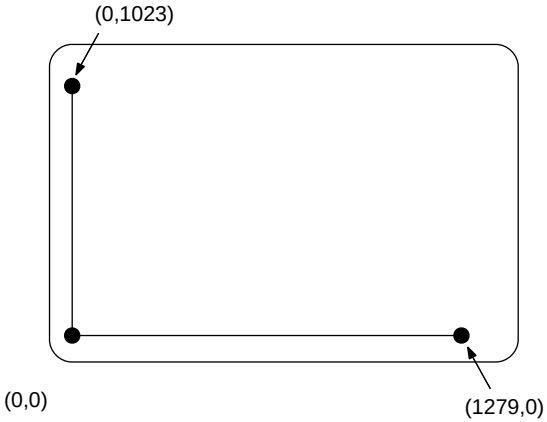
The motifs and alphabets are handled and transformed by procedures. There are over sixty of these. All the procedures are listed in alphabetical order later in the manual with full instructions on how to use them.

Using Graphito

This manual has been arranged to give tutorial help to first-time users and to act as a comprehensive reference guide once experience has been gained

Each module is described and instructions on how to use it are given. These instructions include a list of the *Graphito* procedures available in the particular module. For a full description of each procedure and its effect consult the alphabetical section later in the manual. Many examples of *Graphito* programs and the results they produce are given. Try entering some of these for yourself, and seeing how they work. This is probably the best way to become familiar with *Graphito*.

Screen coordinates



Many procedures in the two-dimensional modules will require coordinates of positions. The screen origin (0,0) is at the bottom left hand corner of the screen. The Screen coordinate system is 0 to 1279 units in the x direction and 0 to 1023 in the y direction. Point (640,512) is, therefore, approximately at the centre otthe screen.

For the three-dimensional modules the situation is different, and slightly more Complicated. There are no absolute coordinates: the user must define a position ffOm Which to view the three-dimensional scene. A full description of how this works IS given in PROCviewpoint on page 105.

Graphito on tape

The material in Graphito is stored on tape in files arranged as follows:

Side A	*Side B*		
2DMOD1	BEETLE	ORCHID	BRITAIN
2DMOD2	BUS	PARROTS	IRELAND
INTERAC	BUTFLY	RABBIT	EUROPE
2DMOD3	CANNON	RHINO	EUROPEF
2DMOD4	CITRUS	ROSE	AFRICA
3DMOD1	EAGLE	TELEPH	AUSTRA
3DMOD2	ELEPHA	THISTLE	NAMERI
EPSONPR	FISH	TRANSIT	SAMERI
MODE5BL	FLOWER	TRICOP	ALPHA1
MODE1BL	GULL	TURTLE	ALPHA2
MODE2BL	GUN	UMBREL	ALPHA3
	HELMET	VASE	ALPHA4
	MUSHRO	VEGTAB	

You will probably find the tape easier to use if, for each side in turn, you catalogue where the files start. Rewind the tape fully, set your tape counter to zero, type ***CAT**, press **RETURN**, then play the tape. As the name of each file appears on the screen note down, against the name of the file in the list above, the tape counter reading. You can then use this list to help you locate files later on.

You can also save the files you use in a particular program onto another tape. To save a file, e.g. BEETLE, the procedure is as follows. Find the beginning of the file on the *Graphito* tape, type ***OPT 1,2** and press **RETURN**, and then type ***LOAD BEETLE** and press **RETURN**. Now play the tape and the file will load onto your Computer- While the file is being loaded onto the computer the file name will appear on the screen and the number of blocks in the file will be counted off. When the file is fully loaded the filename will have four numbers (in hex) after it like this

	BEETLE	08	03C7	00006518	00006518
These are	Filename	no. of blocks	size of file	start address	execution address

Now put the tape onto which you want to save BEETLE into your recorder and type * SAVE BEETLE start address +size, i.e. ***SAVE BEETLE 651B +3C7**, and press **RETURN** and the record button on your tape recorder. The file BEETLE will now be stored on the tape. Note that you can ignore any zeroes at the start of the hex numbers.

Graphito on disc

The Graphito disc is 40 track, single sided, single density.

Error messages

Most of the error messages that arise in Graphito will be 'normal' BASIC errors because *Graphito* programs are written in BASIC.

The BASIC error that is most likely to occur is

No such FN/PROC

This will appear if a Graphito name, be it a motif, alphabet or procedure, has been misspelt, or uppercase letters have been used instead of lower.

Another error message which may appear is

Not enough HIMEM allocated

This means that a program prelude has been altered incorrectly, or not included.

The message

No such variable

referring to a line within the procedure library will appear if an attempt has been made to draw a motif or alphabet without previously loading it.

Most logical errors in graphics programming Will be visually apparent The following list may help you to diagnose the more puzzling faults.

Symptom	*Possible cause*
Blank screen	Plotting is taking place off the screen because wrong values have been entered in a procedure OR No drawing procedure has been called
Intermittent straight lines appear	A very large value of 'scale' is being used or has accumulated
Small dot appears - nothing else happens	A very small value of 'scale' is being used or has accumulated
Changes of motif size or position are taking place exponentially	Failure to use PROCinitialise

2DMOD1

With this module a wide variety of procedures can be used, either singly or in combination to transform two-dimensional motifs and alphabets.

The following resources are available

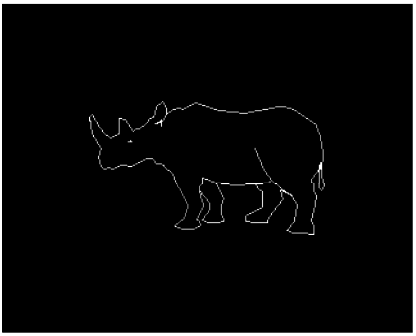
34 motifs
6 alphabets
26 procedures

It is also possible to add to the designs interactively from the keyboard, or by using a joystick.

To use the module

1) Type **LOAD"2DMOD1"** and press **RETURN**. This loads the program prelude and the Graphito resources.
2) Write a BASIC program from line 10 onwards using the Graphito resources (the *Graphito* resources start at line 1000). The simplest program loads a motif and then displays it.

```
10 PROCload("RHINO")
20 PROCdrawandscale(640,512,1)
30 END
```



3) Type RUN and press RETURN.

Program prelude

The program prelude that is loaded with the module is:

```
1 MODE 4:HIMEM=HIMEM-1260-2340-570
2 PROCinitialisememory(4,TRUE,TRUE,TRUE)
```

This can be altered to make best use of memory in the following circumstances:

```
1 MODE 4:HIMEM=HIMEM-2340-570
2 PROCinitialisememory(4,FALSE,TRUE,TRUE)
```

if you are not using motifs

```
1 MODE 4:HIMEM=HIMEM-1260-570
2 PROCinitialisememory(4,TRUE,FALSE,TRUE)
```

if you are not using alphabets

```
1 MODE 4:HIMEM=HIMEM-1260-2340
2 PROCinitialisememory(4,TRUE,TRUE,FALSE)
```
if you are not using interaction

Procedures

The following procedures are available:

PROCalphaslice	PROCinteract
PROCboxwindowin	PROCload
PROCboxwindowout	PROCloadalpha
PROCcircIewindowin	PROC1oadscreen
PROCcirclewindowout	PROCreflectx
PROCdrawanddeflate	PROCref1ecty
PROCdraw and inflate	PROCrestore_fore_col
PROCdrawandreflect	PROCrotate
PROCdrawandscale	PROCsavescreen
PROCfore_to_back_col	PROCscale
PROChshear	PROCstretch
PROChtext	PROCvshear
PROCinitialise	PROCvtext

Details of how to use each of these procedures are given later in the manual.

Using 2DMOD1

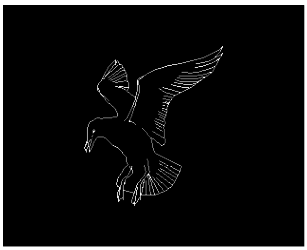
Two-dimensional linear transforms operate on the current motif in memory. The transforms alter the attributes of the motif but do not draw it. Motifs are a list of coordinates with respect to the approximate centre of gravity of the motif. That is, point (0,0) in a list of motif coordinates is near the centre of the motif. Two dimensional transforms and operations on motifs work with respect to the motif origin or centre. Any calls to these transforms must be succeeded by a call to PROCdrawandscale if the motif is to be plotted.

The transforms can be used in a combination sequence (see below). The alterations to the attributes simply accumulate in memory

The examples here show the kind of results that can be obtained by using the procedures in programs with other BASIC features. For ease of comparison they all use the same motif.

The position and scale of motifs can be controlled in a FOR loop.

```
10 PROCload("GULL")
20 PROCdrawandscale(640,512,1)
```

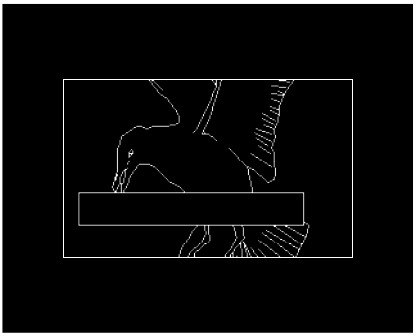


The position and scale of motifs can be controlled in a FOR loop.

```
10 PROCload("GULL")
20 scale=1:xo=250:yp=350
30 FOR i=1 TO 4
40   PROCdrawandscale(xo,yo,scale)
50   xo=xo+300:yo=yo+50*i:scale=scale*0.7
60 NEXT i
```

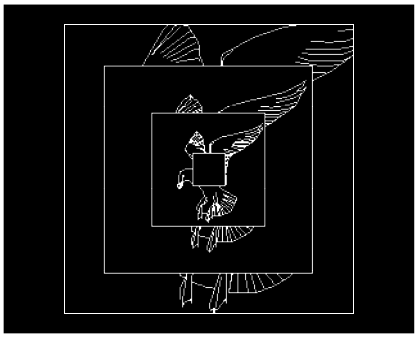
PROCboxwindowin and PROCboxwindowout can be combined to produce a wide range of effects.

```
10 PROCload("GULL")
20 PROCdrawandscale(640,512,1.5)
30 PROCboxwindow(200,250,900,550)
40 PROCboxwindowout(250,350,700,100)
```



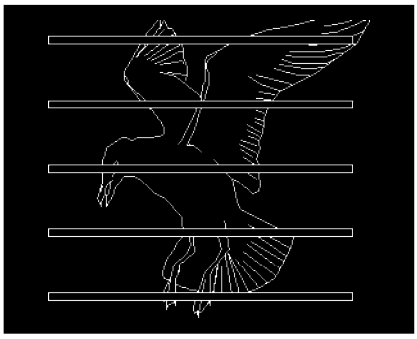
One call of PROCboxwindowin followed by three calls of PROCboxwindowout in a FOR loop. The window dimensions and scale of the motif are reduced after each call. An IF statement ensures that the centre block is blank.

```
   10 PROCload("GULL")
   20 PROCdrawandscale(640,512,1.5)
   30 PROCboxwindowin(200,100,900,900)
   40 boxshrink = 75
   50 scale = 1.4:xs = 250:ys = 150:length = 800:height =
800
   60 FOR i=1 TO 3
   70   xs = xs+i*boxshrink:ys = ys+i*boxshrink
   80   length = length-i*boxshrink*2
   90   height = height-i*boxshrink*2:scale = scale * .57
  100   PROCboxwindowout(xs,ys,length,height)
  110   IF i <> 3 THEN PROCdrawandscale(640,512,scale)
  120NEXT i
```



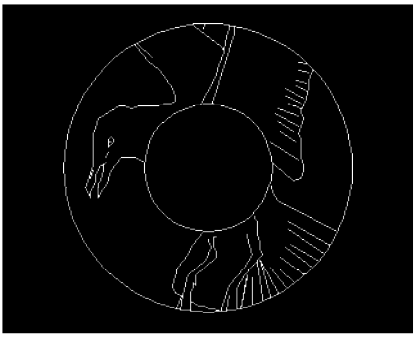
Using PROCboxwindow in different vertical positions creates a series of blank stripes.

```
   10 PROCload("GULL")
   20 PROCdrawandscale(640,512,1.5)
   30 xs = 200:ys = 100
   40 FOR i=1 TO 5
   50   PROCboxwindowout(xs,ys,950,25)
   60   ys = ys + 200
   70   NEXT i
```



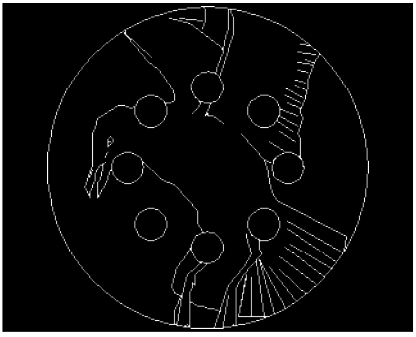
PROCcirclewindowin and PROCcirclewindowout can be similarly combined. One call of each results in an annulus.

```
10 PROCload("GULL")
20 PROCdrawandscale(640,512,2)
30 PROCcirclewindowin(640,512,450)
40 PROCcirclewindowout(640,512,200)
```

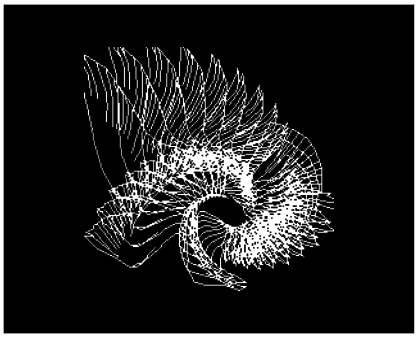


Multiple calls of PROCcirclewindowout can be made so that the centres of the circles themselves form a circle.

```
10 PROCload("GULL")
20 PROCdrawandscale(640,512,2)
30 PROCcirclewindowin(640,512,500)
40 FOR theta=0 TO 360 STEP 45
50 xs=250*COS(RAD(theta)):ys=250*SIN(RAD(theta))
60 PROCcirclewindowout(xs+640,ys+512,50)
70 NEXT theta
```



In the next example PROCrotate is applied nine times, with a scale increase each time. The result is an attractive pattern even though it is not easy to see what the original motif is.



```
10 PROCload("GULL")
20 scale=.6
30 FOR i=1 TO 9
40   PROCrotate(10)
50   PROCdrawandscale(640,512,scale)
60   scale=scale*1.1
70 NEXT i
```

Five applications of PROCrotate together With a scale increase and alteration of the start coordinates give a more pictorial result

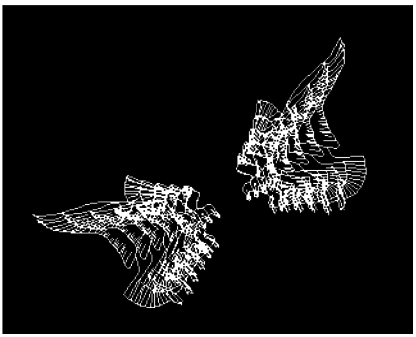
```
10 PROCload("GULL")
20 scale=.4:xs=1100:ys=800
30 FOR i=1 TO 5
40   PROCrotate(10)
50   PROCdrawandscale(xs,ys,scale)
60   scale=scale*1.2
70   xs=xs-220:ys=ys-80
80 NEXT i
```

Similarly, four applications of PROCstretch in directions given by theta = 0,20,40 and 60 produce a 'recognisable' picture. A point to note here is that PROCinitialise is used so that each stretch is applied to the original motif.

```
10 PROCload("GULL")
20 xs=100:dist=30:i=0
30 FOR theta=0 TO 60 STEP 20
40   i=i+1
50   PROCinitialise
60   PROCstretch(0.3,theta)
70   PROCdrawandscale(xs,400,1)
80   xs=xs+250+i*dist
90 NEXT theta
```

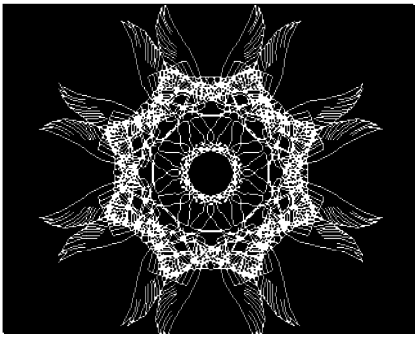
PROCinitialise is also needed for this pattern, which is produced by five applications of PROCdrawandreflect. What would happen if it was omitted?

```
10 PROCload("GULL")
20 dist=280:scale=.7
30 FOR i= 1 TO 5
40   PROCdrawandreflect(670,500,scale,dist,30)
50   dist=dist-40:scale=scale*.85
60   PROCinitialise
70 NEXT i
```



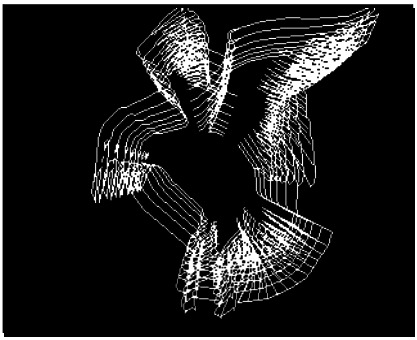
Eight applications of PROCdrawandreflect will produce this pattern. Again PROCinitialise is important - it ensures that theta is always applied to a non-rotated motif.

```
10 PROCload("GULL")
20 FOR theta=0 TO 315 STEP 45
30   PROCdrawandreflect(640,500,.7,200,theta)
40   PROCinitialise
50 NEXT theta
```



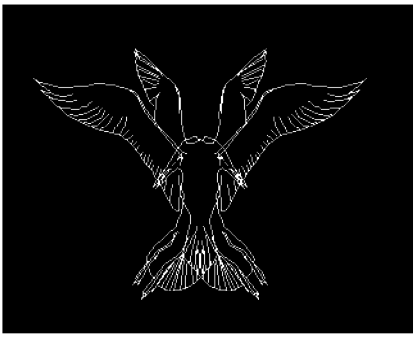
PROCdrawandinflate has some interesting subtleties. Here it is called seven times with decreasing expansion and constant scale. How would it look if the expansion was constant but the scale decreased?

```
10 PROCload("GULL")
20 expansion=2
30 FOR i=1 TO 7
40   PROCdrawandinflate(640,512,0,0,expansion,1)
50   expansion=expansion*.8
60 NEXT i
```



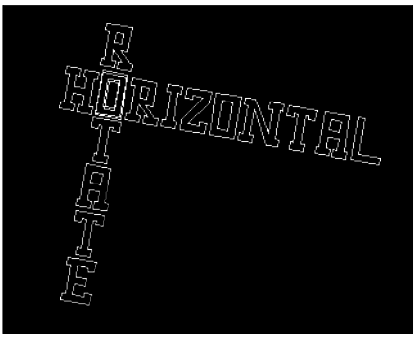
This result was obtained by combining three procedures: PROCstretch, PROCreflecty and PROCrotate, but don't forget that without PROCdrawandsca]e you won't see anything!

```
10 PROCload("GULL")
20 PROCstretch(2,90)
30 PROCreflecty
40 PROCrotate(30)
50 PROCdrawandscale(600,350,.7)
60 PROCreflecty
70 PROCdrawandscale(60,350,.7)
```



Text can also be transformed by the procedures in 2DMOD1. Note that the procedures operate on the entire word and on all subsequent calls of PROCvtext and PROChtext.

```
10 PROCloadalpha("ALPHA1")
20 PROCrotate(-10)
30 PROCvtext("ROTATE",200,950,.7,20)
40 PROChtext("HORIZONTAL",90,620,.7,20)
```

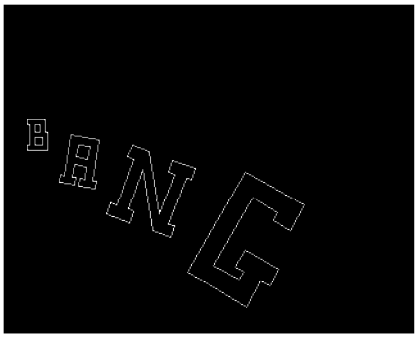


Note the call of PROCinitialise before printing "AND".

```
10 PROCloadalpha("ALPHA1")
20 PROChshear(.5)
30 PROChtext("XREFLECT",10,700,1,10)
40 PROCreflectx
50 PROChtext("XREFLECT",10,650,1,10)
60 PROCinitialise
70 PROChtext("AND",550,300,.3,20)
80 PROChshear(1)
90 PROChtext("SHEAR",200,100,.6,50)
```

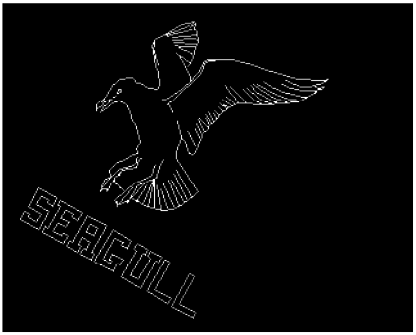
Individual transformations can be applied to single letters

```
10 PROCloadalpha("ALPHA1")
20 PROChtext("B",100,500,.5,10)
30 PROCrotate(-10)
40 PROChtext("A",200,400,.8,10)
50 PROCrotate(-10)
60 PROChtext("N",350,300,1.2,10)
70 PROCrotate(-10)
80 PROChtext("G",600,120,1.9,10)
```



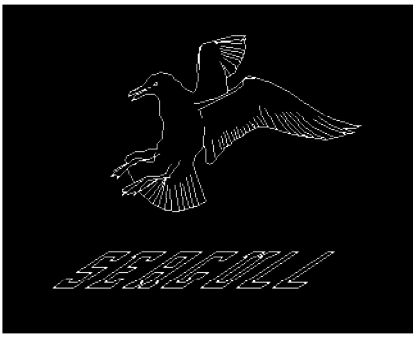
Text and motifs can be transformed together - in this case by the same call of PROCrotate.

```
10 PROCloadalpha("ALPHA1")
20 PROCload("GULL")
30 PROCrotate(-30)
40 PROCdrawandscale(500,600,1)
50 PROChtext("SEAGULL",80,310,.6,20)
```



Now look at this example where the text and motifs are transformed differently. An Initialisation, which in this case is forced by PROCloadalpha, must be applied after the motif has been completed.

```
10 PROCload("GULL")
20 PROCrotate(-45)
30 PROCdrawandscale(500,600,1)
40 PROCloadalpha("ALPHA1")
45 PROCinitialise
50 PROChshear(1.5)
60 PROChtext("SEAGULL",100,100,.6,60)
```



2DMOD2

The module is a reduced version of 2DMOD1. It permits the use of motifs and interaction but not alphabets. It allows the use of some of the Graphito resources in a high resolution mode (MODE 0) or in a mode that has more colours available (MODE 2). It can also be run in medium resolution (MODE 1) with four colours. You will find that the colours and textures available in MODE 5 are also available in MODE 1, but at a higher resolution.

The following resources are available:

34 motifs
12 procedures

To use the module:

1) Type **LOAD"2DMOD2"** and press **RETURN**. This loads the program prelude and the *Graphito* resources.
2) Write a BASIC program from line 10 onwards using the *Graphito* resources (the *Graphito* resources start at line 1 000). The simplest program loads a motif, displays it, and goes into interactive mode.
3) Type **RUN** and press **RETURN**.

Program prelude

The program prelude that is loaded with the module is:

```
1 MODE 2:HIMEM=HIMEM-1260-570
2 PROCinitialisememory(2,TRUE,TRUE)
```

This can be altered to make best use of the computer's memory in the following circumstances:

```
1 MODE 2:HIMEM=HIMEM-570
2 PROCinitialisememory(2,FALSE,TRUE)
```

if you are not using motifs

```
1 MODE 2:HIMEM=HIMEM-1260
2 PROCinitialisememory(2,TRUE,FALSE)
```

if you are not using interaction

In addition, the mode number in line 1 and the first parameter in line 2 (which are both 2 in the version loaded from the disc or tape) can be changed to any other mode number.

Procedures

The following procedures are available

PROCdrawandscale
PROChshear
PROCinitialise
PROCinteract
PROCload
PROCreflectx
PROCreflecty
PROCrotate
PROCscale
PROCstretch
PROCvshear

Details of how to use each of these procedures are given later in the manual.

2DMOD3

This module enables motifs or 'clusters' of motifs to be displayed in networks or 'wallpaper' groups.

34 motifs
6 alphabets
21 procedures

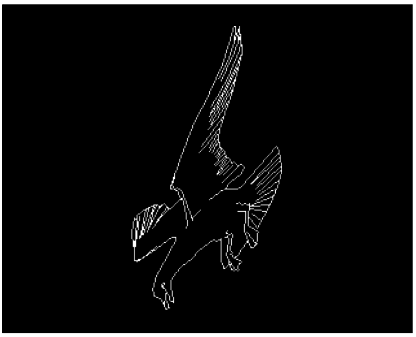
The module contains a procedure allowing users to design motifs of their own.

To use the module

1) Type **LOAD"2DMOD3"** and press **RETURN**. This loads the program prelude and the Graphito resources.
2) Write a BASIC program from line 10 onwards that includes a call of PROCnet. This arranges motifs, or clusters of motifs, in a network pattern. Define a procedure PROCnetmotif from line 1000 onwards (the Graphito resources start at line 2000). This defines the motifs, or clusters, to be arranged by PROCnet.

For example

```
1000 DEF PROCnetmotif(x,y,scale)
1010 PROCinitialise
1020 PROCload("GULL")
1030 PROCstretch(2.5,0)
1040 PROCrotate(60)
1050 PROCdrawandscale(x,y,scale)
1060 ENDPROC
```



defines the motif and

```
10 PROCnet(5,3,250,350,100,100,30,.26)
20 END
```

draws it on a net.

3) Type **RUN** and press **RETURN**

Program prelude

The prelude that is loaded with the module is:

```
1 MODE4 : HIMEM=HIMEM-1260-2340
2 PROCinitialisememory(4,TRUE,TRUE)
```

This can be altered to make best use of the computer's memory in the following circumstances:

```
1 MODE4 : HIMEM=HIMEM-2340
2 PROCinitialisememory(4,FALSE,TRUE)
```

if you are not using motifs

```
1 MODE4 : HIMEM=HIMEM-1260
2 PROCinitialisememory(4,TRUE,FALSE)
```

if you are not using alphabets

Procedures

The following procedures are available

PROCalphaslice	PROCnet
PROCcharnet	PROCreflectx
PROCchardesign	PROCreflecty
PROCdrawandscale	PROCrestore_fore_col
PROCfore_to_back_col	PROCrotate
PROChshear	PROCsavescreen
PROChtext	PROCscale
PROCinitialise	PROCstretch
PROCload	PROCvshear
PROCloadalpha	PROCvtext
PROCloadscreen	

Details of how to use each of these procedures are given later in the manual

Using 2DMOD3

There are exactly 17 ways in which an asymmetric motif can be arranged to form a two dimensional network pattern. These are known as wallpaper groups.

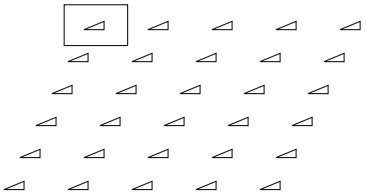
Each can be generated in this system by generating a motif cluster using DEFPROCnetmotif together with an appropriate call to PROCnet. The groups are summarized below using an asymmetric triangle as a motif. Each group is given together with a suggested recipe for PROCnetmotif. In each case adjustments in the x and y parameters of PROCdrawandscale may be necessary to gain the correct symmetry. Note that this method of using elementary motifs to make a motif cluster is not the only to way to proceed. With group 1 7, for example, you could use the generation scheme for group 1 provided you started with a motif that possessed the appropriate rotational symmetry.

The motif 'cluster' generated by PROCnetmotif is boxed in each illustration.

Group 1

This is the basic network group and simply requires a motif to be placed at each point on the net. The motif definition should be;

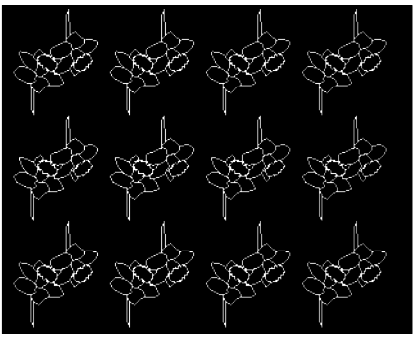
```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 ENDPROC
```

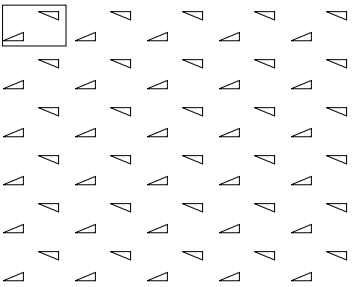


Group 2

This group involves a reflection about the x axis together with an appropriate displacement.

```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCreflectx
1050 PROCdrawandscale(......)
1060 ENDPROC
```

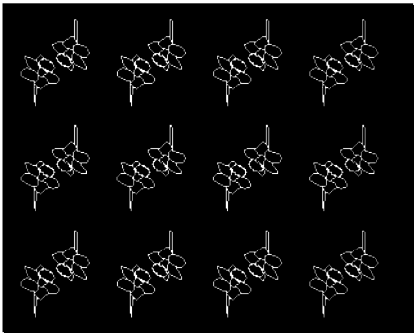


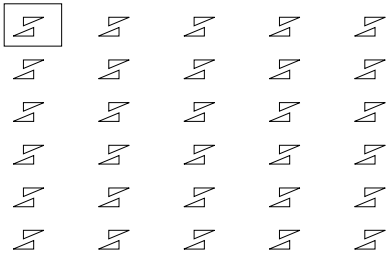


Group 3

The motif cluster in the group is formed by a 180° rotation.

```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCrotate(180)
1050 PROCdrawandscale(......)
1060 ENDPROC
```

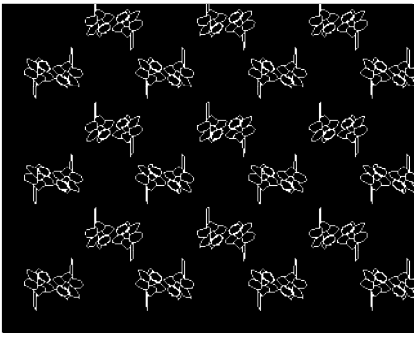


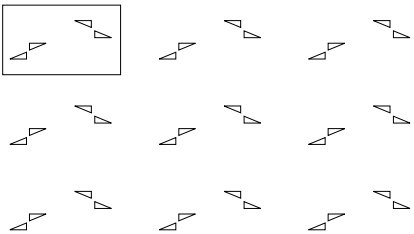


Group 4

This motif cluster is formed from four motifs

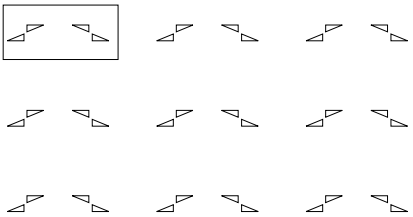
```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCrotate(180)
1050 PROCdrawandscale(......)
1060 PROCinitialise
1070 PROCreflecty
1080 PROCdrawandscale(......)
1090 PROCrotate(-180)
1100 PROCdrawandscale(......)
1110 ENDPROC
```





Group 5

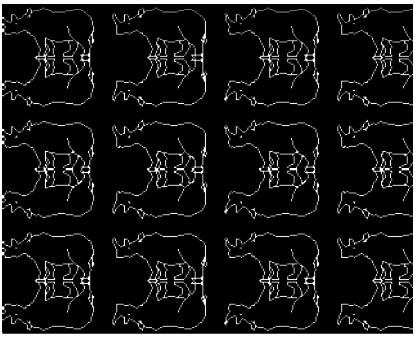
This group is formed using the same PROCnetmotif definition as group 4. The y displacements in PROCdrawandscale need adjusting.

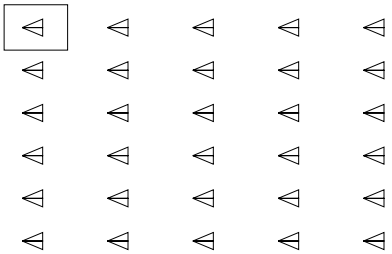


Group 6

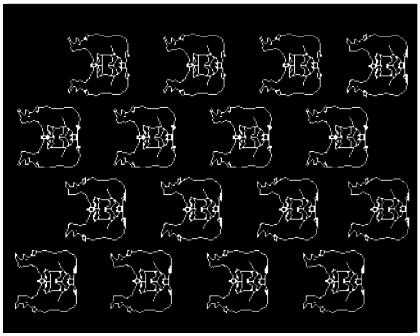
The cluster in this group is formed by x reflection. The cluster is then placed on a rectangular net.

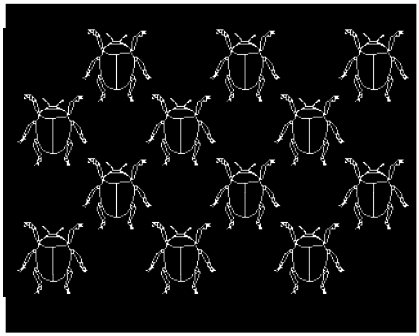
```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCreflectx
1050 PROCdrawandscale(......)
1060 ENDPROC
```





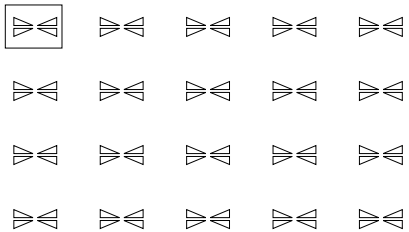
Group 7
This is identical to group 6 except that the clusters are placed in a net with 45° skew. Note that single motifs exhibiting the appropriate symmetry can be used instead of motif clusters. Because the beetle already possesses x (or y) reflection symmetry it can be used on a group 1 net but still possess the properties of group 7.



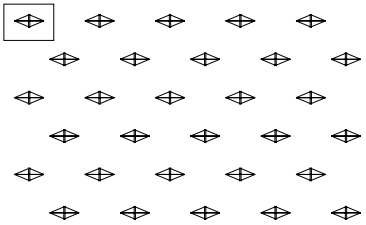


Group 8
This group involves a cluster of four asymmetric motifs formed from x and y reflection. Alternatively it can be formed from x reflection of a motif that already possesses y reflection (or vice versa).

```
1000 DEFPROCnetmotif(x,y,scale)
1010   PROCload("MOTIF")
1020   PROCinitialise
1030   PROCdrawandscale(......)
1040   PROCreflecty
1050   PROCdrawandscale(......)
1060   PROCreflectx
1070   PROCdrawandscale(......)
1080   PROCreflecty
1090   PROCdrawandscale(......)
1100 ENDPROC
```

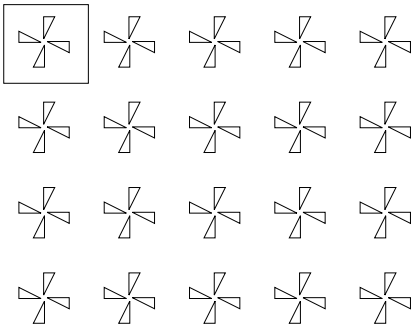


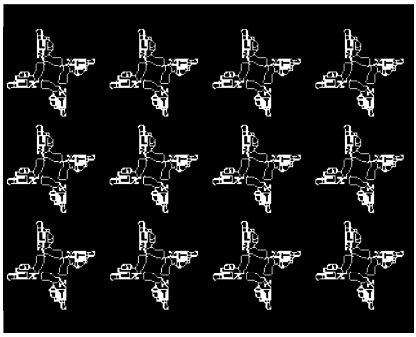
Group 9
This group is identical to 8 except that the clusters are placed on a net with 45° skew.



Group 10
A cluster of four formed by three 90° rotations. The clusters are placed on a rectangular net.

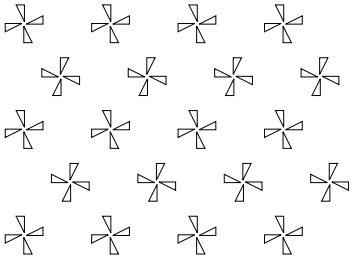
```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCrotate(90)
1050 PROCdrawandscale(......)
1060 PROCrotate(90)
1070 PROCdrawandscale(......)
1080 PROCrotate(90)
1090 PROCdrawandscale(......)
1100 ENDPROC
```





Group 11

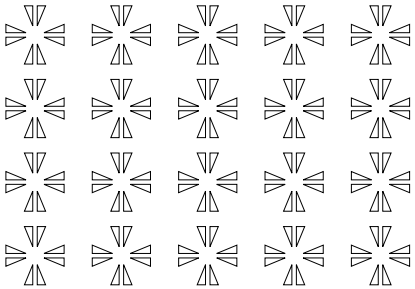
As group 10 except that the clusters are placed on a net with 45° skew



Group 12

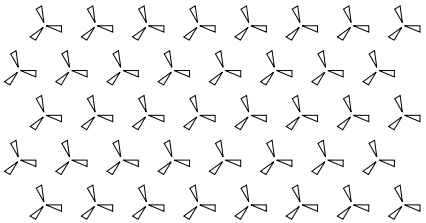
Each cluster consists of eight asymmetric motifs grouped into two sub-clusters as for group 8. These are rotated through 90°.

```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCreflectx
1050 PROCdrawandscale(......)
1060 PROCrotate(90)
1070 PROCdrawandscale(......)
1080 PROCreflecty
1090 PROCdrawandscale(......)
1100 PROCrotate(90)
1110 PROCdrawandscale(......)
1120 PROCreflectx
1130 PROCdrawandscale(......)
1140 PROCrotate(90)
1150 PROCdrawandscale(......)
1160 PROCreflecty
1170 PROCdrawandscale(......)
1180 ENDPROC
```



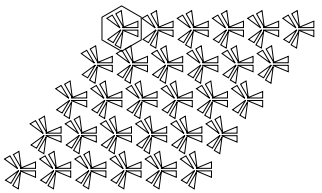
Group 13
A cluster formed from 120° rotation.

```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCrotate(120)
1050 PROCdrawandscale(......)
1060 PROCrotate(120)
1070 PROCdrawandscale(......)
1080 ENDPROC
```



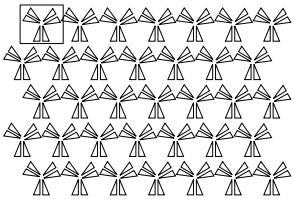
Group 14
Each cluster is formed from six asymmetric motifs and 120° rotation.

```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCrotate(120)
1050 PROCdrawandscale(......)
1060 PROCrotate(120)
1070 PROCdrawandscale(......)
1080 PROCinitialise
1090 PROCreflectx
1100 PROCdrawandscale(......)
1110 PROCrotate(120)
1120 PROCdrawandscale(......)
1130 PROCrotate(120)
1140 PROCdrawandscale(......)
1150 ENDPROC
```



Group 15

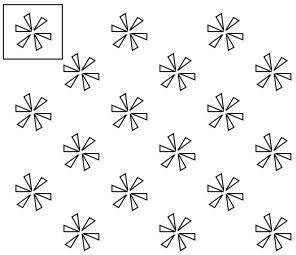
As tor group 14 but different parameters are required in PROCnet. The actual difference between 14 and 15 is quite subtle and requires a detailed study of network grouping that would be out of place in this manual.



Group 16

Each cluster is formed from six asymmetric motifs rotated through 60°

```
1000 DEFPROCnetmotif(x,y,scale)
1010 PROCload("MOTIF")
1020 PROCinitialise
1030 PROCdrawandscale(......)
1040 PROCrotate(60)
1050 PROCdrawandscale(......)
1060 PROCrotate(60)
1070 PROCdrawandscale(......)
1080 PROCrotate(60)
1090 PROCdrawandscale(......)
1100 PROCrotate(60)
1120 PROCdrawandscale(......)
1130 PROCrotate(60)
1140 PROCdrawandscale(......)
1150 ENDPROC
```

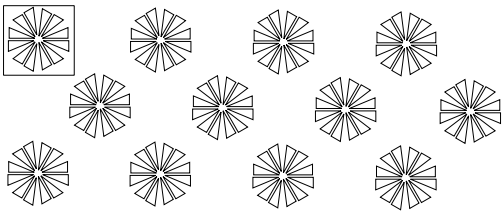


Group 17

Each cluster is formed by generating group 16, then calling:

PROCinitialise
PROCreflectx

and repeating the scheme for 16.



A horizontal band can easily be formed by calling PROCnet with the parameter 'noofrows' set to 1. Similarly for a vertical band 'noofcols' is set to 1.

```
10 PROCload("GUN")
20 PROCnet(6,1,220,0,150,700,0,.3)
```

```
  30 PROCnet(6,1,220,0,150,110,0,.3)
  35 PROCinitialise
  40 PROCloadalpha("ALPHA2")
  50 PROChshear(-.6)
  60 PROChtext("HAVE GUN",120,400,.6,30)
  70 PROCloadalpha("ALPHA1")
  80 PROCinitialise
  90 PROChtext("WILL TRAVEL",600,300,.3,35)

1000 DEF PROCnetmotif(x,y,scale)
1010 PROCinitialise
1020 PROCrotate(45)
1030 PROCdrawandscale(x,y,scale)
1040 ENDPROC
```

Two bands are drawn by calling PROCnet twice with 'noofrows' set to 1. PROCnetmotif contains a single rotation.

2DMOD4

This module enables the same network facilities as 2DMOD3. It does not however include interactive motif design. The main purpose of the module is to allow the generation of mathematical and recursive motifs and the formation of these into networks.

The following resources are available

34 motifs
6 alphabets
33 procedures

The mathematical motifs are mainly circular or harmonic functions. These give 'closed' patterns with varying degrees of symmetry.

Programming instructions, prelude etc. are identical to these in 2DMOD3 (except that the name is changed to 2DMOD4).

Procedures

The following procedures are available:

PROCalphaslice
PROCc_curve
PROCchamet
PROGcolouredflake
PROCdrawandscale
PPROCdragon
PROCellipse
PROCexspiral
PROCfore_to_back_col
PROChshear
PROChtext
PROCinitialise
PROCload
PROCloadalpha
PROCloadscreen
PROCmodcircle
PROCnet
PROCreflectx
PROCreflecty
PROCrestore_fore_col
PROCrotate
PROCsavescreen
PROCscale
PROCsierpinski
PROCsincircle
PROCsnowflake
PROCspiral
PROCsquares
PROCstretch
PROCtree
PROCvshear
PROCvtext
PROCw_curve

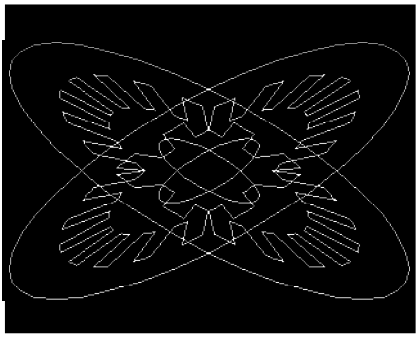
Details of how to use each of these procedures are given later in the manualUsing 2DMOD4

Using 2DMOD4

The mathematical and recursive patterns in this module give rise to a whole new range of interesting results. Any of the two-dimensional transforms can be applied to the mathematical motifs in exactly the same way as they are applied to the digitized pictorial motifs.

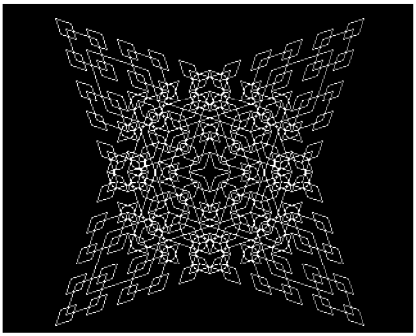
Applying the same transformations to two mathematical motifs.

```
10 PROChshear(1.2)
20 PROCmodcircle(640,512,300,100)
30 PROCellipse(640,512,100,100)
40 PROCellipse(640,512,400,400)
50 PROCreflecty
60 PROCmodcircle(640,512,300,100)
70 PROCellipse(640,512,100,100)
80 PROCellipse(640,512,400,400)
```



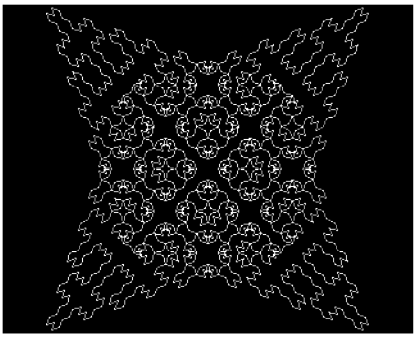
An interesting area that can be explored is 'interference' patterns, where two more simple patterns combine to produce a third. These can be produced with, for instance, recursive squares,

```
10 PROCstretch(2,-45)
20 PROCsquares(640,512,1,10)
30 PROCinitialise:PROCstretch(2,45)
40 PROCsquares(640,512,1,10)
```



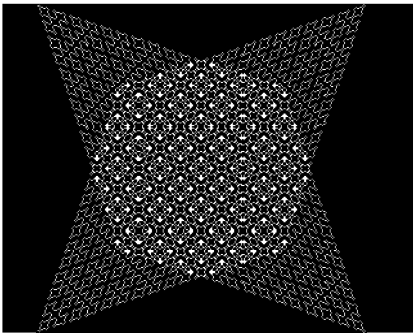
w-curves,

```
10 PROCstretch(2,45)
20 PROCw_curve(4,640,512,1)
30 PROCinitialise
40 PROCstretch(2,-45)
50 PROCw_curve(4,640,512,1)
```



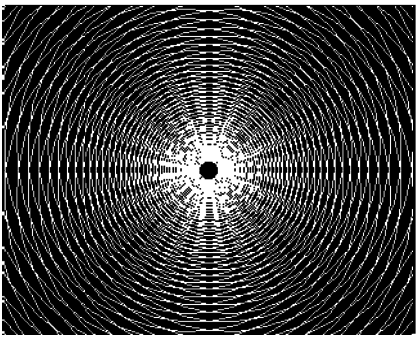
sierpinski curves

```
10 PROCstretch(2,45)
20 PROCsierpinski(5,640,512,1)
30 PROCinitialise
40 PROCstretch(2,-45)
50 PROCsierpinski(5,640,512,1)
```



and spirals.

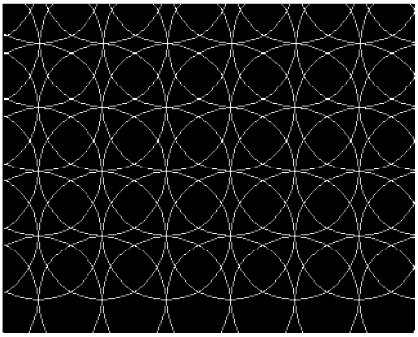
```
10 FOR theta=0 TO 330 STEP 30
20   PROCinitialise
30   PROCrotate(theta)
40   PROCexspiral(640,512,30)
50   PROCreflecty
60   PROCexspiral(640,512,30)
70 NEXT
```



Mathematical motifs can also be used in networks. Proceed as in 2DMOD3: PROCnet is called to generate the clusters and PROCnetmotif generates the motif.

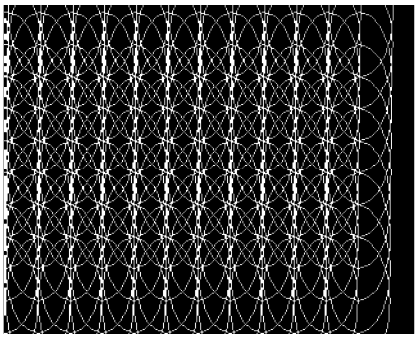
In this case an ellipse with axes of equal length is called, so the motif is a circle and the network is an example of Group 12.

```
  10 PROCnet(8,8,200,200,-100,100,1,1)
1000 DEF PROCnetmotif(x,y,scale)
1010 PROCinitialise
1020 PROCellipse(x,y,200,200)
1030 ENDPROC
```



If ellipses are used the network symmetry 'reduces' to that of Group 8.

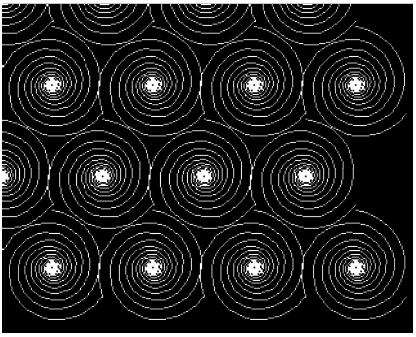
```
  10 PROCnet(13,10,100,100,-100,100,1,1)
1000 DEF PROCnetmotif(x,y,scale)
1010 PROCinitialise
1020 PROCellipse(x,y,100,200)
1030 ENDPROC
```



Here are some other mathematical networks

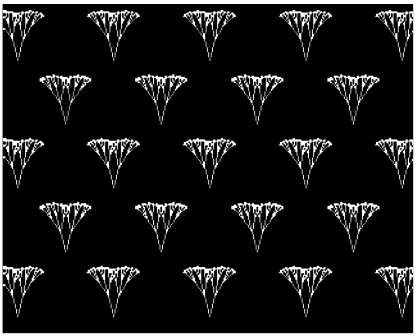
A network group (Group 13) formed from 'exspirals' . Each motif is formed from thre spirals rotated through 120° with respect to each other

```
  10 PROCnet(4,4,316,286,150,200,158,1)
1000 DEF PROCnetmotif(x,y,scale)
1010 FOR theta=90 TO 330 STEP 120
1020  PROCinitialise
1030 PROCrotate(theta)
1040 PROCexspiral(x,y,5)
1050 NEXT theta
1060 ENDPROC
```



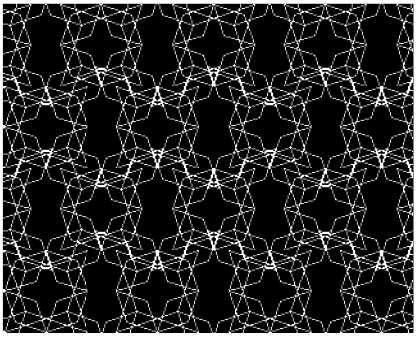
Network of trees.

```
  10 PROCnet(5,5,300,200,40,50,150,0.25)
1000 DEF PROCnetmotif(x,y,scale)
1010 PROCinitialise
1020 PROChshear(0.25)
1030 PROCtree(x,y,scale,FALSE,3,45,.7,4,1,1)
1040 PROCreflecty
1050 PROCtree(x,y,scale,FALSE,3,45,.7,4,1,1)
1060 ENDPROC
```



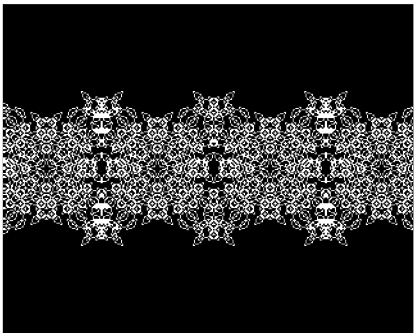
A recursive square network.

```
  10 PROCnet(5,4,350,256,128,128,175,.5)
1000 DEF PROCnetmotif(x,y,scale)
1010 PROCinitialise
1020 PROCstretch(2,-45)
1030 PROCsquares(x,y,scale,30)
1040 PROCinitialise
1050 PROCstretch(2,45)
1060 PROCsquares(x,y,scale,30)
1070 ENDPROC
```



A recursive square band pattern.

```
  10 PROCnet(5,1,350,350,128,512,175,.5)
1000 DEF PROCnetmotif(x,y,scale)
1010 PROCinitialise
1020 PROCstretch(2,-45)
1030 PROCsquares(x,y,scale,8)
1040 PROCinitialise
1050 PROCstretch(2,45)
1060 PROCsquares(x,y,scale,8)
1070 ENDPROC
```



3DMOD1

This module enables a variety of transformations to be applied to three-dimensional models constructed from two-dimensional motifs or text.

The following resources are available

34 motifs
4 alphabets
22 procedures

To use the module

1) Type **LOAD"3DMOD1"** and press **RETURN**. This loads the program prelude and the Graphito resources.
2) Write a BASIC program from line 10 onwards using the Graphito resources (the Graphito resources start at line 1000): the simplest program loads a motif, 'thickens' it, and displays it (see PROCmotif3D for an example).
3) Type **RUN** and press **RETURN**.

Program prelude

The program prelude that is loaded with the module cannot be altered.

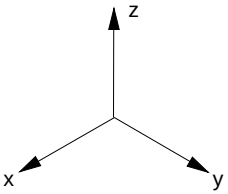
Procedures

The following procedures are available

PROCcylgrid
PROCdecorcube
PROCdecorcyl
PROCdecorsphere
PROCinitialise
PROCload
PROCloadalpha
PROCloadscreen
PROCmotif3D
PROCrotatex
PROCrotatey
PROCrotatez
PROCsavescreen
PROCscale
PROCsetorigin
PROCspheregrid
PROCtextoncube
PROCtext3D
PROCtraceon
PROCtranslate
PROCuniscale
PROCviewpoint

Using 3DMOD1

To plot objects in three dimensions we need a set of three-dimensional coordinates. For 3DMOD1 these are



with the origin at the centre of the screen. Procedures like PROCtext3D and PROCmotif3D do not specify a position (as the equivalent 2D modules do). It is easier for the user to move an object around then plot it than it is to plot an object by specifying absolute coordinates in three-dimensions.

To create a three-dimensional scene the program structure is largely the same as for 2DMOD1. A set of three-dimensional procedures are specified and the object is then plotted and operated on by the procedures. The imporiant difference is in the viepoint specification. A position must be specified from where the object is to be viewed. The module contains a standard or default viewpoint but this can be overidden and the constructed scene viewed from any angle. For a single object scene we can achieve scaling and rotation by changing the viewpoint.

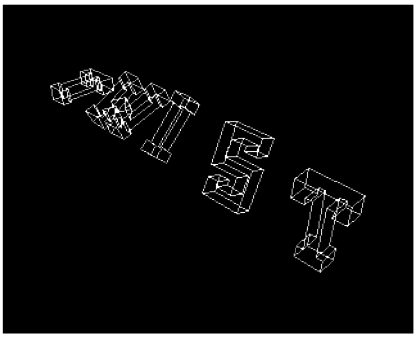
Making an object bigger is the same as moving the viewpoint nearer the object Making an object smaller is equivalent to moving the viewpoint further away. Similarly, rotating an object is equivalent to moving the viewpoint around it. For a multi-object scene, where we may want one object to be bigger than another or one object rotated and the other not, we build up the scene by applying a transformation set to each object. The entire scene is then viewed from any angle.

A program structure is then:

set up a viewpoint
load first object
specify transformations for first object
plot first object
load second object
specify transformations for second object
plot second object
and so on

For example, here each letter in the word TWIST is loaded and manipulated in turn Four applications of PROCrotatex, each of 22.5 degrees, means that the first T is horizontal and the last T is vertical.

```
  5 PROCloadalpha("ALPHA1")
 10 PROCsetorigin(500,350)
 20 PROCviewpoint(1300,-45,55)
 30 PROCtranslate(-700,0,0)
 40 PROCtext3D("T",50,50)
 50 PROCtranslate(250,0,0):PROCrotatex(22.5,0,0)
 60 PROCtext3D("W",50,50)
 70 PROCtranslate(300,0,0):PROCrotatex(22.5,0,0)
 80 PROCtext3D("I",50,50)
 90 PROCtranslate(250,0,0):PROCrotatex(22.5,0,0)
100 PROCtext3D("S",50,50)
110 PROCtranslate(250,0,0):PROCrotatex(22.5,0,0)
120 PROCtext3D("T",50,50)
```

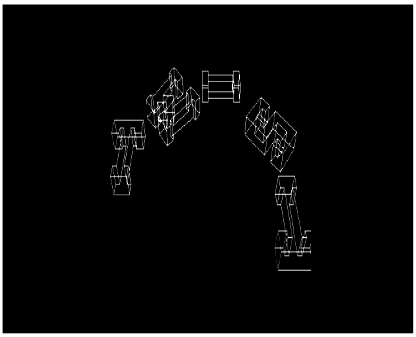


Try experimenting with different values for the parameters in PROGviewpoint ant see what effect this has.

Here is a similar type of result using PROCrotatez.

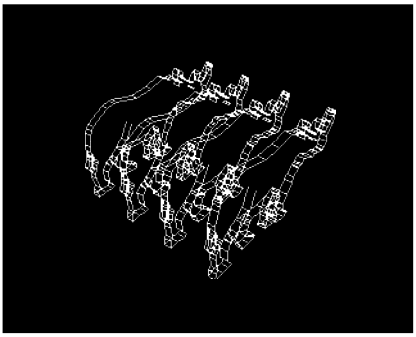
```
  5 PROCloadalpha("ALPHA1")
 10 PROCsetorigin(600,350)
```

```
 20 PROCtranslate(-400,0,0)
 30 PROCtext3D("T",50,50)
 40 PROCrotatez(-45,0,0)
 50 PROCtext3D("W",50,50)
 60 PROCrotatez(-45,0,0)
 70 PROCtext3D("I",50,50)
 80 PROCrotatez(-45,0,0)
 90 PROCtext3D("S",50,50)
100 PROCrotatez(-45,0,0)
110 PROCtext3D("T",50,50)
```



Motifs can of course be used as well. Here a motif is translated and rotated by procedures repeated in a FOR loop.

```
 10 PROCload("RHINO")
 20 FOR shift=0 TO 600 STEP 200
 30   PROCinitialise
 40   PROCviewpoint(2000,45,60)
 50   PROCrotatex(90,0,0)
 60   PROCtranslate(0,shift,0)
 70   PROCmotif3D(30)
 80NEXT shift
```



3DMOD2

This module supplies the user with a simple three-dimensional facility that enables three-dimensional objects (with a small number of faces) to be tranformed and displayed. Three-dimensional scenes can be constructed by placing such objects at difterent positions in three-dimensional space.

Back surface elimination is applied individually to each object. This gives a degree of hidden surface removal. There are two aspects of back surface elimination that prevent it from being true hidden surface removal.

1) Surfaces are considered in their entirety and not sub-divided. This means that if a small part of the surface is visible it is all deemed to be visible.
2) Objects in a multi-object scene are considered independently and individually. This means that if one object shadows another this is NOT taken in to account.

Implementation of true hidden surface removal on a micro-computer is problematic both from memory the memory resource aspect and the time taken for the algorithm to execute.

The following resources are available

4 three-dimensional objects
15 procedures

To use the module

1) Type **LOAD"3DMOD2"** and press **RETURN**. This loads the *Graphito* resources (there is no program prelude).
2) Write a BASIC program from line 10 onwards (the *Graphito* resources start at line 1000) using the *Graphito* resources. This must include a MODE setting and a call to PROCsetdatastructures:

```
10 MODE 4
20 PROCsetdatastructures(20,20)
30 PROCload3Ddata("CHURCH")
40 PROCobject3D
```

3) Type **RUN** and press **RETURN**

Procedures

The following procedures are available

PROCinitialise	PROCscale
PROCloadscreen	PROCsetdatastructures
PROCload3Ddata	PROCsetorigin
PROCobject3D	PROCtraceon
PROCrotatex	PROCtranslate
PROCrotatey	PROCuniscale
PROCrotatez	PROCviewpoint

Details of how to use each of these procedures are given later in the manual.

Using 3DMOD2

The four predrawn objects are included so that users can practise manipulating three-

dimensional objects before devising and defining their own. It is this latter process which will provide the main interest in this module.

Unfortunately setting up even a simple three-dimensional object is quite a long-winded process that is usually carried out in practice by using a special drafting program together with an interactive device suc as a graphics tablet.

DATA statements must have the following form:

```
100 DATA "name of object"
110 DATA number of vertices
120 DATA list of vertices
130 DATA number of surfaces
140 DATA list of the vertices in each surface
```

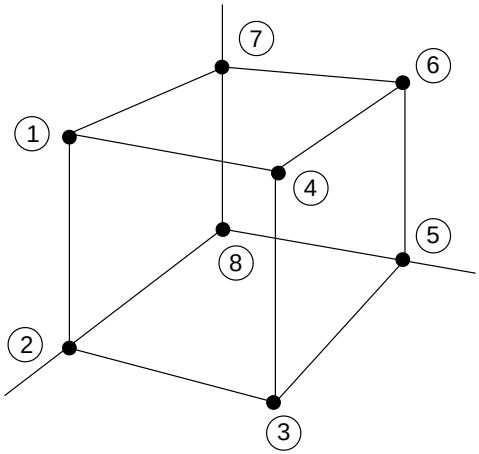
The surface information is necessary for hidden surface calculations.

This is best illustrated by the example of a cube.

```
100 DATA CUBE
110 DATA 8
120 DATA 400,0,400, 400,0,0, 400,400,0
121 DATA 400,400,400, 0,400,0, 0,400,400
122 DATA 0,0,400, 0,0,0
130 DATA 6
```

The list of vertices in each surface is made of six (the number of surfaces) sublists. Each sublist contains the number of vertices in a surface followed by the vertex which is referred to by a vertex number.

The vertex numbers for the cube are:



and there are six surfaces in the cube each made up of four vertices. The final DATA statement is therefore:

```
140 DATA 4,1,2,3,4, 4,3,5,6,4, 4,5,8,7,6
141 DATA 4,8,2,1,7, 4,4,6,7,1, 4,3,2,8,5
```

The vertex numbers in each surface must be listed in anti-clockwise order as seen from the outside of the object

REFERENCE SECTION

PROCalphaslice

2MOD1 2DMOD3 2DMOD4

PROCalphaslice (xs,ys,length,height)

This can be used with the alphabets "ALPHA2" and "ALPHA4". It erases horizontal strips or sets them to the background colour to produce 'sliced' letters. (xs, ys) is the position of the bottom left hand corner of the strip: 'length' and 'height' are the length and height of the strip. Each call to PROCaJphaslice produces one strip, so it must be applied repeatedly, in a FOR loop for instance, to produce a series of strips across the whole height of the text.

examples

ALPHA 2 sliced

ALPHA4 sliced

```
 10 PROCload("RHINO")
 20 PROCdrawandscale(600,780,.8)
 30 PROCloadalpha("ALPHA2")
 40 PROCstretch(2,90)
 50 PROChtext("R",0,300,2,0)
 60 PROChtext("HINO BURGERS",300,300,.5,20)
 70 FOR ys=330 TO 450 STEP 30
 80   PROCalphaslice(0,ys,1300,10)
 90 NEXT ys
100 PROCinitialise
110 PROChtext("REAL RHINO SLICES",0,100,0.4,35)
```

PROCboxwindowin

2DMOD1

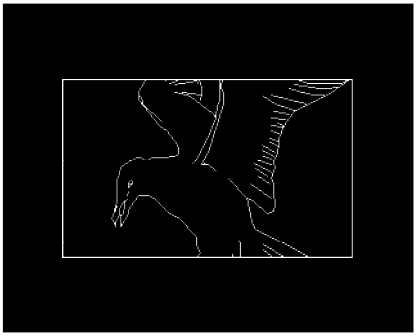
PROCboxwindowin(xs,ys,width,height)

Sets up a rectangular window with dimensions 'width' and 'height'.

(xs,ys) is the position of the bottom left hand corner. Any lines outside this window are deleted

example

```
10PROCload("GULL")
20PROCdrawandscale(640,512,1.5)
30PROCboxwindowin(200,350,900,550)
```



PROCboxwindowout

2DMOD1

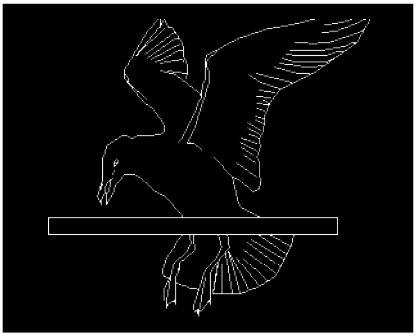
PROCboxwindowout (xs, ys, width, height)

Sets up a rectangular window with dimensions 'width' and 'height'.

(xs, ys) is the position of the bottom left hand corner. Any lines inside this window are deleted.

example

```
10PROCload("GULL")
20PROCdrawandscale(640,512,1.5)
30PROCboxwindowin(200,350,900,50)
```



PROCc_curve

2DMOD4

PROCc_curve(orde,x,y,scale,resolution)

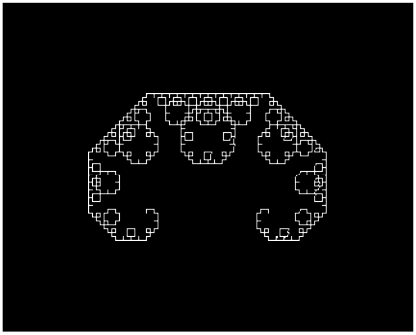
Draws a 'c-curve' of given order at point (x,y). A scale of 1 results in a curve about 600 screen units across. The resolution parameter determines the amount of detail in the pattern. Small resolution values result in a lot of detail. For some 'small resolution' patterns it may be necessary to alter the program prelude to:

```
1 MODE 4 : HIMEM=HIMEM-1260
2 PROCinitialisememory(4,TRUE,FALSE)
```

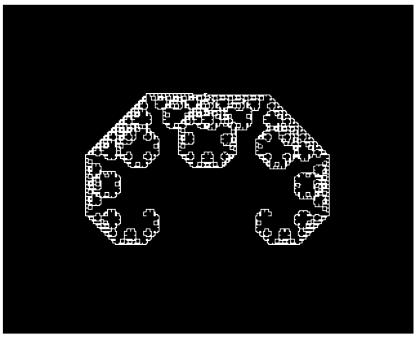
examples

The best known c-curves are of order 2

```
10 PROCc_curve(2,640,410,0.75,15)
```

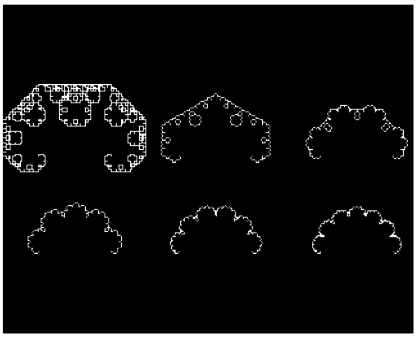


```
10 PROCc_curve(2,640,410,0.75,8)
```



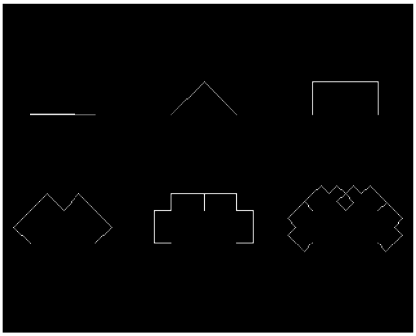
c-curves of orders 2 to 7 (other orders are not permitted)

```
10 order=2
20 FOR y = 600 TO 300 STEP -300
30   FOR x = 220 TO 1100 STEP 440
40     PROCc_curve(order,x,y,0.45,10)
50     order = order + 1
60     NEXT x
70   NEXT y
```



c-curves at different resolutions

```
10 resolution=220
20 FOR y = 700 TO 300 STEP -400
30   FOR x = 220 TO 1100 STEP 440
40     PROCc_curve(2,x,y,0.4,resolution)
50     resolution = resolution/SQR(2)
60     NEXT x
70   NEXT y
```



PROCchardesign

2DMOD3

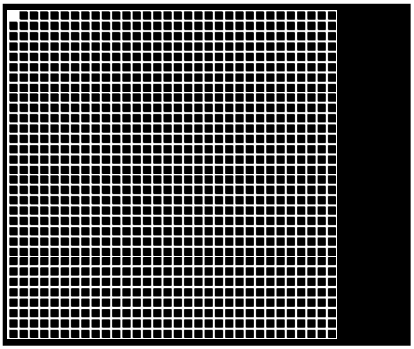
PROCchardesign(columns, rows)

PROCchardesign is a sketchpad facility that can be used to design a motif consisting a group of user-defined characters. Such a motif can then be displayed in a network by using PROCcharnet which is equivalent to PROCnet for line motifs.

User defined characters provide a way of displaying complex motifs quickly on the screen. A character is an 8 x 8 group of pixels that contains a pattern of foreground and background colour. A complex character shape can be displayed on the screen in a fraction of the time that it would take to build up the same shape using the normal MOVE and DRAW commands. Groups of more than one character can be used to represent quite complex motifs.

The two parameters of PROCchardesign indicate the width and height of the motif to be designed. The width and height are specified as a number of characters and the maximum in both directions is 4.

On entering PROCchardesign, you will be asked if you want to design a new motif. If you reply Y for Yes, then you will start with a blank motif, otherwise the user-defined characters will remain set to their previous values, presumably designed in a previous run of the program. If you reply Y a blank planning grid with a flashing cursor appears on the screen. This grid represents your character motif on a large scale and each small square in the grid represents a pixel.



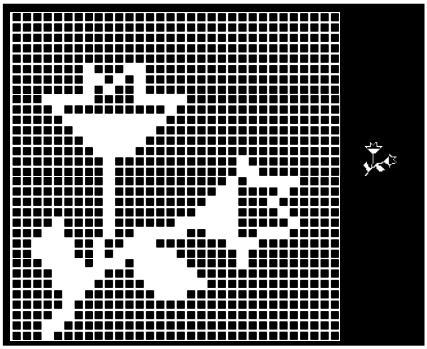
You can now start to design your character motif. This is done by moving the cursor around the rid with the arrow keys and by switching the pixels on or off using various command keys. As you do this, the motif will be continuously displayed at its true size to the right of the grid. The single key commands used in the design process are:

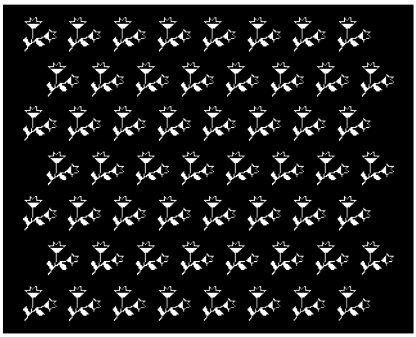
↑ ↓ ← →	moves the cursor around the planning grid
F	switches the square marked by the cursor and all squares subsequent visited to Foreground colour
B	switches the square marked by the cursor and all squares subsequent visited to Background colour
M	allows the flashing cursor to be moved around the grid without affecting the squares visited
Q	Quits PROCchardesign

examples

The facilities described so far were used to design the following motif. When the flower motif had been designed, Q was used to exit PROCchardesign

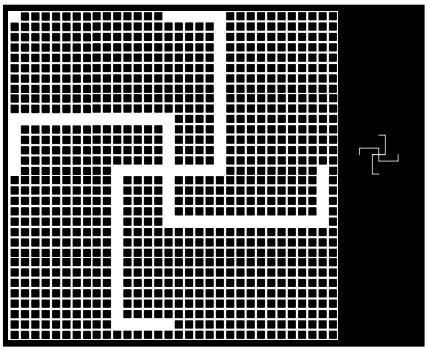
```
10 PROCchardesign(4,4)
20 PROCcharnet(8,7,140,140,50,128,70)
```

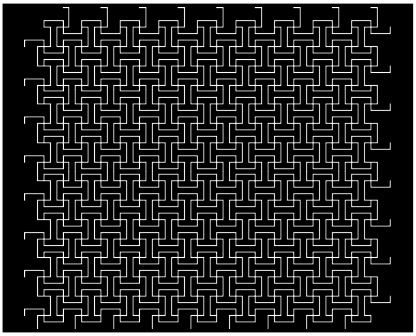




Here is an intriguing example of a net in which the individual motifs are difficult to pick out.

```
10 PROCchardesign(4,4)
20 PROCcharnet(9,16,120,60,50,100,60)
```





When designing motifs for use in networks, it is often desirable that the motif should be ymmetrical in Some way. The character design utility includes two commands that can be used to generate character motifs that exhibit perfect mirror symmetry about a vertical or horizontal axis.

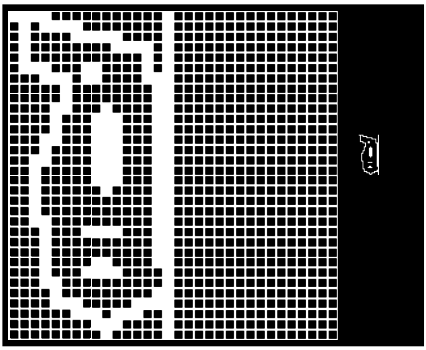
H reflects the top half of the motif about a Horizontal line across the centre, thus creating a motif with perfect mirror symmetry about this line.

V reflects the left half of the motif about a Vertical line down the centre, thus creating a motif with perfect mirror symmetry about this line.

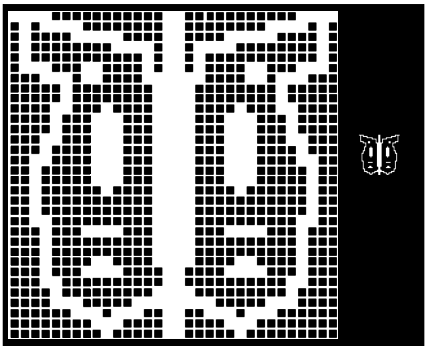
examples

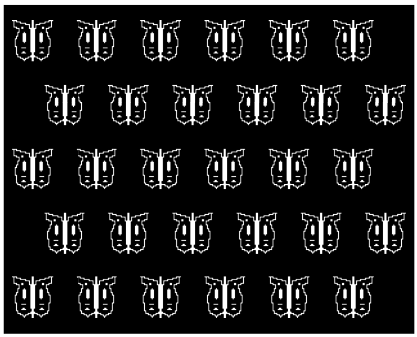
The left half of the butterfly was designed using the commands **F**, **B**, and **M**.

```
10 PROCchardesign(4,4)
20 PROCcharnet(6,5,200,200,50,200,100)
```



The command **V** was used to reflect the left half of the motif in the line down the centre



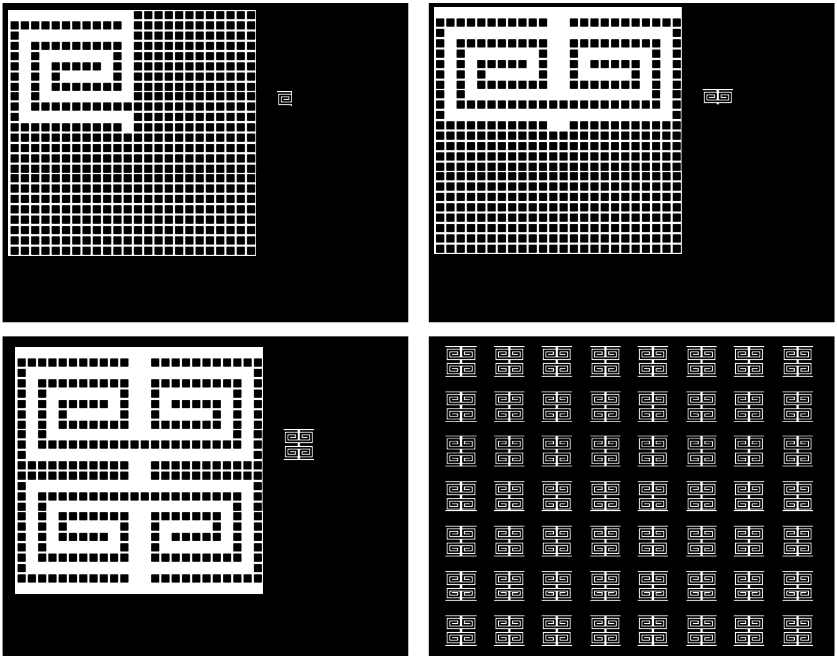


The top left corner of a motif was deisgned using **F**, **B** and **M**.

The command **V** was used to reflect about a vertical axis.

The command **H** was then used to reflect about a horizontal axis

```
10 PROCchardesign(3,3)
20 PROCcharnet(8,7,150,140,50,150,0)
```



Finally, there are three commands for handling completed designs.

I restarts or Initialises the design process with a fresh grid. You will be asked what size of grid you require.

S allows the characters of a motif to be Saved in a file. You will be asked to type the name of the file

L Loads a previously saved motif from a file.

PROCcharnet

2DMOD3 2DMOD4

see **PROCnet**

PROCcirclewindowin

2DMOD1

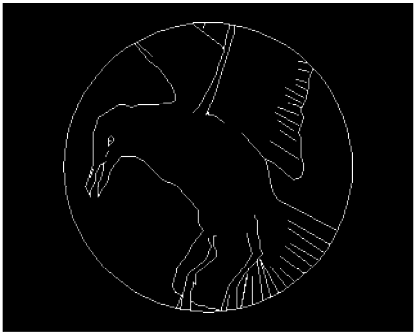
PROCcirclewindowin(xo,yo,radius)

Sets up a circular window of radius 'radius' and centre (xo,yo).

Any lines outside this window are deleted. The effect is immediate and does not affect subsequent plotting.

example

```
10 PROCload("GULL")
20 PROCdrawandscale(640,512,2)
30 PROCcirclewindowin(640,512,450)
```



PROCcirclewindowout

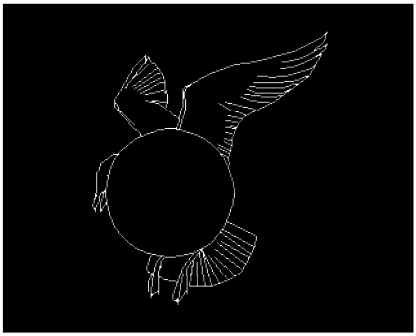
2DMOD1

PROCcirclewindowout(xo,yo,radius)

Sets up a circular window of radius 'radius' and centre (xo,yo). Any lines inside this window are deleted.

example

```
10 PROCload("GULL")
20 PROCdrawandscale(640,512,1.25)
30 PROCcirclewindowout(640,512,200)
```



PROCcolouredflake

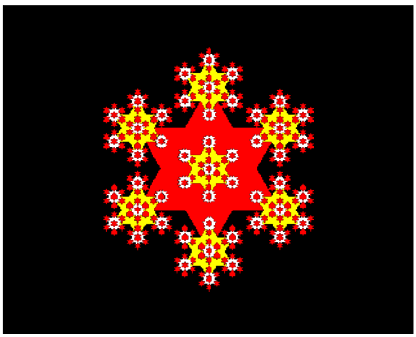
2DMOD4

PROCcolouredflake(x,y,scale,resolution,reduction factor)

As for PROCsnowflake, but parts of the flake are colour filled. Although primarily intended for use with colour, this procedure produces quite effective results in black and white.

example

```
10 PROCcolouredflake(640,512,1,8,0.33)
```



PROCcyclgrid

3DMOD1

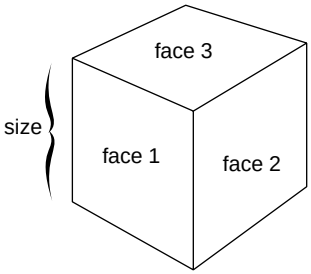
see **PROCdecorcyl**

PROCCdecorcube

3DMOD1

PROCdecorcube(size,face)

Decorates one of three faces of a cube with the current motif. 'size' is the size of the cube and 'face' is the number of the face.

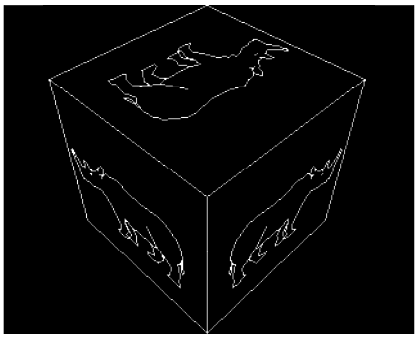


No hidden line removal is employed and the three face numbers are as seen from a viewpoint: phi = 45° theta = 45°. The wewpoint must be constrained to this region. 'size' should be sufficient to accommodate the motif. This means using a value in the ge 750-1000.

examples

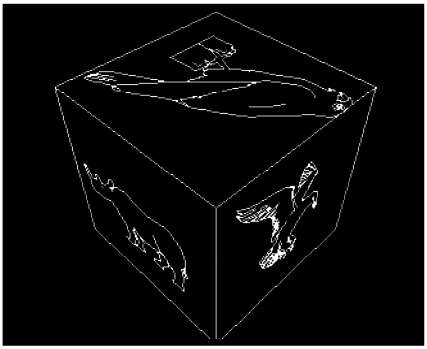
Three calls of PROCdecorcube using the same motif for each face.

```
10 PROCviewpoint(3200,45,45)
20 PROCload("RHINO")
30 PROCtranslate(0,0,250)
40 PROCdecorcube(900,1)
50 PROCdecorcube(900,2)
60 PROCdecorcube(900,3)
```



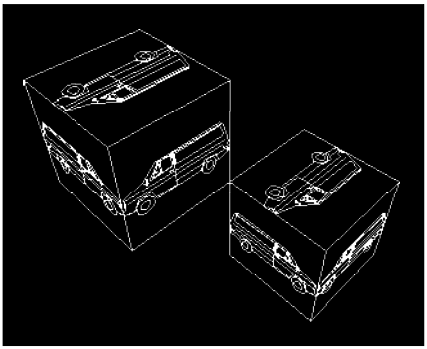
Three calls of PROCdecorcube using a different motif for each face

```
 10 PROCviewpoint(3500,45,45)
 20 PROCload("RHINO")
 30 PROCtranslate(0,0,250)
 40 PROCdecorcube(1000,1)
 50 PROCload("GULL")
 60 PROCdecorcube(1000,2)
 70 PROCload("EAGLE")
 80 PROCdecorcube(1000,3)
```



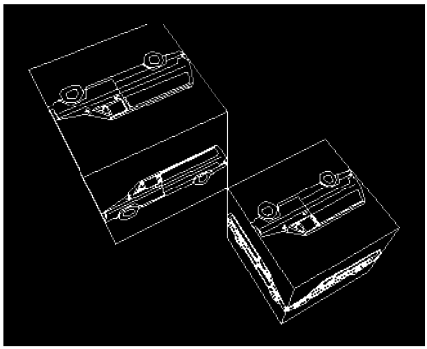
Two cubes translated

```
 10 PROCviewpoint(3000,45,45)
 20 PROCload("TRANSIT")
 30 PROCtranslate(0,-750,0)
 40 PROCdecorcube(750,1)
 50 PROCdecorcube(750,2)
 60 PROCdecorcube(750,3)
 70 PROCinitialise
 80 PROCtranslate(-750,0,-750)
 90 PROCdecorcube(750,1)
100 PROCdecorcube(750,2)
110 PROCdecorcube(750,3)
```



Same scene; different viewpoint

```
10 PROCviewpoint(3000,60,30)
```



PROCdecorcyl PROCcylgrid

3DMOD1

PROCdecorcyl(radius)

PROCcylgrid(radius,size)

These procedures decorate the surface of a cylinder with a motif and a set of grid lines respectively. PROCdecorcyl has the effect of sticking a motif on to the surface of a cylinder. The transformation is non-linear - it is as if the motif were a flat flexible sheet that can be wrapped onto the surface.

The transformation is best perceived when grid lines are drawn on the surface of the cylinder.

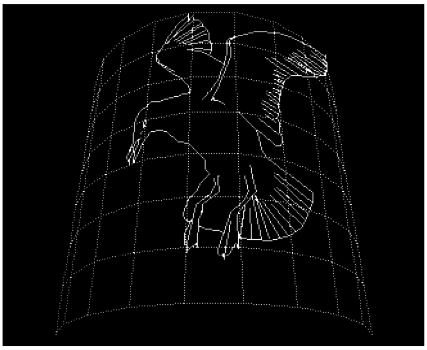
PROCcylgrid draws lines of latitude and longitude on the front surface of the cylinder 'size' specifies the distance between the lines.

The 'radius' parameter of both procedures should be identical and must be compatible with the size of the motif. A value of 300 will do for most motifs. To make the cylinder larger or smaller use PROCuniscale.

Motifs that contain long straight lines are not suitable candidates for this transformation.

example

```
10 PROCinitialise
20 PROCload("GULL")
30 PROCviewpoint(2000,0,120)
40 PROCtranslate(0,0,-100)
50 PROCuniscale(1.5)
60 PROCcylgrid(350,100)
70 PROCdecorcyl(350)
```



PROCdecorsphere PROCspheregrid

3DMOD1

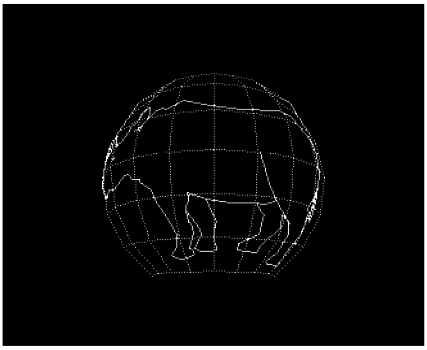
PROCdecorsphere(radius)
PROCspheregrid(radius,size)

These procedures decorate the surface of a sphere with a motif and a set of grid lines respectively. PROCdecorsphere has the effect of sticking a motif on the sw of a sphere, whose radius is specified by 'radius'. PROGsphere shows lines of 'latitude' and 'longitude' on the front hemisphere.

The considerations of radius, size and choice of motif are as for PROCdecorcyl.

example

```
10 PROCload("RHINO")
20 PROCviewpoint(2000,0,105)
30 PROCuniscale(1.5)
40 PROCdecorsphere(300)
50 PROCspheregrid(300,100)
```



PROCdragon

2DMOD4

PROCdragon(order,x,y,scale,resolution)

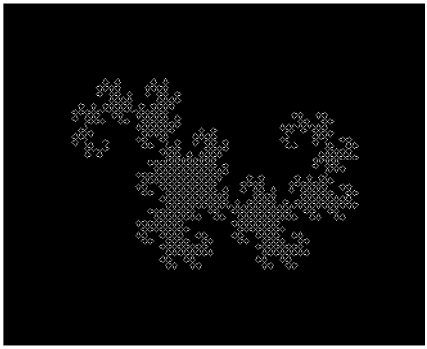
Draws a 'dragon curve' of given order at point (x, y). A scale of 1 results in a curve about 600 screen units across. The resolution parameter determines the length of the individual straight lines that make up the pattern. Small resolution values give a very detailed pattern. For some 'small resolution' patterns it may be necessary to alter the program prelude to:

```
1 MODE 4 : HIMEM=HIMEM-1260
2 PROCinitialisememory(4,TRUE,FALSE)
```

examples

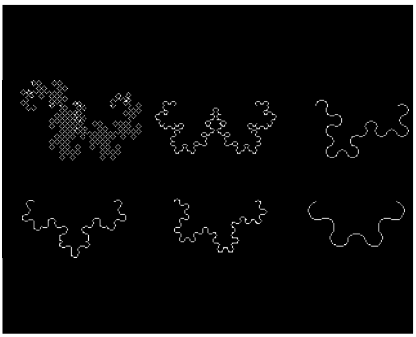
The best known curve with a true dragon-like shape is of order 2

```
10 PROCdragon(2,640,600,1.17,15)
```



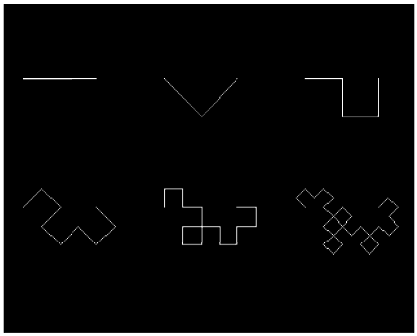
Dragon curves of orders 2 to 7 (other orders are not permitted)

```
10 order=2
20 FOR y=800 TO 500 STEP -300
30   FOR x=220 TO 1100 STEP 440
40     PROCdragon(order,x,y,0.5,15)
50     order=order+1
60     NEXT x
70   NEXT y
```



Dragon curves at different resolutions

```
10 resolution=300
20 FOR y=800 TO 400 STEP -400
30   FOR x=220 TO 1100 STEP 440
40     PROCdragon(2,x,y,0.45,resolution)
50     resolution=resolution/SQR(2)
60     NEXT x
70   NEXT y
```



PROCdrawandinflate(xo,yo,xe,ye,contraction,scale)

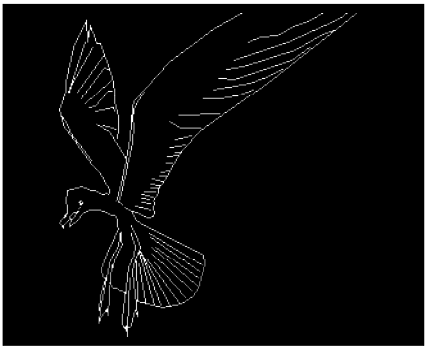
2DMOD1

PROCdrawanddeflate (xo,yo,xe,ye,contraction,scale)

Deflates about a centre (xe, ye) producing a somewhat strange effect. (xo, yo) is the screen drawing position. This procedure is a non-linear transform and can only be used on its own, operating on the original motif.

example

```
10 PROCload("GULL")
20  PROCdrawanddeflate(500,400,0,0,2,1)
```



PROCdrawandinflate

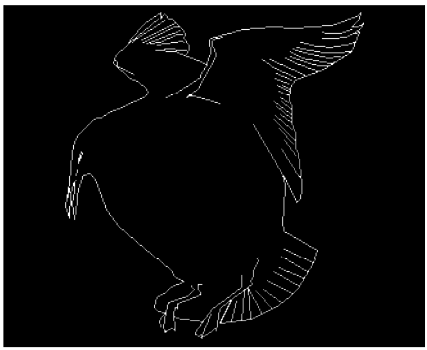
2DMOD1

PROCdrawandinflate(xo,yo,xe,ye,expansion,scale)

Deflates about a centre (xe, ye). (xo, yo) is the drawing centre. This procedure is a vlinear transform and can only be used on its own, operating on the original motif.

example

```
10 PROCload("GULL")
20 PROCdrawandinflate(640,480,0,0,2,1)
```

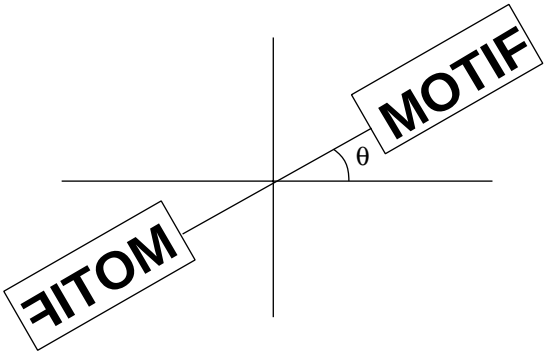


PROCdrawandreflect

2DMOD1

PROCCdrawandreflect(xo,yo,scale,dist,theta)

Generates a reflection about the vertical direction and draws both the original motif and its reflection. The motif pair can also be rotated about an angle 'theta'.



examples

```
10 PROCload("GULL")
20 PROCdrawandreflect(512,450,.8,200,-45)
```

PROCdrawandscale

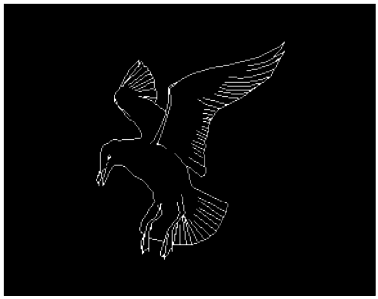
2DMOD1 2DMOD2 2DMOD3 2DMOD4

PROCdrawandscale(xo,yo,scale)

Draws a motif at position(xo,yo). The size of the motif is determined by 'scale'.

example

```
10 PROCload("GULL")
20 PROCdrawandscale(640,512,1)
```



PROCellipse

2DMOD4

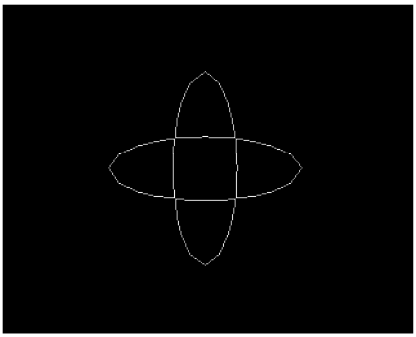
PROCellipse {x, y, xradius, yradius)

Generates an ellipse with centre at (x,y) and eccentricity defined by 'xradius' and 'yradius'. If 'xradius' and 'yradius' are equal a circle is drawn.

example

Two ellipses with same centre

```
10 PROCellipse(640,512,100,300)
20 PROCellipse(640,512,300,100)
```



PROCexspiral

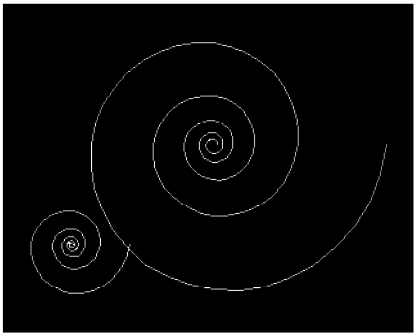
2DMOD4

PROCexspiral(x,y,radius)

Draws an 'expanding' spiral at centre (x,y) with a radius set by the parameter 'radius'.

example

```
10 PROCexspiral(640,512,15)
20 PROCexspiral(200,200,5)
```



PROCfore_to_back_col PROCrestore_fore_col

2DMOD1 2DMOD3 2DMOD4

Changes foreground colour to background colour (the other procedure reverses the process) and is used mainly to 'extend' alphabets.

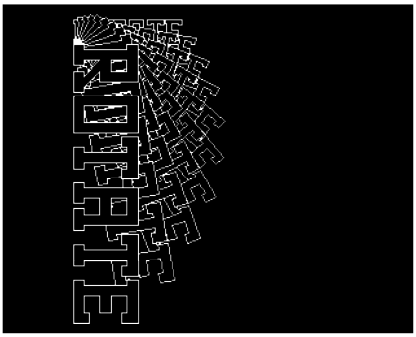
The alphabet styles ALPHA1 and ALPHA2 can be used in combination to provide an outline obscuring alphabet. The alphabet style is as ALPHA 1 but the interior is filled in background colour and wipes out or obscures anything that is already in the area occupied by the alphabet. ALPHA2 is loaded first and the fill colours changed to background colour using PROCfore_to_back_col. The foreground colour is then restored (PROCrestore_fore_col) and ALPHA1 is used to draw an outline around 'wiped' areas.

examples

```
10 PROCload("GULL")
20 PROCdrawandscale(500,500,1)
30 PROCloadalpha("ALPHA2")
40 PROCfore_to_back_col
50 PROChtext("SEAGULL",300,500,.5,20)
60 PROCloadalpha("ALPHA1")
70 PROCrestore_fore_col
80 PROChtext("SEAGULL",300,500,.5,20)
```

Superimposing outline obscuring alphabets. Rotation is controlled by a FOR loop.

```
 10 scale=0.4
 20 FOR theta=0 TO -90 STEP -10
 30   PROCinitialise
 40   PROCloadalpha("ALPHA2")
 50   scale=scale*1.1
 60   PROCrotate(theta)
 70   PROCfore_to_back_col
 80   PROChtext("ROTATE",300,870,scale,10)
 90   PROCrestore_fore_col
 95   PROCinitialise
100   PROCloadalpha("ALPHA1")
110   PROCrotate(theta)
120   PROChtext("ROTATE",300,870,scale,10)
130   NEXT theta
```

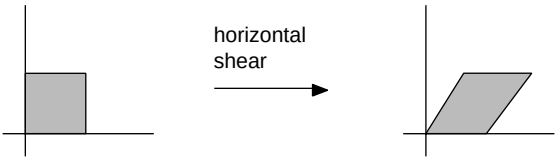


PROChshear

2DMOD1 2DMOD2 2DMOD3 2DMOD4

PROChshear(amount)

Shears the current motif in memory in the horizontal direction. Sheering is an operation that turns a square, for example, into a parallelogram.



See also PROCvshear.

examples

```
10 PROCload("GULL")
20 PROChshear(1.2)
30 PROCdrawandscale(350,512,1)
```

PROChtext

2DMOD1 2DMOD2 2DMOD3

Draws a string of characters, horizontally starting from position (xstart,ystart). The intercharacter gap is given by 'gap' and scale is the pure magnification of each character (and the gap).

See also PROCvtext.

examples

```
10 PROCload("ALPHA1")
20 PROChtext("HORIZONTAL",70,500,0.8,20)
30 PROChtext("TEXT",250,200,1.2,20)
```

PROCinitialise

2DMOD1 2DMOD2 2DMOD3 2DMOD4 3DMOD1 3DMOD2

Restores the motif to it original untransformed state. This procedure can be used after any combination of transforms.

example

```
PROCscale(3)
PROCdrawandscale(640,512,1)
PROCinitialise
PROCscale(2)
PROCdrawandscale(640,512,1)
```

If PROCinitialise was omitted the motif would be scaled (in the second call of PROCscale) by a factor of six.

PROCinteract

2DMOD1 2DMOD2

PROCinteract(mode,number)

Switches the program into an interactive state. The parameter is the number of the mode in which the program is running. While in the interactive state, the user can draw lines and colour areas of the screen. This process is controlled from the keyboard or with a joystick.

PROCinteract can be used to draw and colour pictures and to make additions to pictures drawn using the other utilities.

If you request interaction by setting the last parameter of PROCinitialisememory in a program prelude to TRUE, the program will announce that a special memory block needs to be loaded from a file. This block contains a machine code colouring program and a colour mixture table appropriate for the mode: on a disc system, the appropriate block will be loaded automatically, but on a cassette system, you will have to position the tape. The file required depends on the mode in which the program is running

MODE	file *required*
0 (2DMOD2 only)	MODE1BL
1 (2DMOD2 only)	MODE1BL
2 (2DMOD2 only)	MODE2BL
4	MODE5BL
5	MODE5BL

When PROCinteract is called, a small cross should appear in the centre of the screen. This cross will be used to point to different parts of the screen. It can be moved about by using the 'cursor arrow keys' or by switching to Joystick mode and using one of your

joysficks. (Instructions for switching to Joysitck mode appear later.)

All the commands used in interaction involve pressing a single key. They are described below.

In a two colour mode such as MODE 4, one colour is used for background and the other for foreground plotting. In such a mode, only the foreground colour (usually white) can be used for painting.

In a four colour mode such as MODE 5, three basic colours are available in addition to the background colour. As well as laying down pure colour on the screen, the three basic colours can be combined in various ways to give a number of different colour mixtures. Each mixture has Its own distinctive texture determned by the way in which spots of the basnc colours are organised. These can be combined in horizontal stripes. vertical stripes or in a spotted pattern.

If PROCinteract is used in MODE 5, the basic colours together with all the possible colour mixtures are displayed in a palette at the right of the screen and a mixture can be selected, as described later, by moving the cross into the required mixture. The short line to the right of the palette indicates the mixture that is currently selected.

Note that, in 2DMOD2, PROCinteract will CHAIN the program INTERAC whichs sets up the palette of colours and controls interaction. This is the only way in which sufficient screen memory can be made available for MODES 0, 1 or 2. Because a new program is loaded by PROCmteract, you should first test the mofif handling part of your program without the call or PROCinteract. You should also save your main program before you RUN n with a call of PROCinteract.

The cursor arrows are used to move the cross round the screen in order to carry out various operations. To move the cross quickly, hold down the appropriate key and the cross will proceed across the screen in a series of jumps. To positlon the cross more accurately, tap the appropriate arrow key quickly and the cross will move in very small steps.

To paint a region, move the cross into the regmn and press P for Pant. In a mode with more than two colours, you can select a new colour mixture by moving the cross into the colour you want in the palette on the right (i.e. dip your 'paintbrush' in the 'paint').

Use the command D (in a two colour mode) or move the cross into the letters DR at the bottom of the palette to switch to line drawing mode. Move the cross to the point at which drawing is to start and press F for Fix. Use the arrows to draw to the next point required and press F agam. To continue the same \me, keep doing this. To start a new line, use the command O for Off to switch the line off, move to the start of the a new line and press F again. To switch back to painting in a multi-colour mode, more the end of the line into the colour you want. Remember that you can use this facility to draw and colour your own pictures, or to add lines to pictures bullt up from the supplied motifs and alphabets.

To clear the screen, press C for Clear

To switch to joystick mode, press J for Joysticks. You should now find that you can use one of your joysticks to position the cross. When painting, you should use the fire button instead of P and when drawing, use the fire button instead of F. Pressing J again will switch off joystick control.

Once a picture has been coloured, keys B (for Background) and 1 to 8 can be used to change the basic colours and mixtures containing the basic colours. Pressing one of these keys will instantaneously change one colour to a new one. Pressing the same key seven times will cycle through all the possibilities for one of the basic colours. This gives thousands of possible colour combinations for a given coloured picture. B is used to change the background colour and the other keys are used to change the other basic colours. In a two-colour mode, only key 1 can be used in this way. In a four coliur mode, only keys 1, 2 and 3 can be used and in MODE 2 all eight keys can be used. E for Experiment causes the computer to try random colour changes at one second intervals. Hitting space bar makes it stop. After experimenting with different colour combinations, type R for Restore to restore the standard colour settings.

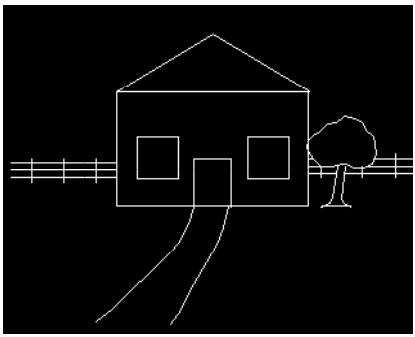
The commands S and L can be used to Save or Load screen dumps. S operates in the same way as PROCsavescreen and L as PROCloadscreen.

Command key summary

↑ ↓ → ←	used to move the cross around the screen
P	Paint the area marked by the cross
C	Clear the screen
D	Draw
F	Fix a point on the line being drawn in drawing mode
O	switch Off the drawing line
J	switch Joystick mode on or off
B, **1** to **8**	instantaneously change one of the basic colours
E	Experiment with different basic colour settings
R	Restore the standard colour settings
Q	Quit PROCinteract
S	Save a screen
L	Load a screen dump

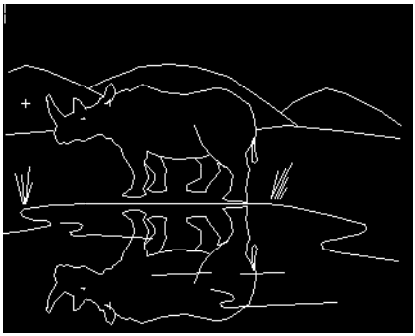
examples

```
  1 MODE 4 : HIMEM=HIMEM-570
  2 PROCinitialisememory(4,FALSE,FALSE,TRUE)
 10 PROCinteract(4)
```



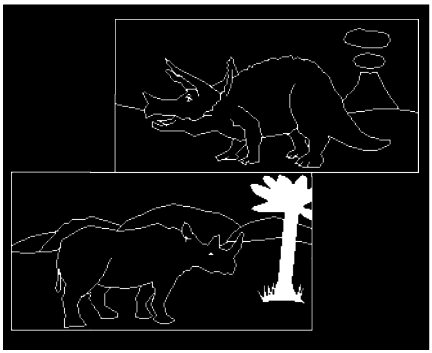
The landscape was added interactively,

```
 1 mode=4
 2 MODE mode : HIMEM=HIMEM-1260-570
 3 PROCinitialisememory(mode,TRUE,FALSE,TRUE)
10 PROCload("RHINO")
20 PROCdrawandscale(512,700,.9)
30 PROCreflectx
40 PROCdrawandscale(512,200,.9)
50 PROCinteract(mode)
```



The landscapes were added interactively.

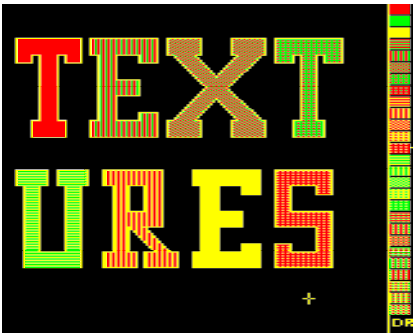
```
 1 mode=4
 2 MODE mode : HIMEM=HIMEM-1260-570
 3 PROCinitialisememory(mode,TRUE,FALSE,TRUE)
10 PROCload("RHINO")
20 PROCreflecty
30 PROCdrawandscale(400,230,.8)
40 PROCload("TRICOP")
50 PROCdrawandscale(810,736,.8)
60 PROCinteract(mode)
```



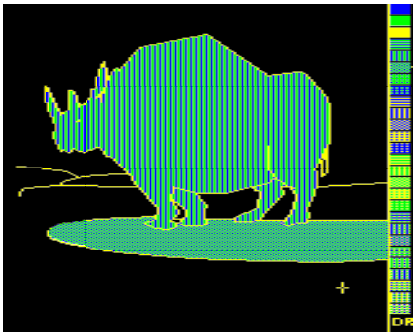
The rhino was coloured interactively.

```
 1 mode=4
 2 MODE mode : HIMEM=HIMEM-1260-2340-570
 3 PROCinitialisememory(mode,TRUE,TRUE,TRUE)
10 PROCload("RHINO")
20 PROCdrawandscale(712,600,1.745)
30 PROCloadalpha("ALPHA1")
40 PROChtext("WHITE",436,660,.6,20)
50 PROChtext("RHINO",388,520,.7,20)
60 PROCinteract(mode)
```

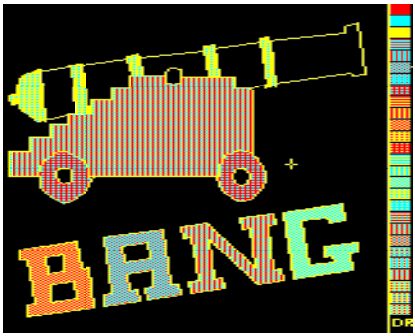

```
 1 mode=5
 2 MODE mode : HIMEM=HIMEM-2340-570
 3 PROCinitialisememory(mode,FALSE,TRUE,TRUE)
10 PROCloadalpha("ALPHA1")
20 PROChtext("TEXT",32,610,1.6,20)
30 PROChtext("URES",32,200,1.6,30)
40 PROCinteract(mode)
```



```
 1 mode=5
 2 MODE mode : HIMEM=HIMEM-1260-570
 3 PROCinitialisememory(mode,TRUE,FALSE,TRUE)
10 PROCload("RHINO")
20 PROCdrawandinflate(600,750,-12,84,2,.8)
30 PROCinteract(mode)
```



```
 1 mode=5
 2 MODE mode : HIMEM=HIMEM-1260-2340-570
 3 PROCinitialisememory(mode,TRUE,TRUE,TRUE)
10 PROCload("CANNON")
20 PROCreflecty
30 PROCdrawandscale(550,700,1.2)
40 PROCinitialise
50 PROCloadalpha("ALPHA1")
60 PROCstretch(1.5,0)
70 PROCrotate(8)
80 PROChtext("BANG",70,50,1.1,20)
90 PROCinteract(mode)
```



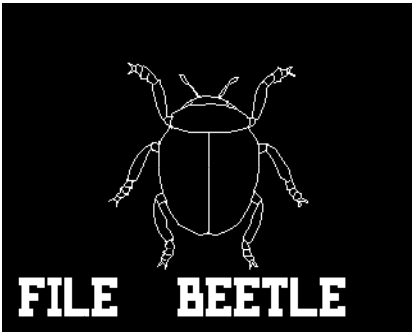
PROCload

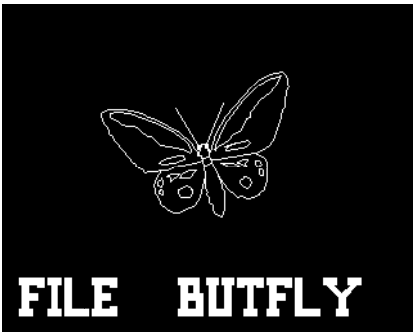
2DMOD1 2DMOD2 2DMOD3 2DMOD4 3DMOD1

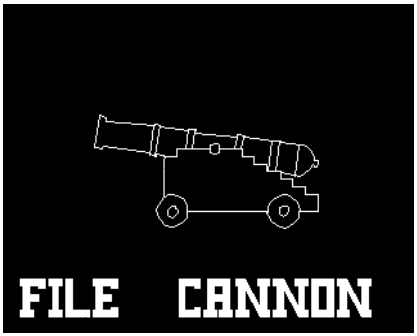
Loads a motif from a disc or tape file into memory. Any subsequent Graphito manipulations operate on this motif, The motif name must be in capitals and enclosed in double quotes. To change the motif being operated on another call of PROCload must be made.

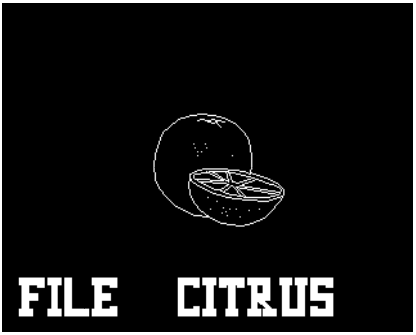
To facilitate easy program development PROCload checks to see if a motif is already in memory. If it is the motif is not reloaded from file. This means that the program can be run many times using the same motif, but the motif is only loaded from file on the execution.

The following motifs are available in the files as named.



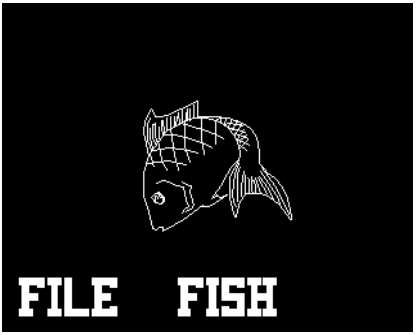


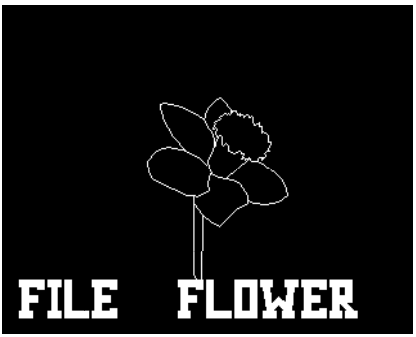



FILE CITRUS

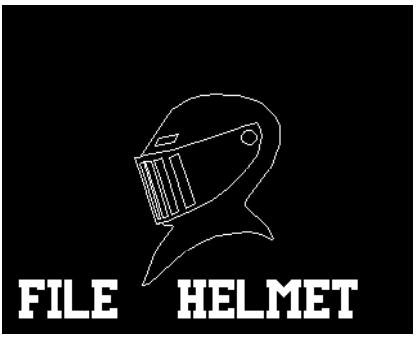

FILE EAGLE

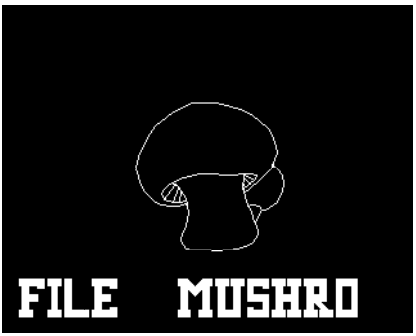

FILE ELEPHA


FILE FISH


FILE FLOWER

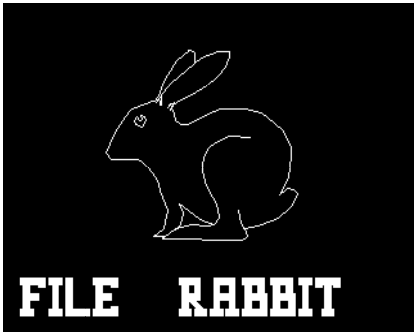

FILE GUN

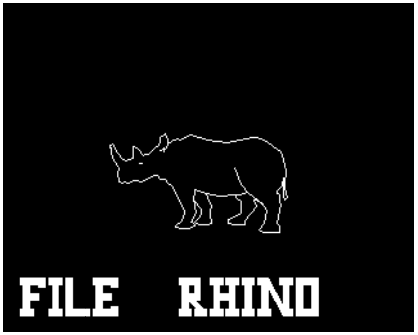

FILE HELMET


FILE MUSHRO

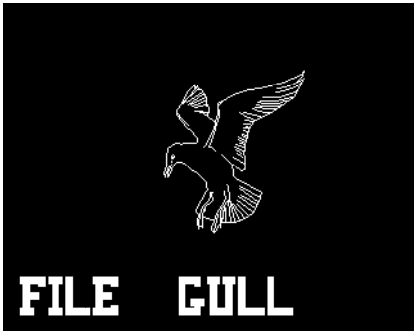

FILE ORCHID


FILE PARROTS


FILE RABBIT

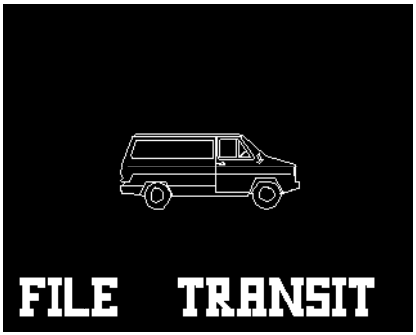

FILE RHINO

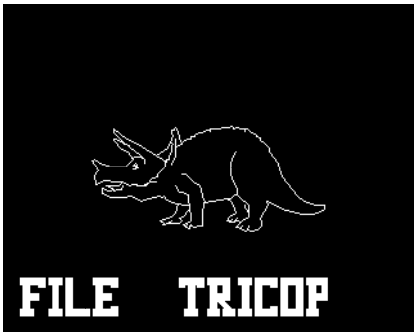

FILE ROSE

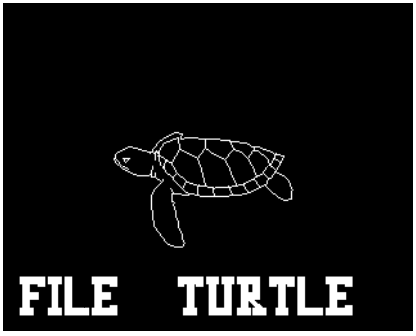

FILE GULL


FILE TELEPH


FILE THISTLE

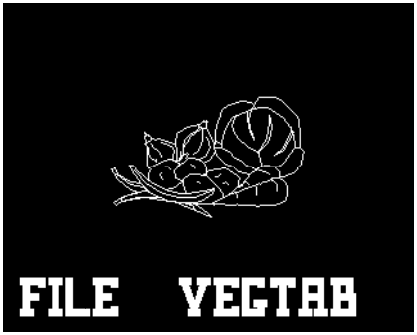

FILE TRANSIT


FILE TRICOP


FILE TURTLE


FILE UMBREL


FILE VASE

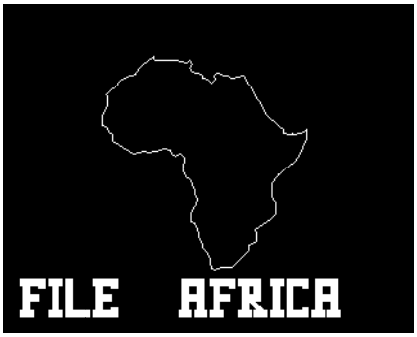

FILE VEGTAB

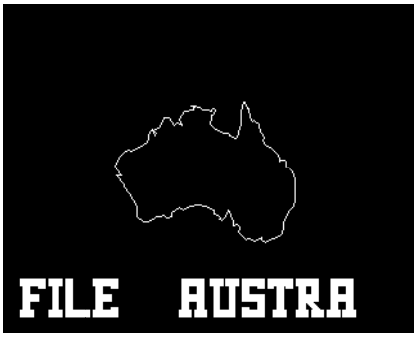

FILE BRITAIN


FILE IRELAND


FILE EUROPE


FILE EUROPEF


FILE AFRICA


FILE AUSTRA


FILE NAMERI


FILE SAMERI

PROCloadalpha

2DMOD1 2DMOD3 2DMOD4 3DMOD1

PROCloadalpha(alphabet)

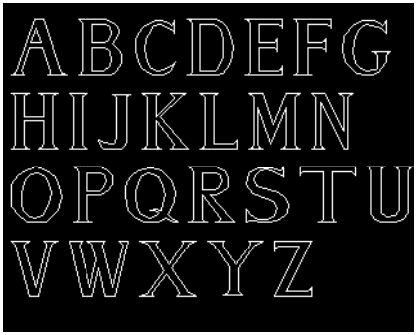
Loads an alphabet from a disc or tape file into memory. Any subsequent Graphito manipulations operate on this alphabet The alphabet name must be in capitals and enclosed in double quotes.

Four predefined alphabets can be called by this procedure.

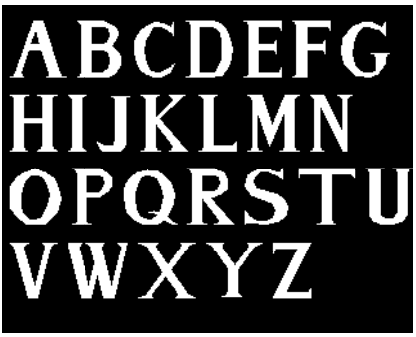
ALPHA1

ALPHA2

ALPHA3



ALPHA4



PROCloadscreen

2DMOD1 2DMOD3 2DMOD4 3DMOD1 3DMQD2

PROCloadscreen

Loads a binary screen dump from a file called SCREEN. This will normally be a file that was saved using PROCsave. It is the programmer's responsibility to ensure that a screen dump is loaded in the same MODE as that in which it was created.

PROCload3Ddata

3DMOD2

PROCload3Ddata(object)

Loads a particular three-dimensional object from a set of four supplied objects, into the program ready for manipulation and display. The object name must be in capitals and enclosed in double quotes. The aim here is to supply a set of simple objects that users can practice on and build up an ability to manipulate, before defining their own objects. The objects are:

RECTSOLID (rectangular solid)
PRISM
OCTASOLID (octagonal solid)
CHURCH

Capital letters must be used for object names.

PROCmodcircle

2DMOD4

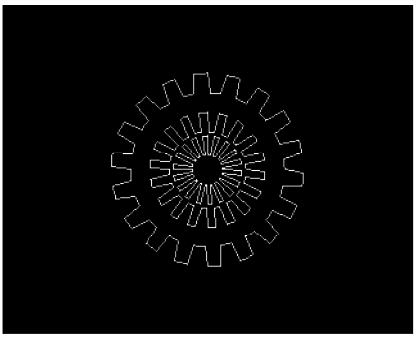
PROCmodcircle(x,y,radius,depth)

Draws a circle of a given radius at centre (x,y). The circumference of the circle is modulated by a square wave interference whose amplitude is given by 'depth'.

examples

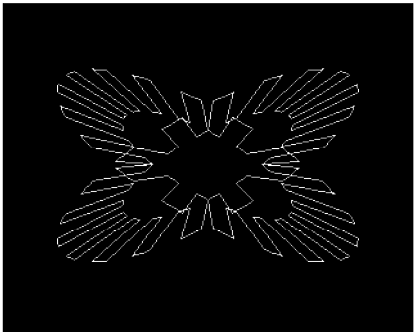
3 concentric motifs generated by a FOR loop

```
10 scale=500
20 FOR circle=1 TO 3
30   scale=scale*0.6
40   PROCmodcircle(640,512,scale,60)
50 NEXT circle
```



This is a good example of the way in which motifs can be transformed into a form that is completely different

```
10 PROChshear(1.2)
20 PROCmodcircle(640,512,300,130)
30 PROCreflecty
40 PROCmodcircle(640,512,300,130)
```



PROCmotif3D

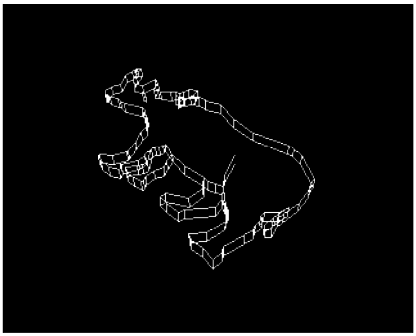
3DMOD1

PROCmotif3D(thick)

Creates a 3D model of a motif parallel to the xy plane. The z thickness of the motif is 'thick'.

example

```
10 PROCload("RHINO")
20 PROCviewpoint(1300,-45,50)
30 PROCmotif3D(30)
```



2DMOD3 2DMOD4

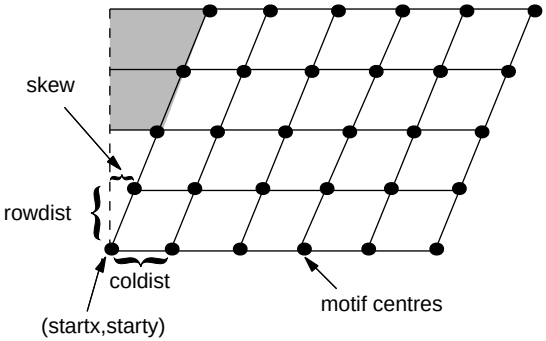
PROCnet(noofcols,noofrows,coldist,rowdist,startx,starty,skew,scale)
PROCcharnet(nootcols,noofrows,coldist,rowdist,startx,starty,skew)

PROCnet supplies the (x,y) coordinates to PROCnetmotif(x,y,scale).

(PROCcharnet initially draws a cross instead of a motif and can be used for planning a network).

The parameters are:

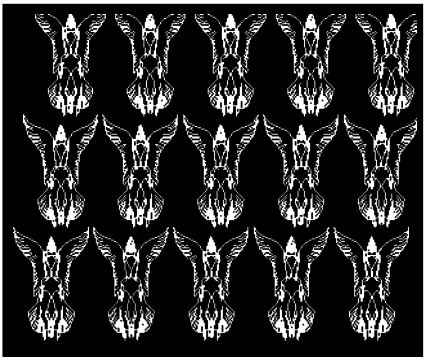
noofcols:	the number of columns in the net
noofrows:	the number of rows in the net
coldist rowdist startx starty skew	} see diagram
scale:	scaling factor for the motif



The shaded area is automatically filled with motfis if necessary. This procedure will generate square nets, rectangular nets and parallelogram nets (both rhombic and rectangular).

example

```
    10 PROCnet(5,3,250,350,100,100,30,.26)
    20 END
  1000 DEF PROCnetmotif(x,y,scale)
  1010   PROCinitialise
  1020   PROCload("GULL")
  1030   PROCstretch(2,90)
  1040   PROCdrawandscale(x,y,scale)
  1050   PROCreflecty
  1060   PROCdrawandscale(x,y,scale)
  1070 ENDPROC
```



Note the positional adjustment or translation away from the net centre at line 1050. In general idiosyncratic adjustments like this have to be made when using motifs. This is because the 'centre' of a motif changes depending on the motif. More details on the use of PROCnet are given in the Introductions to 2DMOD3 and 2DMOD4.

PROCobject3D

3DMOD2

PROCobject3D

Displays the current object in the data structure using either the default viewpoint or a user set viewpoint. Back surface elimination is incorporated into the procedure. Programs will thus have the following form:

```
PROCsetdatastructure(noofsurfaces,noofvertices)
PROCload3Ddata(object)
.
.
.
3D transforms
.
.
.
PROCobject3D
```

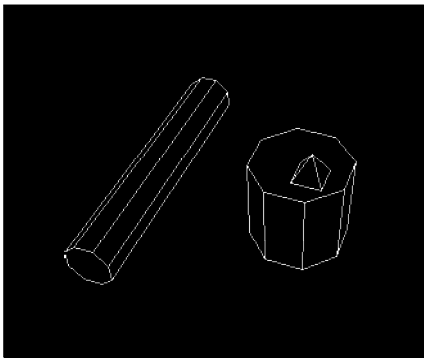
for a single object scene. For a multi-object scene, this sequence is repeated together with PROCinitialise (if required).

examples

A three object scene. Note that even although the octagonal solid is used twice, it must be re-loaded. This is because the back surface elimination routines makes alterations to the current state of the object data structure.

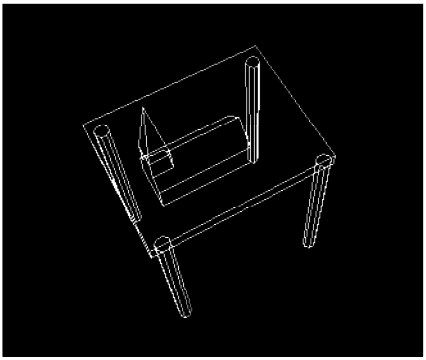
```
10 PROCsetdatastructures(10,16)
20 PROCviewpoint(3000,20,45)
30 PROCload3Ddata("PRISM")
```

```
 40 PROCuniscale(0.3)
 50 PROCtranslate(200,200,400)
 60 PROCobject3D
 70 PROCinitialise
 80 PROCload3Ddata("OCTASOLID")
 90 PROCobject3D
100 PROCinitialise
110 PROCload3Ddata("OCTASOLID")
120 PROCrotatey(90,0,0)
130 PROCscale(3,.5,.5)
```



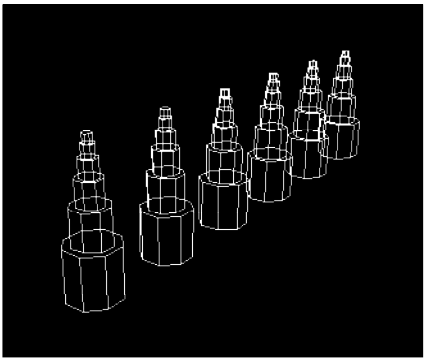
A six object scene. The four 'legs' are set up within a FOR loop (the values in PROCsetdatastructure are for the CHURCH)

```
 10 PROCsetdatastructures(20,20)
 20 PROCviewpoint(5000,60,30)
 30 FOR leg=1 TO 4
 40   PROCinitialise
 50   PROCload3Ddata("OCTASOLID")
 60   PROCscale(.2,.2,3)
 70   IF leg=2 THEN PROCtranslate(0,800,0)
 80   IF leg=3 THEN PROCtranslate(1000,0,0)
 90   IF leg=4 THEN PROCtranslate(1000,800,0)
100   PROCobject3D
110 NEXT leg
120 PROCinitialise
130 PROCload3Ddata("RECTSOLID")
140 PROCscale(2.8,2.2,.2)
150 PROCtranslate(0,0,1200)
160 PROCobject3D
170 PROCinitialise
180 PROCtranslate(400,400,1200)
190 PROCload3Ddata("CHURCH")
200 PROCobject3D
```



An example of a scene composed using a nested loop. The innermost loop generates a 'tower' by stacking octagonal solids. The outer loop repeats the 'tower' generation.

```
 10 PROCsetdatastructures(10,16)
 20 xdis=1000
 30 PROCviewpoint(5000,37,70)
 40 FOR tower=1 TO 6
 50   scale=1:zdis=0
 60   FOR hex=1 TO 6
 70     PROCinitialise
 80     PROCload3Ddata("OCTASOLID")
 90     PROCscale(scale,scale,scale)
100     PROCtranslate(xdis-200*scale,-200*scale,zdis-600)
110     zdis=zdis+400*scale
120     scale=scale*.7
130     PROCobject3D
140   NEXT hex
150   xdis=xdis-800
160 NEXT tower
```



PROCreflectx

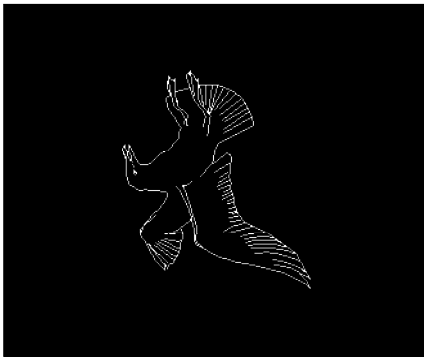
2DMOD1 2DMOD2 2DMOD3 2DMOD4

PROCreflectx

Produces a reflection of the current motif in memory about the horizontal axis.

example

```
10 PROCload("GULL")
20 PROCreflectx
30 PROCdrawandscale(640,512,1)
```



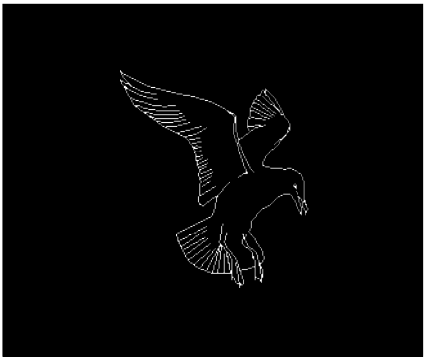
PROCreflecty

2DMOD1 2DMOD2 2DMOD3 2DMOD4

Produces a reflection of the current motif in memory about the vertical axis.

example

```
10 PROCload("GULL")
20 PROCreflecty
30 PROCdrawandscale(640,512,1)
```



PROCrotate

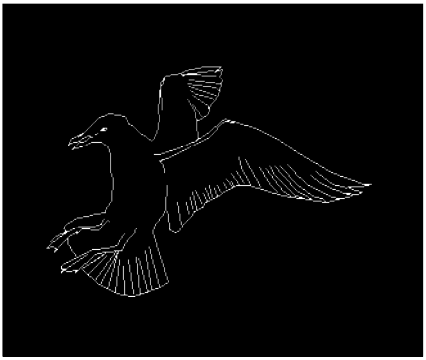
2DMOD1 2DMOD2 2DMOD3 2DMOD4

PROCrotate(angle)

Rotates the current motif in memory about its centre by the amount specified by 'angle'. If 'angle' is positive the rotation is anti-clockwise: if it is negative the rotation is clockwise.

examples

```
10 PROCload("GULL")
20 PROCrotate(-45)
30 PROCdrawandscale(500,512,1.3)
```



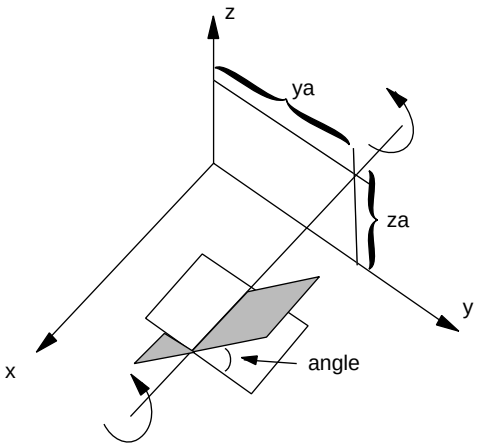
```
10 PROCload("GULL")
20 PROCrotate(-45)
30 PROCdrawandscale(500,512,1.3)
```

PROCrotatex

3DMOD1 3DMOD2

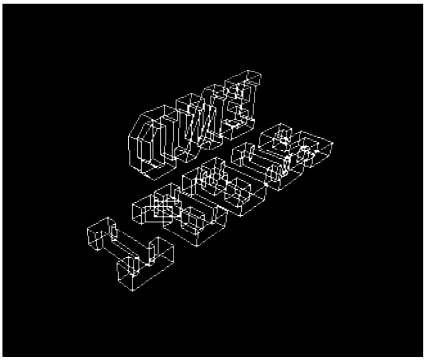
PROCrotatex(angle,ya,xa)

Rotates a three-dimensional object about a line parallel to the x axis. 'angle' is the amount of the rotation, (ya,za) gives the positipn of the line about which the rotation takes place.



example

```
10 PROCloadalpha("ALPHA1")
20 PROCsetorigin(500,350)
30 PROCviewpoint(1300,45,55)
40 PROCtranslate(-600,0,0)
50 PROCtext3D("START",50,50)
60 PROCtranslate(0,0,100)
70 PROCrotatex(90,0,0)
80 PROCtext3D("END",50,50)
```

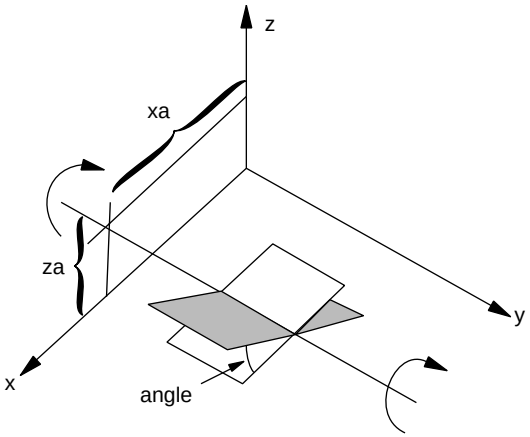


PROCrotatey

3DMOD1 3DMOD2

PROCrotatey(angle,xa,za)

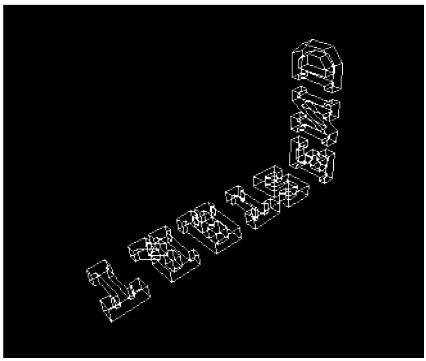
Rotates a three-dimensional object about a line parallel to the y axis. 'angle' is the amount of the rotation, (xa,za) gives the position of the line about which the rotation takes place.



example

```
10 PROCloadalpha("ALPHA1")
20 PROCsetorigin(600,500)
30 PROCviewpoint(2000,45,55)
40 PROCtext3D("START",50,50)
```

```
50 PROCtranslate(0,0,100)
60 PROCrotatey(90,0,0)
70 PROCtext3D("END",50,50)
```

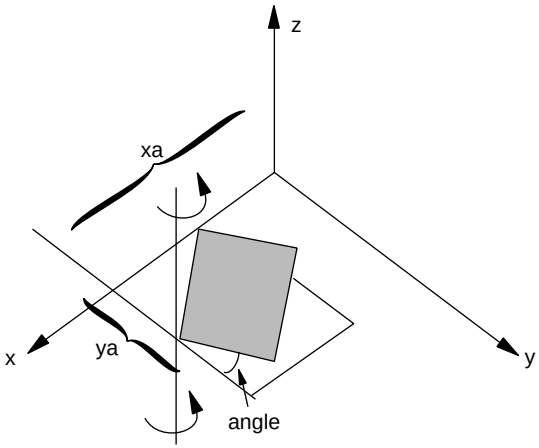


PROCrotatez

3DMOD1 3DMOD2

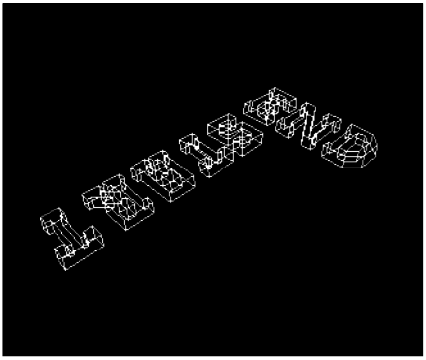
PROCrotatez(angle,xa,ya)

Rotates a three-dimensional object about a line parallel to the z axis. 'angle' is the amount of the rotation, (xa, ya) gives the position of the line about which the rotation takes place.



example

```
10 PROCloadalpha("ALPHA1")
20 PROCviewpoint(2000,45,55)
30 PROCtext3D("START",50,50)
40 PROCrotatez(90,0,0)
50 PROCtranslate(-100,0,0)
60 PROCtext3D("END",50,50)
```



PROCsavescreen

2DMOD1 2DMOD3 2DMOD4 3DMOD1 3DMOD2

PROCsavescreen

Causes a binary dump of the screen memory contents to be saved in a file called SCREEN. This file can be rapidly reloaded later by running a simple programme like

```
10 MODE 4
20 *LOAD SCREEN
```

or by using PROCloadscreen, in another GRAPHITO program.

On disc the SCREEN file can of course be renamed and a set of such files might be useful for rapid generation of screen images during a lecture or demonstration. These screen dumps will also provide an interface with your printer if you have one. Because of variations between different graphics printers, it was not possible to include a printer dump routine within the modules. However, if you have a screen dump routine for your printer, a binary screen dump produced by Graphito can be loaded and printed. We have provided one such screen dump routine for one of the most widely available printers, the EPSON MX-80. To run this:

```
LOAD "EPSONPR"
```

and RUN it with a SCREEN file on screen.

PROCscale

2DMOD1 2DMOD2 2DMOD3 2DMOD4

PROCscale(scale)

Magnifies or reduces the current motif in memory by a factor given by 'scale'. Scaling can also be carried out using the scale parameter in PROCdrawandscale.

```
PROGscale(2)
PROCdrawandscaIe(xo,yo,1)
```

is exactly equivalent to

```
PROCdrawandscaie(xo,yo,2)
```

PROCscale is used on its own when combinations of two-dimensional transforms are used. Note that consecutive calls of PROCscale multiply.

```
PROCscale(3)
PROCscale(3)
```

is exactly equivalent to

```
PROCscale(9)
```

PROCdrawandscale does not affect the current motif scaling. The scale parameter of PROCdrawandscale only affects the size of the motif drawn on the screen.

PROCscale(sx,sy,sz)

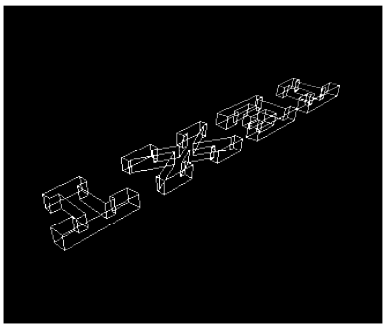
3DMOD1 3DMOD2

Scales a three-dimensional object or motif by a factor of sx in the x direction, sy in the y direction and sz in the z direcfion.

examples

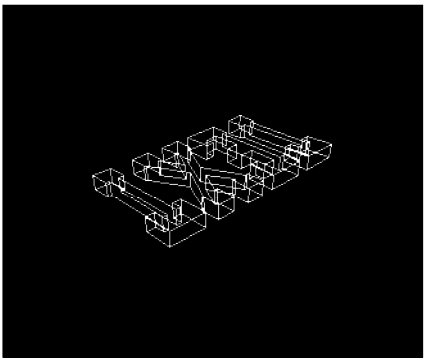
Scaling or stretching in the x direction only means setting sy and sz to 1

```
10 PROCloadalpha("ALPHA1")
20 PROCviewpoint(2000,45,55)
30 PROCtext3D("START",50,50)
40 PROCrotatez(90,0,0)
50 PROCtranslate(-100,0,0)
60 PROCtext3D("END",50,50)
```



Scaling or stretching in the y direction only means setting sx and sz to 1. Scaling in the z direction is equivalent to increasing 'thick' in PROCtext3D

```
10 PROCload("ALPHA1")
20 PROCsetorigin(500,350)
30 PROCviewpoint(1300,45,55)
40 PROCscale(1,2,1)
50 PROCtranslate(-600,-400,0)
60 PROCtext3D("TEXT",50,50)
```



PROCsetdatastructures

3DMOD2

PROCsetdatastmctures(nootsurtaces,nootvertices)

Sets up a data structure that will contain the three-dimensional object that is to be manipulated and displayed. 'noofsurfaces' is the number of surfaces that the object possesses and 'noofvertices' is the number of vertices that the object possesses. For example a cube contains six surfaces and eight vertices.

PROCsetdatastructures must be called for every object that is used in a multi-object scene.

PROCsetorigin(xo,ye)

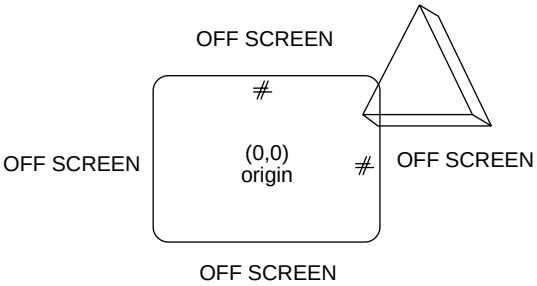
3DMOD1 3DMOD2

PROCsetorigin(xo,yo)

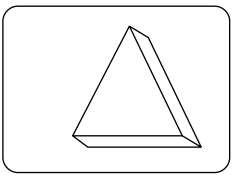
Overrides the default origin and sets the origin at position (xo,yo). It can be used in conjunction with PROCtraceon to locate or make visible information that has been plotted off the screen.

example

Although some information is visible on screen PROCtraceon indicates that plotting is taking place off screen. The indicators show that this is in the area north east of the origin. PROCsetorigin should be called with positive xo and yo



A call of PROCsetorigin (700,700) has made the bottom left hand comer of the screen point (700,700). This is the same as 'moving' the screen as a 'window' in a north easterly direction



PROCsierpinski

2DMOD4

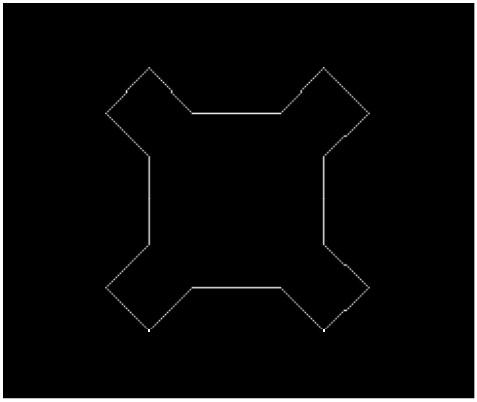
PROCsierpinski(order,x,y,scale)

Draws a Sierpinski curve of given 'order' centred on (x, y). A scale of 1 results in a curve occupying an area that is 512 screen units square.

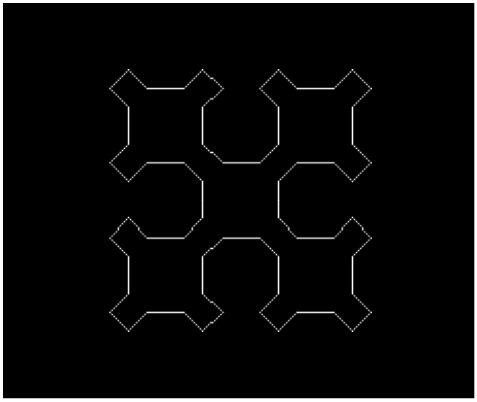
examples

Sierpinski curves of orders 1 to 4

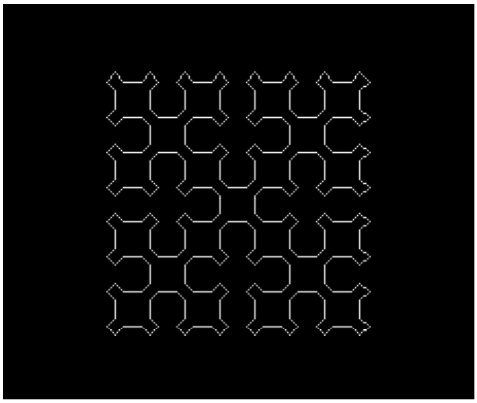
```
10 PROCsierpinski(1,640,512,1)
```



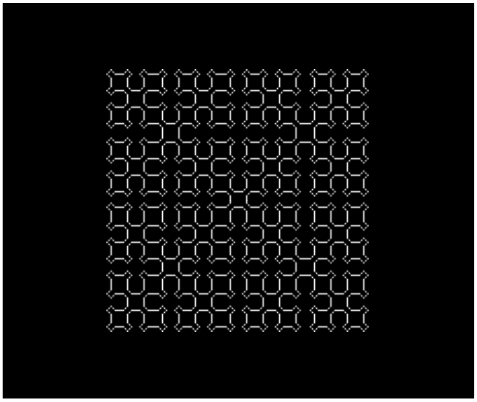
```
10 PROCsierpinski(2,640,512,1)
```



```
10 PROCsierpinski(3,640,512,1)
```



```
10 PROCsierpinski(4,640,512,1)
```



PROCsincircle

2DMOD4

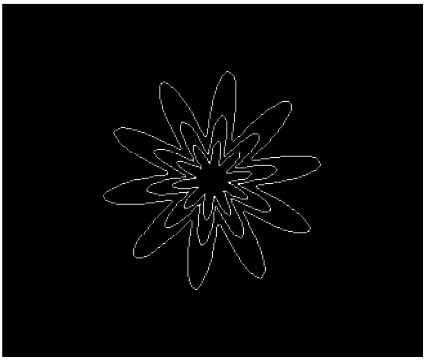
PROCsincircle(x,y,radius,depth)

Draws a circle of a given radius at centre (x,y). The circumference of the circle is modulated by a sine wave whose amplitude is given by 'depth'.

examples

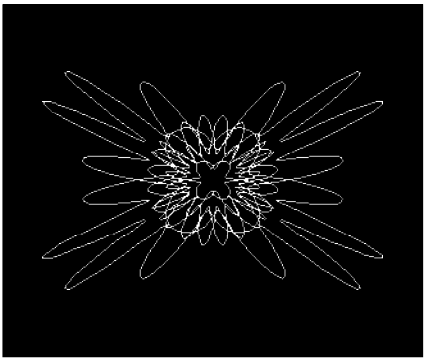
Three concentric 'sincircles' generated by a FOR loop

```
10 scale=400:depth=170
20 FOR i=1 TO 3
30   scale=scale*0.6:depth=depth*0.6
40   PROCsincircle(640,512,scale,depth)
50 NEXT i
```



Shear and reflection produce an explosion effect

```
10 scale=400:depth=170
20 FOR i=1 TO 3
30   scale=scale*0.6:depth=depth*0.6
40   PROChshear(1.2)
50   PROCsincircle(640,512,scale,depth)
60   PROCreflecty
70   PROCsincircle(640,512,scale,depth)
80 NEXT i
```



PROCsnowflake

2DMOD4

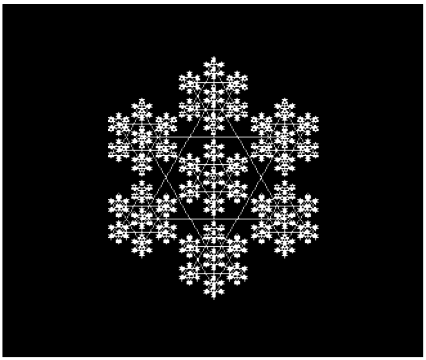
PROCsnowflake(x,y,scale,resolution,reduction factor)

Draws a snowflake or 'Koch flake' pattern centred at (x,y). A scale of 1 results in a snowflake that is about 700 screen units across. 'resolution' determines the amount of detail in the flake. 'reduction' determines the ratio of the size of the smaller flakes that make up the pattern to the size of the outer flake.

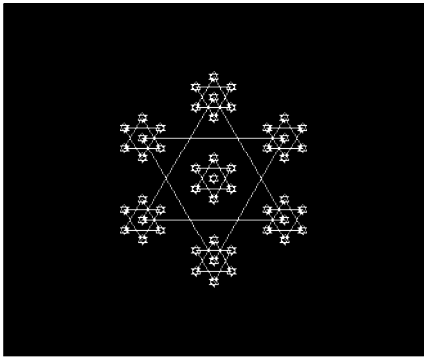
See also PROCcolouredflake.

examples

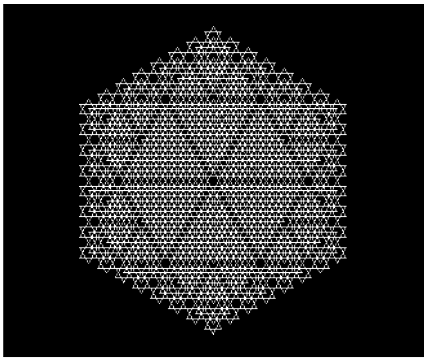
```
10 PROCsnowflake(640,512,1,8,0.33)
```



```
10 PROCsnowflake(640,512,1,8,0.25)
```



```
30 PROCsnowflake(640,512,1,30,0.5)
```



PROCspheregrid

3DMOD1

See **PROCdecorsphere**

PROCspiral

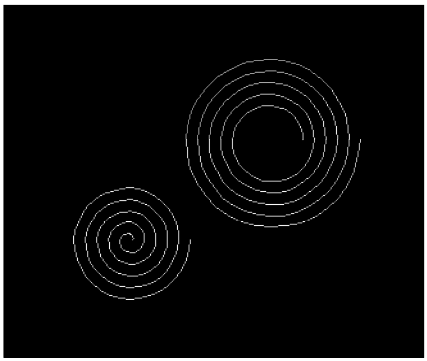
2DMOD4

PROGspiral(x,y,radius)

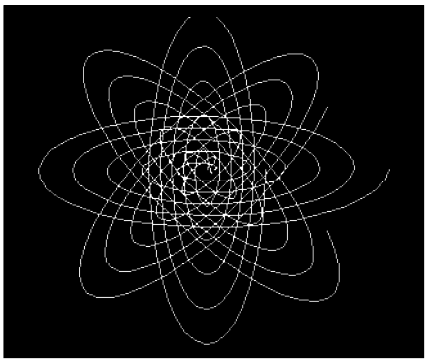
Draws a 'linear' spiral at centre (x,y) having radius set by the parameter 'radius'.

examples

```
10 PROCspiral(640,512,100)
20 PROCspiral(200,200,10)
```



```
10 FOR theta=45 TO 360 STEP 45
20   PROCinitialise
30   PROCstretch(3,theta)
40   PROCspiral(640,550,10)
50 NEXT
```



PROCsquares

2DMOD4

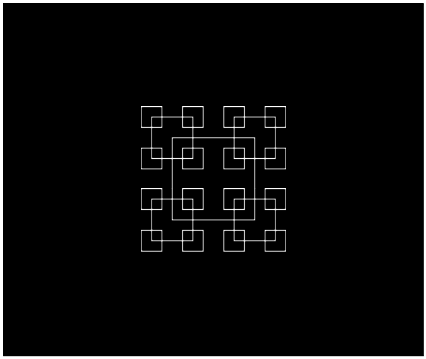
PROCsquares(x,y,scale,resolution)

Draws a pattern of recursive squares centred on (x, y). A scale of 1 results in a pattern occupying an area that is 512 screen units square.

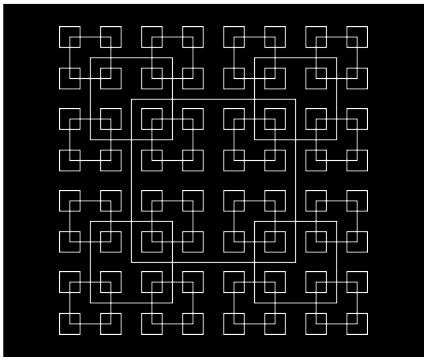
'resolution' determines the level of detail that will be included in the pattern. Its value should be the approximate size of the smallest squares in the pattern.

examples

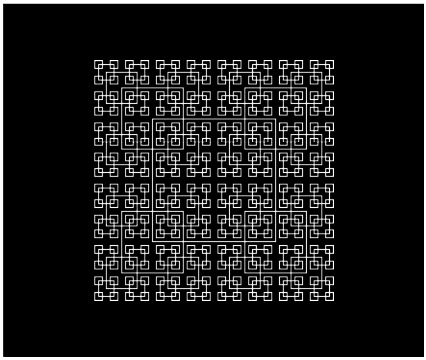
```
10 PROCsquares(640,512,1,32)
```



```
10 PROCsquares(640,512,2,32)
```



```
10 PROCsquares(640,512,1.5,8)
```



PROCstretch

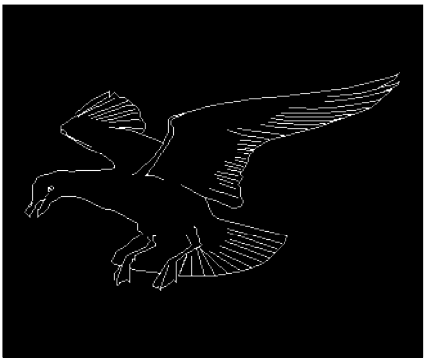
2DMOD1 2DMOD2 2DMOD3 2DMOD4

PROCstretch(scale,theta)

Stretches the current motif in memory along a direction given by theta

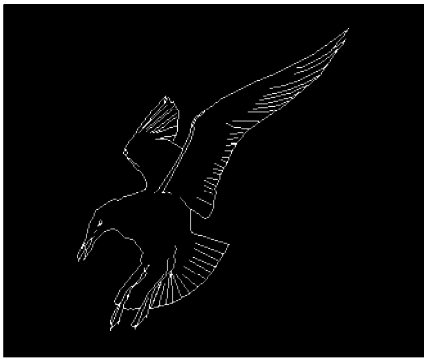
examples

```
10 PROCload("GULL")
20 PROCstretch(2,0)
30 PROCdrawandscale(450,512,1)
```

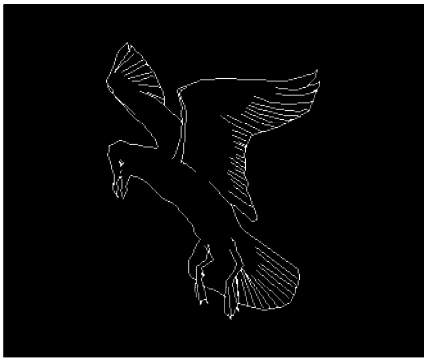


```
10 PROCload("GULL")
20 PROCstretch(1.5,90)
30 PROCdrawandscale(450,410,1)
```

```
10 PROCload("GULL")
20 PROCstretch(1.7,45)
30 PROCdrawandscale(500,390,.9)
```



```
10 PROCload("GULL")
20 PROCstretch(1.7,-45)
30 PROCdrawandscale(640,512,1)
```



PROCtextoncube

3DMOD1

PROCtextoncube(string$,size,face,gap,where,scale)

Draws text on a selected face of a cube. The parameters are:

string$	the text to be drawn (this must be in capitals and enclosed by double quotes)
size	the size of the cube that the text is drawn on
face	the face on which the text appears (see PROCdecorcube)
gap	the gap between letters

where the position of the text on the face of the cube
1 means draw at the 'top' edge of the face
2 means draw in the centre of the face
3 means draw at the 'bottom' edge of the face

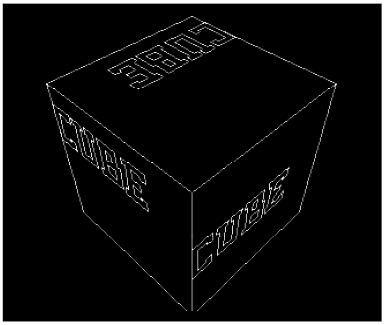
scale the size of the letters

Note that the 'size' of the cube must accommodate the length of the text. Scaling above 1 is problematic and may cause displacements with respect to the cube sides that take the text out of the cube. 'Scale' should be less than or equal to 1.

examples

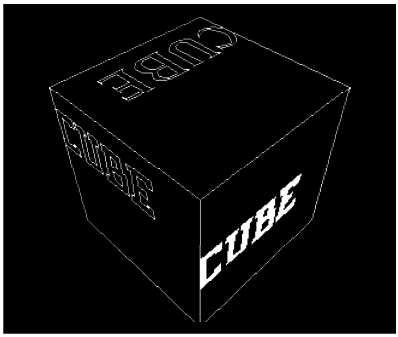
Same text on each face in different positions

```
10 PROCviewpoint(2800,45,45)
20 PROCtranslate(0,0,200)
30 PROCloadalpha("ALPHA1")
40 PROCtextoncube("CUBE",800,1,20,1,1)
50 PROCtextoncube("CUBE",800,2,20,2,1)
60 PROCtextoncube("CUBE",800,3,20,3,1)
```



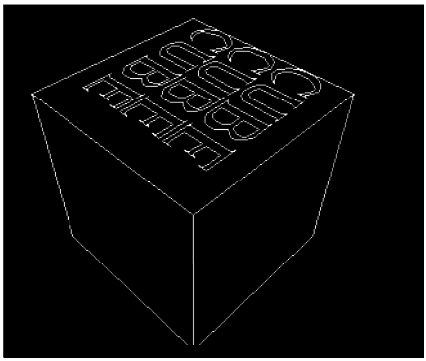
As previous example but using different alphabet styles

```
10 PROCviewpoint(2800,45,45)
20 PROCtranslate(0,0,200)
30 PROCloadalpha("ALPHA1")
40 PROCtextoncube("CUBE",800,1,20,1,1)
50 PROCloadalpha("ALPHA2")
60 PROCtextoncube("CUBE",800,2,20,2,1)
70 PROCloadalpha("ALPHA3")
80 PROCtextoncube("CUBE",800,3,20,3,1)
```



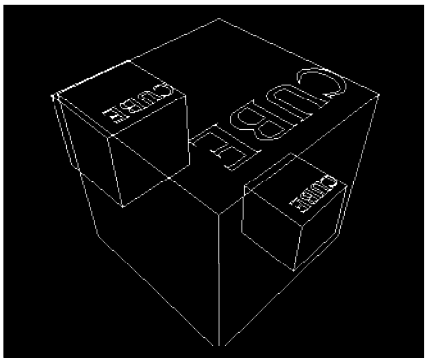
Same text at different positions on the same face

```
 10 PROCviewpoint(2800,45,45)
 20 PROCtranslate(0,0,200)
 30 PROCloadalpha("ALPHA3")
 40 PROCtextoncube("CUBE",800,3,20,1,1)
 50 PROCtextoncube("CUBE",800,3,20,2,1)
 60 PROCtextoncube("CUBE",800,3,20,3,1)
```



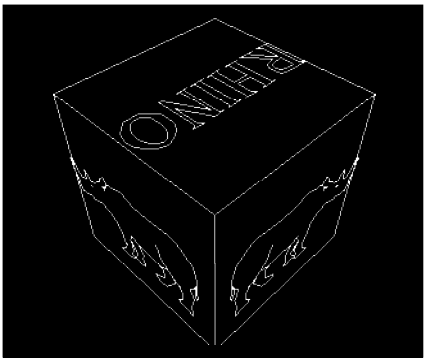
Three cubes involving translation and scale

```
 10 PROCviewpoint(2800,45,45)
 20 PROCtranslate(0,0,200)
 30 PROCloadalpha("ALPHA3")
 40 PROCtextoncube("CUBE",800,3,20,1,1)
 50 PROCuniscale(.5)
 60 PROCtranslate(-400,200,-400)
 70 PROCtextoncube("CUBE",800,3,20,1,1)
 80 PROCinitialise
 90 PROCuniscale(.5)
100 PROCtranslate(300,-300,400)
110 PROCtextoncube("CUBE",800,3,20,1,1)
```



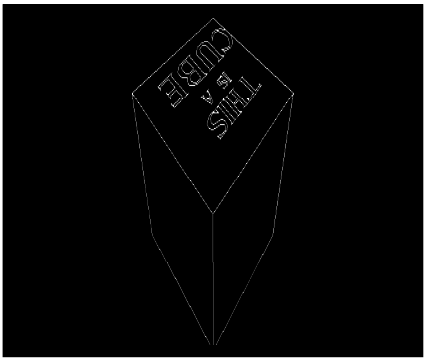
Mixing text and motifs on cube

```
10 PROCviewpoint(2800,45,45)
20 PROCtranslate(0,0,200)
30 PROCload("RHINO")
40 PROCloadalpha("ALPHA3")
50 PROCtextoncube("RHINO",800,3,20,2,1)
60 PROCdecorcube(800,1)
70 PROCdecorcube(800,2)
```



Different scaled text on the same face

```
10 PROCviewpoint(2800,45,45)
20 PROCtranslate(0,0,200)
30 PROCloadalpha("ALPHA3")
40 PROCtextoncube("THIS",800,3,20,1,.7)
50 PROCtextoncube("IS A",800,3,20,2,.4)
60 PROCtextoncube("CUBE",800,3,20,3,1)
```



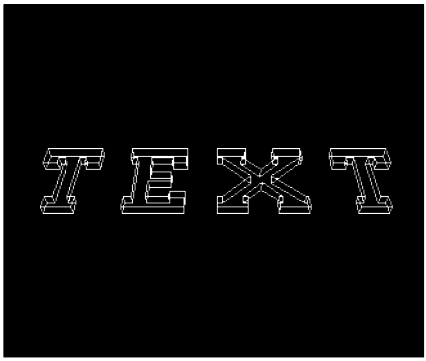
PROCtext3D('string',thick,gap)

Creates a string of characters parallel to the xy plane. The z thickness of the characters is 'thick' and the intercharacter spacing is 'gap'.

examples

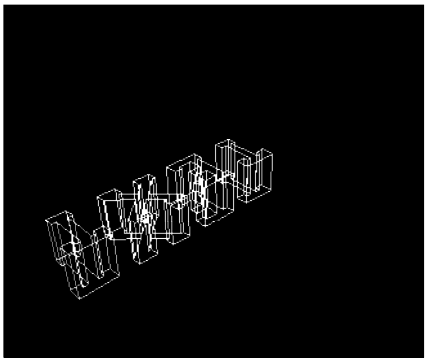
Thin characters viewed from the default viewpoint

```
10 PROCloadalpha("ALPHA1")
20 PROCtranslate(-400,0,0)
30 PROCtext3D("TEXT",20,70)
```



Thick characters viewed from a user-defined viewpoint

```
10 PROCloadalpha("ALPHA1")
20 PROCviewpoint(1900,45,60)
30 PROCtext3D("TEXT",150,70)
```

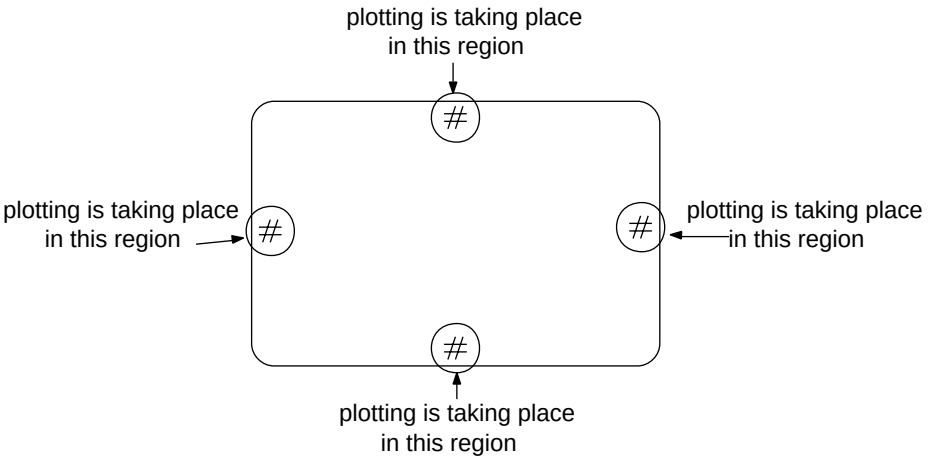


PROCtraceon

3DMOD1 3DMOD2

PROCtraceon

Switches on a trace facility that indicates when plotting is taking place off the screen. In three-dimensional work it is very easy to achieve a blank screen. For example, specifying a viewpoint that is too close or using a 'wrong' combination of three-dimensional transforms can easily produce a set of screen coordinates that do not fall within the physical limits of the screen (0-1279 horizontally by 0-1023 vertically). If PROCtraceon is used, an indication (#) will appear whenever plotting is taking place off the screen. The indicator appears at the top, bottom, left hand or right hand edge of the screen.



If one of the indicators appears then either the program should be corrected or the origin (normally set to the centre of the screen) can be moved so that it 'windows' the area where plotting is taking place. This is done by using PROCsetorigin.

PROCtraceon increases plotting time and should only be used when necessary.

PROCtranslate

3DMOD1 3DMOD2

PROCtranslate(tx,tr,tz)

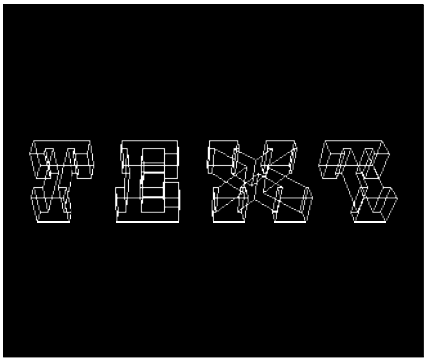
Moves a text or motif model in three-dimensional space by amounts tx, ty and tz. When a three-dimensional scene is composed each object in the scene can be placed where

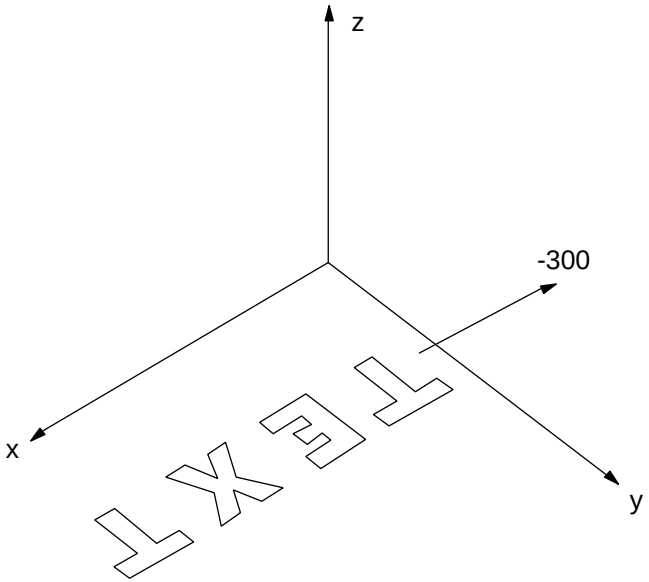
desired using this procedure. The entire scene can then be viewed. The scene is composed in three-dimensional space and mapped into two- dimensional space. The final position and size of each object depends on its positioning in three-dimensional space. Text and motif models are created in the xy plane. In most cases text should be centred about the y axis by using a negative value for tx.

examples

Created text is shifted to be symmetrical about the y axis

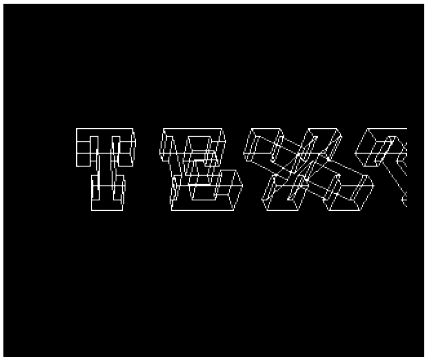
```
10 PROCloadalpha("ALPHA1")
20 PROCtranslate(-300,0,0)
30 PROCsetorigin(500,500)
40 PROCtext3D("TEXT",70,70)
```





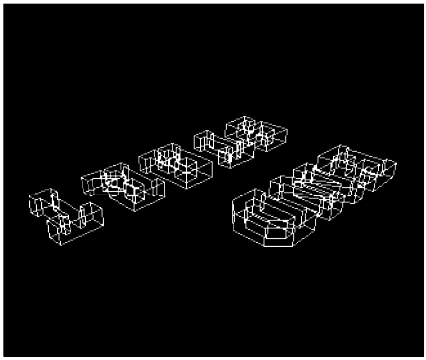
The same text not 'symmetricized'. The text is now to the right of the viewpoint

```
10 PROCloadalpha("ALPHA1")
30 PROCsetorigin(250,500)
40 PROCtext3D("TEXT",70,70)
```



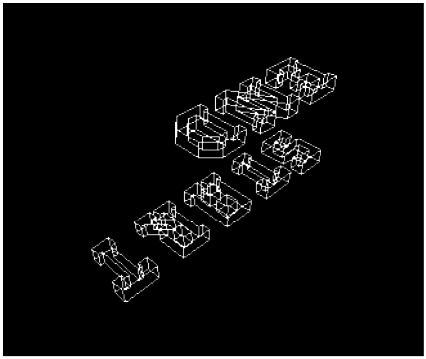
Moving in the xy plane

```
10 PROCloadalpha("ALPHA1")
20 PROCsetorigin(500,500)
30 PROCviewpoint(1300,45,60)
40 PROCtranslate(-600,-300,0)
50 PROCtext3D("START",50,70)
60 PROCtranslate(0,400,0)
70 PROCtext3D("END",50,70)
```



Moving up in the xz plane

```
10 PROCloadalpha("ALPHA1")
20 PROCsetorigin(500,350)
30 PROCviewpoint(1300,45,45)
40 PROCtranslate(-700,-300,-200)
50 PROCtext3D("START",50,70)
60 PROCtranslate(0,0,300)
70 PROCtext3D("END",50,70)
```

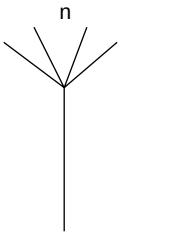


PROCtree

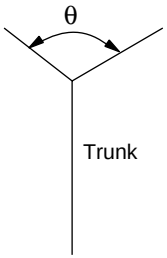
2DMOD4

PROGtree(x,y,scale,rand,branches,branchspread,reductiontactor,height, branchcol,leafcol)

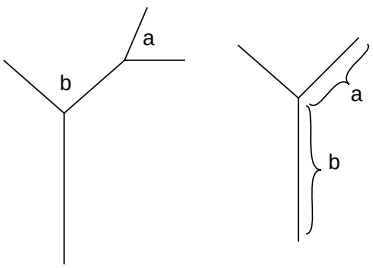
Draws a tree patterns with its base at (x,y). A scale of 1 results in a tree with a trunk that is 256 screen units long. 'branchcol' is the colour number to be used for branches and 'leafcol' is the colour number to be used for leaves (at the tips of the topmost branches). 'rand' is TRUE for a random tree and FALSE for a non-random tree. The remaining parameters have the following significance:



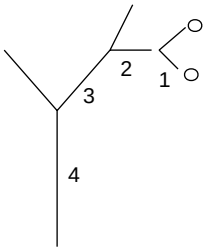
branches = n



branchspread = θ

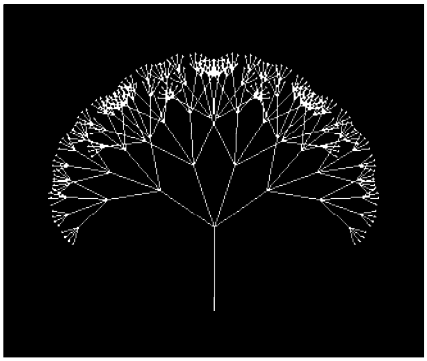


reductionfactor = a/b



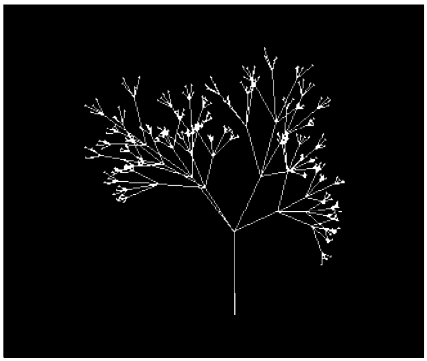
hieght = number of branches from root to leaf

```
10 PROCtree(640,100,1,FALSE,4,110,0.8,5,1,1)
```



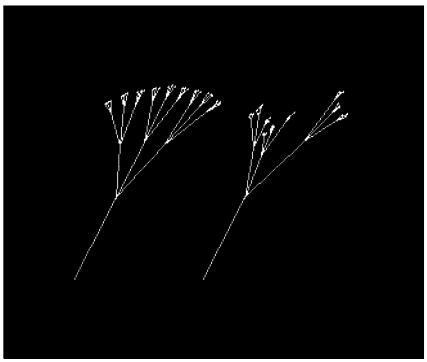
The same parameters as before, but the tree now includes random variations in shape

```
10 PROCtree(640,100,1,TRUE,4,110,0.8,5,1,1)
```



Applying transformations to trees

```
10 PROChshear(0.5)
20 PROCtree(400,100,1,FALSE,3,45,0.7,4,1,1)
30 PROCtree(800,100,1,TRUE,3,45,0.7,4,1,1)
```



PROCuniscale

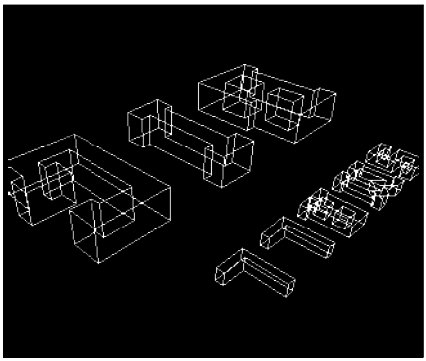
3DMOD1 3DMOD2

PROCuniscale(scale)

Magnifies or reduces uniformly in the x, y and z directions by a factor given by 'scale'.

example

```
 5 PROCloadalpha("ALPHA1")
10 PROCsetorigin(500,350)
20 PROCviewpoint(1300,45,55)
30 PROCuniscale(2)
40 PROCtranslate(-700,-300,0)
50 PROCtext3D("BIG",50,100)
60 PROCinitialise
70 PROCuniscale(0.7)
80 PROCtranslate(-600,300,0)
90 PROCtext3D("SMALL",50,100)
```



Note that PROCtranslate is applied after PROCuniscaie each time. If it is applied before, the translation will also be scaled

PROCviewpoint

3DMOD1 3DMOD2

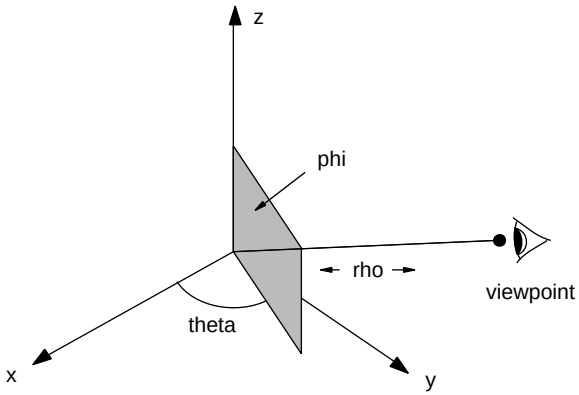
PROCviewpoint(rho,theta,phi)

Specifies a viewpoint that overrides the default viewpoint A viewpoint is a point in three-dimensional space from which the scene is viewed. The position of the viewpoint is used to calculate a two-dimensional mapping of the three-dimensional scene and this is displayed on the screen. The position of a viewpoint is best specified by a distance (rho) and two angles theta and phi.

rho — is the distance from the viewpoint to the origin of the three-dimensional coordinate system. The larger rho is, the further is the viewpoint from the scene and the smaller the scene will appear.

theta — is the rotational angle that the viewpoint makes with the x axis. Changing theta and keeping the other two parameters fixed moves the viewpoint around the scene at constant height and distance.

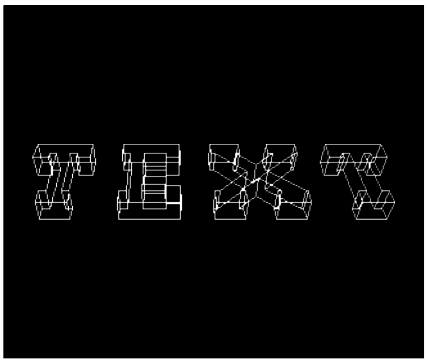
phi — is the elevation angle. When phi is reduced to zero the viewpoint is directly above the scene.



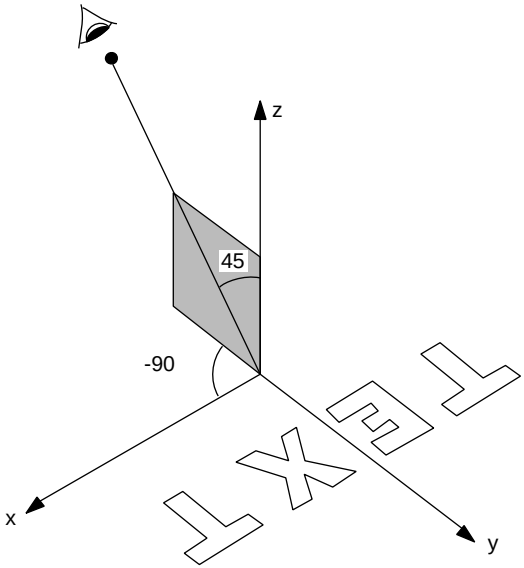
examples

Default values - no call to PROCviewpoint

```
20 PROCloadalpha("ALPHA1")
30 PROCtranslate(-300,0,0)
40 PROCsetorigin(500,500)
50 PROCtext3D("TEXT",50,70)
```

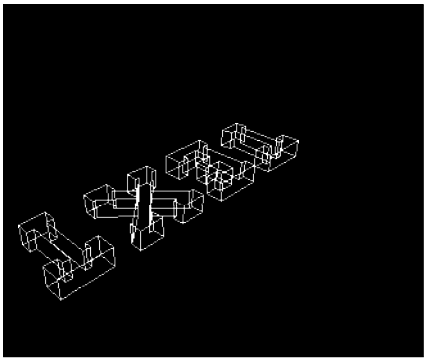


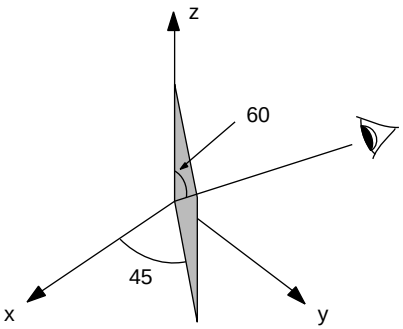
The default values are rho = 1000, theta = -90, phi = 45. The scene in the examples is made up of 'thick' letters in the xy plane



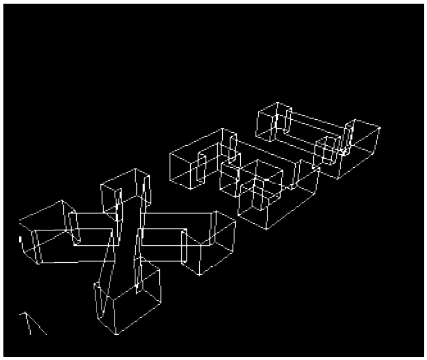
Moving in along the viewpoint vector

```
10 PROCviewpoint(1300,45,60)
```

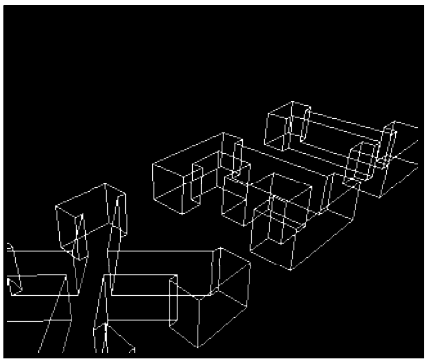




```
10 PROCviewpoint(800,45,60)
```

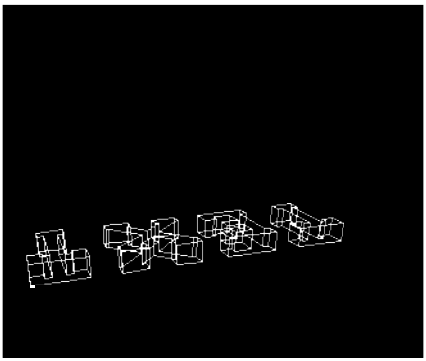


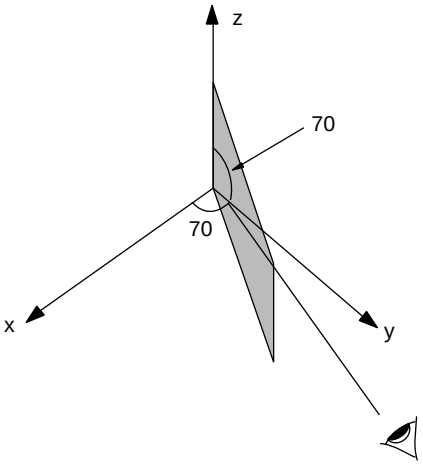
```
10 PROCviewpoint(600,45,60)
```



High phi gives a shallow view

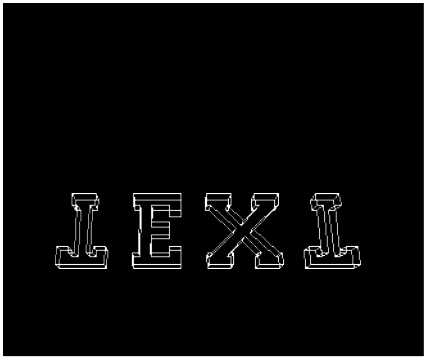
```
10 PROCviewpoint(1300,70,70)
```





Looking up from underneath

```
10 PROCviewpoint(1300,-90,-160)
```



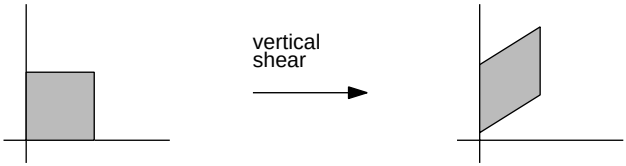
PROCvshear

2DMOD1 2DMOD2 2DMOD3 2DMOD4

PROCvshear(amount)

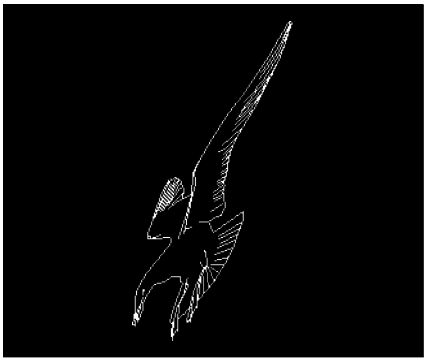
Shears the current motif in memory in the vertical direction. Shearing is an operation that turns a square, for example, into a parallelogram.

See also PROChshear.



example

```
10 PROCload("GULL")
20 PROCvshear(1.2)
30 PROCdrawandscale(640,300,1)
```



PROCvtext

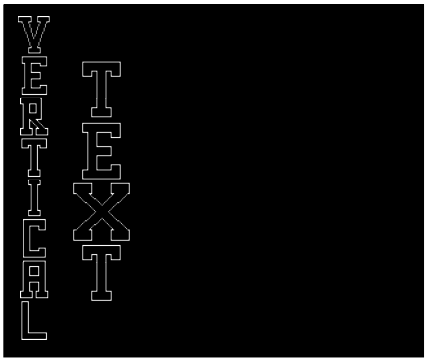
2DMOD1 2DMOD3 2DMOD4

PROCvtext(string$,xstart,ystart,scale,gap)

Draws a string of characters, vertically starting from position (xstart, ystart).
'gap' and 'scale' are as in PROChtext.

example

```
10 PROCloadalpha("ALPHA1")
20 PROCvtext("VERTICAL",70,1100,0.6,20)
30 PROCvtext("TEXT",250,900,0.9,20)
```



PROCw-curve

2DMOD4

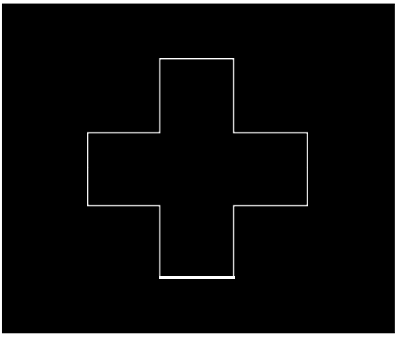
PROCw-curve(order,x,y,scale)

Draws a w-curve of a given 'order' centred on (x, y). A scale of 1 results in a curve occupying an area 512 screen units square.

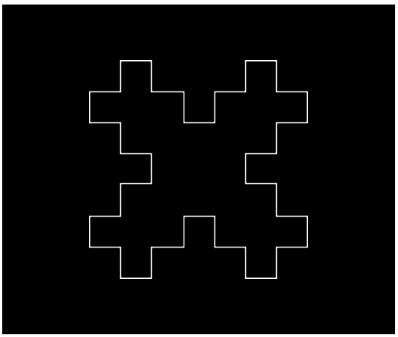
examples

w-curves of orders 1 to 4

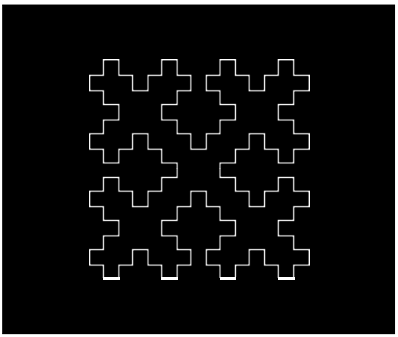
```
10 PROCw_curve(1,640,512,1)
```



```
10 PROCw_curve(2,640,512,1)
```



```
10 PROCw_curve(3,640,512,1)
```



```
10 PROCw_curve(4,640,512,1)
```