Computer software for use in the teaching of secondary school mathematics

PUBLISHER

Loughborough University of Technology

REPOSITORY RECORD

R. Johnson, Keith. 2021. “Computer Software for Use in the Teaching of Secondary School Mathematics”. Loughborough University. https://doi.org/10.26174/thesis.lboro.14716947.v1.

COMPUTER SOFTWARE FOR USE IN THE TEACHING OF SECONDARY SCHOOL MATHEMATICS

by

KEITH RICHARD JOHNSON, B.Sc.

A project submitted in partial fulfilment of
the requirements for the award of the degree
of M.Sc. in Mathematical Education of the
Loughborough University of Technology,
January 1984.

Supervisor: R. P. KNOTT, M.Sc., Ph.D.

Abstract

The project is concerned with software for the Sinclair Spectrum computer which I have written for use in teaching mathematics. It commences with a short introduction outlining the reason I undertook the study.

Chapter 1 considers a program on transformation geometry, which illustrates rotations, reflections, translations, enlargements, shears and stretches applied to any shape formed by joining points with a set of continuous straight lines. The coordinates of the points are input by the user, or the preset object (a triangle) may be chosen. Any combination of the transformations is allowed, and the transformation may be applied to the current image, the previous image or to the original object.

Chapter 2 contains three programs. The first illustrates relations between patterns of dots arranged as squares and triangles. The program on the Fibonacci Sequence illustrates how the sequence is generated, using the breeding patterns of rabbits. It also shows that the ratio of consecutive pairs of terms in the sequence approaches the golden ratio. The third program is concerned with prime numbers and will print the sieve of Eratosthenes, list the primes, test a number to see if it is prime, or output the N^{th} prime, when N is input.

The program in chapter 3 shows the locus of a point, P, under given conditions. The conditions, which are chosen by the user, are illustrated graphically and the locus is plotted.

All the programs in the chapters are described in two ways. Firstly, how they are controlled and what they do when executed, secondly, ways in which I have used the programs with various groups of pupils.

The listings for the programs, together with accompanying notes, are given in the appendices at the end of the project.

Acknowledgements

I should like to acknowledge those people who have assisted, directly and indirectly, in the production of this project.

Firstly, to my supervisor, Dr Knott, for his helpful suggestions concerning the presentation of the project and for the interest which he has shown in it.

Secondly, to my wife, without whose patience, understanding and encouragement the project would not have been completed.

To the pupils of Queen Elizabeth's Mercian School, Tamworth who have been 'guinea-pigs' for the software.

Finally, to Mr T. Jones of Queen Elizabeth's Mercian School who lent me his Sinclair printer when mine broke down.

Declaration

This project presents the results of my use of the computer in the classroom, and is entirely my own work. The software produced has been developed by me from my ideas on how to present the particular topics to pupils.

Contents

Introduction

During recent years the price of microcomputers has dropped so markedly that they are now commonplace in homes and schools. The potential use of the microcomputer in schools is enormous, but any use should be an improvement on traditional methods if it is to be of value. Many of the early attempts at computer assisted learning failed, in my opinion, in this respect. For example, programs which were meant to test a child's ability in a particular topic gave a score but failed to diagnose where the child needed help.

The programs which I have written in this project have exploited the graphics capability of the computer to create interest and to improve understanding in various aspects of mathematics. The topics chosen are some which I consider need to be demonstrated dynamically or which pupils find difficult using traditional methods. The demonstrations are intended to be teacher controlled but, as more computers become available for use by the children, the programs could be modified to make them pupil controlled, so that each child may progress at his own pace.

I decided to produce my own software on a Sinclair Spectrum for the following reasons. Firstly, the only computer available in the school where I work is a Research Machines RML 380Z, which can be used in few rooms because of transportation problems. The computer also needs to be booked at least a week in advance, which eliminates any spontaneous use. Even when more computers are available in the school, they will be housed in a computer room and priority will be given to Computer Studies classes. There will be little opportunity to use them regularly in mathematics lessons. I required a computer that was easily moved from room to room if necessary, which would be available for use at any time during mathematics lessons. I chose the Sinclair Spectrum because it was small, inexpensive and capable of high resolution graphics.

The reason for producing my own software was that I have found no secondary school mathematics software for the Spectrum. Software is available for other computers, including the 380Z, but none of those covering the topics of this project was considered satisfactory. All the programs on transformation geometry, for example, plotted the image without showing the transformation dynamically. One program for the BBC computer showing the locus of a point on a ladder which is slipping down a wall considered only the midpoint, and did not allow generalisation.

Since buying the Spectrum, I have used it in the classroom mainly for demonstration purposes, that is as an 'electronic blackboard'. The programs which I have produced fall into two categories; short programs which are mostly used as an introduction to a topic or for clarification and longer programs used for teaching a topic. The short programs were usually written in an evening for use in the next day's lesson, examples of which are Pythagoras' Theorem, the sine curve and coordinates. The programs contained in this project are the longer ones which have been developed over a period of time and are intended for use throughout the particular topic.

The listings of the programs and the diagrams in chapters one and three have been produced using a Sinclair printer, which were then copied on a photocopier. By this method, diagrams may be produced to issue to the pupils, if required.

In this study mention is also made of short programs, sometimes consisting of only a few lines, written during the lesson to help teach a particular concept. Little work has been done on using computer programming to teach mathematical concepts in this way, but with more pupils entering secondary school being able to write computer programs, this could be another worthwhile use of the computer, especially in introducing basic ideas of algebra.

Chapter One

One of the main criteria for using the computer in the teaching of mathematics is that the computer has an advantage over other methods of presentation.

Transformation Geometry, the subject of this chapter, is certainly a topic about which this is true. Diagrams can be produced much more accurately and quickly than on a blackboard. Of course the size of the display screen is limiting, but the ability to produce dynamic demonstrations of the transformations, together with the facility for repeating them, far outweighs this restriction.

The chapter is in two parts:

1. Explanation of the program's execution.
2. Examples of ways in which I have used the program, and how it might be used.

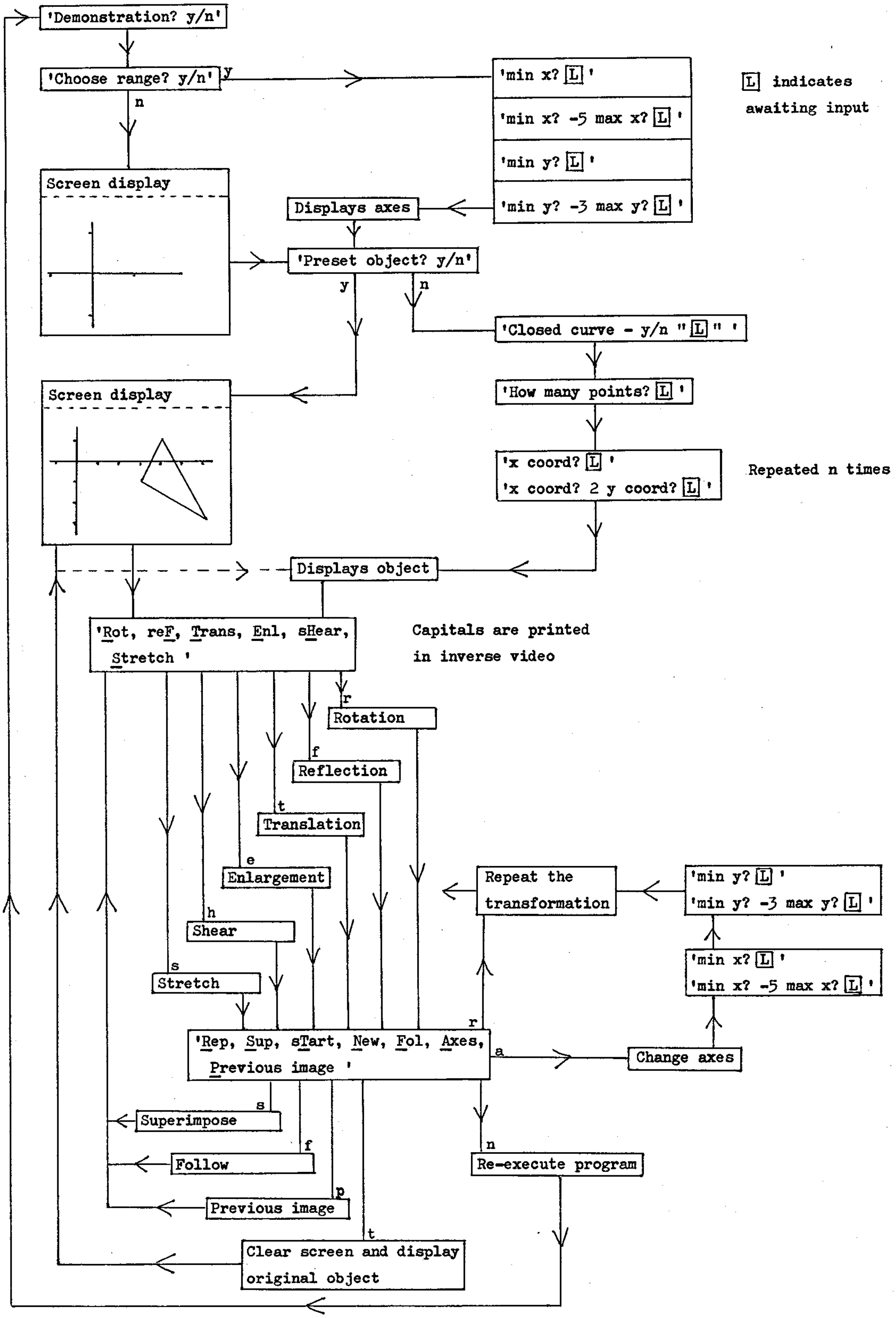
The program listing, with explanation, is given in appendix A.

The title of the program is 'transforms'.

The flow diagram on the following page shows how the program is executed and controlled using the various keys.

FLOW DIAGRAM TO SHOW THE EXECUTION OF 'transforms' PROGRAM

Messages which appear at the bottom of the screen are printed in quotes (').



L indicates awaiting input

Repeated n times

Capitals are printed in inverse video

The program is executed by pressing 'RUN' followed by 'ENTER', which will result in the screen being cleared.

All the messages are written on the bottom two lines of the screen and, as are the graphic displays, written white on a black background. I have found by experiment that this combination gives greater clarity and viewing distance. The use of different colours for objects and images was tried, but the Spectrum computer is unable to display more than two colours in any character square (8 pixels x 8 pixels). The tests resulted in parts of figures which overlapped others being changed in colour; this caused confusion. It was decided therefore to use just two colours.

Control of the program is achieved by a single press of a key - usually any key, but some instructions require specific keys to be pressed. When input is required, the data is typed and followed by pressing the 'ENTER' key.

Message - 'Demonstration? y/n'.

After certain of the transformations have been executed, there is a section of the subroutine which demonstrates the properties of that particular transformation. Pressing the 'y' key will result in the demonstration being given, pressing the 'n' key will result in the by-passing of that section of the subroutine. (No other key will produce a response.)

Message - 'Choose range? y/n'.

Throughout the program the scale of the axes will be increased if necessary to accommodate any image for which the existing axes are too small (this will result in the screen being cleared and then redrawn). The scale is calculated so as to fill as much of the screen as possible. The user has the option of setting a range of values on the x and y scales.

Pressing the 'n' key will result in the initial ranges of -1 to 1 on both the x and y scales; the axes are then displayed.

Pressing the 'y' key will produce messages asking for the minimum and maximum values of the ranges of x and y to be input. The automatic adjustment will still operate if these ranges are too small.

It was decided not to number the scales for two reasons. Firstly, I felt that it would not add to the clarity of the diagram and, secondly, because each number character is printed in an 8 pixel x 8 pixel square, multiples of eight pixels would have to be used for the scales to ensure that the numbers were printed next to the marks on the axes. This would result in smaller diagrams.

Because of the decision not to number the axes, it is necessary to ensure that the axes always contain zero. To achieve this, if the minimum and maximum of the range are both positive, the minimum is set to 0. If the minimum and maximum of the range are both negative, then the maximum is set to 0.

Message - 'Preset object? y/n'.

The preset object is written into the program as a DATA statement. Pressing the 'y' key will result in the initial object being set to this (see the display on the flow diagram on page 4).

Pressing the 'n' key enables the object data to be input as follows:

Message - 'Closed curve - y/n'.

Pressing the 'y' key results in the last point of the object being joined to the first, forming a closed shape.

Pressing the 'n' key results in the first and last points remaining unjoined.

Message - 'How many points?'.

Input the number of vertices of the object.

Message - 'x coord?' 'y coord?'.

Input the vertices as coordinates, in the order which they are to be joined (V_1, V_2, V_3, ... V_n).

The object is drawn as follows:

plot V_1, draw V_1V_2, draw V_2V_3, ... draw $V_{n-1}V_n$.

If the response to 'closed curve' was 'y' then draw V_nV_1.

Two examples of objects, together with the appropriate responses, are shown on the following page.

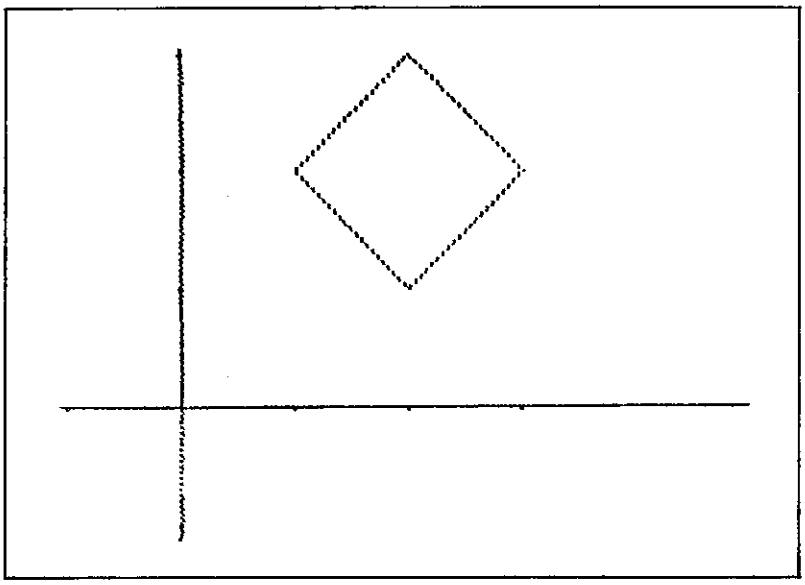


Fig. 1.1
The object resulting from inputting the data below.

```
'Closed curve - y/n'      y
'How many points?'        4
'x coord?' 2 'y coord?' 1          'x coord?' 1 'y coord?' 2
'x coord?' 2 'y coord?' 3          'x coord?' 3 'y coord?' 2
```

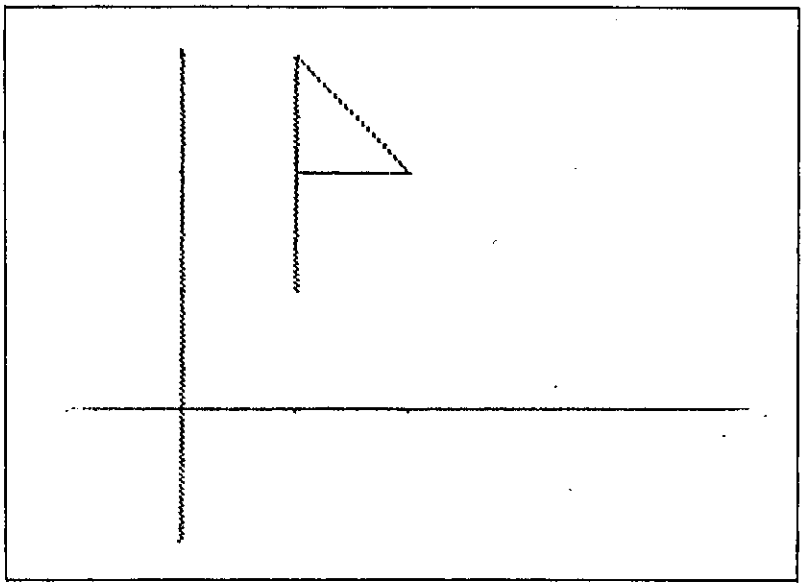


Fig. 1.2
The object resulting from inputting the data below.

```
'Closed curve - y/n'      n
'How many points?'        4
'x coord?' 1 'y coord?' 1          'x coord?' 1 'y coord?' 3
'x coord?' 2 'y coord?' 2          'x coord?' 1 'y coord?' 2
```

Message - 'Rot, reF, Trans, Enl, sHear, Stretch'.

The capitals are written in inverse video (black on a white square) to make them stand out as the key letters. Pressing the appropriate letter executes the transformation:

'r' - rotation
'f' - reflection
't' - translation
'e' - enlargement
'h' - shear
's' - stretch.

After execution of the transformation, the bottom of the screen remains blank until a key is pressed. This is so that the next instruction message does not distract from the diagram.

Message - 'Rep, Sup, sTart, New, Fol, Axes, Previous image' (capitals as above).

Pressing 'r' (repeat) results in the repetition of the transformation previously performed.

Pressing 'n' (new) results in the program being executed from the beginning.

Pressing 't' (start) results in the screen being cleared, the original object being set as the object for the transformation to follow and the screen display as, for example, figure 1.1 or figure 1.2.

Pressing 's' (superimpose), 'f' (follow) or 'p' (previous image) leaves the display unaltered. For the transformation next performed, the object will be:

's' - the original object,
'f' - the image of the transformation executed previously,
'p' - the object of the transformation executed previously.

Pressing 'a' (axes) enables the ranges of the axes to be changed as before. The transformation previously executed is then repeated with these new scales.

The Transformations

General

Any key except 'f' may be used to initiate the next part of the display. The 'f' key is used when any construction line or demonstration line is required to flash (i.e. drawn repeatedly using the 'exclusive or' command, OVER 1). The line flashes when the 'f' key is held down. Depressing and releasing any other key will continue the construction or demonstration.

Rotation

Message - 'Angle of rot, in degrees? $\boxed{L}$'.

Any angle, positive or negative, may be input.

Message - 'x coord of centre of rot? $\boxed{L}$ ' (input required)

- 'y coord of centre of rot? $\boxed{L}$ ' (input required).

The centre of rotation is plotted, and intermediate images are drawn every 5° to show the rotation. (If the angle of rotation is not a multiple of 5, then the first image is drawn so that each subsequent image is found by rotating 5°.) Each image is erased before the next is drawn, until the final image is displayed.

Demonstration (if desired - see page 5)

Taking each vertex in turn:

1. Lines are drawn from the object to the centre of rotation and then to the image. This demonstrates that the distance from the centre of rotation remains constant and also that the angle between the lines equals the angle of rotation.
2. An arc is drawn from the object to the image to indicate that each point moves in a circular path.
3. The lines and arc are then erased.

The diagram on the next page shows the final image of the object in figure 1.1 when the input responses are 60, 1, 1. The demonstration lines have been drawn in red for clarity.

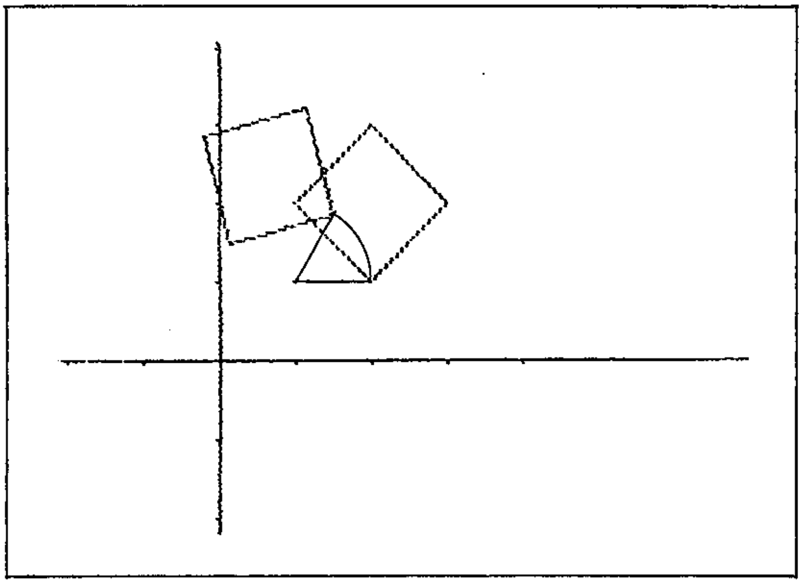


Fig. 1.3
Rotation of 60°
about (1,1).

Reflection

Since I have used the properties of a reflection, i.e. the mirror line is the mediator of the line segment joining an object point to its image, to construct the final image, there is no separate demonstration of these properties.

Message - 'Invariant line y = mx + c, y/n

It was felt that most pupils below the 6th form (i.e. those for whom the program is intended) would have met equations of straight lines in the form of either y = mx + c or x = c, rather than ax + by + c = 0. This subroutine has therefore been written in two parts.

Pressing the 'y' key produces the response 'm?$\boxed{L}$' 'c?$\boxed{L}$', each requiring input.

Pressing the 'n' key produces the response:

'Invariant line x = c. c?$\boxed{L}$'.

The invariant line is drawn when c has been input.

The image of each point is produced as follows:

1. A line is drawn from the object point perpendicular to the mirror line so that the mirror line bisects it.
2. The line is erased, leaving the end point (the image) plotted.

When all the image points are plotted, they are joined to form the image.

Figures 1.4 and 1.5 show reflections with the inputs -1 (following an 'n' response) and 1, 4 (following a 'y' response)? respectively.

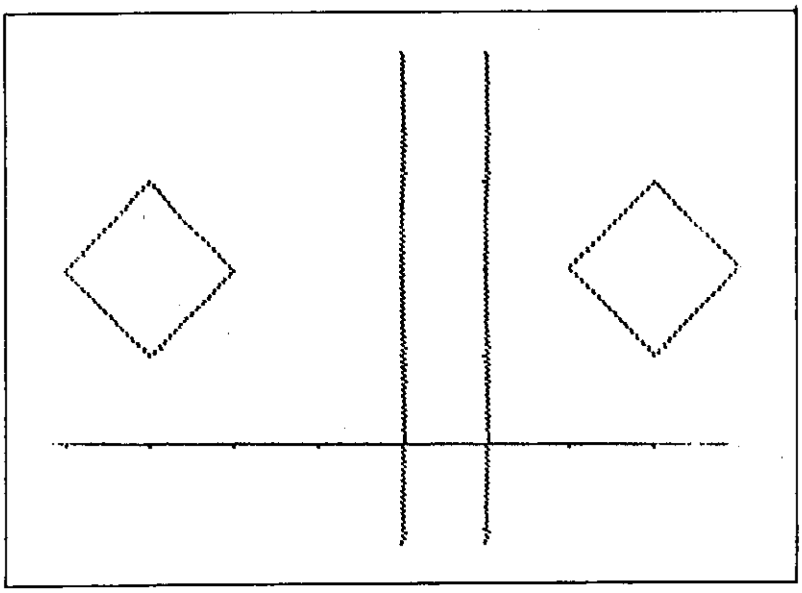


Fig. 1.4
Reflection in
x = -1.

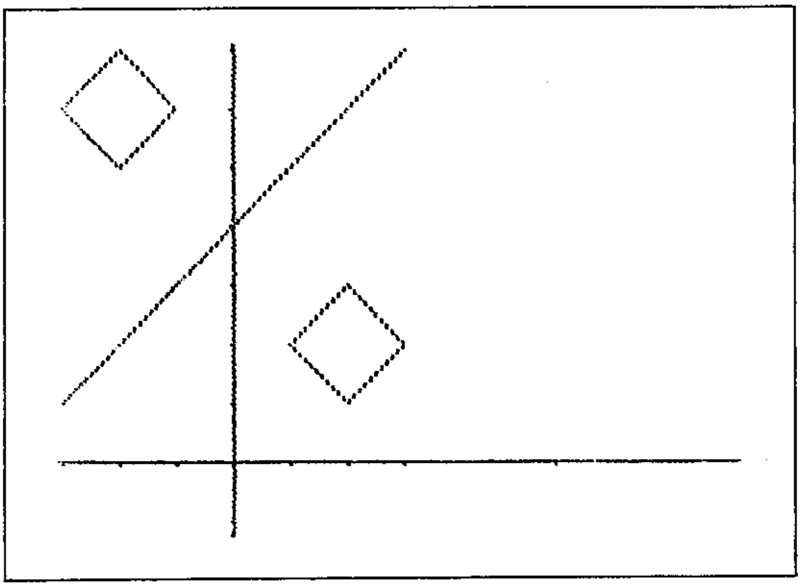


Fig. 1.5
Reflection in
y = x + 4.

The two diagrams below show the initial stages of the reflection in the line y = x + 4. The mirror line is drawn first (figure 1.6a). Figure 1.6b shows the first stage of constructing the image of A. It can be seen that, because of the OVER 1 command, part of the object has been erased. If OVER 0 were used, in which case the line AB would not be erased, it would be unclear which of the two images, that of A or that of B, was being constructed. In this case particularly, holding down the 'f' key to cause the line to flash on and off improves the clarity of the construction.

A further depression of a key results in a display as in figure 1.6a, but with the image of A plotted.

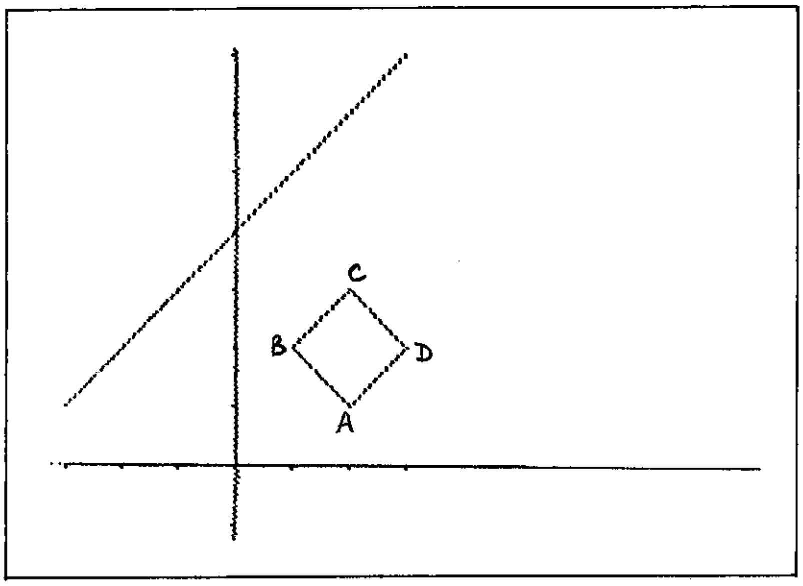


Fig. 1.6a
The letters have been added to clarify the explanation.

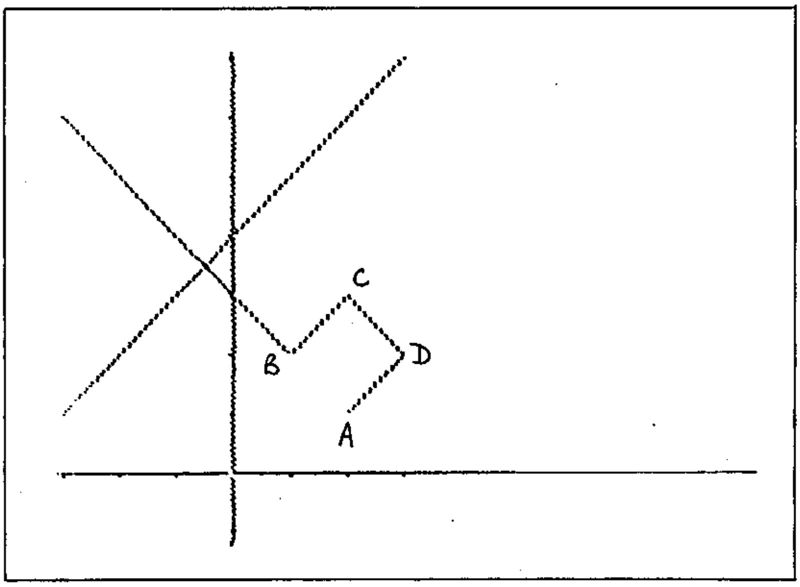


Fig 1.6b

Translation

Message - 'Vector. X comp? $\boxed{L}$ ' - requires the input of the x component of the translation vector.

'Vector. X comp? 4 Y comp? $\boxed{L}$ ' - requires the input of the y component of the translation vector.

On depressing any key the object is translated onto the image in ten stages. The intermediate stages are drawn and then erased after a slight pause.

Demonstration

Using each point of the object in turn:

1. A line is drawn from the object to the image.
2. The line is erased.

The final stage:

1. Draws all the lines indicated above to show that they are all parallel and equal in length.
2. Erases all the lines.

The diagram below shows the object and the final image for the vector $\begin{pmatrix}4\\3\end{pmatrix}$. (The demonstration lines have been drawn in red for clarity.)

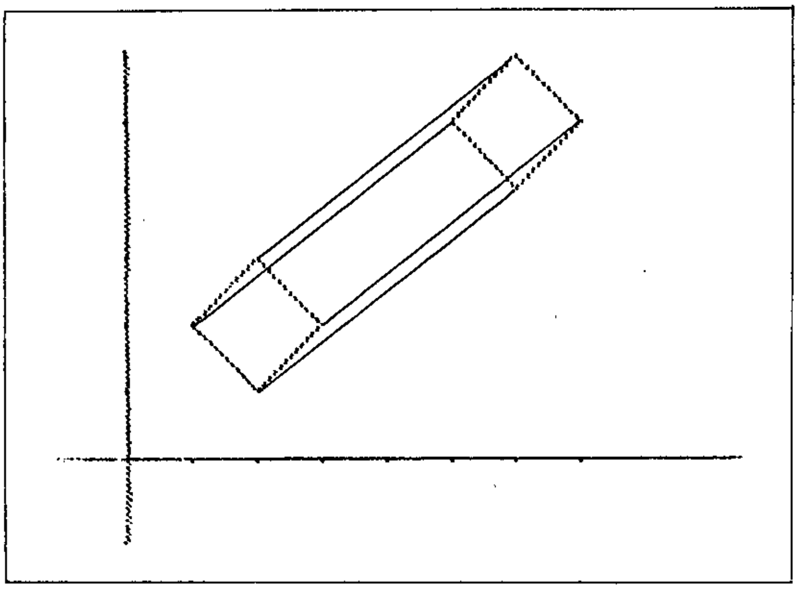


Fig. 1.7
Translation,
vector $\begin{pmatrix}4\\3\end{pmatrix}$.

Enlargement

Message - 'C of Enl. x coord? $\boxed{L}$ ' - requires input of the x coordinate of the centre of enlargement.

- 'C of Enl. x coord? 1 y coord? $\boxed{L}$ ' - requires input of the y coordinate of the centre of enlargement.

Message - 'Scale factor? $\boxed{L}$ ' - input may be any real number.

The centre of enlargement is plotted.

The image of each point in turn is constructed as follows:

1. A line is drawn from the centre of enlargement to the object.
2. If the scale factor is positive then the line is erased.
3. The first line, enlarged by the scale factor is drawn from the centre of enlargement.
4. The line (lines, if s.f. is negative) is erased, and the image (the endpoint of the second line) is plotted.

When all the image points have been plotted, they are joined to form the image.

The diagram below shows an enlargement of scale factor 2, centre of enlargement (1, 3).

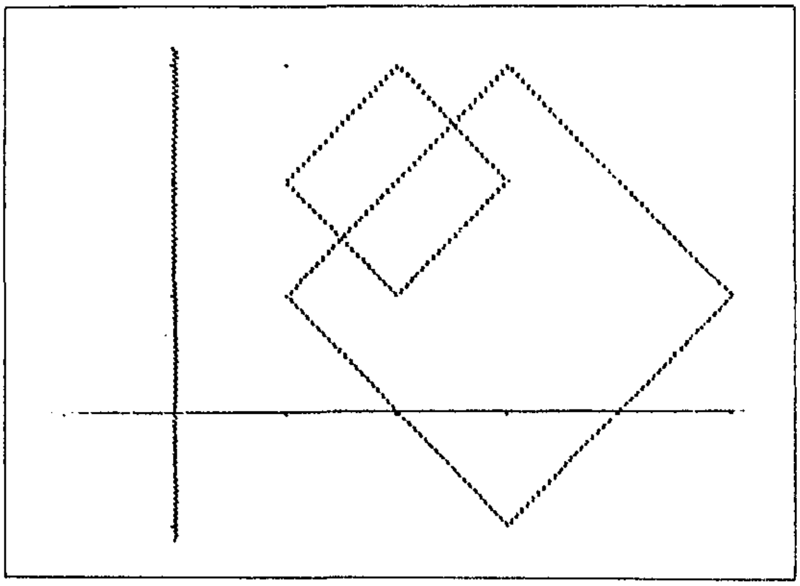


Fig. 1.8
Enlargement
scale factor 2,
centre (1, 3).

Shear

The data for the invariant line is input as for Reflection, page 10.

This is followed by the messages:

'Object x coord? $\boxed{L}$ '

'Object x coord? 1 y coord? $\boxed{L}$ '

'Image x coord? $\boxed{L}$ '

'Image x coord? 2 y coord? $\boxed{L}$ '.

These require an object point and its image under the shear to be input. If either the object point (x,y) or the image point (xx,yy) lies on the invariant line, or the line joining them is not parallel to the invariant line, the message 'Incorrect information - press any key' is printed. Pressing any key results in the input messages above, requiring the object point and its image to be input again.

The invariant line is drawn and the object shape is then transformed in ten stages. Each intermediate stage is drawn and then erased.

The following two diagrams show an intermediate stage (figure 1.9) and the final image (figure 1.10) for the shear with invariant line $y = x + 4$ and $(1, 3) \rightarrow (2, 4)$.

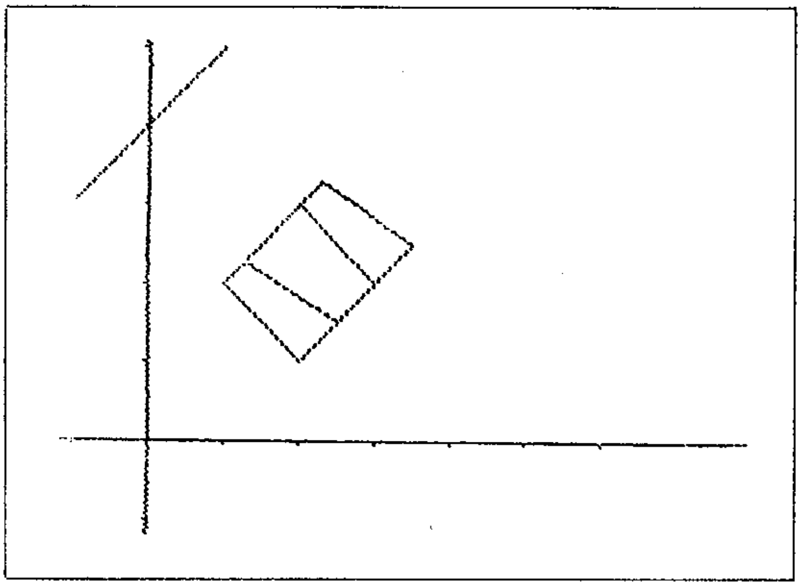


Fig. 1.9

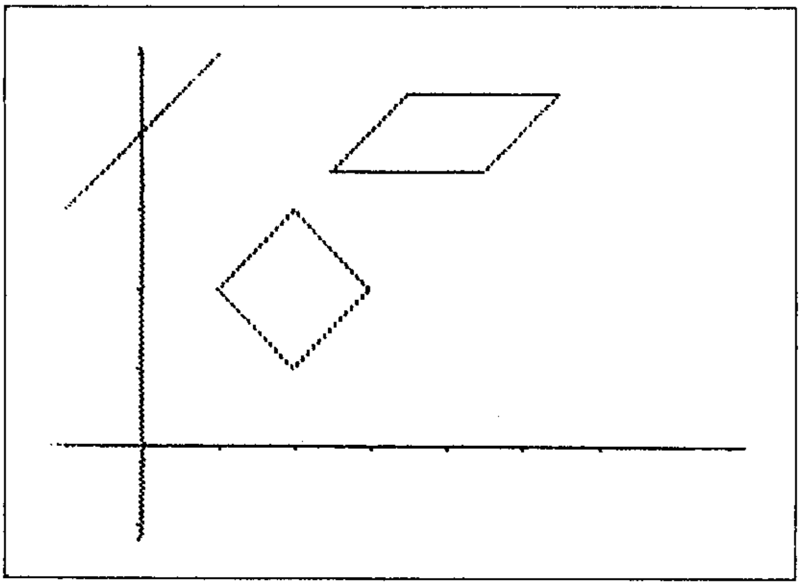


Fig. 1.10

Figure 1.9 shows that some parts of the image coincide with the object. If OVER 1 was used for the whole image, these parts would be erased, and so OVER 0 is used where this happens, OVER 1 for the rest of the image.

Figure 1.10 is an example of where the transformation may be required to be repeated with extended axes, to show more of the invariant line.

Demonstration

Each point is demonstrated as follows:

a) If the object point is the same distance from the invariant line as (x,y) and on the same side of the invariant line, then

   1. A line is drawn joining (x,y) to (xx,yy)
   2. A line is drawn from the object point to its image.

   This demonstrates that both points shear the same distance in the same direction.

   3. Both lines are erased.

b) If the object point is not the same distance from the invariant line as (x,y) or if they are on opposite sides then:

   1. A line is drawn through the object point and through (x,y) to meet the invariant line. (The scales will have been adjusted, if necessary,

so that this point on the invariant line is included in the diagram.)

2. A line is drawn from (x,y) to (xx,yy).
3. A line is drawn through where the first line meets the invariant line and through (xx,yy).
4. A line is drawn from the object, parallel to the invariant line, to meet the third line (at the image).

This demonstrates the method of finding the image of a point, given the invariant line, another point together with its image. It also shows that points are sheared in proportion to their distances from the invariant line, and that points on opposite sides of the invariant line shear in opposite directions.

5. All the four lines are erased.

Figure 1.11 shows the shear after the first three stages of the demonstration. Stage 1 is shown in red, stage 2 in green, stage 3 in black, for clarity.

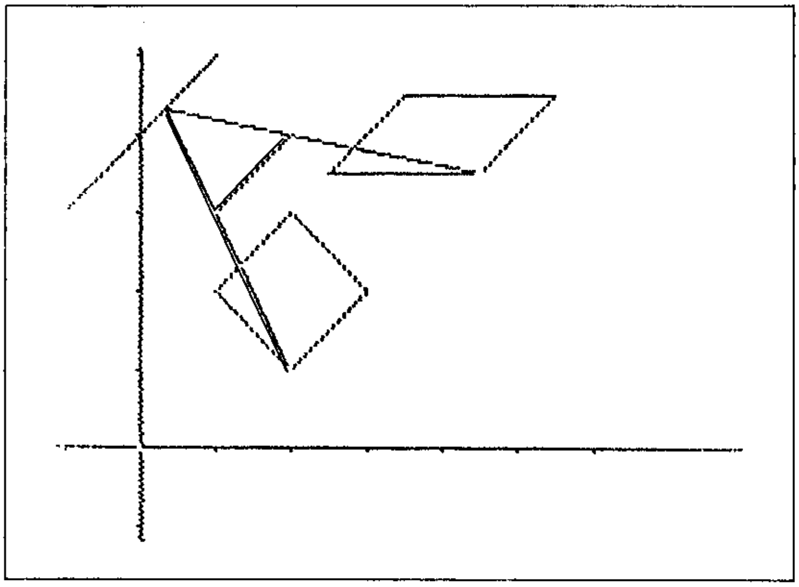


Fig. 1.11

Figure 1.12 (see the next page) shows the addition of the fourth line of the demonstration indicating the effect of the 'exclusive or' command. Holding down the 'f' key will cause the alternate displays of figures 1.11 and 1.12 (see previous comments in Reflection, page 12).

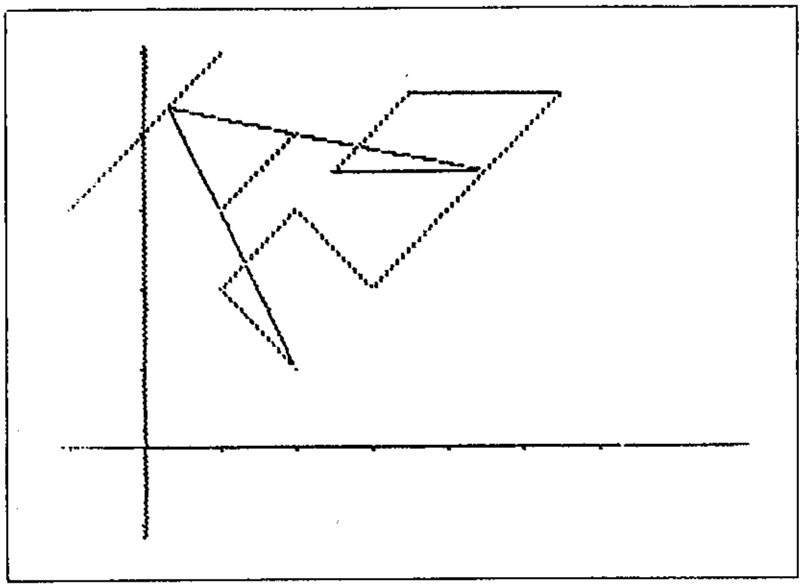


Fig. 1.12

Figure 1.13 shows a shear with x = -1 invariant and (1,2)->(1,3). (The demonstration lines have been drawn in red, for clarity.)

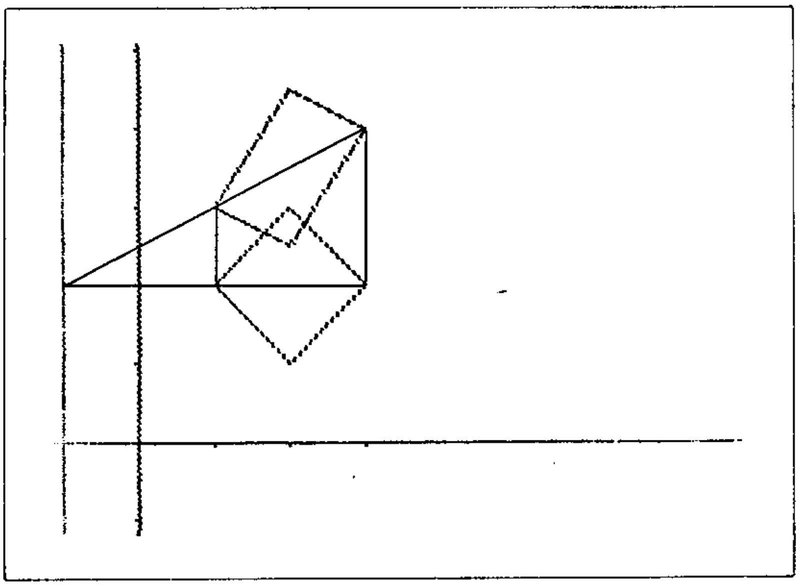


Fig. 1.13

Stretch

The data for the invariant line is input as for Reflection, page 10.

This is followed by the message - 'scale factor? $\boxed{L}$ '.

Any value may be input for the scale factor. Input of 0 results in the shape being transformed onto the invariant line. A negative scale factor is equivalent to a reflection in the invariant line, followed by a stretch taking the modulus of the scale factor.

The invariant line is drawn and the object shape is then transformed onto its image in 10 stages. Each intermediate stage is drawn and then erased after a slight pause.

Figure 1.14 shows the object of figure 1.1 and its image after a stretch from the line x = -1, scale factor 0.5.

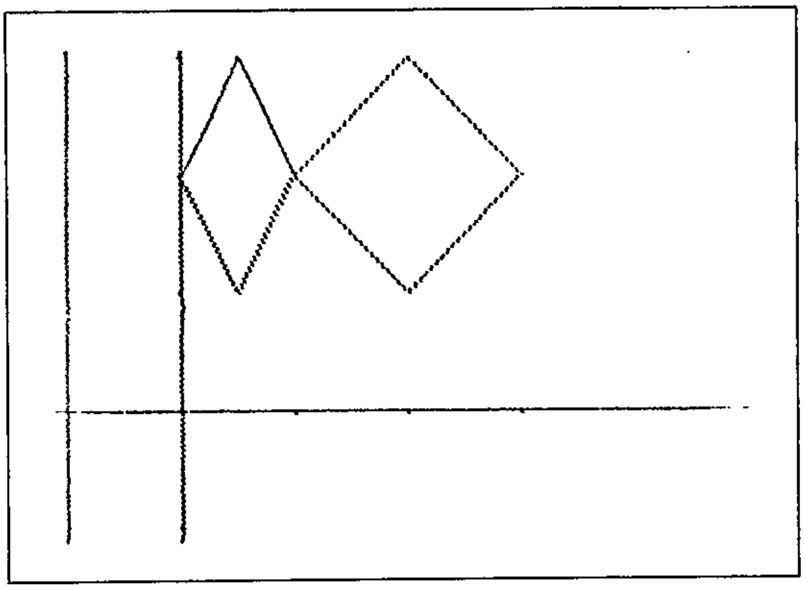


Fig. 1.14

Figure 1.15 (see next page) shows the image after a stretch from the line y = 2x + 2, scale factor 2.

In the situation where part of an image coincides with part of the object (e.g. an edge of the object is perpendicular to the invariant line) it is drawn as explained on page 16 for the shear.

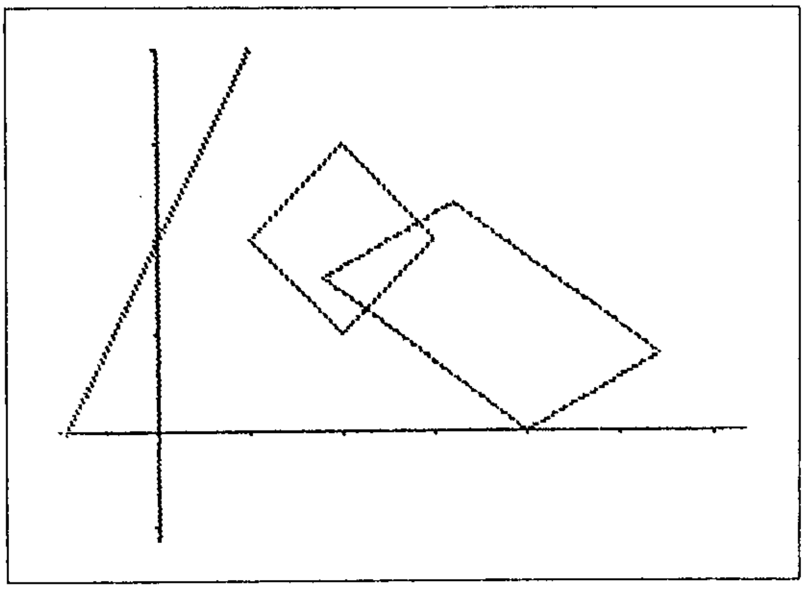


Fig. 1.15

Demonstration

Each point is treated in turn as follows:

1. A line is drawn from the object, perpendicular to the invariant line, to meet the invariant line. (Where necessary, the axes will have been adjusted to accommodate this point on the diagram.)
2. The line is then erased.
3. A line is drawn from this point on the invariant line, perpendicular to the invariant line, whose length is that of the first line multiplied by the scale factor.
4. This line is then erased.

The demonstration shows that each point is transformed perpendicular to the invariant line. The distance of the image from the invariant line equals the distance of the object from the invariant line multiplied by the scale factor. Points on the invariant line are invariant.

Figures 1.16 and 1.17 show stages 1 and 3 of the demonstration, applied to the stretch of figure 1.14.

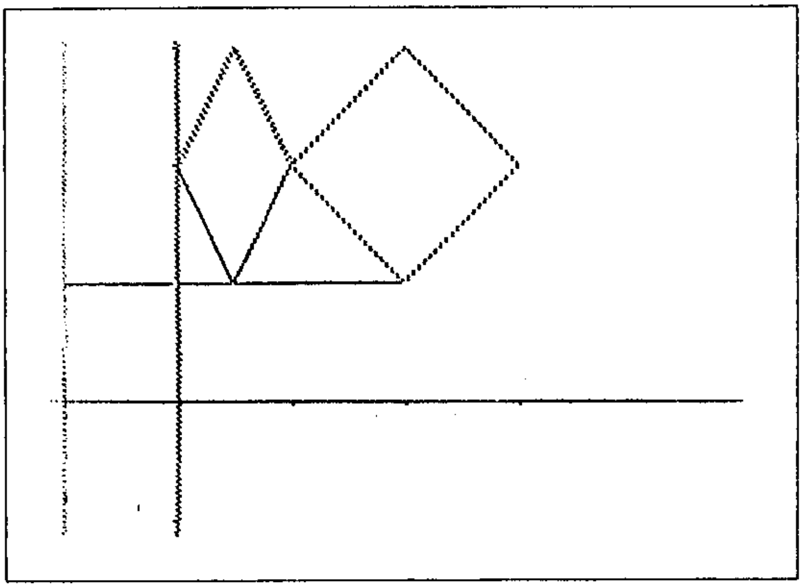


Fig. 1.16

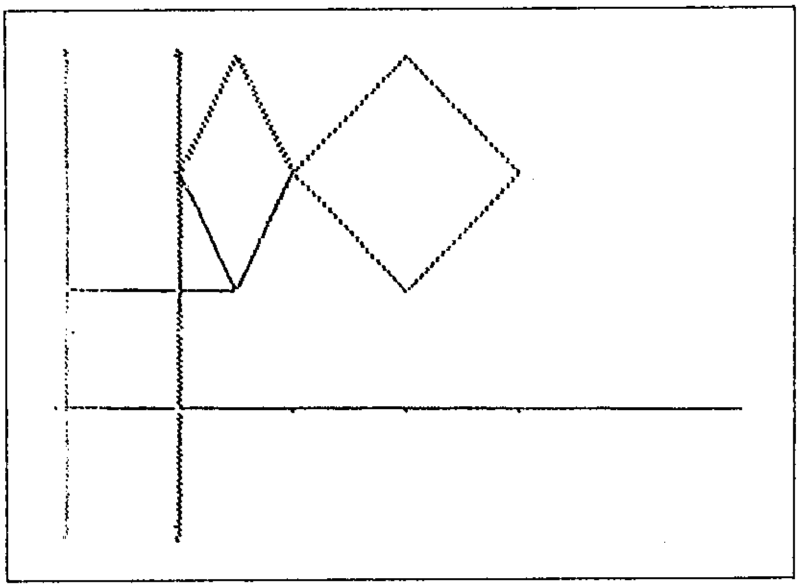


Fig. 1.17

Using the program in the classroom

It is not intended that the computer simulation should be the only means of illustrating transformations - much of the preliminary work on reflections and rotations, for example, will involve the use of mirrors and tracing paper.

I shall explain some of the ways in which I have used the program with groups of children, and suggest further uses.

1. Investigating Translations - 4th year group in the 50-75% ability range.

This series of lessons followed work on displacement vectors, and so the children were familiar with representing displacements by a column matrix.

Before the beginning of the lesson, the translation vector $\begin{pmatrix}10\\6\end{pmatrix}$ had been input. The initial display was as in figure 1.18, although I would suggest turning down the brilliance control to black out the screen until required. This is a useful teaching point, whenever the children's attention is required to be directed away from the screen.

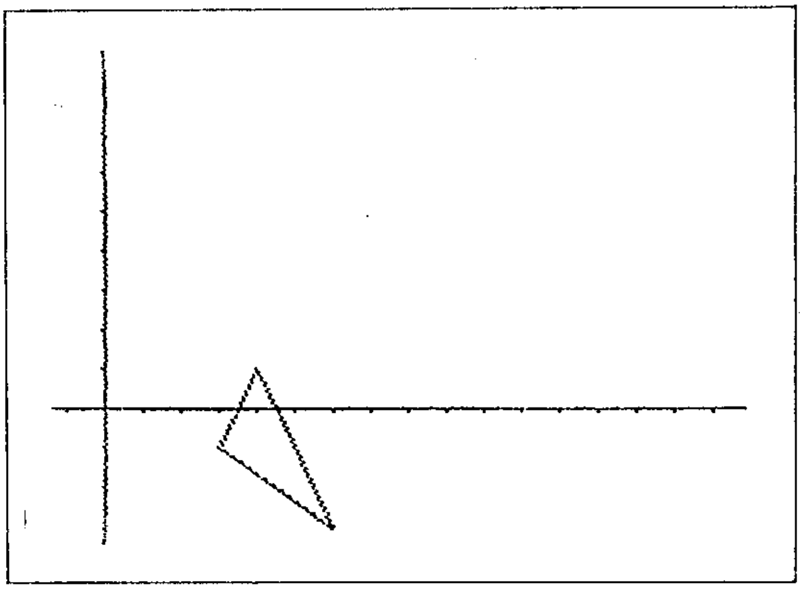


Fig 1.18

It was explained that a new transformation was to be investigated, the translation was then demonstrated. Questioning the pupils brought out the property that all the points move the same distance and in the same direction (see figure 1.19); the transformation was described by the children as 'a straight move, without turning'.

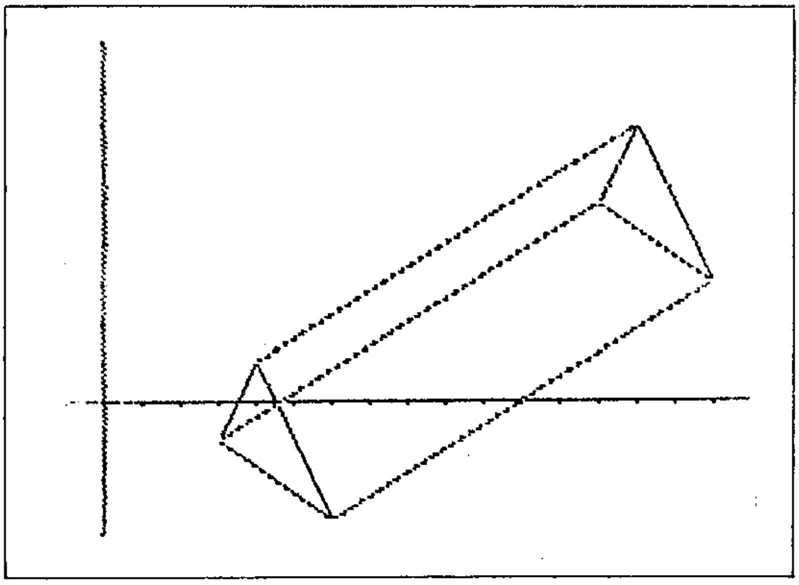


Fig. 1.19

It was agreed that this description was insufficient to describe the transformation precisely. Since the idea of a displacement vector was recently acquired, it was soon suggested that a column vector could be used to describe the translation. $\begin{pmatrix}10\\6\end{pmatrix}$ was found for the example given.

The children were then told to choose a vector of their own, and to illustrate its translation. One pupil asked whether they had to start with a triangle. Subsequently the shape shown in figure 1.20 was chosen by the class as the object.

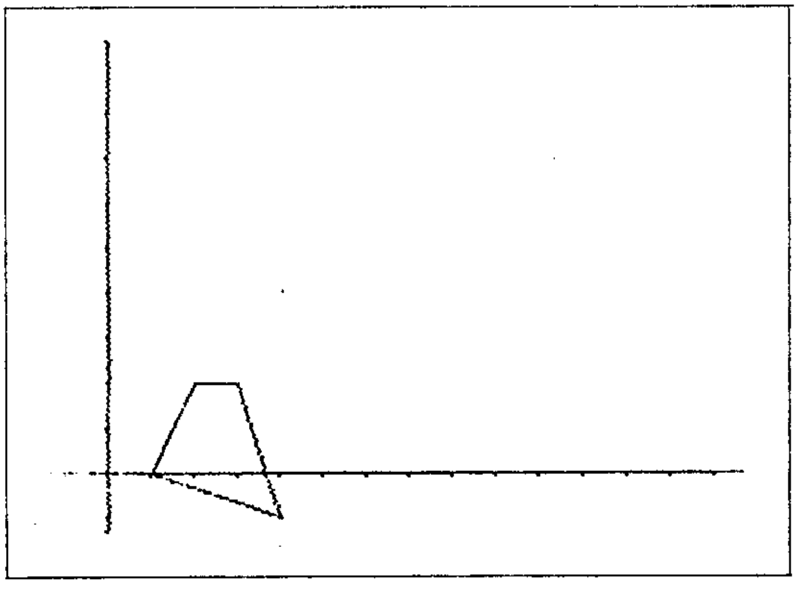


Fig. 1.20

I found that a small number of pupils had transposed the vector components in carrying out the translation and so, in the next lesson these and other pupils were invited to show their translations, using the computer. I believe that to discover their error themselves in this way was more effective in rectifying it than if I had told them why they were wrong. This led on to some guided discussion to try to give the pupils a better 'feel' for how changing a vector will affect the translation. Using the computer to superimpose the different images, I asked for a vector which would cause the object to move the same amount up as to the right, vectors which would translate the object higher, more to the right than up, vertically, horizontally, how would the object move under a specified translation, and so on. At the end of the session it was clear to all the pupils that the top (x) component controlled the horizontal displacement and the bottom (y) component controlled the vertical displacement.

A similar approach was adopted with translations whose vectors had negative components.

This work was consolidated with an exercise which contained two types of questions, those which required the pupils to draw objects with given coordinates and to translate them with given vectors, and those requiring the pupils to draw objects and images with given coordinates and then to give the vector to describe the translation from one to the other.

The idea of an inverse translation was introduced by illustrating the translation $\begin{pmatrix}10\\6\end{pmatrix}$ and then asking what translation would transform this image back onto the original. The immediate response was '$\begin{pmatrix}6\\10\end{pmatrix}$', which I illustrated using the computer. 'Oh! No, $\begin{pmatrix}-6\\-10\end{pmatrix}$' was the next response, which was also shown on the computer. The correct answer was then suggested. Each suggestion had been written on the blackboard, beside $\begin{pmatrix}10\\6\end{pmatrix}$, and then eliminated when found to be incorrect.

In the subsequent work on inverses, I asked more guided questions before asking for the inverse to be stated. For the translation $\begin{pmatrix}-4\\5\end{pmatrix}$, I asked in what general direction the object had been translated, how far left and how far up, and similar questions to lead to the inverse. Nearly all the pupils were then able to write down $\begin{pmatrix}4\\-5\end{pmatrix}$ as the the required vector. Using the computer to test the answers and the blackboard to record the results, the simple rule was established for finding the inverse of a translation, when given its vector.

Graham Ruddock, in 'Children's Understanding of Mathematics: 11-16' (1981), concludes from his research that double translations were found to be particularly difficult. Elsewhere in the book, however, it is suggested that combinations of translation vectors provide a concrete use of directed numbers and gives meaning to the addition of directed numbers, especially negative.

I concluded the topic by considering the combination of translations and found that the pupils were able to add directed numbers more efficiently than when presented in a more abstract way. Using the computer display, the pupils had little difficulty in understanding how translations combine to give the equivalent of a single translation. Initially pairs of translations were illustrated on the screen and the children were asked what transformation would map the original object onto the second image. Nearly all the pupils responded with a translation. The pairs of translations were written on the blackboard and it was soon realised that the vectors needed to be added to obtain the single translation, which was then demonstrated by using the computer (see figures 1.21, 1.22 and 1.23).

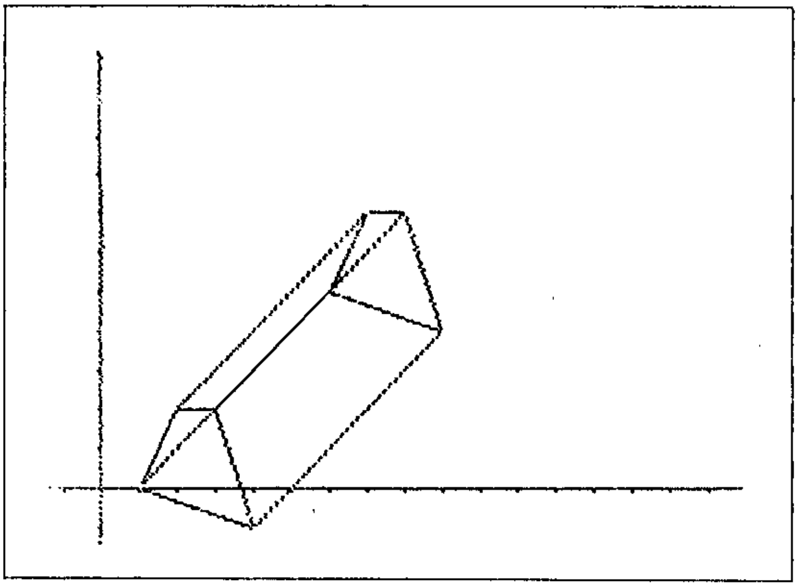


Fig. 1.21
Translation $\begin{pmatrix} 5 \\ 5 \end{pmatrix}$.
(The line shown in red is eliminated on the screen because of OVER 1 command.)

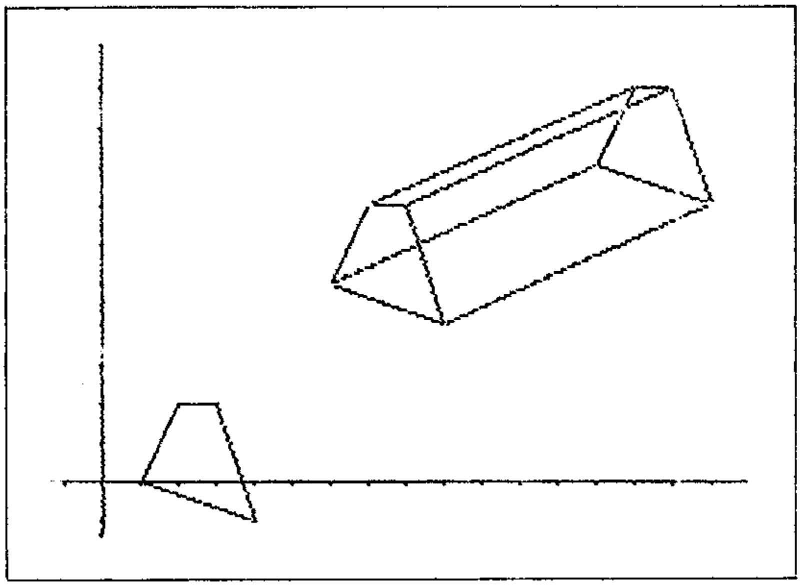


Fig. 1.22
Translation $\begin{pmatrix}7\\3\end{pmatrix}$.

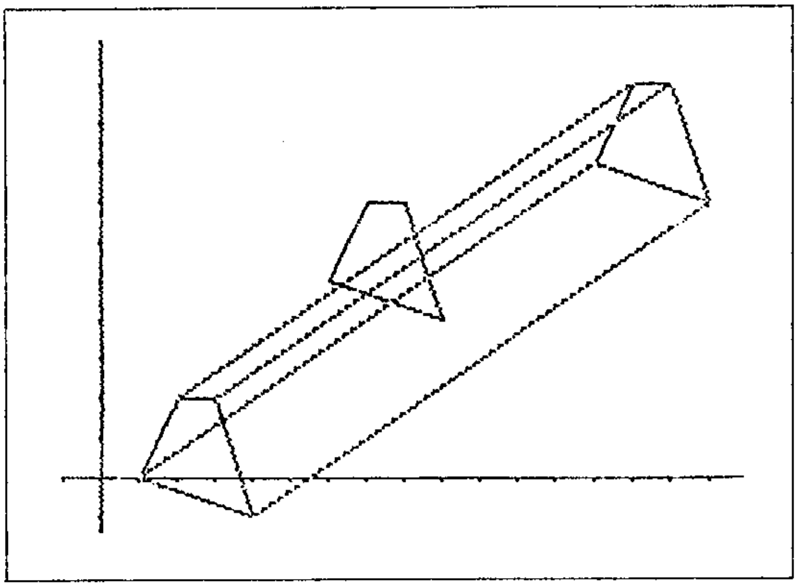


Fig. 1.23
Translation from the original object of $\begin{pmatrix}12\\8\end{pmatrix}$.

I then posed the problem of finding the second of a pair of translation vectors which combine to give a stated translation vector. It was my intention only to consider pairs of translations, but a mistake by one boy led to the consideration of more than two. He was asked which translation combines with $\begin{pmatrix}-5\\4\end{pmatrix}$ to give the equivalent of $\begin{pmatrix}-13\\6\end{pmatrix}$. His answer of $\begin{pmatrix}-7\\2\end{pmatrix}$ was shown on the display (see figures 1.24 1.25 and 1.26) after which he stated that he should have said $\begin{pmatrix}-8\\2\end{pmatrix}$. This correction was then illustrated by pressing 'Previous image' option and inputting $\begin{pmatrix}-8\\2\end{pmatrix}$ (see figure 1.27).

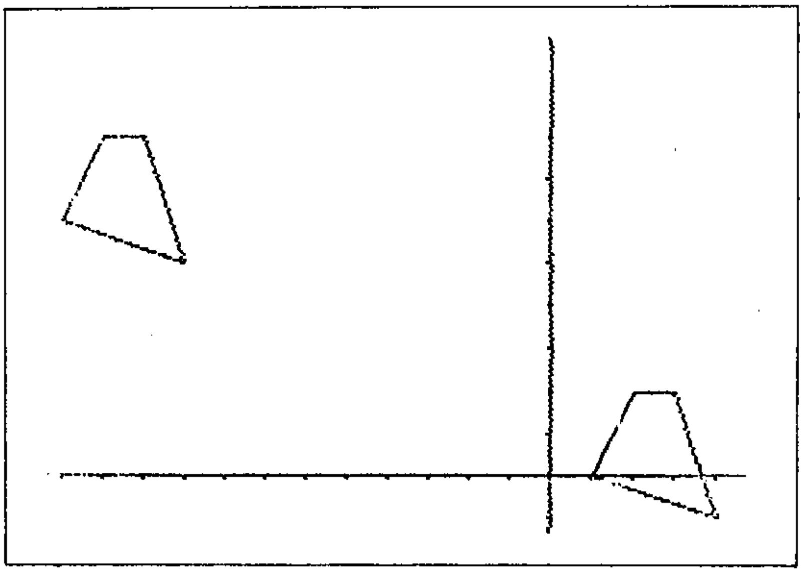


Fig. 1.24
Translation
$\begin{pmatrix}-13\\6\end{pmatrix}$.

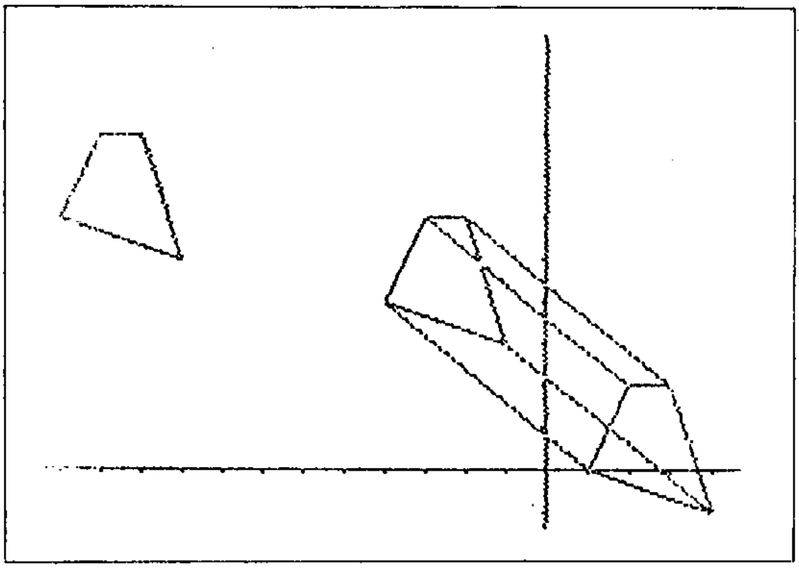


Fig. 1.25
Translation
$\begin{pmatrix}-5\\4\end{pmatrix}$.

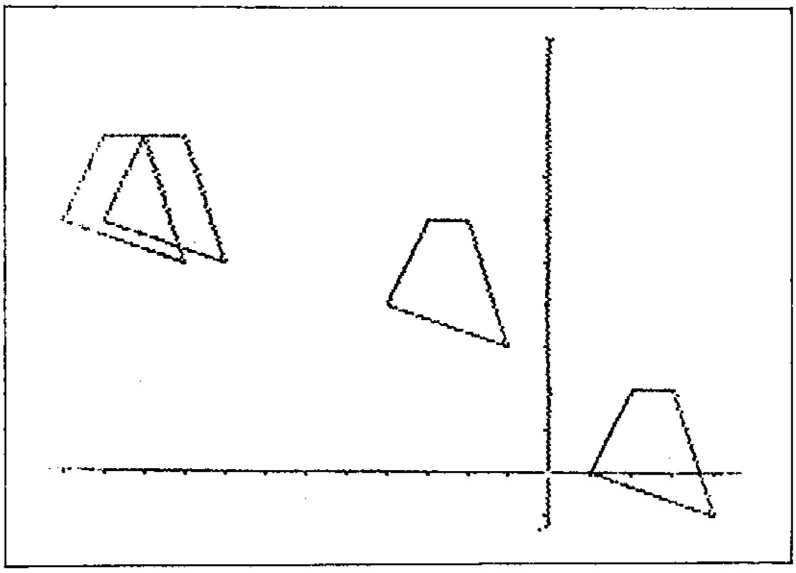


Fig. 1.26

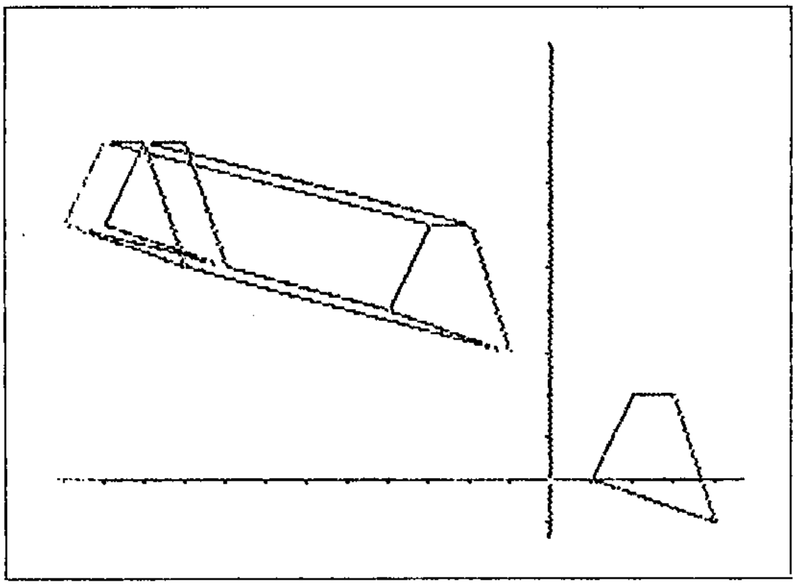


Fig. 1.27
Translation
$\begin{pmatrix}-8\\2\end{pmatrix}$.

I returned to the pupil's first image and asked him what translation was needed to transform this image onto the final position, to which he replied correctly. The class was then able to generalise a rule for the combination of any number of vectors.

Conclusions

(i) The children were interested in the lessons, and no-one asked what the purpose was of learning about translations (a response which I have encountered when using more formal methods of instruction). This was probably partly due to the novelty of using the computer but also, I believe, because the translations were being demonstrated dynamically.

(ii) The children were able to give meaning to negative numbers and combine them with meaning, although it is doubtful if many could abstract this to addition of directed numbers.

(iii) The stimulus of the computer display enabled the subject to extend beyond the level normally expected of a pupil in this ability range.

2. The Combinations of Rotations and Reflections - 4th year group in the 25-50% ability range.

During the lessons before the computer was used, the pupils had been investigating the effect of the transformation matrices
P $\begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}$, Q $\begin{pmatrix}-1 & 0\\0 & 1\end{pmatrix}$, R $\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$, S $\begin{pmatrix}0 & -1\\-1 & 0\end{pmatrix}$, X $\begin{pmatrix}0 & -1\\1 & 0\end{pmatrix}$, Y $\begin{pmatrix}-1 & 0\\0 & -1\end{pmatrix}$,
Z $\begin{pmatrix}0 & 1\\-1 & 0\end{pmatrix}$, I $\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$, on the figure with coordinates O (0,0), A (1,1), B (2,1), C (2,0). The resulting figure is shown below.

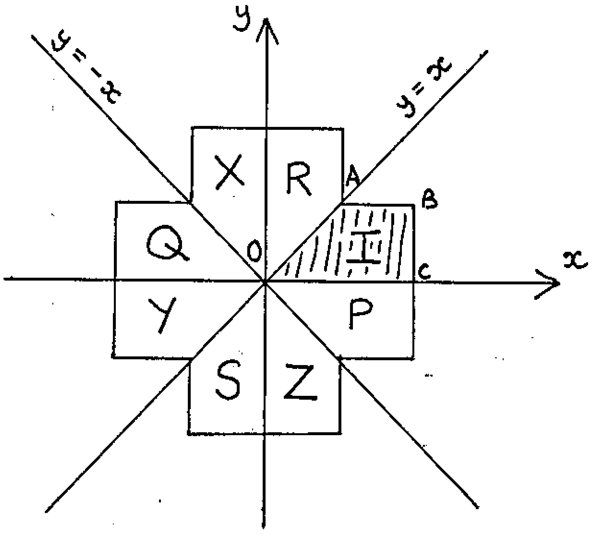


Fig. 1.28

After discussion with the pupils, it was agreed that all the transformations represented by the matrices were rotations or reflections; the pupils used tracing paper to find the transformations. The results were then tested using the computer. The opportunity arose to remind the pupils that to describe a rotation precisely needs the centre of rotation in addition to the angle, and also that a convention for the direction of rotation was necessary to avoid ambiguity. It was also shown that a rotation of 270° was equivalent to a rotation of -90° about the same centre.

The following list was produced:

Reflection in the x-axis - P
in the y-axis - Q
in $y = x$ - R
in $y = -x$ - S

Rotation about O of 90° - X
180° - Y
270° - Z
The identity transformation - I

In considering the combinations of reflections and rotations, I followed a similar approach of pupil investigation using tracing paper, together with illustrations of the transformations using the computer. A combination table as shown below was produced on a worksheet and pupils were asked to find what single transformation is equivalent to 'P followed by Q'.

Second Transformation

First Transformation

	I	X	Y	Z	P	Q	R	S
I								
X								
Y								
Z								
P								
Q								
R								
S								

Fig. 1.29

All the pupils produced the answer 'Y', which was demonstrated as in figures 1.30, 1.31 and 1.32, and then recorded on the table.

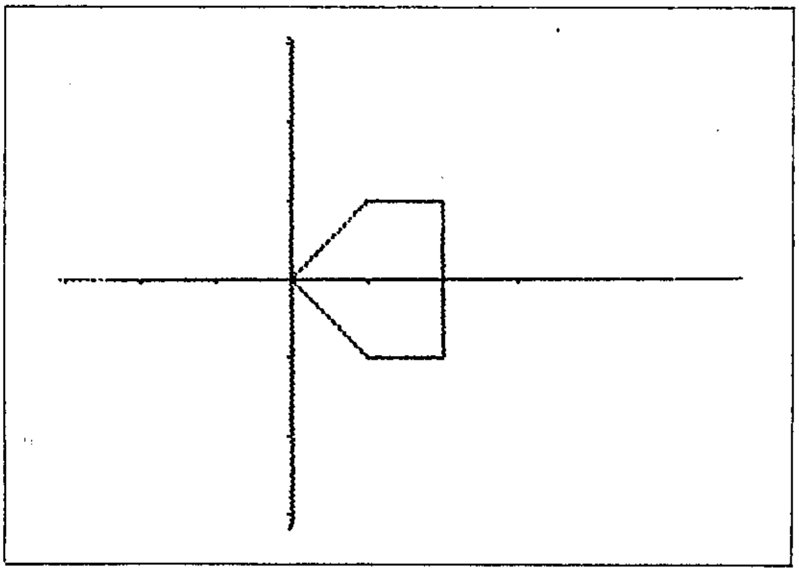


Fig. 1.30
A reflection
in the x-axis.

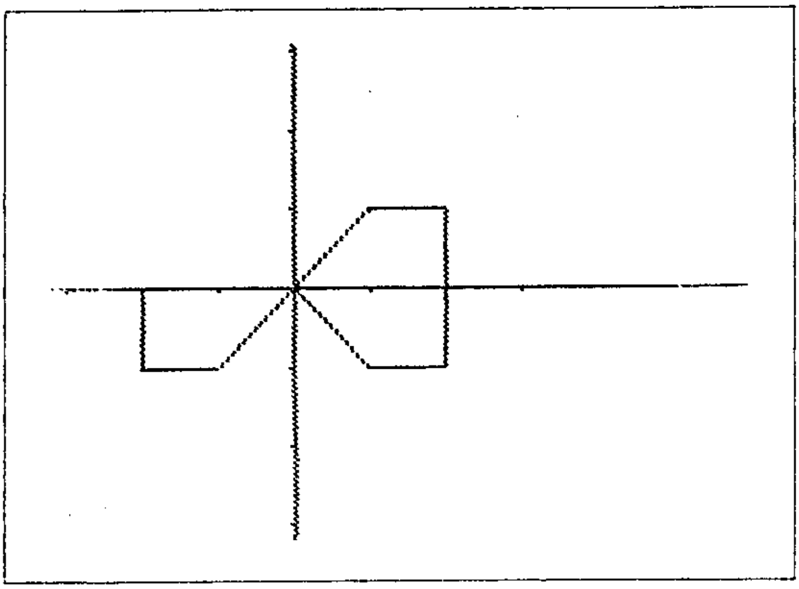


Fig. 1.31
'P followed
by Q'.

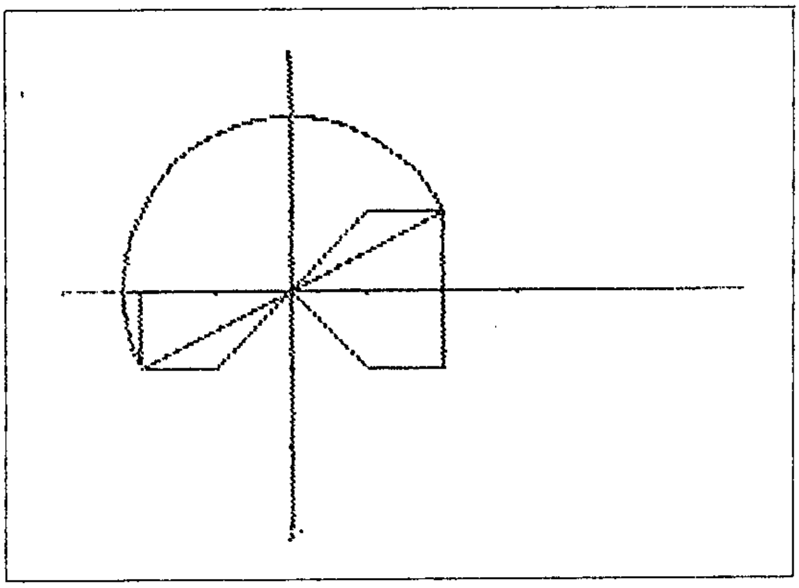


Fig. 1.32
Part of the
demonstration
to show that
'P followed
by Q' is
equivalent to a
180° rotation
about O.

I then asked the pupils to complete the row of the table where P was the first transformation. This showed later to be a poor choice of rows because, although most pupils had the correct transformation, the reasoning was incorrect in some cases. This became apparent when 'P followed by X is equivalent to R' was demonstrated on the computer (see figures 1.33, 1.34 and 1.35).

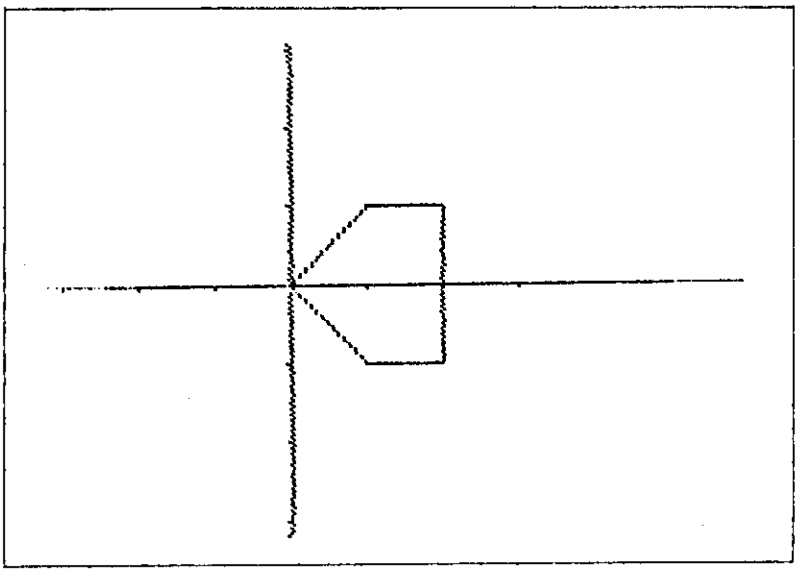


Fig. 1.33
Transformation
P.

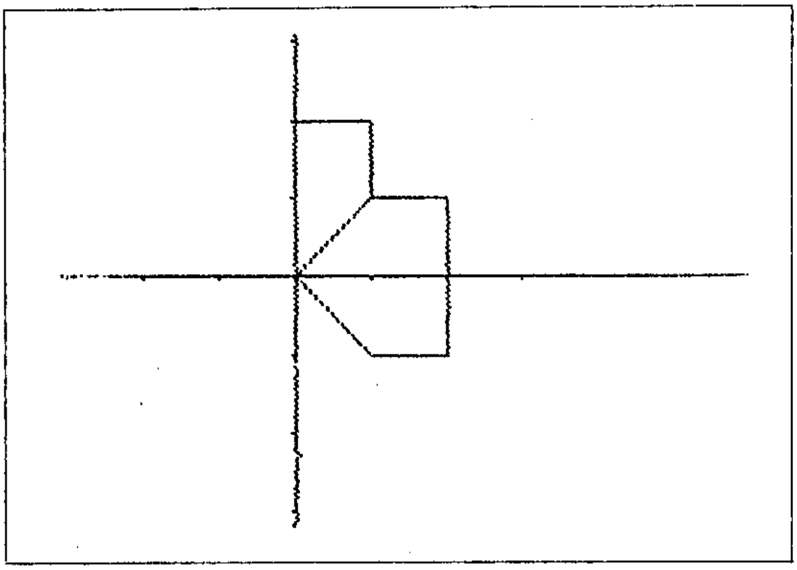


Fig. 1.34
'P followed
by X'.

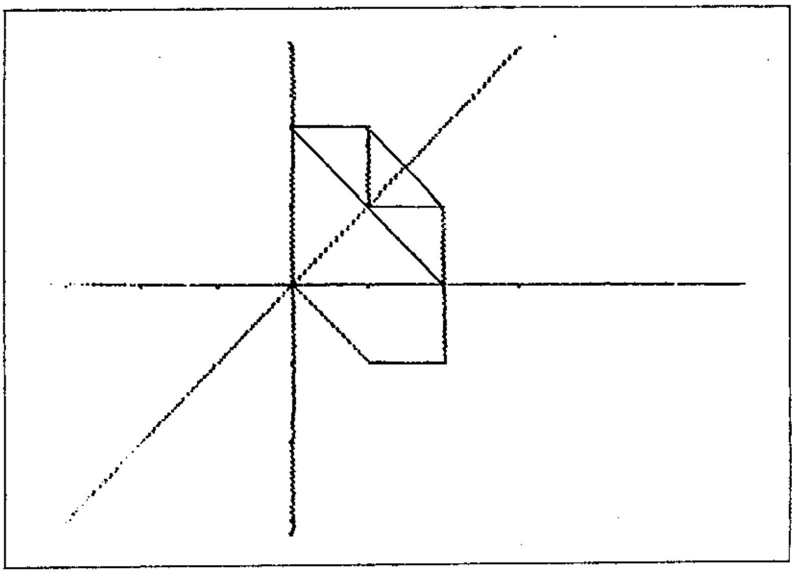


Fig 1.35
'P followed by
X is equivalent
to R'.
(The series of
construction
lines are
shown in red.)

One girl remarked that she had found transformation R by considering which transformation mapped the shape labelled P onto the shape labelled X in figure 1.28. The whole row could be completed with the correct answers using this reasoning, and so it was necessary to ensure that 'P followed by X' was correctly interpreted. I believe that the computer demonstration helped greatly in this respect but the teacher needs to be aware of this error in thinking, where the pupil interprets the single transformations as different positions of the original object. I think that for demonstrating transformation A followed by transformation B it would have been better to have chosen A and B such that $BA = C$, but $CA \neq B$ to ensure that the correct answer cannot be achieved from this incorrect reasoning.

The remainder of the table was completed with few mistakes.

Subsequent lessons used the results of this table to investigate how transformation matrices combine to give the matrix which represents the combined transformation.

3. Combinations of Transformations - 5th year group in the top 20% ability range.

General combinations of transformations are unlikely to be within the understanding of the majority of pupils, but I think that more able pupils can benefit by the study of the subject. I have taught this topic in the past using the S.M.P. textbooks, but found that the time taken by the pupils to produce diagrams and the inaccuracy of the resulting diagrams usually led to a lack of interest after a short while. The use of the computer helped remove these restrictions.

The first of a series of lessons was spontaneous in that, when the class arrived in the room, the computer was set up from a previous lesson, for demonstrating the combination of translations as already outlined. I was asked about the computer program and showed the pupils how I had used it with the 4th year class. This led to the question 'What else can it do?' and onto the discussion of combining transformations in general. The class was eager to suggest combinations to investigate. With my guidance, the task was approached in a more systematic way, starting with simple examples of one type of transformation, leading to general examples.

Rotation

The first reaction of the pupils was to state that a rotation followed by a rotation was equivalent to a single rotation. Investigation revealed that a translation may also result.

The 'discoveries' are shown below.

(a) A rotation of a° followed by a rotation of $-a^{\circ}$ or $360 - a^{\circ}$ about the same centre is equivalent to the identity (see figures 1.36, 1.37 and 1.38).

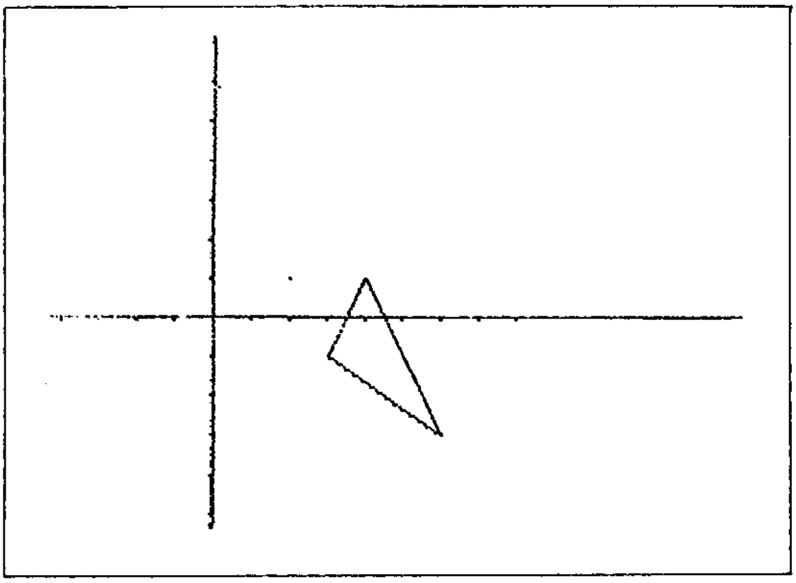


Fig. 1.36
The initial
(preset) object.

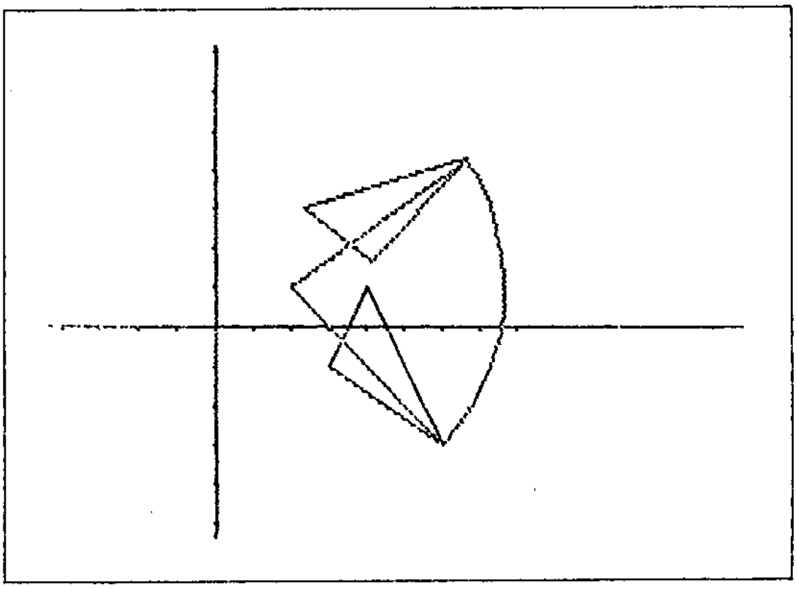


Fig. 1.37
A rotation
of 80°
about (2,1).

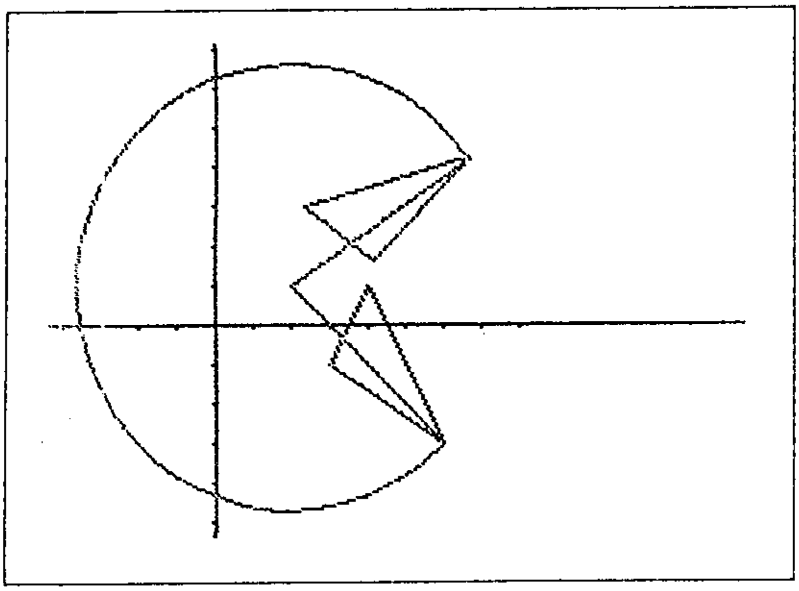


Fig. 1.38
The second
rotation
of 280°
about (2,1).

(b) A rotation of a° followed by a rotation of b° is equivalent to a rotation of $(a + b)^{\circ}$.

Discussion led to the conclusion that there is no relation between the three centres of rotation (unless all the rotations have the same centre) since the centre for the $(a + b)^{\circ}$ rotation varied with a and b. Figures 1.39, 1.40 and 1.41 illustrate this. The centre for the $(a + b)^{\circ}$ rotation (shown in red) has been found by construction.

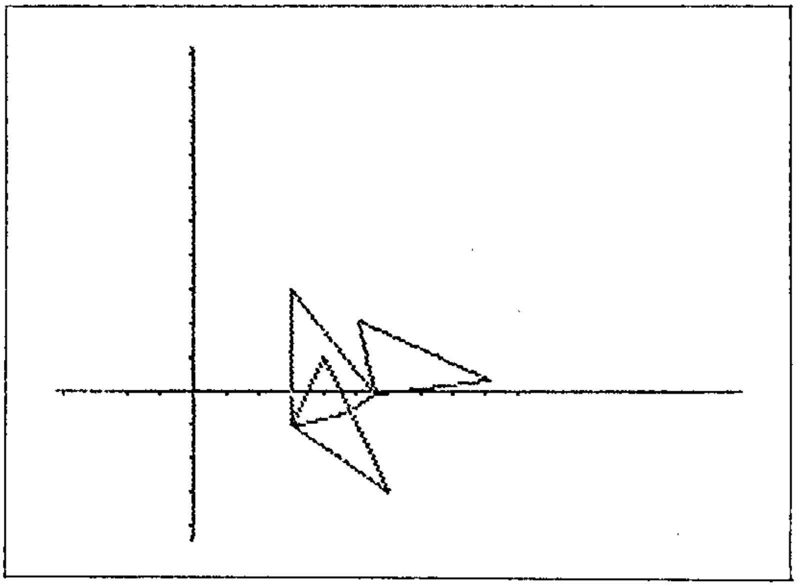


Fig. 1.39
A rotation
of 40°
about (3,3).

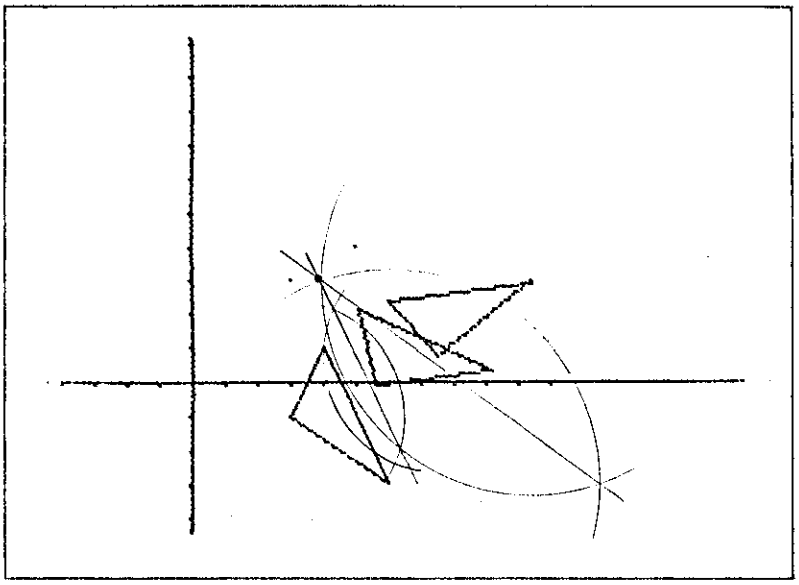


Fig. 1.40
A second
rotation
of 30°
about (5,4).

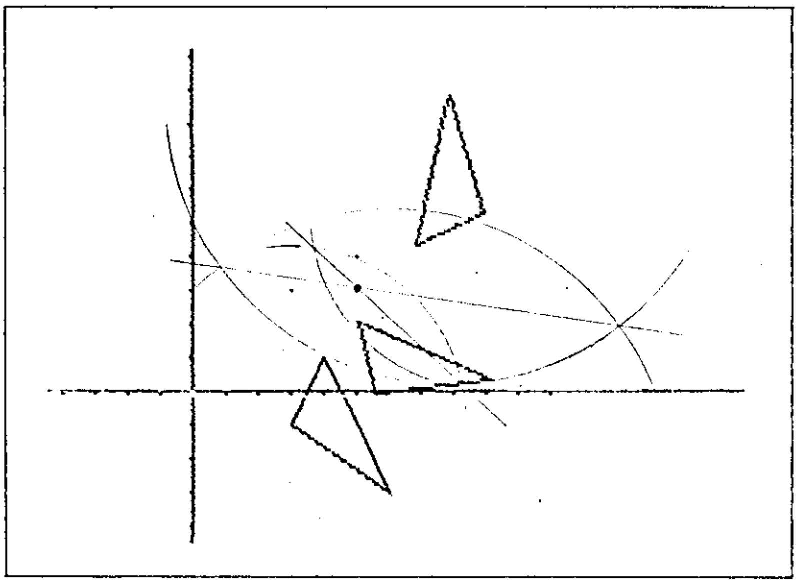


Fig. 1.41
A second
rotation
of 100°
about (5,4)

(c) A rotation of a° followed by a rotation of -a° about a different centre is equivalent to a translation.

Figures 1.42 and 1.43 show a rotation of 40° about (3,3), followed by a rotation of -40° about (10,3). The final image may also be obtained by translating the original triangle.

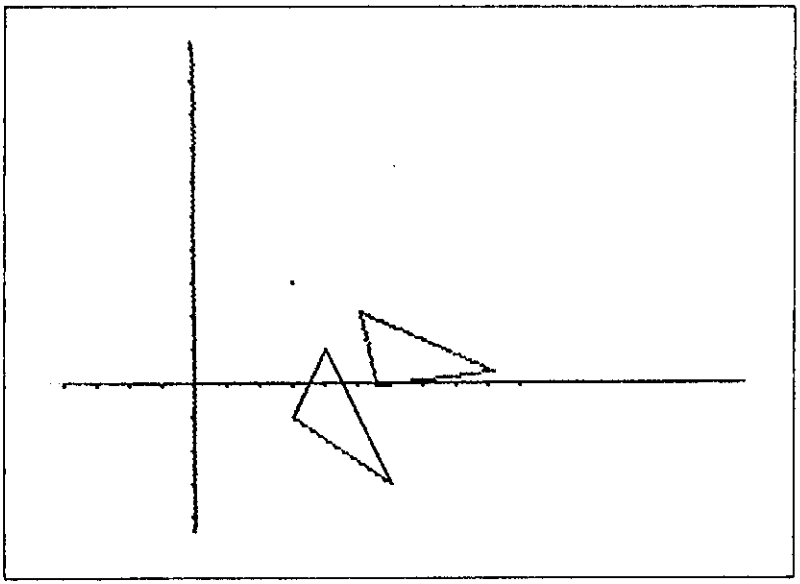


Fig. 1.42

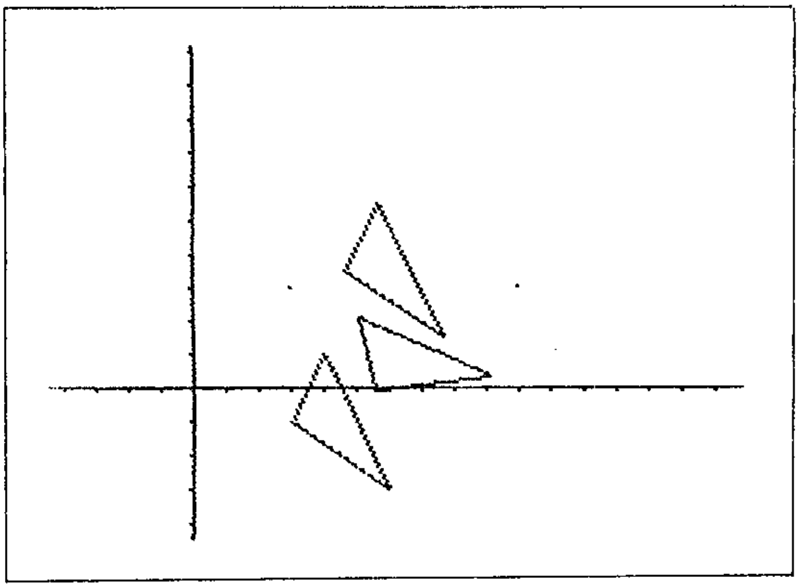


Fig. 1.43

Figures 1.44 and 1.45 show that by varying the angle a, 60° in this example, a different translation results, and so no relation between the vector joining the two centres and the translation could be found.

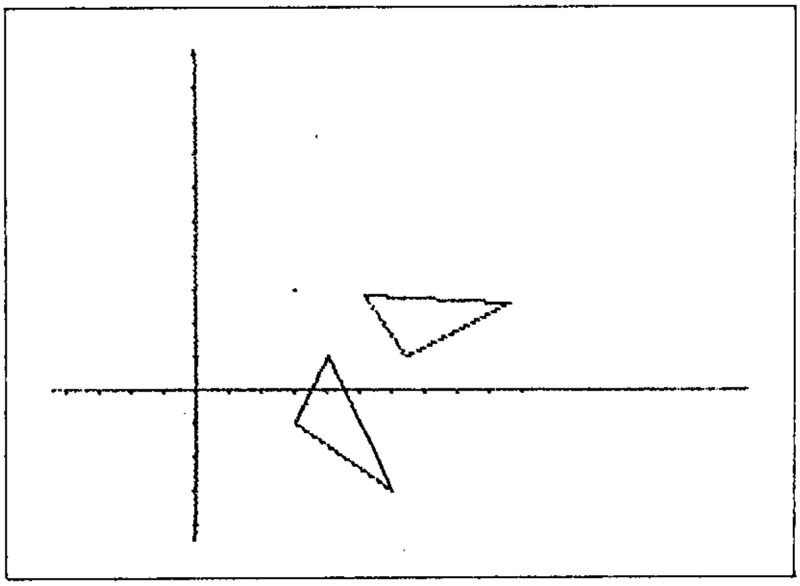


Fig. 1.44
A rotation
of 60°
about (3,3).

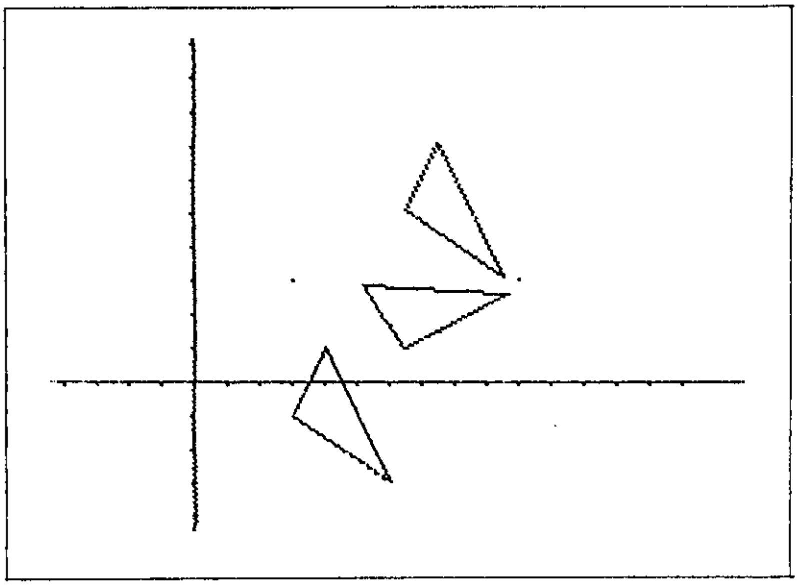


Fig. 1.45
A second
rotation
of -60°
about (10,3).

When a = 180, however, we obtain the following result -

(d) A half turn about X followed by a half turn about Y is equivalent to a translation $2\overrightarrow{XY}$ (see figures 1.46, 1.47 and 1.48).
Reversing the order of the rotations gives translation $2\overrightarrow{YX}$ (see figures 1.49, 1.50 and 1.51).

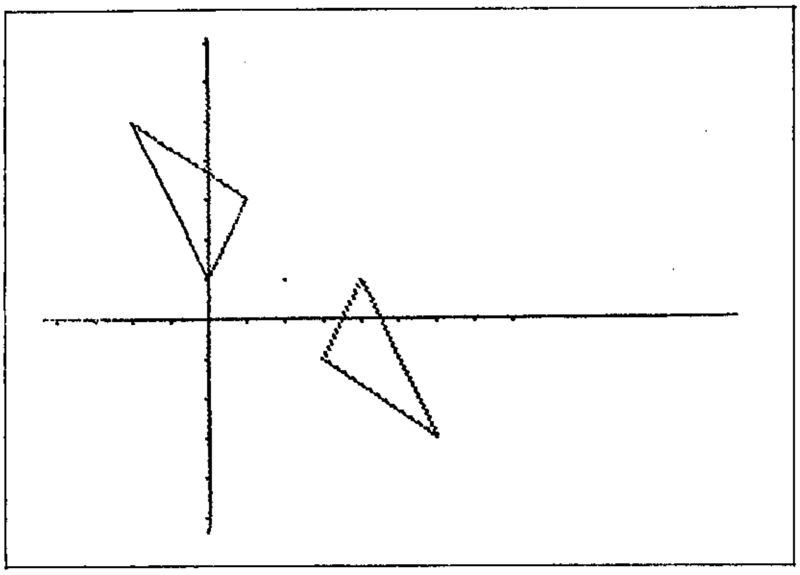


Fig. 1.46
A rotation
of 180°
about (2,1).

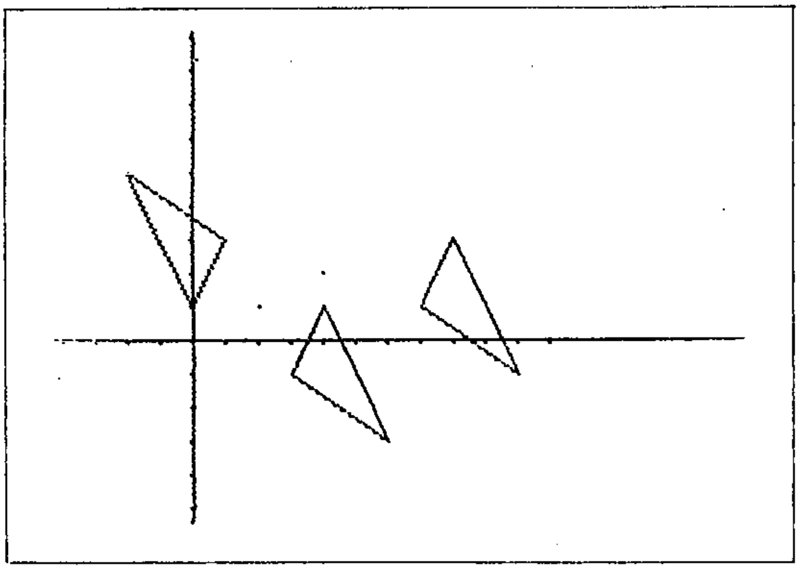


Fig. 1.47
A second
rotation
of 180°
about (4,2).

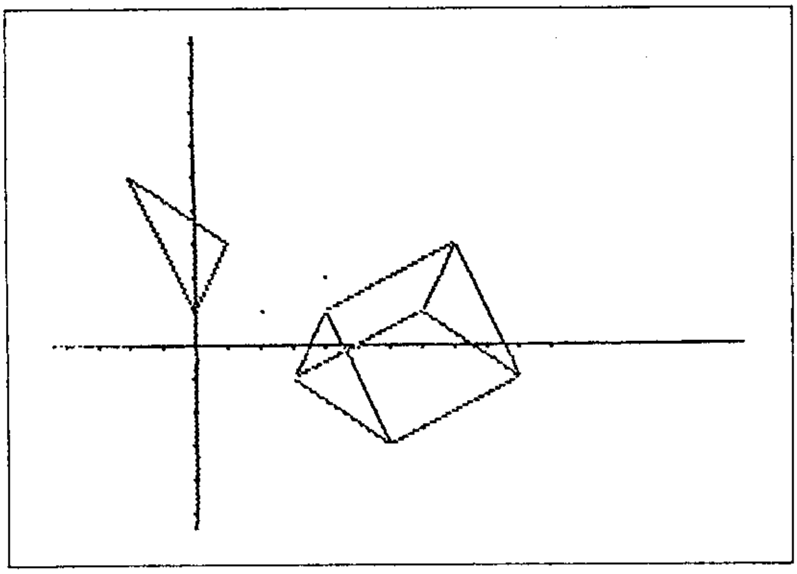


Fig. 1.48
The two
rotations are
equivalent to
a single
translation
$\begin{pmatrix}4\\2\end{pmatrix}$.

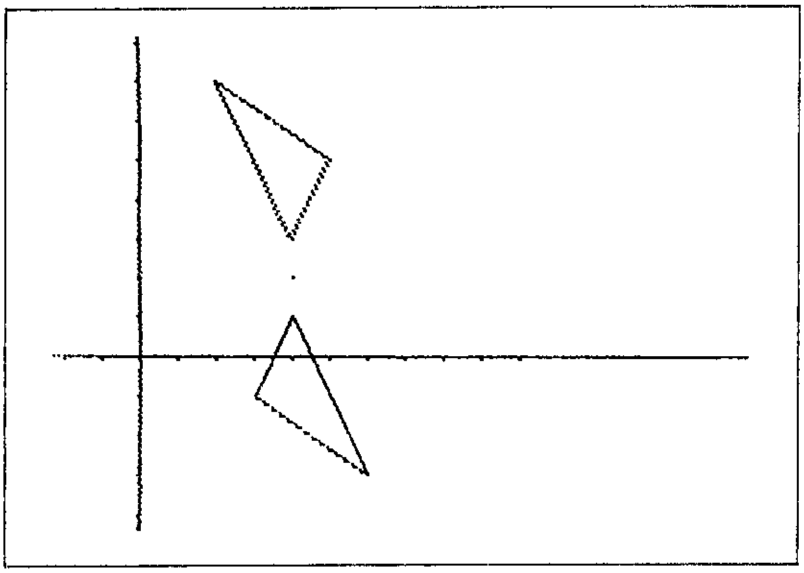


Fig. 1.49
A rotation
of 180°
about (4,2).

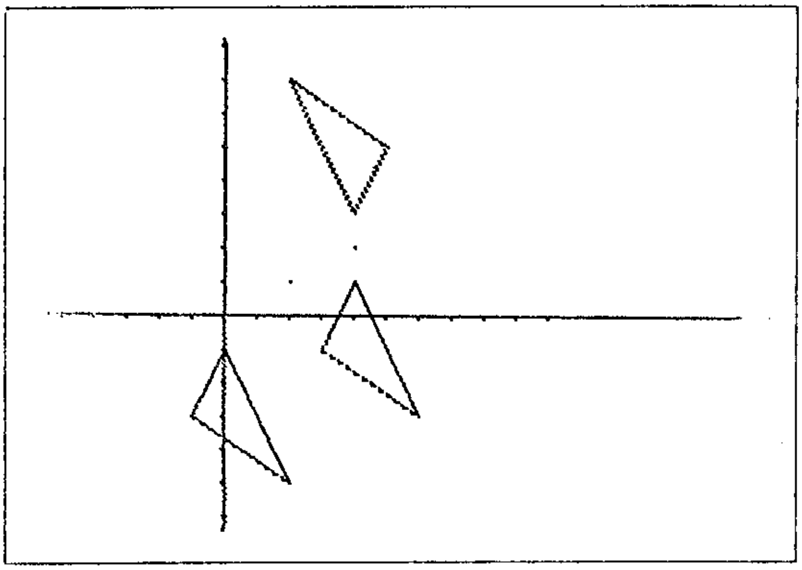


Fig. 1.50
A second
rotation
of 180°
about (2,1).

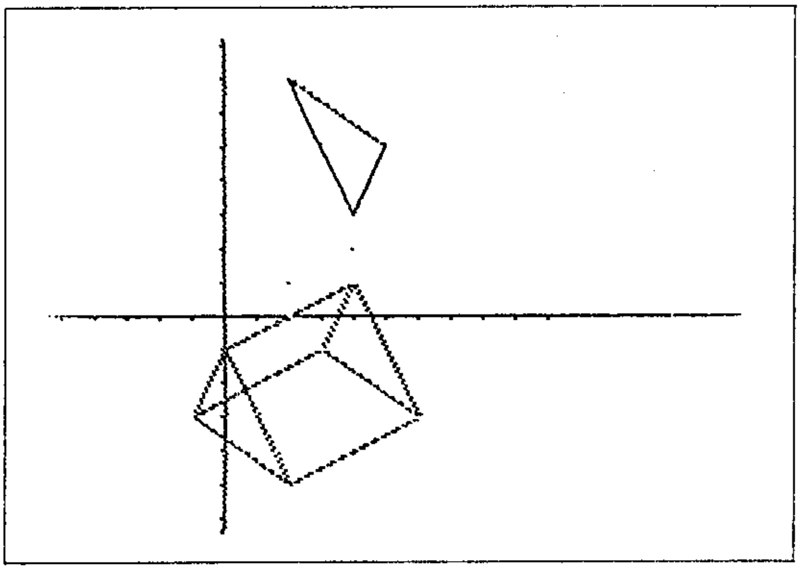


Fig. 1.51
The two
rotations are
equivalent to
a single
translation
$\begin{pmatrix}-4\\-2\end{pmatrix}$.

Reflection

More thought went into answering the question 'What single transformation is equivalent to a reflection followed by another reflection?', and it was agreed that a rotation or a translation was possible. The conclusions were as follows -

(a) A reflection followed by a reflection in the same mirror line is equivalent to the identity transformation.

(b) A reflection in two parallel lines, m_1 and m_2, is equivalent to a translation. The translation is perpendicular to the mirror lines and equal to twice the displacement from m_1 to m_2. Figures 1.52, 1.53 and 1.54 illustrate this. The axes are adjusted where necessary to accommodate the images.

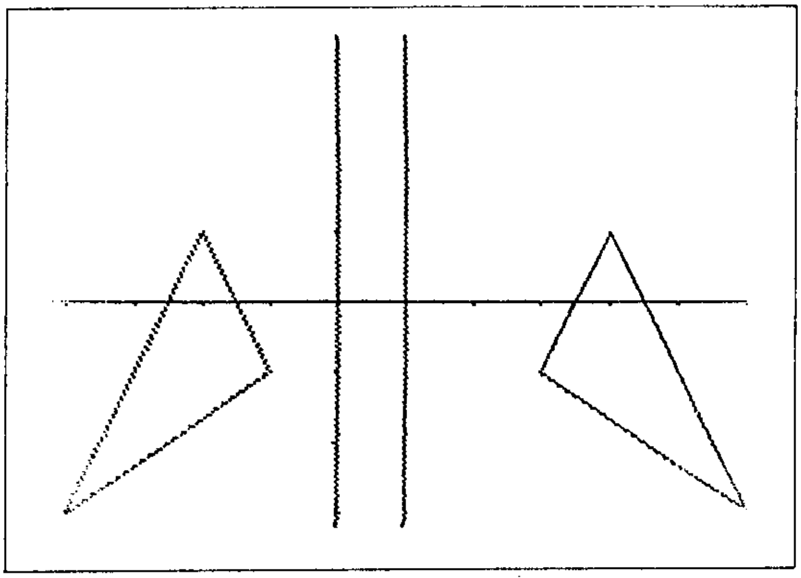


Fig. 1.52
A reflection
in the line
x = 1.

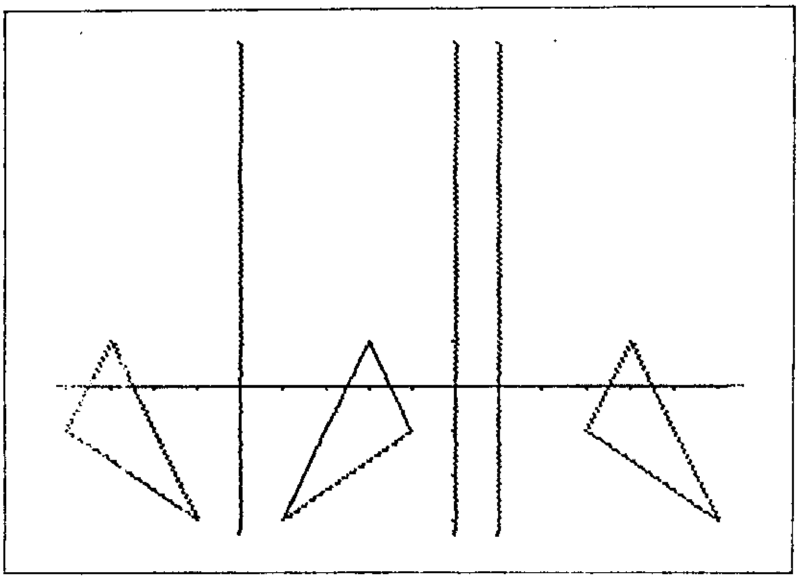


Fig. 1.53
A second
reflection
in the line
x = -5.

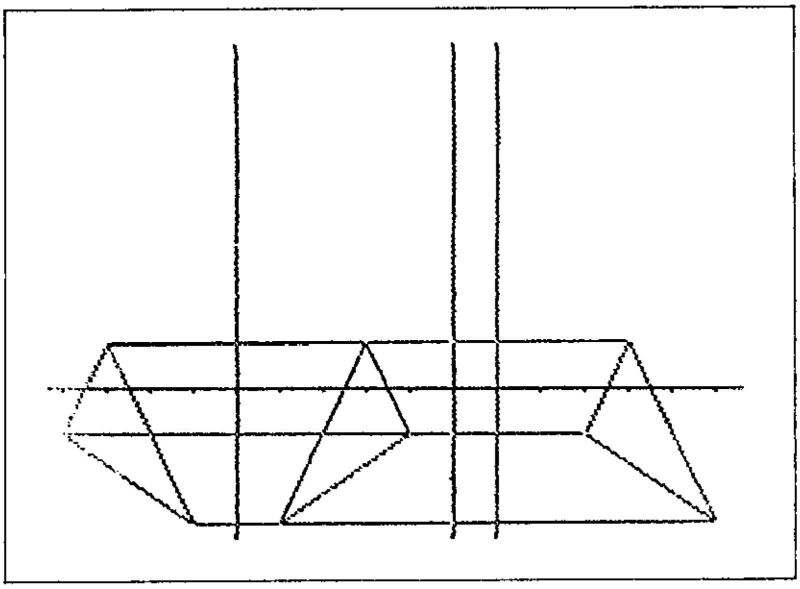


Fig. 1.54
The two reflections combine to give a translation $\begin{pmatrix}-12\\0\end{pmatrix}$.

Reversing the order of the reflections reverses the translation. Figures 1.55, 1.56 and 1.57 show that a reflection in the line x = -5, followed by a reflection in the line x = 1, is equivalent to a translation $\begin{pmatrix}12\\0\end{pmatrix}$.

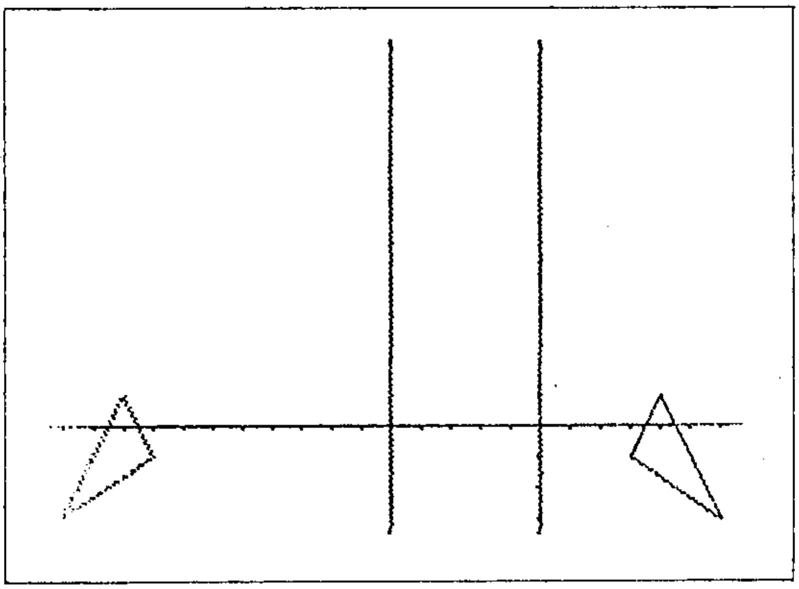


Fig. 1.55
A reflection in the line x = -5.

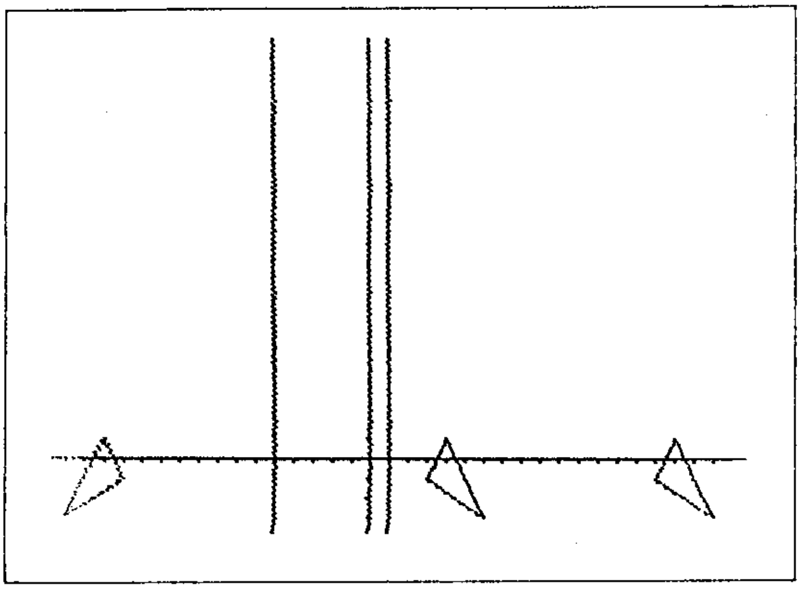


Fig. 1.56
A second reflection in the line $x = 1$.

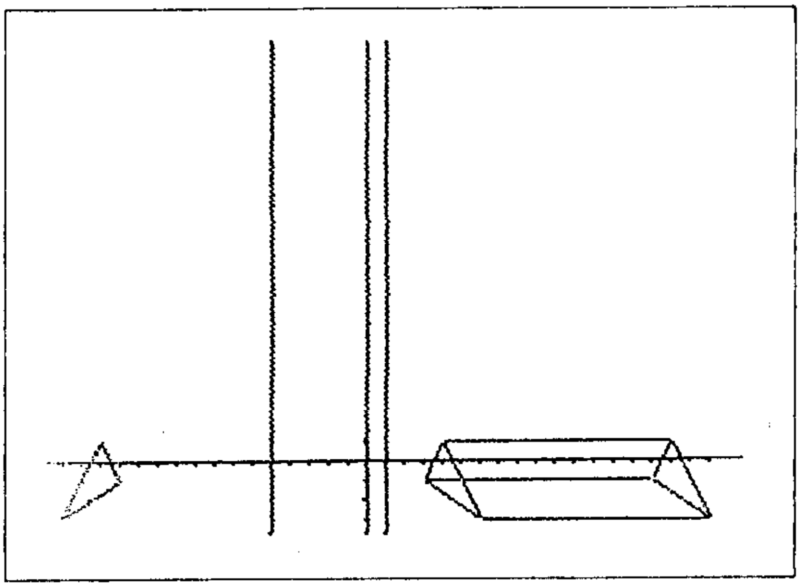


Fig. 1.57
The two reflections combine to give a translation $\begin{pmatrix}12\\0\end{pmatrix}$.

(c) A reflection in line m_1 followed by a reflection in line m_2, where m_1 and m_2 intersect, is equivalent to a rotation. The centre of the rotation is the point where m_1 and m_2 intersect. The angle of rotation is twice the angle between m_1 and m_2, in the sense of the rotation from m_1 to m_2.

Figures 1.58 to 1.60 show an example. Figures 1.61 to 1.63 use the same reflections but in the reverse order, which gives a rotation in the opposite direction.

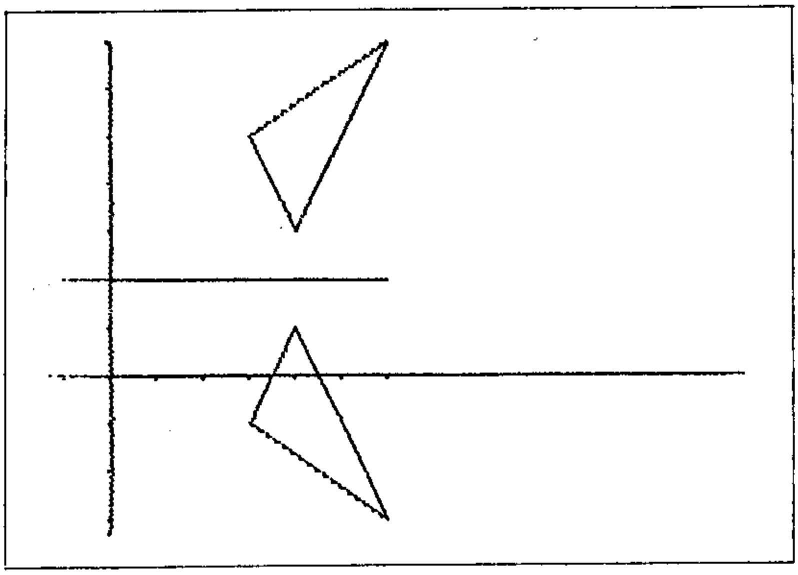


Fig. 1.58
A reflection
in the line
$y = 2$.

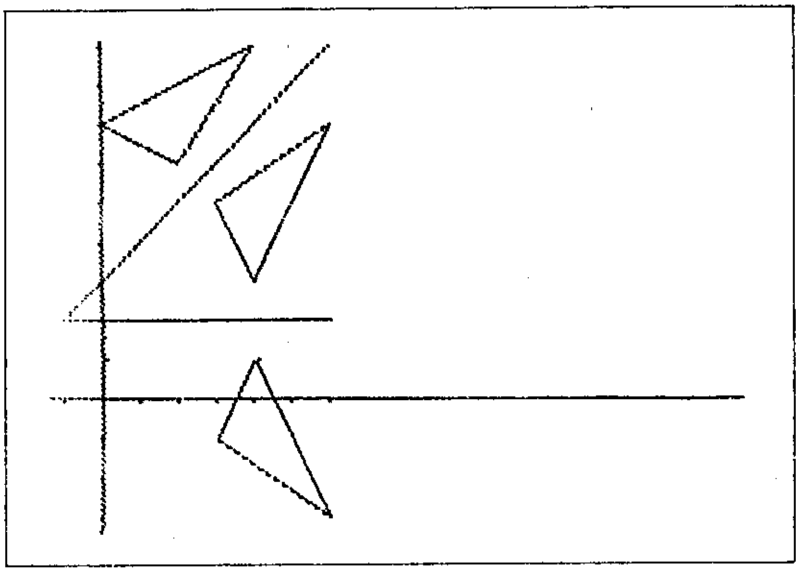


Fig. 1.59
A second
reflection
in the line
$y = x + 3$.

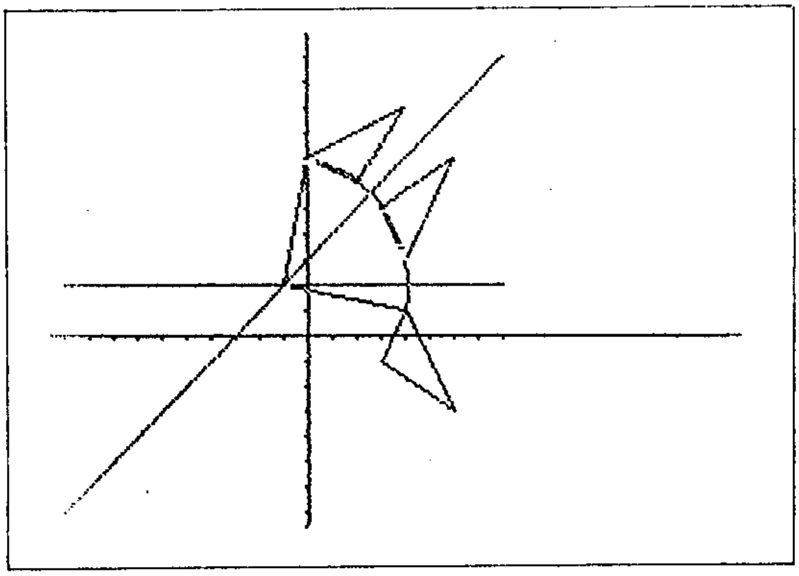


Fig. 1.60
The two
reflections
are equivalent
to a rotation
of 90°
about (-1,2).

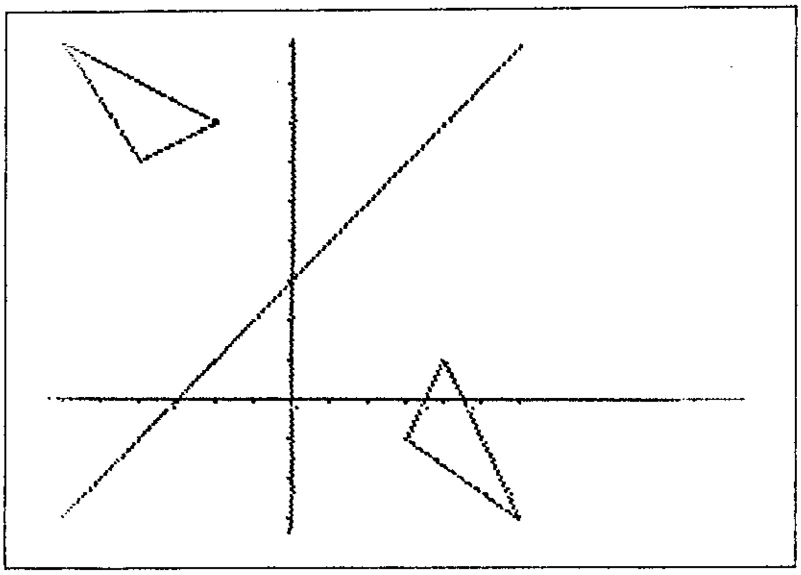


Fig. 1.61
A reflection
in the line
$y = x + 3$.

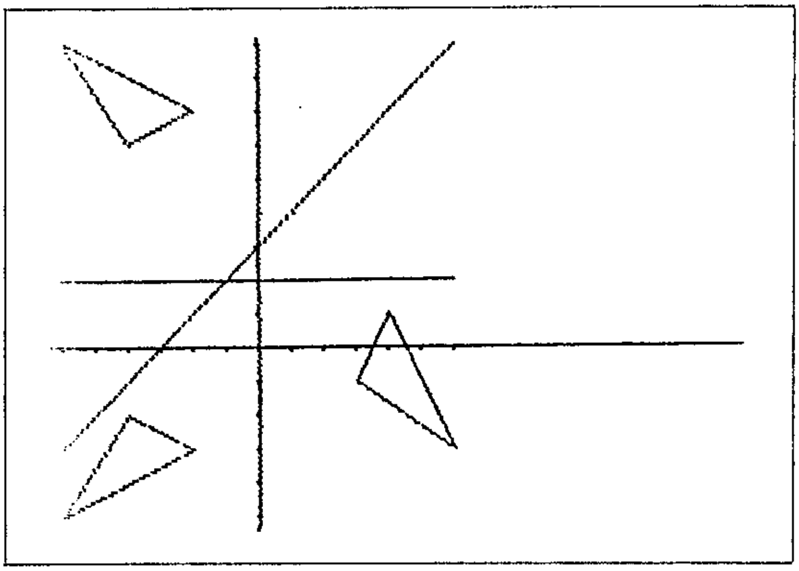


Fig. 1.62
A second
reflection
in the line
$y = 2$.

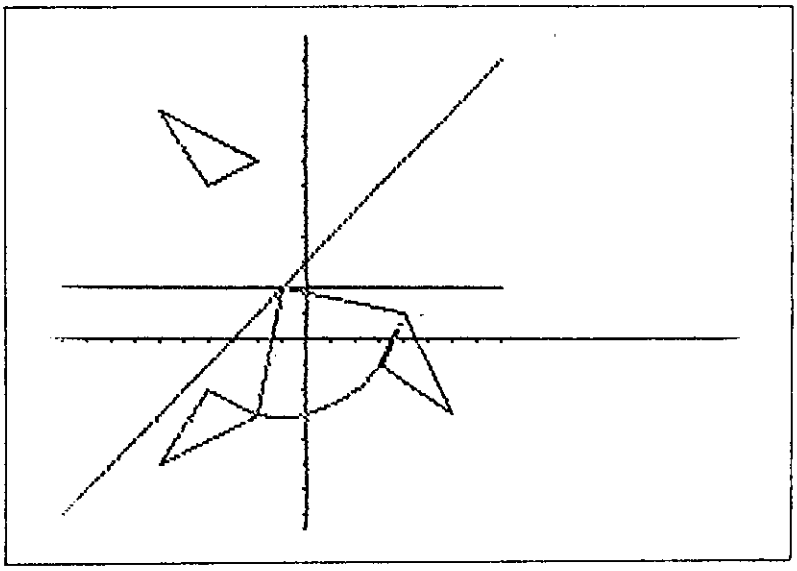


Fig. 1.63
The two
reflections
are equivalent
to a rotation
of -90°
about $(-1,2)$.

Enlargement

(a) I began the discussion on enlargements by considering the inverse of an enlargement with scale factor 2, centre (1,1).

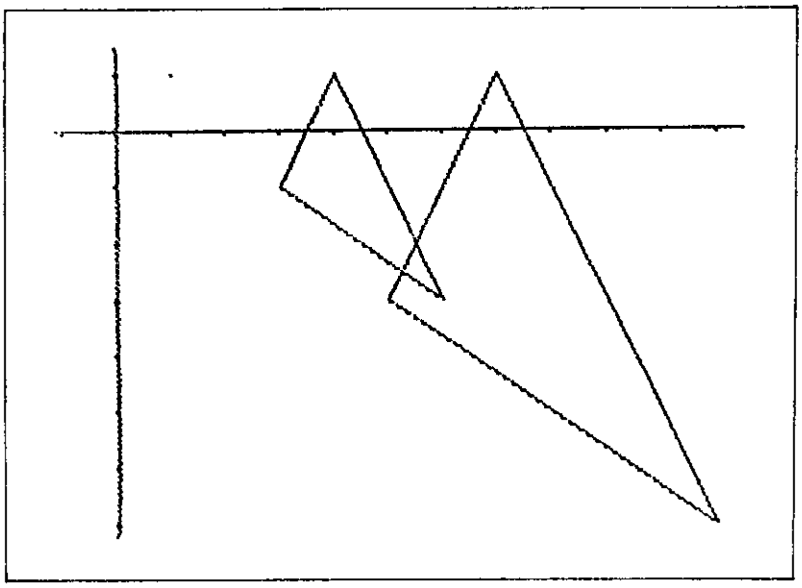


Fig. 1.64
An enlargement
scale factor 2,
centre (1,1).

It was agreed that the inverse would be an enlargement with centre (1,1). The suggestion of a scale factor of -2 for the inverse was demonstrated using the computer (see figure 1.65). I found that the ease with which a suggested answer could be tested and, if necessary, rejected encouraged the pupils to respond more readily. The investigational approach lessened the feeling of inadequacy at giving the wrong answer. Further discussion led to the method of finding the scale factor for the inverse.

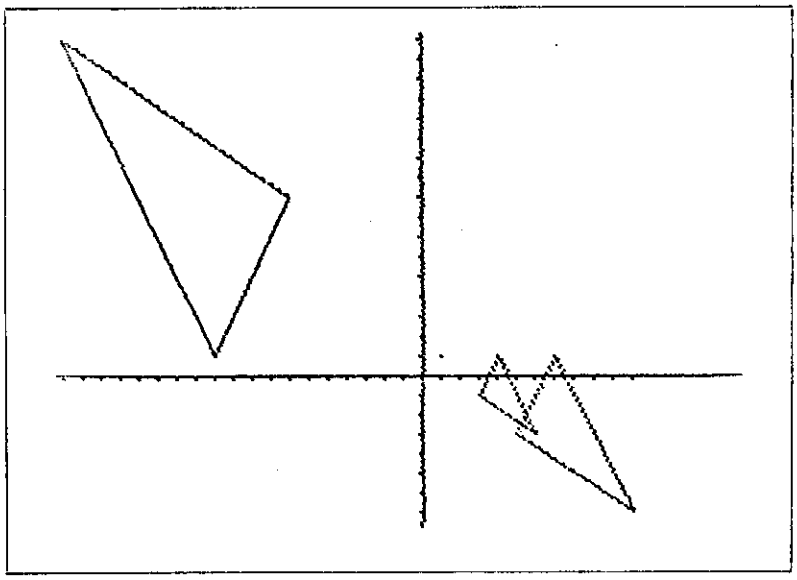


Fig. 1.65
A second
enlargement
scale factor -2,
centre (1,1)
does not map
the image onto
the original
object.

(b) An enlargement of scale factor 2, for example, followed by an enlargement of scale factor 3, with the same centre of enlargement, is equivalent to a single enlargement with the same centre. Figures 1.66 and 1.67 show the triangle with vertices (3,6), (4,8) and (6,6) after an enlargement scale factor 2, centre (1,6) followed by an enlargement scale factor 3, centre (1,6). (The scale has been altered in figure 1.67 to accommodate the second enlargement.)

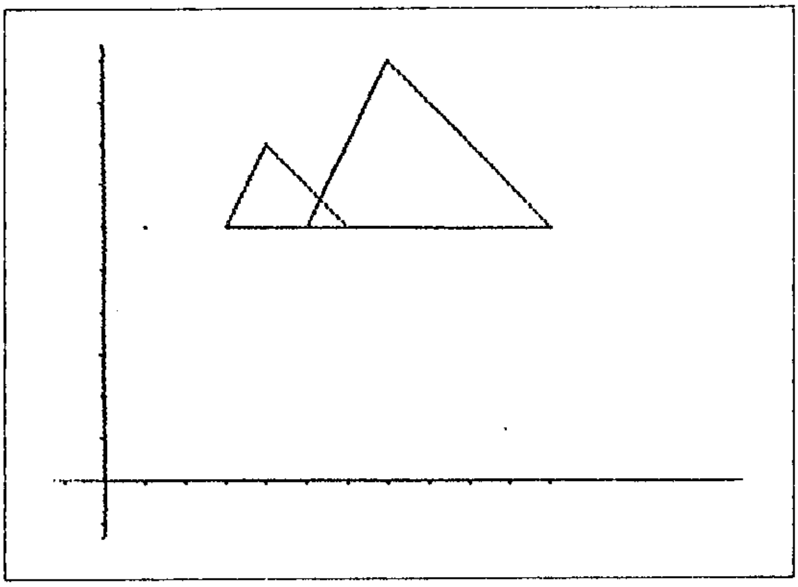


Fig. 1.66

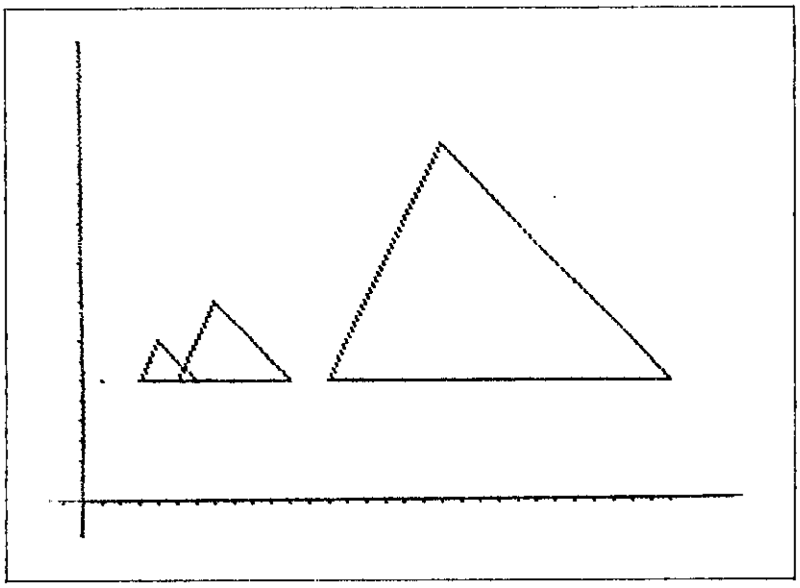


Fig. 1.67

The first suggestion was that the scale factor of the single enlargement was 5 (shown in figure 1.68) but it was soon realised that the product of 2 and 3 was required, not the sum.

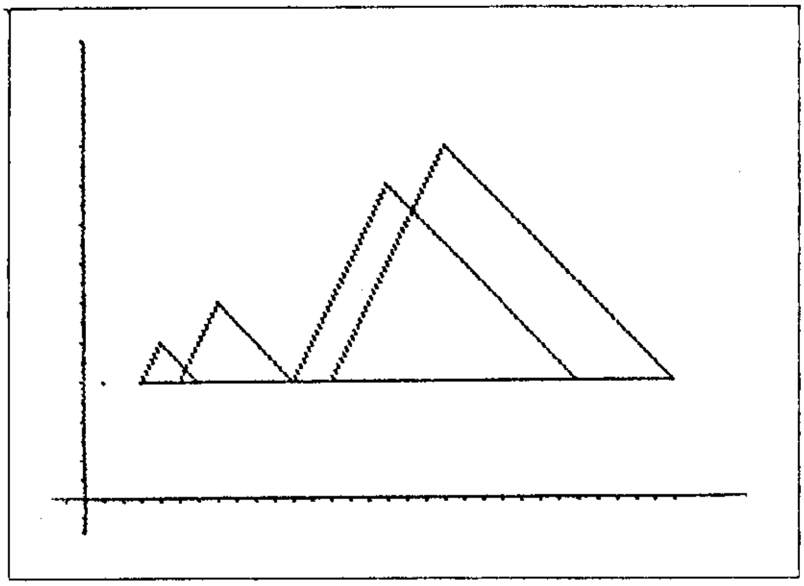


Fig. 1.68
The original triangle enlarged with scale factor 5, centre (1,6).

(c) Further investigation with different centres of enlargement suggested that when two enlargements combine to give a single enlargement, the three centres are collinear. The following series of diagrams shows the development.

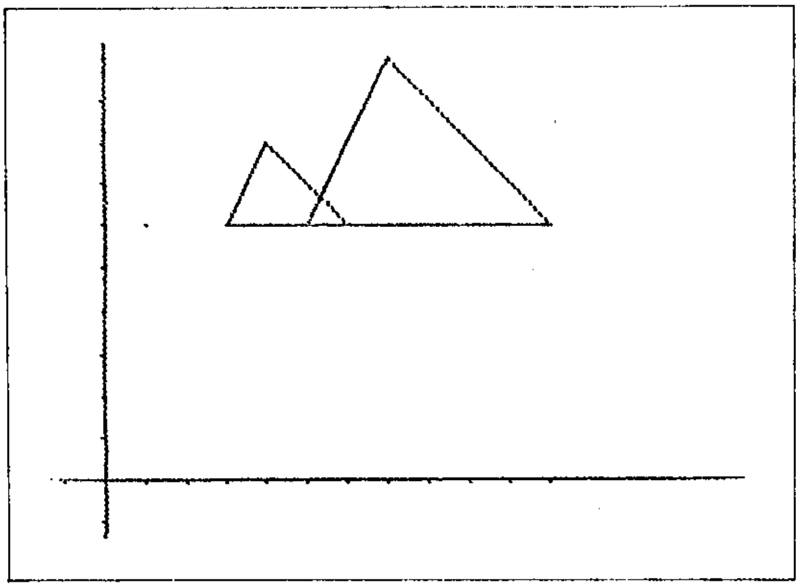


Fig. 1.69
An enlargement scale factor 2, centre (1,6).

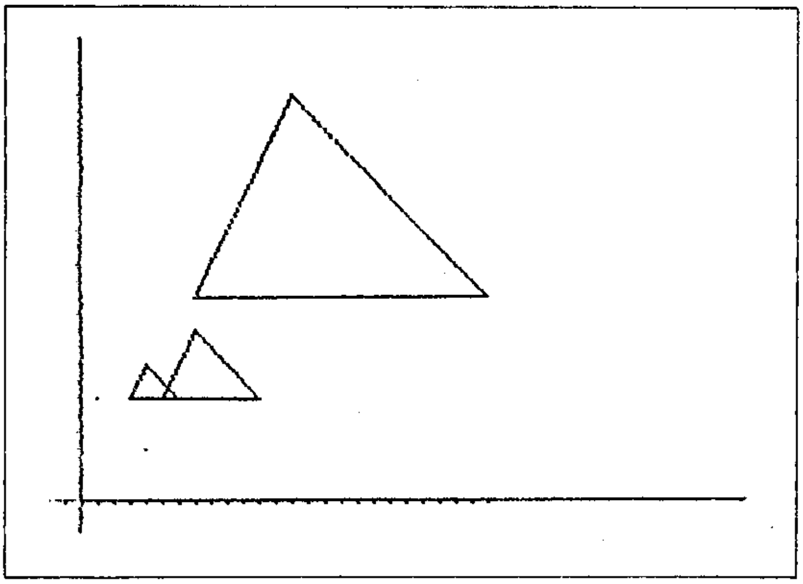


Fig. 1.70
A second
enlargement
scale factor 3,
centre (4,3).

The figure above suggested that the two enlargements were equivalent to a single enlargement, scale factor 6, whose centre lies on the line x + y = 7. The point (2,5) was tried as the centre of enlargement (see figure 1.71).

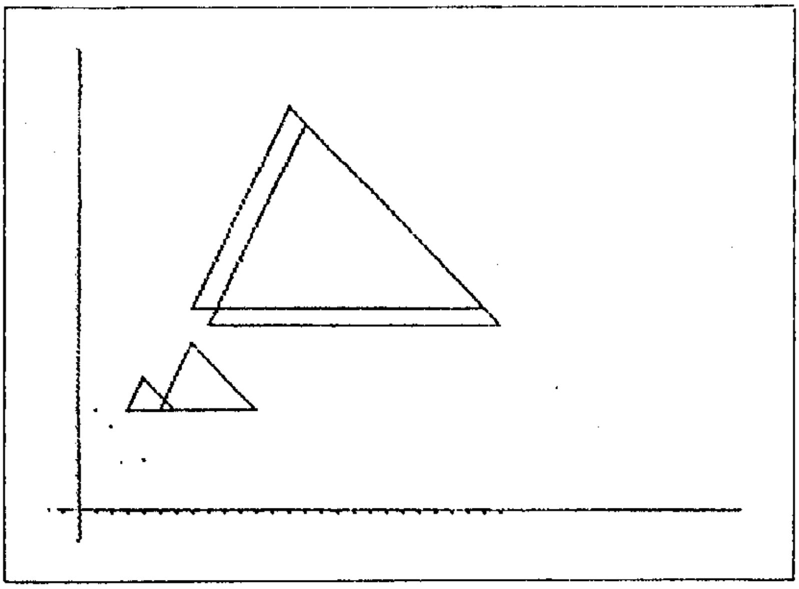


Fig. 1.71
The original
triangle
enlarged with
scale factor 6,
centre (2,5).

The original triangle was then enlarged with scale factor 6, using (2.1,4.9) as the centre of enlargement - this is shown in figure 1.72. (2.2,4.8) was found to be the required centre.

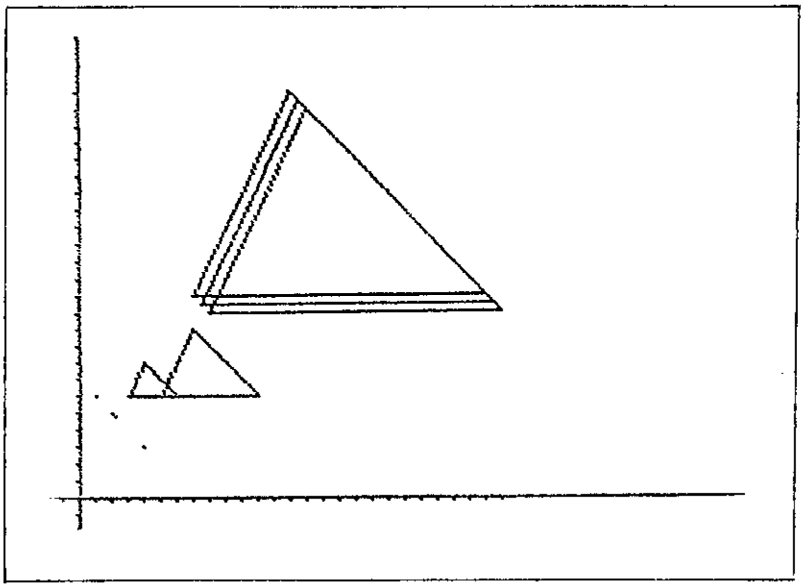


Fig. 1.72
A further enlargement of the original triangle using (2.1,4.9) as the centre.

The pupils then tried examples of their own on graph paper to confirm that the three centres were collinear. Figure 1.73 shows an enlargement scale factor 2 with centre (1,6), followed by an enlargement scale factor -3 with centre (4,3). The centre for the single enlargement which is equivalent to their combination has been found by construction (in red).

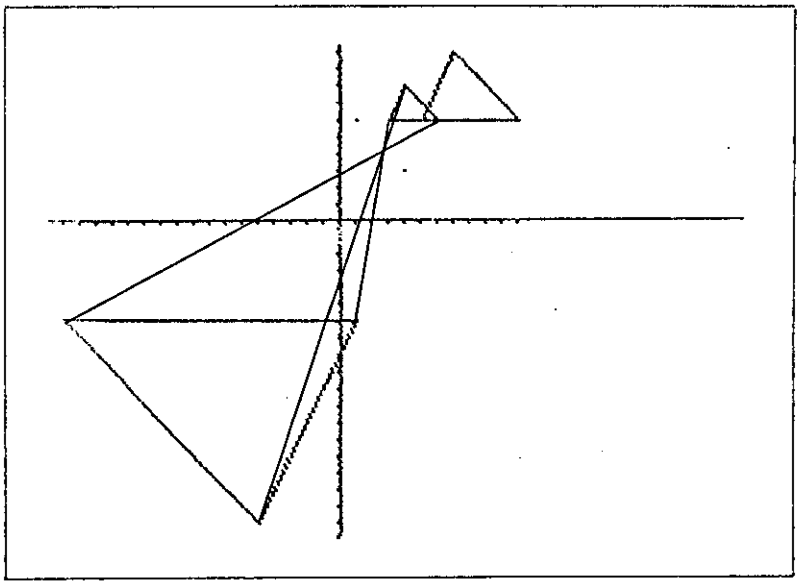


Fig. 1.73

No pupil noticed that (2.2,4.8) divides the line joining (1,6) and (4,3) in the ratio 2:3. A sixth form group would be capable of investigating this, using vector methods, to obtain the general rule:

An enlargement scale factor m, centre A, followed by an enlargement scale factor n, centre B, is equivalent to an enlargement scale factor mn, centre X, where $\overrightarrow{AX} = \frac{n-1}{mn-1}\overrightarrow{AB}$, $mn \neq 1$.

(d) I asked the class if there was an exception to the rule that the combination of two enlargements was equivalent to a single enlargement whose scale factor is the product of the two single scale factors. The identity transformation was suggested, and, after some prompting, a translation (i.e. when $mn = 1$ in the rule above). This was then investigated.

Figures 1.74 to 1.76 show that an enlargement centre $C_1(1,7)$ scale factor 2, followed by an enlargement centre $C_2(13,1)$ scale factor $\frac{1}{2}$, is equivalent to a translation $\frac{1}{2}\overrightarrow{C_1C_2}$, i.e. $\begin{pmatrix} 6 \\ -3 \end{pmatrix}$.

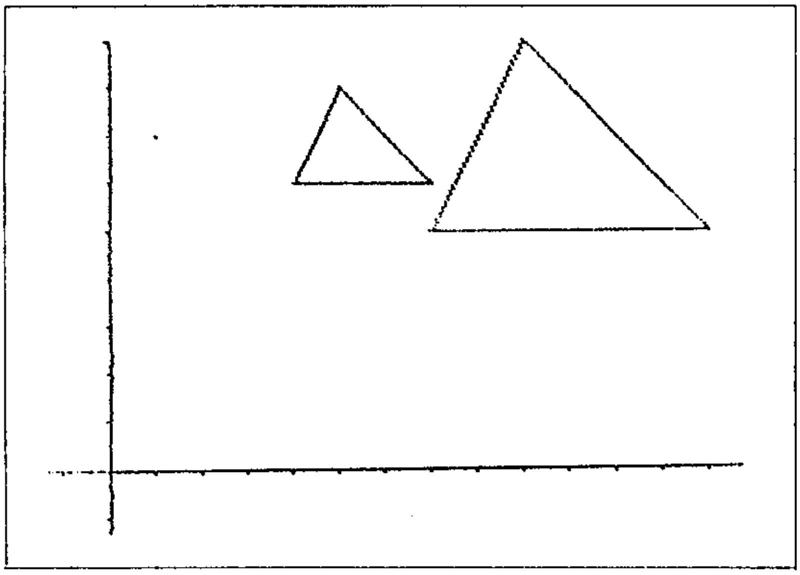


Fig. 1.74
An enlargement scale factor 2, centre (1,7), applied to the triangle with vertices (4,6), (5,8) and (7,6).

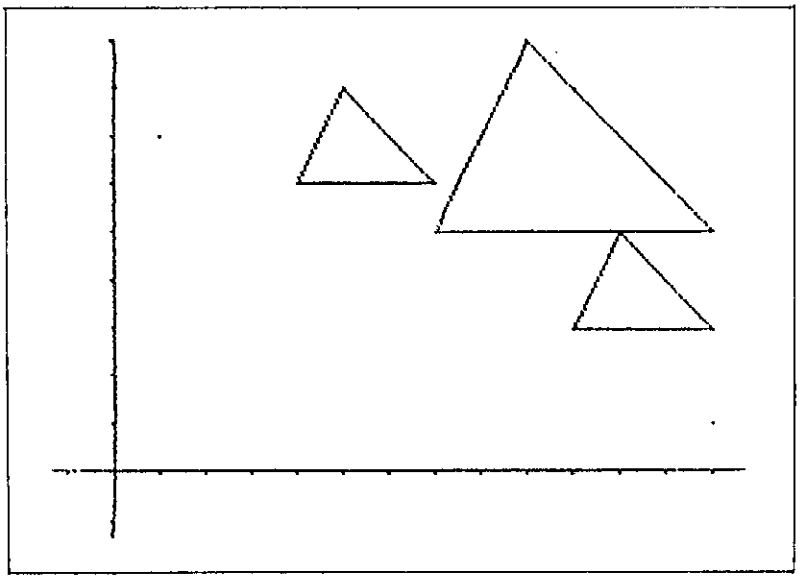


Fig. 1.75
A second enlargement scale factor $\frac{1}{2}$, centre (13,1).

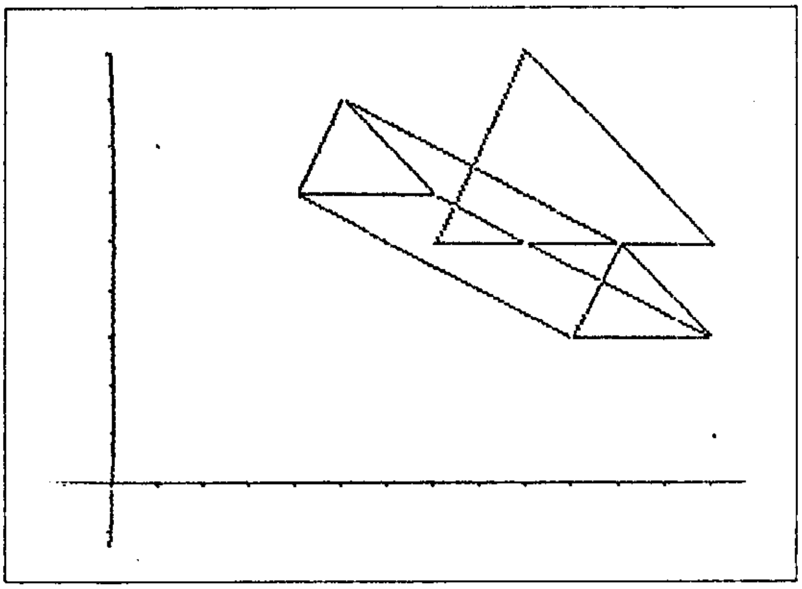


Fig 1.76
The two enlargements combine to give a single translation $\begin{pmatrix}6\\-3\end{pmatrix}$.

When asked what the effect of reversing the two scale factors would be, a translation of $\begin{pmatrix}-6\\3\end{pmatrix}$ was suggested, but it was found to be $\begin{pmatrix}-12\\6\end{pmatrix}$, as shown in figures 1.77 to 1.79. (The scale of the axes is changed in figure 1.78 to accommodate the second image.)

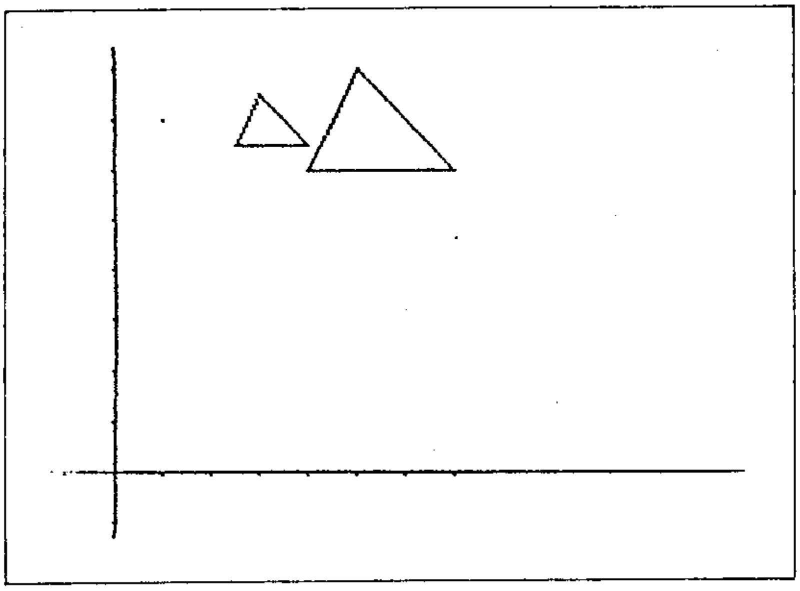


Fig. 1.77
An enlargement scale factor $\frac{1}{2}$, centre (1,7).

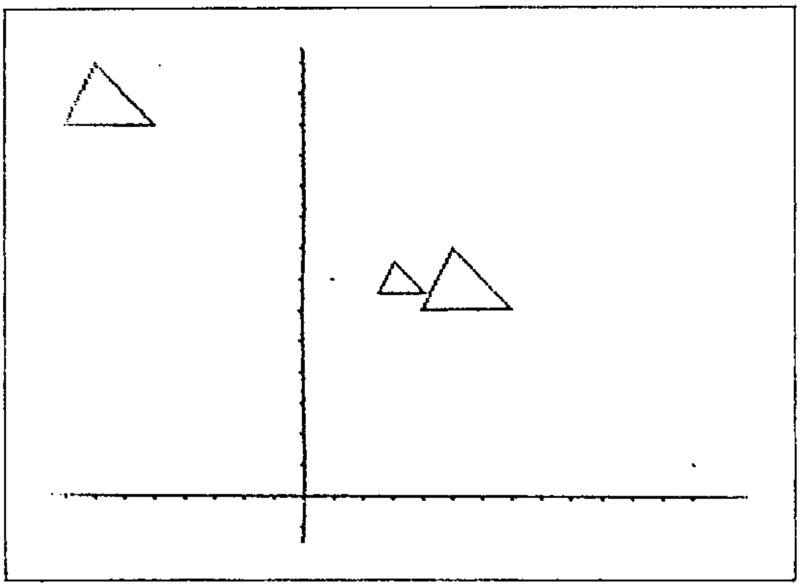


Fig. 1.78
A second
enlargement,
scale factor 2,
centre (13,1).

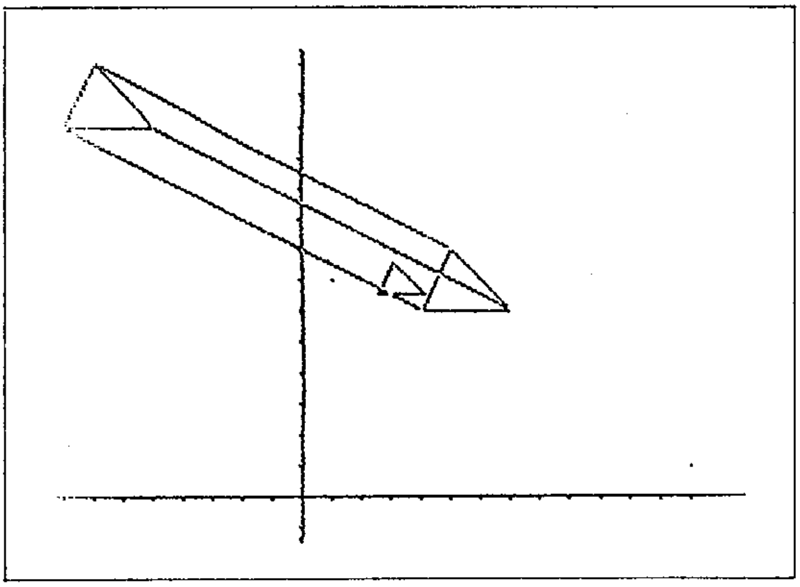


Fig. 1.79
The two
enlargements
combine to give
a single
translation
$\begin{pmatrix}-12\\6\end{pmatrix}$.

The series of diagrams in figures 1.80 to 1.82 shows a similar result after an enlargement with scale factor $\frac{1}{3}$ followed by an enlargement with scale factor 3. Figures 1.83 to 1.85 show an enlargement with scale factor 3 followed by an enlargement with scale factor $\frac{1}{3}$.

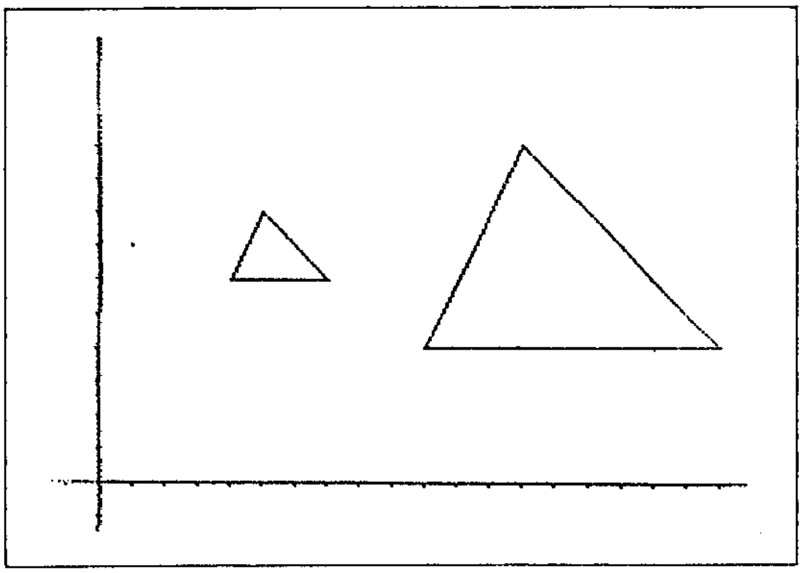


Fig. 1.80
An enlargement
scale factor 3,
centre (1,7).

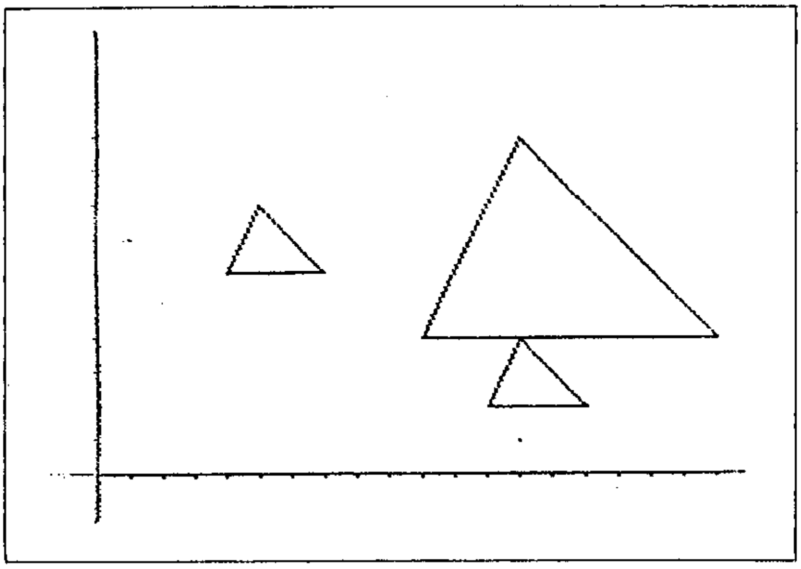


Fig. 1.81
A second
enlargement
scale factor $\frac{1}{3}$,
centre (13,1).

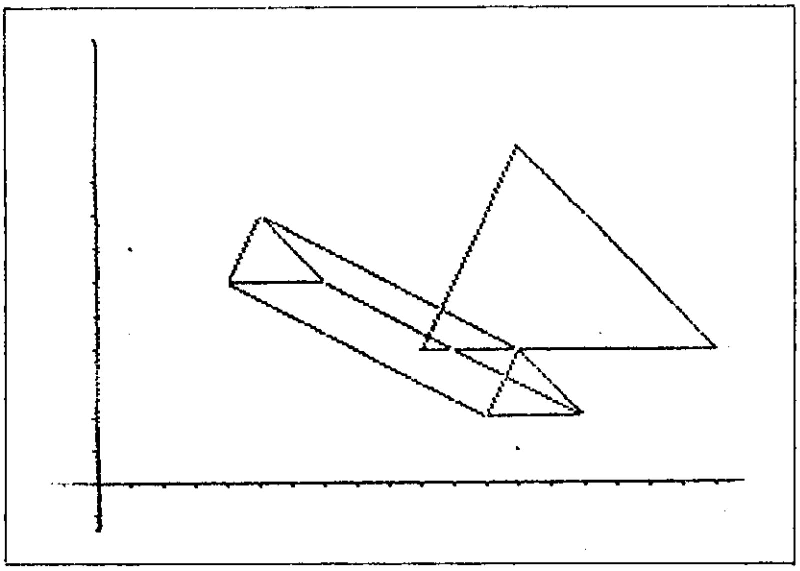


Fig 1.82
The two
enlargements
combine to give
a single
translation
$\begin{pmatrix} 8 \\ -4 \end{pmatrix}$.

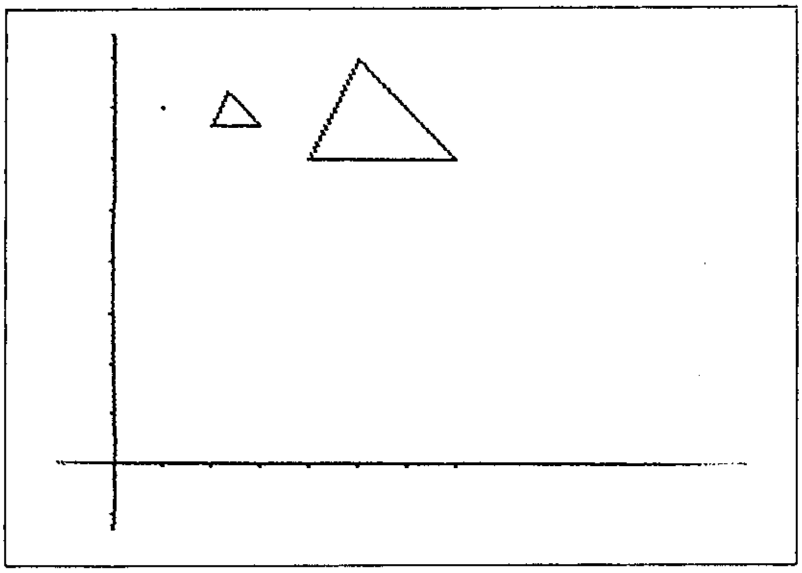


Fig. 1.83
An enlargement
scale factor $\frac{1}{3}$,
centre (1,7).

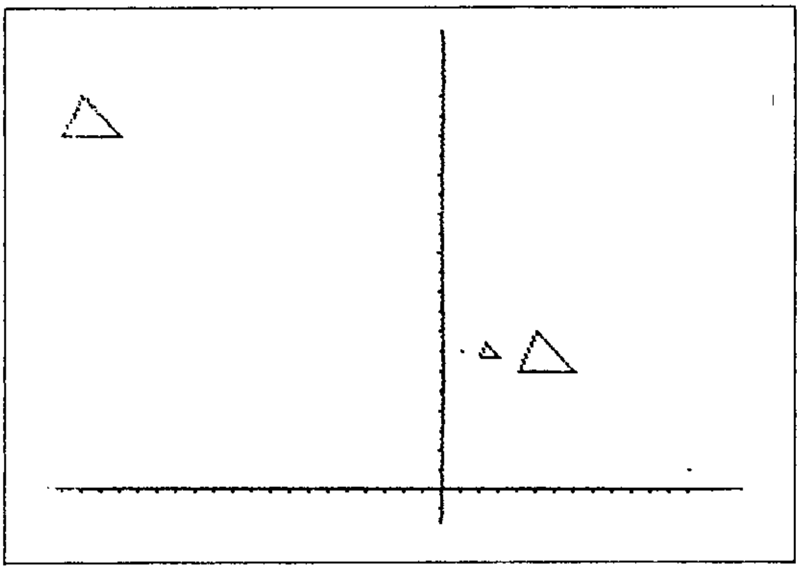


Fig. 1.84
A second
enlargement
scale factor 3,
centre (13,1).

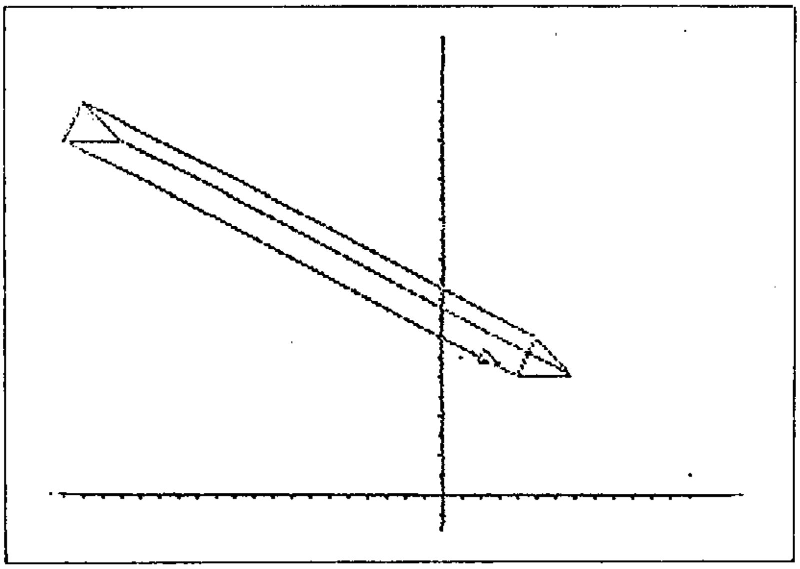


Fig. 1.85
The two
enlargements
combine to give
a single
translation
$\begin{pmatrix}-24\\12\end{pmatrix}$.

Although we did not investigate negative scale factors, it was found that, for $n > 0$, an enlargement scale factor n with centre C_1 followed by an enlargement scale factor $\frac{1}{n}$ with centre C_2 was equivalent to a translation described by the vector $(1 - \frac{1}{n})\overrightarrow{C_1C_2}$.

At this stage, I felt that a number of pupils in the group were beginning to find the work too difficult, but I think that further investigation by 6th form pupils would be valuable. The proof of the theorem stated above would appear ideally suited to vector methods; the discovery in the previous section c could be investigated using the ratio theorem.

If we consider the three triangles in figure 1.75 (shown again in figure 1.86, labelled A, B and C for clarity) it is possible to obtain similar results to those above by considering:

(i) the enlargement which is equivalent to the translation from A to C, followed by an enlargement which maps C onto B,

(ii) the enlargement which is equivalent to an enlargement which maps B onto C, followed by a translation from C onto A,

(iii) the translation which maps A onto C, where A and C are the images of B under enlargements with the same scale factor but with different centres of enlargement.

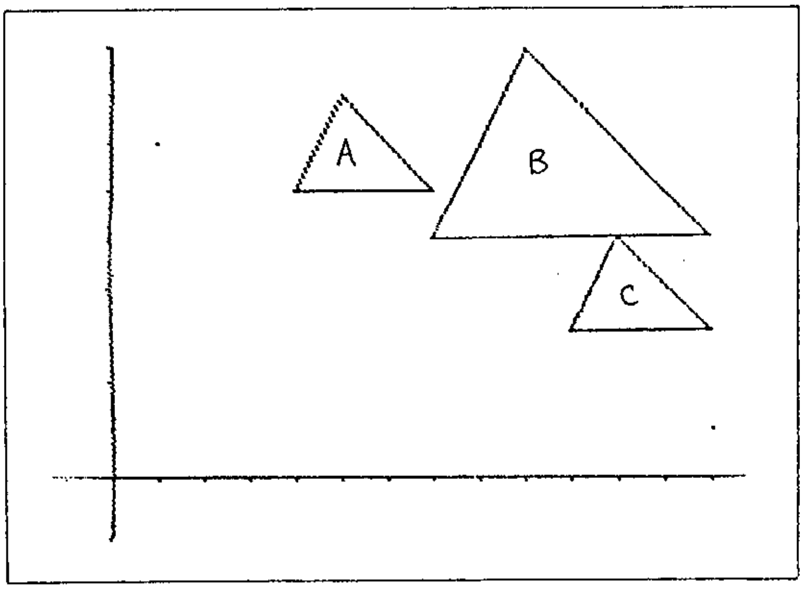


Fig. 1.86

Stretch

I have included this transformation in the program because it appears in some G.C.E. 'O' level texts, but I have not used it with pupils. It is possible to use the program to demonstrate the properties of a stretch, i.e. that points are transformed perpendicular to the invariant line in proportion to their distances from it, the constant of proportionality being the scale factor. Two-way stretches may be shown as the combination of two one-way stretches (see figures 1.87 and 1.88).

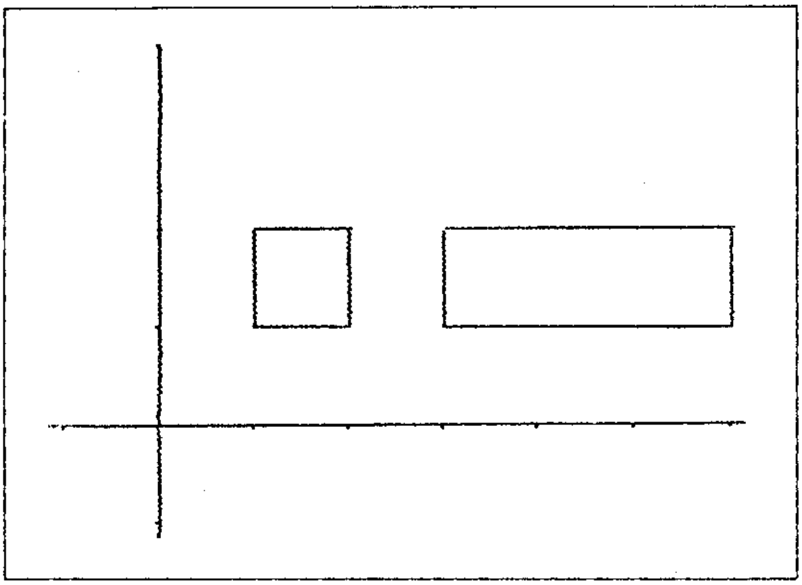


Fig. 1.87
A stretch from the y-axis with scale factor 3, applied to a square with vertices (1,1), (2,1), (2,2) and (1,2).

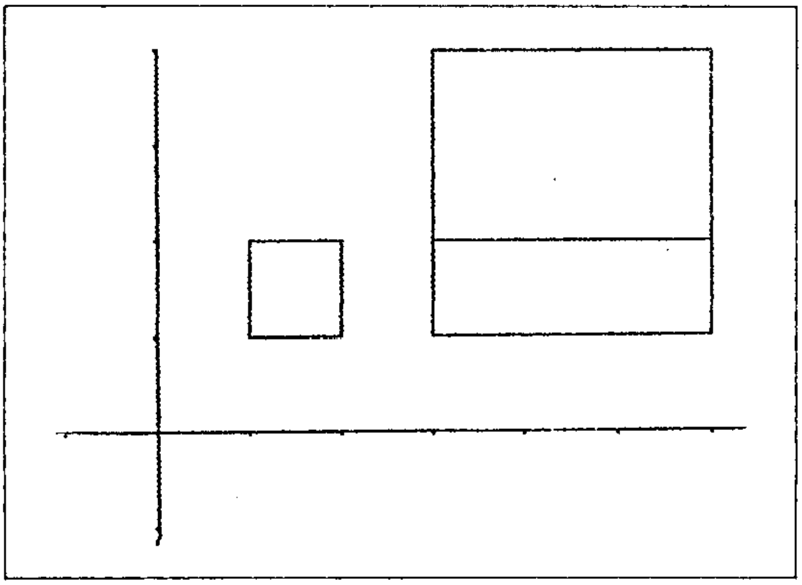


Fig. 1.88
The result of a second stretch from the x-axis with scale factor 2.

The combination of two stretches with the same scale factor and perpendicular invariant lines is equivalent to an enlargement with the same scale factor, and centre where the invariant lines cross. Figure 1.89 shows a stretch from the y-axis with scale factor 3, followed by a stretch from the x-axis with scale factor 3. The red lines have been drawn to show that the combination is equivalent to an enlargement, centre (0,0), scale factor 3.

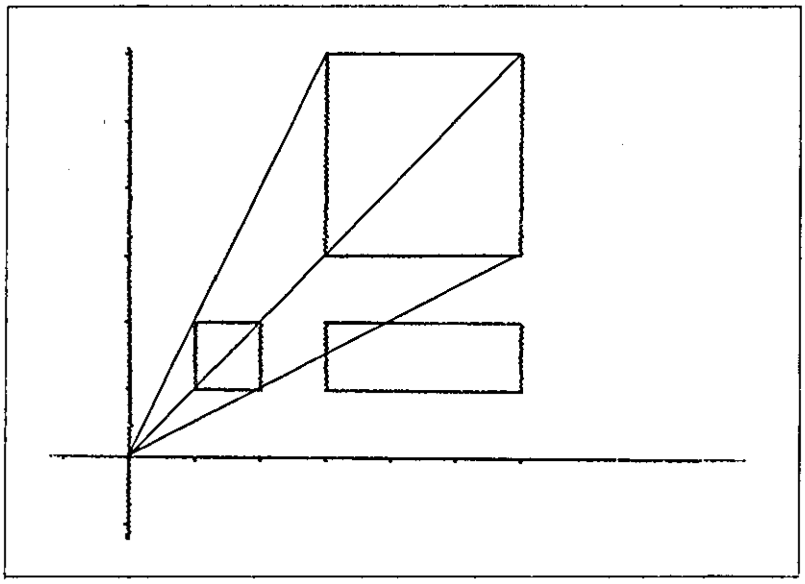


Fig. 1.89

Shear

This is undoubtedly the most difficult of the transformations for the children to understand, and it is becoming less common for 'O' level syllabuses to include the shear. I think that the main problem is that children are unable to easily represent the transformation in a concrete way, and so the concept of a shear becomes an abstract one. The S.M.P. text introduces the shear as the transformation of a pile of thin exercise books, as shown in figure 1.90.

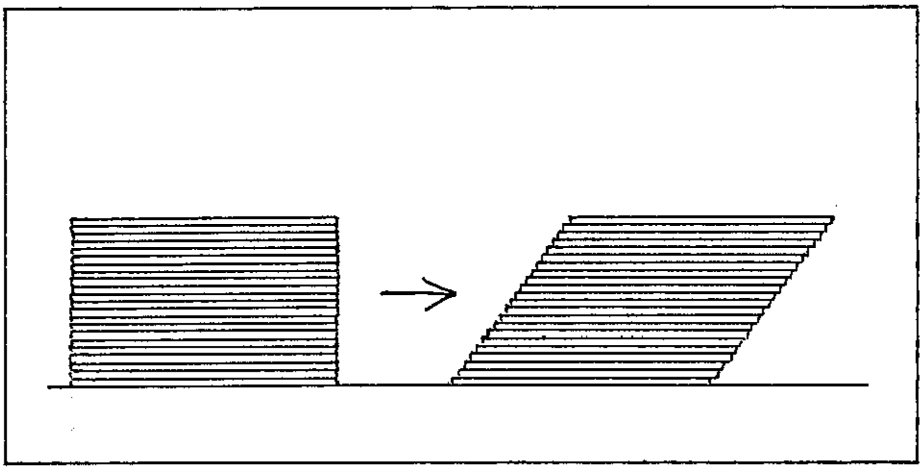


Fig. 1.90

This idea can easily be transferred to a shear where the invariant line is part of the object, as in figure 1.91, but understanding becomes noticeably more difficult when the invariant line lies outside the object, as in figure 1.92, or passes through the object, as in figure 1.93. I found that using the computer to illustrate various shears helped the pupils to understand the transformation better. It helped to reinforce the properties that only points on the invariant line remain unchanged, points shear in proportion to their distances from the invariant line, and points on opposite sides of the invariant line shear in opposite directions.

The object for all the shears in this section is the rectangle whose coordinates are (2,6), (2,8), (5,8) and (5,6).

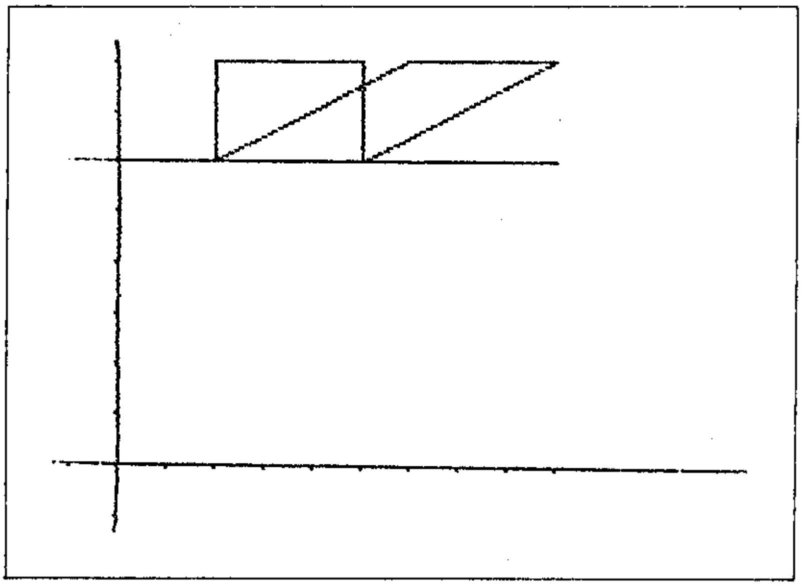


Fig. 1.91
A shear with
invariant line
$y = 6$ and
$(2,7) \rightarrow (4,7)$.

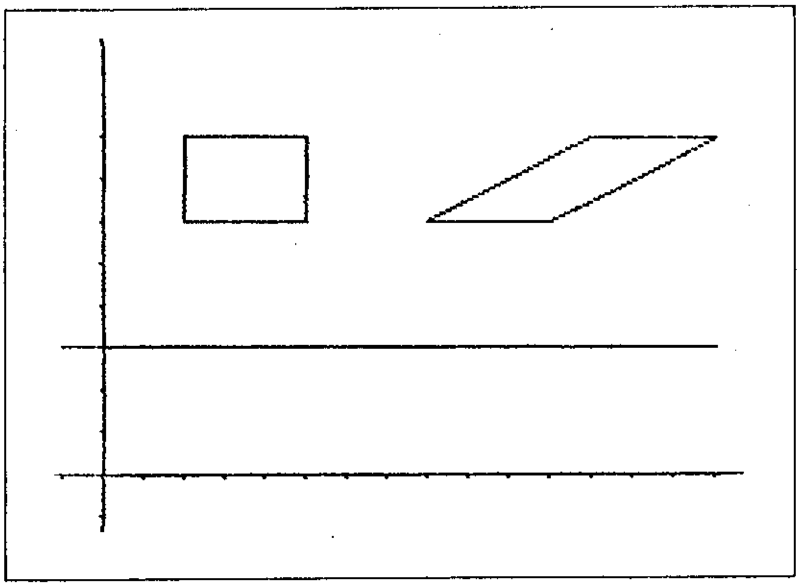


Fig. 1.92
A shear with
invariant line
y = 3 and
(1,4)→(3,4).

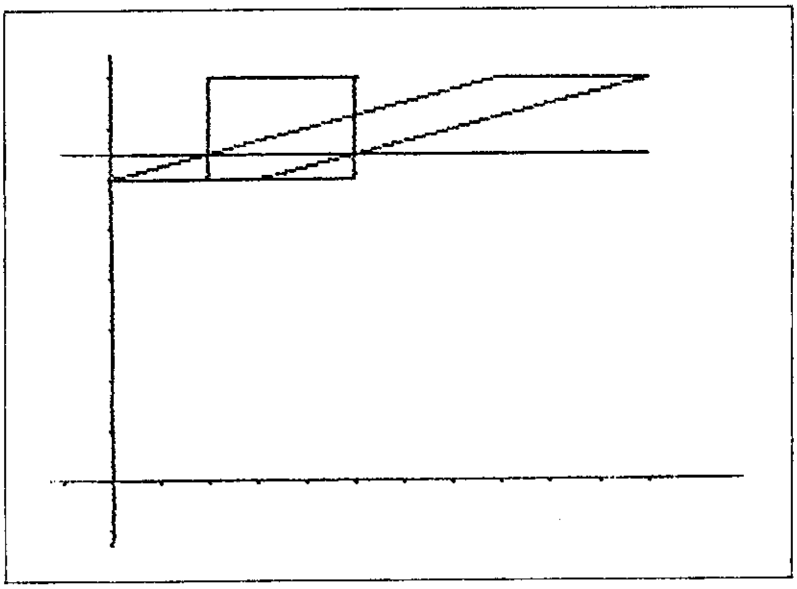


Fig. 1.93
A shear with
invariant line
y = $6\frac{1}{2}$ and
(2,7)→(4,7).

Another difficulty which I have encountered with pupils is in constructing an image for a given object and shear. The computer program will demonstrate this construction (which can be justified using similar triangles) and may also be used to demonstrate the combination of shears, but I consider that such a study is not suitable for pupils below sixth form level.

I conclude this section by illustrating some possible results which may be found when investigating the combination of shears.

(a) Two shears with the same invariant line are equivalent to the identity or to a single shear with the same invariant line.

Figures 1.94 and 1.95 illustrate this, and also show a further difficulty of having an invariant line which is not horizontal; this requires a level of understanding far greater than the exercise books analogy of figure 1.90.

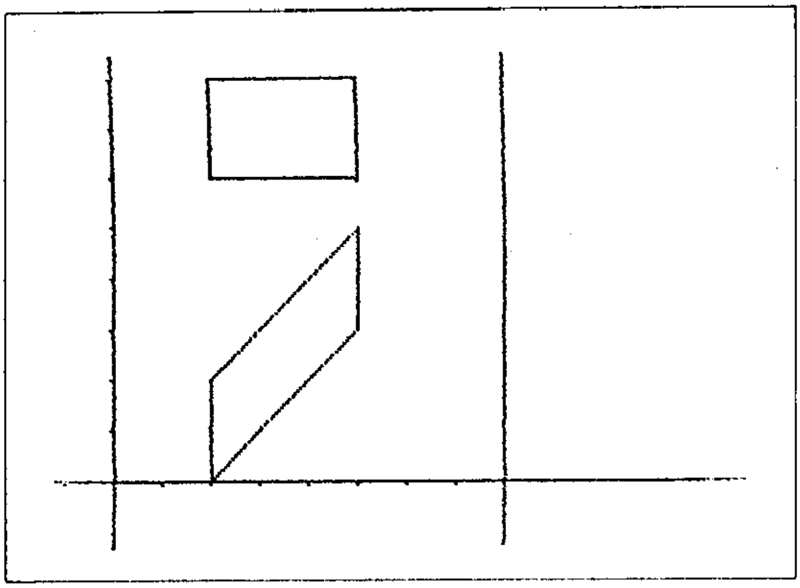


Fig. 1.94
A shear with invariant line $x = 8$, and $(5,6) \rightarrow (5,3)$.

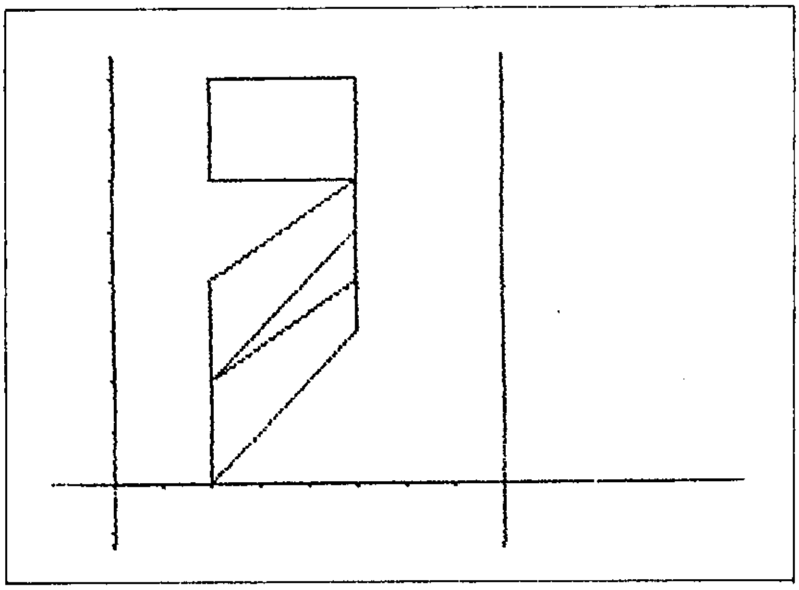


Fig. 1.95
A second shear with invariant line $x = 8$, and $(5,3) \rightarrow (5,8)$.

(b) Two shears with parallel invariant lines are equivalent to either a translation parallel to the invariant lines, or a single shear whose invariant line is parallel to the others.

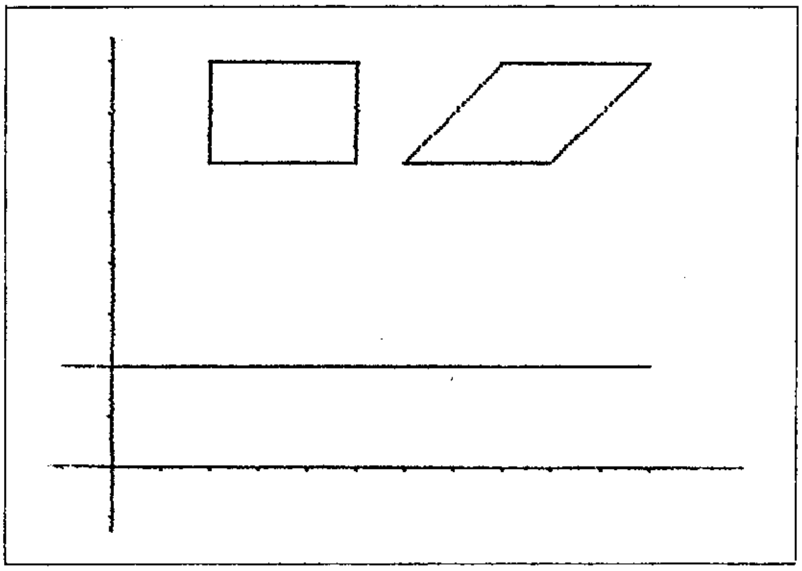


Fig. 1.96
A shear with invariant line $y = 2$ and $(2,3) \rightarrow (3,3)$.

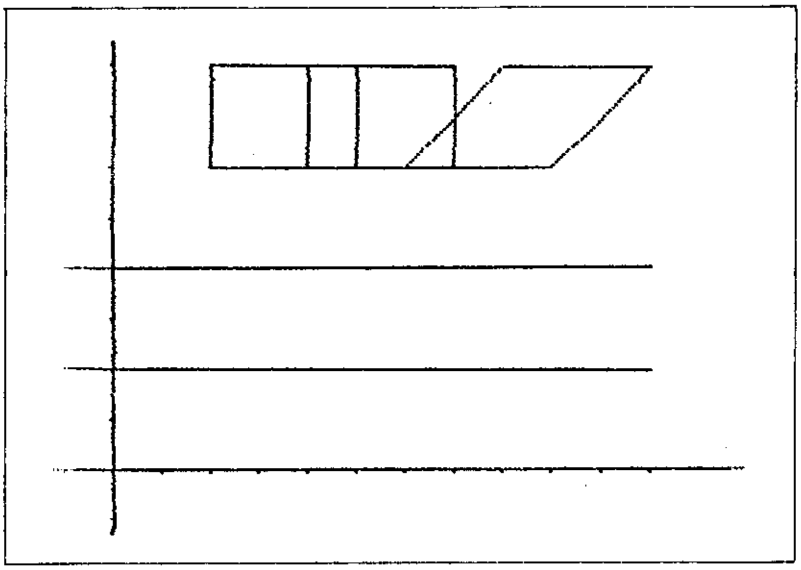


Fig. 1.97
A second shear with invariant line $y = 4$ and $(6,6) \rightarrow (4,6)$ combines with the first to give a translation $\binom{2}{0}$.

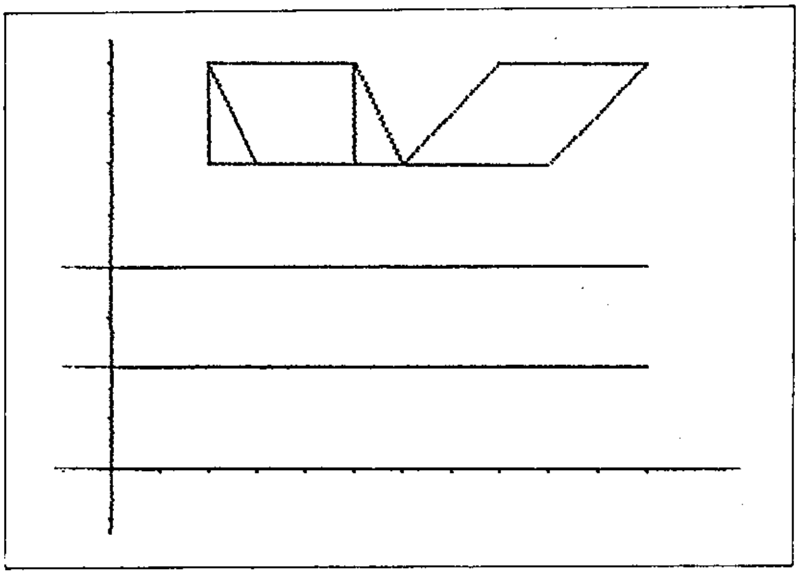


Fig. 1.98
A second shear with invariant line $y = 4$ and $(6,6) \rightarrow (3,6)$ combines with the first to give a shear with invariant line $y = 8$ and $(2,6) \rightarrow (3,6)$.

(c) Two shears with intersecting invariant lines are not equivalent to a single shear.
(The vertices have been labelled in figure 1.100 to show that the transformation ABCD onto A"B"C"D" is not a shear since AA", BB", CC" and DD" are not parallel.)

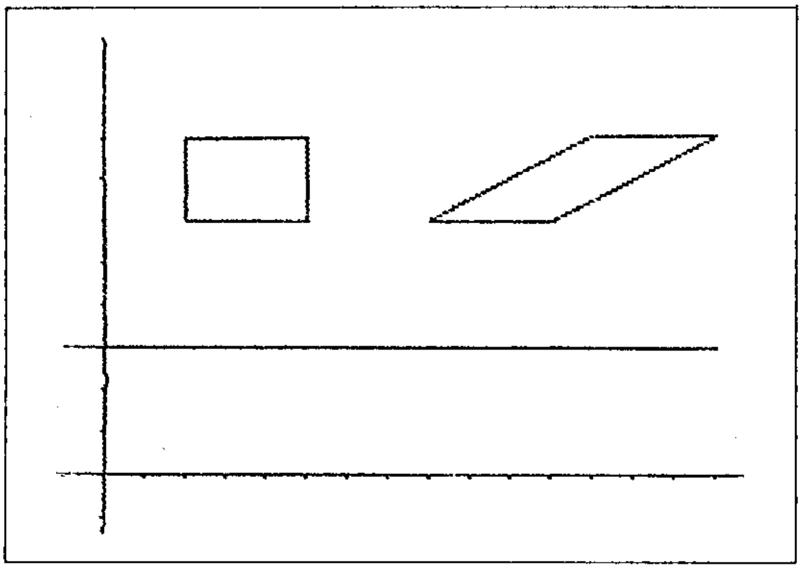


Fig. 1.99
A shear with invariant line $y = 3$ and $(1,4) \rightarrow (3,4)$.

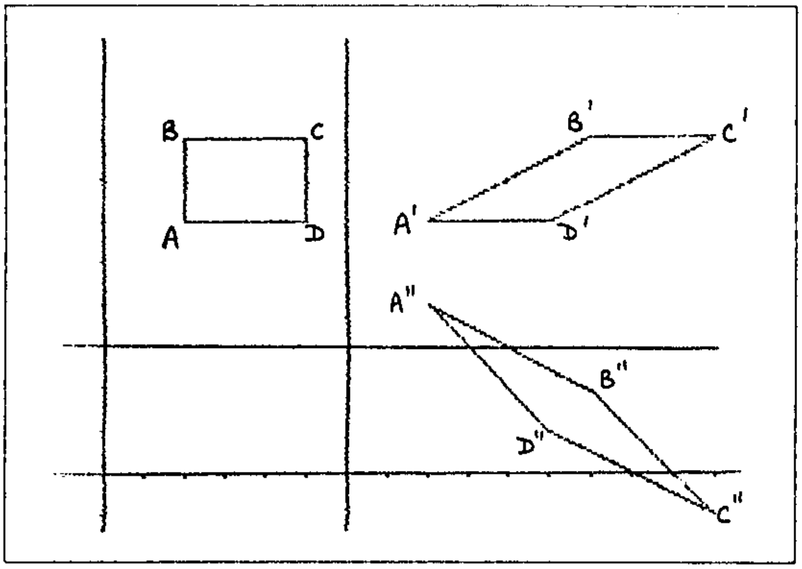


Fig. 1.100
A second shear with invariant line $x = 6$ and $(7,6) \rightarrow (7,5)$.

In teaching the shearing transformation, I have found that the pupils are often left with a feeling of incompleteness because the set of shears is not closed under combination, and also because they are unable to find another single transformation which is equivalent to one shear followed by another.

Chapter Two

In the previous chapter the dynamic capabilities of the computer were the main features. This chapter contains three programs, each concerned with an aspect of the study of number, which illustrate different reasons for using the computer in teaching the various topics. The program listings are given in appendices B, C and D.

I shall describe the programs as before, the execution and the use of the program in the classroom.

1. Number patterns. Program name 'num-pat'.

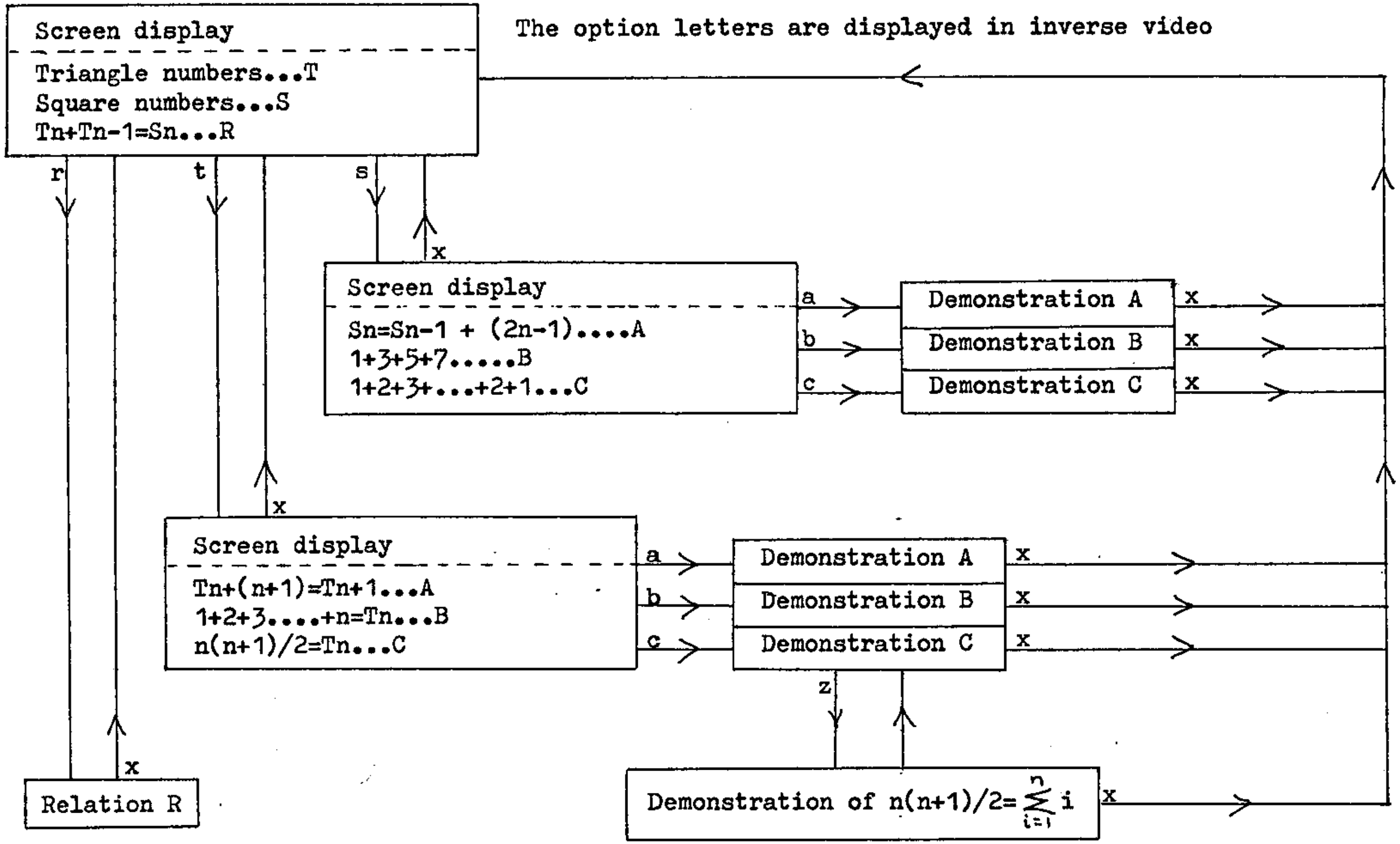
The program shows three basic patterns involving square numbers, three involving triangle numbers, and a relation between square numbers and triangle numbers. The flow diagram on the next page shows how the program is controlled using various keys. In general, during any particular demonstration, pressing and releasing a key will advance that demonstration a step.

The 'x' key is used to exit from the demonstration (usually at the end of a stage).

The 'z' key is used for a specific purpose during one of the demonstrations (see the flow diagram).

Following the RUN command, the screen is blank until a key is pressed, after which the initial options are displayed.

FLOW DIAGRAM TO SHOW THE EXECUTION OF 'num-pat' PROGRAM



The option letters are displayed in inverse video

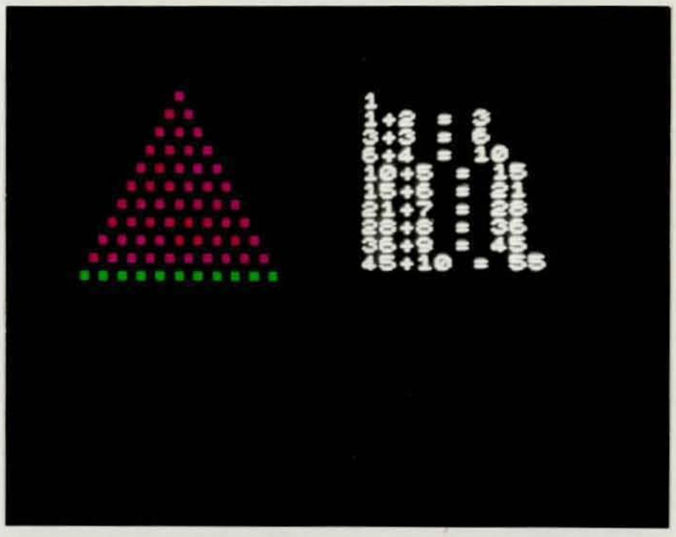
Triangle numbers

(a) $t_n + (n + 1) = t_{n+1}$. (t_n is the n^{th} triangle number.)

The demonstration begins with one dot to represent t_1.

Each stage of the demonstration shows that by adding one more dot than was added in the previous stage a triangular arrangement is generated, giving the sequence 1, 3, 6, 10, ...

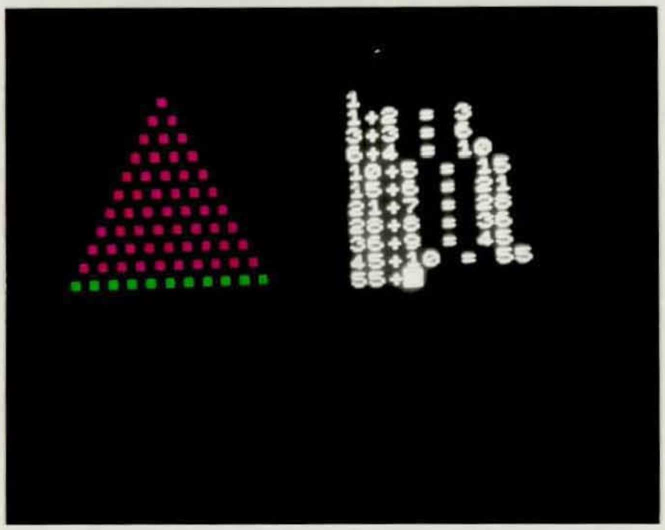
The following series of figures shows one stage. (Each step is advanced by pressing and releasing any key except 'x'.)



```
1
1+2 = 3
3+3 = 6
6+4 = 10
10+5 = 15
15+6 = 21
21+7 = 28
28+8 = 36
36+9 = 45
45+10 = 55
```

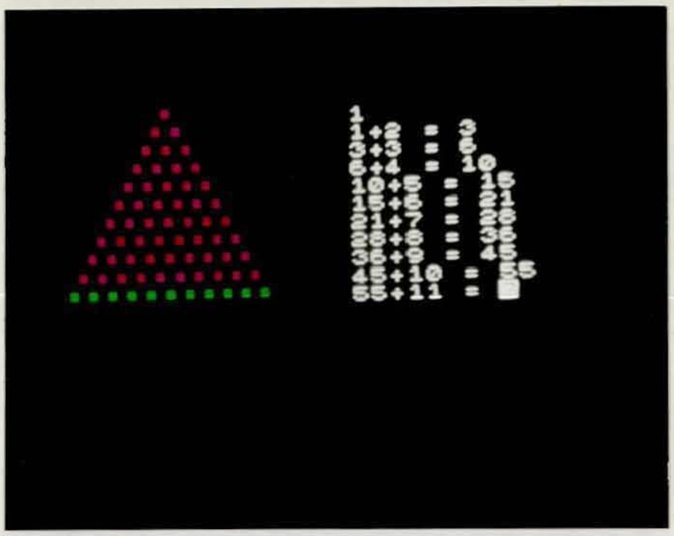
Fig. 2.1

Step 1. A row of green dots is added to the previous triangle. Releasing the key while the green dots are being printed results in figure 2.1, releasing after they are printed results in figure 2.2. The question mark flashes to allow time for the pupils to respond with the number of green dots.



```
1
1+2 = 3
3+3 = 6
6+4 = 10
10+5 = 15
15+6 = 21
21+7 = 28
28+8 = 36
36+9 = 45
45+10 = 55
55+
```

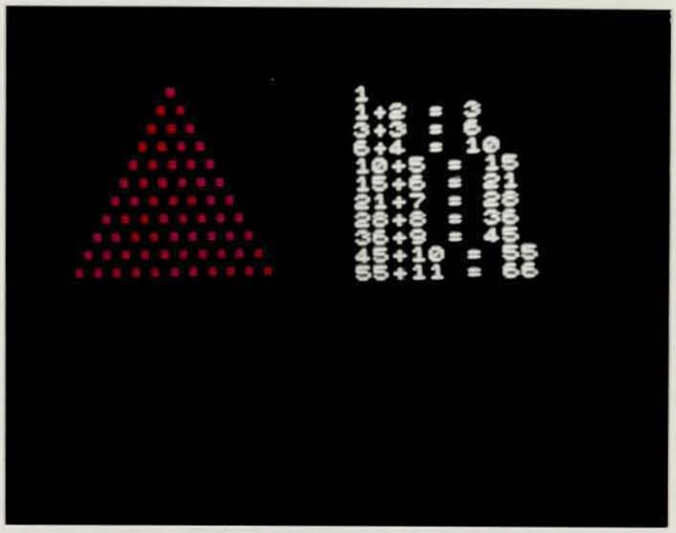
Fig. 2.2



```
1
1+2 = 3
3+3 = 6
6+4 = 10
10+5 = 15
15+6 = 21
21+7 = 28
28+8 = 36
36+9 = 45
45+10 = 55
55+11 =
```

Fig. 2.3

Step 2. The number of the green dots is printed. The question mark flashes to allow time for the pupils to respond with the total number of dots.

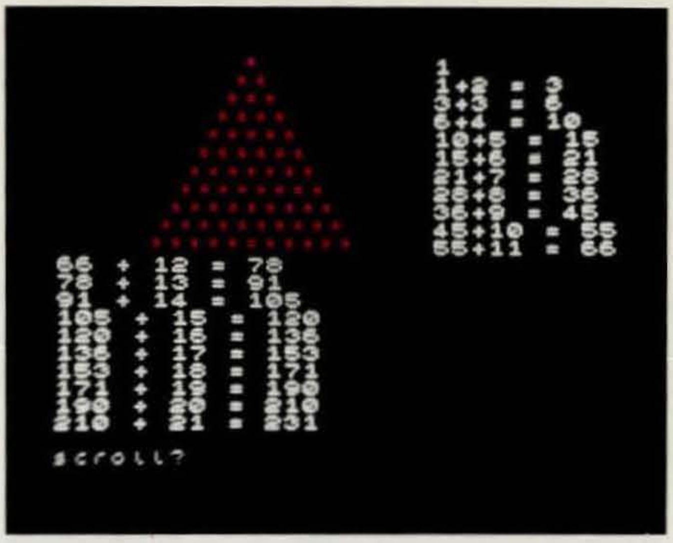


```
1
1+2 = 3
3+3 = 6
6+4 = 10
10+5 = 15
15+6 = 21
21+7 = 28
28+8 = 36
36+9 = 45
45+10 = 55
55+11 = 66
```

Fig. 2.4

Step 3. The green dots are changed to magenta and the total number of dots is printed.

Pressing the 'x' key at this stage results in the sequence being generated without the pattern of dots (see figure 2.5). This will also happen when any key is pressed after the stage which shows 190 + 20 = 210 (210 is the largest triangle of dots which can be accommodated on the screen).



```
1
1+2 = 3
3+3 = 6
6+4 = 10
10+5 = 15
15+6 = 21
21+7 = 28
28+8 = 36
36+9 = 45
45+10 = 55
55+11 = 66
```

```
66  + 12 = 78
78  + 13 = 91
91  + 14 = 105
105  + 15  = 120
120  + 16  = 136
136  + 17  = 153
153  + 18  = 171
171  + 19  = 190
190  + 20  = 210
210  + 21  = 231
scroll?
```

Fig. 2.5

The sequence continued without the pattern of dots.

The sequence is continued by pressing and releasing a key, the previous results being scrolled up. Holding down the key will halt the scrolling until the key is released. Pressing the 'x' key at this stage returns to the initial option display.

(b) $t_n = \sum_{k=1}^{n} k$. (The sum of the first n natural numbers is the n^{th} triangle number.)

This demonstration displays each triangle in turn. The following sequence of steps is then generated by pressing and releasing a key (to execute each step).

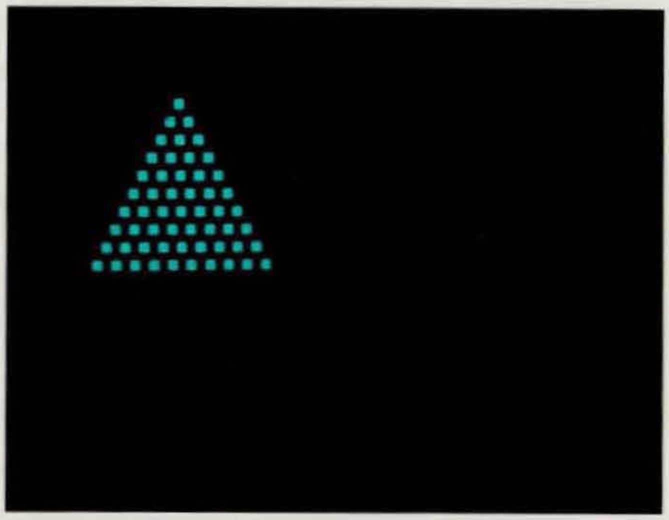


Fig. 2.6

Step 1 prints a triangular arrangement of dots.

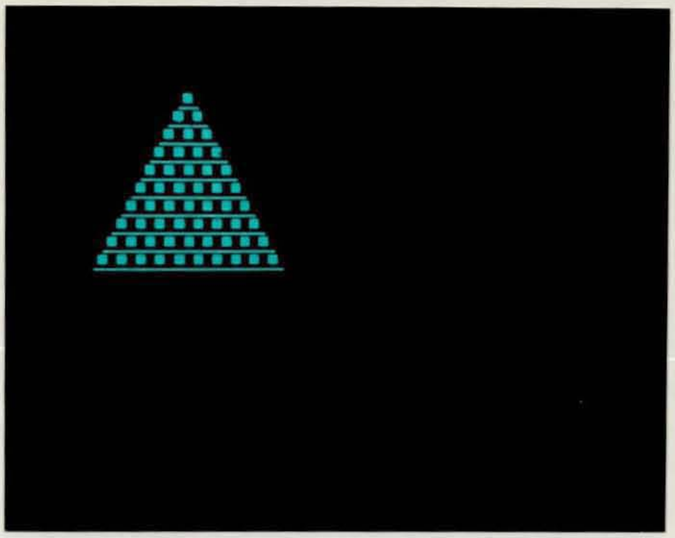


Fig. 2.7

Step 2 divides the triangle into rows of dots, by drawing horizontal lines.

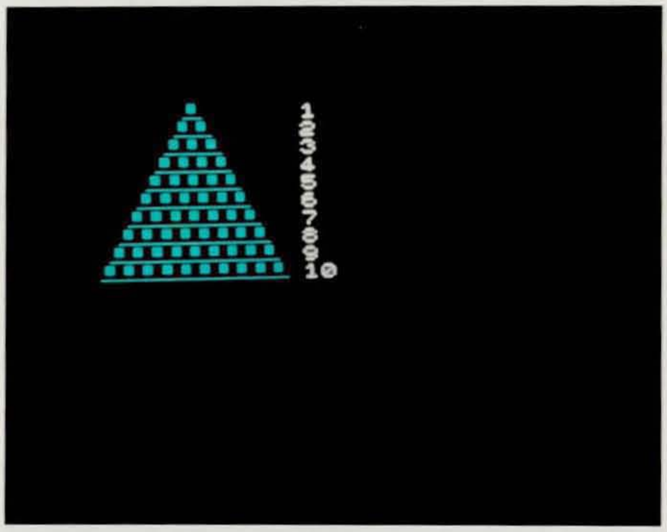


Fig. 2.8

Step 3 prints, at the side of the triangle, the number of dots in each row. (These are consecutive natural numbers.)

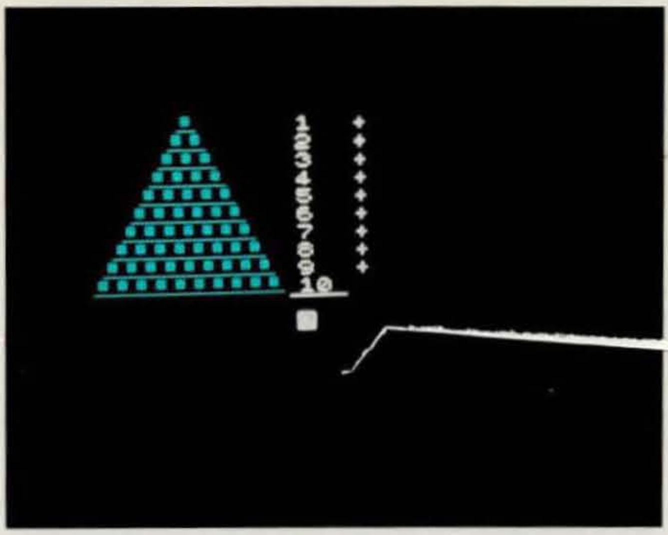


Fig. 2.9

Step 4 prints these numbers as an addition sum. The '?' flashes to allow time for the pupils to respond with the total.

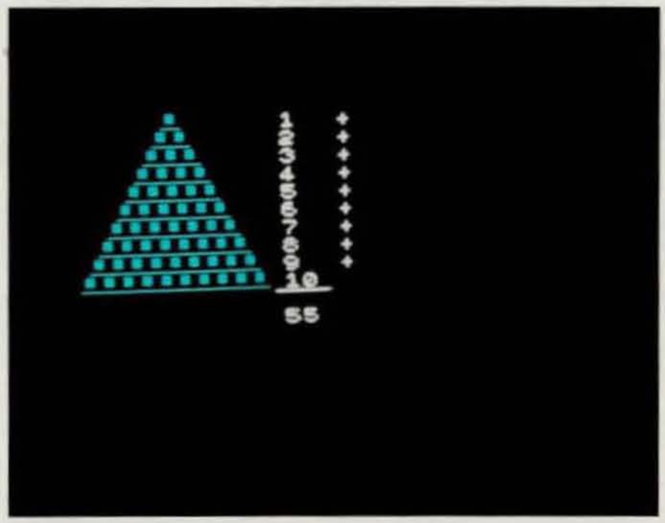


Fig. 2.10

Step 5 prints the total number in place of the question mark.

A return to the initial option display is achieved by pressing the 'x' key at this stage, or by pressing any key after the 19th triangle number (which is the largest that can be accommodated on the screen) has been shown.

(c) $n(n + 1)/2 = t_n$.

Using, in turn, a rectangular arrangement of dots 1x2, 2x3, 3x4, 4x5, etc. the following series of steps (shown for 8x9) is executed by pressing and releasing a key for each step.

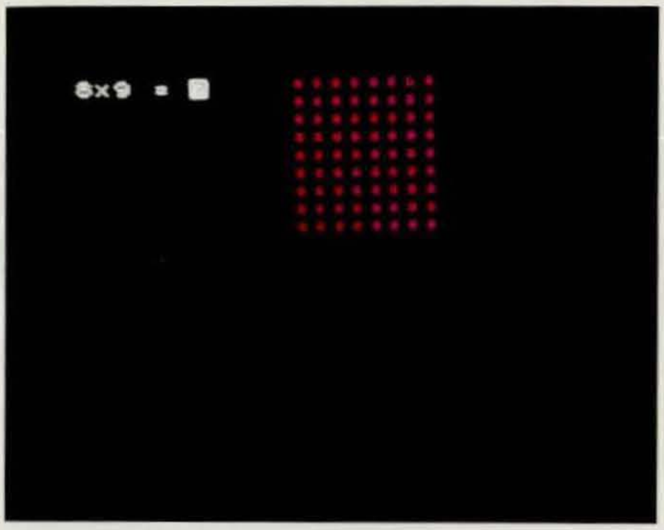


Fig. 2.11

Step 1 prints the rectangle of dots together with its dimensions. The '?' flashes to allow time for the pupils to respond with the total number of dots.

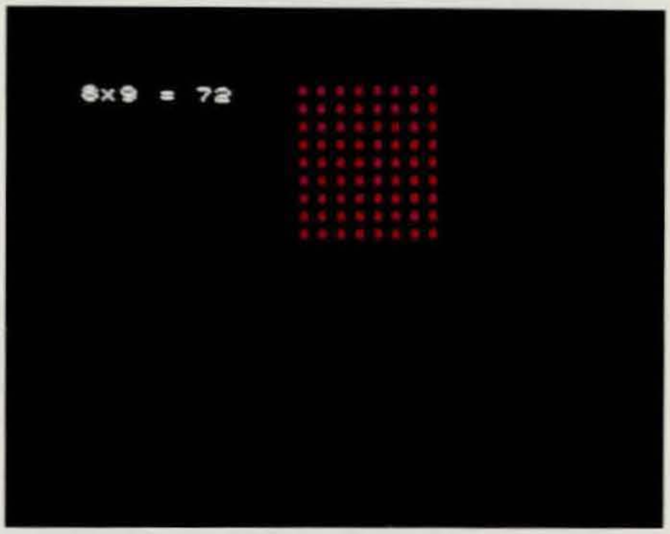
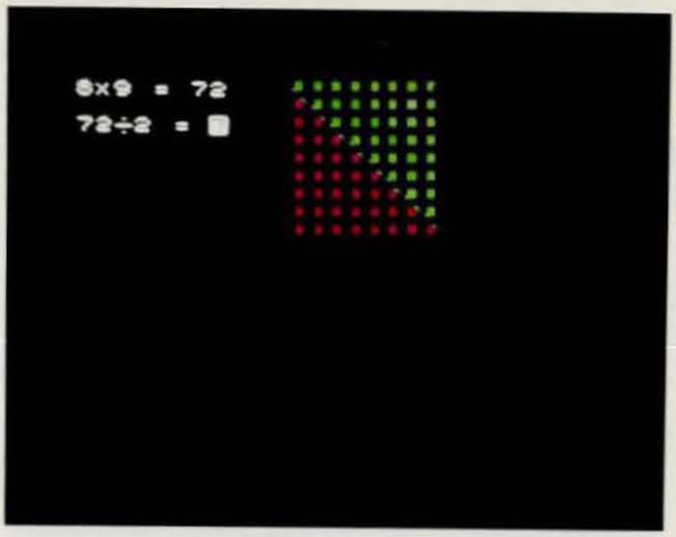


Fig. 2.12

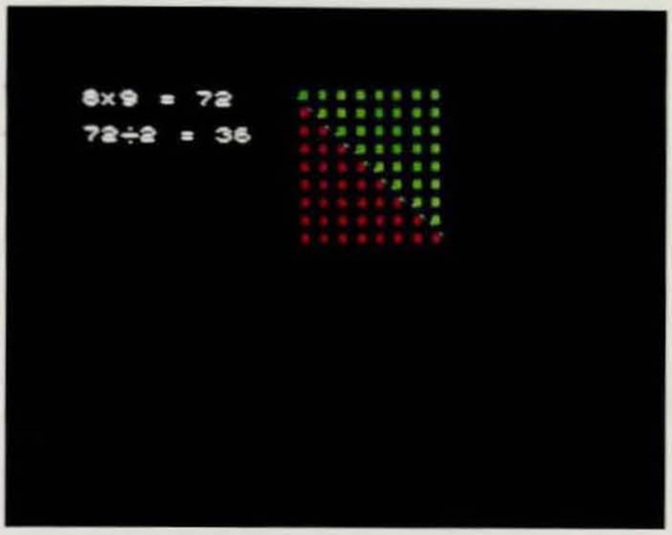
Step 2 prints the total number of dots, in place of the flashing question mark.



8x9 = 72
72÷2 = ■

Fig. 2.13

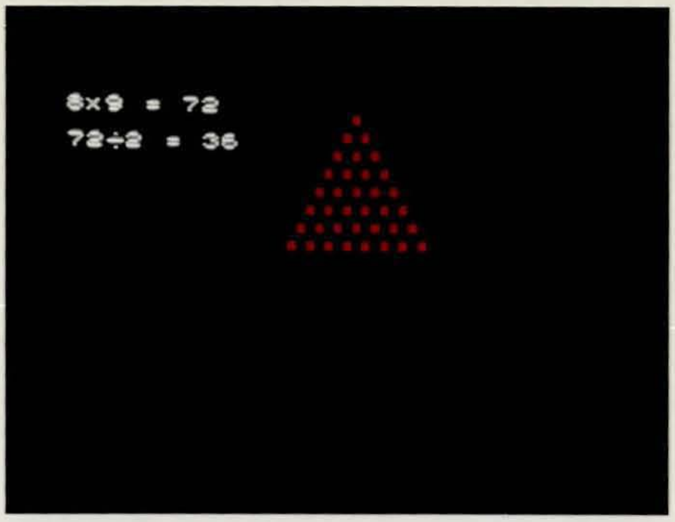
Step 3 divides the rectangular arrangement of dots into two equal numbers by drawing a diagonal line and changing half of the dots to green. The question mark flashes until the next step is initiated.



8x9 = 72
72÷2 = 36

Fig. 2.14

Step 4 prints the number of dots in each half, in place of the question mark.

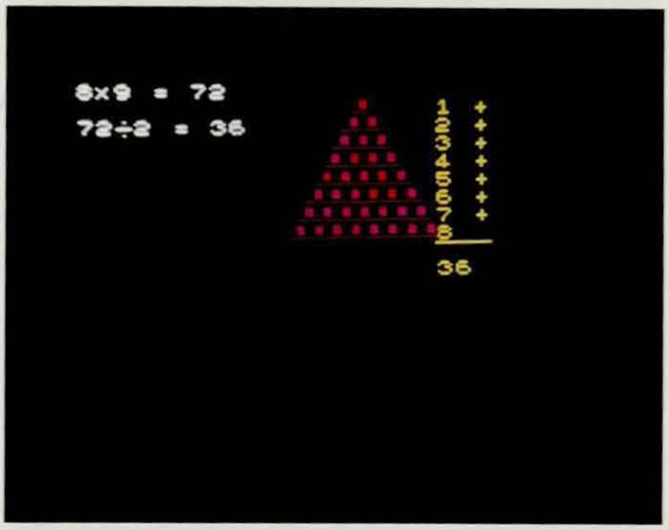


```
8x9 = 72
72÷2 = 36
```

Fig. 2.15

Step 5 erases the upper half of the rectangle and transforms the remaining right angled triangle into an isosceles triangle, row by row.

Repeatedly pressing the 'z' key at this stage generates the stages of demonstration (b) above, for the triangle which is on the screen. Figure 2.16 shows the completed demonstration.

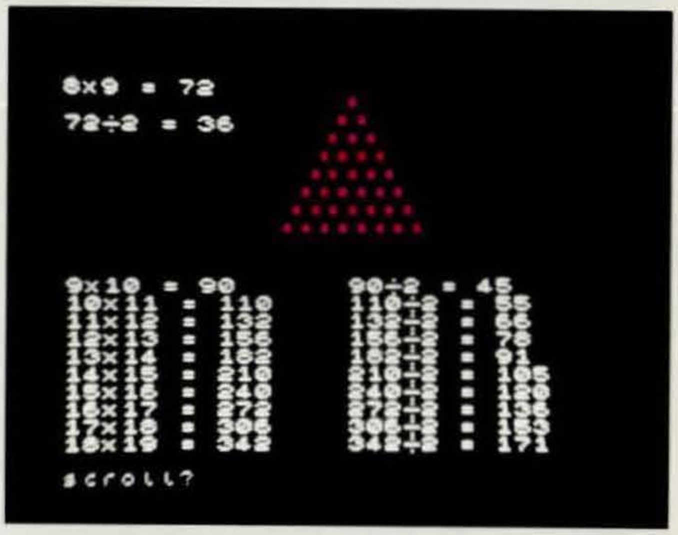


```
8x9 = 72
72÷2 = 36
        1 +
        2 +
        3 +
        4 +
        5 +
        6 +
        7 +
        8
        ____
        36
```

Fig. 2.16

An optional step to demonstrate that $\sum_{i=1}^{n} i = \frac{n(n + 1)}{2}$, shown for n = 8.

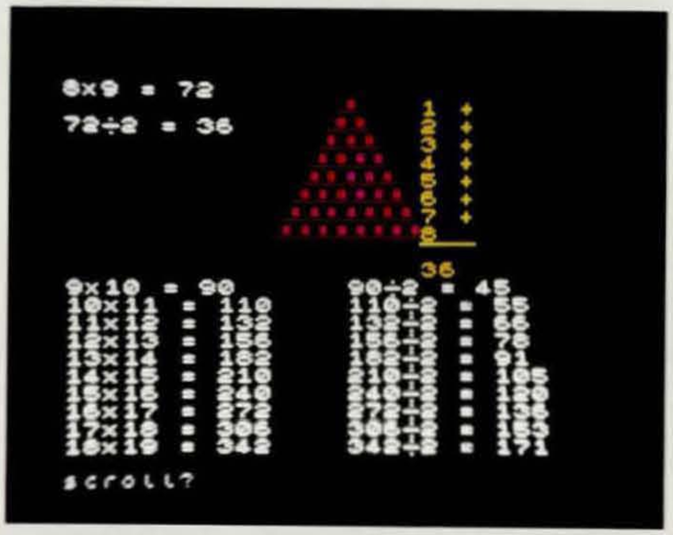
The sequence is printed without the demonstration of dots when 17 x 18 ÷ 2 = 153 has been demonstrated, or by pressing the 'x' key at one of the stages shown in figures 2.15 and 2.16. This is shown in figures 2.17 and 2.18.



```
8x9 = 72
72÷2 = 36

9x10 = 90       90÷2 = 45
10x11 = 110     110÷2 = 55
11x12 = 132     132÷2 = 66
12x13 = 156     156÷2 = 78
13x14 = 182     182÷2 = 91
14x15 = 210     210÷2 = 105
15x16 = 240     240÷2 = 120
16x17 = 272     272÷2 = 136
17x18 = 306     306÷2 = 153
18x19 = 342     342÷2 = 171
scroll?
```

Fig. 2.17

The result of pressing the 'x' key after the stage shown in figure 2.15.



```
8x9 = 72
72÷2 = 36
                  1 +
                  2 +
                  3 +
                  4 +
                  5 +
                  6 +
                  7 +
                  8
                  36
9x10 = 90       90÷2 = 45
10x11 = 110     110÷2 = 55
11x12 = 132     132÷2 = 66
12x13 = 156     156÷2 = 78
13x14 = 182     182÷2 = 91
14x15 = 210     210÷2 = 105
15x16 = 240     240÷2 = 120
16x17 = 272     272÷2 = 136
17x18 = 306     306÷2 = 153
18x19 = 342     342÷2 = 171
scroll?
```

Fig. 2.18

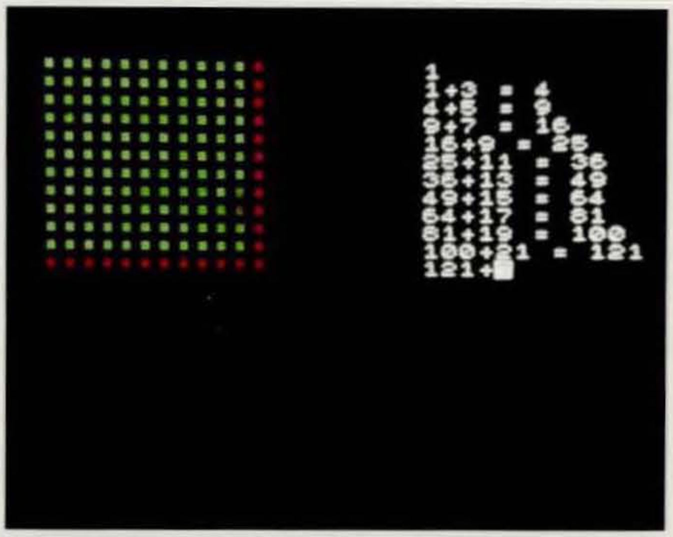
The result of pressing the 'x' key after the stage shown in figure 2.16.

Pressing the 'x' key at this stage returns to the initial option display. Pressing any other key continues the sequence.

Square numbers

(a) $S_n = S_{n-1} + (2n-1)$.

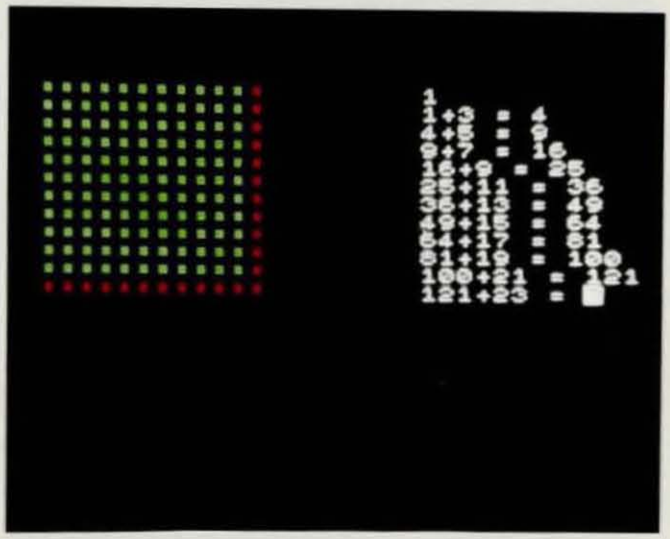
The demonstration begins with one dot to represent S_1, the first square number. Each stage demonstrates that by adding the n^{th} odd number to the $(n-1)^{th}$ square number gives the n^{th} square number. The sequence of figures 2.19 - 2.21 shows one stage, each step being advanced by pressing and releasing a key.



```
1
1+3 = 4
4+5 = 9
9+7 = 16
16+9 = 25
25+11 = 36
36+13 = 49
49+15 = 64
64+17 = 81
81+19 = 100
100+21 = 121
121+█
```

Fig. 2.19

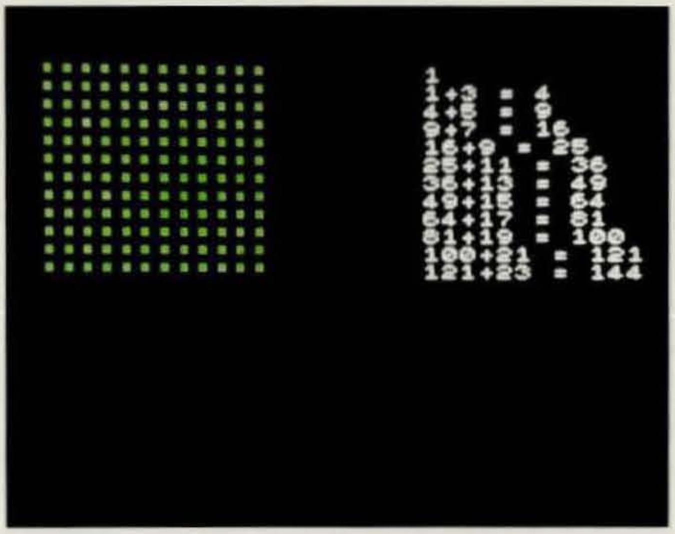
Step 1 adds the red dots to the previous square, prints the number of green dots in the previous square and the flashing '?' to allow time for the pupils to respond with the number of additional dots.



```
1
1+3 = 4
4+5 = 9
9+7 = 16
16+9 = 25
25+11 = 36
36+13 = 49
49+15 = 64
64+17 = 81
81+19 = 100
100+21 = 121
121+23 = █
```

Fig. 2.20

Step 2 overprints the '?' with the additional number of dots and prints a second '?' which also flashes until the next step is initiated.

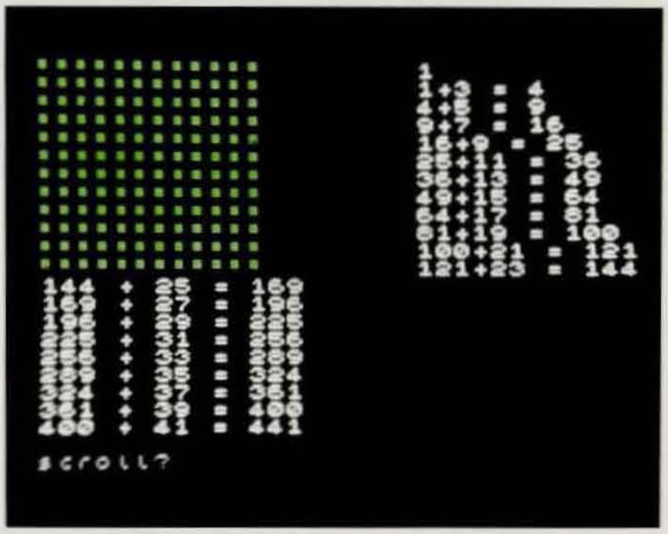


```
1
1+3  = 4
4+5  = 9
9+7  = 16
16+9  = 25
25+11  = 36
36+13  = 49
49+15  = 64
64+17  = 81
81+19  = 100
100+21  = 121
121+23  = 144
```

Fig. 2.21

Step 3 changes the red dots to green and overprints '?' with the total number of dots (the next square number).

Pressing the 'x' key at this stage, or any key when 361 + 39 = 400 has been demonstrated, results in the sequence being printed as in figure 2.22.



```
                                1
                                1+3  = 4
                                4+5  = 9
                                9+7  = 16
                                16+9  = 25
                                25+11  = 36
                                36+13  = 49
                                49+15  = 64
                                64+17  = 81
                                81+19  = 100
                                100+21  = 121
                                121+23  = 144
144 + 25 = 169
169 + 27 = 196
196 + 29 = 225
225 + 31 = 256
256 + 33 = 289
289 + 35 = 324
324 + 37 = 361
361 + 39 = 400
400 + 41 = 441

scroll?
```

Fig. 2.22

The sequence continues without the dot demonstration.

Pressing the 'x' key returns to the initial option display. Pressing any other key will halt the printing and scrolling until it is released, when the process continues.

(b) $\sum_{k=1}^{n}(2k-1) = S_n = n^2$.

Each stage of this demonstration consists of displaying a square arrangement of dots (starting with 1) and then dividing the dots to show that the square is equivalent to the sum of the first n odd numbers. One stage (for n = 9) is shown in figures 2.23 - 2.27.

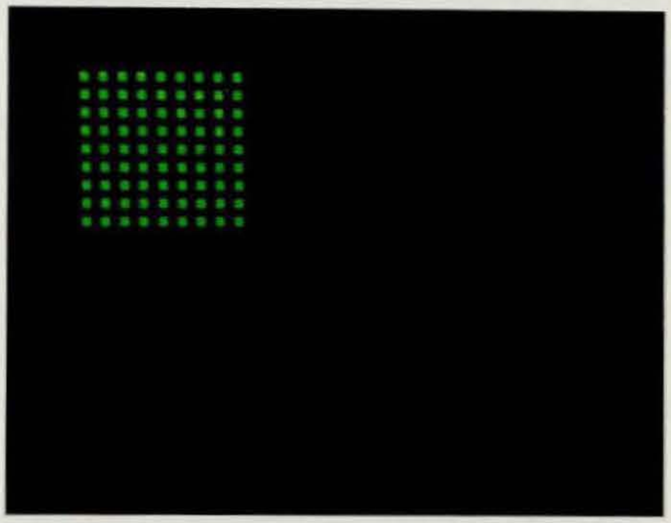


Fig. 2.23

Step 1 prints a square arrangement of dots.

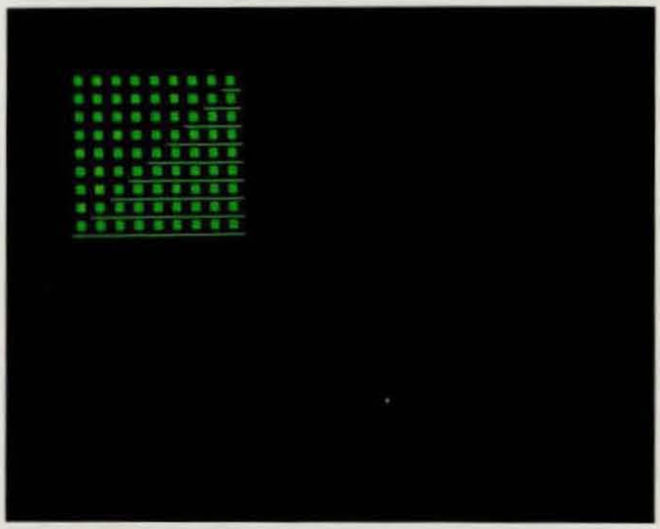


Fig. 2.24

Step 2 divides the dots into gnomons by drawing L-shaped lines.

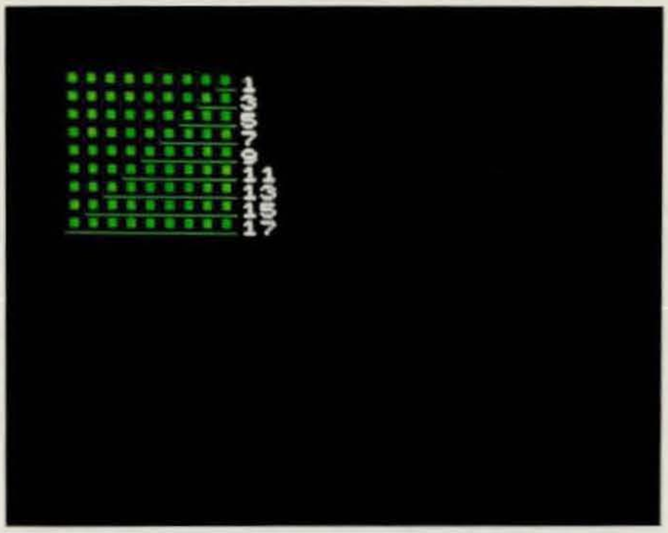


Fig. 2.25

Step 3 prints the number of dots in each gnomon (consecutive odd numbers).

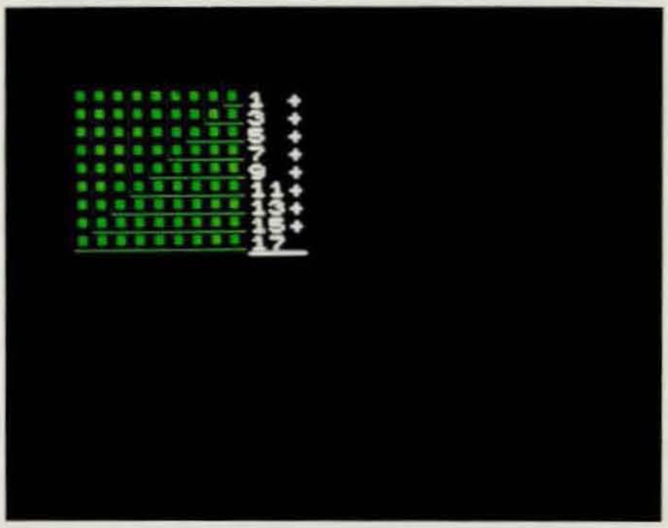


Fig. 2.26

Step 4 prints these numbers as an addition sum.

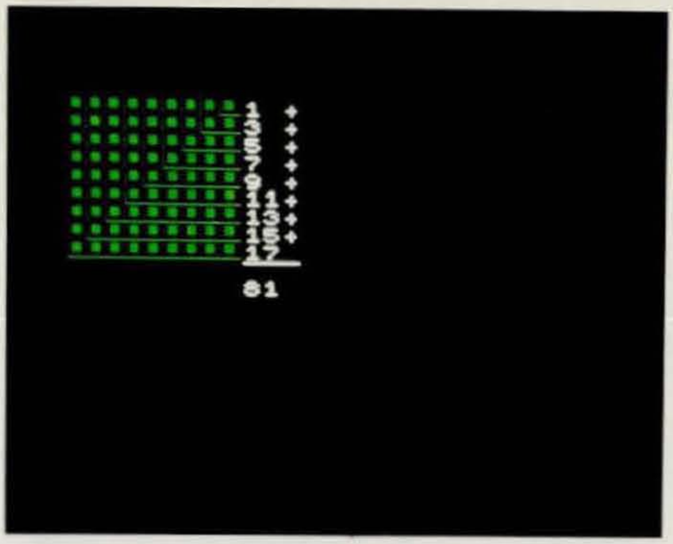


Fig. 2.27

Step 5 prints the total number of dots.

The initial option display returns if the 'x' key is pressed at the end of the stage, or if any key is pressed after $1 + 3 + 5 + \ldots\ldots\ldots + 37 = 361$ has been demonstrated.

(c) $1 + 2 + 3 + \ldots. + n + \ldots. + 3 + 2 + 1 = S_n = n^2$.

This section is similar to (b) above except that the square arrangement of dots is divided as in the following sequence of diagrams.

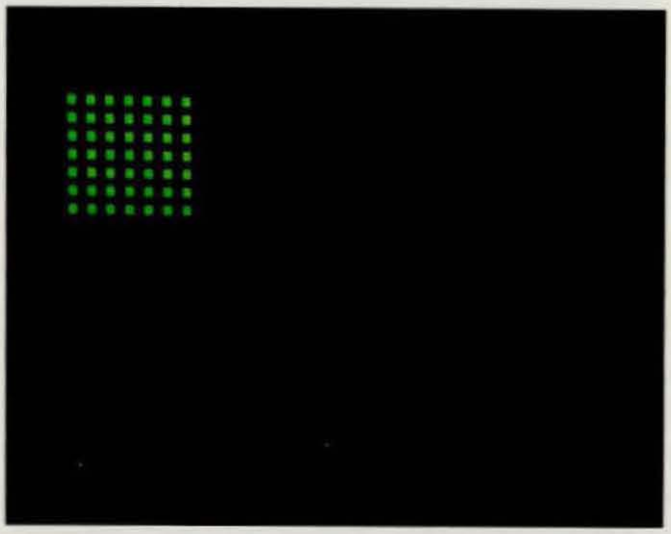


Fig. 2.28

Step 1 prints a square arrangement of dots.

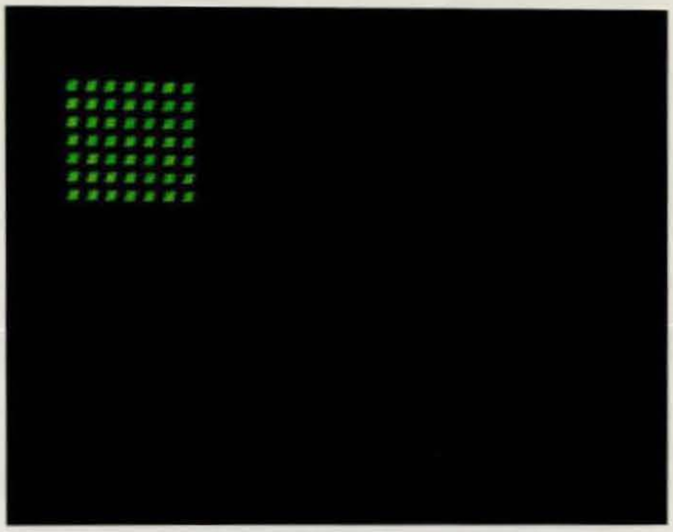


Fig. 2.29

Step 2 divides the square into diagonal rows of dots.

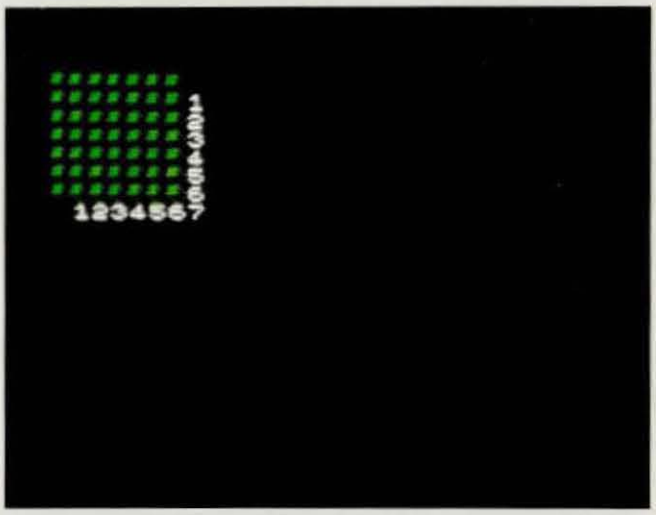
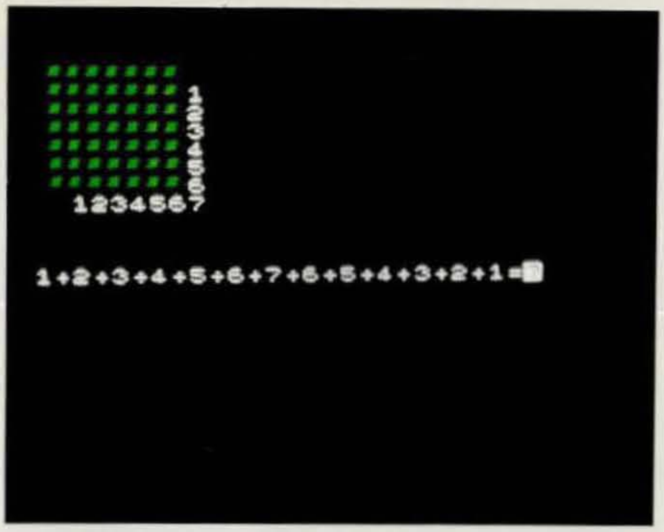


Fig. 2.30

Step 3 prints the number of dots in each diagonal row.

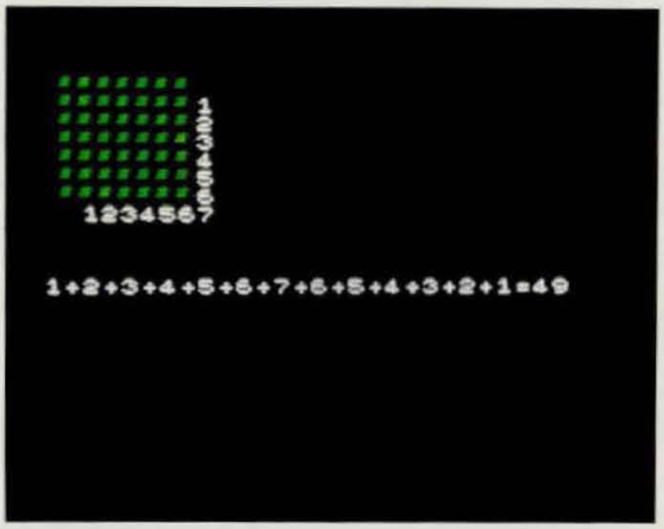


1234567

1+2+3+4+5+6+7+6+5+4+3+2+1=

Fig. 2.31

Step 4 prints the sum of these numbers below the square, as shown. The question mark flashes to allow time for the pupils to respond with the total number of dots.



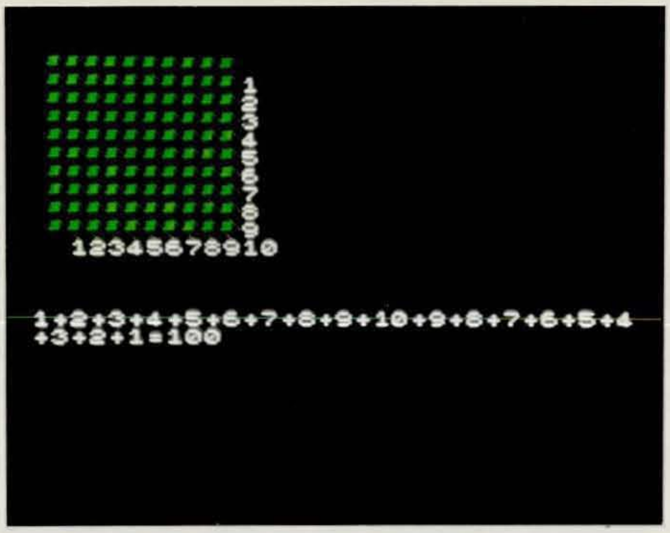
1234567

1+2+3+4+5+6+7+6+5+4+3+2+1=49

Fig. 2.32

Step 5 prints the total number of dots in the square.

Pressing the 'x' key at this stage results in the initial option display. Pressing any key at the stage shown in figure 2.33 will also return to the initial option display.



```
1+2+3+4+5+6+7+8+9+10+9+8+7+6+5+4
+3+2+1=100
```

Fig. 2.33

The largest square demonstrated for option c.

The relation between triangle and square numbers - $t_n + t_{n-1} = S_n$.

Taking successive pairs of consecutive triangle numbers, the computer demonstrates that the sum of two consecutive triangle numbers is a square number. The following sequence of diagrams shows the steps of one stage, for the numbers 66 and 78.

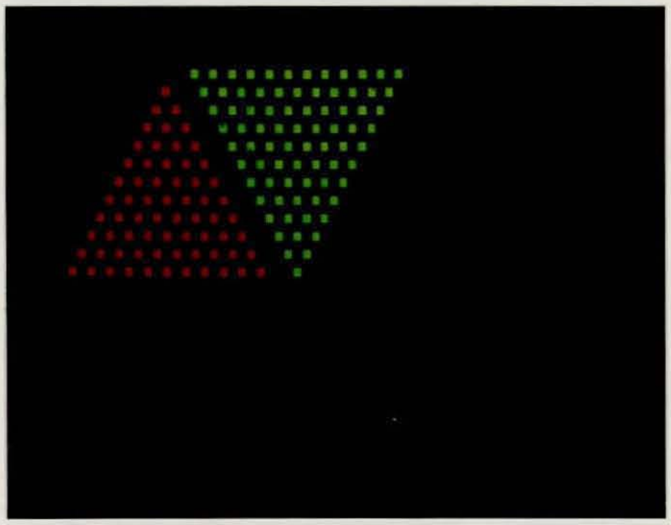


Fig. 2.34

Step 1 prints two triangular arrangements of dots.

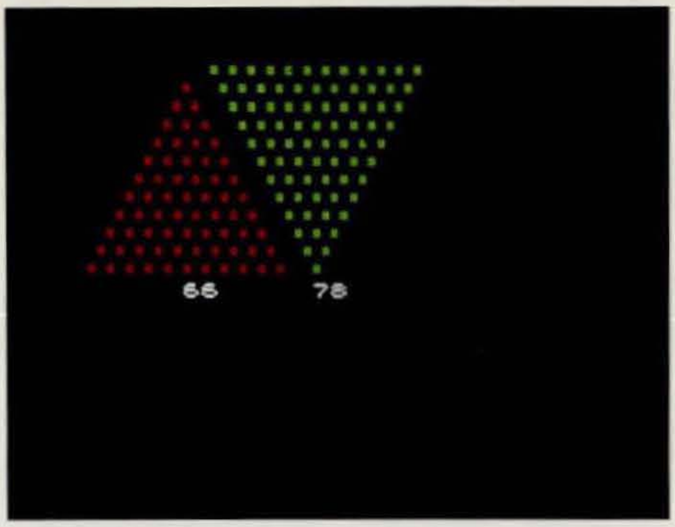


Fig. 2.35

Step 2 prints the number of dots in each of the triangles (two consecutive triangle numbers).

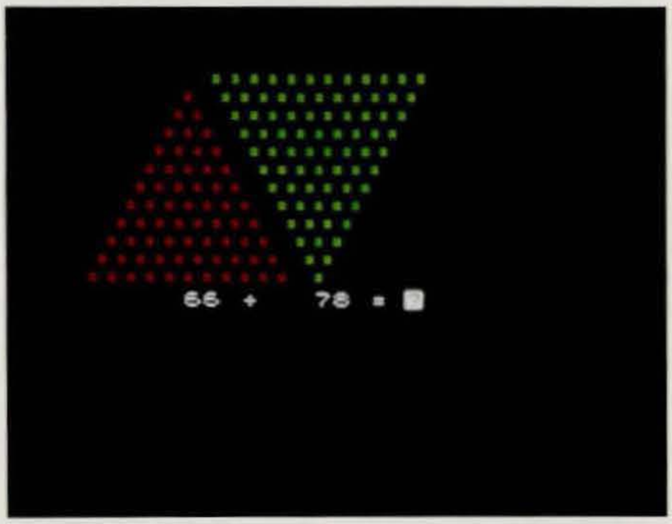
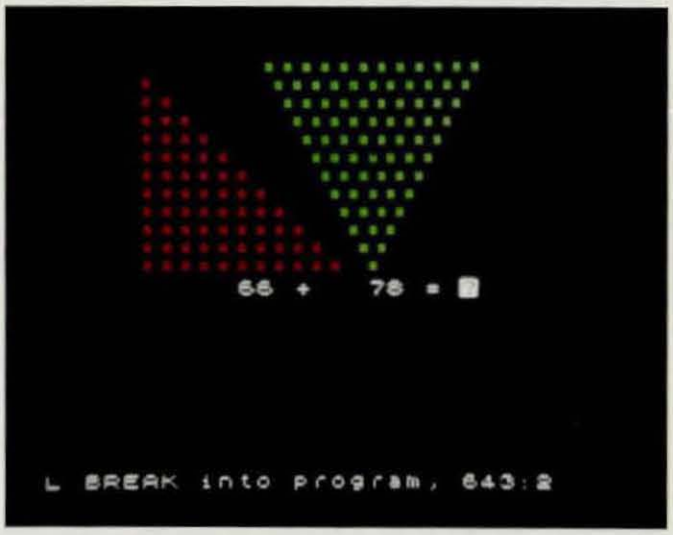


Fig. 2.36

Step 3 prints '+' between the two numbers and a flashing question mark to allow time for the pupils to respond with the sum of the two numbers.

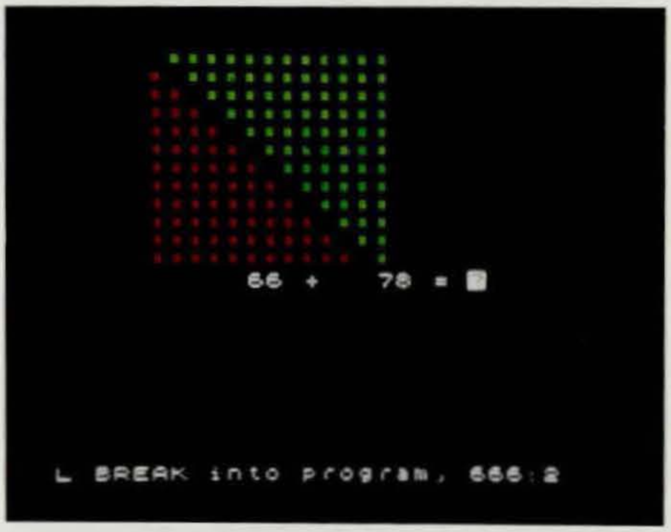
The next step is executed in three parts, with about two seconds between each part, after a single press of a key. (So that each part could be photographed, it was necessary to press the BREAK key - the CONT key continues the demonstration.)



```
66 +   78 = ▮

L BREAK into program, 643:2
```

Fig. 2.37

Step 4a. The left triangle is transformed row by row into a right angled triangle.



```
66 +   78 = ▮

L BREAK into program, 666:2
```

Fig. 2.38

Step 4b. The right triangle is transformed row by row into a right angled triangle.

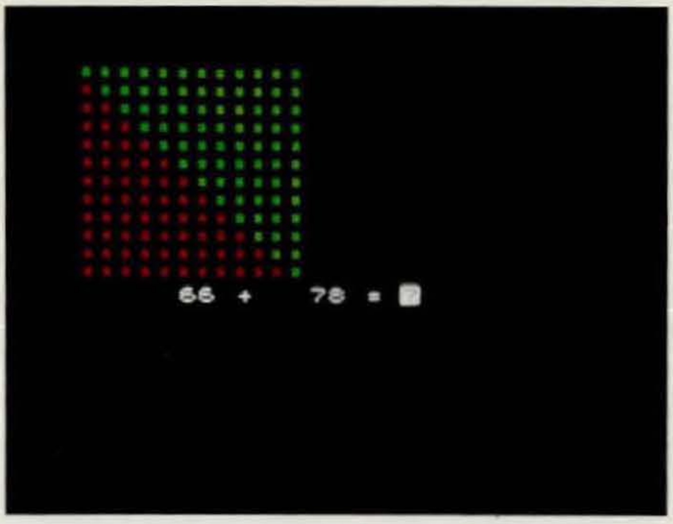

66 + 78 = ?

Fig. 2.39

Step 4c. The right triangle is moved to the left so that the two triangles together form a square.

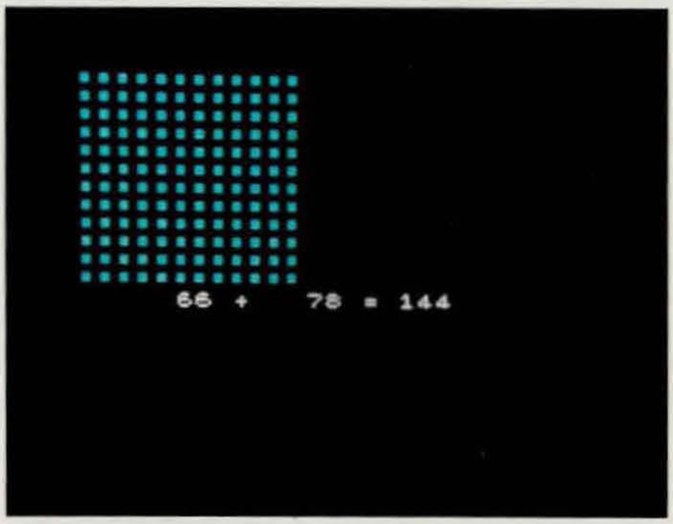
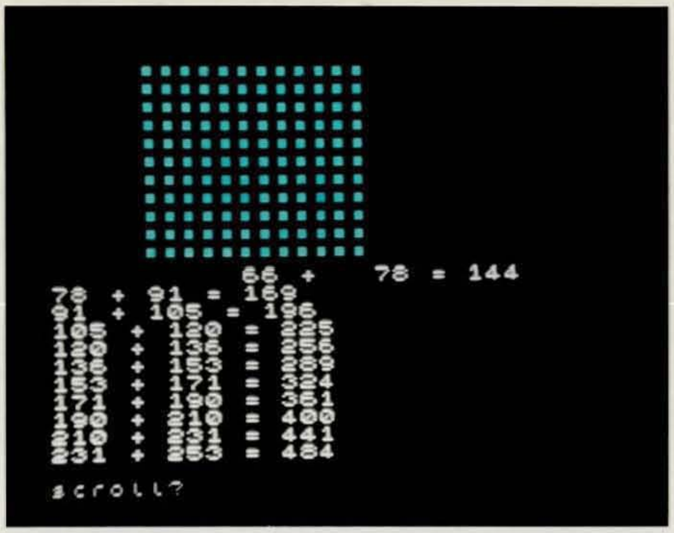

66 + 78 = 144

Fig. 2.40

Step 5 changes the colour of all the dots to cyan, and prints the total in place of '?'.

Pressing the 'x' key at this stage results in figure 2.41. A similar list is printed after the stage which shows 190 + 210 = 400 (the largest square which can be accommodated on the screen).



```
              66 +    78 = 144
78 + 91 = 169
91 + 105 = 196
105 + 120 = 225
120 + 136 = 256
136 + 153 = 289
153 + 171 = 324
171 + 190 = 361
190 + 210 = 400
210 + 231 = 441
231 + 253 = 484

scroll?
```

Fig. 2.41

The sequence has been generated without the demonstration of dots. (One line of dots of the square has scrolled off the top of the screen.)

Pressing the 'x' key at this stage returns to the initial option display. Pressing any other key continues the sequence.

Using the program

The program was originally written for the Sinclair ZX 81 computer and was later transferred to the Sinclair Spectrum. The initial purpose of using the computer to teach this topic was to stimulate interest from low ability second year pupils, last lesson on Fridays. I felt that, in addition to providing some practice in arithmetic, the study of number patterns could be enjoyable for these pupils, but most text books which include the topic involve too much reading for the least able pupils. Using the computer, however, produced an enthusiasm which was not always present with this group of pupils.

Each section of the program constituted a lesson's work (thirty minutes). The order in which I introduced each demonstration is evident from the order in which they appear in the program, each being added as it was written. When I first used the program, one relation was introduced each week, but to sustain the interest more effectively, I will, in future, spread the lessons throughout the year.

Each lesson took the form of demonstrating a few stages of the relation or pattern, with the pupils recording the results in their exercise books. After the relation had been 'discovered', the pupils were encouraged to complete the next stage before the computer demonstration. Allowing some of the children to advance the demonstration contributed to their enthusiasm.

Since the program has been completed, I have used it with a more able 4th year group to introduce sequences. The pupils were already familiar with the patterns of square and triangle numbers and so the computer program was being used as a reminder and also to promote discussion of how the sequences of triangle and square numbers may be defined. I introduced the notation t_n for the n^{th} term in the sequence of triangle numbers and s_n for the square numbers. With this notation, the relations $t_n + (n+1) = t_{n+1}$, $t_n = \frac{n(n+1)}{2}$, $s_n + (2n+1) = s_{n+1}$ and $s_n = n^2$ were established.

Previous work with this group had included flow diagrams, and so flow diagrams were produced to generate the sequences of triangle and square numbers. Using these flow diagrams, computer programs to list the sequences were written (as a class) and subsequently run on the computer.

The third group of pupils with whom I have used the number patterns program was in the first year of the Advanced level mathematics course, for introducing basic ideas on induction, and iteration. The pupils were shown each demonstration in turn and asked to generalise the results. It was agreed that the relations, although being demonstrated for specific values of n, had not been proved, and so the class set about this task. Taking $s_n = n^2$ as a definition, $s_{n+1} = s_n + (2n+1)$ was proved using elementary algebra.

Considering $t_n = 1 + 2 + 3 + \ldots + n$ gave the opportunity to introduce the $\sum$ notation. With this definition of t_n, the relation $t_{n+1} = t_n + (n+1)$ was proved, since $\sum_{i=1}^{n} i + (n+1) = \sum_{i=1}^{n+1} i$.

Proof by induction was developed more formally when the identity $\sum_{i=1}^{n} i = n(n+1)/2$ was considered (see figure 2.16 for the demonstration). Demonstrations b and c for the square numbers gave

$\sum_{i=1}^{n}(2i-1) = n^2$ and $\sum_{i=1}^{n-1} i + \sum_{i=1}^{n} i = n^2$, which were shown to be identical, and so the first was chosen for proof by induction.

Returning to the sequences for t_n and s_n, ways of defining these and sequences in general were considered, firstly by giving the general term, $t_n = \frac{n(n+1)}{2}$, $s_n = n^2$, secondly using an iterative definition. For the triangle and square numbers this involved eliminating n from $t_{n+1} = t_n + (n+1)$ and $t_n = \frac{n(n+1)}{2}$ to give $t_{n+1} = \frac{1}{2}(2t_n + 1 + \sqrt{8t_n + 1})$, and eliminating n from $s_n = n^2$ and $s_{n+1} = s_n + (2n+1)$ to give $s_{n+1} = s_n + 2\sqrt{s_n} + 1$.

As all the pupils in the group were familiar with BASIC, a short program was produced which generated the sequence iteratively:

```
10 LET u = 1
20 PRINT u
30 LET u = (2*u + 1 + SQR(8*u + 1))/2
40 GOTO 20
```

This generates the triangle numbers. Replacing line 30 with `LET u = u + 2*SQRu + 1` generates the square numbers.

To enforce the idea of iteration, a standard exercise was taken from an 'A' level textbook which involved generating a sequence from an iterative definition and then proving the given expression for the n^{th} term. Two programs were used, one for generating the sequence from the iterative definition, the other from the n^{th} term. An example is given in figure 2.42. Subsequent questions needed only to change lines 10 and 30.

Sequence $u_1 = 1,\ u_{k+1} = 2u_k + 1.$	General term $u_n = 2^n - 1.$
Program (a) 10 LET u = 1 20 PRINT u 30 LET u = 2*u + 1 40 GOTO 20	Sequence generated 1 3 7 15 31 63 127 255 511 1023 2047 4095 etc.
Program (b) 10 LET n = 0 20 LET n = n + 1 30 PRINT 2↑n - 1 40 GOTO 20	

Fig. 2.42

In discussion with the pupils, most considered that writing these programs helped them to understand better the idea of a sequence, especially those defined iteratively.

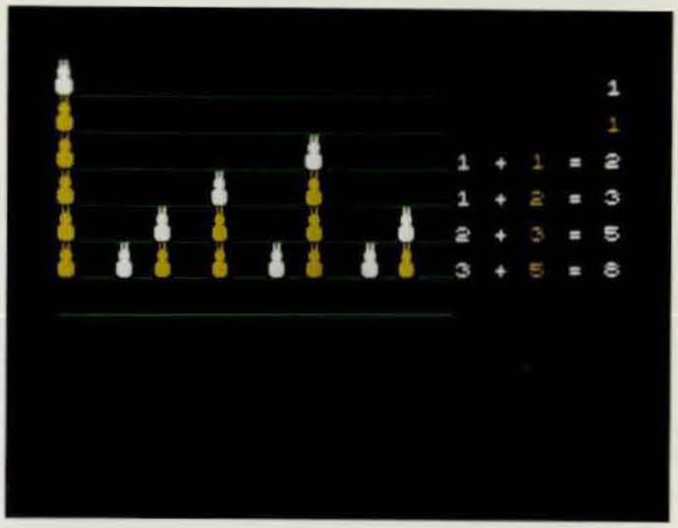
Conclusion

Although other means could have been used to teach the topics which I have outlined, using the computer in this way provides a welcome change from the more traditional methods. The children were motivated, and some valuable ideas were learned. Producing simple computer programs as described helped to reinforce the understanding of these ideas.

The next program was written initially for similar reasons to that on number patterns, namely to promote an interest in mathematics with the least able pupils who are so often denied the chance to investigate mathematics because of the language used in textbooks. It resulted, however, in being of use to other groups of pupils.

2. The Fibonacci Sequence. Program name 'fibonacci'.

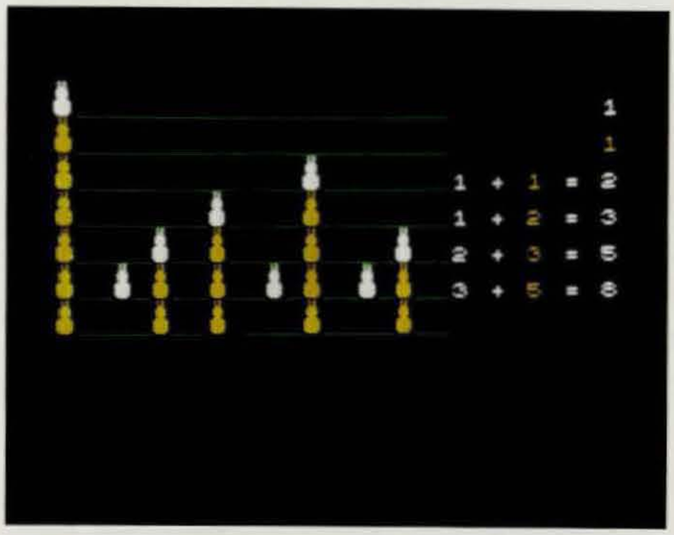
The program was saved using SAVE LINE 1, which results in the program running immediately following the LOAD command. Initially the screen is blank until a key is pressed, which begins the demonstration of the Fibonacci Sequence by illustrating rabbits. Fibonacci is said to have generated the sequence by considering a pair of rabbits (represented by a single rabbit on the screen). Rabbits can only breed when one month old. (Breeding rabbits are represented in yellow, young rabbits in white.) Each stage (one month) of the demonstration is advanced by pressing and releasing a key, and consists of the following steps (shown here for the 7th month).



1
1
1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
3 + 5 = 8

Fig. 2.43

Step 1 draws a green horizontal line (to represent grass).



1
1
1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
3 + 5 = 8

Fig. 2.44

Step 2 draws the rabbits which were old enough to breed the previous month (yellow).

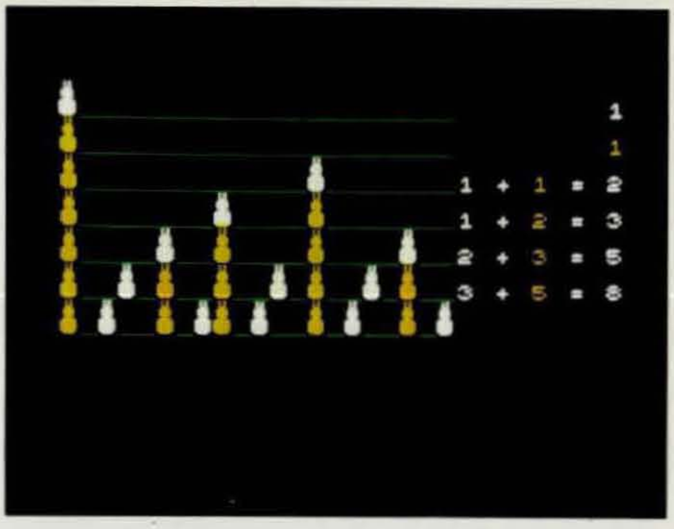


Fig. 2.45

Step 3 draws their offspring for the current month (white).

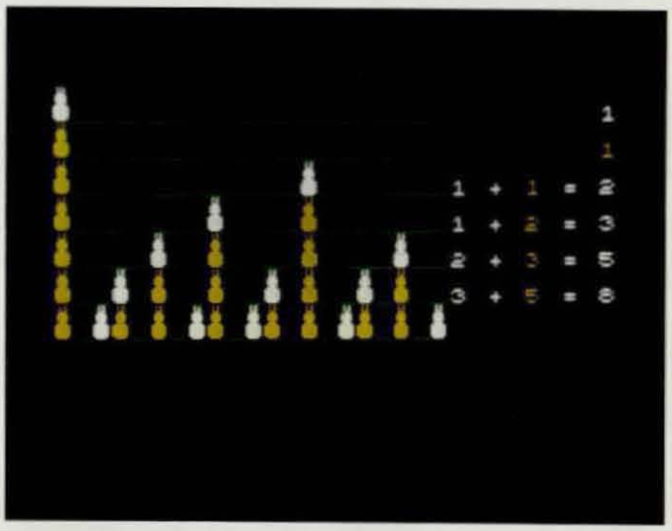
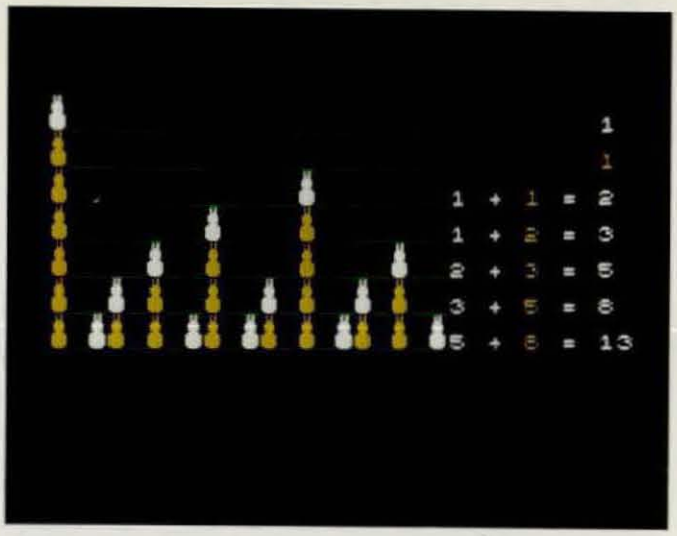


Fig. 2.46

Step 4 draws their offspring from the previous month, now old enough to breed (yellow).

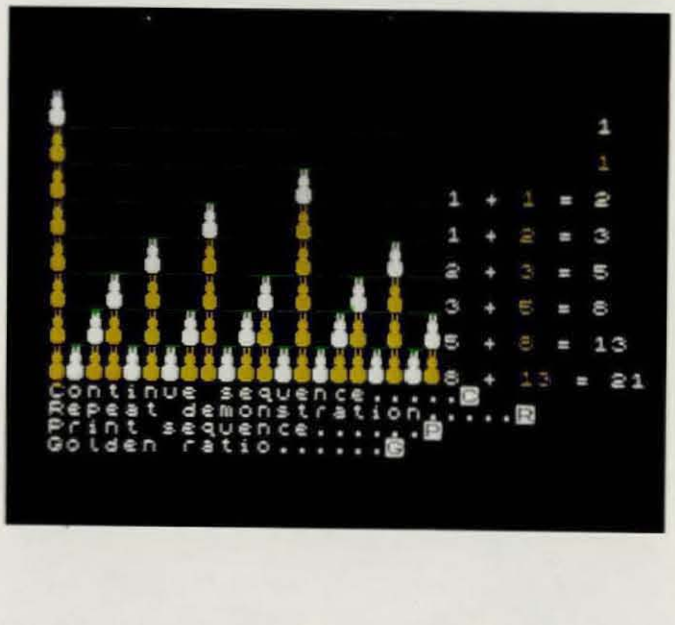


```
        1
        1
1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
3 + 5 = 8
5 + 8 = 13
```

Fig. 2.47

Step 5 prints the sum of white and yellow rabbits for the month.

Following the demonstration for the 8th month, pressing and releasing a key results in the option message as shown in figure 2.48

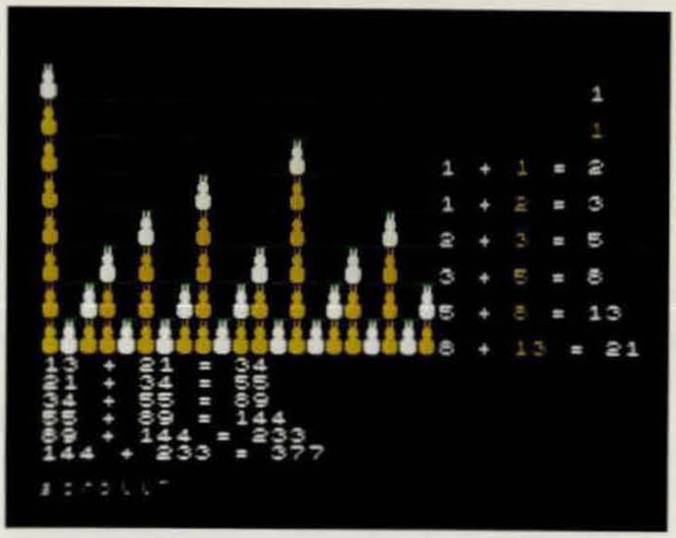


```
         1
         1
1 + 1  = 2
1 + 2  = 3
2 + 3  = 5
3 + 5  = 8
5 + 8  = 13
8 + 13 = 21
```

Fig. 2.48

The results of pressing the appropriate keys are as follows:

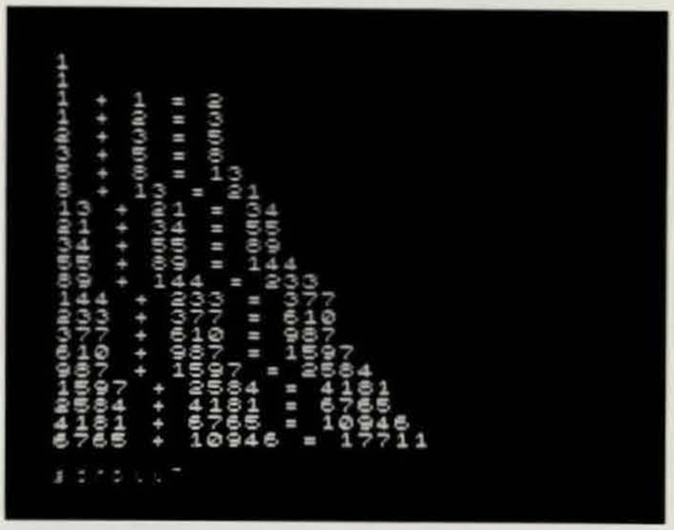
'r' repeats the rabbit sequence, from the first month.



```
                                          1
                                          1
                              1 + 1 = 2
                              1 + 2 = 3
                              2 + 3 = 5
                              3 + 5 = 8
                              5 + 8 = 13
                              8 + 13  = 21
13  + 21  = 34
21  + 34  = 55
34  + 55  = 89
55  + 89  = 144
89  + 144  = 233
144  + 233  = 377
```

Fig. 2.49

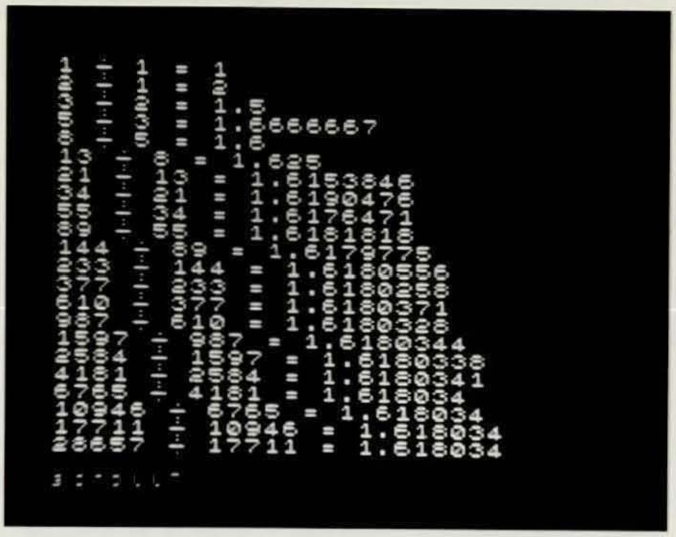
'c' continues the sequence.



```
1
1
1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
3 + 5 = 8
5 + 8 = 13
8 + 13  = 21
13  + 21  = 34
21  + 34  = 55
34  + 55  = 89
55  + 89  = 144
89  + 144  = 233
144  + 233  = 377
233  + 377  = 610
377  + 610  = 987
610  + 987  = 1597
987  + 1597  = 2584
1597  + 2584  = 4181
2584  + 4181  = 6765
4181  + 6765  = 10946
6765  + 10946  = 17711
```

Fig. 2.50

'p' prints the sequence from the first term.



```
1 ÷ 1 = 1
2 ÷ 1 = 2
3 ÷ 2 = 1.5
5 ÷ 3 = 1.6666667
8 ÷ 5 = 1.6
13 ÷ 8 = 1.625
21 ÷ 13 = 1.6153846
34 ÷ 21 = 1.6190476
55 ÷ 34 = 1.6176471
89 ÷ 55 = 1.6181818
144 ÷ 89 = 1.6179775
233 ÷ 144 = 1.6180556
377 ÷ 233 = 1.6180258
610 ÷ 377 = 1.6180371
987 ÷ 610 = 1.6180328
1597 ÷ 987 = 1.6180344
2584 ÷ 1597 = 1.6180338
4181 ÷ 2584 = 1.6180341
6765 ÷ 4181 = 1.618034
10946 ÷ 6765 = 1.618034
17711 ÷ 10946 = 1.618034
28657 ÷ 17711 = 1.618034
```

Fig. 2.51

'g' prints the ratio of successive pairs of terms of the Fibonacci Sequence.

Pressing any key will continue the sequences of figures 2.49, 2.50 and 2.51. Pressing the 'x' key will clear the screen and then display the options shown in figure 2.48.

Using the program

I have used this program in similar ways to that on number patterns, as a light-hearted way of introducing the Fibonacci sequence. With the least able groups, I was only concerned with the sequence of numbers and how they could be generated. The more able fourth year group again followed up the demonstration by producing and running a simple computer program to generate the sequence. The pupils were particularly interested in the rate at which the terms increased (almost exponentially) and so the sequence was printed until the numbers were expressed in standard form. I took this opportunity to do some revision work with the class on standard form and rounding off.

The display shown in figure 2.51 was added to use with an able fifth year class who had just completed some work on quadratic equations. As a final problem I had asked them to find the length of a rectangle whose width was one unit, such that taking away a unit square would leave a similar rectangle. The solution gives the golden ratio which was shown, using the program, to be the limit of the ratio of successive terms of the Fibonacci Sequence.

The program provided another example for the sixth form of defining a sequence iteratively, where each term is generated from the two previous terms.

Proof of the identity $u_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1 + \sqrt{5}}{2}\right)^n - \left(\frac{1 - \sqrt{5}}{2}\right)^n\right)$ where $u_1 = 1$, $u_2 = 1$ and $u_{k+1} = u_k + u_{k-1}$ gave an example of complete induction since the proof involves the assumption that the identity is true for $n \leq k$.

The printout for option 'g' demonstrates that $\lim_{n \to \infty} \frac{u_n}{u_{n-1}} = \varnothing$ (the golden ratio) and may be used to illustrate the concept of a limit.

The final program in this chapter is concerned with the prime numbers. It was written as a result of work on prime numbers being undertaken with a very able third year class. The pupils had produced the sieve of Eratosthenes for numbers up to 200, and such questions as 'Is 983 a prime number?' and 'What is the 1000th prime number?' arose. To answer these questions we set about producing a computer program to find the prime numbers. Some of the pupils had a good knowledge of BASIC and the following program was produced.

```
 10 DIM p(1000)
 20 LET m = 1
 30 LET p(1) = 2
 40 LET k = 3
 50 PRINT p(m)
 60 FOR j = 1 TO m
 70 IF INT(k/p(j))*p(j) = k THEN LET k = k + 1: GOTO 60
 80 NEXT j
 90 LET m = m + 1
100 LET p(m) = k
110 LET k = k + 1
120 GOTO 50
```

This prints out the prime numbers in a list.

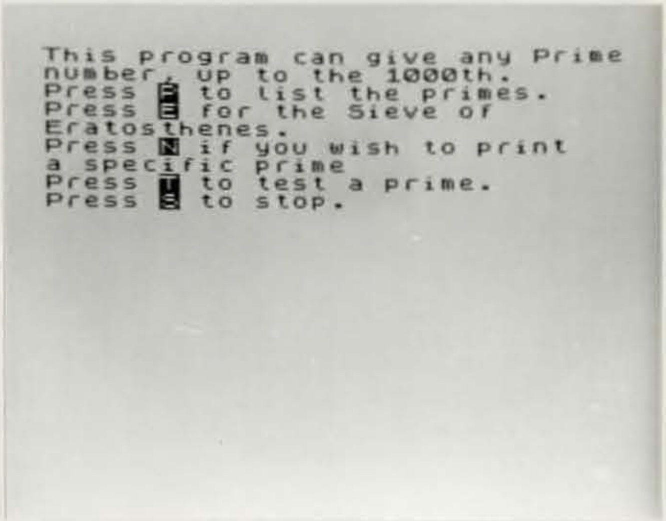
I had to translate the flow diagram decision box 'Is k divisible by the previous prime numbers?' into BASIC (line 70), but the remainder of the program was mostly pupils' work.

It had been noted by one pupil that, in completing the sieve, no other numbers were deleted after 13. From this, discussion led to the conclusion that if a number, k, had no prime factor which was less than $\sqrt{k}$, it would have no factor greater than $\sqrt{k}$. This led to line 65 `IF p(j)>SQRk THEN GOTO 90` being inserted, which decreased the number of steps to find each successive prime.

On the Sinclair Spectrum, the program was slow to execute, and so I produced the program 'prime'. It is similar to that above, but I have speeded up the execution by using the fact that any prime number over 3 is of the form 6n ± 1 (this is easily proved using simple algebra) and so not all the natural numbers need to be tested. Setting up the array p is still very slow and so, after the initial execution, the program was saved using `SAVE LINE 3`. This ensures that the array p contains the prime numbers on loading the program, by running the program from line 3, so avoiding `DIM p(1000)` which would set all the members of the array to zero.

I find this program a convenient way of storing the prime mumbers, which can be output as a list, or as the sieve of Eratosthenes. Children enjoy checking their results for the sieve against those displayed by the computer.

<u>3. Prime numbers.</u> Program name 'prime'.

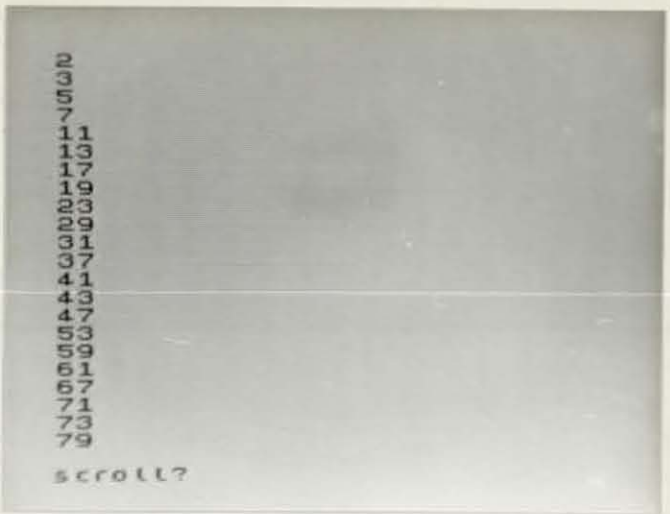


```
This program can give any Prime
number, up to the 1000th.
Press P to list the primes.
Press E for the Sieve of
Eratosthenes.
Press N if you wish to print
a specific prime
Press T to test a prime.
Press S to stop.
```

Fig. 2.52

Initial display, on loading the program.

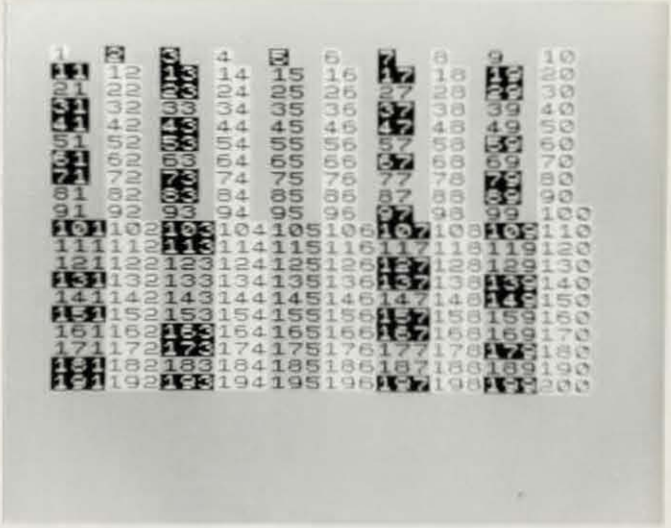
The following series of figures shows the results of pressing the various keys.



```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79

scroll?
```

Fig. 2.53

Pressing 'p' lists the prime numbers up to the 1000th.

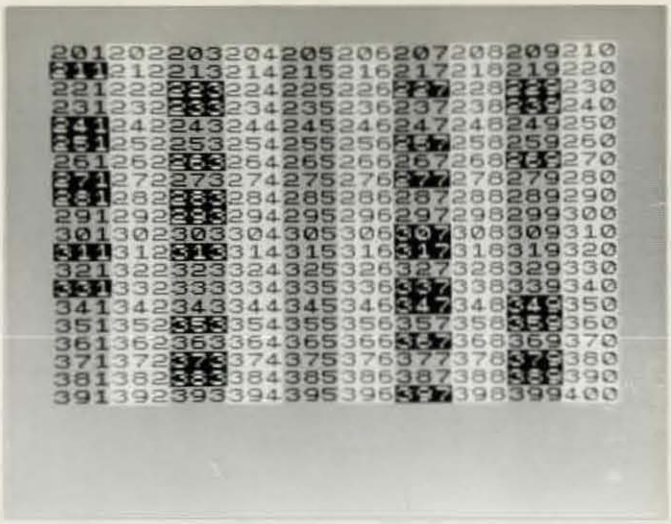


```
1  2  3  4  5  6  7  8  9  10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100
101102103104105106107108109110
111112113114115116117118119120
121122123124125126127128129130
131132133134135136137138139140
141142143144145146147148149150
151152153154155156157158159160
161162163164165166167168169170
171172173174175176177178179180
181182183184185186187188189190
191192193194195196197198199200
```

Fig. 2.54

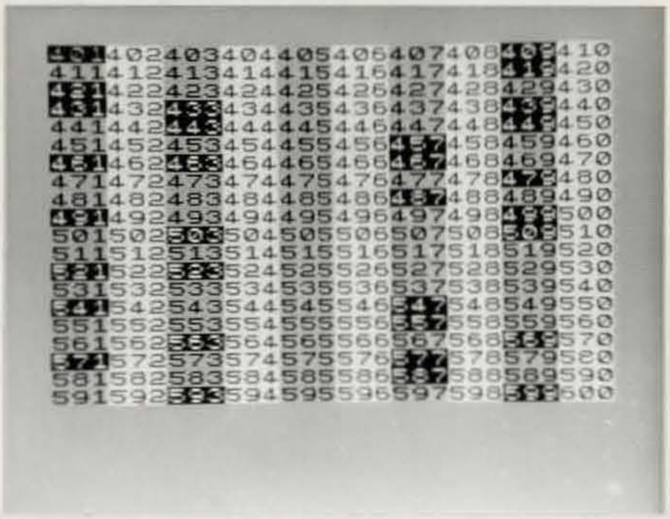
Pressing 'e' results in the sieve of Eratosthenes being printed. Prime numbers are printed white on a black background. Other numbers are printed in black on white, the background being set to BRIGHT 1 for even numbers, BRIGHT 0 for odd numbers greater than 1.

Pressing any key continues the sieve.



```
201202203204205206207208209210
211212213214215216217218219220
221222223224225226227228229230
231232233234235236237238239240
241242243244245246247248249250
251252253254255256257258259260
261262263264265266267268269270
271272273274275276277278279280
281282283284285286287288289290
291292293294295296297298299300
301302303304305306307308309310
311312313314315316317318319320
321322323324325326327328329330
331332333334335336337338339340
341342343344345346347348349350
351352353354355356357358359360
361362363364365366367368369370
371372373374375376377378379380
381382383384385386387388389390
391392393394395396397398399400
```

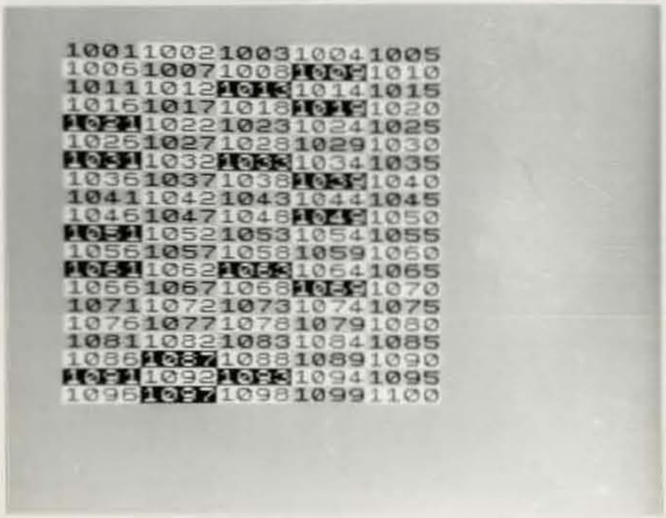
Fig. 2.55



```
401402403404405406407408409410
411412413414415416417418419420
421422423424425426427428429430
431432433434435436437438439440
441442443444445446447448449450
451452453454455456457458459460
461462463464465466467468469470
471472473474475476477478479480
481482483484485486487488489490
491492493494495496497498499500
501502503504505506507508509510
511512513514515516517518519520
521522523524525526527528529530
531532533534535536537538539540
541542543544545546547548549550
551552553554555556557558559560
561562563564565566567568569570
571572573574575576577578579580
581582583584585586587588589590
591592593594595596597598599600
```

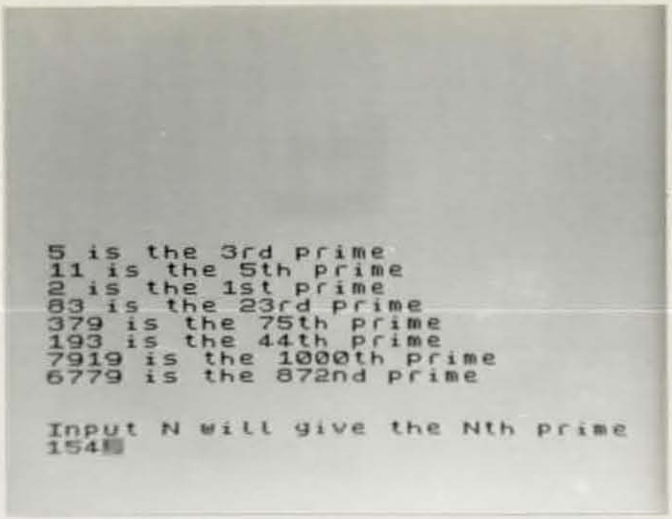
Fig. 2.56

After 1000 numbers have been printed, the sieve continues as in figure 2.57, with five 4-figure numbers to each line.



```
10011002100310041005
10061007100810091010
10111012101310141015
10161017101810191020
10211022102310241025
10261027102810291030
10311032103310341035
10361037103810391040
10411042104310441045
10461047104810491050
10511052105310541055
10561057105810591060
10611062106310641065
10661067106810691070
10711072107310741075
10761077107810791080
10811082108310841085
10861087108810891090
10911092109310941095
10961097109810991100
```

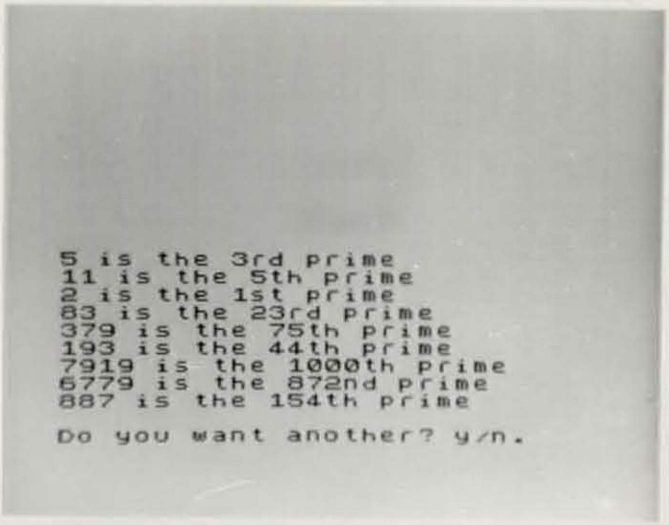
Fig. 2.57



```
5 is the 3rd prime
11 is the 5th prime
2 is the 1st prime
83 is the 23rd prime
379 is the 75th prime
193 is the 44th prime
7919 is the 1000th prime
6779 is the 872nd prime

Input N will give the Nth prime
154█
```

Fig. 2.58

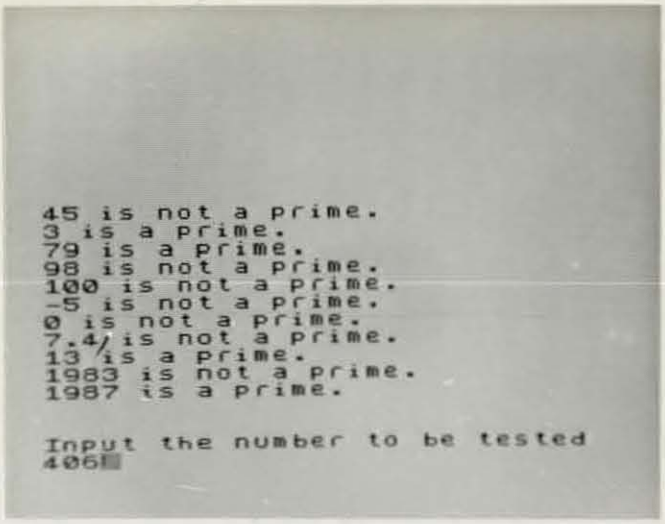
Option 'n' - the n^{th} prime number is printed when n is input. The screen display shown is after input of 3, 5, 1, 23, 75, 44, 1000 and 872. 154 is about to be input.



```
5 is the 3rd prime
11 is the 5th prime
2 is the 1st prime
83 is the 23rd prime
379 is the 75th prime
193 is the 44th prime
7919 is the 1000th prime
6779 is the 872nd prime
887 is the 154th prime

Do you want another? y/n.
```

Fig. 2.59

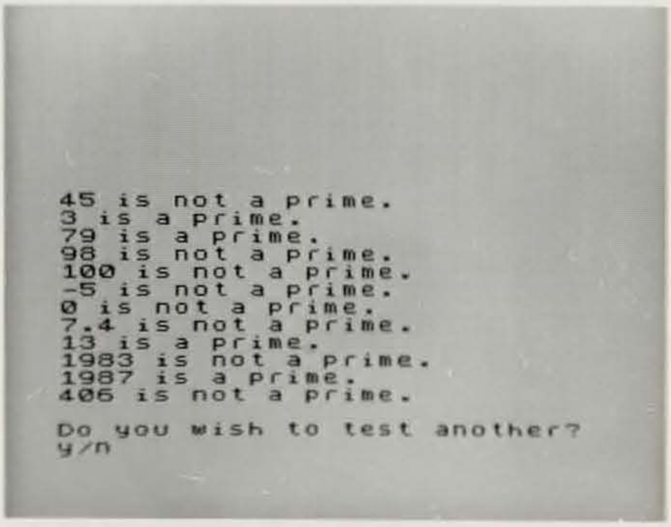
The screen display after 154 is input. Pressing the 'y' key will await input, as in figure 2.59, pressing the 'n' key will return to the initial display of figure 2.52.



```
45 is not a prime.
3 is a prime.
79 is a prime.
98 is not a prime.
100 is not a prime.
-5 is not a prime.
0 is not a prime.
7.4 is not a prime.
13 is a prime.
1983 is not a prime.
1987 is a prime.

Input the number to be tested
406
```

Fig. 2.60

Option 't' - tests whether the number which is input is prime or not. Any number up to 7919^2 may be tested (7919 is the 1000th prime). The figure above shows the screen display after 45, 3, 79, 98, 100, -5, 0, 7.4, 13, 1983 and 1987 have been input. 406 is about to be input.



```
45 is not a prime.
3 is a prime.
79 is a prime.
98 is not a prime.
100 is not a prime.
-5 is not a prime.
0 is not a prime.
7.4 is not a prime.
13 is a prime.
1983 is not a prime.
1987 is a prime.
406 is not a prime.

Do you wish to test another?
y/n
```

Fig. 2.61

The screen display after 406 has been input. Pressing the 'y' key will await input as in figure 2.60, pressing the 'n' key will return to the initial display of figure 2.52.

Chapter Three

The final program which I have produced for inclusion in this study is concerned with loci. The idea of a locus occurs throughout many mathematics texts, but I have found that a great deal of time is required to investigate the topic in a practical way, to any depth. Using the computer to draw the loci lifts the burden of producing numerous time-consuming diagrams, and allows the pupils to concentrate on the mathematics behind the locus. I use this program to promote discussion about loci, how they change when the conditions are changed and, where possible, the underlying mathematics.

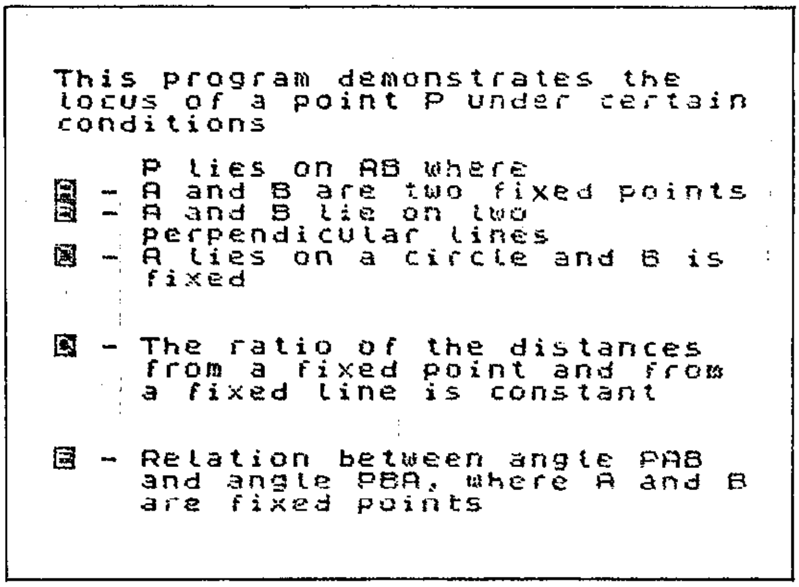
There are five main sections in the program, four of which stem from the School Mathematics Project 'O' level course, the other is intended for use at 'A' level.

Throughout the program pressing a key will execute the next stage. So that only one stage is executed at a time, the program continues to run only when the key is released. Each locus is drawn by (i) drawing any necessary construction lines to find a position of P, (ii) erasing those lines and plotting the position of P. This is then repeated for the next position of P, until all the positions have been plotted. During the drawing of the locus, holding down the 'm' key will halt this process at the stage after the construction lines are drawn, holding down the 'n' key will halt the process after these lines have been erased. Releasing the key continues the demonstration. The scale is adjusted after the required input, to accommodate the figures on the screen. When the locus is complete, pressing the 'r' key repeats the demonstration, pressing another key returns the initial option display to the screen.

The program name is 'loci'. On pressing RUN, the initial options are displayed on the screen (see figure 3.1). As with the transformations program, the printing is white on a black background.

I have, so far, used this program with two groups of pupils, an able 4th year set and with a sixth form group. I shall describe the way in which I have used it within each section.

The program listing is given in appendix E.



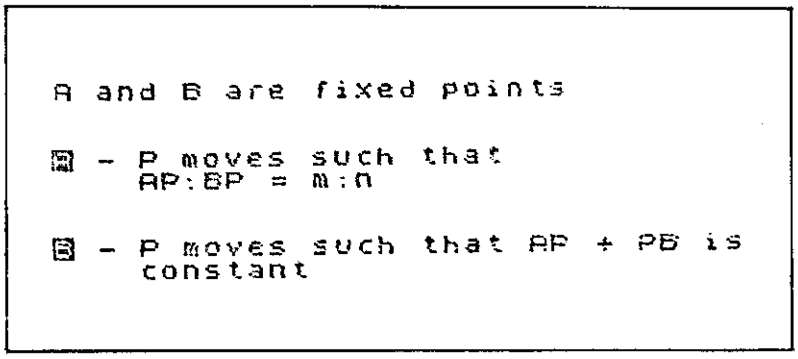
```
This program demonstrates the
locus of a point P under certain
conditions

    P lies on AB where
▣ - A and B are two fixed points
▣ - A and B lie on two
    perpendicular lines
▣ - A lies on a circle and B is
    fixed


▣ - The ratio of the distances
    from a fixed point and from
    a fixed line is constant


▣ - Relation between angle PAB
    and angle PBA, where A and B
    are fixed points
```

Fig. 3.1

The option is chosen by pressing the appropriate key.

Option A

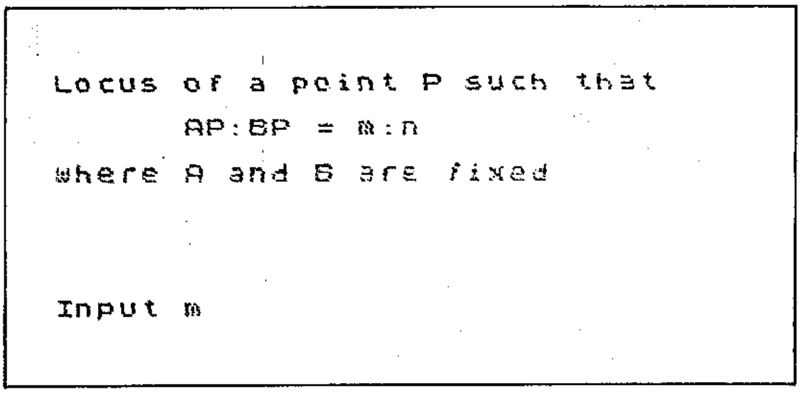
Figure 3.2 shows the screen display when option A is chosen.



```
A and B are fixed points

▣ - P moves such that
    AP:BP = m:n

▣ - P moves such that AP + PB is
    constant
```

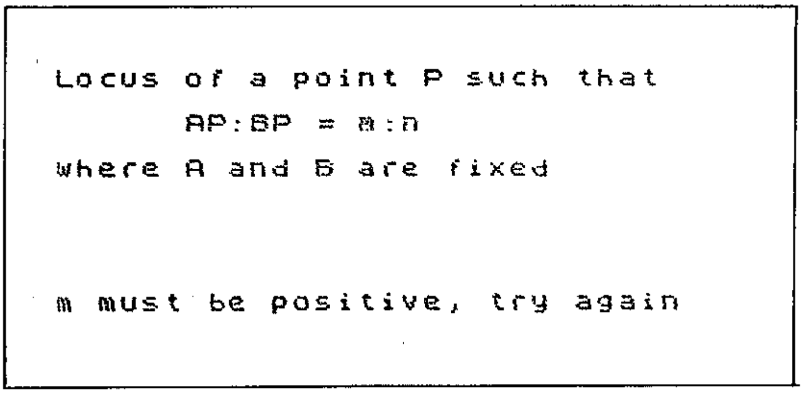
Fig. 3.2

Pressing and releasing the 'a' key at this stage results in the display as in figure 3.3, which requires m to be input. If m is not positive then the error message in figure 3.4 is printed. When m has been accepted, n, which must also be positive, is input (see fig.3.5). After n has been input, the letters m and n are replaced with their values, and the message in figure 3.6 is printed. When a key is pressed and released the locus is drawn as already explained.



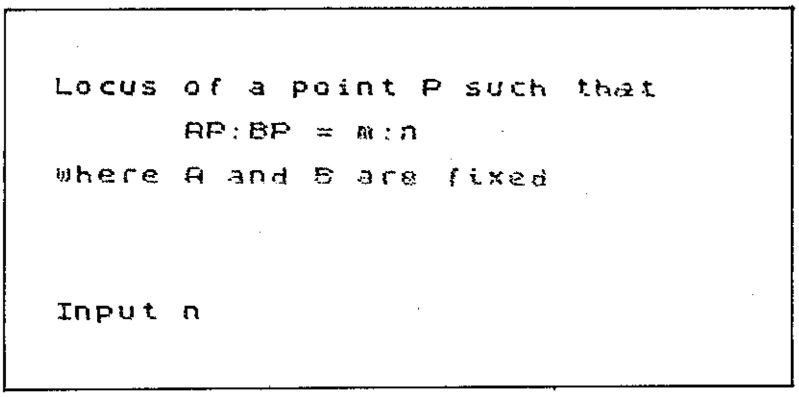
```
Locus of a point P such that
      AP:BP = m:n
where A and B are fixed


Input m
```

Fig. 3.3
Input of m is awaited.



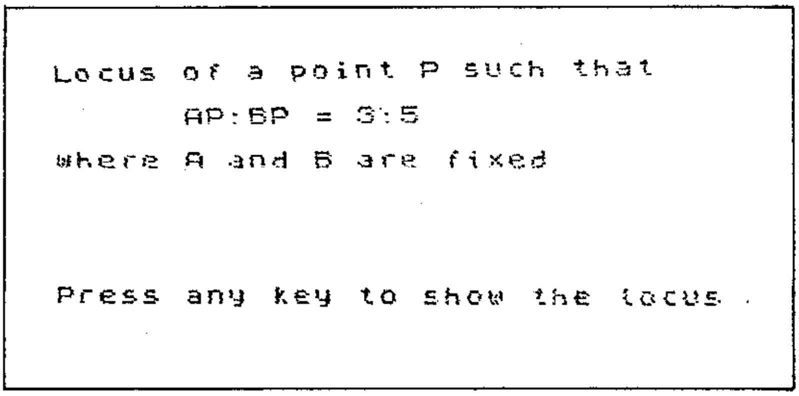
```
Locus of a point P such that
      AP:BP = m:n
where A and B are fixed


m must be positive, try again
```

Fig. 3.4
Error message if m is not positive.



```
Locus of a point P such that
      AP:BP = m:n
where A and B are fixed


Input n
```

Fig. 3.5
Input of n is awaited.



```
Locus of a point P such that
      AP:BP = 3:5
where A and B are fixed


Press any key to show the locus.
```

Fig. 3.6
The values of m and n are printed.

I used this section with the 4th year group in an introductory lesson on loci. It had been explained what was meant by a locus and I asked what the locus of P was, where P was a constant distance from a fixed point. Some pupils suggested a circle immediately, but others needed a concrete example, for which I used a 'conker' being whirled round on a string. When asked the locus of P, where the distances of P from two fixed points, A and B, were equal, there was no response. Asking for one position of P produced the answer 'midway between A and B'. At this stage, I asked the pupils to draw a diagram with A and B marked, and to draw some possible positions of P, after which the mediator was suggested. I then demonstrated the locus using the computer - this is shown in figures 3.7-3.9.

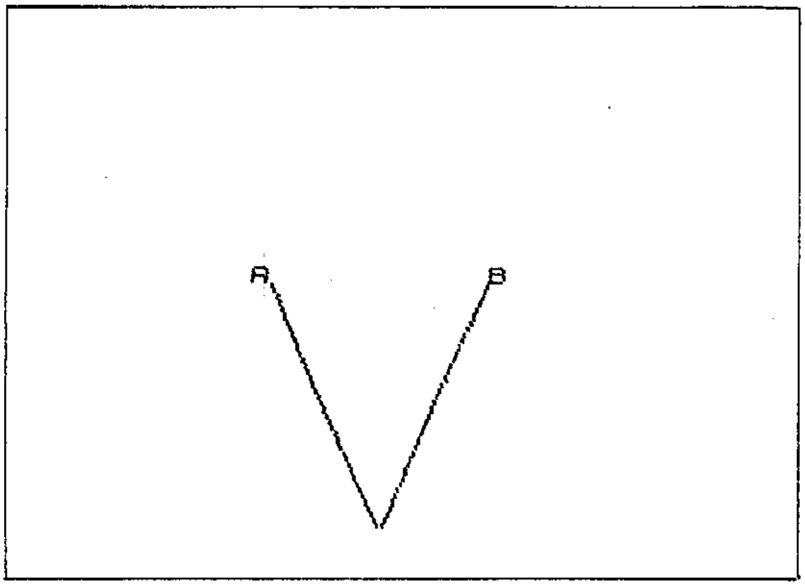


Fig. 3.7
The 'm' key is held down to show the first position of P.

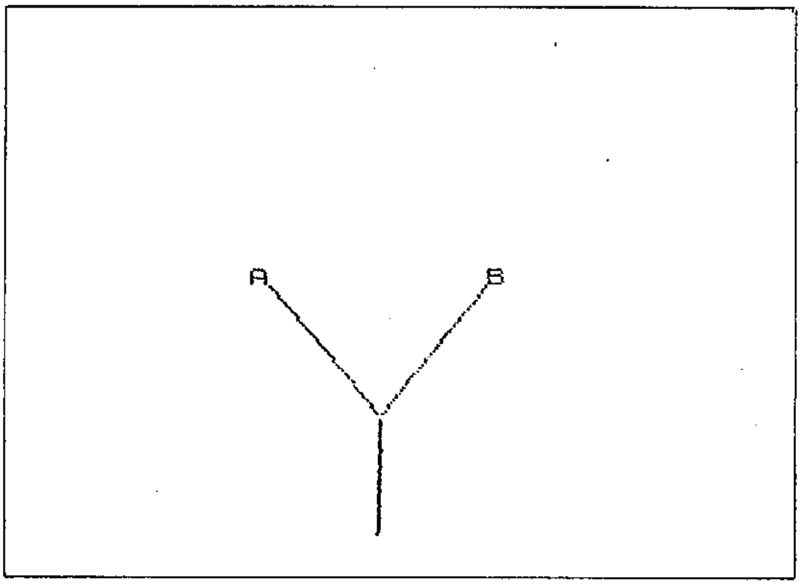


Fig. 3.8
The 'm' key is held down to show P in an intermediate position.

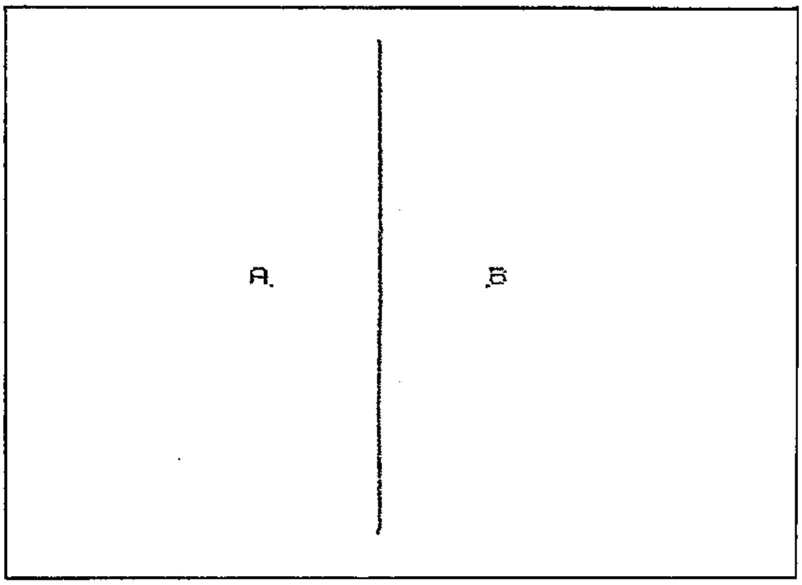


Fig. 3.9
The completed
locus.

Following this demonstration the class considered what the locus of P would be if the ratio PA:PB was not 1:1. With the ratio 3:5, the first suggestion was that the locus would be a line perpendicular to AB cutting AB in the ratio 3:5. Another suggestion was that it would curve something like a parabola. No-one predicted a circle, and the pupils were surprised when the locus was drawn. Figures 3.10-3.12 show the demonstration of the locus.

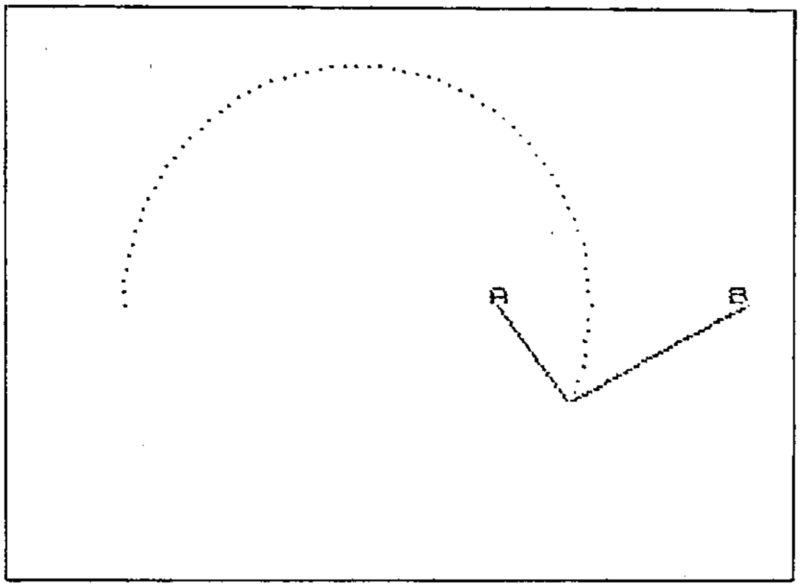


Fig. 3.10
An intermediate
position to
show that
PA:PB = 3:5.

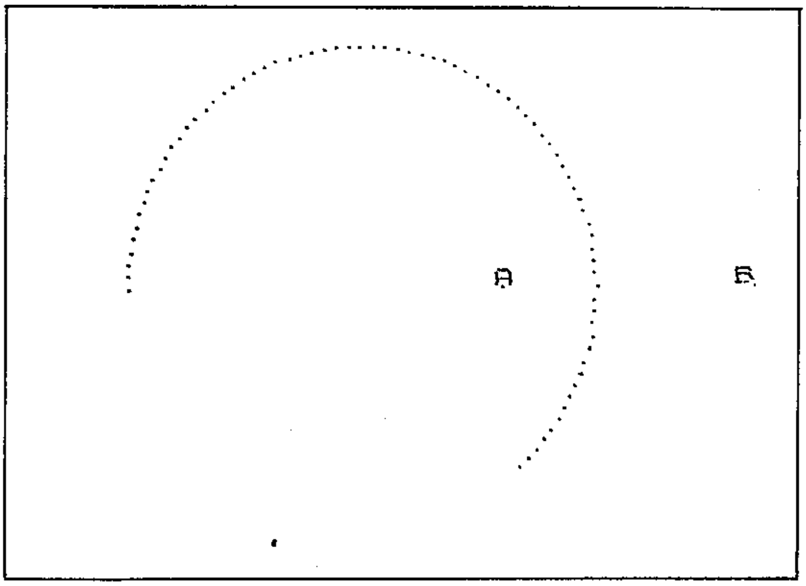


Fig. 3.11
An intermediate position shown without the construction lines (holding down the 'n' key).

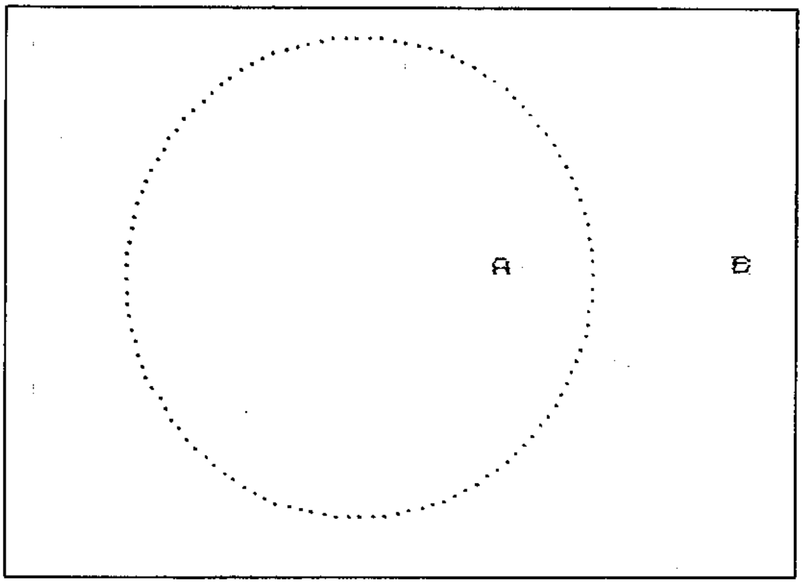


Fig. 3.12
The complete locus.

Further discussion led to the realisation that the line through A and B formed a diameter of the circle, and that the centre was to the left of A. Pressing a key plots the centre of the circle and draws the complete circle (see figure 3.13).

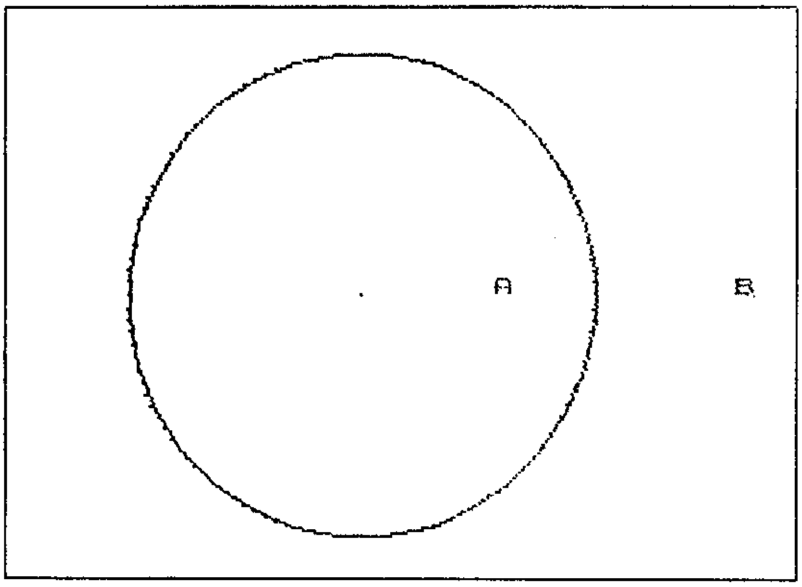


Fig. 3.13

Several other ratios were demonstrated to show that a circle was produced in each case. Figures 3.14 and 3.15 show that when $m/n > 1$ the circle encloses B. (The computer program automatically adjusts the positions of A and B to accommodate the complete figure on the screen.)

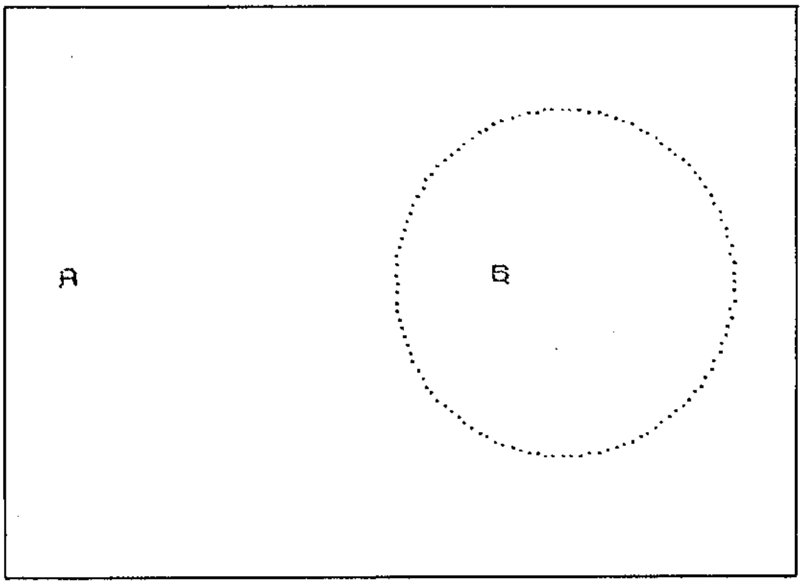


Fig. 3.14
The locus of
P when
PA:PB = 3:1.

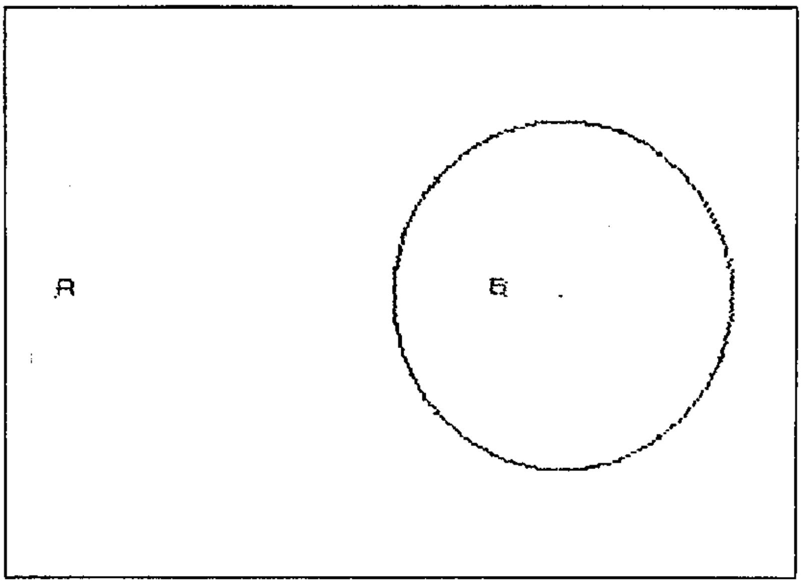
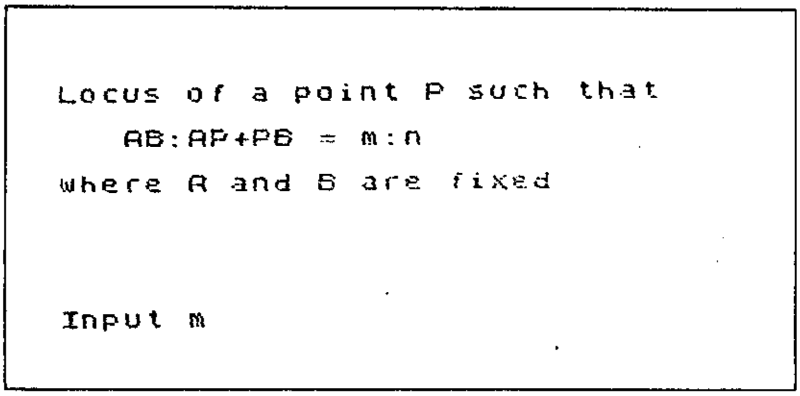


Fig. 3.15
The complete circle has been drawn, and its centre plotted.

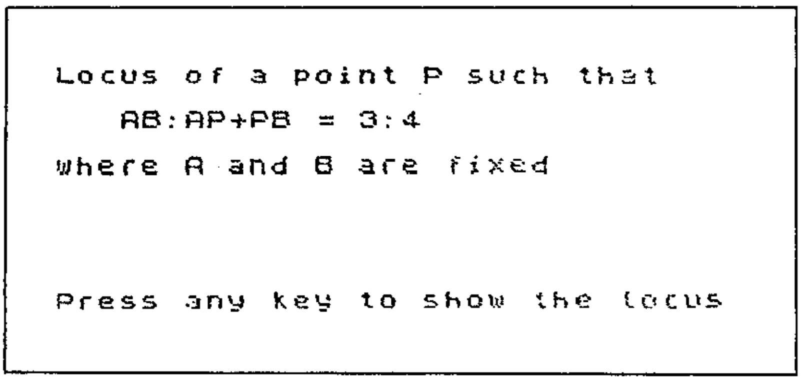
To conclude this lesson on loci, I asked the pupils what they thought the locus of P would be if AP + PB were constant. Most thought that it would be a circle. Demonstration of the locus under this condition required the 'b' key to be pressed at the stage shown in figure 3.2 - this resulted in figure 3.16.



```
Locus of a point P such that
   AB:AP+PB = m:n
where A and B are fixed


Input m
```

Fig. 3.16

The ratio 3:2 was suggested, but this resulted in an error message saying that n could not be less than m, and so the ratio 3:4 was chosen (see figure 3.17).



```
Locus of a point P such that
   AB:AP+PB = 3:4
where A and B are fixed


Press any key to show the locus
```

Fig. 3.17

Figure 3.18 shows the stage by which most pupils realised that the locus was not a circle.

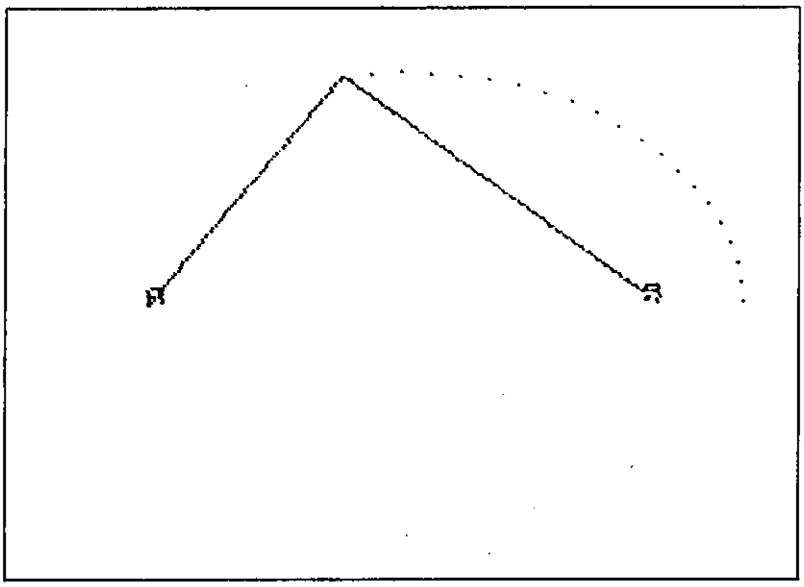


Fig. 3.18
An intermediate
stage with
the 'm' key
held down.

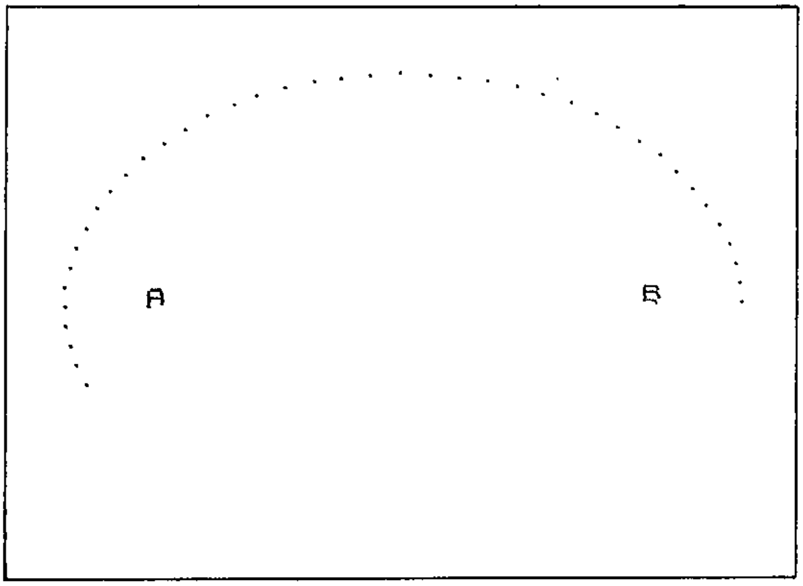


Fig. 3.19
A later stage
with the 'n'
key held down.

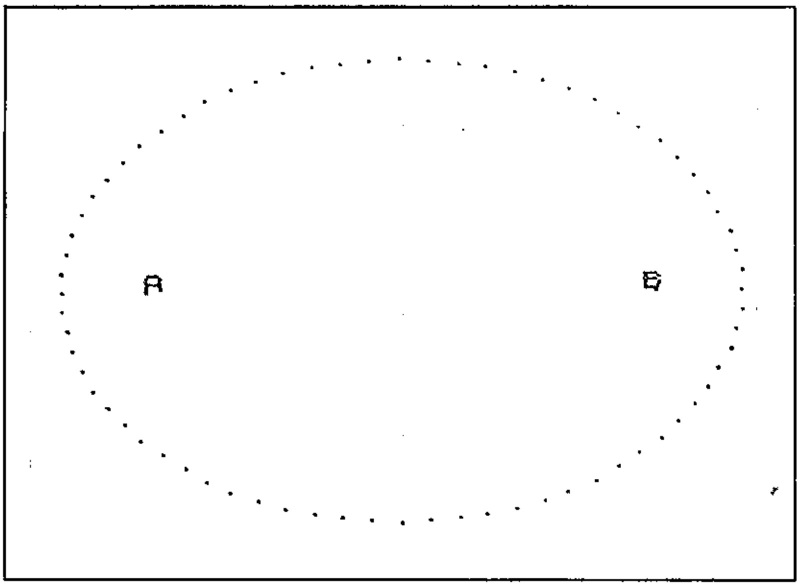


Fig. 3.20
The complete
locus of P.

Some other values were chosen for m and n, which showed that the ellipse approached a circle as m/n approached 0 and that the ellipse became long and thin as m/n approached 1. After discussion, it was agreed that m and n could be equal, in which case P is on AB. This was confirmed using the computer. Figure 3.21 shows an intermediate stage in plotting the locus when PA + PB = AB.

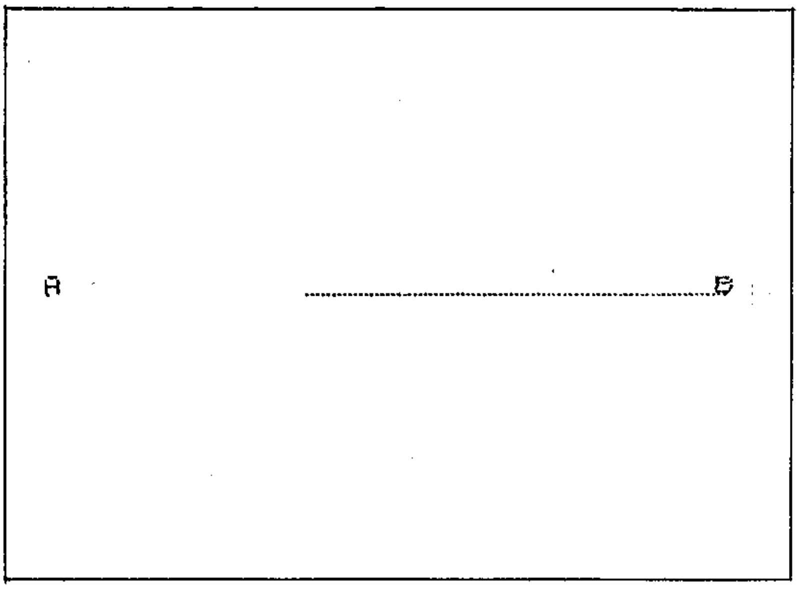


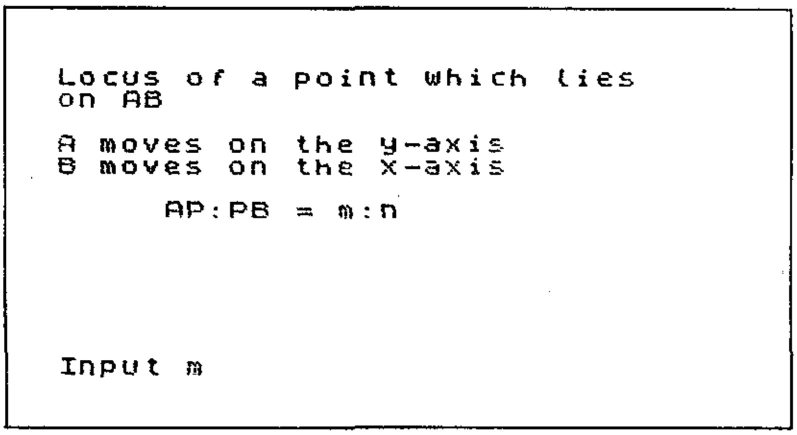
Fig. 3.21

With the sixth form group of pupils, I used this section of the program to reinforce the idea of a locus and how to find its cartesian equation. I first demonstrated that the locus of P such that PA = PB was the mediator of AB (see figure 3.9). I then asked how the equation of the mediator of two points could be found. The suggestion from the pupils was to find the equation of a line perpendicular to AB through the midpoint of AB, which was found for two specified points. I then showed that the same equation could be obtained by writing down the condition on P, i.e. $\sqrt{(x-x_1)^2 + (y-y_1)^2} = \sqrt{(x-x_2)^2 + (y-y_2)^2}$, and simplifying it.

The demonstration that the locus of P when PA:PB = 3:5 is a circle (see figure 3.13) surprised this group as it had the fourth year pupils. The suggestion that the equation could be found by finding the centre of the circle and its radius led to the emergence of some useful ideas concerning the division of a line in a given ratio, internally and externally. Using A as (0,0) and B as (8,0) gave the centre $(-4\frac{1}{2},0)$ and the radius $7\frac{1}{2}$ units. It was pointed out that we were assuming that the locus was a circle without proof. By considering the condition on P, i.e. PA:PB = 3:5, gave $\sqrt{x^2 + y^2} : \sqrt{(x-8)^2 + y^2} = 3:5$, which led to the required equation $x^2 + y^2 + 9x - 36 = 0$. The pupils recognised this as the equation of a circle, and worked through several examples using different ratios and different points. I referred to these results in a later lesson which dealt with the division of a line in a given ratio more formally.

Option B

Figure 3.22 shows the display which results from choosing initial option B as shown in figure 3.1.

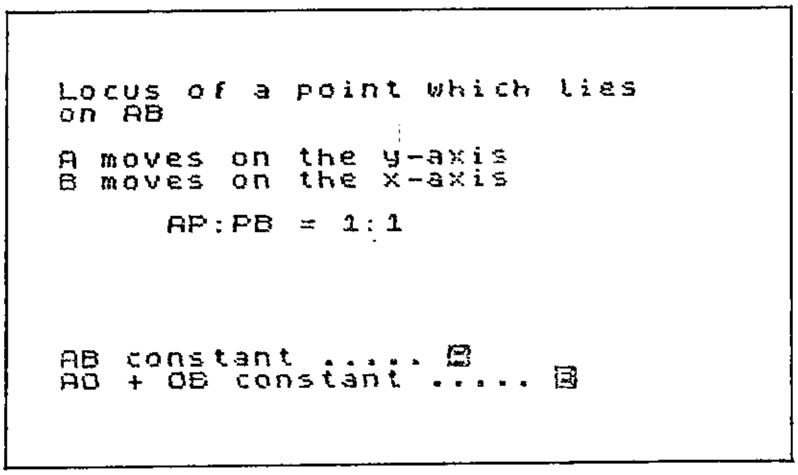


```
Locus of a point which lies
on AB

A moves on the y-axis
B moves on the x-axis

     AP:PB = m:n




Input m
```

Fig. 3.22

This section of the program was written as a follow-up to an investigation in the School Mathematics Project book G (4th year) which asks what the locus is of a man who is standing on a slipping ladder which has one end leaning against a wall and the other end on the ground. The first suggestion from the class was that the man would fall in a straight line, but most pupils thought that the locus would be a curve. It was suggested by one pupil that the curve would be concave - later questioning revealed that she was thinking of an envelope. No-one suggested that the path would be circular.

As before, I asked the pupils to draw a diagram to illustrate the locus by plotting the midpoint of the ladder in various positions. All the children were surprised to find that the locus was a quarter circle. This investigation was extended to consider a line of fixed length moving with one end on each of the axes. To set up the situation using the computer, m and n were both input as 1 (see figure 3.23).



```
Locus of a point which lies
on AB

A moves on the y-axis
B moves on the x-axis

      AP:PB = 1:1



AB constant ..... a
AO + OB constant ..... b
```

Fig. 3.23

I had already input m and n into the computer, and had also pressed the 'a' key for option A. This displays the two axes, and awaits a key to be pressed to show the locus. By turning down the contrast of the television, the screen appeared blank until I was ready for the demonstration.

Figures 3.24-3.26 show the demonstration.

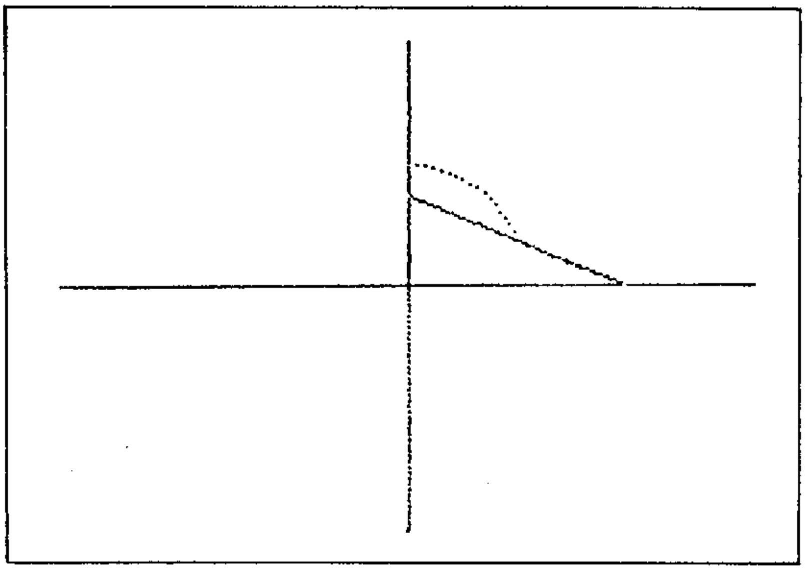


Fig 3.24
The 'm' key is held down to show AB in the first quadrant.

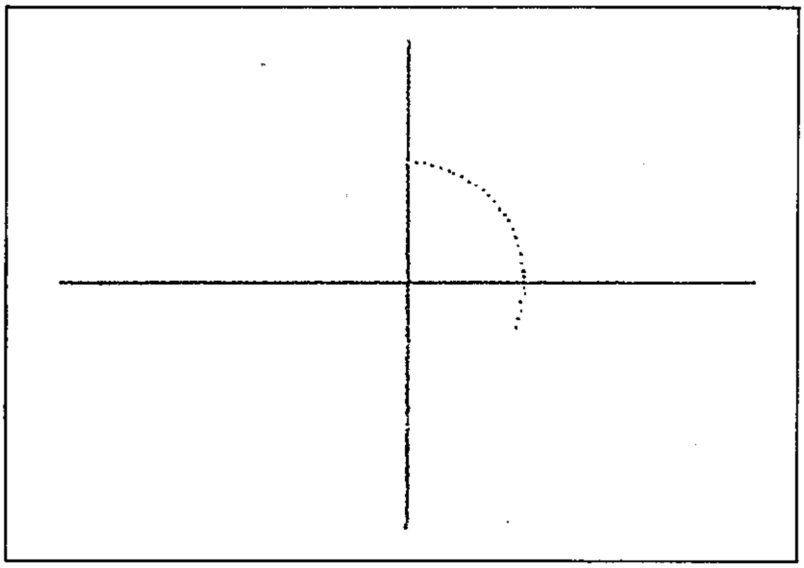


Fig 3.25
The 'n' key is held down to show the locus when AB is in the second quadrant.

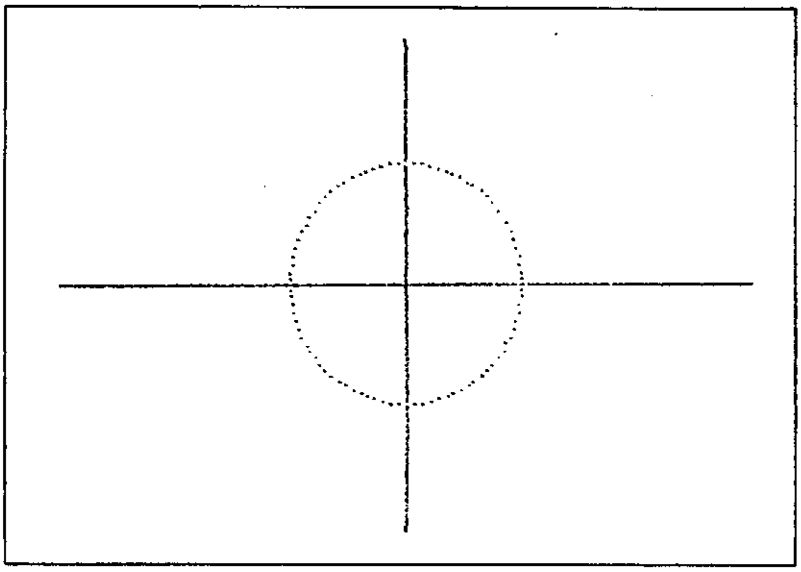


Fig 3.26
The complete locus.

This demonstration was followed by discussion on how the locus changes when the man is higher or lower than the centre of the ladder. One pupil suggested that the locus would be an ellipse, which was confirmed by computer demonstration using several values of m and n. Figures 3.27 and 3.28 show two examples.

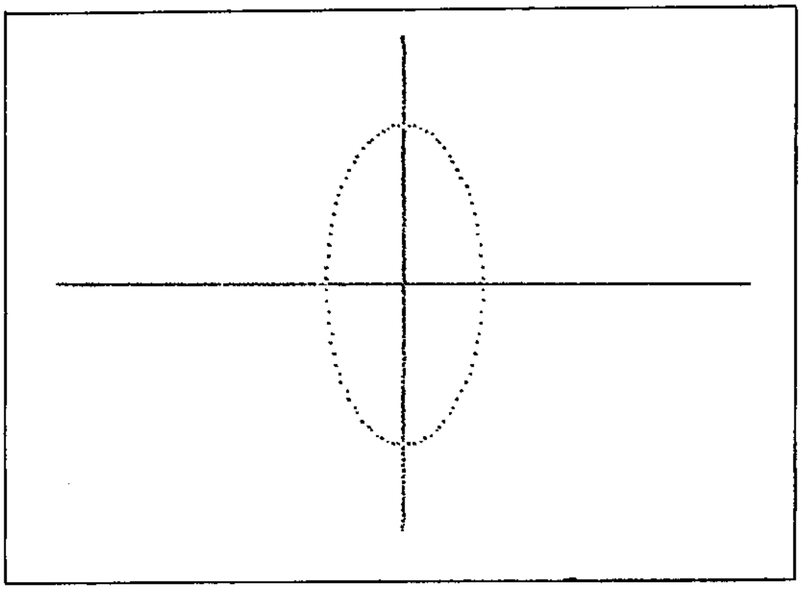


Fig. 3.27
PA:PB = 1:2

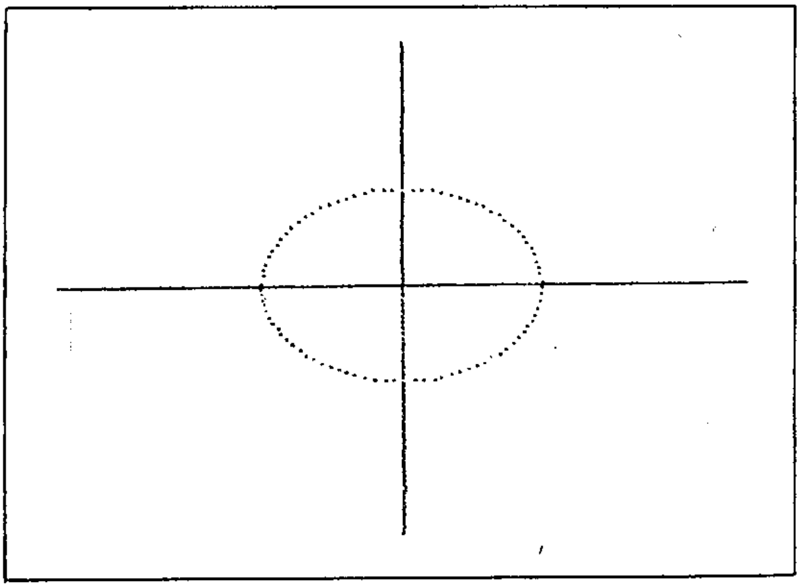


Fig. 3.28
PA:PB = 3:2

The pupils were interested in these findings, and so we considered the locus of P, where P divides AB externally. I introduced this by drawing on the blackboard a ladder whose top was leaning over a garden wall. Guided discussion led to this being expressed as the ratio of -1:2, when the centre of the ladder was at the top of the wall. The pupils were very able 4th year pupils, and so had little trouble in visualising how thw ladder would move although appreciating that, in practice, such motion would be impossible. Figure 3.29 shows part of the locus for PA:PB = -1:2, halted so that the relative lengths may be discussed. Figure 3.30 shows a similar stage for PA:PB = 5:-2.

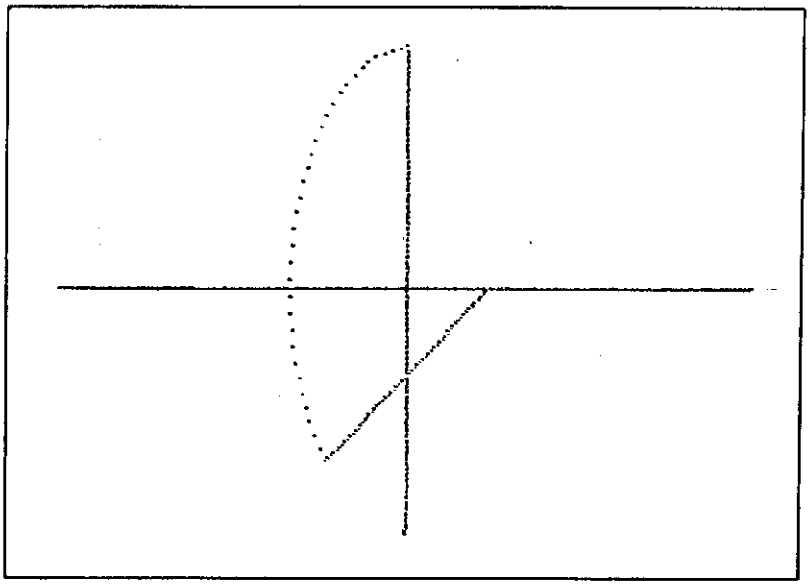


Fig. 3.29

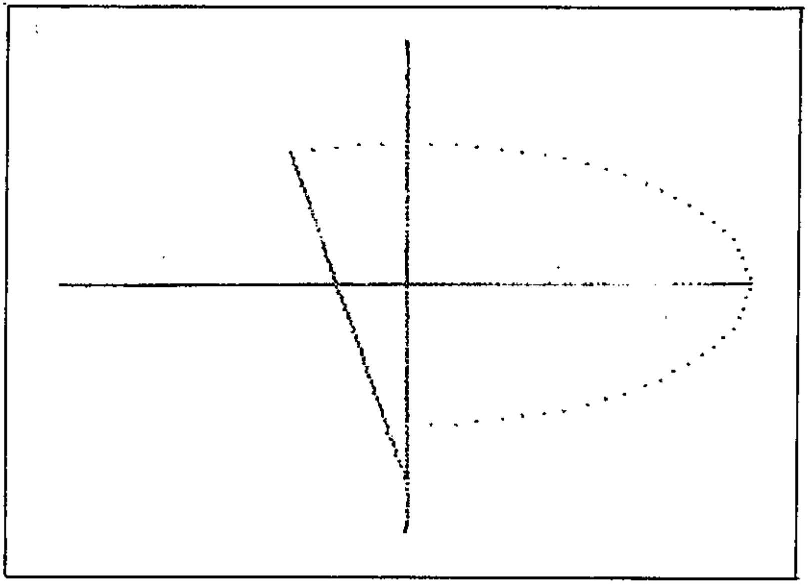


Fig. 3.30

Questioning the pupil who had expected a concave curve confirmed that she had been thinking of the envelope of an astroid which she had met earlier in the S.M.P. course. I asked if the pupils recalled an envelope which they had previously drawn similar to the astroid. Some remembered drawing the envelope of a parabola by loining points on the two axes. The difference between the two envelopes was realised (AO + OB is constant in the latter case) and so I asked what they thought the locus of P would be under this condition. The general opinion was that a curve would be produced. Several ratios were demonstrated using the computer, some of which are shown in figures 3.31-3.37.

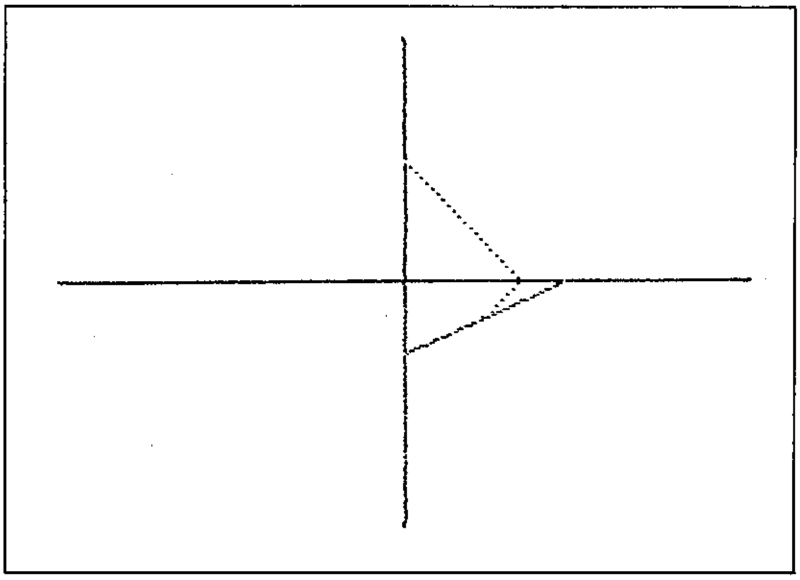


Fig. 3.31
An intermediate
stage for
PA:PB = 1:1,
with the 'm'
key held down.

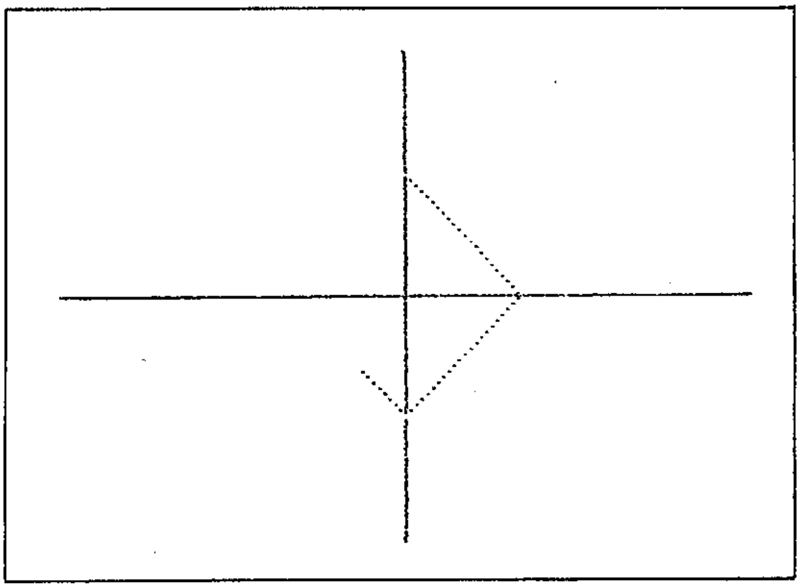


Fig. 3.32
An intermediate
stage for
PA:PB = 1:1,
with the 'n'
key held down.

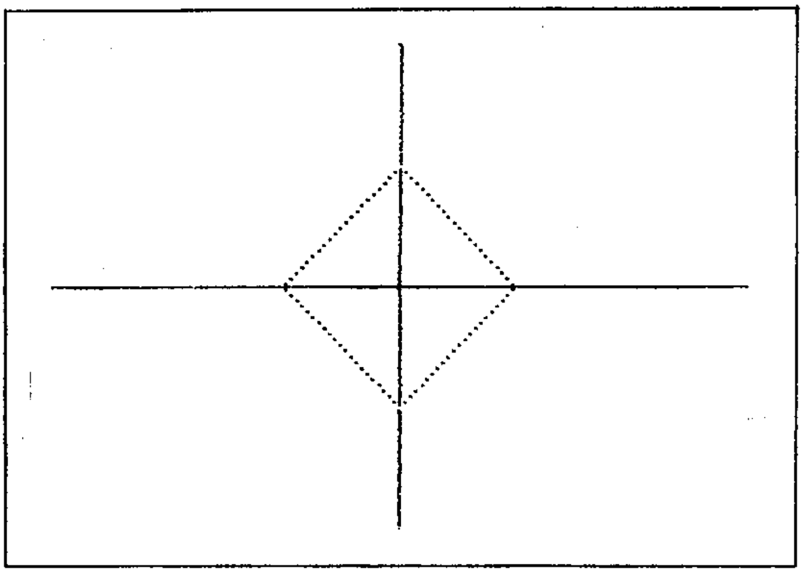


Fig. 3.33
The completed
locus for
PA:PB = 1:1.

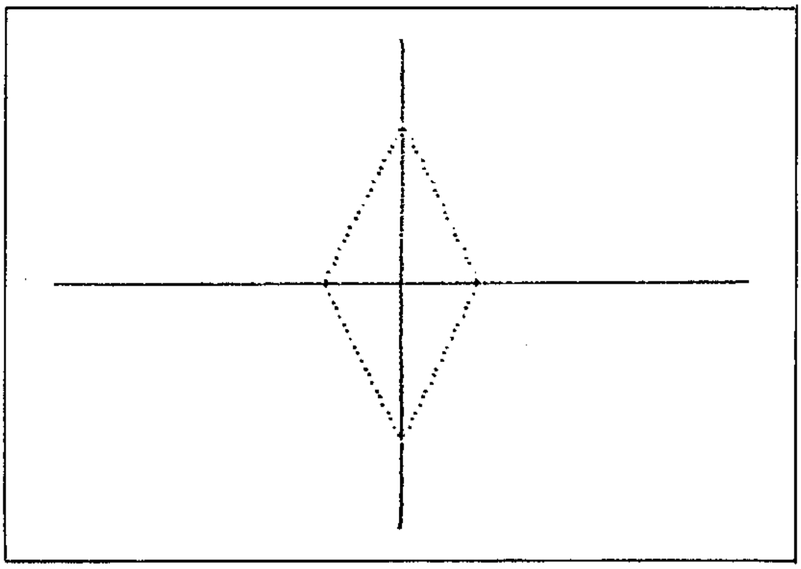


Fig. 3.34
The completed
locus for
PA:PB = 1:2.

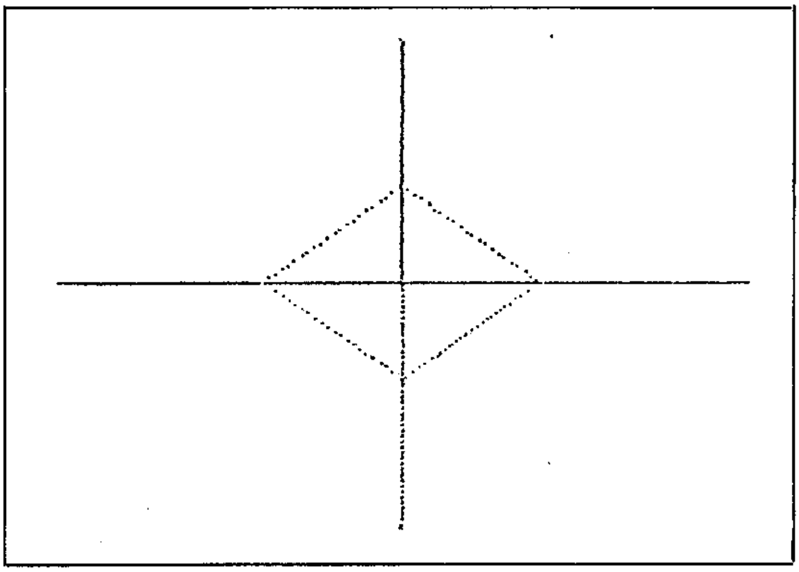


Fig. 3.35
The completed
locus for
PA:PB = 3:2.

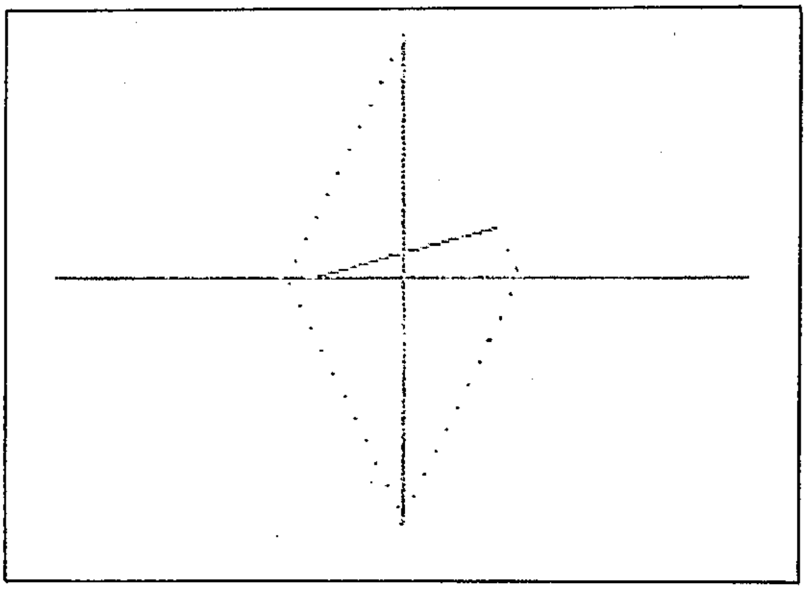


Fig. 3.36
An intermediate
stage for
PA:PB = -1:2.

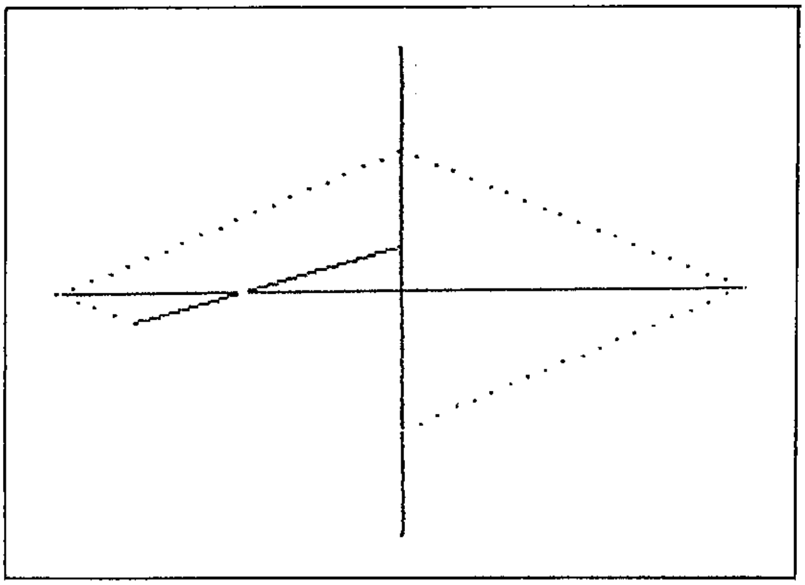
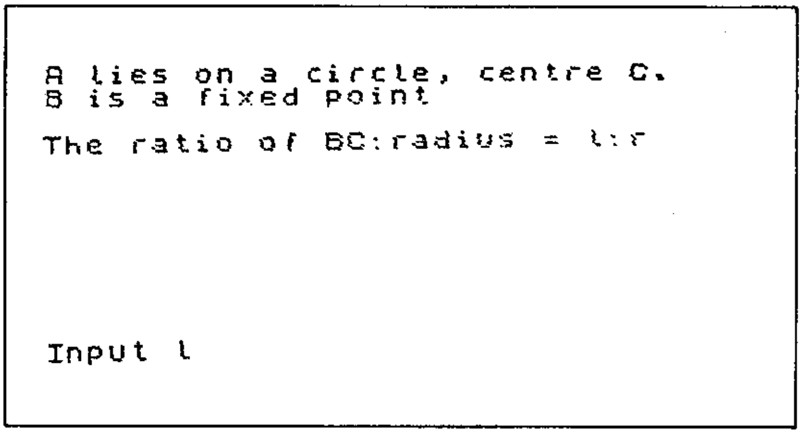


Fig. 3.37
An intermediate
stage for
PA:PB = 5:-2.

The computer program will also demonstrate that, when m or n is zero, the locus of P lies on an axis.

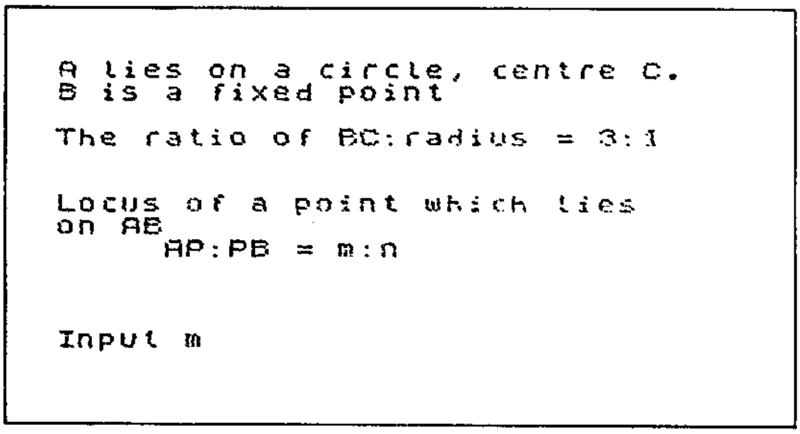
Option C

Figure 3.38 shows the display which results from choosing option C (see figure 3.1). Error messages occur if l is negative or r is not positive. Figure 3.39 shows the display when l=3 and r=1; m and n are now required to be input. An error message occurs if both m and n are zero. Ratios are adjusted so that, if m or n is negative, the numerically larger is made positive. If both m and n are negative the ratio is expressed as the equivalent ratio of positive numbers.



```
A lies on a circle, centre C.
B is a fixed point

The ratio of BC:radius = l:r





Input l
```

Fig. 3.38



```
A lies on a circle, centre C.
B is a fixed point

The ratio of BC:radius = 3:1

Locus of a point which lies
on AB
     AP:PB = m:n


Input m
```

Fig. 3.39

This section was also developed from an investigation in the S.M.P. course. A rotating wheel has a piece of elastic attached to its rim at point A, the other end of the elastic is fixed at point B. The middle of the elastic is marked and the children investigate the locus of this midpoint by drawing several positions of the wheel. With the 4th year group I followed the procedure of the previous lessons, the pupils producing a diagram to find the locus, which is a circle, followed by the computer produced diagrams to investigate the results for different positions of P on AB. Before using the computer I asked the class why the locus was a circle, to which one pupil stated that the original circle had been enlarged with scale factor $\frac{1}{2}$. Most of the pupils realised that this was so because the way that they had constructed the locus was as they would construct the enlargement. It was suggested that if PA:PB = 2:3 then the scale factor woulb be 2/3, but was found to be 3/5. Figures 3.40–3.43 show this.

By holding down the 'm' key for figure 3.41, the drawing of the locus was halted to enable the pupils to see that BP:BA = 3:5, giving an enlargement, scale factor 3/5, centre of enlargement B.

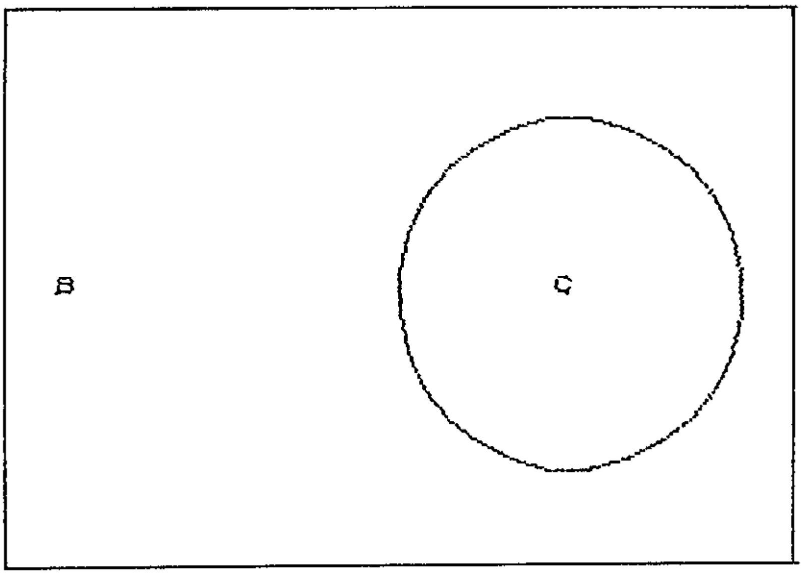


Fig. 3.40
B is the fixed point.
The circle, centre C, represents the wheel.

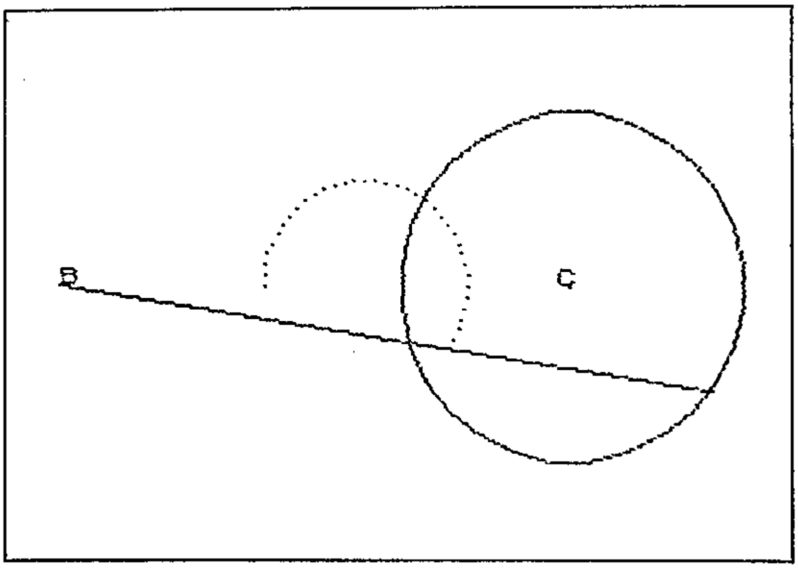


Fig. 3.41
An intermediate stage in demonstrating the locus.

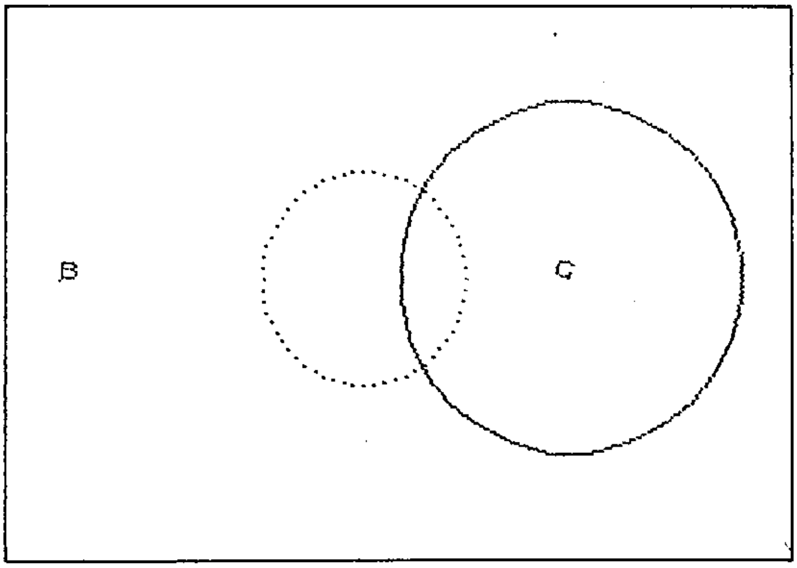


Fig. 3.42
The completed locus.

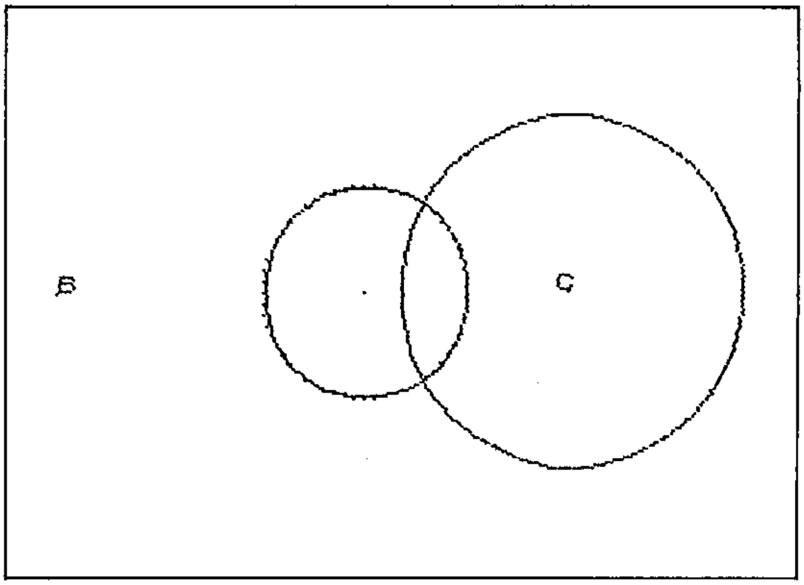


Fig. 3.43
The final stage of the demonstration plots the centre of the locus, and draws the circle.

Generalising this finding produced the result that if AP:PB = m:n the circle is enlarged with scale factor n/(m+n). Negative values of m and n were discussed (i.e. P lies outside AB) and some values were chosen to test whether our result was true in these cases. Figure 3.44 shows that m=-1 and n=2 gives an enlargement scale factor 2, and figure 3.45 shows that m=3 and n=)2 gives an enlargement scale factor -2. Both results satisfy n/(m+n) for the scale factor.

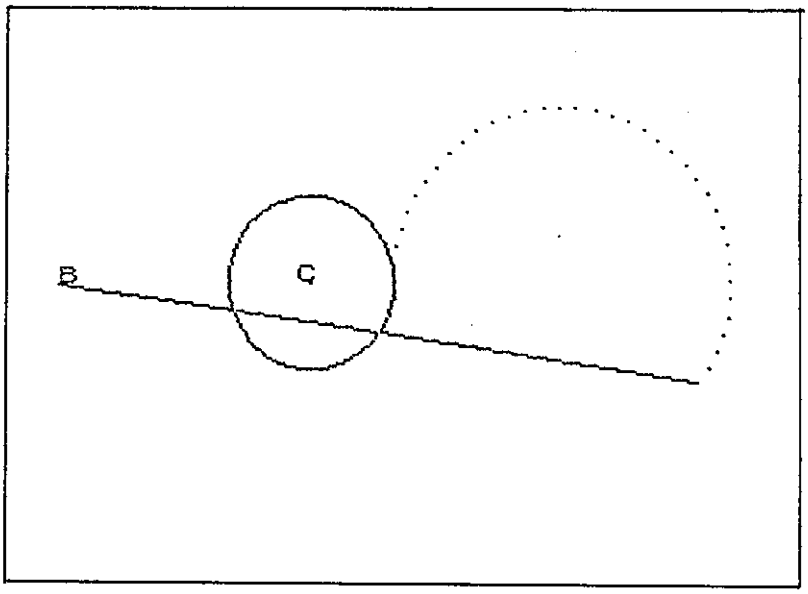


Fig 3.44
An intermediate stage where PA:PB = -1:2.

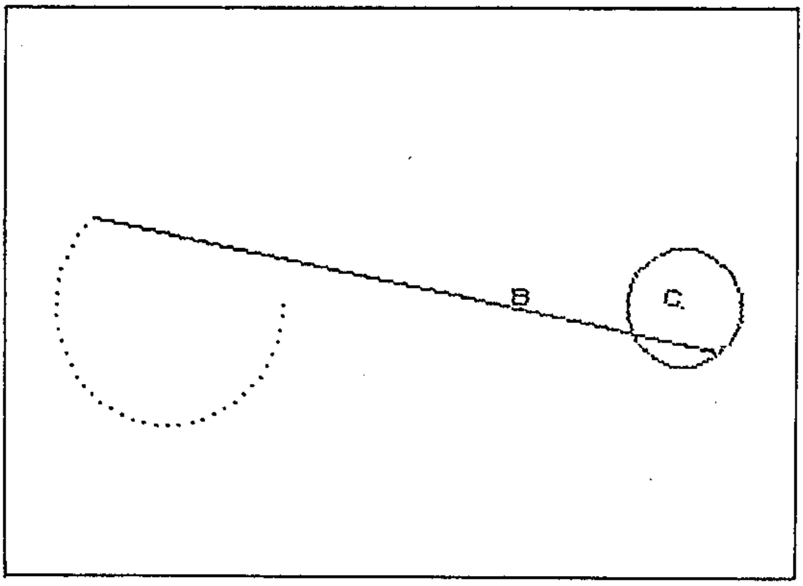


Fig. 3.45
An intermediate
stage where
PA:PB = 3:-2.

When asked if having B inside the circle would affect the result, most pupils thought that the locus would still be a circle. The computer demonstration confirmed this and also that the scale factor for the enlargement was n/(m+n). Figures 3.46-3.49 show various values which result in B inside the original circle.

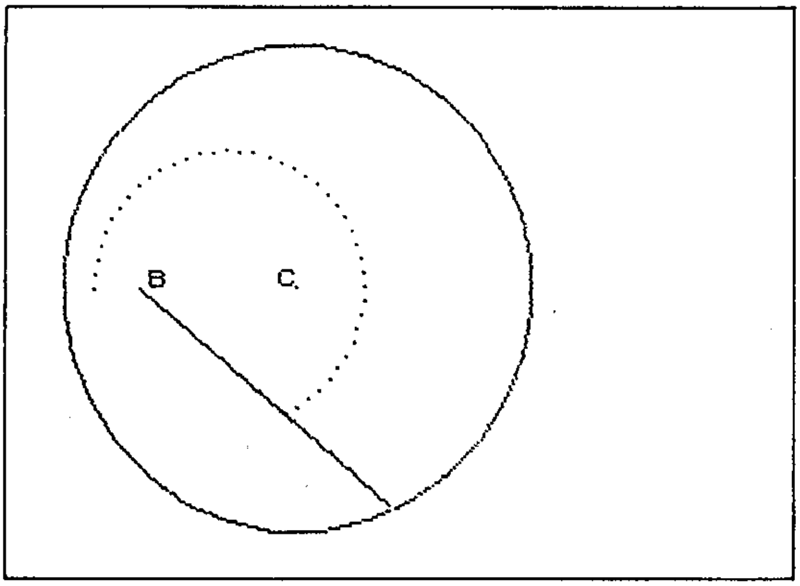


Fig. 3.46
1:r = 2:3,
m:n = 3:4,
enlargement
scale factor
4/7.

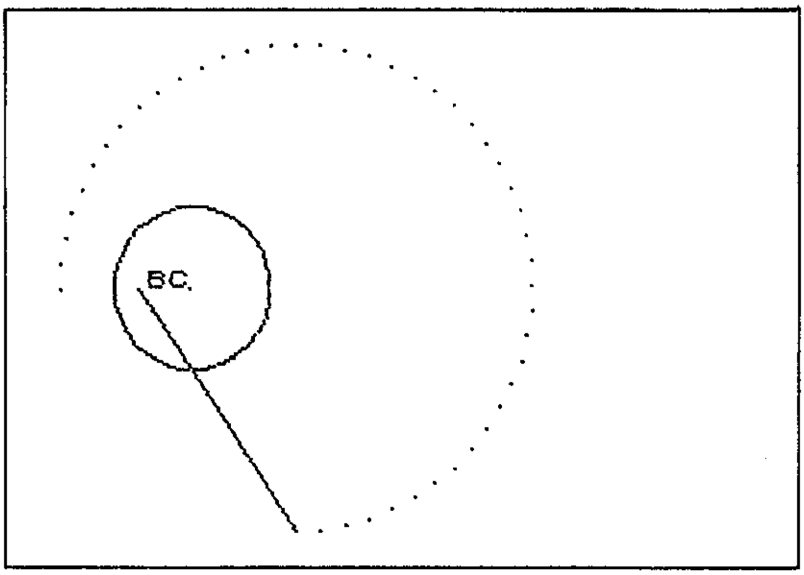


Fig. 3.47
1:r = 2:3,
m:n = -2:3,
enlargement
scale factor
3.

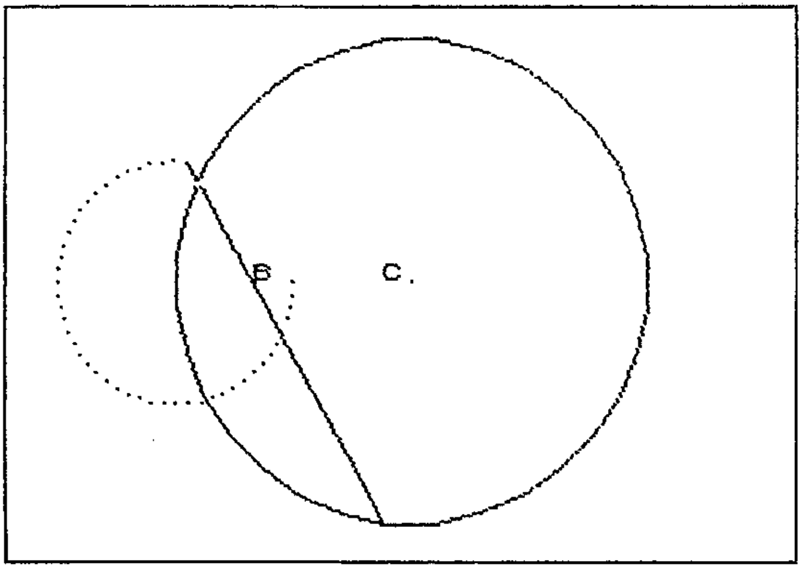


Fig. 3.48
1:r = 2:3,
m:n = 3:-1,
enlargement
scale factor
-1/2.

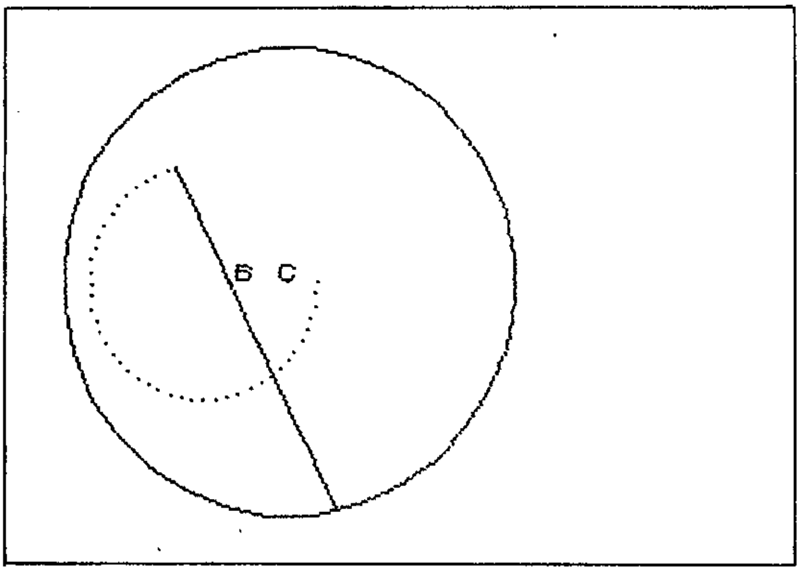
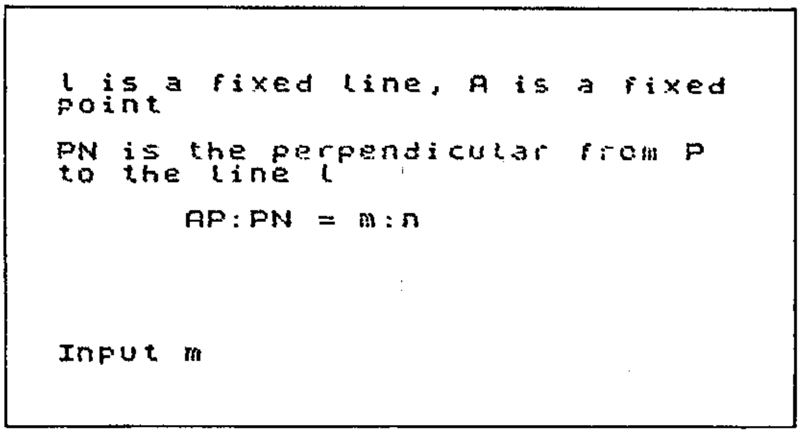


Fig. 3.49
1:r = 1:4,
m:n = 3:-1,
enlargement
scale factor
-1/2.

Option D

When the 'd' key is pressed at the stage shown in figure 3.1, the following display results.



```
l is a fixed line, A is a fixed
point

PN is the perpendicular from P
to the line l

      AP:PN = m:n



Input m
```

Fig. 3.50

The loci which result from the various values of m and n are (i) a parabola if m=n, (ii) an ellipse if $m<n$, (iii) a hyperbola if $m>n$.

I decided to produce this section of the program to use with the 6th form group as an introduction to the conic sections. In teaching the topic in the past, I have felt that pupils are often unable to visualise how the curve changes with different eccentricities. By using the computer to show the locus being drawn point by point, the relation between the curve and its eccentricity was seen more readily than from a text book diagram.

The first values which I input were m=1 and n=1, which led to a parabola. Apart from saying that the locus passed midway between the fixed point and line, no-one suggested what shape would result.

Figure 3.51 shows the parabola being plotted, with the construction lines clearly showing PA = PN.

Figure 3.52 shows the completed locus.

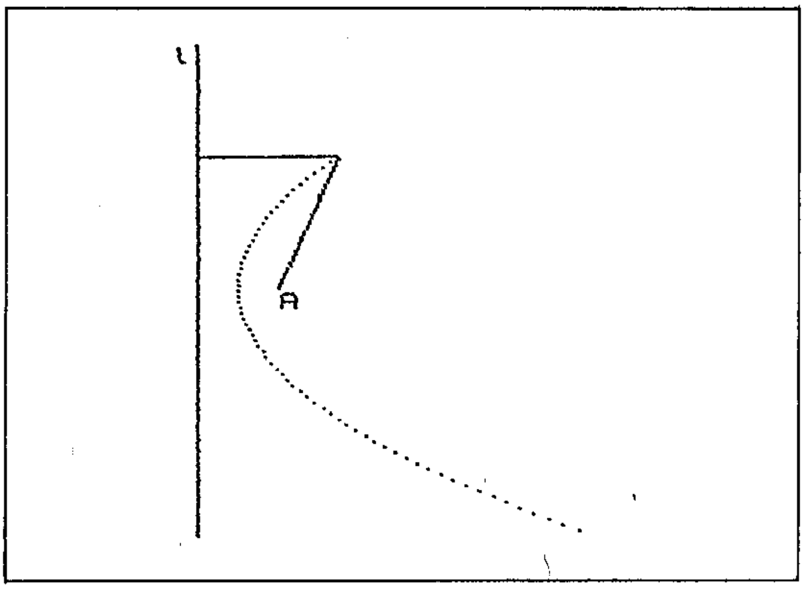


Fig. 3.51

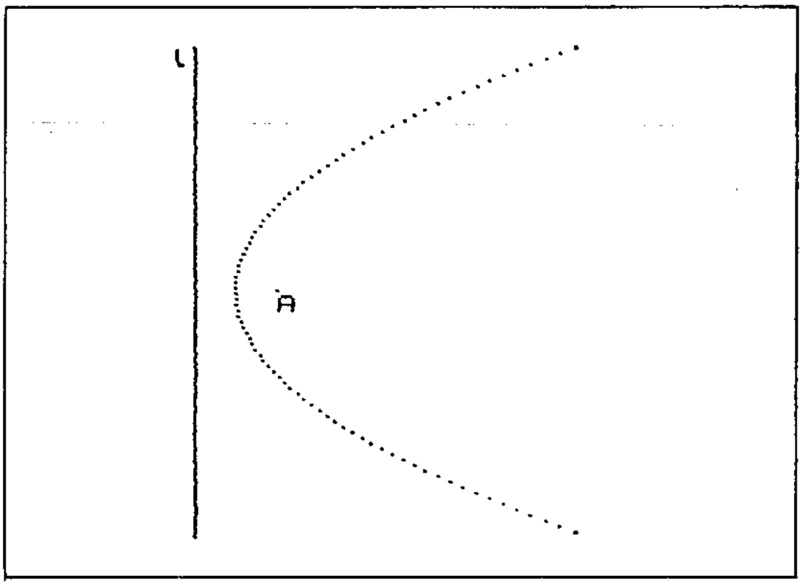


Fig. 3.52

At this stage I introduced the concepts of focus, directrix and eccentricity.

Figure 3.53 shows the locus produced when m:n = 3:5, which gives an ellipse. Several values were input to give an eccentricity which was less than 1, from which it could be seen that, as m/n approached 0, the ellipse approached a circle, and as m/n approached 1 the ellipse became longer. The pupils quickly realised that the ellipse has two foci and so it was shown that, by letting $m/n \rightarrow 1$, a parabola may be compared to an ellipse with its second focus at infinity.

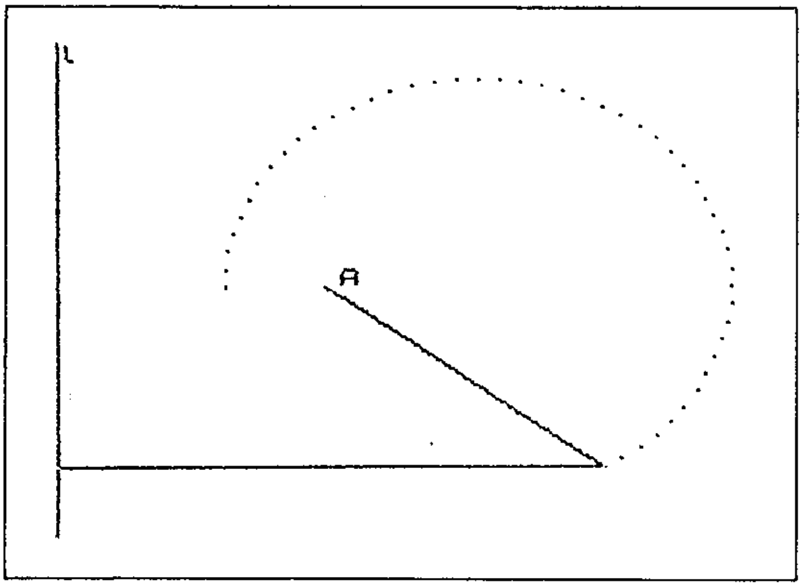


Fig. 3.53
A stage in the plotting of the locus when m:n = 3:5.

Figures 3.54-3.56 show the locus when m=7 and n=6. The hyperbola is drawn in two stages, the pause after the first part has been drawn allowing time for discussion. At first the pupils thought that the curve was a parabola, but on further consideration they realised that the parabola approached two parallel lines, but the hyperbola approached two intersecting lines.

Pressing the 'a' key after the hyperbola has been plotted draws the asymptotes as in figure 3.57.

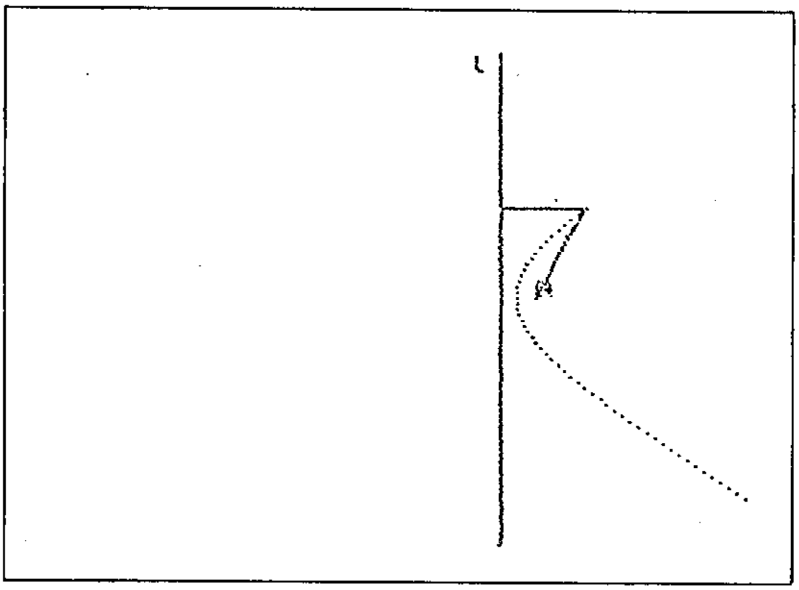


Fig. 3.54
PA:PN = 7:6.

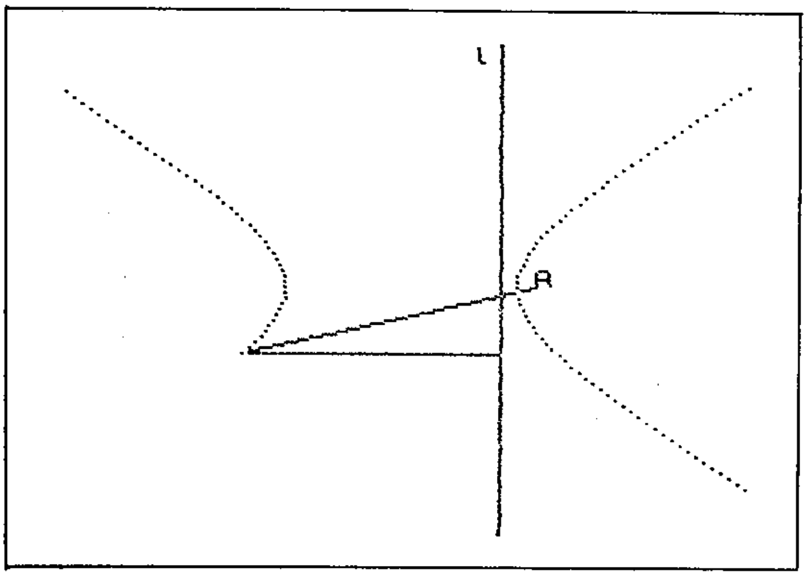


Fig. 3.55
The second
part of the
hyperbola.

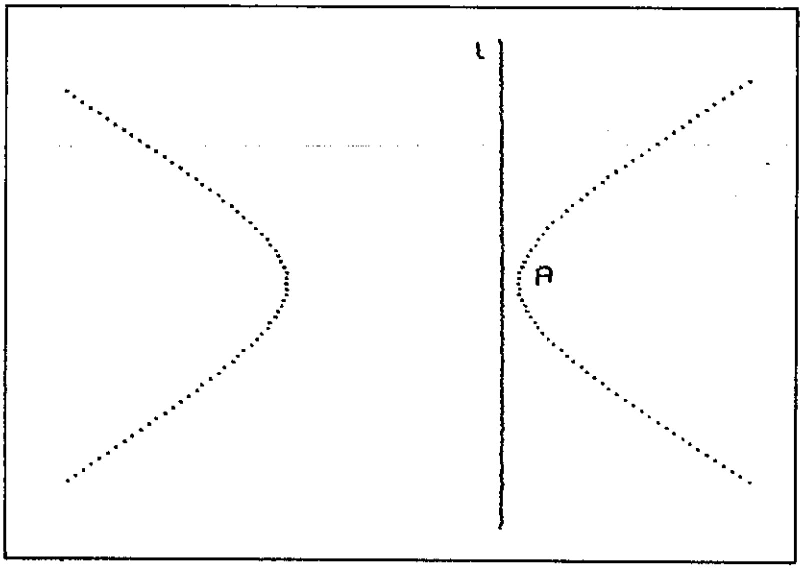


Fig. 3.56
The complete
hyperbola.

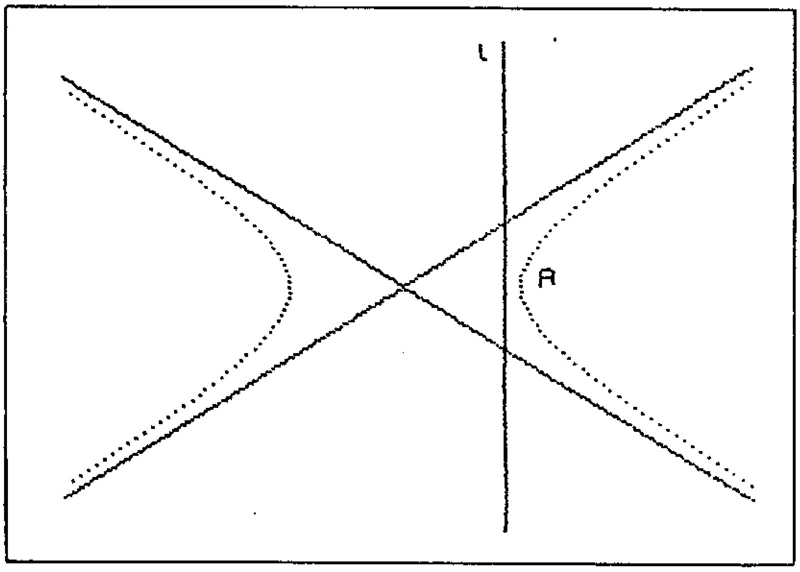


Fig. 3.57
The asymptotes
have been
drawn.

As before, different values of m and n were input to see how these affected the hyperbola. As m → n the asymptotes became less steep, showing that the hyperbola became nearer to the shape of the parabola. As m/n became larger, the asymptotes became steeper.

Figure 3.58 shows the completed hyperbola and its asymptotes when m/n = 2.

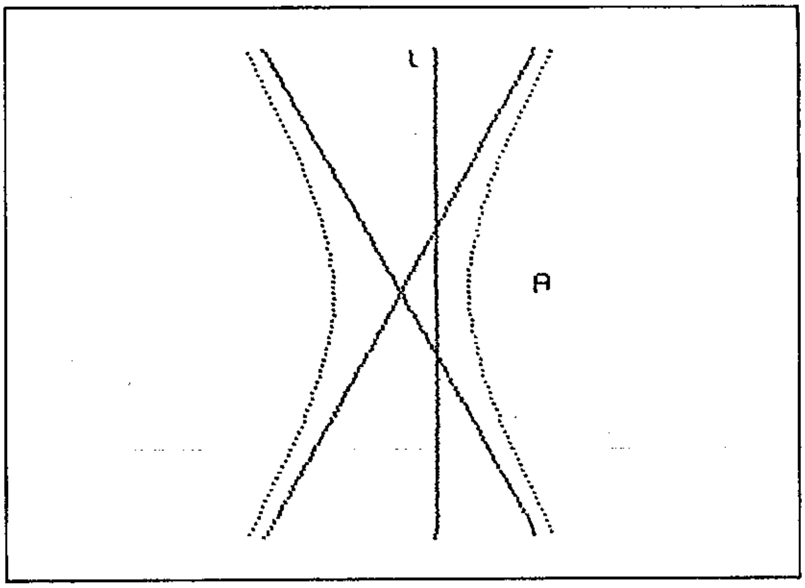
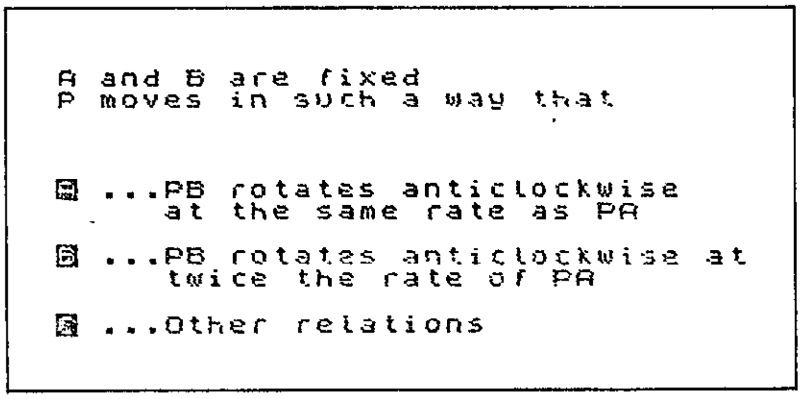


Fig. 3.58

Option E

Pressing the 'e' key at the stage shown in figure 3.1 produces the display of figure 3.59.



```
A and B are fixed
P moves in such a way that


▣ ...PB rotates anticlockwise
      at the same rate as PA

▣ ...PB rotates anticlockwise at
      twice the rate of PA

▣ ...Other relations
```

This section was written to be used in conjunction with an investigation in the S.M.P. 'O' level course which concerns two rotating double-ended searchlights at A and B. Different relations between the angles of rotation give various loci for the point of intersection of the searchlight beams. The pupils are supplied with several copies of the diagram in figure 3.60, and plot the intersection of the required lines.

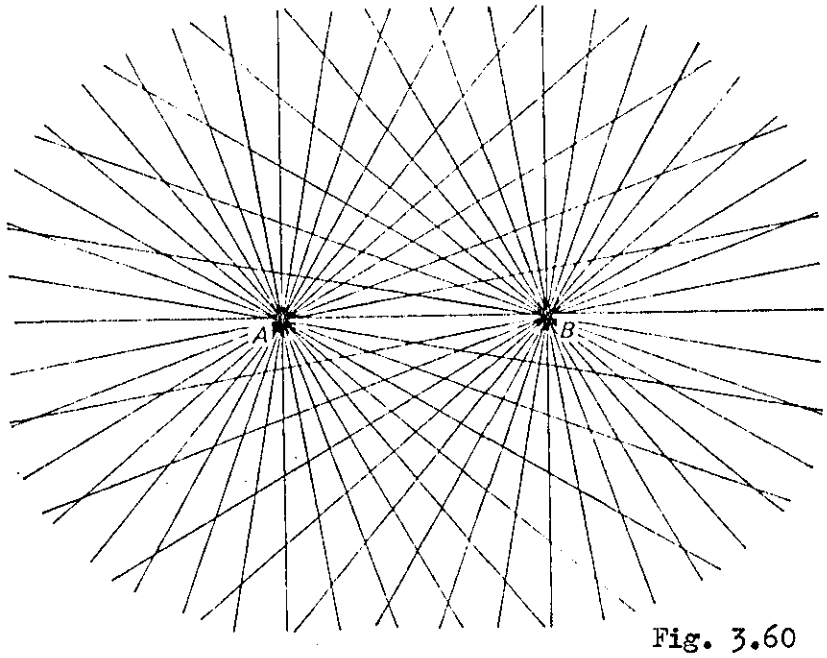


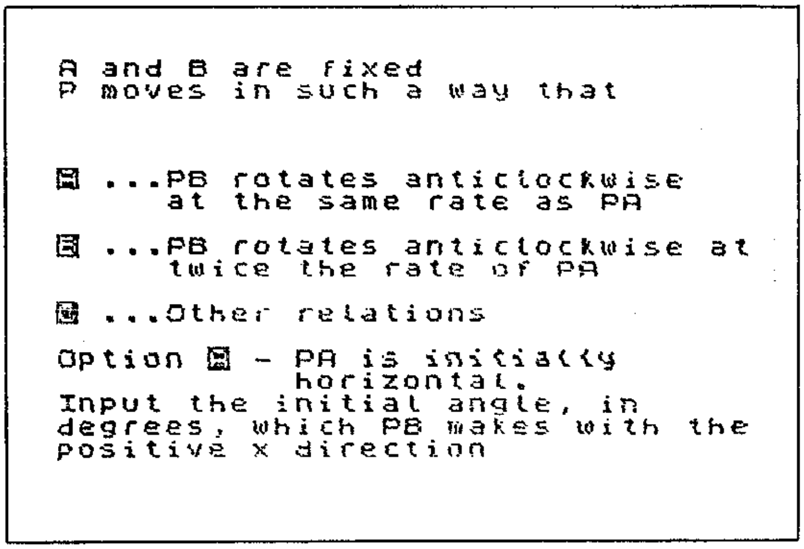
Fig. 3.60

I have found from experience that some pupils are confused because of the number of lines in the diagram. Using the computer helped to make the situation clearer.

This section has three options - 'a' where the two searchlights rotate at the same rate, 'b' where one searchlight rotates at twice the rate of the other, 'c' which allows the user to define the relation between the searchlights.

(a) When the 'a' key is pressed figure 3.61 results, which requires the initial angle which the searchlight through B makes with the horizontal to be input.

Figure 3.62 shows the initial positions of the searchlights when 60 has been input. Part of the letter B has been erased because of the 'exclusive or' print command.



```
A and B are fixed
P moves in such a way that


▣ ...PB rotates anticlockwise
     at the same rate as PA

▣ ...PB rotates anticlockwise at
     twice the rate of PA

▣ ...Other relations

Option ▣ - PA is initially
           horizontal.
Input the initial angle, in
degrees, which PB makes with the
positive x direction
```

Fig. 3.61

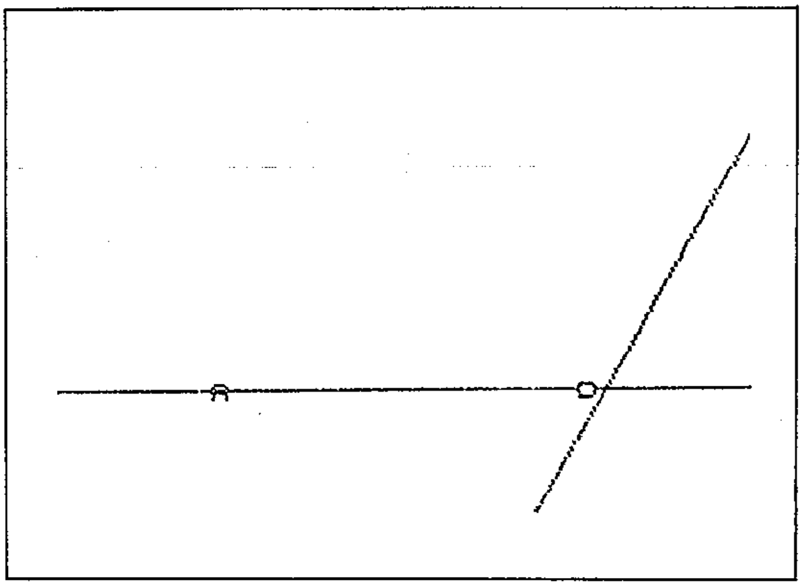


Fig. 3.62

At this stage I asked the pupils to draw the line through B in red to mark the initial position of this searchlight, and then demonstrated the locus on the computer. By using the 'm' key, each position was held so that the pupils could plot the points on their diagrams and also discuss what was happening. Figure 3.63 shows one position of P.

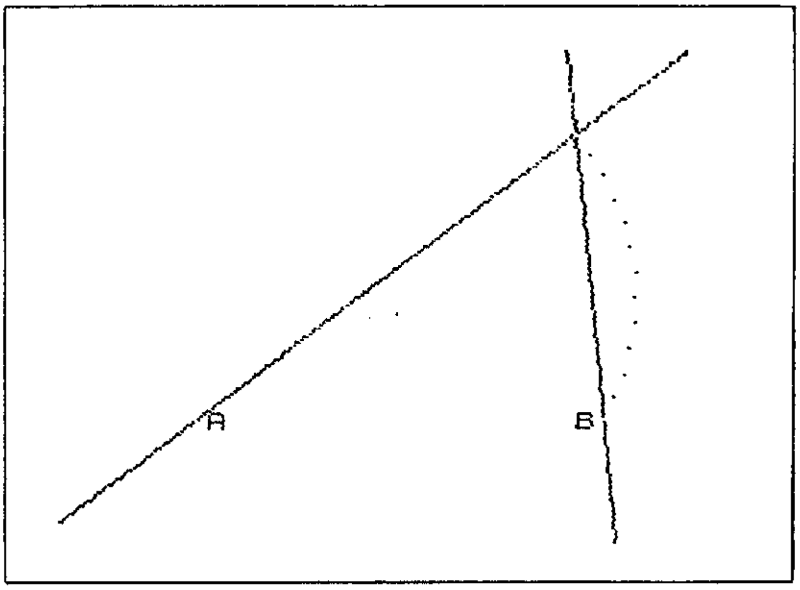


Fig. 3.63

As expected from having previously taught this topic, a number of pupils were unsure how to continue from where the line through B was horizontal. I had halted the locus plotting at this position so that the pupils could discuss how they thought the locus would continue. Some thought that the locus would be symmetrical about AB, 'like a figure 8', but on showing the next position (see figure 3.64) the pupils were able to complete the circle (see figure 3.65).

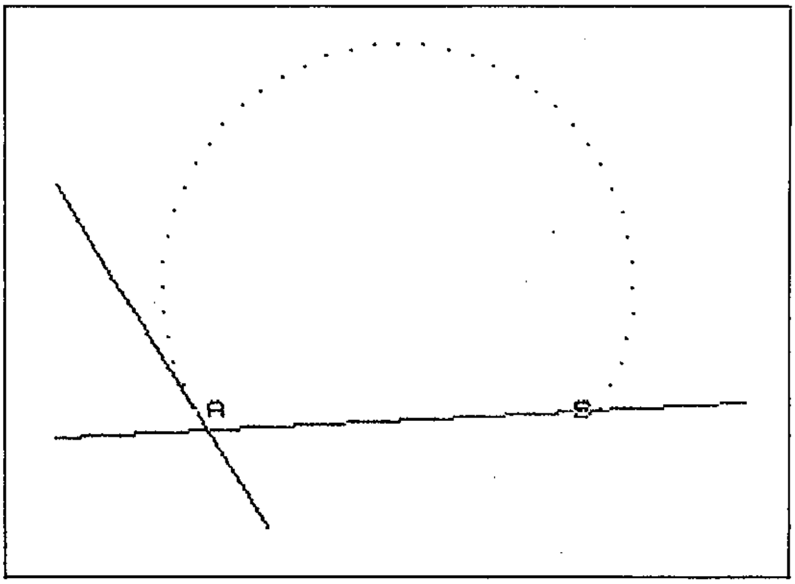


Fig. 3.64
The next
position
plotted after
PB becomes
horizontal.

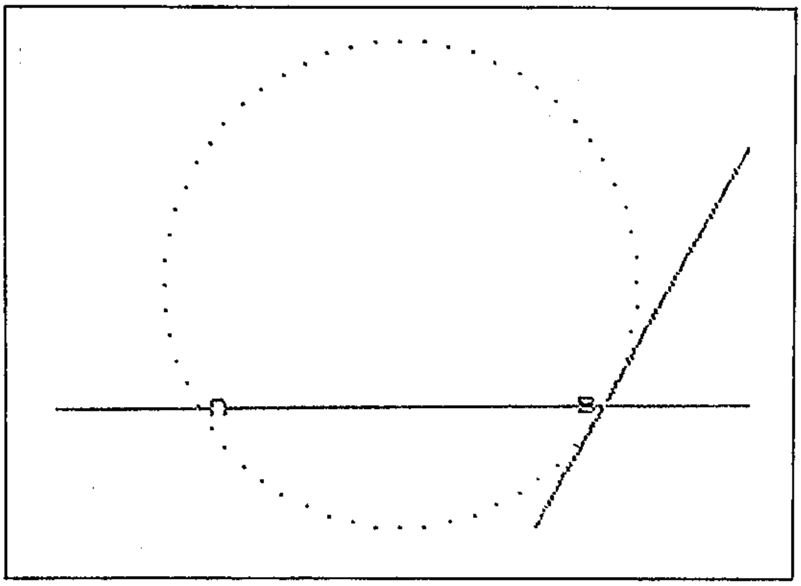


Fig. 3.65
The final positions of the lines are as the initial positions.

Having found that the locus was a circle, I pressed the 'r' key, which repeated the demonstration. From figure 3.63, the pupils were able to see that, because the angle subtended from AB was 60°, the locus from B to A was an arc of a circle. From figures 3.63 and 3.64, where angle APB = 120°, the circle is completed because opposite angles of a cyclic quadrilateral are supplementary.

Figure 3.66 shows an intermediate stage of drawing the locus when the initial angle which PB makes with the horizontal is 120°.

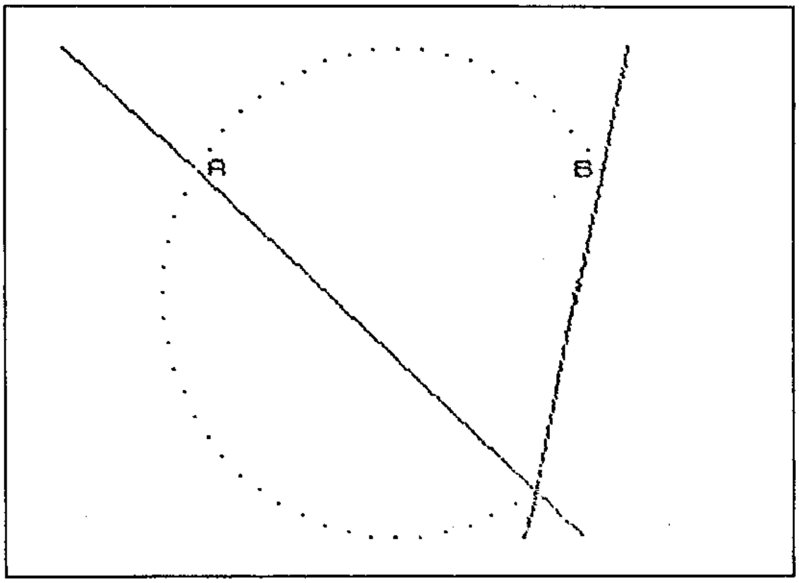


Fig. 3.66

(b) Pressing the 'b' key results in the locus where PB rotates at twice the angle of PA. Initially A and B are plotted with both searchlight beams horizontal.

I used this demonstration in a similar way to that outlined above, advancing in steps to enable the pupils to mark the positions of P on their diagrams. Figures 3.67 and 3.68 show intermediate stages, clearly indicating that the locus is a circle centre B, radius AB. This was easily proved by joining A through B - if angle PAB = k°, then the exterior angle PBA = $2k^\circ$, making the triangle PBA isosceles, and PB = AB.

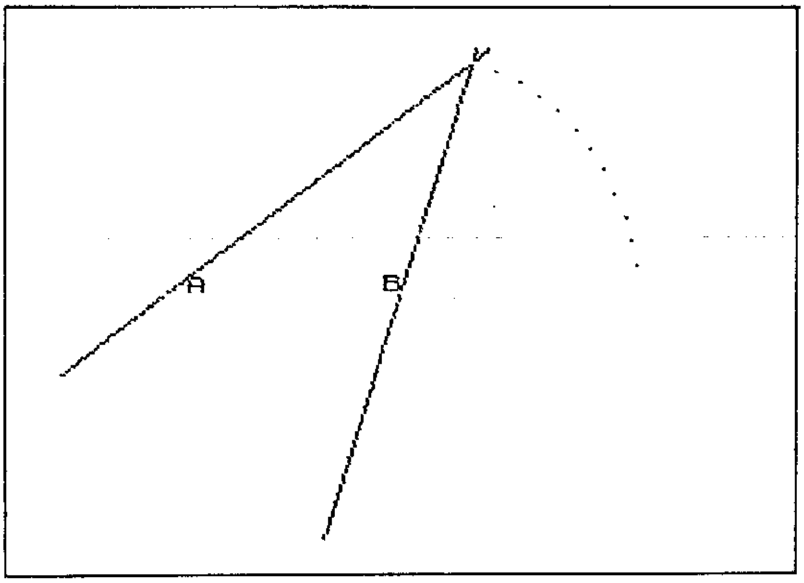


Fig. 3.67

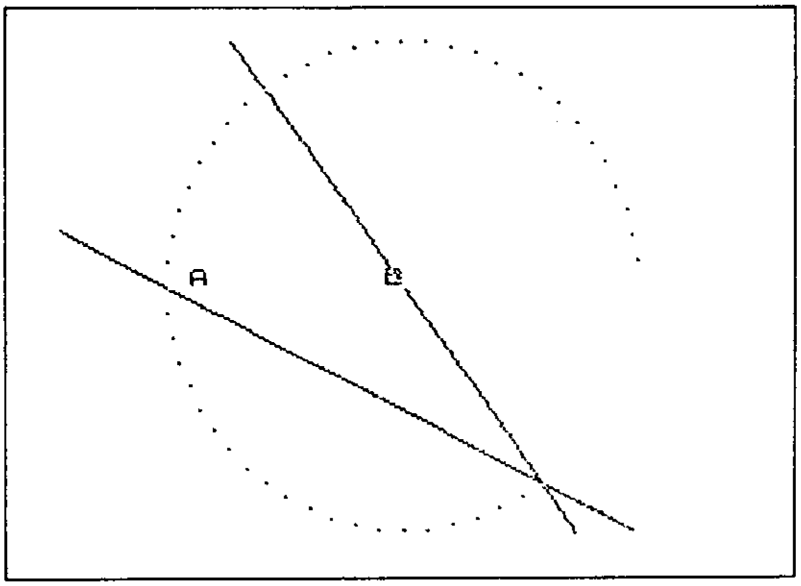
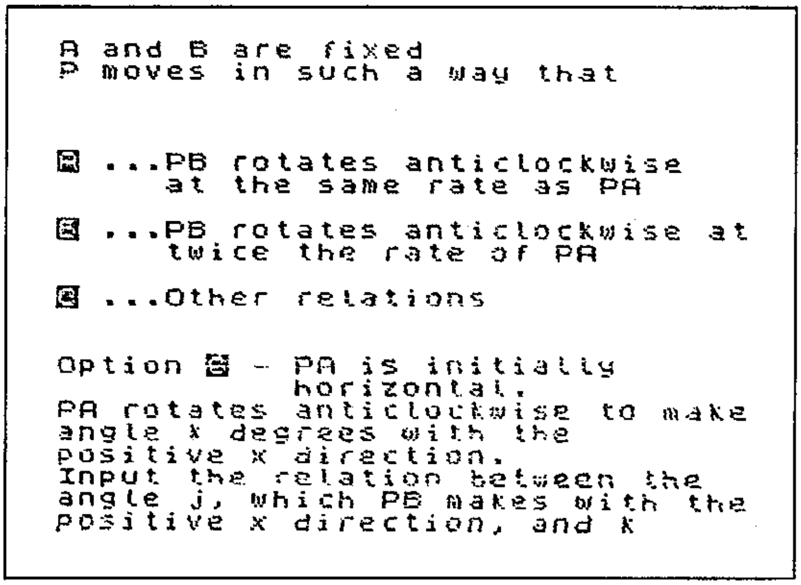


Fig. 3.68

(c) Pressing the 'c' key results in the message displayed in figure 3.69. Printed at the bottom of the screen is j = "L", which indicates that a string is required to be input. The string is evaluated in the program to find j, the angle which PB makes with AB.



```
A and B are fixed
P moves in such a way that


a ...PB rotates anticlockwise
     at the same rate as PA

b ...PB rotates anticlockwise at
     twice the rate of PA

c ...Other relations


Option c - PA is initially
           horizontal.
PA rotates anticlockwise to make
angle k degrees with the
positive x direction.
Input the relation between the
angle j, which PB makes with the
positive x direction, and k
```

Fig. 3.69

Since a circle could be predicted for options 'a' and 'b' above the circles were drawn to fill the screen vertically, with the positions of A and B adjusted accordingly. Because the locus in this section cannot be predicted, A and B have been plotted 30 pixels apart, in the centre of the screen. This was found to be suitable for the relations which were considered. Unlike the previous two loci, the plotting is in a continuous loop. Holding down the 'm' key halts the process with the two lines drawn, the 'n' key halts the process without the lines. Pressing the 'x' key will exit from the loop and return to the initial option display.

The S.M.P. text suggests other relations which may be considered but many pupils find these difficult to plot. The remainder of the loci which I shall illustrate were therefore shown on the computer without the pupils reproducing them.

The first relation which was considered was j = 120 - k (see figures 3.70-3.73). I halted this locus at the stage shown in figure 3.70 to ask the class what happens to the curve. Most could see that the curve approached a straight line, and this was a way of introducing the idea of an asymptote which they could easily understand (although I did not use the word 'asymptote'). Figure 3.71 shows that the curve continues from the other direction of the asymptote. At the stage shown in figure 3.72, the pupils predicted that the locus would meet the first point plotted. Figure 3.73 shows the completed locus by holding down the 'n' key. Pressing the 'x' key at this stage returns to the initial option display.

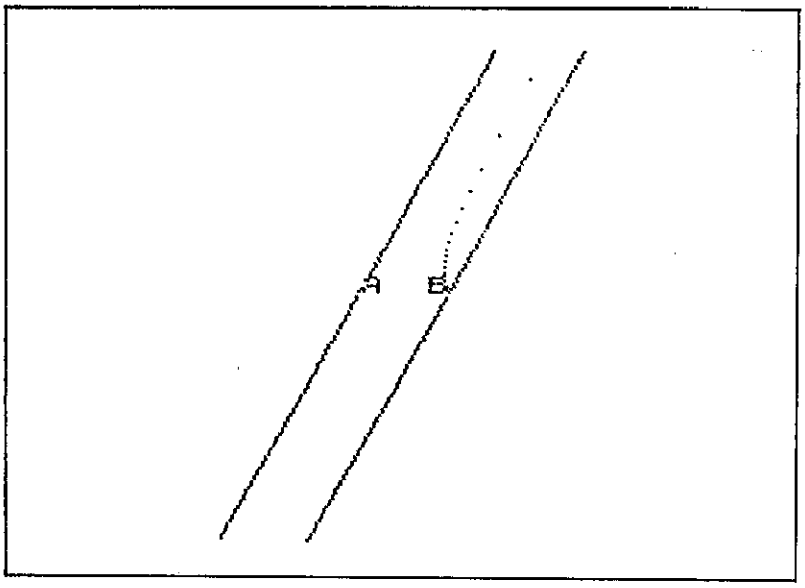


Fig. 3.70

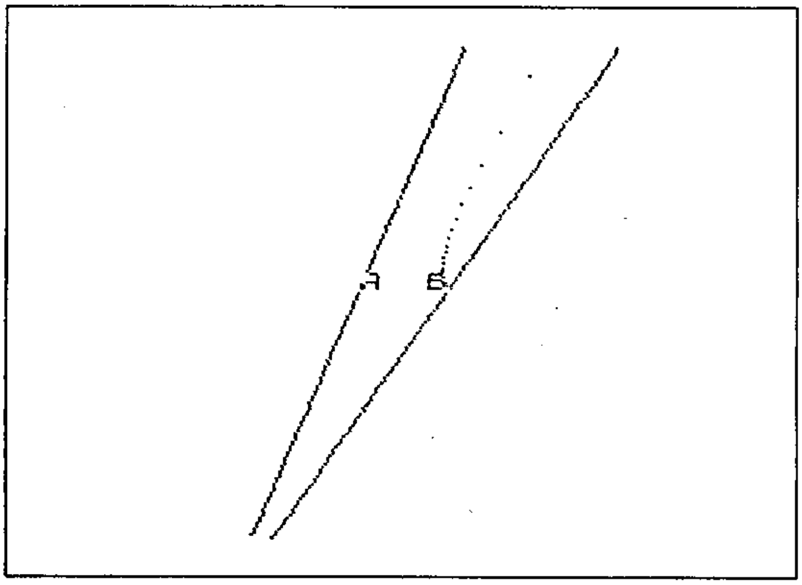


Fig. 3.71

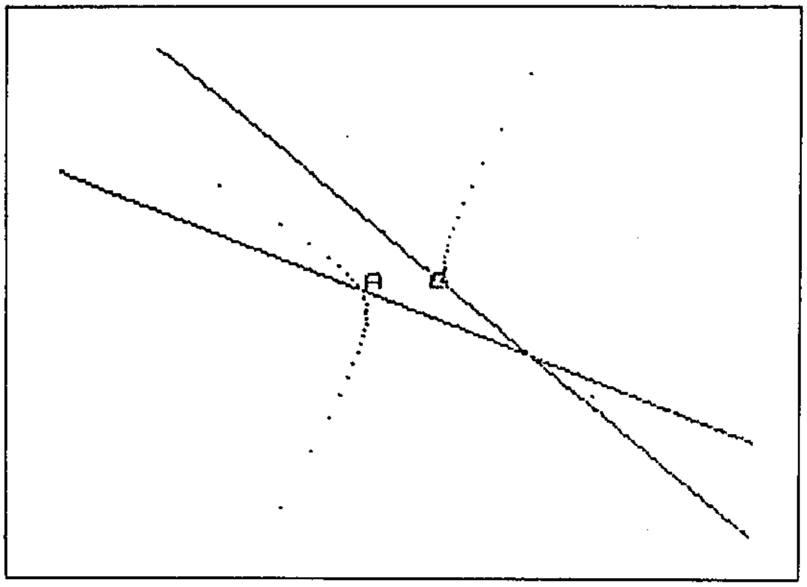


Fig. 3.72

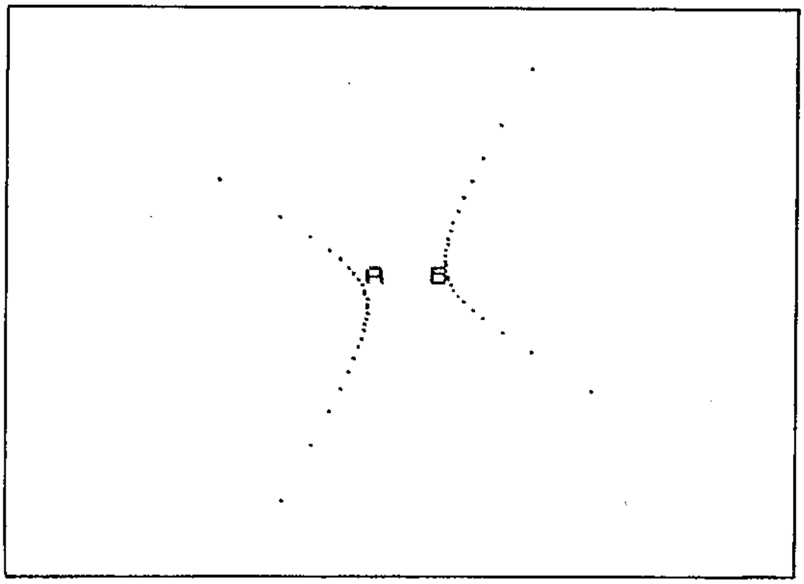


Fig. 3.73
The completed
locus is a
rectangular
hyperbola.

The pupils were keen to try various other relations, some of which are shown below. Figure 3.74 shows another rectangular hyperbola from the relation j = 90 - k. It can be demonstrated that any relation of the form j = θ - k will result in a rectangular hyperbola, unless θ = 180 when the mediator of AB is produced (see figure 3.75).

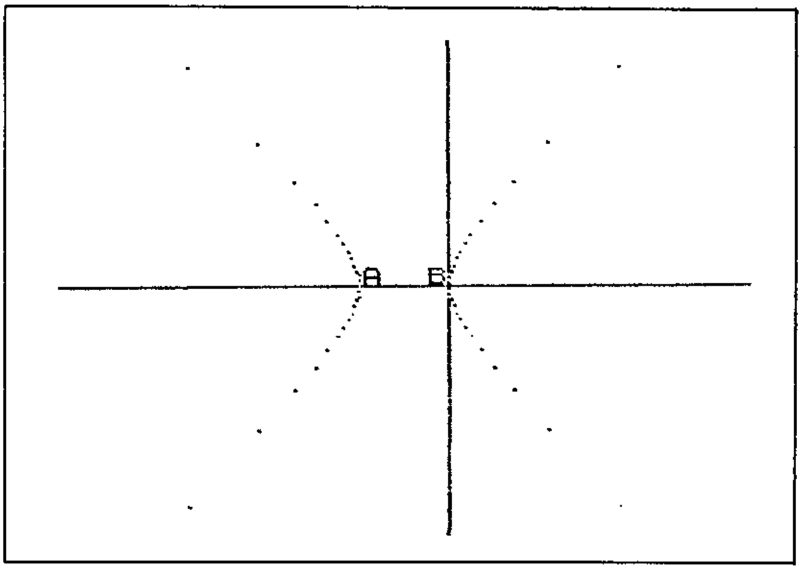


Fig. 3.74
The complete locus for the relation $j = 90 - k$, showing the final position of the searchlights.

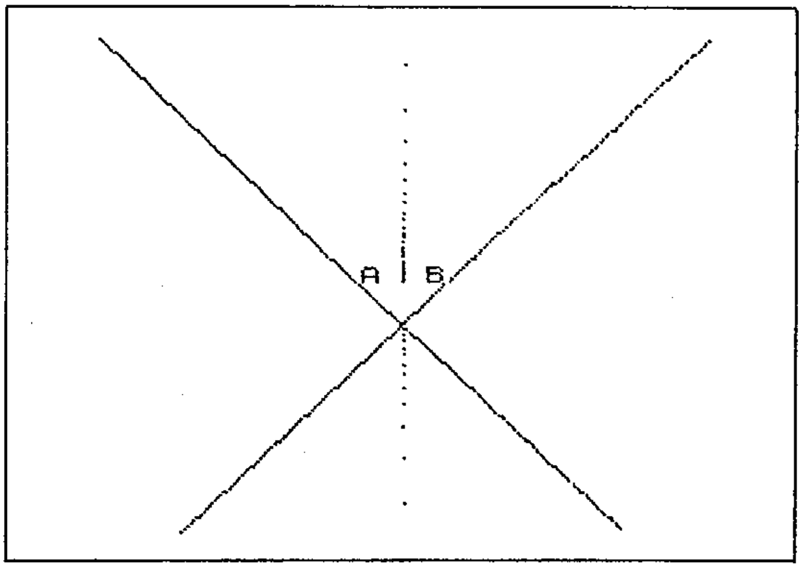


Fig. 3.75
An intermediate stage in plotting the locus for the relation $j = 180 - k$.

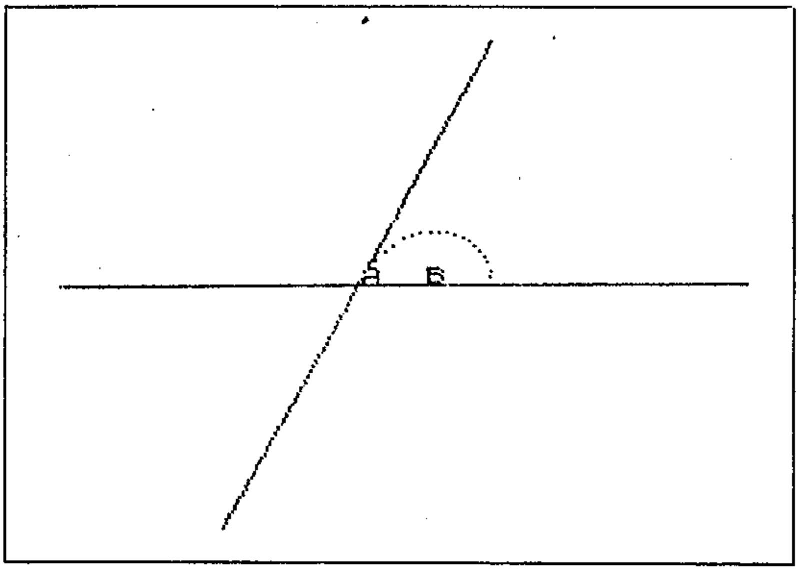


Fig. 3.76
A stage in the plotting of the relation $j = 3k$ (input as 3*k).

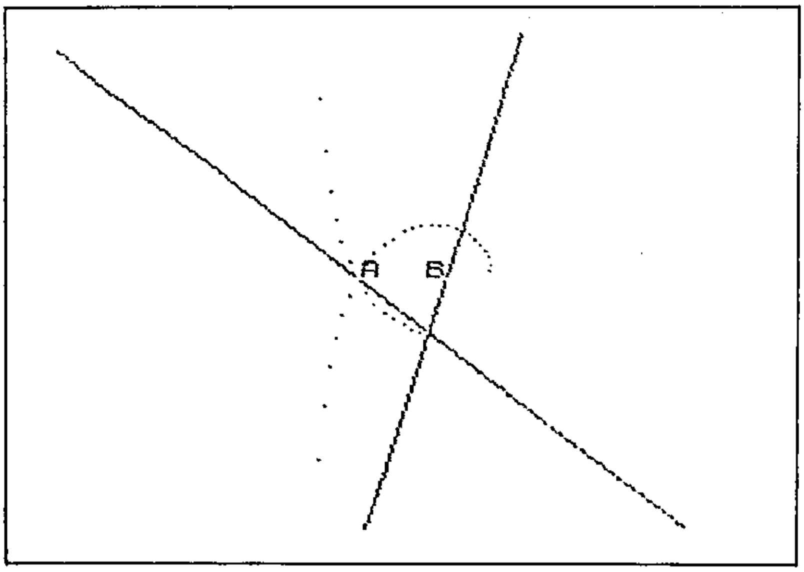


Fig. 3.77
A further stage in plotting the relation
j = 3k.

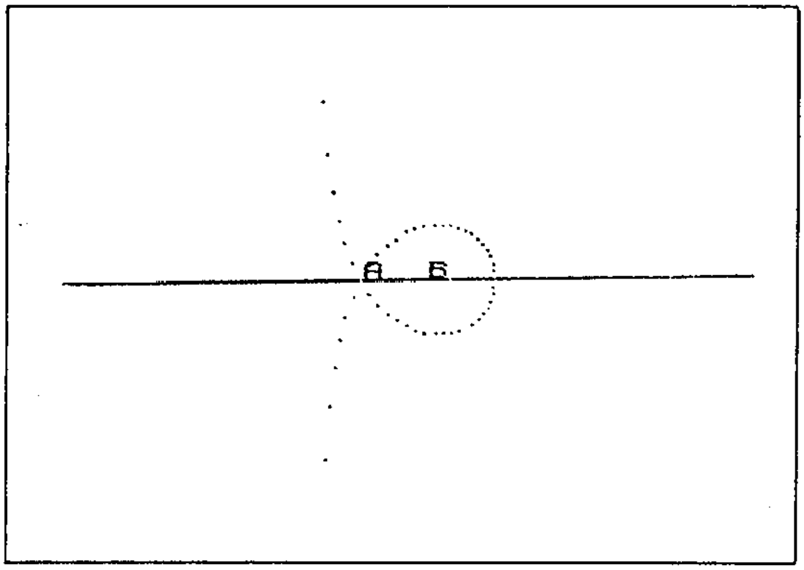


Fig. 3.78
The relation j = 3k, showing the searchlights returned to their initial positions.

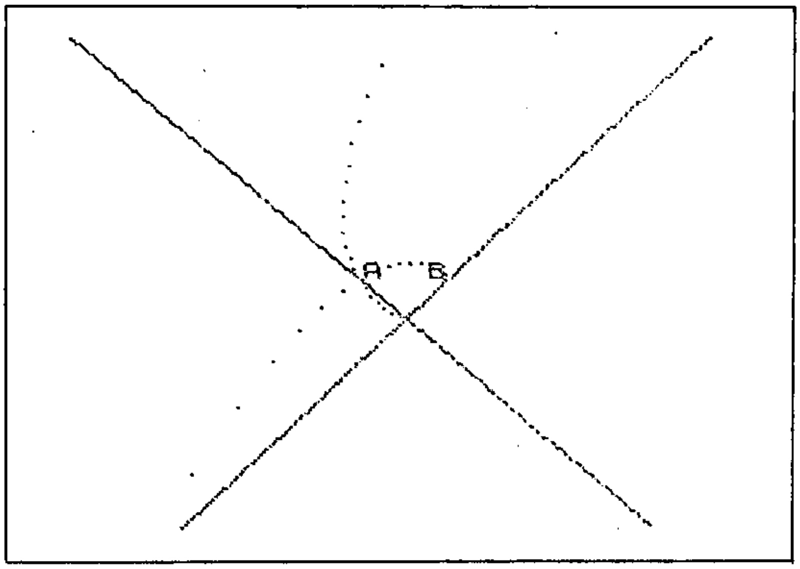


Fig. 3.79
An intermediate stage of the locus when
j = 120 + 2k.

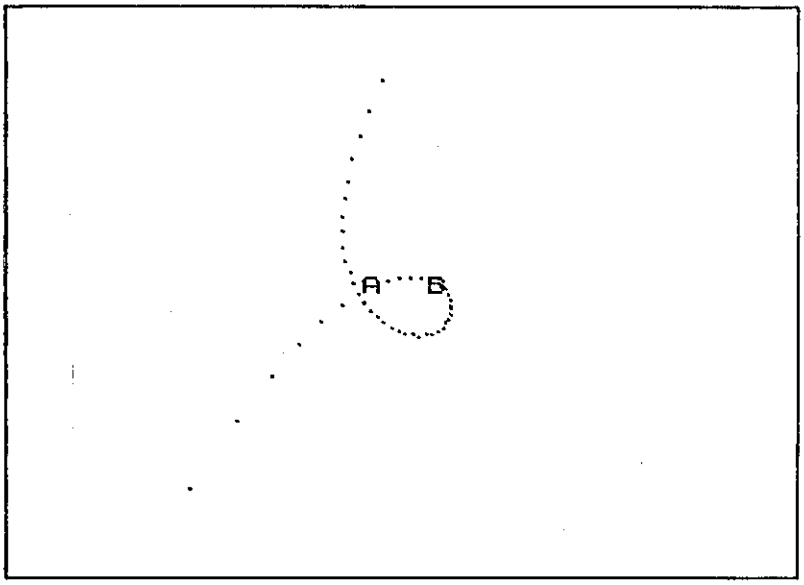


Fig. 3.80
The completed locus for the relation $j = 120 + 2k$.

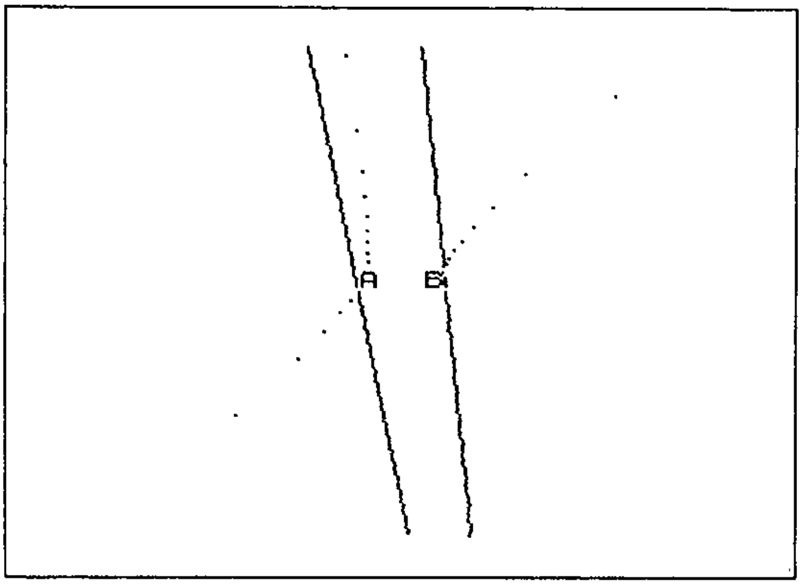


Fig. 3.81
A stage in plotting the relation $j = 120 - 2k$.

I halted the demonstration as in figure 3.81 to ask the pupils how they thought that it would continue. Most thought that the branch about to be drawn would pass through B, but this curve had three asymptotes, as can be seen from figure 3.82.

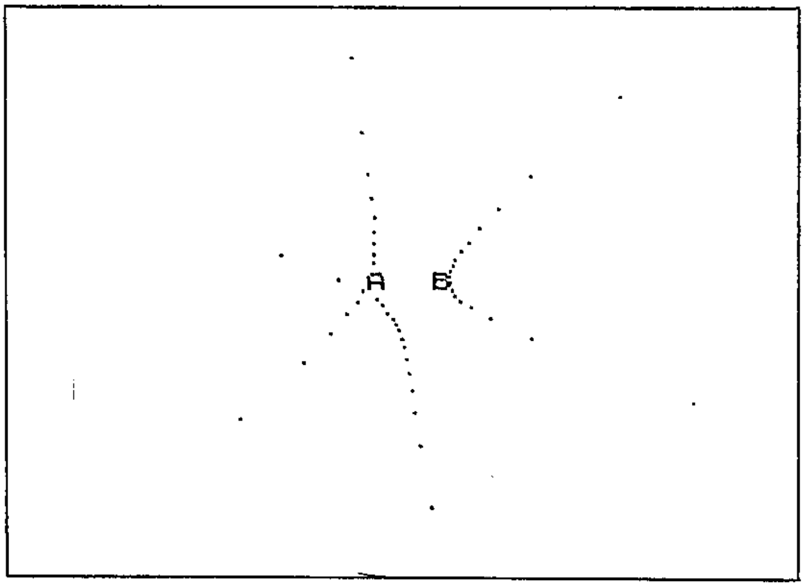


Fig. 3.82
The completed locus for the relation $j = 120 - 2k$.

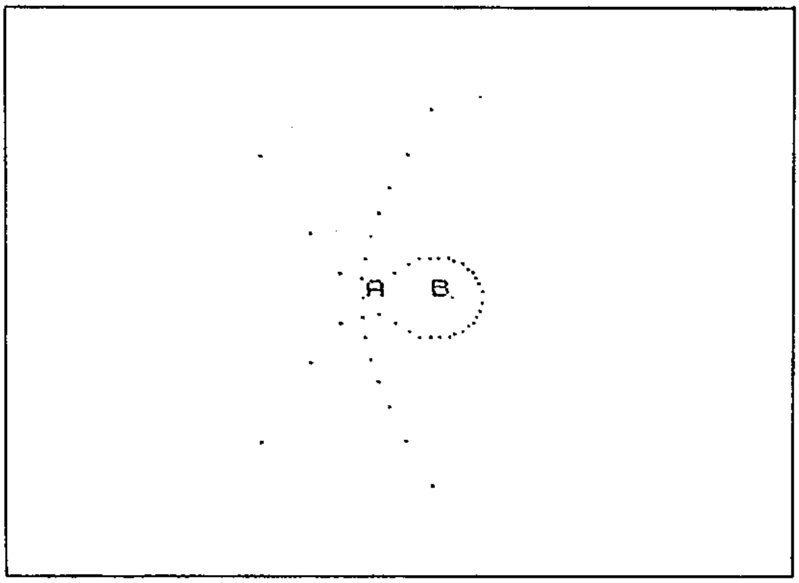


Fig. 3.83
The completed locus for the relation $j = 4k$.

Conclusion

I found that the idea of a locus as the path which a point traces out when moving under certain conditions was more easily understood by the pupils when demonstrated using the computer, than by methods which involve only drawing. The demonstrations promoted discussion among the pupils and an interest in investigating how different conditions changed the locus. They were keen to experiment and to try to predict the outcomes.

Although I do not expect many pupils to remember which loci resulted from the different conditions, the series of lessons was valuable in showing the pupils that mathematics is not just learning facts and techniques. In addition, as has been shown, other aspects of mathematics were reinforced by considering why certain loci were produced.

At the sixth form level, demonstrating the locus of P dynamically helped the pupils to understand the technique of finding the cartesian equation by writing down the condition which P must satisfy. When teaching the conic sections in a later lesson, I felt that a greater insight had been achieved by investigating the curves using the computer.

Bibliography

HART, K.M. (editor) (1981)
Children's Understanding of MATHEMATICS : 11-16
London : John Murray

JEGER, M. (1966)
Transformation Geometry
London : George Allen & Unwin Limited

LAND, F. (1960)
The Language of Mathematics
London : John Murray

MAXWELL, E.A. (1975)
Geometry by Transformations
Cambrige University Press

VICKERS, S. (1982)
ZX Spectrum BASIC programming
Cambridge : Sinclair Research Limited

School Mathematics Project, Books 1-5, A-H, X, Y, Z
Cambridge University Press

Appendix A - The listing for the program 'transforms'.

```
   1 BORDER 0: PAPER 0: INK 7: C
LS : POKE 23659,0: PRINT AT 22,0
;"Demonstration?  y/n": POKE 236
59,2
```

1 - Sets the screen background to black and writing to white. CLS clears the screen. POKE 23659,0 enables printing on the bottom two lines of the screen (normally used only for the input messages). POKE 23659,2 returns to normal.

```
   2 IF INKEY$="n" THEN LET d$="
n": GO TO 5
   3 IF INKEY$<>"y" THEN GO TO 2
   4 LET d$="y"
   5 IF INKEY$<>"" THEN GO TO 5
```

2 - Sets d$ to "y" or "n" depending on which key is pressed.

3 - The INKEY$ control is used so that this operation is carried out by pressing one key only. No other key will produce an exit from the 2-3-2 loop.

5 - The program continues only when the key is released.

```
   6 CLS : RESTORE : OVER 0
   7 LET mnx=-1: LET mxx=1: LET
mny=-1: LET mxy=1
```

6 - RESTORES DATA statement for the preset object: OVER 0 is the normal printing mode.

7 - Sets minimum values of the axis scales to -1 (mnx and mny), maximum values (mxx and mxy) of the axis scales to 1.

```
  10 DEF FN f(x)=(x+ox)*ss+5
  11 DEF FN g(y)=(y+oy)*ss+5
```

10 - FN f and FN g are used for the PLOT command. ss = the number of pixels per unit of the axes. ox, oy translate the origin from (5,5) on the screen.

```
  12 POKE 23659,0: PRINT AT 22,0
;"Choose range? y/n ": POKE 2365
9,2
  13 IF INKEY$="y" THEN GO SUB 2
80: GO TO 16
  14 IF INKEY$<>"n" THEN GO TO 1
3
```

12 - Option message to choose the ranges of x and y.

Pressing the 'y' key results in the execution of the subroutine 280 to input minimum and maximum values of x and y (mnx, mny, mxx, mxy). The 'n' key is the only other key which allows the continuation of the program.

```
  15 IF INKEY$<>"" THEN GO TO 15
  16 LET minx=mnx: LET maxx=mxx:
 LET miny=mny: LET maxy=mxy
```

15 - The program only continues when the key is released.

16 - The values of minx, maxx, miny and maxy will change during the execution of the program. This line is so that the original values are not lost.

```
  18 LET ss=0: LET ff=0: LET gg=
0: LET hh=0: LET ii=0: GO SUB 30
0
```

18 - ss = number of pixels per unit, ff = number of images on the screen, gg = number of invariant lines of the form y=mx+c on the screen, hh = number of centres of enlargement/rotation, ii = the number of lines of the form x=c: SUB 300 draws axes.

```
  20 REM Input the coordinates o
f the object
  22 POKE 23659,0: PRINT AT 22,0
;"Preset object? y/n ": POKE 236
59,2
  23 IF INKEY$="y" THEN READ c$,
n: LET r$="y": GO TO 30
```

22 - Option message to choose the preset object, or not.
'y' is pressed if the preset object is required. The DATA statement is on line 970. c$ = "y" to indicate that the shape is closed. n = the number of vertices of the object.

```
  24 IF INKEY$<>"n" THEN GO TO 2
3
  25 IF INKEY$<>"" THEN GO TO 25
  26 LET r$="n"
  27 INPUT "Closed curve - y/n "
;c$: IF c$<>"y" AND c$<>"n" THEN
 GO TO 27
  28 INPUT "How many points?  ";
n
  30 IF INKEY$<>"" THEN GO TO 30
  31 DIM m(10): DIM c(10): DIM k
(10): DIM u(10): DIM z(10): DIM
a(n): DIM b(n): DIM v(10,n+1): D
IM w(10,n+1): DIM x(n+1): DIM y(
n+1): DIM t(n): DIM s(n): DIM e(
n+1): DIM f(n+1): DIM p(n+1): DI
M q(n+1)
```

Only 'y' or 'n' keys continue the execution of the program.
25 - The program only continues when the key is released.
26 - r$ = the response (y or n) to the question 'Preset object?'
27 - Requires "y" or "n" to be input in response to the message 'Closed curve - y/n'.
n = the number of vertices of the object.
30 - Program only continues when a key is released.
31 - Sets up arrays to store the images displayed on the screen (maximum 10). m = gradients of invariant lines, c = their y-intercepts. k = values of the x=k invariant lines. (u,z) = the centres of enlargement/rotation. a,b are used to compute and draw the sides of the images. (v,w) = coordinates of the various images. (x,y) = coordinates of the object of the current transformation. s,t are the vector components of the sides of the object. e,f store the final image of the current transformation. p,q - various uses.

```
  35 IF r$="y" THEN GO SUB 950:
GO TO 42
  40 FOR j=1 TO n: INPUT "x coor
d?  ";x(j);" y coord?  ";y(j): N
EXT j
```

35 - SUB 950 inputs the DATA for the preset object.
40 - Inputs the coordinates for the object, if the preset object is not required

```
  42 FOR j=1 TO n
  45 IF x(j)>maxx THEN LET maxx=
INT x(j)+(INT x(j)<>x(j)): LET t
t=1
  46 IF x(j)<minx THEN LET minx=
INT x(j): LET tt=1
  47 IF y(j)<miny THEN LET miny=
INT y(j): LET tt=1
  48 IF y(j)>maxy THEN LET maxy=
INT y(j)+(INT y(j)<>y(j)): LET t
t=1
  49 NEXT j
  50 IF tt=1 THEN GO SUB 300
  55 LET x(n+1)=x(1): LET y(n+1)
=y(1)
  60 FOR j=1 TO n: LET s(j)=x(j+
1)-x(j): LET t(j)=y(j+1)-y(j): N
EXT j
  75 GO SUB 523: GO SUB 730
  90 POKE 23659,0: PRINT AT 22,0
;"Rot, reF, Trans, Enl, sHear,
 Stretch          ": POKE 23659,2
  95 LET g$=""
 100 IF INKEY$="r" THEN GO SUB 1
000: GO TO 220
 110 IF INKEY$="f" THEN GO SUB 2
000: GO TO 220
 120 IF INKEY$="t" THEN GO SUB 3
000: GO TO 220
 130 IF INKEY$="e" THEN GO SUB 4
000: GO TO 220
 140 IF INKEY$="h" THEN GO SUB 5
000: GO TO 220
 145 IF INKEY$="s" THEN GO SUB 7
000: GO TO 220
 150 GO TO 100
 220 IF INKEY$="" THEN GO TO 220
 225 IF INKEY$<>"" THEN GO TO 22
5
```

42 - 49 Each point is tested to see if it lies within the existing ranges of x and y. If not, the range is extended to include (x_j, y_j). (The range is extended to include the next integer.)

If the range has been extended, the flag tt is set to 1 (it was initially set to 0, in SUB 300).

50 - SUB 300 redraws the axes, if necessary.

55 - 60 s and t are the components of the vectors which form the sides of the object.

75 - SUB 523 draws the object, SUB 730 stores the object in memory.

90 - 150 Jumps to the transformation subroutine corresponding to the key which is pressed: r - rotation, f - reflection, t - translation, e - enlargement, h - shear, s - stretch.

No other key will produce a response.

220 - 225 On return from the transformation subroutine, a key needs to be pressed and released to continue execution.

```
 230 POKE 23659,0: PRINT AT 22,0
;"Rep, Sup, sTart, New, Col, Exe
s, Previous image": POKE 23659,2
 235 IF INKEY$="a" THEN LET g$="
a": GO SUB 280: GO SUB 850: GO S
UB 300: GO SUB 530: GO SUB ret:
GO TO 220
```

230 - Prints the message listing the options as explained in the notes on the program.

235 - 'a' response allows the axes scales to be changed (SUB 280). SUB 850 tests if all the images and lines on display are within the new ranges, increasing the range if necessary. SUB 300 draws new axes. SUB 530 draws all images, lines, centres of enlargement or rotation, on the new axes. SUB ret returns and repeats the previous transformation.

```
 240 IF INKEY$="r" THEN LET g$="
r": CLS : GO SUB 490: GO SUB 530
: GO SUB ret: GO TO 220
```

240 - 'r' response repeats the transformation. CLS clears the screen. SUB 490 draws the axes. SUB 530 and SUB ret as above. The value of ret is the current transformation subroutine address. (See note below *.)

```
 250 IF INKEY$="s" THEN LET g$="
s": GO SUB ret: GO SUB 730: GO S
UB 910: GO TO 90
```

250 - 's' response sets the original object as the object for the next transformation. SUB 730 stores the current image in the memory. SUB 910 recalls the original object.

```
 255 IF INKEY$="p" THEN LET g$="
p": GO SUB ret: GO SUB 775: GO T
O 90
 260 IF INKEY$="t" THEN LET g$="
t": OVER 0: GO SUB 800: CLS : GO
 SUB 300: GO TO 42
```

255 - 'p' response sets the previous object as the current object (SUB 775).

260 - 't' executes the program starting with the display of the original object, on the original scales (unless they have been altered by response 'a' above).
SUB 800 recalls the original object.

```
 265 IF INKEY$="f" THEN LET g$="
f": GO SUB ret: GO SUB 730: GO T
O 90
 270 IF INKEY$<>"n" THEN GO TO 2
35
 275 IF INKEY$<>"" THEN GO TO 27
5
 276 GO TO 1
```

265 - 'f' response sets the current image as the object for the next transformation.

270 - 276 If none of the other keys have been pressed, only 'n' will produce a response.
'n' response reruns the program from the beginning.

*Repeating the transformation requires erasing the current image. Doing this using the OVER 1 command resulted in erasing parts of the axes, etc. on occasions. I therefore decided to achieve the desired effect by clearing the screen and redrawing the display as it was prior to the transformation.

```
 280 IF INKEY$<>"" THEN GO TO 28
0
 282 INPUT "min x? ";mnx;" max x
? ";mxx
 285 INPUT "min y? ";mny;" max y
? ";mxy
 290 RETURN
 300 REM axes and scales
 301 LET tt=0
 305 CLS
 310 IF maxx<0 THEN GO TO 370
 320 IF minx<0 THEN GO TO 350
 330 LET sx=INT (250/maxx): LET
minx=0: LET ox=0
 340 GO TO 400
 350 LET sx=INT (250/(maxx-minx)
): LET ox=-minx
 360 GO TO 400
 370 LET sx=INT (-250/minx): LET
 maxx=0: LET ox=-minx
 400 IF maxy<0 THEN GO TO 460
 410 IF miny<0 THEN GO TO 440
 420 LET sy=INT (170/maxy): LET
miny=0: LET oy=0
 430 GO TO 470
 440 LET sy=INT (170/(maxy-miny)
): LET oy=-miny
 450 GO TO 470
 460 LET sy=INT (-170/miny): LET
 maxy=0: LET oy=-miny
 470 IF sx>sy THEN LET ss=sy
 480 IF sx<=sy THEN LET ss=sx
 490 PLOT ox*ss+5,0: DRAW 0,175
 500 PLOT 0,oy*ss+5: DRAW 255,0
 510 FOR j=0 TO maxx-minx: PLOT
5+j*ss,oy*ss+4: NEXT j
 520 FOR j=0 TO maxy-miny: PLOT
ox*ss+4,5+j*ss: NEXT j
 522 RETURN
 523 REM Draw object
 524 PLOT FN f(x(1)),FN g(y(1))
 525 FOR k=1 TO n-(c$="n"): DRAW
 s(k)*ss,t(k)*ss: NEXT k
 526 PLOT FN f(x(1)),FN g(y(1))
 528 RETURN
```

280 - 290 Subroutine to change the ranges of the scales.

300 - 522 Subroutine to compute the scales and axes.

310 - 370 If both minx and maxx are less than 0, maxx is set to 0. If both minx and maxx are more than 0, minx is set to 0. The range of x is maxx-minx. sx gives the number of pixels per unit necessary on the x-axis.
ox is the horizontal translation from minx to the y-axis.

400 - 460 As above but for the y range.

470 - 480 ss is set to the smaller of sx and sy.

490 - 500 Draws the axes.

510 - Marks the scale on the x-axis.

520 - Marks the scale on the y-axis.

522 - End of the subroutine.

523 - 528 Subroutine to draw the object.
c$ = "n" indicates that the shape is not closed, and so the first and last vertices are not joined.

```
 530 OVER 0: REM Redraw screen
 535 FOR g=1 TO ff
 540 PLOT FN f(v(g,1)),FN g(w(g,
1))
 550 FOR k=1 TO n
 560 LET a(k)=(v(g,k+1)-v(g,k))*
ss: LET b(k)=(w(g,k+1)-w(g,k))*s
s
 565 DRAW a(k),b(k)
 570 NEXT k
 590 NEXT g
 600 FOR k=1 TO hh: PLOT FN f(u(
k)),FN g(z(k)): NEXT k
 610 FOR k=1 TO gg: GO SUB 630:
NEXT k
 615 FOR k=1 TO ii: PLOT FN f(k(
k)),0: DRAW 0,175: NEXT k
 620 RETURN
```

530 - 620 Subroutine to redraw all the images, lines, etc. on display.
  535 - 590 Draws all the previous images.
    540 - Plots the first vertex of the image.
    550 - 570 Draws the edges of the image.
  600 - Plots all previous centres of enlargement/rotation.
  610 - Draws all previous invariant lines of the form y=mx+c (see SUB 630).
  615 - Draws all previous lines of the form x=k.
620 - End of the subroutine.

```
 630 LET lowx=minx
 640 LET ly=m(k)*lowx+c(k)
 650 IF ly>maxy OR ly<miny THEN
LET lowx=lowx+1: GO TO 640
 660 LET hix=maxx
 670 LET hy=m(k)*hix+c(k)
 680 IF hy>maxy OR hy<miny THEN
LET hix=hix-1: GO TO 670
 690 PLOT FN f(lowx),FN g(ly): D
RAW (hix-lowx)*ss,(hy-ly)*ss
 700 RETURN
```

630 - 700 Subroutine to draw lines of the form y = mx + c.
  630 - 650 Finds the lowest value of x (lowx) such that mx + c lies within the range of y by setting lowx=minx and increasing lowx by 1 until the value of m*lowx+c lies between miny and maxy.
  660 - 680 Finds the highest value of x (hix) so that the value of m*hix + c, hiy, lies between miny and maxy.
  690 - Draws the line from x = lowx to x = hix.
700 - End of the subroutine.

```
 730 LET ff=ff+1
 735 FOR k=1 TO n
 740 LET v(ff,k)=x(k): LET w(ff,
k)=y(k)
 750 NEXT k
 760 LET v(ff,n+1)=v(ff,1): LET
w(ff,n+1)=w(ff,1)
 770 RETURN
 775 LET ff=ff+1
 776 FOR k=1 TO n: LET v(ff,k)=e
(k): LET w(ff,k)=f(k): NEXT k
 777 LET v(ff,n+1)=v(ff,1): LET
w(ff,n+1)=w(ff,1)
 778 RETURN
 780 LET gg=gg+1: LET m(gg)=m: L
ET c(gg)=c: RETURN
 790 LET hh=hh+1: LET u(hh)=x: L
ET z(hh)=y: RETURN
```

730 - 770 Subroutine which sets the next members of arrays v and w with the x and y coordinates respectively of the current object.

775 - 778 Subroutine which sets the next members of arrays v and w with the x and y coordinates of the current image. (These are stored in arrays e and f when the subroutine is called.)

780 - Subroutine to store the gradient and the y-intercept of the current invariant line of the form y=mx+c in arrays m and c.

790 - Subroutine to store the coordinates of the current centre of enlargement/rotation in arrays u and z.

```
 800 FOR k=1 TO n
 810 LET x(k)=v(1,k): LET y(k)=w
(1,k)
 815 NEXT k
 820 LET ff=0: LET gg=0: LET hh=
0: LET ii=0
 830 LET minx=mnx: LET maxx=mxx:
 LET miny=mny: LET maxy=mxy
 840 RETURN
```

800 - 840 Subroutine to reset the program with the original object.

800 - 815 Sets the object for the next transformation as the original object (stored in v(1,1 to n) and w(1,1 to n)).

820 - Resets the parameters for drawing the display.

830 - Resets the scales to the original values or those which were input following the 'a' response in line 230.

```
 850 LET minx=mnx: LET maxx=mxx:
 LET miny=mny. LET maxy=mxy
 854 FOR k=1 TO ff
 855 FOR j=1 TO n. LET pp=v(k,j)
: GO SUB 870: LET qq=w(k,j): GO
SUB 885: NEXT j
 856 NEXT k
 860 FOR k=1 TO hh: LET pp=u(k):
 GO SUB 870: LET qq=z(k): GO SUB
 885: NEXT k
 863 FOR k=1 TO gg: LET qq=c(k):
 GO SUB 885: NEXT k
 866 FOR k=1 TO ii: LET pp=k(k):
 GO SUB 870: NEXT k
 868 RETURN
 870 IF pp>maxx THEN LET maxx=IN
T pp+(INT pp<>pp): LET tt=1
 875 IF pp<minx THEN LET minx=IN
T pp: LET tt=1
 880 RETURN
 885 IF qq<miny THEN LET miny=IN
T qq: LET tt=1
 890 IF qq>maxy THEN LET maxy=IN
T qq+(INT qq<>qq): LET tt=1
 895 RETURN
 900 LET ii=ii+1: LET k(ii)=c: R
ETURN
 910 LET x(1)=v(1,1): LET y(1)=w
(1,1)
 915 FOR k=1 TO n
 920 LET x(k+1)=v(1,k+1)
 930 LET y(k+1)=w(1,k+1)
 935 LET s(k)=x(k+1)-x(k): LET t
(k)=y(k+1)-y(k)
 940 NEXT k
 945 RETURN
```

850 - 868 Subroutine to change the ranges of the axes, i.e. the values of mnx, mxx, mny and mxy

854 - 856 Tests all the images on the screen to ensure that the ranges are increased to include them on the new scales.

860 - Tests centres of enlargement/rotation, as above.

863 - Tests the y-intercepts of the invariant lines of the form y = mx + c, as above.

866 - Tests values of k for invariant lines of the form x = k, as above.

870 - 880 Subroutine to increase the range of x to include pp from subroutine 850.
tt is set to 1 if the range has been increased.

885 - 895 Subroutine to increase the range of y to include qq from subroutine 850.
tt is set to 1 if the range has been increased.

900 - Subroutine to store the current value of c in array k, for the invariant lines of the form x = c.

910 - 945 Subroutine to set the object for the next transformation as the original object. It differs from subroutine 800 in that all images on the screen are not erased. This subroutine is called when the 'superimpose' option is chosen in line 230.

```
 950 REM Preset object.DATA give
s the number of points followed
by the coordinates
 960 FOR k=1 TO n: READ x(k): RE
AD y(k): NEXT k
 970 DATA "y",3,6,-3,3,-1,4,1
 980 RETURN
1000 IF INKEY$<>"" THEN GO TO 10
00: REM Rotation about (x,y),ang
le a
1010 LET ret=1000: IF g$="r" THE
N GO TO 1220
1015 IF g$="p" THEN GO TO 1510
1020 IF g$="f" OR g$="s" THEN GO
 TO 1450
1025 IF g$="a" THEN GO TO 1070
```

950 - 980 Subroutine to set the transformation object to the preset object.

970 - "y" indicates that the preset object is closed. The number of vertices is 3. Coordinates (6,-3),(3,-1),(4,1).

1000 - 1700 Rotation subroutine. The subroutine is not executed until the key is released.
ret is the line number for the start of the subroutine.
g$ = "r" indicates that the subroutine has been called by the 'repeat' option of line 230, "p" by the 'previous image' option, "f" by the 'follow' option, "s" by the 'superimpose' option, "a" by the 'axes' option.
The subroutine continues at the appropriate line number, depending on the value of g$.

```
1030 INPUT "Angle of rot. in deg
rees? ";a
1040 INPUT "x coord of centre of
 rot? ";x
1050 INPUT "y coord of centre of
 rot? ";y
1060 GO SUB 790
1070 IF a<>0 THEN GO TO 1140
```

1030 - Input the angle of rotation.

1040 - Input the x coordinate of the centre of rotation.

1050 - Input the y coordinate of the centre of rotation.
1060 - SUB 790 stores these coordinates.

```
1080 IF x<minx THEN LET minx=INT
 x: LET tt=1
1090 IF x>maxx THEN LET maxx=INT
 x+(INT x<>x): LET tt=1
1100 IF y<miny THEN LET miny=INT
 y: LET tt=1
1110 IF y>maxy THEN LET maxy=INT
 y+(INT y<>y): LET tt=1
1120 IF tt=1 THEN GO SUB 300: GO
 SUB 530
1130 PLOT FN f(x),FN g(y)
1135 RETURN
```

1080 - 1135 This section is only executed if the angle of rotation is 0, i.e. the transformation is the identity.
1080 - 1110 Tests x and y, the centre of rotation coordinates, for inclusion in the ranges of the axes. tt = 1 if the ranges have been increased.

1120 - If tt = 1 then SUB 300 clears the screen and redraws the axes, SUB 530 redraws the screen display.
1130 - Plots the centre of rotation.

```
1140 FOR k=1 TO n
1150 LET radius=SQR (ABS (x-x(k)
)↑2+ABS (y-y(k))↑2)
1160 IF (x-radius)<minx THEN LET
 minx=INT (x-radius): LET tt=1
1170 IF (x+radius)>maxx THEN LET
 maxx=INT (x+radius)+(INT (x+rad
ius)<>(x+radius)): LET tt=1
1180 IF (y-radius)<miny THEN LET
 miny=INT (y-radius): LET tt=1
1190 IF (y+radius)>maxy THEN LET
 maxy=INT (y+radius)+(INT (y+rad
ius)<>(y+radius)): LET tt=1
1200 NEXT k
1210 IF tt=1 THEN GO SUB 300: GO
 SUB 530
1220 IF a=0 THEN RETURN
```

1150 - radius = the distance of the object point from the centre of rotation.

1160 - 1190 The ranges of x and y are extended to allow a rotation of 360° about the centre of rotation.

1210 - The screen is redrawn if necessary.

1220 - The transformation subroutine begins here if 'repeat' option has been used. If the angle of rotation is 0 and g$ = "r" then the centre of rotation has already been plotted and RETURN is executed. If angle $\neq$ 0, then the ranges will have been adjusted.

```
1225 PLOT FN f(x),FN g(y): LET d
=PI/180
1230 LET b=a-5*INT (a/5)
1240 IF a>0 THEN LET bb=5
1250 IF a<0 THEN LET bb=-5: LET
b=b-5
1255 IF b=0 THEN LET b=5
1260 LET s=SIN (b*d): LET c=COS
(b*d)
1270 LET e=x(1)*c-y(1)*s+x*(1-c)
+y*s: LET f=x(1)*s+y(1)*c+y*(1-c
)-x*s
1275 FOR k=1 TO n: LET e(k)=s(k)
*ss*c-t(k)*ss*s: LET f(k)=s(k)*s
s*s+t(k)*ss*c: NEXT k
```

1225 - Plots the centre of enlargement. d = 1°, in radians.

1230 - b = a(modulo 5). b is the angle of the first image. bb = the increase in angle for each successive image (5 or -5, depending on a).
If a is a multiple of 5, then b = 5 or -5 depending on a.

1260 - s = the sine of the angle of rotation for the first intermediate image. c = the cosine of the angle.

1270 - (e,f) are the coordinates of the first point of this image.

1275 - arrays e and f contain the vector components of the sides of the image.

```
1280 FOR j=b TO a STEP bb
1290 OVER 1
1300 LET p=FN f(e): LET q=FN g(f
)
1310 FOR k=1 TO n: LET p(k)=e(k)
: LET q(k)=f(k): NEXT k
1320 IF j=a THEN OVER 0: GO TO 1
400
1330 PLOT p,q
1340 FOR k=1 TO n-(c$="n"): DRAW
 p(k),q(k): NEXT k
1350 PLOT p,q
1360 LET s=SIN ((j+bb)*d): LET c
=COS ((j+bb)*d)
1370 LET e=x(1)*c-y(1)*s+x*(1-c)
+y*s: LET f=x(1)*s+y(1)*c+y*(1-c
)-x*s
1380 FOR k=1 TO n: LET e(k)=s(k)
*ss*c-t(k)*ss*s: LET f(k)=s(k)*s
s*s+t(k)*ss*c: NEXT k
1400 PLOT p,q
1410 FOR k=1 TO n-(c$="n"): DRAW
 p(k),q(k): NEXT k
1420 PLOT p,q
1430 NEXT j
```

1280 - 1430 See notes below.*

1300 - (p,q) are the coordinates for the first point of the image, adjusted for scale.

1310 - The arrays e and f have been calculated in the previous execution of the loop.

1320 - The final image is to be drawn if j = a.

1330 - 1350 Draws the image.

1360 - The sine and cosine of the angle of rotation for the next image.

1370 - (e,f) are the coordinates of the first point of the next image.

1380 - The vector components of the sides of the next image.

1400 - 1420 Erases the image (or draws the final image).

* For a rotation of angle a about (x,y) $\begin{pmatrix} x_k \\ y_k \end{pmatrix} \rightarrow \begin{pmatrix} \cos a & -\sin a \\ \sin a & \cos a \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ Let s = sina and c = cosa.

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} \Rightarrow \begin{matrix} cx - sy + t_x = x \Rightarrow t_x = x(1-c) + sy \\ sx + cy + t_y = y \Rightarrow t_y = y(1-c) - sx \end{matrix} \quad \text{and so} \quad \begin{pmatrix} x_k \\ y_k \end{pmatrix} \rightarrow \begin{pmatrix} cx_k - sy_k + x(1-c) + sy \\ sx_k + cy_k + y(1-c) - sx \end{pmatrix}$$

$$\text{Also} \quad \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} \rightarrow \begin{pmatrix} cx_{k+1} - sy_{k+1} + x(1-c) + sy \\ sx_{k+1} + cy_{k+1} + y(1-c) - sx \end{pmatrix} \quad \text{and so} \quad \begin{matrix} (x_{k+1} - x_k) \rightarrow c(x_{k+1} - x_k) - s(y_{k+1} - y_k) \\ (y_{k+1} - y_k) \rightarrow s(x_{k+1} - x_k) + c(y_{k+1} - y_k) \end{matrix}$$

If s_k and t_k are the components of the vector joining (x_k, y_k) to (x_{k+1}, y_{k+1}), then $s_k \rightarrow cs_k - st_k$ and $t_k \rightarrow ss_k + ct_k$. These images are evaluated as e_k and f_k and then transferred to p_k, q_k on the next execution of the loop 1280-1430.This is so that the time-consuming calculations are carried out between the image being drawn and its being erased. This makes the time that the image is on the screen longer than the time between erasing the image and drawing the next.

```
1440 IF d$="y" THEN GO SUB 1520:
 GO TO 1580
1445 RETURN
1450 IF a=0 THEN RETURN
1455 LET x(1)=(p-5)/ss-ox: LET y
(1)=(q-5)/ss-oy
1460 FOR k=1 TO n
1470 LET x(k+1)=x(k)+p(k)/ss: LE
T y(k+1)=y(k)+q(k)/ss: LET s(k)=
p(k)/ss: LET t(k)=q(k)/ss
1480 NEXT k
1490 LET g$="g"
1500 RETURN
```

1440 - SUB 1520 sets arrays e and f with the coordinates of the final image. Line 1580 starts the demonstration.

1450 - 1500 Executed only if the transformation subroutine has been called using the 'follow' or 'superimpose' option. The present image becomes the object for the next transformation if the option is 'follow'. 'Superimpose' will store these coordinates in arrays v and w, and then set the first members of these arrays, i.e. the coordinates of the original object, as the object for the subsequent transformation.

```
1510 IF a=0 THEN RETURN
1520 LET e(1)=(p-5)/ss-ox
1530 LET f(1)=(q-5)/ss-oy
1540 FOR k=1 TO n
1550 LET e(k+1)=e(k)+p(k)/ss: LE
T f(k+1)=f(k)+q(k)/ss
1560 NEXT k
1570 RETURN
```

1510 - 1570 Executed only if the transformation subroutine has been called using the 'previous image' option.
The coordinates of the image are stored in arrays e and f.
The coordinates for the object remain unchanged.

```
1580 OVER 1
1585 FOR k=1 TO n
1590 IF INKEY$="" THEN GO TO 159
0
1591 IF INKEY$<>"" AND INKEY$<>"
f" THEN GO TO 1591
1595 PLOT FN f(x(k)),FN g(y(k)):
 DRAW (x-x(k))*ss,(y-y(k))*ss: P
LOT FN f(x),FN g(y): DRAW (e(k)-
x)*ss,(f(k)-y)*ss
1600 IF INKEY$="" THEN GO TO 160
0
1601 IF INKEY$="f" THEN GO TO 16
05
1602 IF INKEY$<>"" THEN GO TO 16
02
1603 GO TO 1630
1605 PLOT FN f(x(k)),FN g(y(k)):
 DRAW (x-x(k))*ss,(y-y(k))*ss: P
LOT FN f(x),FN g(y): DRAW (e(k)-
x)*ss,(f(k)-y)*ss
1610 GO TO 1595
1630 PLOT FN f(x(k)),FN g(y(k))
1640 DRAW (e(k)-x(k))*ss,(f(k)-y
(k))*ss,a/180*PI
1650 IF INKEY$="" THEN GO TO 165
0
1652 IF INKEY$="f" THEN GO TO 16
60
1655 IF INKEY$<>"" THEN GO TO 16
55
1660 PLOT FN f(x(k)),FN g(y(k))
1670 DRAW (e(k)-x(k))*ss,(f(k)-y
(k))*ss,a/180*PI
1680 IF INKEY$="f" THEN GO TO 16
30
1685 PLOT FN f(x(k)),FN g(y(k)):
 DRAW (x-x(k))*ss,(y-y(k))*ss: P
LOT FN f(x),FN g(y): DRAW (e(k)-
x)*ss,(f(k)-y)*ss
1690 NEXT k
1700 OVER 0: RETURN
```

1580 - 1700 Demonstration of the properties of a rotation.

1591 - 1610 Draws lines from the centre of rotation to the object point and to the image point.

Holding down the 'f' key results in the lines flashing off and on.

Demonstration continues at line 1630 when any other key is pressed (the lines are left drawn on the screen).

1630 - 1680 Draws an arc from the object point (x_k, y_k) to the image point (e_k, f_k) with the centre of rotation as the centre for the arc.

The arc may be caused to flash by holding down the 'f' key.

1685 - Erases the lines and arc.

```
2000 IF INKEY$<>"" THEN GO TO 20
00: REM Reflection in the line y
=mx+c or x=c
2001 LET ret=2000: LET tt=0
2002 IF g$="r" THEN GO TO 2155
2003 IF g$="p" THEN RETURN
2004 IF g$="f" OR g$="s" THEN GO
 TO 2310
2005 IF g$="a" THEN GO TO 2020
2006 POKE 23659,0: PRINT AT 22,0
;"Invariant line,y=mx+c, y/n
                          ": PO
KE 23659,2
2007 IF INKEY$="n" THEN GO TO 25
00
2008 IF INKEY$<>"y" THEN GO TO 2
007
2009 IF INKEY$<>"" THEN GO TO 20
09
2010 INPUT "m? ";m;"  c? ";c
2013 IF c<miny THEN LET miny=c:
LET tt=1
2014 IF c>maxy THEN LET maxy=c:
LET tt=1
```

2000 - 2970 Subroutine for the Reflection transformation. The subroutine is in two parts, one for invariant lines of the form y = mx + c, the other for invariant lines of the form x = c.

The subroutine is only executed when the key is released.

ret = the return address if GOSUB ret is called after the transformation has been completed.

Jumps to various line numbers according to the option which called the subroutine.

2006 - Requires the form of the invariant line to be chosen.

If the invariant line is of the form x = c, then the subroutine continues at line 2500.

2010 - 2014 Input m and c. The y intercept, c, is tested for inclusion in the y range. tt = 1 if the range has been increased.

```
2015 GO SUB 780: LET den=1+(ABS
m)↑2
2020 FOR k=1 TO n
2025 LET p(k)=2*(y(k)*m-c*m-x(k)
*(ABS m)↑2)/den
2030 LET e(k)=x(k)+p(k)
2040 IF e(k)<minx THEN LET minx=
INT e(k): LET tt=1
2045 IF e(k)>maxx THEN LET maxx=
INT e(k)+(INT e(k)<>e(k)): LET t
t=1
2050 LET q(k)=2*(c-y(k)+m*x(k))/
den
2055 LET f(k)=y(k)+q(k)
2060 IF f(k)<miny THEN LET miny=
INT f(k): LET tt=1
2065 IF f(k)>maxy THEN LET maxy=
INT f(k)+(INT f(k)<>f(k)): LET t
t=1
2070 NEXT k
```

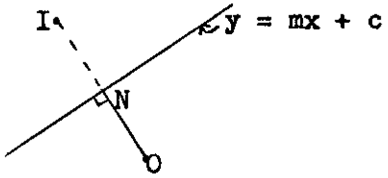
2015 - SUB 780 stores the values of m and c in memory. den = $1 + m^2$.

2025 - 2030 Evaluates the x-coordinate of the image.

2040 - 2045 Tests the x-coordinate for inclusion in the range of x. tt = 1 if the range has been increased.

2050 - 2065 As above, for the y-coordinate of the image.

2020 - 2070 evaluates the coordinates of the image of each point in turn using the following formulae:



If the object, O, has coordinates (x_k, y_k) then $ON = \dfrac{mx_k - y_k + c}{\sqrt{1+m^2}}$

$$\vec{ON} = \begin{pmatrix} \dfrac{m}{\sqrt{1+m^2}} \\ \dfrac{1}{\sqrt{1+m^2}} \end{pmatrix} ON = \begin{pmatrix} \dfrac{my_k - mc - m^2 x_k}{\sqrt{1+m^2}} \\ \dfrac{c - y_k + mx_k}{\sqrt{1+m^2}} \end{pmatrix}$$

$$\vec{OI} = 2\vec{ON} = \begin{pmatrix} p \\ q \end{pmatrix} \qquad I = (x_k + p, y_k + q) = (e, f)$$

```
2080 IF tt=1 THEN GO SUB 300: GO
 SUB 530
2085 LET e(n+1)=e(1): LET f(n+1)
=f(1)
2090 OVER 0: GO SUB 630
2155 OVER 1
2160 FOR k=1 TO n
2165 FOR u=1 TO 2
2170 IF INKEY$="" THEN GO TO 217
0
2175 IF INKEY$="f" THEN GO TO 21
85
2180 IF INKEY$<>"" THEN GO TO 21
80
2185 PLOT FN f(x(k)),FN g(y(k)):
 DRAW (e(k)-x(k))*ss,(f(k)-y(k))
*ss
2190 IF u=2 THEN OVER 0: PLOT FN
 f(e(k)),FN g(f(k)): OVER 1
2191 IF INKEY$="" THEN GO TO 219
1
2192 NEXT u
2195 IF INKEY$="f" THEN GO TO 21
65
2200 NEXT k
2202 IF INKEY$="" THEN GO TO 220
2
2206 IF INKEY$<>"" THEN GO TO 22
06
2210 OVER 0
2240 FOR k=1 TO n-(c$="n")
2250 PLOT FN f(e(k)),FN g(f(k))
2270 LET p(k)=(e(k+1)-e(k))*ss:
LET q(k)=(f(k+1)-f(k))*ss
2280 DRAW p(k),q(k)
2290 NEXT k
2295 PLOT FN f(e(1)),FN g(f(1))
2300 RETURN
```

2080 - Redraws the screen if the scales have been changed.

2155 - The 'exclusive or' print command.

2165 - 2192 is executed twice, the second time erasing the construction line and drawing the image.

2170 - 2180 Pause until a key is pressed and released or until the 'f' key is held down.

2185 - Plots the object and draws a line to the image.

2190 - Plots the image when the line is erased, i.e. when u = 2.

2191 - Pause until a key is pressed.

2195 - Repeats the routine above if the 'f' key is held down.

2202 - 2206 Pause until a key is pressed and released.

2240 - 2295 Draws the final image. If (c$ = "n") = 1, i.e. the shape is not closed, (e_n,f_n) is not joined to (e_1,f_1).

End of Reflection subroutine.

```
2320 FOR k=1 TO n
2330 LET x(k)=e(k): LET y(k)=f(k
): LET s(k)=p(k)/ss: LET t(k)=q(
k)/ss
2340 NEXT k
2345 LET x(n+1)=x(1): LET y(n+1)
=y(1)
2350 OVER 0: RETURN
2500 IF INKEY$<>"" THEN GO TO 25
00
2501 LET ret=2500: LET tt=0
2505 IF g$="p" THEN RETURN
2510 IF g$="r" THEN GO TO 2750
2520 IF g$="f" OR g$="s" THEN GO
 TO 2940
2525 IF g$="a" THEN GO TO 2570
2530 INPUT "Invariant line x=c,
c? ";c
2535 IF c<minx THEN LET minx=c:
LET tt=1
2536 IF c>maxx THEN LET maxx=c:
LET tt=1
2537 GO SUB 900
2570 FOR k=1 TO n
2580 LET s=(x(k)-c)
2590 LET e(k)=x(k)-2*s
2600 IF e(k)<minx THEN LET minx=
INT e(k): LET tt=1
2610 IF e(k)>maxx THEN LET maxx=
INT e(k)+(INT e(k)<>e(k)): LET t
t=1
2650 NEXT k
```

2320 - 2350 Executed when the 'follow' option has been chosen. The image of the reflection becomes the object for the next transformation.

2500 - 2850 Reflection in lines of the form x = c.
Continues only when the key is released.
ret = the return address if the subroutine is recalled.
2505 - 2525 As 2002 - 2005.

2530 - Input the mirror line in the form x = c.

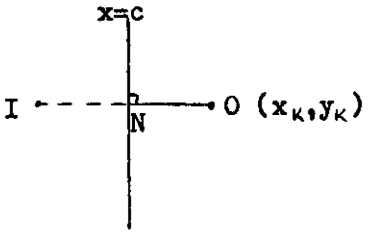
2535 - 2536 Tests c for inclusion in the range of x.

SUB 900 stores this value of c in the array k.

2580 - 2590 Computes the x-coordinate of the image (the y-coordinate remains unchanged).

2600 - 2610 Tests the x-coordinate of the image for inclusion in the range of x.

2570 - 2650 evaluates the coordinates of the image of each point in turn using the following formulae:



$ON = x_K - c = s$ $\qquad \overrightarrow{ON} = \begin{pmatrix} -s \\ 0 \end{pmatrix}$

$\overrightarrow{OI} = 2\overrightarrow{ON} = \begin{pmatrix} -2s \\ 0 \end{pmatrix}$

$I = (e_K, y_K)$ where $e_K = x_K - 2s$.

```
2660 IF tt=1 THEN GO SUB 300: GO
 SUB 530
2670 LET e(n+1)=e(1)
2680 OVER 0
2740 PLOT FN f(c),0: DRAW 0,175
2750 OVER 1
2760 FOR k=1 TO n
2765 FOR u=1 TO 2
2770 IF INKEY$="" THEN GO TO 277
0
2775 IF INKEY$="f" THEN GO TO 27
90
2780 IF INKEY$<>"" THEN GO TO 27
80
2790 PLOT FN f(x(k)),FN g(y(k)):
 LET s=2*(c-x(k))*ss: DRAW s,0
2800 IF u=2 THEN OVER 0: PLOT FN
 f(e(k)),FN g(y(k)): OVER 1
2810 IF INKEY$="" THEN GO TO 281
0
2820 NEXT u
2825 IF INKEY$="f" THEN GO TO 27
65
2830 NEXT k
2840 IF INKEY$="" THEN GO TO 284
0
2850 IF INKEY$<>"" THEN GO TO 28
50
2855 OVER 0
2870 FOR k=1 TO n-(c$="n")
2880 PLOT FN f(e(k)),FN g(y(k))
2890 LET p(k)=(e(k+1)-e(k))*ss:
LET q(k)=(y(k+1)-y(k))*ss
2900 DRAW p(k),q(k)
2910 NEXT k
2915 OVER 0: PLOT FN f(e(1)),FN
g(y(1))
2930 RETURN
2940 FOR k=1 TO n
2950 LET x(k)=e(k): LET s(k)=p(k
)/ss: LET t(k)=q(k)/ss
2960 NEXT k
2965 LET x(n+1)=x(1): LET y(n+1)
=y(1)
2970 RETURN
```

2660 - Redraws the screen if necessary.

2740 - Draws the mirror line.

2750 - 2850 Constructs the image as 2160 - 2200.

2855 - 2930 Draws the image as 2210 - 2300.

2940 - 2970 Executed if the transformation subroutine has been called from the 'follow' option. The image becomes the object for the next transformation.
s(k) and t(k) are the vector components for the sides of the image.

```
3000 IF INKEY$<>"" THEN GO TO 30
00: REM Translation. Input x com
ponent and y component of transl
ation vector
3005 LET ret=3000: LET tt=0
3010 IF g$="r" THEN GO TO 3160
3015 IF g$="a" THEN GO TO 3035
3017 IF g$="p" THEN RETURN
3020 IF g$<>"f" AND g$<>"s" THEN
 GO TO 3030
3023 IF x=0 AND y=0 THEN RETURN
3027 GO TO 5445
3030 INPUT "Vector.X comp? ";x;"
 Y comp? ";y
3031 IF INKEY$<>"" THEN GO TO 30
31
3035 IF x=0 AND y=0 THEN RETURN
3040 FOR k=1 TO n
3090 LET e(k)=x(k)+x
3100 LET f(k)=y(k)+y
3110 IF e(k)<minx THEN LET minx=
INT e(k): LET tt=1
3120 IF e(k)>maxx THEN LET maxx=
INT e(k)+(INT e(k)<>e(k)): LET t
t=1
3130 IF f(k)<miny THEN LET miny=
INT f(k): LET tt=1
3140 IF f(k)>maxy THEN LET maxy=
INT f(k)+(INT f(k)<>f(k)): LET t
t=1
3150 NEXT k
3160 IF tt=1 THEN GO SUB 300: GO
 SUB 530
3170 LET e(n+1)=e(1): LET f(n+1)
=f(1)
3175 IF INKEY$="" THEN GO TO 317
5
3176 IF INKEY$<>"" THEN GO TO 31
76
3180 IF x=0 AND y=0 THEN RETURN
```

3000 - 3500 Translation subroutine.

3000 - The subroutine is executed only when the key is released.

3023 - The identity transformation.

3027 - From line 5445 the image is set as the object for the next transformation, for 'follow' or 'superimpose' options.

3030 - Input the x and y components of the translation vector.

3090 - 3100 Translates (x_K, y_K) onto (e_K, f_K) with vector $\begin{pmatrix} x \\ y \end{pmatrix}$.

3110 - 3140 Tests e_K and f_K for inclusion in the ranges of the x and y axes.

3160 - Redraws the screen if necessary.

Pause until a key is pressed and released.

```
3185 OVER 1
3190 FOR k=1 TO 10
3195 FOR m=1 TO 2
3197 IF k=10 THEN OVER 0
3200 PLOT FN f(x(1)+x*k/10),FN g
(y(1)+y*k/10)
3210 FOR j=1 TO n-(c$="n")
3220 LET p=s(j)*ss: LET q=t(j)*s
s
3225 DRAW p,q
3230 NEXT j
3235 IF c$="y" THEN PLOT FN f(x(
1)+x*k/10),FN g(y(1)+y*k/10)
3240 IF k=10 THEN GO TO 3260
3250 NEXT m
3260 NEXT k
3320 IF INKEY$="" THEN GO TO 332
0
3325 IF INKEY$<>"" THEN GO TO 33
25
3326 IF d$="n" THEN OVER 0: RETU
RN
```

3185 - The 'exclusive or' printing command.

3190 - 3260 Draws 10 stages of the translation. The 10th (final) image is drawn using the OVER 0 command so that it will not erase any lines with which it coincides.

3226 - d$ contains the response to the option for the demonstration. If the response was 'no' then RETURN.

```
3330 FOR k=1 TO n
3335 FOR m=1 TO 2
3340 PLOT FN f(x(k)),FN g(y(k))
3350 DRAW x*ss,y*ss
3355 IF INKEY$="" THEN GO TO 335
5
3358 IF INKEY$="f" THEN GO TO 33
65
3360 IF INKEY$<>"" THEN GO TO 33
60
3365 NEXT m
3368 IF INKEY$="f" THEN GO TO 33
35
3369 IF (k=n AND INKEY$="") OR I
NKEY$="f" THEN GO TO 3368
3370 NEXT k
3372 IF INKEY$<>"" THEN GO TO 33
72
3375 FOR m=1 TO 2
3380 FOR k=1 TO n
3390 PLOT FN f(x(k)),FN g(y(k))
3400 DRAW x*ss,y*ss
3410 NEXT k
3415 IF INKEY$="" THEN GO TO 341
5
3420 IF INKEY$="f" THEN GO TO 34
30
3425 IF INKEY$<>"" THEN GO TO 34
25
3430 NEXT m
3435 IF INKEY$="f" THEN GO TO 33
75
3440 OVER 0
3500 RETURN
```

3330 - 3500 Demonstration of the translation properties.

3330 - 3370 For each point in turn, a line is drawn to its image and then erased (when a key is pressed).
Pressing any key initiates the next stage of the demonstration.
Holding down the 'f' key will cause the line to flash.

3375 - 3435 As above, but the translation lines are drawn for all the points before they are erased.

```
4000 IF INKEY$<>"" THEN GO TO 40
00: REM Enlargement.Input centre
 of enlargement followed by the
scale factor
4005 LET ret=4000: LET tt=0
4006 IF g$="p" THEN RETURN
4008 IF g$<>"r" THEN GO TO 4015
4009 IF sf=1 THEN RETURN
4010 IF g$="r" THEN GO TO 4190
4015 IF g$="a" THEN GO TO 4050
4020 IF g$<>"f" AND g$<>"s" THEN
 GO TO 4030
4027 GO TO 5445
4030 INPUT "C of Enl.X coord?";x
;" Y coord?";y
4040 INPUT "Scale factor? ";sf
4045 GO SUB 790
4050 IF x<minx THEN LET minx=INT
 x: LET tt=1
4060 IF x>maxx THEN LET maxx=INT
 x+(INT x<>x): LET tt=1
4070 IF y<miny THEN LET miny=INT
 y: LET tt=1
4080 IF y>maxy THEN LET maxy=INT
 y+(INT y<>y): LET tt=1
4083 IF tt=1 THEN GO SUB 300: GO
 SUB 530
4087 IF sf=1 THEN PLOT FN f(x),F
N g(y): RETURN
```

4000 - 4620 Enlargement subroutine.

4009 - The identity transformation.

4030 - Input the centre of enlargement (x,y).

4040 - Input the enlargement scale factor.

4045 - SUB 790 stores the coordinates of the centre of enl.

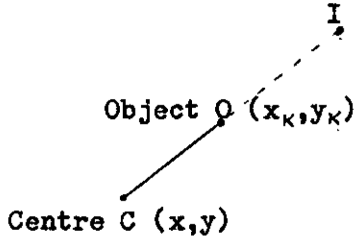
4050 - 4083 Adjusts the scales, if necessary, to include the centre of enlargement.

4087 - If sf = 1, then the transformation is the identity, and the centre of enlargement is plotted.

```
4090 FOR k=1 TO n
4100 LET e(k)=x+(x(k)-x)*sf: LET
 f(k)=y+(y(k)-y)*sf
4110 IF e(k)<minx THEN LET minx=
INT e(k): LET tt=1
4120 IF e(k)>maxx THEN LET maxx=
INT e(k)+(INT e(k)<>e(k)): LET t
t=1
4130 IF f(k)<miny THEN LET miny=
INT f(k): LET tt=1
4140 IF f(k)>maxy THEN LET maxy=
INT f(k)+(INT f(k)<>f(k)): LET t
t=1
4150 NEXT k
4160 IF tt=1 THEN GO SUB 300: GO
 SUB 530
4170 LET e(n+1)=e(1): LET f(n+1)
=f(1)
```

4090 - 4170 For each object point, the coordinates of the image are computed. The scales are adjusted if these coordinates are outside the range of the x and y axes.

Each image is found as follows:



$$\overrightarrow{CO} = \begin{pmatrix} x_K - x \\ y_K - y \end{pmatrix}$$

$$\overrightarrow{CI} = \overrightarrow{CO} * \text{scale factor}$$

$$= \begin{pmatrix} (x_K - x)sf \\ (y_K - y)sf \end{pmatrix}$$

$$I = (x + (x_K - x)sf,\ y + (y_K - y)sf)$$

```
4175 OVER 0
4180 PLOT FN f(x),FN g(y)
4190 IF sf<0 THEN GO TO 4400
4200 FOR k=1 TO n
4210 LET p(k)=(x(k)-x)*ss
4220 LET q(k)=(y(k)-y)*ss
```

4180 - Plots the centre of enlargement.

4190 - The subroutine continues at 4400 if the scale factor is negative.

$\begin{pmatrix} p_K \\ q_K \end{pmatrix}$ is the vector from the centre of enlargement to the object point (x_K,y_K), adjusted for scale.

```
4225 OVER 1
4230 FOR m=1 TO 2
4232 IF INKEY$="" THEN GO TO 423
2
4235 IF INKEY$="f" THEN GO TO 42
40
4238 IF INKEY$<>"" THEN GO TO 42
38
4240 PLOT FN f(x),FN g(y): DRAW
p(k),q(k)
4245 IF INKEY$="" THEN GO TO 424
5
4250 NEXT m
4252 IF INKEY$="f" THEN GO TO 42
30
4253 IF INKEY$="" THEN GO TO 425
2
4255 FOR m=1 TO 2
4260 IF INKEY$="" THEN GO TO 426
0
4265 IF INKEY$="f" THEN GO TO 42
70
4268 IF INKEY$<>"" THEN GO TO 42
68
4270 PLOT FN f(x),FN g(y)
4280 DRAW p(k)*sf,q(k)*sf
4281 IF m=2 THEN OVER 0. PLOT FN
 f(e(k)),FN g(f(k)): OVER 1
4285 IF INKEY$="" THEN GO TO 425
5
4290 NEXT m
4300 IF INKEY$="f" THEN GO TO 42
55
4305 NEXT k
4310 IF INKEY$="" THEN GO TO 431
0
4311 IF INKEY$<>"" THEN GO TO 43
11
4315 OVER 0. PLOT FN f(x),FN g(y
)
4320 FOR k=1 TO n-(c$="n")
4330 PLOT FN f(e(k)),FN g(f(k))
4340 LET p(k)=(e(k+1)-e(k))*ss:
LET q(k)=(f(k+1)-f(k))*ss
4350 DRAW p(k),q(k)
4360 NEXT k
4370 RETURN
```

4230 - 4253 Draws a line from the centre of enlargement to the object point, and then erases it (when a key is pressed and released.
Holding down the 'f' key causes the line to flash.

4255 - 4300 Draws a line from the centre of enlargement to the image point. When a key is pressed and released the line is erased and the image point is plotted.
Holding down the 'f' key causes the line to flash off and on.

Pause until a key is depressed and released.

4315 - Plots the centre of enlargement.

4320 - 4360 Draws the image.

```
4400 FOR k=1 TO n
4410 LET p(k)=(x(k)-x)*ss
4420 LET q(k)=(y(k)-y)*ss
4430 OVER 1
4440 IF INKEY$="" THEN GO TO 444
0
4450 PLOT FN f(x),FN g(y): DRAW
p(k),q(k)
4460 IF INKEY$="f" THEN GO TO 44
90
4470 IF INKEY$<>"" THEN GO TO 44
70
4490 IF INKEY$="f" THEN PLOT FN
f(x),FN g(y): DRAW p(k),q(k): PL
OT FN f(x),FN g(y): DRAW p(k),q(
k): GO TO 4490
4500 IF INKEY$="" THEN GO TO 449
0
4510 PLOT FN f(x),FN g(y): DRAW
p(k)*sf,q(k)*sf
4520 IF INKEY$="f" THEN GO TO 45
40
4530 IF INKEY$<>"" THEN GO TO 45
30
4540 IF INKEY$="f" THEN PLOT FN
f(x),FN g(y): DRAW p(k)*sf,q(k)*
sf: PLOT FN f(x),FN g(y): DRAW p
(k)*sf,q(k)*sf: GO TO 4540
4550 IF INKEY$="" THEN GO TO 454
0
4560 PLOT FN f(x),FN g(y): DRAW
p(k),q(k)
4570 PLOT FN f(x),FN g(y): DRAW
p(k)*sf,q(k)*sf
4580 OVER 0: PLOT FN f(x),FN g(y
): PLOT FN f(e(k)),FN g(f(k))
4585 IF INKEY$<>"" THEN GO TO 45
85
4590 NEXT k
4595 IF INKEY$="" THEN GO TO 459
5
4600 IF INKEY$<>"" THEN GO TO 46
00
4620 GO TO 4320
```

4400 - 4620 Executed only if the scale factor is negative. The procedure is as 4200 - 4315 except that the line from the centre of enlargement to the object is not erased before the line from the centre to the image has been drawn.

```
5000 IF INKEY$<>"" THEN GO TO 50
00: REM Shear. Input invariant l
ine y=mx+c or x=c, and a point w
ith its image
5005 LET ret=5000: LET tt=0
5006 IF g$<>"r" THEN GO TO 5020
5008 IF x=xx AND y=yy THEN RETUR
N
5010 IF g$="r" THEN GO TO 5250
5020 IF g$="f" OR g$="s" THEN GO
 TO 5440
5024 IF g$="p" THEN RETURN
5025 IF g$="a" THEN GO TO 5070
5030 POKE 23659,0: PRINT AT 22,0
;"Invariant line,y=mx+c, y/n
                               ":
 POKE 23659,2
5031 IF INKEY$="n" THEN GO TO 55
00
5032 IF INKEY$<>"y" THEN GO TO 5
031
5033 IF INKEY$<>"" THEN GO TO 50
33
5034 INPUT "m? ";m;"  c? ";c
5035 IF c<miny THEN LET miny=INT
 c: LET tt=1
5036 IF c>maxy THEN LET maxy=INT
 c+(INT c<>c): LET tt=1
5037 GO SUB 780
5040 IF INKEY$<>"" THEN GO TO 50
40
5045 INPUT "Object x coord?";x;"
 y coord?";y
5050 INPUT "Image x coord?";xx;"
 y coord?";yy
5055 IF INKEY$<>"" THEN GO TO 50
55
5060 LET d=m*x-y+c: LET a=xx-x:
LET b=yy-y: LET error=0
5062 IF a=0 AND b=0 THEN LET k=g
g: GO SUB 630: RETURN
```

5000 - 5970 Subroutine for the Shear transformation.

5008 - The identity transformation.

5030 - Allows the choice of the form of the invariant line, y = mx + c or x = c.

5031 - If the invariant line is of the form x = c, then the subroutine continues at line 5500.

5034 - Input m and c.

5035 - 5036 Tests if c lies within the range of the y-axis. tt = 1 if the range is increased.

5037 - SUB 780 stores m and c.

5045 - 5050 Input an object (x,y) and its image (xx,yy) under the shear.

5060 - $\begin{pmatrix} a \\ b \end{pmatrix}$ is the vector from (x,y) to (xx,yy).

5062 - If a = 0 and b = 0 then the transformation is the identity. SUB 630 draws the invariant line.

```
5063 IF a=0 THEN POKE 23659,0: P
RINT AT 22,0;"Incorrect informat
ion - Press    any key
                 ": POKE 23659,2: LET
 error=1
5064 IF error=1 AND INKEY$="" TH
EN GO TO 5064
5065 IF error=1 AND INKEY$<>"" T
HEN GO TO 5040
5066 IF ABS (b/a-m)>1E-4 OR y=m*
x+c THEN POKE 23659,0: PRINT AT
22,0;"Incorrect information - Pr
ess    any key
      ": POKE 23659,2: LET error=
1
5067 IF error=1 AND INKEY$="" TH
EN GO TO 5067
5068 IF error=1 AND INKEY$<>"" T
HEN GO TO 5040
5070 IF x<minx THEN LET minx=INT
 x: LET tt=1
5072 IF y<miny THEN LET miny=INT
 y: LET tt=1
5073 IF x>maxx THEN LET maxx=INT
 x+(INT x<>x): LET tt=1
5074 IF y>maxy THEN LET maxy=INT
 y+(INT y<>y): LET tt=1
5075 IF xx<minx THEN LET minx=IN
T xx: LET tt=1
5076 IF yy<miny THEN LET miny=IN
T yy: LET tt=1
5077 IF xx>maxx THEN LET maxx=IN
T xx+(INT xx<>xx): LET tt=1
5078 IF yy>maxy THEN LET maxy=IN
T yy+(INT yy<>yy): LET tt=1
5079 IF tt=1 THEN GO SUB 300: GO
 SUB 530
```

5063 - 5065 An error message is printed if a = 0, since b $\neq$ 0 and the invariant line is not vertical.
Pressing a key returns to 5040 which requires (x,y) and (xx,yy) to be input again. If a $\neq$ 0 then the subroutine continues.

5066 - 5068 An error message is printed if $\frac{b}{a} \neq m$, since each point must shear parallel to the invariant line, or if y = mx + c, i.e. (x,y) lies on the invariant line.

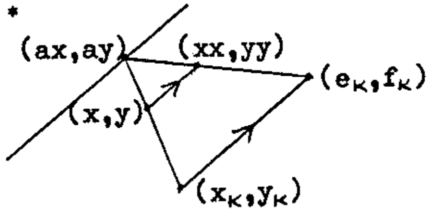
5070 - 5079 The ranges of the x and y axes are extended, if necessary, to include (x,y) and (xx,yy).

```
5080 FOR k=1 TO n
5085 LET s=(m*x(k)-y(k)+c)/d
5090 LET e(k)=x(k)+s*a
5094 IF ABS (s-1)<1E-4 THEN LET
ax=x: GO TO 5100
5095 LET ax=x+(x(k)-x)/(1-s)
5100 LET f(k)=y(k)+s*b
5104 IF ABS (s-1)<1E-4 THEN LET
ay=y: GO TO 5110
5105 LET ay=y+(y(k)-y)/(1-s)
```

5085 - 5105 (See below for the mathematical formulae.*)
$d = mx + c - y$ (previously calculated). $s = \dfrac{mx_K + c - y_K}{mx + c - y}$

(e_K,f_K) is the image of (x_K,y_K). (ax,ay) is the point on the invariant line as shown in the diagram, unless (x,y) and (x_K,y_K) are the same distance from the invariant line, in which case ax and ay are given the values of x and y respectively. Lines 5094 and 5104 avoid division by 0.

*



$$\text{Distance of } (x,y) \text{ from the invariant line} = \frac{mx + c - y}{\sqrt{1 + m^2}} = d_1 .$$

$$\text{Distance of } (x_K,y_K) \text{ fron the invariant line} = \frac{mx_K + c - y_K}{\sqrt{1 + m^2}} = d_2 .$$

$$e_K = x_K + \frac{ad_2}{d_1} , \; f_K = y_K + \frac{bd_2}{d_1} .$$

```
5110 IF e(k)<minx THEN LET minx=
INT e(k): LET tt=1
5115 IF ax<minx THEN LET minx=IN
T ax: LET tt=1
5120 IF e(k)>maxx THEN LET maxx=
INT e(k)+(INT e(k)<>e(k)): LET t
t=1
5125 IF ax>maxx THEN LET maxx=IN
T ax+(INT ax<>ax): LET tt=1
5130 IF f(k)<miny THEN LET miny=
INT f(k): LET tt=1
5135 IF ay<miny THEN LET miny=IN
T ay: LET tt=1
5140 IF f(k)>maxy THEN LET maxy=
INT f(k)+(INT f(k)<>f(k)): LET t
t=1
5145 IF ay>maxy THEN LET maxy=IN
T ay+(INT ay<>ay): LET tt=1
5150 NEXT k
5160 IF tt=1 THEN GO SUB 300: GO
 SUB 530
5170 LET e(n+1)=e(1): LET f(n+1)
=f(1)
```

5110 - 5170 Extends the ranges, if necessary, of the x and y axes to include (e_K,f_K) and also (ax,ay) - this is necessary since (ax,ay) will be required on the diagram in the demonstration which follows the transformation (if d$ = "y").

```
5180 OVER 0: GO SUB 630
5250 OVER 1
5280 FOR u=1 TO 10
5285 FOR k=1 TO n
5290 LET p(k)=x(k)+(e(k)-x(k))*u
/10: LET q(k)=y(k)+(f(k)-y(k))*u
/10
5295 NEXT k
5300 LET p(n+1)=p(1): LET q(n+1)
=q(1)
5305 FOR j=1 TO 2
5310 IF u=10 THEN LET j=2. OVER
0
5312 PLOT FN f(p(1)),FN g(q(1))
5315 FOR k=1 TO n-(c$="n"): IF u
=10 THEN OVER 0: GO TO 5325
5316 LET infg=x(k+1)-x(k)
5317 IF ABS infg<1E-4 THEN GO TO
 5325



5318 IF ABS ((y(k+1)-y(k))/infg-
m)>1E-4 OR ABS ((p(k)-x(k))/infg
)>=1 THEN GO TO 5325
5319 IF SGN infg<>SGN (p(k)-x(k)
) THEN GO TO 5321
5320 OVER 0. DRAW (x(k+1)-p(k))*
ss,(y(k+1)-q(k))*ss: OVER 1: DRA
W (p(k+1)-x(k+1))*ss,(q(k+1)-y(k
+1))*ss: GO TO 5330
5321 DRAW (x(k)-p(k))*ss,(y(k)-q
(k))*ss: PLOT FN f(x(k)),FN g(y(
k)): OVER 0: DRAW (p(k+1)-x(k))*
ss,(q(k+1)-y(k))*ss: OVER 1: GO
TO 5330
5325 DRAW (p(k+1)-p(k))*ss,(q(k+
1)-q(k))*ss
5330 NEXT k
5332 IF j=1 THEN PAUSE 10
5335 NEXT j
5340 NEXT u
5345 IF d$="y" THEN GO TO 6000
5430 RETURN
```

5180 - Draws the invariant line.

5280 - 5340 Draws 10 intermediate stages of the shear, the 10th being the final image.

5285 - 5295 Evaluates the intermediate points for the images, using k/10 of the vector joining (x_k,y_k) to (e_k,f_k).

5305 - 5335 Each image is drawn twice, the second time to erase the image, except the final image.

5312 - Plots the first point of the intermediate image.

5315 - 5325 See notes on the next page *.

u = 10 - the final image.

5316 - 5317 If $x_{k+1} - x_k = 0$ then the side of the object is not parallel to the invariant line.

5318 - Since infg $(x_{k+1} - x_k)$ does not equal zero, infg may be used as a divisor without the risk of error. GOTO 5325 if the side of the object does not have a gradient m or if the intermediate image does not coincide with the object.

5319 - Case 2 (see notes).

5320 - Case 1 (see notes).

5321 - Case 2 (see notes).

5325 - Not executed if 5320 or 5321 has been drawn.

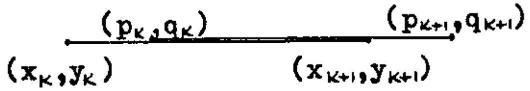
5345 - Jumps to 6000 if the demonstration is required.

* Notes on section 5315 - 5325

Because it is likely that an edge (line) of an object is parallel to the invariant line, this means that the drawing of the intermediate image will erase part of the object if OVER 1 is used. This section has been written to avoid this, as follows:

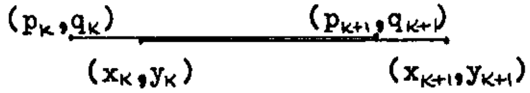
Consider the line and its image (shown in red). We have two possibilities depending on the direction of the shear.

Case 1



We require (p_k,q_k) to (x_{k+1},y_{k+1}) to be drawn using OVER 0 and (x_{k+1},y_{k+1}) to (p_{k+1},q_{k+1}) to be drawn using OVER 1.

Case 2



We require (p_k,q_k) to (x_k,y_k) to be drawn using OVER 1 and (x_k,y_k) to (p_{k+1},q_{k+1}) to be drawn using OVER 0.

```
5440 IF x=xx AND y=yy THEN RETUR
N
5442 LET e(n+1)=e(1): LET f(n+1)
=f(1)
5445 FOR k=1 TO n
5450 LET x(k)=e(k): LET y(k)=f(k
): LET s(k)=e(k+1)-e(k): LET t(k
)=f(k+1)-f(k)
5460 NEXT k
5465 LET x(n+1)=x(1): LET y(n+1)
=y(1)
5470 RETURN
```

5440 - The identity transformation.

5442 - 5470 Executed if the 'follow' or 'superimpose' option is chosen. The image of the shear becomes the object for the next transformation.

```
5500 IF INKEY$<>"" THEN GO TO 55
00
5501 LET ret=5500: LET tt=0
5503 IF g$="p" THEN RETURN
5505 IF g$<>"r" THEN GO TO 5520
5508 IF x=xx AND y=yy THEN RETUR
N
5510 IF g$="r" THEN GO TO 5750
5520 IF g$="f" OR g$="s" THEN GO
 TO 5940
5525 IF g$="a" THEN GO TO 5570
5530 INPUT "Invariant line x=c,
c? ";c
5535 IF c<minx THEN LET minx=c:
LET tt=1
5536 IF c>maxx THEN LET maxx=c:
LET tt=1
5537 GO SUB 900
5540 IF INKEY$<>"" THEN GO TO 55
40
5545 INPUT "Object x coord?";x;"
 y coord?";y
5550 INPUT "Image x coord?";xx;"
 y coord?";yy
5555 IF INKEY$<>"" THEN GO TO 55
55
5560 LET a=0: LET b=yy-y: LET d=
x-c: LET error=0
5561 IF x<>xx OR x=c THEN LET er
ror=1: POKE 23659,0: PRINT AT 22
,0;"Incorrect information - Pres
s any key                   ": P
OKE 23659,2
5562 IF error=1 AND INKEY$="" TH
EN GO TO 5562
5563 IF error=1 AND INKEY$<>"" T
HEN GO TO 5540
5568 IF b=0 THEN PLOT FN f(c),0:
 DRAW 0,175: RETURN
```

5500 - 5970 This part of the subroutine is executed in a similar way to 5000 - 5430, but for invariant lines of the form x=c.

5530 - Input c, where x = c.

5535 - 5536 tt = 1 if the range of x has been increased to accommodate c.

5537 - SUB 900 stores the value of c.

5545 - 5555 Input an object point (x,y) and its image (xx,yy) under the shear.

5560 - $\begin{pmatrix}a\\b\end{pmatrix}$ is the vector from (x,y) to (xx,yy) and so a = 0.
d = the distance of (x,y) from the line x = c.

5561 - 5563 Error message if x $\neq$ xx or if x = c, i.e. (x,y) lies on the line x = c

5568 - b = 0 gives the identity, in which case the invariant line is drawn and RETURN is executed.

```
5570 IF x<minx THEN LET minx=INT
 x: LET tt=1
5571 IF y<miny THEN LET miny=INT
 y: LET tt=1
5572 IF x>maxx THEN LET maxx=INT
 x+(INT x<>x): LET tt=1
5573 IF y>maxy THEN LET maxy=INT
 y+(INT y<>y): LET tt=1
5574 IF xx<minx THEN LET minx=IN
T xx: LET tt=1
5575 IF yy<miny THEN LET miny=IN
T yy: LET tt=1
5576 IF xx>maxx THEN LET maxx=IN
T xx+(INT xx<>xx): LET tt=1
5577 IF yy>maxy THEN LET maxy=IN
T yy+(INT yy<>yy): LET tt=1
5580 IF tt=1 THEN GO SUB 300: GO
 SUB 530
```

5570 - 5580 Increases the ranges of the axes, if necessary, to include (x,y) and (xx,yy).

$x_K - c$ is the distance from (x_K,y_K) to the line $x = c$.

(x_K,y_K) will shear vertically a distance $(yy - y)\frac{x_K - c}{x - c} = s$.

So, $e_K = x_K$ and $f_K = y_K + s$.
(ax,ay) is, as previously, the point where the line through (x,y) and (x_K,y_K) meets the invariant line. If $x_K - c = x - c$ then ay is given the value of y.

```
5585 FOR k=1 TO n
5590 LET s=b/d*(x(k)-c)
5600 LET f(k)=y(k)+s: LET e(k)=x
(k)
5605 IF ABS (b-s)<1E-4 THEN LET
ay=y: GO TO 5630
5610 LET ay=y+(y(k)-y)*b/(b-s)
5630 IF f(k)<miny THEN LET miny=
INT f(k): LET tt=1
5635 IF ay<miny THEN LET miny=IN
T ay: LET tt=1
5640 IF f(k)>maxy THEN LET maxy=
INT f(k)+(INT f(k)<>f(k)): LET t
t=1
5645 IF ay>maxy THEN LET maxy=IN
T ay+(INT ay<>ay): LET tt=1
5650 NEXT k
5660 IF tt=1 THEN GO SUB 300: GO
 SUB 530
5670 LET f(n+1)=f(1): LET e(n+1)
=x(n+1)
5680 OVER 0
5740 PLOT FN r(c),0: DRAW 0,175
```

5630 - 5660 Increases the ranges of the x and y axes, if necessary, to accommodate (e_K,f_K) and (ax,ay).

5740 - Draws the invariant line.

```
5750 OVER 1
5760 FOR u=1 TO 10
5770 FOR k=1 TO n
5780 LET q(k)=y(k)+(f(k)-y(k))*u
/10
5790 NEXT k
5800 LET q(n+1)=q(1)
5810 FOR j=1 TO 2
5820 IF u=10 THEN LET j=2: OVER
0
5830 PLOT FN f(x(1)),FN g(q(1))
5840 FOR k=1 TO n-(c$="n"): IF u
=10 THEN OVER 0: GO TO 5850
5841 IF y(k+1)=y(k) THEN GO TO 5
850
5842 IF x(k)=0 AND x(k+1)=0 THEN
 OVER 0: GO TO 5850
5843 IF x(k+1)<>x(k) OR ABS ((q(
k)-y(k))/(y(k+1)-y(k)))>=1 THEN
GO TO 5850
5845 IF SGN (y(k+1)-y(k))<>SGN (
f(k)-y(k)) THEN GO TO 5848
5847 OVER 0: DRAW 0,(y(k+1)-q(k)
)*ss: OVER 1: DRAW 0,(q(k+1)-y(k
+1))*ss: GO TO 5860
5848 DRAW 0,(y(k)-q(k))*ss: OVER
 0: DRAW 0,(q(k+1)-y(k))*ss: OVE
R 1: GO TO 5860
5850 DRAW (x(k+1)-x(k))*ss,(q(k+
1)-q(k))*ss
5855 OVER 1
5860 NEXT k
5870 IF j=1 THEN PAUSE 10
5880 NEXT j
5890 NEXT u
5900 IF d$="y" THEN GO TO 6000
5920 OVER 0
5930 RETURN
5940 IF x=xx AND y=yy THEN RETUR
N
5942 LET e(n+1)=e(1): LET f(n+1)
=f(1)
5945 FOR k=1 TO n
5950 LET y(k)=f(k): LET s(k)=e(k
+1)-e(k): LET t(k)=f(k+1)-f(k)
5960 NEXT k
5965 LET x(n+1)=x(1): LET y(n+1)
=y(1)
5970 RETURN
```

5760 - 5890 Draws 10 intermediate stages of the shear.

5780 - The intermediate points will be (x_k, q_k) as the x coordinate does not change.

5820 - The final image is drawn only once, in the OVER 0 mode.

5842 - $x_k = 0$ and $x_{k+1} = 0 \Rightarrow$ the image and object sides coincide with the y-axis, and so OVER 0 is used.

5843 - This condition occurs if the image and object lines do not coincide.

5845 - 5848 Draws the image line when it overlaps the object line, as before in lines 5319 - 5321.

5850 - Not executed if 5847 or 5848 has been executed.

5900 - Jump to 6000 if the demonstration is required.

5930 - End of the subroutine.

5940 - Identity transformation, therefore RETURN.

5942 - 5970 Subroutine to set the image of the shear as the object for the next transformation.

```
6000 IF INKEY$="" THEN GO TO 600
0: REM Demonstration for shear
6001 IF INKEY$<>"" THEN GO TO 60
01
6005 OVER 1
6010 FOR k=1 TO n
6020 LET e=e(k)-x(k): LET f=f(k)
-y(k)
6025 IF ABS e<1E-4 AND ABS f<1E-
4 THEN GO TO 6480
6030 LET p(k)=x-x(k): LET q(k)=y
-y(k)
6035 IF ABS (a-e)<1E-4 AND ABS (
b-f)<1E-4 THEN GO TO 6130
6040 LET mm=1
6050 FOR u=1 TO mm
6060 IF SGN a<>SGN e OR SGN b<>S
GN f THEN PLOT FN f(x(k)),FN g(y
(k)): DRAW p(k)*ss,q(k)*ss: GO T
O 6090
6070 IF (ABS a<ABS e) OR (ABS b<
ABS f) THEN PLOT FN f(x(k)),FN g
(y(k)): DRAW p(k)*ss*f/(f-b),q(k
)*ss*f/(f-b)
6080 IF (ABS a>ABS e) OR (ABS b>
ABS f) THEN PLOT FN f(x),FN g(y)
: DRAW p(k)*ss*b/(f-b),q(k)*ss*b
/(f-b)
6090 NEXT u
6100 IF INKEY$="" THEN GO TO 610
0
6110 IF INKEY$="f" THEN LET mm=2
: GO TO 6050
6120 IF INKEY$<>"" THEN GO TO 61
20
6130 LET mm=1
6140 FOR u=1 TO mm
6150 PLOT FN f(x),FN g(y): DRAW
a*ss,b*ss
6160 NEXT u
6170 IF INKEY$="" THEN GO TO 617
0
6180 IF INKEY$="f" THEN LET mm=2
: GO TO 6140
6190 IF INKEY$<>"" THEN GO TO 61
90
```

6000 - 6500 Subroutine to demonstrate the properties of a shear.

6020 - $\begin{pmatrix}e\\f\end{pmatrix}$ is the vector which shears (x_K,y_K) onto its image.

6025 - If e = 0 and f = 0 then no demonstration is given as the point lies on the invariant line.

6035 - If the object point is the same distance from the invariant line as the specified object, omit 6040 - 6120.

6060 - If (x_K,y_K) is on the opposite side of the invariant line from the specified object (x,y) then a line is drawn from (x_K,y_K) to (x,y) to cross the invariant line at (ax,ay).

6070 - If (x,y) is nearer the invariant line than (x_K,y_K) and on the same side, a line is drawn from (x_K,y_K) through (x,y) and extended to meet the invariant line at (ax,ay).

6080 - If (x_K,y_K) is nearer the invariant line than (x,y) and on the same side, a line is drawn from (x,y) through (x_K,y_K) and extended to meet the invariant line at (ax,ay).

6050 - 6090 is executed once unless the 'f' key is held down, when the line is drawn and then erased, causing it to flash. The program continues to 6130 only when the line has been drawn an odd number of times in the OVER 1 mode. This ensures that the line is not erased at this stage.

6140 - 6190 A line is drawn from (x,y) to (xx,yy) with the option of 'flashing'by holding down the 'f' key.

```
6200 LET p(K)=xx-e(K): LET q(K)=
yy-f(K)
6205 IF ABS (a-e)<1E-4 AND ABS (
b-f)<1E-4 THEN GO TO 6300
6210 LET mm=1
6220 FOR u=1 TO mm
6230 IF SGN a<>SGN e OR SGN b<>S
GN f THEN PLOT FN f(e(K)),FN g(f
(K)): DRAW p(K)*ss,q(K)*ss: GO T
O 6260
6240 IF (ABS a<ABS e) OR (ABS b<
ABS f)>1E-4 THEN PLOT FN f(e(K))
,FN g(f(K)): DRAW p(K)*ss*f/(f-b
),q(K)*ss*f/(f-b)
6250 IF (ABS a>ABS e) OR (ABS b>
ABS f) THEN PLOT FN f(xx),FN g(y
y): DRAW p(K)*ss*b/(f-b),q(K)*ss
*b/(f-b)
6260 NEXT u
6270 IF INKEY$="" THEN GO TO 627
0
6280 IF INKEY$="f" THEN LET mm=2
: GO TO 6220
6290 IF INKEY$<>"" THEN GO TO 62
90
6300 LET mm=1
6310 FOR u=1 TO mm
6320 PLOT FN f(x(K)),FN g(y(K)):
 DRAW e*ss,f*ss
6330 NEXT u
6340 IF INKEY$="" THEN GO TO 634
0
6350 IF INKEY$="f" THEN LET mm=2
: GO TO 6310
6360 IF INKEY$<>"" THEN GO TO 63
60
```

6200 - $\begin{pmatrix}p\\q\end{pmatrix}$ is the vector from (e_K,f_K) to (xx,yy).

6205 - If the object (x_K,y_K) and the specified object (x,y) are the same distance from the invariant line, omit 6210 - 6290.

6210 - 6290 As section 6040 - 6120, to draw a line from (e_K,f_K) to (xx,yy) produced if necessary to meet the invariant line at (ax,ay).

6300 - 6360 Draws a line from (x_K,y_K) to its image (e_K,f_K).

```
6370 LET p(k)=x-x(k): LET q(k)=y
-y(k)
6375 IF ABS (a-e)<1E-4 AND ABS (
b-f)<1E-4 THEN GO TO 6410
6380 IF SGN a<>SGN e OR SGN b<>S
GN f THEN PLOT FN f(x(k)),FN g(y
(k)): DRAW p(k)*ss,q(k)*ss: GO T
O 6410
6390 IF (ABS a<ABS e) OR (ABS b<
ABS f) THEN PLOT FN f(x(k)),FN g
(y(k)): DRAW p(k)*ss*f/(f-b),q(k
)*ss*f/(f-b)
6400 IF (ABS a>ABS e) OR (ABS b>
ABS f) THEN PLOT FN f(x),FN g(y)
: DRAW p(k)*ss*b/(f-b),q(k)*ss*b
/(f-b)
6410 PLOT FN f(x),FN g(y): DRAW
a*ss,b*ss
6420 LET p(k)=xx-e(k): LET q(k)=
yy-f(k)
6425 IF ABS (a-e)<1E-4 AND ABS (
b-f)<1E-4 THEN GO TO 6455
6430 IF SGN a<>SGN e OR SGN b<>S
GN f THEN PLOT FN f(e(k)),FN g(f
(k)): DRAW p(k)*ss,q(k)*ss: GO T
O 6455
6440 IF (ABS a<ABS e) OR (ABS b<
ABS f) THEN PLOT FN f(e(k)),FN g
(f(k)): DRAW p(k)*ss*f/(f-b),q(k
)*ss*f/(f-b)
6450 IF (ABS a>ABS e) OR (ABS b>
ABS f) THEN PLOT FN f(xx),FN g(y
y): DRAW p(k)*ss*b/(f-b),q(k)*ss
*b/(f-b)
6455 PLOT FN f(x(k)),FN g(y(k)):
 DRAW e*ss,f*ss
6460 IF INKEY$="" THEN GO TO 645
0
6470 IF INKEY$<>"" THEN GO TO 64
70
6480 NEXT k
6500 OVER 0: RETURN
```

6370 - 6500 **Erases the lines previously drawn in section** 6040 - 6360 **by redrawing them using the OVER 1 mode.**

```
7000 IF INKEY$<>"" THEN GO TO 70
00: REM Stretch. Input invariant
 line y=mx+c or x=c, and the
 scale factor.
7005 LET ret=7000: LET tt=0
7006 IF g$<>"r" THEN GO TO 7020
7008 IF sf=1 THEN RETURN
7010 IF g$="r" THEN GO TO 7250
7020 IF g$="f" OR g$="s" THEN GO
 TO 7440
7024 IF g$="p" THEN RETURN
7025 IF g$="a" THEN GO TO 7070
7030 POKE 23659,0: PRINT AT 22,0
;"Invariant line,y=mx+c. y/n
                                ":
 POKE 23659,2
7031 IF INKEY$="n" THEN GO TO 75
00
7032 IF INKEY$<>"y" THEN GO TO 7
031
7033 IF INKEY$<>"" THEN GO TO 70
33
7034 INPUT "m? ";m;"  c? ";c
7035 IF c<miny THEN LET miny=INT
 c: LET tt=1
7036 IF c>maxy THEN LET maxy=INT
 c+(INT c<>c): LET tt=1
7037 GO SUB 780
7040 IF INKEY$<>"" THEN GO TO 70
40
7050 INPUT "Scale factor?  ";sf
7060 IF sf=1 THEN LET k=99: GO S
UB 630: RETURN
7075 IF tt=1 THEN GO SUB 300: GO
 SUB 530
```

7000 - 7970 Subroutine to stretch an object with a given scale factor from an invariant line.

7008 - The identity transformation.

7030 - As for the shear, this subroutine is written in two parts, one for invariant lines of the form y = mx + c, the other for invariant lines of the form x = c.

7031 - If the invariant line is of the form x = c, then the program continues at line 7500.

7034 - Input m and c for the line y = mx + c.

7035 - 7036 increases the range of the y-axis, if necessary, to include c.

7037 - SUB 780 stores m and c.

7050 - Input the scale factor for the stretch, sf.

7060 - If sf = 1 then the transformation is the identity.

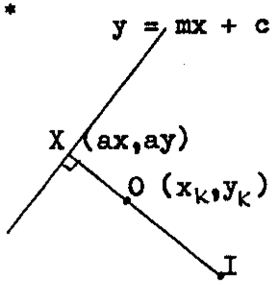
7075 - Redraws the screen if the range of the y-axis was increased.

```
7080 FOR k=1 TO n
7085 LET d=(m*x(k)-y(k)+c)/(m*m+
1): LET s=d*(sf-1)
7090 LET e(k)=x(k)+s*m
7095 LET ax=x(k)-d*m
7100 LET f(k)=y(k)-s
7105 LET ay=y(k)+d
7110 IF e(k)<minx THEN LET minx=
INT e(k): LET tt=1
7115 IF ax<minx THEN LET minx=IN
T ax: LET tt=1
7120 IF e(k)>maxx THEN LET maxx=
INT e(k)+(INT e(k)<>e(k)): LET t
t=1
7125 IF ax>maxx THEN LET maxx=IN
T ax+(INT ax<>ax): LET tt=1
7130 IF f(k)<miny THEN LET miny=
INT f(k): LET tt=1
7135 IF ay<miny THEN LET miny=IN
T ay: LET tt=1
7140 IF f(k)>maxy THEN LET maxy=
INT f(k)+(INT f(k)<>f(k)): LET t
t=1
7145 IF ay>maxy THEN LET maxy=IN
T ay+(INT ay<>ay): LET tt=1
7150 NEXT k
```

7085 - 7105 Evaluates the coordinates of the image of each object point (x_K, y_K) and also where the perpendicular from (x_K, y_K) meets the invariant line (see notes below *).

7100 - 7145 Adjusts the ranges of the x and y axes, if necessary, to accommodate these points on the screen.

*



$$OX = \frac{mx_K - y_K + c}{\sqrt{1 + m^2}} \qquad IX = OX * sf \qquad OI = OX * (sf - 1)$$

$$\overrightarrow{OI} = (sf - 1)\overrightarrow{XO} = (sf - 1)\begin{pmatrix} \frac{m}{\sqrt{1 + m^2}} \\ \frac{-1}{\sqrt{1 + m^2}} \end{pmatrix} OX = (sf - 1)\begin{pmatrix} \frac{m(mx_K - y_K + c)}{1 + m^2} \\ \frac{mx_K - y_K + c}{1 + m^2} \end{pmatrix} = s\begin{pmatrix} m \\ -1 \end{pmatrix}$$

$$I = (x_K + sm, y_K - s) \quad X = (x_K - dm, y_K + d) \text{ where } d = \frac{mx_K - y_K + c}{1 + m^2}$$

```
7160 IF tt=1 THEN GO SUB 300: GO
 SUB 530
7170 LET e(n+1)=e(1): LET f(n+1)
=f(1)
7180 OVER 0: GO SUB 630
7250 OVER 1
7280 FOR v=1 TO 10
7285 FOR k=1 TO n
7290 LET p(k)=x(k)+(e(k)-x(k))*v
/10: LET q(k)=y(k)+(f(k)-y(k))*v
/10
7295 NEXT k
7300 LET p(n+1)=p(1): LET q(n+1)
=q(1)
7305 FOR j=1 TO 2
7310 IF v=10 THEN LET j=2: OVER
0
7312 PLOT FN f(p(1)),FN g(q(1))
7315 FOR k=1 TO n-(c$="n"): IF v
=10 THEN OVER 0: GO TO 7390
7320 LET infg=x(k+1)-x(k): IF AB
S infg<1E-4 THEN GO TO 7340
7325 IF m=0 THEN GO TO 7390
7330 IF ABS ((y(k+1)-y(k))/infg+
1/m)>1E-4 THEN GO TO 7390
7335 GO TO 7350
7340 IF m<>0 THEN GO TO 7390
7345 IF x(k)=0 AND m=0 THEN OVER
 0: GO TO 7390
```

7160 - Redraws the screen if the ranges have been increased.

7180 - Draws the invariant line.

7280 - 10 intermediate stages of the stretch are drawn and erased.

7290 - (p_k, q_k) are the coordinates of the intermediate images.

7312 - First point of the image.

7320 - If $x_k = x_{k+1}$ then the object line is vertical. Jumps to 7340 to see if the gradient of the invariant line is 0.

7340 - If the gradient is not zero, then jump to 7390.

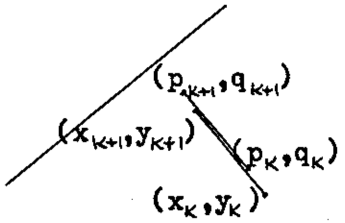
7345 - If $x_k = 0$ and $m = 0$ then the object line is perpendicular to the invariant line and also coincides with the y-axis.

7320 - 7345 establishes whether or not the line joining (x_k, y_k) to (x_{k+1}, y_{k+1}) is perpendicular to the invariant line. If it is, the intermediate images may coincide with the object, in which case the image needs to be drawn so that parts of the object are not erased. The notes on the following page explain how this is achieved.

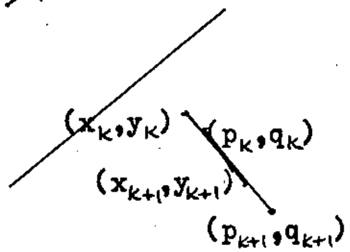
```
7350 IF SGN (y(k+1)-q(k+1))<>SGN
 (y(k)-q(k)) THEN GO TO 7385
7355 IF ABS (y(k)-f(k))<ABS (y(k
+1)-f(k+1)) THEN GO TO 7370
7360 IF sf>1 AND ABS (q(k+1)-y(k
+1))<ABS (y(k)-y(k+1)) THEN GO T
O 7380
7365 IF sf<1 AND ABS (q(k)-y(k))
<ABS (y(k+1)-y(k)) THEN GO TO 73
75
7366 GO TO 7390
7370 IF sf<1 AND ABS (q(k+1)-y(k
+1))<ABS (y(k)-y(k+1)) THEN GO T
O 7380
7371 IF sf>1 AND ABS (q(k)-y(k))
<ABS (y(k+1)-y(k)) THEN GO TO 73
75
7373 GO TO 7390
```

7350 - 7373 is executed if the object line is perpendicular to the invariant line. Six possibilities exist where the object and image coincide. These are shown below (the images have been drawn in red).

1. $sf<1$

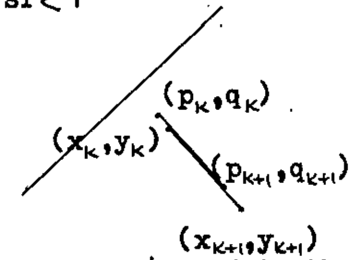


2. $sf>1$

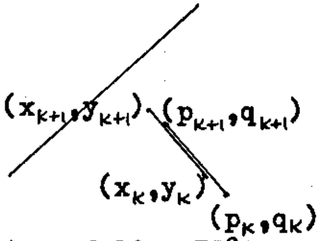


Cases 1 and 2 lead to the execution of line 7375 .

3. $sf<1$

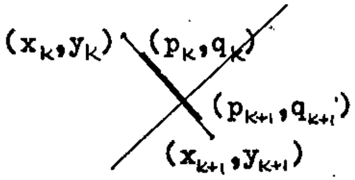


4. $sf>1$

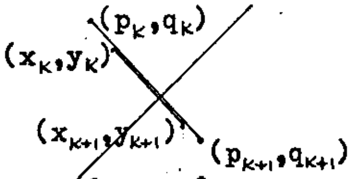


Cases 3 and 4 lead to the execution of line 7380.

5. $sf<1$



6. $sf>1$



Case 5 leads to lines 7386 and 7390, Case 6 to 7385.

```
7375 OVER 0: DRAW (x(k+1)-p(k))*
ss,(y(k+1)-q(k))*ss: OVER 1: DRA
W (p(k+1)-x(k+1))*ss,(q(k+1)-y(k
+1))*ss: GO TO 7420
7380 DRAW (x(k)-p(k))*ss,(y(k)-q
(k))*ss: PLOT FN f(x(k)),FN g(y(
k)): OVER 0: DRAW (p(k+1)-x(k))*
ss,(q(k+1)-y(k))*ss: OVER 1: GO
TO 7420
7385 IF sf>1 THEN DRAW (x(k)-p(k
))*ss,(y(k)-q(k))*ss: OVER 0: PL
OT FN f(x(k)),FN g(y(k)): DRAW (
x(k+1)-x(k))*ss,(y(k+1)-y(k))*ss
: OVER 1: DRAW (p(k+1)-x(k+1))*s
s,(q(k+1)-y(k+1))*ss: GO TO 7420
7386 IF sf<1 THEN OVER 0
7390 DRAW (p(k+1)-p(k))*ss,(q(k+
1)-q(k))*ss
7395 OVER 1
7420 NEXT k
7422 IF j=1 THEN PAUSE 10
7424 NEXT j
7426 NEXT u
7428 IF d$="y" THEN GO TO 8000
7429 OVER 0
7430 RETURN
7440 IF sf=1 THEN RETURN
7442 LET e(n+1)=e(1): LET f(n+1)
=f(1)
7445 FOR k=1 TO n
7450 LET x(k)=e(k): LET y(k)=f(k
): LET s(k)=e(k+1)-e(k): LET t(k
)=f(k+1)-f(k)
7460 NEXT k
7465 LET x(n+1)=x(1): LET y(n+1)
=y(1)
7470 RETURN
```

7375 - 7390 Draws the line from (p_k,q_k) to (p_{k+1},q_{k+1}) using the OVER 0 mode where the line coincides with the object line and OVER 1 mode elsewhere.

7428 - Jump to 8000 if the demonstration is required.

7440 - 7470 If sf = 1 then the transformation is the identity. This section is executed when the 'follow' or 'superimpose' option is chosen. The image of the stretch is set as the object for the next transformation. If 'superimpose' was chosen then this object will be stored in arrays u and v, and then the first members of arrays u and v will be set as the next object.

```
7500 IF INKEY$<>"" THEN GO TO 75
00
7501 LET ret=7500: LET tt=0
7503 IF g$="p" THEN RETURN
7505 IF g$<>"r" THEN GO TO 7520
7508 IF sf=1 THEN RETURN
7510 IF g$="r" THEN GO TO 7750
7520 IF g$="f" OR g$="s" THEN GO
 TO 7940
7525 IF g$="a" THEN GO TO 7570
7530 INPUT "Invariant line x=c,
c? ";c
7535 IF c<minx THEN LET minx=c:
LET tt=1
7536 IF c>maxx THEN LET maxx=c:
LET tt=1
7537 GO SUB 900
7540 IF INKEY$<>"" THEN GO TO 75
40
7550 INPUT "Scale factor?  ";sf
7560 IF sf=1 THEN PLOT FN f(c),0
: DRAW 0,175: RETURN
7580 IF tt=1 THEN GO SUB 300: GO
 SUB 530
7585 FOR k=1 TO n
7590 LET d=(x(k)-c): LET s=d*(sf
-1)
7600 LET f(k)=y(k): LET e(k)=x(k
)+s
7630 IF e(k)<minx THEN LET minx=
INT e(k): LET tt=1
7640 IF e(k)>maxx THEN LET maxx=
INT e(k)+(INT e(k)<>e(k)): LET t
t=1
7650 NEXT k
7660 IF tt=1 THEN GO SUB 300: GO
 SUB 530
7670 LET f(n+1)=f(1): LET e(n+1)
=e(1)
7680 OVER 0
7740 PLOT FN f(c),0: DRAW 0,175
```

7500 - 7970 Subroutine for a stretch when the invariant line is of the form x = c.

7530 - Input c, where x = c is the invariant line.

7537 - SUB 900 stores the value of c.

7550 - Input the scale factor, sf.

7560 - If sf = 1 the transformation is the identity, in which case the invariant line is drawn, followed by RETURN.

7580 - Redraws the screen if necessary.

7590 - d = distance of the object point from the invariant line.
s = displacement from the object point to its image.

7600 - The image of (x_K,y_K) is (e_K,y_K) as the stretch is horizontal.

7640 - Adjusts the scales and redraws the screen as necessary.

7740 - Draws the invariant line.

```
7750 OVER 1
7760 FOR u=1 TO 10
7770 FOR k=1 TO n
7780 LET p(k)=x(k)+(e(k)-x(k))*u
/10
7790 NEXT k
7800 LET p(n+1)=p(1)
7810 FOR j=1 TO 2
7820 IF u=10 THEN LET j=2: OVER
0
7830 PLOT FN f(p(1)),FN g(y(1))
7840 FOR k=1 TO n-(c$="n"): IF u
=10 THEN OVER 0: GO TO 7905
7845 IF y(k+1)<>y(k) THEN GO TO
7905
7850 IF y(k)=0 THEN OVER 0: GO T
O 7905
7855 IF SGN (x(k+1)-p(k+1))<>SGN
 (x(k)-p(k)) THEN GO TO 7895
7860 IF ABS (x(k)-e(k))<ABS (x(k
+1)-e(k+1)) THEN GO TO 7880
7865 IF sf>1 AND ABS (p(k+1)-x(k
+1))<ABS (x(k)-x(k+1)) THEN GO T
O 7890
7870 IF sf<1 AND ABS (p(k)-x(k))
<ABS (x(k+1)-x(k)) THEN GO TO 78
85
7875 GO TO 7905
7880 IF sf<1 AND ABS (p(k+1)-x(k
+1))<ABS (x(k)-x(k+1)) THEN GO T
O 7890
7881 IF sf>1 AND ABS (p(k)-x(k))
<ABS (x(k+1)-x(k)) THEN GO TO 78
85
7883 GO TO 7905
```

7750 - 7883 Draws 10 intermediate images for the stretch as described in 7250 - 7373.

```
7885 OVER 0: DRAW (x(k+1)-p(k))*
ss,0: OVER 1: DRAW (p(k+1)-x(k+1
))*ss,0: GO TO 7910
7890 DRAW (x(k)-p(k))*ss,0: PLOT
 FN f(x(k)),FN g(y(k)): OVER 0:
DRAW (p(k+1)-x(k))*ss,0: OVER 1:
 GO TO 7910
7895 IF sf>1 THEN DRAW (x(k)-p(k
))*ss,0: OVER 0: DRAW (x(k+1)-x(
k))*ss,0: OVER 1: DRAW (p(k+1)-x
(k+1))*ss,0: GO TO 7910
7900 IF sf<1 THEN OVER 0
7905 DRAW (p(k+1)-p(k))*ss,(y(k+
1)-y(k))*ss: OVER 1
7910 NEXT k
7915 NEXT j
7920 NEXT u
7925 IF d$="y" THEN GO TO 8000
7930 OVER 0: RETURN
7940 IF sf=1 THEN RETURN
7942 LET e(n+1)=e(1): LET f(n+1)
=f(1)
7945 FOR k=1 TO n
7950 LET x(k)=e(k): LET s(k)=e(k
+1)-e(k): LET t(k)=f(k+1)-f(k)
7960 NEXT k
7955 LET x(n+1)=x(1): LET y(n+1)
=y(1)
7970 RETURN
```

7885 - 7920 As 7375 - 7426.

7925 - Jump to 8000 if the demonstration was requested.

7940 - 7970 As 7440 - 7470.

```
8000 IF INKEY$="" THEN GO TO 800
0: REM Demonstration for stretch
8001 IF INKEY$<>"" THEN GO TO 80
01
8005 OVER 1
8010 FOR k=1 TO n
8020 LET e=e(k)-x(k): LET f=f(k)
-y(k)
8025 IF e=0 AND f=0 THEN GO TO 8
300
8030 FOR u=1 TO 2
8040 IF INKEY$="" THEN GO TO 804
0
8050 IF INKEY$="f" THEN GO TO 80
70
8060 IF INKEY$<>"" THEN GO TO 80
60
8070 PLOT FN f(x(k)),FN g(y(k)):
 DRAW -e*ss/(sf-1),-f*ss/(sf-1)
8100 IF INKEY$="" THEN GO TO 810
0
8105 NEXT u
8110 IF INKEY$="f" THEN GO TO 80
30
8120 IF INKEY$="" THEN GO TO 811
0
8130 FOR u=1 TO 2
8140 IF INKEY$="" THEN GO TO 814
0
8150 IF INKEY$="f" THEN GO TO 81
70
8160 IF INKEY$<>"" THEN GO TO 81
60
8170 PLOT FN f(e(k)),FN g(f(k)):
 DRAW -e*ss*sf/(sf-1),-f*ss*sf/(
sf-1)
8200 IF INKEY$="" THEN GO TO 820
0
8205 NEXT u
8210 IF INKEY$="f" THEN GO TO 81
30
8220 IF INKEY$="" THEN GO TO 821
0
8235 IF INKEY$<>"" THEN GO TO 82
35
```

8000 - 8350 Demonstration of the properties of the stretch.
Pressing a key and releasing it moves the demonstration into the next stage.

8020 - $\begin{pmatrix} e \\ f \end{pmatrix}$ is the vector from (x_k, y_k) to (e_k, f_k).

8025 - If the object point is on the invariant line, GOTO 8300.

8030 - 8120 Draws a line from the object point onto the invariant line, perpendicular to it.
Erases the line when a key is pressed and released.
The line flashes off and on when the 'f' key is held down.

8130 - 8235 Draws a line from the image point onto the invariant line, perpendicular to it.
Erases the line on the second execution of the loop.
Holding down the 'f' key will cause the line to flash.

```
8240 FOR u=1 TO 2
8250 IF INKEY$="" THEN GO TO 825
0
8260 IF INKEY$="f" THEN GO TO 82
70
8265 IF INKEY$<>"" THEN GO TO 82
65
8270 PLOT FN f(x(k)),FN g(y(k)):
 DRAW e*ss,f*ss
8275 IF INKEY$="" THEN GO TO 827
5
8280 NEXT u
8285 IF INKEY$="f" THEN GO TO 82
40
8290 IF INKEY$="" THEN GO TO 828
5
8300 NEXT k
8305 IF INKEY$="" THEN GO TO 830
5
8306 IF INKEY$<>"" THEN GO TO 83
06
8340 OVER 0
8350 RETURN
```

8240 - 8350 Draws the line from the object point to the image point and then erases it. Holding down the 'f' key will cause the line to flash.

Appendix B - The listing for the program 'num-pat'.

```
   1 INK 7: BORDER 0: PAPER 0: C
LS : RESTORE 4
   2 PAUSE 0
   3 FOR k=0 TO 7: READ x: POKE
USR "a"+k,x: NEXT k
   4 DATA 0,8,0,0,127,0,0,8
   5 LET i=0
  10 PRINT "Triangle numbers...▨
                Square numbers...▨
                Tn+Tn-1=Sn...▨"
  20 IF INKEY$="t" THEN GO TO 17
0
  22 IF INKEY$="r" THEN GO TO 50
0
  30 IF INKEY$<>"s" THEN GO TO 2
0
  31 CLS
  35 PRINT "Sn=Sn-1 + (2n-1)....
▨               1+3+5+7.....▨
                1+2+3+...+2+1...▨"
  36 IF INKEY$="b" THEN GO TO 10
00
  37 IF INKEY$="c" THEN GO TO 80
0
  38 IF INKEY$="x" THEN CLS : GO
 TO 5
  39 IF INKEY$<>"a" THEN GO TO 3
6
```

1 - Sets background to black, writing to white. Restores DATA line 4

2 - Pause until a key is pressed.

3 - Sets the user defined graphics 'a' to ÷ which will be used for division on the display as / is unfamiliar to the pupils for whom this program was developed.

10 - Prints the option message.

20 - 30 Jumps to the appropriate section of the program according to the reponse to the options.
Only the 't', 'r', and 's' keys produce a response.

31 - clears the screen.

35 - Prints the options for the square number demonstrations.

36 - 39 Jumps to the appropriate section for the square numbers option.
Pressing the 'x' key returns to the options in line 10.
Only the 'a', 'b', 'c' and 'x' keys produce a response.

```
  40 CLS
  50 LET i=i+1
  60 INK 3
  70 FOR k=1 TO i
  80 PRINT AT i,k-1;"■";AT k,i-1
;"■": NEXT k
  85 INK 7
  86 IF i=1 THEN GO TO 115
  90 PRINT AT i,20;(i-1)↑2;"+";
FLASH 1;"?"
  95 IF INKEY$="" THEN GO TO 95
  96 IF INKEY$<>"" THEN GO TO 96
 100 FLASH 0: PRINT AT i,20;(i-1
)↑2;"+";2*i-1;" = "; FLASH 1;"?"
 105 IF INKEY$="" THEN GO TO 105
 106 IF INKEY$<>"" THEN GO TO 10
6
 110 FLASH 0: PRINT AT i,20;(i-1
)↑2;"+";2*i-1;" = ";i↑2
 111 GO TO 125
 115 IF INKEY$<>"" THEN GO TO 11
5
 116 IF INKEY$="" THEN GO TO 116
 117 IF INKEY$<>"" THEN GO TO 11
7
 120 PRINT AT i,20;"1"
 122 IF INKEY$="" THEN GO TO 122
 123 IF INKEY$<>"" THEN GO TO 12
3
 125 INK 4
 130 FOR k=1 TO i: PRINT AT i,k-
1;"■";AT k,i-1;"■": NEXT k
 150 IF INKEY$="" THEN GO TO 150
 155 FLASH 0
 160 IF i=20 OR INKEY$="x" THEN
GO TO 310
 161 GO TO 50
```

40 - 161 Demonstration of $s_{n-1} + 2n - 1 = s_n$.
i = the number of dots in the side of the square, initially set to 1 and increased by 1 at each execution of the loop.

60 - 80 Prints a magenta gnomon of dots around the green square of dots, as in figure 2.19.

86 - Jumps to 115 if the first square (1 dot) is displayed.

90 - Prints the number of dots in the green square, followed by " + ?". The question mark flashes.

95 - 96 Pause until a key is pressed and released.

100 - Overprints '?' with the number of dots in the gnomon, followed by " = ?". The question mark flashes.

110 - Overprints '?' with the total sum of dots on display.

111 - Omits lines 115 - 123 when i > 1.

115 - Demonstration continues only when the option key is released.

116 - Pause until a key is pressed and released.

120 - Prints '1' for the first square (one dot).

122 - 123 Pause.

125 - 130 Changes the dots of the gnomon to green, making the next square.

150 - Pause until a key is pressed.

155 - Normal mode of flashing.

160 - Jumps to the section which displays the sequence without the dot demonstration when the 'x' key is pressed or the screen is filled (20 by 20 dot square).

161 - Repeats the demonstration for the next square.

```
 170 CLS
 171 PRINT "Tn+(n+1)=Tn+1...▨
              1+2+3....+n=Tn...▨
              n(n+1)/2=Tn...▨"
 172 IF INKEY$="c" THEN GO TO 12
00
 173 IF INKEY$="b" THEN GO TO 14
00
 174 IF INKEY$="x" THEN CLS : GO
 TO 5
 175 IF INKEY$<>"a" THEN GO TO 1
72
 176 CLS : LET s=0
 177 IF INKEY$<>"" THEN GO TO 17
7
 180 LET i=i+1
 185 LET s=s+i
 190 FOR k=1 TO i
 200 INK 4: PRINT AT i,9-INT (i/
2)+k;"▝" AND INT (i/2)*2=i;"▘" A
ND INT (i/2)*2<>i
 201 NEXT k
 202 IF INKEY$="" THEN GO TO 202
 203 IF INKEY$<>"" THEN GO TO 20
3
 206 IF i=1 THEN GO TO 220
 208 INK 7: PRINT AT i,20;s-i;"+
"; FLASH 1;"?"
 210 IF INKEY$="" THEN GO TO 210
 211 IF INKEY$<>"" THEN GO TO 21
1
 212 FLASH 0: PRINT AT i,20;s-i;
"+";i;" = "; FLASH 1;"?"
 215 IF INKEY$="" THEN GO TO 215
 216 IF INKEY$<>"" THEN GO TO 21
6
 218 FLASH 0: PRINT AT i,20;s-i;
"+";i;" = ";s: GO TO 225
 220 INK 7: PRINT AT i,20;"1"
 225 FOR k=1 TO i
 230 INK 3: PRINT AT i,9-INT (i/
2)+k;"▝" AND INT (i/2)*2=i;"▘" A
ND INT (i/2)*2<>i
 240 NEXT k
 250 IF INKEY$="" THEN GO TO 250
 255 IF s=210 OR INKEY$="x" THEN
 GO TO 420
 260 GO TO 180
```

170 - Clears the screen.

171 - Prints the options for the triangle numbers demonstrations.

172 - 175 Pressing the 'x' key returns to the initial options. Pressing the 'a', 'b' or 'c' key jumps to the corresponding demonstration.
No other key produces a response.

176 - s is used to store successive triangle numbers.

177 - Demonstration A continues when the key is released.

180 - i stores successive integers.

185 - Generates the next triangle number.

190 - 201 Prints a row of green dots below the previous triangle of dots. To keep the pattern of dots running diagonally, when i is even the dot is in the top right of the character square, when i is odd the dot is in the top left corner.

202 - 203 Pause.

206 - If the first triangle number is displayed (1 dot) GOTO 220.

208 - 218 Prints, in stages, the numerical values of
$t_{i-1} + i = t_i$, where t_n is the n^{th} triangle number.
The question mark flashes at each stage until a key is pressed, when it is overwritten with the numerical value.

220 - "1" is printed for the first triangle number.

225 - 240 Changes the row of green dots to magenta, the colour of the previous triangle.

250 - 260 Repeats the demonstration for the next triangle number when a key is pressed. If the 'x' key is pressed or the screen is full the sequence above is printed without the dots.

```
 300 CLS
 305 LET i=0
 310 IF INKEY$<>"" THEN GO TO 31
0
 312 INK 7
 315 LET i=i+1
 320 PRINT ;(i-1)↑2;" + ";2*i-1;
" = ";i↑2
 325 IF INKEY$="x" THEN CLS : GO
 TO 5
 330 GO TO 310
```

300 - 305 Redundant unless it is required that the sequence below is printed beginning with the first term at the top of the screen, in which case line 160 needs to be changed - GOTO 310 is replaced with GOTO 300.

310 - 330 When the key is released, the numerical values of the sequence $s_{i-1} + (2i - 1) = s_i$ continues being printed below the last square which was demonstrated.
Pressing a key halts the sequence until the key is released.
Pressing the 'x' key returns to the initial option display.

```
 400 CLS
 405 LET s=0: LET i=0
 420 INK 7
 425 IF INKEY$<>"" THEN GO TO 42
5
 430 LET i=i+1
 440 LET s=s+i
 450 PRINT s-i;" + ";i;" = ";s
 455 IF INKEY$="x" THEN CLS : GO
 TO 5
 460 GO TO 420
```

400 - 405 Redundant unless it is required that the sequence below is printed beginning with the first term at the top of the screen, in which case line 255 needs to be changed - GOTO 420 is replaced with GOTO 400.

420 - 460 When the key is released the numerical values of the sequence $t_{i-1} + i = t_i$ continues, being printed below the last triangle which was demonstrated.
Holding down a key halts the sequence, until the key is released.
Pressing the 'x' key returns to the initial option display.

```
 500 CLS
 505 LET s=0: LET n=-1
 515 LET n=n+1
 516 CLS
 517 LET s=s+n
 520 FOR i=0 TO n
 525 INK 3
 530 FOR k=1 TO i
 540 PRINT AT i,9-INT (i/2)+k;"▝
" AND INT (i/2)*2=i;"▘" AND INT
(i/2)*2<>i
 545 NEXT k
 550 FOR j=i TO n
 555 INK 4
 560 PRINT AT i,11-INT (i/2)+j;"
▝" AND INT (i/2)*2=i;"▘" AND INT
 (i/2)*2<>i
 570 NEXT j
 580 NEXT i
 590 IF INKEY$="" THEN GO TO 590
 591 IF INKEY$<>"" THEN GO TO 59
1
 600 INK 7
 610 PRINT AT n+1,11+n/2;s+n+1;A
T n+1,10;
 615 IF s=0 THEN GO TO 698
 616 PRINT s
 617 IF INKEY$="" THEN GO TO 617
 618 IF INKEY$<>"" THEN GO TO 61
8
 619 PRINT AT n+1,10.5+n/4;"+";A
T n+1,14+n/2;"="; FLASH 1;AT n+1
,16+n/2;"?"
 620 IF INKEY$="" THEN GO TO 620
 621 IF INKEY$<>"" THEN GO TO 62
1
```

500 - 740 Demonstration of the relation $t_n + t_{n-1} = s_n$.

515 - n stores successive integers. Initially n = 0.

517 - s stores the current triangle number.

520 - n is the number of rows of dots in the larger triangle.

530 - 545 Prints a row of dots (magenta) of the first triangle. The rows alternate with the dots in the top left and then in the top right corner of a character square to ensure a diagonal pattern.

550 - 570 Prints a green row of dots for the second triangle.

580 - Next row of dots of the triangles.

610 - Prints the number of dots in the first triangle.

616 - Prints the number of dots in the second triangle.

617 - 618 Pause until a key is pressed and released.

619 - Prints '+' between the two numbers, followed by ' = ?'. The question mark flashes.

620 - 621 Pause until a key is pressed and released.

```
 622 INK 3
 623 FOR i=n TO 0 STEP -1
 625 FOR k=1 TO i
 630 PRINT AT i,9-INT (i/2)+k;"
";AT i,9-INT (n/2)+k;"■"
 640 NEXT k
 641 NEXT i
 643 FOR q=0 TO 100: NEXT q
 645 INK 4
 646 FOR i=n TO 0 STEP -1
 650 FOR j=i TO n
 660 PRINT AT i,11-INT (i/2)+j;"
";AT i,11-INT (n/2)+j;"■"
 662 NEXT j
 664 NEXT i
 666 FOR q=0 TO 100: NEXT q
 670 FOR i=n TO 0 STEP -1
 672 PRINT AT i,11-INT (n/2)+n;"
";AT i,10-INT (n/2)+i;"■"
 675 NEXT i
 685 IF INKEY$="" THEN GO TO 685
 690 IF INKEY$<>"" THEN GO TO 69
0
 691 INK 7: FLASH 0: PRINT AT n+
1,16+n/2;(n+1)↑2
 692 INK 5
 693 FOR i=0 TO n
 694 FOR j=0 TO n
 695 PRINT AT i,10-INT (n/2)+j;"
■"
 696 NEXT j
 697 NEXT i
 698 IF INKEY$="" THEN GO TO 698
 699 IF INKEY$="x" OR n=19 THEN
GO TO 710
 700 GO TO 515
 710 IF INKEY$<>"" THEN GO TO 71
0
 712 INK 7: PRINT : LET n=n+1: L
ET s=s+n
 715 LET n=n+1
 720 LET s=s+n
 730 PRINT s-n;" + ";s;" = ";n↑2
 735 IF INKEY$="x" THEN CLS : GO
 TO 5
 740 GO TO 715
```

622 - 641 Rearranges the first triangle of dots into a right-angled triangle, row by row.

643 - Pause for about two seconds.

645 - 664 Rearranges the second triangle into a right-angled triangle, row by row.

666 - Pause for about two seconds.

670 - 675 Moves the second triangle next to the first triangle to form a square.

685 - 690 Pause until a key is pressed and released.

691 - Overwrites the question mark with the number of dots in the square.

692 - 697 Changes the colour of all the dots, row by row, to cyan.

698 - 700 Pressing a key repeats the demonstration for the next pair of triangle numbers. Pressing the 'x' key or if the screen is full (n = 19) jumps to line 710.

710 - The program continues only when the key is released.

712 - 740 Continues printing the sequence without the dot display. Pressing the 'x' key returns to the initial option display.

```
 800 CLS : INK 4
 805 FOR i=1 TO 10
 810 FOR j=1 TO i
 820 FOR k=1 TO i
 830 PRINT AT j,k;"■"
 840 NEXT k
 850 NEXT j
 854 IF INKEY$<>"" AND i=1 THEN
GO TO 854
 855 IF INKEY$="" THEN GO TO 855
 856 IF INKEY$<>"" THEN GO TO 85
6
 860 FOR l=1 TO i
 870 PLOT 8*l,171: DRAW 8*(i+1-l
)-1,-8*(i+1-l)+1
 880 PLOT 4,175-8*l: DRAW 8*(i+1
-l)-1,-8*(i+1-l)+1
 890 NEXT l
 900 IF INKEY$="" THEN GO TO 900
 901 IF INKEY$<>"" THEN GO TO 90
1
 905 INK 7
 910 FOR l=1 TO i-1: PRINT AT l+
1,i+1;l;AT i+1,l+1;l: NEXT l
 911 PRINT AT i+1,i+1;i
 915 IF INKEY$="" THEN GO TO 915
 916 IF INKEY$<>"" THEN GO TO 91
6
 918 PRINT '''
 919 IF i=1 THEN GO TO 980
 920 FOR l=1 TO i: PRINT l;"+";:
 NEXT l
 930 FOR l=i-1 TO 1 STEP -1: PRI
NT l;"+" AND l<>1;: NEXT l
 940 PRINT "=";: FLASH 1: PRINT
"?";
 950 IF INKEY$="" THEN GO TO 950
 951 IF INKEY$<>"" THEN GO TO 95
1
 960 FLASH 0: PRINT ;CHR$ 8;i↑2
 970 IF INKEY$="" THEN GO TO 970
 980 CLS
 981 IF INKEY$="x" THEN GO TO 5
 985 INK 4
 990 NEXT i
 995 INK 7: GO TO 5
```

800 - 995 Demonstrates 1 + 2 + 3 .... n .... 3 + 2 + 1 = n^2.

805 - The demonstration shows the first 10 square numbers.

810 - 850 Prints a square of green dots.

854 - The demonstration will not begin until the option key (c) is released.

855 - 856 Pause until a key is pressed and released.

860 - 890 Draws straight lines to divide the square into diagonal rows of dots.

Pause.

905 - 911 Prints the number of dots in the diagonal rows, at the edge of the square.

Pause.

918 - Three lines space beneath the square.

919 - Omits lines 920 - 970 for the first square.

920 - 940 Prints '1 + 2 + 3 +' up to the value of i, followed by the values of '(i-1) + (i-2)+ ... + 2 + 1 = ?'
The question mark flashes.

Pause until a key is pressed and released.

960 - Overprints the question mark with the square number.

970 - 981 Pressing a key continues the demonstration.
Pressing the 'x' key returns to the initial option display.

990 - Repeats the demonstration for the next square.

995 - Return to the initial option display after the 10x10 square.

```
1000 CLS : INK 4
1005 FOR i=1 TO 20
1010 FOR j=1 TO i
1020 FOR k=1 TO i
1030 PRINT AT j,k;"■"
1040 NEXT k
1050 NEXT j
1055 IF INKEY$<>"" AND 1=1 THEN
GO TO 1055
1060 IF INKEY$="" THEN GO TO 106
0
1061 IF INKEY$<>"" THEN GO TO 10
61
1070 FOR l=1 TO i: PLOT 8*(i+1-l
)-2,171: DRAW 0,-8*l-2: DRAW 8*l
,0: NEXT l
1080 IF INKEY$="" THEN GO TO 108
0
1081 IF INKEY$<>"" THEN GO TO 10
81
1082 INK 7
1085 FOR l=1 TO i: PRINT AT l,i+
1;2*l-1: NEXT l
1087 IF INKEY$="" THEN GO TO 108
7
1088 IF INKEY$<>"" THEN GO TO 10
88
1090 FOR l=1 TO i-1: PRINT AT l,
i+3;"+": NEXT l
1092 PLOT 8*i+8,167-8*i: DRAW 24
,0
1093 IF INKEY$="" THEN GO TO 109
3
1094 IF INKEY$<>"" THEN GO TO 10
94
1096 PRINT AT i+2,i+1;i↑2
1097 IF INKEY$="" THEN GO TO 109
7
1098 IF INKEY$="x" OR i=19 THEN
CLS : GO TO 5
1099 CLS : INK 4
1100 NEXT i
```

1000 - 1100 Demonstration of 1 + 3 + 5 + ... + (2n-1) = n^2.

1010 - 1050 Prints a square arrangement of green dots.

1055 - The demonstration begins only when the option key (b) is released.

1060 - 1061 Pause until a key is pressed and released.

1070 - Divides the square into gnomons.

Pause.

1082 - 1085 Prints the number of dots in each gnomon, at the edge of the square.

Pause.

1090 - Prints a column of +'s next to the odd numbers.

1092 - Underlines the bottom number.

Pause.

1096 - Prints the total of the odd numbers.

1097 - 1098 The demonstration continues only when a key is pressed. The initial option returns when the 'x' key is pressed or when the 19x19 square has been demonstrated.

1100 - Repeat the demonstration for the next square.

```
1200 IF INKEY$<>"" THEN GO TO 12
00
1201 LET p=1: CLS
1205 LET p=p+1
1220 INK 3
1225 FOR j=1 TO p
1230 FOR k=1 TO p-1
1235 PRINT AT j,k+10;"■"
1240 NEXT k
1242 NEXT j
1245 INK 7: PRINT AT 1,0;p-1;"x"
;p;" = "; FLASH 1;"?";
1246 IF INKEY$="" THEN GO TO 124
5
1247 IF INKEY$<>"" THEN GO TO 12
47
1250 FLASH 0: PRINT CHR$ 8;p*(p-
1)
1255 IF INKEY$="" THEN GO TO 125
5
1256 IF INKEY$<>"" THEN GO TO 12
56
1260 INK 4
1261 FOR j=1 TO p
1262 FOR k=j TO p-1
1263 PRINT AT j,k+10;"■"
1264 NEXT k
1265 NEXT j
1266 PLOT 84,171: DRAW 8*p,-8*p
1268 INK 7
1270 PRINT AT 3,0;p*(p-1);"A2 =
"; FLASH 1;"?";
1275 IF INKEY$="" THEN GO TO 127
5
1276 IF INKEY$<>"" THEN GO TO 12
76
1280 FLASH 0: PRINT CHR$ 8;p*(p-
1)/2
1285 IF INKEY$="" THEN GO TO 128
5
1286 IF INKEY$<>"" THEN GO TO 12
85
```

1200 - 1399 Demonstration of n(n+1)/2 = t_n.

1220 - 1242 Prints a rectangular arrangement of magenta dots n by (n+1). Initially n = 1.

1245 - Prints the values 'n x (n+1) = ?' for the current rectangle. The question mark flashes.

1246 - 1247 Pause until a key is pressed and released.

1250 - Overprints '?' with the value of n(n+1).
CHR$8 = backspace.

Pause.

1260 - 1265 Overprints half the rectangular array of dots in green.

1266 - Draws the diagonal line to divide the two halves.

1268 - 1270 Prints, in white, the value of n(n+1), followed by ' ÷ 2 = ?' The question mark flashes.

Pause.

1280 - Overprints '?' with the value of n(n+1)/2.

Pause.

```
1290 OVER 1
1291 FOR j=1 TO p
1292 FOR k=j TO p-1
1293 PRINT AT j,k+10;"*"
1294 NEXT k
1295 NEXT j
1296 PLOT 84,171: DRAW 8*p,-8*p
1305 OVER 0: INK 3
1310 FOR j=p-1 TO 1 STEP -1
1320 FOR k=1 TO j-1
1330 PRINT AT j,j-k+10;" ";AT j,
INT ((p+j+1)/2)-k+10;"*" AND INT
 ((p-j)/2)*2<>p-j;"*" AND INT ((
p-j)/2)*2=p-j
1340 NEXT k
1350 NEXT j
1360 IF INKEY$="" THEN GO TO 136
0
1370 IF INKEY$="z" THEN GO SUB 1
500
1380 IF INKEY$="x" THEN PRINT AT
 p+2,0: GO TO 1396
1385 CLS : OVER 0
1395 IF p<18 THEN GO TO 1205
1396 INK 7
1397 IF INKEY$<>"" THEN GO TO 13
97
1398 PRINT p;"x";p+1;" = ";p*(p+
1);TAB 15;p*(p+1);"A2 = ";p*(p+1
)/2: LET p=p+1: IF INKEY$="x" TH
EN CLS : GO TO 5
1399 GO TO 1398
```

1290 - 1296 Prints the dots and line as in 1261 - 1266 using the 'exclusive or' command, thus erasing them. This leaves a right-angled triangle of magenta dots, of size n(n+1)/2.

1305 - Normal printing mode, colour magenta.

1310 - 1350 Rearranges the triangle of dots into an isosceles triangle, row by row.

1360 - 1380 Pressing and releasing a key continues the demonstration.
Pressing the 'x' key results in the sequence being printed without the dot demonstration.
Pressing the 'z' key executes subroutine 1500.

1395 - If p<18 the demonstration is repeated for the next triangle number. The screen is filled when p = 18.

1397 - The sequence below is printed when the 'x' key is released.

1398 - 1399 Prints the sequence without the demonstration.
For example, if p = 8,'8 x 9 = 72   72 ÷ 2 = 36' is printed.
Pressing the 'x' key returns to the initial option display.

```
1400 CLS : LET i=0
1401 IF INKEY$<>"" THEN GO TO 14
01
1402 LET i=i+1
1403 FOR j=1 TO i
1405 FOR k=1 TO j
1410 INK 5: PRINT AT j,9-INT (j/
2)+k;"■" AND INT (j/2)*2=j;"■ " A
ND INT (j/2)*2<>j
1420 NEXT k
1430 NEXT j
1435 IF INKEY$="" THEN GO TO 143
5
1436 IF INKEY$<>"" THEN GO TO 14
36
1437 FOR j=1 TO i: PLOT 82-4*j,1
69-8*j: DRAW 8*j,0: NEXT j
1438 IF INKEY$="" THEN GO TO 143
8
1439 IF INKEY$<>"" THEN GO TO 14
39
1440 INK 7: FOR j=1 TO i: PRINT
AT j,11+i/2;j: NEXT j
1441 IF INKEY$="" THEN GO TO 144
1
1442 IF INKEY$<>"" THEN GO TO 14
42
1443 FOR j=1 TO i: PRINT AT j-1,
14+i/2;"+" AND j<>1: NEXT j
1445 IF i<>1 THEN GO TO 1460
1450 IF INKEY$="" THEN GO TO 145
0
1451 IF INKEY$<>"" THEN GO TO 14
51
1455 CLS : GO TO 1401
1460 INK 7: PLOT 85+4*i,167-8*i:
 DRAW 24,0: FLASH 1: PRINT AT i+
2,11+i/2;"?";
1465 IF INKEY$="" THEN GO TO 146
5
1466 IF INKEY$<>"" THEN GO TO 14
66
1470 FLASH 0: PRINT CHR$ 8;i*(i+
1)/2
```

1400 - 1495 Demonstrates $\sum_{i=1}^{n} i = t_n$.

1401 - The demonstration only continues when the option key (b) is released.

1402 - i increases by 1 at each stage of the demonstration.

1403 - 1430 Prints the i^{th} triangle number as an arrangement of cyan dots.

1435 - 1436 Pause until a key is pressed and released.

1437 - Draws horizontal lines to divide the triangle into rows of dots.

Pause.

1440 - Prints the number of dots in each row, at the side of the triangle.

Pause.

1443 - Prints a column of +'s beside the numbers.

1445 - 1455 If the first triangle number (1) is being demonstrated, then pause until a key is pressed and released after which return to line 1401 to demonstrate the next triangle number. For subsequent triangle numbers continue with line 1460.

1460 - Draws a line under the column of numbers, with a flashing question mark below.

Pause.

1470 - Overprints '?' with the current triangle number.

```
1480 IF INKEY$="" THEN GO TO 148
0
1485 CLS
1490 IF INKEY$="x" OR i=19 THEN
GO TO 5
1491 IF INKEY$<>"" THEN GO TO 14
91
1495 GO TO 1402
1500 IF INKEY$<>"" THEN GO TO 15
00
1505 INK 3: FOR j=1 TO p-1: PLOT
 80+(p+1-j)*4,162-8*j: DRAW 8*j+
2,0: NEXT j
1510 IF INKEY$="" THEN GO TO 151
0
1520 IF INKEY$<>"" THEN GO TO 15
20
1530 INK 6: FOR j=1 TO p-1: PRIN
T AT j+1,p+10;j: NEXT j
1540 IF INKEY$="" THEN GO TO 154
0
1550 IF INKEY$<>"" THEN GO TO 15
50
1560 FOR j=1 TO p-1: PRINT AT j,
p+12;"+" AND j<>1: NEXT j
1570 PLOT 80+8*p,167-8*p: DRAW 2
4,0
1580 PRINT FLASH 1;AT p+2,11+p;"
?";
1590 IF INKEY$="" THEN GO TO 159
0
1600 IF INKEY$<>"" THEN GO TO 16
00
1610 PRINT FLASH 0;AT p+2,11+p;"
 ";AT p+2,10+p;p*(p-1)/2
1620 IF INKEY$="" THEN GO TO 162
0
1630 RETURN
```

1480 - 1495 When the 'x' key is pressed or the screen is filled (after the 19th triangle) return to the initial option display.Any other key continues with the demonstration of the next triangle.

1500 - 1630 Demonstration of $\sum_{i=1}^{n} i = t_n$ after $n(n+1)/2 = t_n$ has been demonstrated.

1505 - Draws horizontal lines to divide the triangle into rows of dots.

1510 - 1520 Pause until a key is pressed and released.

1530 - Prints the number of dots in each horizontal row.

Pause.

1560 - Prints a column of +'s beside the numbers.

1570 - Draws a line below the column of numbers.

1580 - Draws a flashing question mark below the line.

Pause.

1610 - Overprints '?' with the current triangle number.

1620 - 1630 RETURN when a key is pressed.

Appendix C - The listing for the program 'fibonacci'.

```
2000 REM Fibonacci
2001 PAPER 0: BORDER 0: INK 7: C
LS
2002 GO SUB 3000
2005 PAUSE 0
2010 FOR k=0 TO 23: READ a: POKE
 USR "a"+k,a: NEXT k
2020 DATA 0,8,0,0,127,0,0,8,40,4
0,40,56,124,124,124,56,124,254,2
54,254,254,254,124,0
```

2001 - Sets the whole screen black.
2002 - SUB 3000 sets up array r$.
2005 - Pause until a key is pressed.
2010 - 2020 Sets up the user-defined graphics from the DATA in line 2020. "a" = ÷ "b" = rabbit head, "c" = rabbit body. It was decided to use ÷ for division in the printout as the pupils for whom this program was written were not familiar with the use of /.

```
2021 CLS
2022 LET w=0: LET y=1
2025 LET c=0: GO SUB 2200
```

2021 - Clears the screen.
2022 - w and y are used to store successive Fibonacci numbers.
2025 - SUB 2200 prints a green horizontal line (to represent grass) for the (c+1)th month.

```
2029 GO SUB 2500
2030 INK 7: PRINT AT 0,0;"●";AT
1,0;"●"
2035 GO SUB 2500
2037 GO SUB 2270
2038 GO SUB 2500
```

2029 - Pause until a key is pressed and released.
2030 - Prints the first rabbit of the first month, in white.
2035 - Pause.
2037 - SUB 2270 prints '1' at the right of the first row.
2038 - Pause.

```
2040 FOR c=1 TO 7
2044 GO SUB 2200
2045 GO SUB 2500
2046 INK 6
2050 FOR j=1 TO 21
2060 IF r$(c,j)<>"y" THEN GO TO
2080
2070 PRINT AT 2*c,j-1;"▲";AT 2*c
+1,j-1;"■"
2080 NEXT j
2082 GO SUB 2500
2085 INK 7
2090 FOR j=1 TO 21
2100 IF r$(c+1,j)<>"w" THEN GO T
O 2120
2110 PRINT AT 2*c,j-1;"▲";AT 2*c
+1,j-1;"■"
2120 NEXT j
2125 GO SUB 2500
2130 INK 6
2140 FOR j=1 TO 21
2150 IF r$(c,j)<>"w" THEN GO TO
2170
2160 PRINT AT 2*c,j-1;"▲";AT 2*c
+1,j-1;"■"
2170 NEXT j
2175 GO SUB 2500
2177 LET z=y
2178 LET y=y+w
2179 LET w=z
2180 GO SUB 2250
2181 GO SUB 2500
2185 NEXT c
```

2040 - 2185 One stage of the demonstration.

2044 - SUB 2200 draws a green horizontal line for the (c+1)th stage).

2045 - Pause.

2046 - Sets colour of printing to yellow.

2050 - 2080 Prints a yellow rabbit wherever "y" appears in the string r$(c), i.e. mature rabbits from the previous month. Each rabbit occupies two character squares, column (j-1) on lines 2c and (2c+1).

2082 - Pause.

2085 - Sets the colour of printing to white.

2090 - 2120 Prints a white rabbit where "w" appears in the string r$(c+1), i.e. the current month.

2125 - Pause.

2130 - Sets colour of printing to yellow.

2140 - 2170 Prints a yellow rabbit where "w" appears in the string r$(c), i.e. the rabbits which were born in the previous month and mature in the current month.

2175 - Pause.

2177 - 2179 Sets y to the number of yellow rabbits and w to the number of white rabbits.

2180 - Prints the sequence at the right of the row.

2181 - Pause.

```
2190 GO TO 2400
2200 INK 4
2205 FOR j=0 TO 20
2210 PRINT AT 2*c+1,j;"_"
2215 NEXT j
2220 RETURN
2250 INK 7
2260 IF y<>1 THEN GO TO 2300


2265 IF w<>0 THEN INK 5
2270 PRINT AT 2*c+1,29;"1"
2280 RETURN



2300 PRINT AT 2*c+1,21;y-w;" + "
;
2305 INK 6: PRINT w;: INK 7: PRI
NT " = ";y
2310 RETURN
```

2190 - Jump to by-pass the subroutines.

2200 - 2220 Draws a green horizontal line 21 characters long.

2250 - Sets colour of printing to white.

2260 - Jumps to line 2300 from the 3rd stage onwards.

2265 - 2270 Prints "1" at the right hand side of the screen, white at the first stage (w=0), yellow at the second (w=1).

2280 - RETURN executed only for stages 1 and 2.

2300 - 2305 Prints at the right of the screen, in white, the value of (y-w), the number of white rabbits displayed, followed by "+", then the value, printed in yellow, of w, the number of yellow rabbits. The line is completed with "=" followed by the value of y, the total number of rabbits.

2310 - End of subroutine 2250.

```
2400 PRINT "Continue sequence...
..▣          Repeat demonstration
.....▣       Print sequence......
▣            Golden ratio......▣"
2401 IF INKEY$="c" THEN GO SUB 2
700: GO TO 2409
```

2400 - Prints the options.

2401 - Option 'c' - SUB 2700 clears the bottom of the screen. Line 2409 continues printing the sequence without the rabbit demonstration.

```
2402 IF INKEY$="r" THEN GO TO 20
01
```

2402 - Option 'r' - returns to the start of the program, to clear the screen and repeat the rabbit demonstration.

```
2403 IF INKEY$="g" THEN CLS : GO
 TO 2600
```

2403 - Option 'g' clears the screen and prints the ratios of successive pairs of numbers in the Fibonacci sequence.

```
2404 IF INKEY$<>"p" THEN GO TO 2
401
```

2404 - Jumps to 2401 if 'p' is not pressed. Only one of the four keys will produce an exit from the loop.

```
2405 CLS : LET w=1: LET y=1
2406 PRINT w'y
2409 PRINT w;" + ";y;" = ";w+y
```

2405 - Option 'p' - sets w and y to 1. Clears the screen and prints 1 on the first two lines of the screen (line 2406).

2409 - Continues printing the values of w '+' y '=' (w+y), where w and y are consecutive numbers in the sequence, for options 'p' and 'c'.

```
2410 LET z=y: LET y=y+w: LET w=z
2415 IF INKEY$="x" THEN CLS : GO
 TO 2400
2420 GO TO 2409
2500 IF INKEY$="" THEN GO TO 250
0
2510 IF INKEY$<>"" THEN GO TO 25
10
2520 RETURN
```

2410 - Sets w and y as the next pair of consecutive numbers.

2415 - Clears screen and displays options when the 'x' key is pressed.

2420 - Continues the sequence if 'x' has not been pressed.

2500 - 2520 Subroutine to pause until a key is pressed and released.

```
2600 LET w=1: LET y=1
2610 PRINT w;" ÷ ";y;" = ";w/y
2620 LET x=w: LET w=w+y: LET y=x
2630 IF INKEY$="x" THEN CLS : GO
 TO 2400
2640 GO TO 2610
```

2600 - Option 'g' - sets w and y to 1.

2610 - Prints the values of w '÷' y '=' (w/y).

2620 - Sets w and y to the next pair of numbers in the sequence.

2630 - Jumps to the option display if the 'x' key is pressed.

2640 - Continues the sequence of ratios if 'x' is not pressed.

```
2700 PRINT AT 15,0
2710 FOR j=1 TO 32
2720 PRINT "     ";
2730 NEXT j
2740 PRINT AT 15,0
2750 RETURN
3000 DIM r$(8,21)
3010 LET r$(1)="wsssssssssssssss
sssss"
3020 LET r$(2)="ysssssssssssssss
sssss"
3030 LET r$(3)="ysssssssssssswss
sssss"
3040 LET r$(4)="ysssssssswssssyss
sssss"
3050 LET r$(5)="ysssswssyssssyss
sswss"
3060 LET r$(6)="ysswsyssysswsyss
wsyss"
3070 LET r$(7)="yswysyswyswysysw
ysysw"
3080 LET r$(8)="ywyywywyywyywywy
ywywy"
3090 RETURN
```

2700 - 2750 Subroutine to clear the option display from the bottom of the screen and set printing to line 15 of the screen.

3000 - 3090 Subroutine to set up the string array r$ which contains the information for printing the rabbits.
r$(k) is the k^{th} row.
y indicates where a yellow rabbit is to be printed.
w indicates where a white rabbit is to be printed.
s indicates a space.

The printout below was obtained by :- FOR k = 1 TO 8: PRINT r$(k): NEXT k.
It shows the final arrangement of white and yellow rabbits after all 8 months have been demonstrated.
(A 9th stage would not have fitted on one line.)

```
wssssssssssssssssssss
yssssssssssssssssssss
ysssssssssssswsssssss
ysssssssswssssysssssss
ysssswssyssssyssssswss
ysswsyssysswsyssswsyss
yswysyswyswysyswysysw
ywyywywyywyywywyywywy
```

Appendix D - The listing for the program 'prime'.

```
   1 GO TO 11
   3 BRIGHT 0: BORDER 7: PAPER 7
: INK 0: CLS
   4 PRINT "This program can giv
e any Prime number, up to the 10
00th.","Press N to list the prim
es.","Press E for the Sieve of
      Eratosthenes.""Press P if
 you wish to print    a specific
 prime","Press T to test a prime
.        Press S to stop."
   5 IF INKEY$="e" THEN GO TO 19
0
   6 IF INKEY$="s" THEN GO TO 45
0
   7 IF INKEY$="n" THEN CLS : GO
 TO 300
   8 IF INKEY$="p" THEN GO TO 18
0
   9 IF INKEY$="t" THEN CLS : GO
 TO 360
  10 GO TO 5
```

1 - Executed only on the initial RUN, to set up the array p with the prime numbers.

3 - Sets the background white, printing black. BRIGHT 0 is a lower brilliance than BRIGHT 1.

4 - Prints the initial option display.

5 - 10 Jumps to the appropriate part of the program when the 'e', 's', 'n', 'p' or 't' key is pressed. No other key will produce a response.

```
  11 REM This part of the progra
m, to line 170 generates the pri
mes
  14 DIM p(1000)

  15 LET p(1)=2: LET p(2)=3
  16 LET m=3
  20 LET j=0
  21 LET t=0

  25 LET j=j+1
  30 LET p=6*j-1
  40 LET s=INT SQR p
  50 FOR k=1 TO m-1
  60 IF INT (p/p(k))=p/p(k) THEN
 GO TO 100
  70 IF p(k)>s THEN GO TO 90
  80 NEXT k

  90 LET p(m)=p: LET m=m+1
  95 IF m=1001 THEN GO TO 3
 100 LET p=6*j+1
 105 LET s=INT SQR p
 110 FOR k=1 TO m-1
 120 IF INT (p/p(k))=p/p(k) THEN
 GO TO 25
 130 IF p(k)>s THEN GO TO 160
 140 NEXT k
 160 LET p(m)=p: LET m=m+1
 165 IF m=1001 THEN GO TO 3
 170 GO TO 25
```

11 - 170 Executed only when setting array p with the first 1000 prime numbers.

14 - Sets up a one-dimensional array p of dimension 1000. Each member is initially set to 0.

15 - The first two prime numbers.

16 - m is a counter for the array p.

20 - 21 Sets j and t initially to 0.

25 - 140 Uses the fact that all primes above 3 are of the form $6j \pm 1$ where j is a natural number to avoid considering all the natural numbers, or all the odd numbers, when finding the primes.

p is the number being tested whether it is prime or not.

25 - Increases j by 1.

30 - Considers p as $6j - 1$.

40 - $s = \sqrt{p}$ rounded down to an integer.

50 - 80 Tests if p is divisible by the primes p_1 to p_{m-1} as far as p_K where $p_K > \sqrt{p}$. If p has no prime factor less than $\sqrt{p}$, it will have no prime factor greater than $\sqrt{p}$.
GOTO 100 if p is composite. GOTO 90 if p is prime.

90 - Sets the next member of array p with the value of p, and increases m by 1.

95 - If m = 1001 then the array p is full, return to line 3.

100 - 140 As the above section, considering $p = 6j + 1$.
GOTO 25 if p is not prime.
GOTO 160 if p is prime.

160 - Sets the next member of array p with the value of p, and increases m by 1.

165 - As 95.

```
 180 CLS
 181 IF INKEY$<>"" THEN GO TO 18
1
 182 FOR m=1 TO 1000: PRINT p(m)
 183 IF INKEY$="X" THEN GO TO 3
 185 NEXT m
 186 STOP
```

180 - 186 Executed when initial option 'p' is chosen.
180 - Clears the screen.
181 - The program only continues when the key is released.
182 - 186 Prints all the primes up to the 1000th, unless the 'x' key is pressed, which returns to the initial option display.

```
 190 CLS : BRIGHT 1
 191 IF INKEY$<>"" THEN GO TO 19
1
 192 PRINT "1"
 196 PAPER 0: INK 7: PRINT AT 0,
3;"2";AT 0,6;"3": PAPER 7: INK 0
 198 LET t=0: LET m=3: LET a=10:
 LET b=3: LET c=200
 200 FOR w=p(m-1)+1 TO p(m)-1
 210 BRIGHT (INT (w/2)*2=w)
 220 PRINT AT INT ((w-1)/a)-20*t
,b*(w-(INT ((w-1)/a))*a-1);w
 225 IF INT (w/c)*c=w THEN GO SU
B 275
 230 NEXT w
 240 PAPER 0: INK 7:
 250 PRINT AT INT ((p(m)-1)/a)-2
0*t,b*(p(m)-(INT (p(m)/a))*a-1);
p(m);
 260 PAPER 7: INK 0
 265 IF INKEY$="x" THEN GO TO 3
 270 LET m=m+1: GO TO 200
```

190 - 286 Prints the Sieve of Eratosthenes when the 'e' option is chosen.

190 - clears screen and sets brilliance to BRIGHT 1.
191 - Pause until the key is released.
192 - Prints the number '1' in the sieve.
196 - Prints '2' and '3' in the sieve, white on black.
198 - m = the number of the current prime, a = the number of numbers per line, b = the number of spaces per number, c = the number of numbers to fill the screen.
200 - w takes the values of the numbers between the last prime printed, and the next to be printed.
210 - BRIGHT 1 if w is even, BRIGHT 0 if w is odd.
220 - Prints the value of w in the appropriate place.
240 - Black background, white print.
250 - Prints the next prime in the appropriate position.
260 - Sets white background, black print.
265 - Pressing the 'x' key returns to the initial options.
270 - Increases m by 1 and continues the sieve.

```
 275 IF INKEY$="" THEN GO TO 275
 276 IF INKEY$="x" THEN GO TO 3
 280 IF INKEY$<>"" THEN GO TO 28
0
 281 LET t=t+1
 282 IF w<>1000 THEN GO TO 284
 283 LET a=5: LET b=4: LET c=100
: LET t=10
 284 IF w=10000 THEN LET b=5
 285 BRIGHT 0: CLS
 286 RETURN
```

Subroutine 275 - 286 is called from line 225 when the screen is full.

275 - 280 Pause until a key is pressed and released. Pressing the 'x' key returns to the initial option display.

281 - t = the number of multiples of 200 (c = 200) which have been printed.

283 - When 1000 is reached each number has 4 digits (b = 4), 5 per line (a = 5), 100 per screen (c = 100), t = the number of multiples of 100 which have been printed.

284 - When 10000 is reached each number has 5 digits (b = 5).

```
 300 IF INKEY$<>"" THEN GO TO 30
0
 305 INPUT ;"Input N will give t
he Nth prime ";m
 310 IF m>1000 THEN INPUT "N mus
t not be more than 1000.   Try a
gain.";m
 320 PRINT AT 20,0;p(m);" is the
 ";m;
 321 IF m>=4 AND m<=20 THEN PRIN
T "th prime": GO TO 330
 322 LET mm=m-INT (m/10)*10
 323 PRINT "th" AND mm=0;"st" AN
D mm=1;"nd" AND mm=2;"rd" AND mm
=3;"th" AND mm>=4;" prime"
 330 POKE 23659,0: PRINT AT 22,0
;"Do you want another? y/n.": PO
KE 23659,2
 340 IF INKEY$="y" THEN GO TO 30
0
 345 IF INKEY$<>"n" THEN GO TO 3
40
 346 IF INKEY$<>"" THEN GO TO 34
6
 350 GO TO 3
```

300 - 350 Executed when the 'n' key is pressed after the initial option display.

300 - The program continues only when the key is released.

305 - Prints an input message for which prime is required.

310 - Error message if input is greater than 1000.

320 - Prints the mth prime followed by "is the " and the value of m.

321 - 323 Prints "th" if 4 ≤ m ≤ 20, "st" if the unit digit of m, mm, is 1, "nd" if the unit digit is 2, "rd" if the unit digit is 3, "th" otherwise, followed by "prime".

330 - POKE 23659,0 allows printing on the bottom two lines of the screen. POKE 23659,2 returns to normal.

340 - 350 Pressing the 'n' key returns to the initial option display. Pressing the 'y' key jumps to 300.

```
 360 IF INKEY$<>"" THEN GO TO 36
0
 365 INPUT "Input the number to
be tested "'m
 366 IF m>(p(1000))↑2 THEN PRINT
 AT 20,0;m;" is too large ": GO
TO 400
 370 IF m<=0 OR m=1 OR (INT m)<>
m THEN PRINT AT 20,0;m;" is not
a prime.": GO TO 400
 375 FOR k=1 TO m
 380 IF m/p(k)=INT (m/p(k)) THEN
 PRINT AT 20,0;m;" is not a prim
e.": GO TO 396
 385 IF p(k)>SQR m THEN GO TO 39
5
 390 NEXT k
 395 PRINT AT 20,0;m;" is a prim
e."
 400 POKE 23659,0: PRINT AT 22,0
;"Do you wish to test another?
  y/n": POKE 23659,2
 410 IF INKEY$="y" THEN GO TO 36
5
 415 IF INKEY$<>"n" THEN GO TO 4
00
 416 IF INKEY$<>"" THEN GO TO 41
6
 420 GO TO 3
 450 STOP
```

360 - 420 Executed when the 't' key is pressed after the initial option display.

- 360 - Pause until the key is released.
- 365 - Prints a message requesting input of the number to be tested.
- 366 - Error message if the number is larger than the square of the 1000th prime. Then jumps to line 400.
- 370 - Prints the value of m followed by " is not a prime" if m ≤ 0, m = 1 or if m is not an integer.
- 375 - 390 Tests if m is divisible by each of the primes.
  - 380 - If m is divible by a prime the message 'm is not a prime' is printed, then jump to line 396 (400).
  - 385 - It is only necessary to test for divisibilty by primes which are less than $\sqrt{m}$.
  - 395 - Prints when m is a prime.
- 400 - Gives the option of inputting another number for testing.

- 410 - 420 Jump to 365 if 'yes' response, jump to the initial option display if 'no'.

450 - Executed if the 's' key is pressed after the initial option display.

Appendix E - The listing for the program 'loci'.

```
   5 BORDER 0: PAPER 0: INK 7: C
LS
  10 OVER 0: CLS : PRINT "This p
rogram demonstrates the   locus 
of a point P under certaincondit
ions"''"       P lies on AB where
          A - A and B are two fixe
d pointsB - A and B lie on two  
           perpendicular lines 
          C - A lies on a circle a
nd B is       fixed   "
  12 PRINT ''"D - The ratio of t
he distances        from a fixed p
oint and from       a fixed line i
s constant"
  14 PRINT ''"E - Relation betwe
en angle PAB        and angle PBA,
 where A and B      are fixed poin
ts"
  20 IF INKEY$="e" THEN GO TO 40
00
  25 IF INKEY$="d" THEN GO TO 30
00
  30 IF INKEY$="c" THEN GO TO 20
00
  35 IF INKEY$="b" THEN GO TO 10
00
  40 IF INKEY$<>"a" THEN GO TO 2
0
  45 IF INKEY$<>"" THEN GO TO 45
  50 CLS : PRINT "A and B are fi
xed points"'''"A - P moves such 
that                AP:BP = m:n"'
''"B - P moves such that AP + PB
 is      constant"
  55 IF INKEY$="b" THEN GO TO 60
0
  60 IF INKEY$<>"a" THEN GO TO 5
5
  61 IF INKEY$<>"" THEN GO TO 61
```

5 - Sets background colour to black, printing to white.

10 - Prints the option display shown in figure 3.1.

20 - 45 Jumps to line 4000 if option E is chosen, 3000 if option D, 2000 if option C, 1000 if option B.
The program continues with line 50 if option A is chosen.
No other key (except 'BREAK') produces a response.

50 - Prints the option display shown in figure 3.2.

55 - 61 Jumps to line 600 if option B is chosen.
Continues with line 65 if option A is chosen.

```
  65 CLS : PRINT "Locus of a poi
nt P such that "''"        AP:BP =
 m:n "''"where A and B are fixed
";AT 10,0;"Input m"
  70 INPUT m
  80 IF m<=0 THEN PRINT AT 10,0;
"m must be positive, try again":
 GO TO 70
  90 PRINT AT 10,0;"Input n
                  "
 100 INPUT n
 105 IF INKEY$<>"" THEN GO TO 10
5
 110 IF n<=0 THEN PRINT AT 10,0;
"n must be positive, try again":
 GO TO 100
 120 PRINT AT 2,14;m;":";n;AT 10
,0;"Press any key to show the lo
cus"
 130 GO SUB 900: CLS
 140 IF m=n THEN GO TO 470
 150 LET ab=ABS (m-n)
 160 LET ay=m: IF m<n THEN LET a
y=-m
 170 LET ax=m*ab/(m+n)
 180 IF m>n THEN LET a=0
 190 IF m<n THEN LET b=255
```

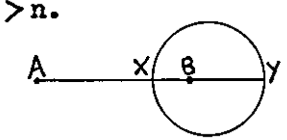
65 - 595 Locus of P such that AP/PB is constant.
65 - Prints the display shown in figure 3.3.

70 - awaits the input of m.
80 - Continues only when a positive number has been input. An error message is printed if $m \leqslant 0$.
90 - 100 Requires n to be input.

105 - 110 Continues only if n is positive. Error message if $n \leqslant 0$.

120 - Prints the values of m and n in place of the letters m and n displayed on the screen.
130 - SUB 900 - pause until a key is pressed and released.
140 - If m=n the locus is the mediator of AB, drawn at line 470.
150 - 190 The locus is a crcle. Two situations are possible.

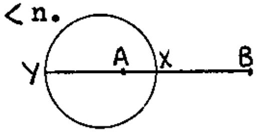
1. $m > n$.



AY:BY = m:n   AX:BX = m:n

Sets $AB = |m-n|$, $AY = m$, $AX = \frac{m}{m+n}AB$

2. $m < n$.



As above but AY = -m.

180 - 190 If $m > n$ then A is on the left of the screen, if $m < n$ then B is on the right of the screen.

```
 200 LET xs=INT (250/m): IF m<n
THEN LET xs=INT (250/n)
 210 LET r=ABS (ay-ax)/2
 220 LET ys=INT (87/r)
 230 IF xs>ys THEN LET sf=ys: GO
 TO 250
 240 LET sf=xs
 250 LET r=r*sf
 255 IF m<n THEN LET a=255-ab*sf
 260 IF m>n THEN LET b=ab*sf
 270 PRINT AT 10,a/8;"A";TAB b/8
-1;"B": PLOT a,88: PLOT b,88
 280 GO SUB 900
 290 LET c=sf*(ax+ay)/2+a
```

200 - 240 Evaluates the scale (pixels per unit) to accommodate the diagram on the screen, horizontally and vertically. If $m>n$ then xs = 250/AY, if $m<n$ then xs = 250/YB.

210 - r = the radius of the circle.

220 - ys = the y scale.

230 - sf (scale factor) is set to the smaller of ys and xs.

255 - When $m<n$ the position of A on the screen is set.

260 - When $m>n$ the position of B on the screen is set.

270 - Plots and labels A and B on the screen.

280 - Pause until a key is pressed and released.

290 - c = the x coordinate of the centre of the circle.

```
 300 FOR j=-60 TO 60
 310 LET x=c+r*COS (PI/60*j): LE
T y=-r*SIN (PI/60*j)
 330 OVER 0: PLOT a,88: OVER 1:
DRAW x-a,y
 340 OVER 0: PLOT b,88: OVER 1:
DRAW x-b,y
 350 IF INKEY$="m" THEN GO TO 35
0
 360 OVER 0: PLOT a,88: OVER 1:
DRAW x-a,y
 370 OVER 0: PLOT b,88: OVER 1:
DRAW x-b,y
 380 OVER 0
 390 PLOT x,88+y
 395 IF INKEY$="n" THEN GO TO 39
5
 400 NEXT j
 410 GO SUB 900
 420 PLOT c,88
 430 GO SUB 900
 440 CIRCLE c,88,r
 450 LET rep=270: GO SUB 950
 460 GO TO 10
```

300 - 400 Draws the locus (a circle) with θ from -180° to 180°, stepping 3° ($\theta = 3j$).

310 - Evaluates the x and y coordinates of the current position of P, where $x = a + r\cos\theta$ and $y = -r\sin\theta$.

330 - Plots A and draws a line from A to P.

340 - Plots B and draws a line from B to P.

350 - The demonstration halts while the 'm' key is held down.

360 - Plots A and erases the line AP.

370 - Plots B and erases the line BP.

390 - Plots the current position of P.
395 - The demonstration halts while the 'n' key is held down.

410 - Pause.

420 - Plots the centre of the circle.

440 - Draws the complete circle.

450 - Sub 950 jumps to line 'rep' when the 'r' key is pressed, to repeat the demonstration. Pressing any other key returns to line 10, which displays the initial options.

```
 470 LET a=80: LET b=160
 480 PRINT AT 10,9;"A";TAB 20;"B
": PLOT 80,88: PLOT 160,88
 485 GO SUB 900
 490 FOR y=-88 TO 87
 510 OVER 0: PLOT a,88: OVER 1:
DRAW 40,y
 520 OVER 0: PLOT b,88: OVER 1:
DRAW -40,y
 530 IF INKEY$="m" THEN GO TO 53
0
 540 OVER 0: PLOT a,88: OVER 1:
DRAW 40,y
 550 OVER 0: PLOT b,88: OVER 1:
DRAW -40,y
 560 OVER 0
 570 PLOT 120,88+y
 575 IF INKEY$="n" THEN GO TO 57
5
 580 NEXT y
 590 LET rep=470: GO SUB 950
 595 GO TO 10
 600 CLS : PRINT "Locus of a poi
nt P such that "''"    AB:AP+PB =
 m:n "''"where A and B are fixed
";AT 10,0;"Input m"
 605 INPUT m
 610 IF m<=0 THEN PRINT AT 10,0;
"m must be positive, try again":
 GO TO 605
 615 PRINT AT 10,0;"Input n
"
 620 INPUT n
 625 IF n<=0 THEN PRINT AT 10,0;
"n must be positive.
 Try again": GO TO 620
 630 IF n<m THEN PRINT AT 10,0;"
n cannot be less than m.
Try again": GO TO 620
```

470 - 480 Plots and labels A and B.

485 - Pause.

490 - 580 Draws the mediator of AB.

510 - Plots A and draws AP.

520 - Plots B and draws BP.

530 - Halts the demonstration while the 'm' key is held down.

540 - Plots A and erases AP.

550 - Plots B and erases BP.

570 - Plots P.

575 - Halts the demonstration while the 'n' key is held down.

590 - SUB 950 pauses until a key is pressed and released. If the 'r' key is pressed the demonstration is repeated.

600 - 740 Locus of P where AP + PB is constant.

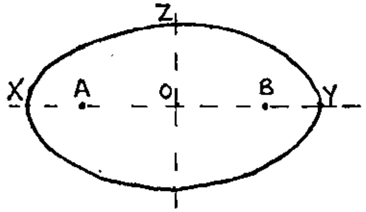
600 - Prints the display requesting input of the ratio AB:AP+PB.

605 - 630 Inputs m and n where AB:AP+PB = m:n. Error messages if $n \leq 0$, $m \leq 0$ or if $n < m$.

```
 635 LET ys=250: LET xs=250/2/n:
 IF m<>n THEN LET ys=87/SQR (n*n
-m*m)
 640 IF xs<ys THEN LET sf=xs: GO
 TO 650
 645 LET sf=ys
 650 LET a=n*sf: LET b=SQR (n*n-
m*m)*sf: LET ab=2*m*sf: LET ax=1
27-ab/2: LET bx=ax+ab
 655 PRINT AT 2,14;m;":";n;AT 10
,0;"Press any key to show the lo
cus
   "
 660 GO SUB 900
 665 CLS : PRINT AT 10,ax/8;"A";
TAB bx/8-1;"B": PLOT ax,88: PLOT
 bx,88
 670 GO SUB 900
 675 IF m=n THEN GO TO 800
```

```
 685 FOR t=0 TO 355 STEP 5
 690 LET x=a*COS (PI/180*t): LET
 y=b*SIN (PI/180*t)
 695 PLOT ax,88: OVER 1: DRAW ab
/2+x,y
 700 OVER 0: PLOT bx,88: OVER 1:
 DRAW x-ab/2,y
 705 IF INKEY$="m" THEN GO TO 70
5
 710 OVER 0: PLOT ax,88: OVER 1:
 DRAW ab/2+x,y
 715 OVER 0: PLOT bx,88: OVER 1:
 DRAW x-ab/2,y
 720 IF INKEY$="n" THEN GO TO 72
0
 725 OVER 0: PLOT 127+x,88+y
 730 NEXT t
 735 LET rep=660: GO SUB 950
 740 GO TO 10
```

The locus is an ellipse.



If AB = 2m and XY = 2n, then OY = n.

Also AZ = n, and so OZ = $\sqrt{n^2 - m^2}$.

635 - xs is the horizontal scale so that XY = 250 pixels. ys is the vertical scale so that OZ = 87 pixels.

640 - 645 sf = the smaller of xs and ys.

650 - a and b are such that the equation of the ellipse is $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$. ax and bx are the x coordinates of the foci.

655 - Prints the values of m and n in place of the letters m and n on the screen.

660 - Pause until a key is pressed and released.

665 - Plots and labels A and B.

675 - If m = n then GOTO 800. (The locus of P is AB.)

685 - 730 Draws $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ using x = a cost, y = b sint for t = 0° to 360° in steps of 5°.

690 - Evaluates the coordinates of P.

695 - Plots A and draws AP.

700 - Plots B and draws BP.

705 - Halts the demonstration while the 'm' key is held down.

710 - Plots A and erases AP.

715 - Plots B and erases BP.

720 - Halts the demonstration while the 'n' key is held down.

725 - Plots the position of P.

735 - Pause until a key is pressed. If 'r' is pressed, then GOTO rep otherwise return to the initial option display.

```
 800 FOR x=bx TO ax STEP -2
 805 OVER 0: PLOT ax,88: OVER 1:
 DRAW ab,0: OVER 0: PLOT bx,88
 810 IF INKEY$="m" THEN GO TO 81
0
 815 PLOT ax,88: OVER 1: DRAW ab
,0: OVER 0: PLOT bx,88
 820 IF INKEY$="n" THEN GO TO 82
0
 825 PLOT x,88
 830 NEXT x
 835 LET rep=660: GO SUB 950
 840 GO TO 10
 900 IF INKEY$="" THEN GO TO 900
 910 IF INKEY$<>"" THEN GO TO 91
0
 920 RETURN
 950 IF INKEY$="" THEN GO TO 950
 960 IF INKEY$="r" THEN CLS : GO
 TO rep
 970 IF INKEY$<>"" THEN GO TO 97
0
 980 RETURN
```

800 - 840 Executed if m = n, i.e. the locus of P is AB.
800 - 830 Plots points along AB, stepping 2 pixels.
805 - Plots A, draws AB, plots B.
810 - Halts the demonstration while the 'm' key is pressed.
815 - Plots A, erases AB, plots B.
820 - Halts the demonstration while the 'n' key is pressed.
825 - Plots the current position of P.
835 - rep = the line number for repeating the demonstration.

900 - 920 Subroutine to PAUSE until a key is pressed and released.

950 - 980 Subroutine to pause until a key is pressed and released. Pressing the 'r' key returns to the line number which starts the repeat of the demonstration. Any other key executes RETURN.

```
1000 IF INKEY$<>"" THEN GO TO 10
00
1005 CLS : PRINT "Locus of a poi
nt which lies      on AB"'"A mov
es on the y-axis"'"B moves on th
e x-axis"'"       AP:PB = m:n"
1010 PRINT AT 13,0;"Input m"
1015 INPUT m
1020 PRINT AT 10,0;"

                    Input n"
1030 INPUT n
1040 IF m=0 AND n=0 THEN PRINT A
T 10,0;"m and n cannot both be 0
": GO TO 1005
1045 IF m<0 AND ABS m>n THEN LET
 m=-m: LET n=-n
1046 IF n<0 AND ABS n>m THEN LET
 m=-m: LET n=-n
```

1000 - 1780 Locus of P where P lies on AB, A lies on the y-axis, B lies on the x-axis.

1005 - Prints the message asking for m to be input, where AP : PB = m : n (see figure 3.22).

1015 - Input m

1020 - Asks for n to be input.

1030 - Input n.

1040 - Error message if m and n are both zero.

1045 - If either m or n is negative, the ratio is expressed so that the numerically larger is positive, e.g. -5:3 → 5:-3.

1046 - If both m and n are negative, the ratio is expressed using positive numbers, e.g. -7:-3 → 7:3.

```
1050 LET ab=86
1055 IF m<0 THEN LET ab=INT (86*
(m+n)/n)
1060 IF n<0 AND m/(m+n)>126/86 T
HEN LET ab=(m+n)/m*126
```

1050 - ab = 86 so that AB occupies half the screen height when it is vertical.

1055 - If m<0, then P divides AB externally. AB = 86(m+n)/n so that PB = 86 pixels when vertical.

1060 - If n<0, then P divides AB externally, but P lies on AB produced. PA = mAB/(m+n). PA is maximum when AB is horizontal.
If mAB/(m+n) >126 pixels then AB = 126(m+n)/m to fit on the screen.

```
1070 PRINT AT 6,13;m;":";n;AT 12
,0;"AB constant ..... A ''"AO +
OB constant ..... B"
1080 IF INKEY$="a" THEN GO TO 15
00
1090 IF INKEY$<>"b" THEN GO TO 1
080
1100 IF INKEY$<>"" THEN GO TO 11
00
1105 PRINT ''"Press any key to s
how the locus": GO SUB 900
1110 CLS : PLOT 128,0: DRAW 0,17
5
1120 PLOT 0,88: DRAW 255,0
1130 GO SUB 900
1140 IF m>n THEN GO TO 1300
1145 LET j=1
1150 OVER 1
```

1070 - Overprints the letters m and n on the screen with their values, and asks for a choice of options.

1080 - Jumps to line 1500 if option A is chosen, i.e. AB is constant.

1090 - 1100 Continues only if the 'b' key is pressed (or the 'a' key at line 1080).

1105 - 1430 Option B.

1105 - Prints a message to press any key to show the locus.

1110 - 1120 Draws horizontal and vertical axes, with the origin at the centre of the screen.

1130 - Pause.

1140 - If m>n then the overall width of the locus is greater than the overall height.

1150 - 'Exclusive or' printing command.

```
1155 FOR y=-ab*j TO (ab-1)*j STE
P j*4

1160 LET x=ab-ABS y
1170 LET x=x*j

1175 PLOT 128+x,88
1180 IF m<0 THEN DRAW -x*n/(m+n)
,-y*n/(m+n): GO TO 1200
1190 DRAW -x,-y
1200 IF INKEY$="m" THEN GO TO 12
00
1205 PLOT 128+x,88
1210 IF m<0 THEN DRAW -x*n/(m+n)
,-y*n/(m+n): GO TO 1230
1220 DRAW -x,-y
1230 PLOT 128+m*x/(m+n),88-n*y/(
m+n)
1235 IF INKEY$="n" THEN GO TO 12
35
1240 NEXT y
1250 IF j=1 THEN LET j=-1: GO TO
 1155
1260 LET rep=1110: GO SUB 950
1270 GO TO 10
```

1155 - 1250 Draws the locus in two halves, one with j = 1, the other with j = -1. The complete locus will be a rhombus whose larger diagonal lies on the y-axis and so y is chosen as the independent variable. Taking x as the independent variable may result in very few positions of P being plotted.

1160 - x = the current position of B.

1170 - x is negative for the second execution of the loop.

(y = the current position of A on the y-axis.)

1175 - Plots the current position of B.

1180 - If $m<0$ then a line is drawn from B to P.

1190 - If $m\geqslant0$ then a line is drawn from B to A.

1200 - Halts the demonstration while the 'm' key is held down.

1205 - 1220 Plots B and erases the line.

1230 - Plots the current position of P.

1235 - Halts the demonstration while the 'n' key is held down.

1250 - Loop 1155 - 1240 is executed a second time with j = -1.

1260 - Sets rep = the line number which starts the locus demonstration.

Loop 1310 - 1390 is executed 3 times. Firstly from x = 0 to ab when AB moves from the vertical to the horizontal in the first quadrant. Secondly from x = ab to -ab when AB moves from horizontal to horizontal below the x-axis. Thirdly from x = -ab to 0 when AB moves from the horizontal back to the vertical. As 1155 - 1240, the locus is a rhombus but here the longer diagonal is horizontal and so x is taken as the independent variable.

```
1300 LET j=1: LET a=0: LET b=ab
1305 OVER 1
1310 FOR x=-a*j TO (b-1)*j STEP
j*4
1330 LET y=ab-ABS x
1340 LET y=y*j
1345 PLOT 128,88+y
1350 IF n<0 THEN DRAW x*m/(m+n),
-y*m/(m+n): GO TO 1360
1355 DRAW x,-y
1360 IF INKEY$="m" THEN GO TO 13
60
1365 PLOT 128,88+y
1370 IF n<0 THEN DRAW x*m/(m+n),
-y*m/(m+n): GO TO 1380
1375 DRAW x,-y
1380 PLOT 128+m*X/(m+n),88+n*y/(
m+n)
1385 IF INKEY$="n" THEN GO TO 13
85
1390 NEXT x
1400 IF j=1 AND a=0 THEN LET j=-
1: LET a=ab: GO TO 1310
1410 IF j=-1 AND a=ab THEN LET b
=0: LET j=1: GO TO 1310
1420 LET rep=1110: GO SUB 950
1430 GO TO 10
```

1330 - Evaluates y, the current position of A.

1345 - Plots A.

1350 - If n < 0 then a line is drawn from A to P.

1355 - If n ≥ 0 then AB is drawn.

1360 - Halts the demonstration while the 'm' key is held down.

1365 - 1375 Plots A and erases the line.

1380 - Plots the current position of P.

1385 - Halts the demonstration while the 'n' key is held down.

1400 - 1410 j = 1, a = 0 during the first execution of the loop,
j = -1, a = ab during the second execution of the loop,
j = 1, b = 0 during the third execution of the loop.

1420 - rep = the line number for the beginning of the demonstration.

1500 - 1780 AP:PB = m:n, AB is constant. The locus of P is an ellipse. The coordinates for P are evaluated using the parametric form of the equation $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$,

i.e. x = a sint, y = b cost.

```
1500 IF INKEY$<>"" THEN GO TO 15
00
1510 PRINT ''"Press any key to s
how the locus": GO SUB 900
1520 CLS : PLOT 128,0: DRAW 0,17
5
1530 PLOT 0,88: DRAW 255,0
1540 GO SUB 900
```

1500 - Pause until the key is released.
1510 - Prints a message to press any key for the locus.
1520 - 1530 Draws a horizontal and vertical axis with the origin at the centre of the screen.
1540 - Pause.

```
1545 IF m=0 THEN GO TO 1600
1546 IF n=0 THEN GO TO 1900
```

1545 - If m = 0, the locus of P is the y-axis.
1546 - If n = 0, the locus of P is the x-axis.

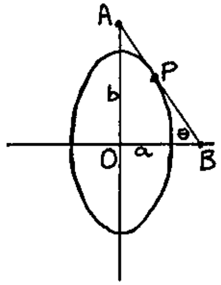
```
1550 LET a=m/(m+n)*ab
1560 LET b=a*n/m
1570 OVER 1
```

1550 - Evaluates a and b for the equation of the ellipse, a = mAB/(m+n), b = nAB/(m+n).

```
1580 FOR t=0 TO 356 STEP 4
1590 LET x=a*SIN (PI/180*t)
1600 LET y=b*COS (PI/180*t)
1610 LET h=ab/SQR (n*n*x*x+m*m*y
*y)
1620 LET ob=n*x*h
1630 LET oa=m*y*h
```

1580 - 1750 Draws the locus (ellipse) by drawing positions of AB and plotting P where AP:PB = m:n.

1580 - 1630



x = a sint, y = b cost

$$\tan\theta = \frac{y}{OB - x} = \frac{y}{\frac{n}{m}.x} = \frac{my}{nx}$$

$$OB = AB\cos\theta = AB.\frac{nx}{\sqrt{(nx)^2 + (my)^2}}$$

$$OA = AB\sin\theta = AB.\frac{my}{\sqrt{(nx)^2 + (my)^2}}$$

```
1640 IF n<0 THEN PLOT 128,88-oa:
 DRAW x,oa+y: GO TO 1680
1650 PLOT 128+ob*SGN m,88
1660 IF m<0 THEN DRAW ob+x,y: GO
 TO 1680
1670 DRAW -ob,oa
1680 IF INKEY$="m" THEN GO TO 16
80
1690 IF n<0 THEN PLOT 128,88-oa:
 DRAW x,oa+y: GO TO 1730
1700 PLOT 128+ob*SGN m,88
1710 IF m<0 THEN DRAW ob+x,y: GO
 TO 1730
1720 DRAW -ob,oa
1730 IF INKEY$="n" THEN GO TO 17
30
1740 PLOT 128+x,88+y
1750 NEXT t
1760 LET rep=1520
1770 GO SUB 950
1780 GO TO 10
```

1640 - If n<0 then A is plotted and AP is drawn since P lies on AB produced.

1650 - Plots B.

1660 - If m<0 then BP is drawn since P lies on BA produced.

1670 - Draws AB if both m and n are non-negative.

1680 - Halts the demonstration while the 'm' key is held down.

1690 - 1720 Erases the line drawn above.

1730 - Halts the demonstration while the 'n' key is held down.

1740 - Plots P.

1760 - 1780 If the 'r' key is pressed, the demonstration is repeated from line 1520. Any other key returns to the initial option display.

```
1800 OVER 1: LET j=1
1810 FOR y=ab*j TO -ab*j STEP -3
*j
1820 LET x=SQR (ab*ab-y*y)*j
1830 PLOT 128,88+y: DRAW x,-y
1840 IF INKEY$="m" THEN GO TO 18
40
1850 PLOT 128,88+y: DRAW x,-y: P
LOT 128,88+y
1860 IF INKEY$="n" THEN GO TO 18
50
1870 NEXT y
1880 IF j=1 THEN LET j=-1: GO TO
 1810
1885 LET rep=1520
1890 GO SUB 950
1895 GO TO 10
1900 OVER 1
1905 LET j=1: LET k=0
1910 LET as=0
1915 LET af=ab
1920 FOR x=as TO af STEP 3*j
1925 LET y=SQR (ab*ab-x*x)*j
1930 PLOT 128,88+y: DRAW x,-y
1935 IF INKEY$="m" THEN GO TO 19
35
1940 PLOT 128,88+y: DRAW x,-y: P
LOT 128+x,88
1945 IF INKEY$="n" THEN GO TO 19
45
1950 NEXT x
1960 IF j=1 AND k=0 THEN LET j=-
1: LET k=1: LET as=ab: LET af=-a
b: GO TO 1920
1970 IF j=-1 THEN LET j=1: LET a
s=-ab: LET af=0: GO TO 1920
1975 LET rep=1520
1980 GO SUB 950
1990 GO TO 10
```

1800 - 1895 The locus of P = the locus of A.

1810 - y = the current position of A.

1820 - x = the current position of B.

1830 - Plots A and draws AB.

1840 - Halts the demonstration while the 'm' key is held down.

1850 - Erases AB and plots P (P = A).

1860 - Halts the demonstration while the 'n' key is held down.

1880 - Repeats the above loop for AB in quadrants 3 and 2.

1885 - If the 'r' key is pressed, the demonstration is repeated from line 1520. Any other key returns to the initial option display.

1900 - 1990 The locus of P = the locus of B.

1910 - 1915 Sets as and af for AB in the first quadrant.

1920 - x = the current position of B.

1925 - y = the current position of A.

1930 - Plots A and draws AB.

1935 - Halts the demonstration while the 'm' key is held down.

1940 - Erases AB and plots P (P = B).

1945 - Halts the demonstration while the 'n' key is held down.

1960 - Repeats 1920 -1950 with AB in the 4th and 3rd quadrants.

1970 - Repeats 1920 - 1950 with AB in the 2nd quadrant.

1975 - 1990 If the 'r' key is pressed, the demonstration is repeated from line 1520. Any other key returns to the initial option display.

```
2000 IF INKEY$<>"" THEN GO TO 20
00
2005 CLS : PRINT "A lies on a ci
rcle, centre C.  B is a fixed p
oint"''"The ratio of BC:radius =
 l:r"
2010 PRINT AT 12,0;"Input l"
2020 INPUT l
2030 IF l<0 THEN PRINT AT 10,0;"
l cannot be negative, try again"
: GO TO 2020
2040 PRINT AT 10,0;"
                     "''"Input r"
2050 INPUT r
2060 IF r<=0 THEN PRINT AT 10,0;
"r must be positive, try again
": GO TO 2050
2070 IF INKEY$<>"" THEN GO TO 20
70
2080 PRINT AT 3,25;l;":";r
2090 PRINT AT 6,0;"Locus of a po
int which lies     on AB"''"
AP:PB = m:n"
2100 PRINT AT 10,0;"
                     "''"Input m"
2110 INPUT m
2120 PRINT AT 10,0;"
                     "''"Input n"
2130 INPUT n
2135 IF INKEY$<>"" THEN GO TO 21
35
2140 IF m=0 AND n=0 THEN PRINT A
T 10,0;"m and n cannot both be 0
": GO TO 2110

2150 IF m<0 AND ABS m>n THEN LET
 m=-m: LET n=-n
2160 IF n<0 AND ABS n>m THEN LET
 m=-m: LET n=-n
2165 PRINT AT 8,13;m;":";n''''"P
ress any key for the locus": GO
SUB 900
```

2000 - 2400 The locus of P, on AB, where A lies on a circle, centre C, and B is fixed.

2000 - Pause until the key is released.

2005 - Prints the message as in figure 3.38 and awaits the input of l.

2020 - Input l.

2030 - Error message if $l<0$. ($l = 0$ results in B and C being coincident.)

2040 - Prints an instruction to input r.

2050 - Input r.

2060 - Error message if $r \leq 0$.

2070 - Pause until the key is released.

2080 - Overprints the letters l and r with their values.

2090 - 2160 Inputs m and n where AP:PB = m:n. Error message if m = 0 and n = 0.

2150 - 2160 The ratio is expressed so that the numerically larger number is positive.

2165 - Overprints the letters m and n with their values.

The locus of P is a circle which is an enlargement of the circle, centre C. Centre of enlargement = B, scale factor = $n/(m+n)$.

```
2170 LET ll=l+r: IF l<r THEN LET
 ll=2*r
```

2170 - ll = the distance from B to the opposite side of the fixed circle, or its diameter if B lies inside the circle, i.e. when $l<r$.

```
2175 LET ys=INT (87/r): LET xs=I
NT (255/ll)
```

2175 - ys = the vertical scale needed to accommodate the fixed circle on the screen. xs = the horizontal scale (pixels per unit) to accommodate ll on the screen.

```
2180 IF m<0 THEN LET ll=ll*n/(m+
n): LET xs=INT (255/ll)
```

2180 - If $m<0$ the enlargement scale factor is greater than 1, in which case the horizontal distance to accommodate the enlarged circle is $ll*n/(m+n)$. xs is also adjusted.

```
2200 IF n<0 THEN LET ll=(l+r)*m/
(m+n): LET xs=INT (255/ll)
```

2200 - If $n<0$ the enlargement scale factor is negative. ll = the distance from the extreme left of the locus to the extreme right of the fixed circle.

```
2205 LET rr=r*ABS n/(m+n): IF rr
>r THEN LET ys=INT 87/rr
2210 IF ys>xs THEN LET sf=xs: GO
 TO 2230
2220 LET sf=ys
2230 IF l>=r THEN LET b=0
```

2205 - rr is the radius of the circle which is the locus of P. If $rr>r$ then the vertical scale, ys, is reduced in proportion.

2210 - The scale factor, sf, is the smaller of ys and xs, to accommodate the whole demonstration.

2230 - If $l \geqslant r$, B lies on or outside the fixed circle, in which case B is plotted on the extreme left of the screen.

```
2235 IF l<r THEN LET b=(r-l)*sf:
 IF m<0 THEN LET b=(r-l)*n/(m+n)
*sf
```

2235 - If $l<r$, B lies inside the circle, in which case b = the distance from B to the fixed circle (shortest distance). If $m<0$ (enlargement s.f. > 1) and B lies inside the fixed circle, the locus of P will lie partly outside the fixed circle, so b = the shortest distance from B to the locus of P.

```
2240 IF n<0 AND ll>2*r THEN LET
b=(l+r)*-n/(m+n)*sf
```

2240 - If $n<0$ (enlargement s.f. < 0) and $ll>2r$ (part of the locus lies outside the fixed circle) then b = the distance from B to the extreme left of the locus.

```
2250 LET c=b+l*sf: LET cc=b+(c-b
)*n/(m+n)
2251 LET r=r*sf: LET rr=rr*sf*SG
N n
```

2250 - c = the position of C, the centre of the fixed circle, from the left of the screen. cc = position of the centre of locus P.

2251 - r = radius of the fixed circle. rr = radius of locus P.

```
2255 CLS
2260 PRINT AT 10,b/8;"B";TAB c/8
-1;"C": PLOT b,88: PLOT c,88
2265 CIRCLE c,88,r
2270 GO SUB 900
2275 FOR j=-30 TO 30
2280 LET x=c+r*COS (PI/30*j): LE
T y=-r*SIN (PI/30*j)
2285 LET xx=cc+rr*COS (PI/30*j):
 LET yy=-rr*SIN (PI/30*j)
2290 IF n<0 THEN OVER 1: PLOT x,
88+y: DRAW xx-x,yy-y: GO TO 2310


2300 OVER 0: PLOT b,88: OVER 1
2305 IF m<0 THEN DRAW xx-b,yy: G
O TO 2310
2307 DRAW x-b,y
2310 IF INKEY$="m" THEN GO TO 23
10
2315 IF n<0 THEN OVER 1: PLOT x,
88+y: DRAW xx-x,yy-y: GO TO 2330
2320 OVER 0: PLOT b,88: OVER 1
2325 IF m<0 THEN DRAW xx-b,yy: G
O TO 2330
2327 DRAW x-b,y
2330 OVER 0
2340 PLOT xx,88+yy
2345 IF INKEY$="n" THEN GO TO 23
45
2350 NEXT j
2355 GO SUB 900
2360 PLOT b+(c-b)*n/(m+n),88


2370 GO SUB 900
2380 CIRCLE cc,88,rr


2390 LET rep=2255: GO SUB 950
2400 GO TO 10
```

2255 - Clears the screen.
2260 - Plots and labels B and C.
2265 - Draws the circle with centre C.
2270 - Pause until a key is pressed and released.
2275 - 2350 Draws the locus of P.
  2280 - (x,y) = current position of A on the fixed circle.
  2285 - (xx,yy) = the current position of P.
  2290 - If $n<0$, the enlargement scale factor is negative, and so A is plotted and AP is drawn (AP passes through B).

  2300 - If $n \geqslant 0$ then B is plotted.
  2305 - If $m<0$ then P lies on BA produced, so BP is drawn.
  2307 - If neither m nor n is negative, then BA is drawn.
  2310 - The demonstration is halted while the 'm' key is pressed.
  2315 - 2327 Erases the line drawn above.

  2340 - Plots the current position of P.
  2345 - Halts the demonstration while the 'n' key is held down.

2355 - Pause until a key is pressed and released.
2360 - Plots the centre of the circle which is the locus of P.

2370 - Pause.
2380 - Draws the circle which is the locus of P.

2390 - 2400 Pause until a key is pressed. If 'r' is pressed then the demonstration is repeated from line 2255, any other key returns to the initial option display.

```
3000 IF INKEY$<>"" THEN GO TO 30
00
3005 CLS : PRINT "l is a fixed l
ine, A is a fixed point"''"PN is
 the perpendicular from P  to th
e line l"''"          AP:PN = m:n"
3010 PRINT AT 12,0;"Input m"
3020 INPUT m
3030 IF m<=0 THEN PRINT AT 12,0;
"m must be positive, try again":
 GO TO 3020
3040 PRINT AT 12,0;"Input n
                   "
3050 INPUT n
3060 IF INKEY$<>"" THEN GO TO 30
60
3070 IF n<=0 THEN PRINT AT 12,0;
"n must be positive, try again":
 GO TO 3050
3080 PRINT AT 6,14;m;":";n''''''
"Press any key for the locus    "



3085 GO SUB 900: CLS
3090 IF m>n THEN GO TO 3500
3100 IF m<n THEN GO TO 3300
```

3000 - Pause until the option key is released.

3005 - 3070 Prints the message as in figure 3.50 which requires m and n to be input.
Error message if either m or n is negative.

3080 - Overprints the letters m and n on the screen with their values, and prints a message instructing the user to press a key for the locus.

3085 - Pause until a key is pressed and released, then clear screen.
3090 - If m>n then the locus will be a hyperbola.
3100 - If m<n then the locus will be an ellipse.

```
3110 REM parabola
3115 PRINT TAB 5;"l"
3120 PLOT 50,0: DRAW 0,175
3130 PRINT AT 11,10;"A": PLOT 30
,88
3135 GO SUB 900
3140 FOR y=-87 TO 87 STEP 2
3145 OVER 1
3150 LET x=15+y*y/60
3160 PLOT 50,88+y: DRAW x,0
3170 OVER 0: PLOT 30,88: OVER 1:
 DRAW x-30,y
3180 IF INKEY$="m" THEN GO TO 31
80
3190 PLOT 50,88+y: DRAW x,0
3200 OVER 0: PLOT 30,88: OVER 1:
 DRAW x-30,y
3210 OVER 0
3220 PLOT x+50,88+y
3225 IF INKEY$="n" THEN GO TO 32
25
3230 NEXT y
3235 LET rep=3085
3240 GO SUB 950
3250 GO TO 10
```

3110 - 3230 Draws the locus of P where PN:AP = 1:1, i.e. the locus is a parabola.

3115 - Draws and labels the fixed line, l, at x = 50.

3130 - Plots and labels the focus, A.

Pause.

3150 - 3170 Evaluates the x-coordinate for the parabola. Plots N, where N is the foot of the perpendicular from P to the directrix. Draws NP. Plots A and draws AP.

3180 - Halts the demonstration while the 'm' key is pressed.

3190 - 3200 Erases PN and AP.

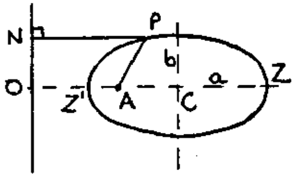
3220 - Plots P.

3225 - Halts the demonstration while the 'n' key is pressed.

3235 -3250 Pause until a key is pressed. The 'r' key will repeat the demonstration. Any other key returns to the initial option display.

```
3300 REM ellipse
3305 LET a=INT (m*250/(m+n))
3310 LET b=a/n*SQR (n*n-m*m)
3315 IF b>87 THEN LET a=INT (87*
n/SQR (n*n-m*m))
3320 LET b=a/n*SQR (n*n-m*m)
3325 LET oa=a*(n*n-m*m)/m/n
3330 LET o=a*n/m
3335 LET s=a*m/n
3340 IF INKEY$<>"" THEN GO TO 33
40
3345 PRINT "l": PLOT 0,0: DRAW 0
,175: PRINT AT 10,oa/8+1;"A": PL
OT oa,88
3350 GO SUB 900
3360 FOR t=0 TO 355 STEP 5
3370 LET x=-a*COS (PI/180*t): LE
T y=b*SIN (PI/180*t)
```

3300 - 3480 Draws the locus of P when $AP:PN = m:n$, $m<n$, i.e. an ellipse, eccentricity, $e = m/n$.



Taking C as the origin and $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ as the equation of the ellipse, $b = a\sqrt{1 - e^2} = \frac{a}{n}\sqrt{n^2 - m^2}$.

$OZ = OC + CZ = \frac{a}{e} + a = a(\frac{1}{e} - 1) = \left(\frac{m+n}{m}\right)a.$

3305 - 3335 Computes the scale to accommodate the ellipse on the screen. oa = the distance OA on the above diagram, o = distance OC, s = distance AC.

3340 - Pause until the key is released.

3345 - Draws the directrix, l, and the focus, A.

Pause.

3370 - Evaluates the coordinates of P using the parametric form $x = -a \cos t$ and $y = b \sin t$. This ensures that the ellipse is drawn starting at point Z' in the diagram.

```
3380 OVER 0: PLOT oa,88: OVER 1:
 DRAW s+x,y
3390 PLOT 0,y+88: DRAW o+x,0
3400 IF INKEY$="m" THEN GO TO 34
00
3410 OVER 0: PLOT oa,88: OVER 1:
 DRAW s+x,y
3420 PLOT 0,y+88: DRAW o+x,0
3430 IF INKEY$="n" THEN GO TO 34
30
3440 OVER 0: PLOT o+x,y+88
3450 NEXT t
3460 LET rep=3340
3470 GO SUB 950
3480 GO TO 10
```

3380 - Plots A and draws AP.

3390 - Plots N and draws PN.

3400 - Halts the demonstration while the 'm' key is pressed.

3410 - Plots A and erases AP.

3420 - Plots N and erases PN.

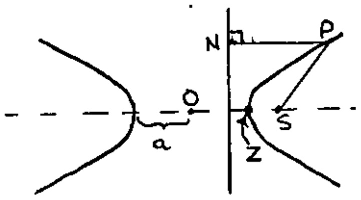
3430 - Halts the demonstration while the 'n' key is pressed.

3440 - Plots the current position of P.

3460 - 3480 Pause until a key is pressed. Pressing the 'r' key repeats the demonstration, any other key returns to the initial option display.

```
3600 REM Hyperbola
3610 LET a=50*n/m
3620 LET b=a/n*SQR (m*m-n*n)
3630 LET z=127+a*n/m
3640 LET s=127+a*m/n
3650 IF INKEY$<>"" THEN GO TO 36
50
3660 PRINT AT 0,z/8-1;"l": PLOT
z,0: DRAW 0,175: PRINT AT 10,s/8
;"A": PLOT s,88
3670 GO SUB 900
3680 LET j=1
```

3600 - 3930 Locus of P where AP:PN = m:n, m>n, i.e. a hyperbola whose eccentricity, e = m/n.



Using the equation $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$,

$b = a\sqrt{e^2 - 1} = \frac{a}{n}\sqrt{m^2 - n^2}$.

O is the origin, S is the focus.

OS = ae = $\frac{am}{n}$, directrix - $x = \frac{a}{e} = \frac{an}{m}$.

3610 - a = 50n/m ensures that the two foci are 50 pixels apart. This was found to be satisfactory for all likely values of m and n.

3660 - Draws and labels the directrix, l.
Plots and labels the focus, A.

3670 - Pause until a key is pressed and released.

```
3690 FOR y=-86*j TO 86*j STEP 2*
j
3700 LET x=j*a/b*SQR (b*b+y*y)+1
27
3705 IF x>255 OR x<0 THEN GO TO
3770
3710 OVER 0: PLOT s,88: OVER 1:
DRAW x-s,y
3720 PLOT z,y+88: DRAW x-z,0
3730 IF INKEY$="m" THEN GO TO 37
30
3740 OVER 0: PLOT s,88: OVER 1:
DRAW x-s,y
3750 PLOT z,y+88: DRAW x-z,0
3760 OVER 0: PLOT x,y+88
3765 IF INKEY$="n" THEN GO TO 37
65
3770 NEXT y
```

3690 - 3770 Executed with j = 1, i.e. y from -86 to 86 in steps of 2. The loop is then executed a second time with j = -1, which draws the left part of the hyperbola.

Uses $x = \frac{a}{b}\sqrt{b^2 + y^2}$ during the first execution,

$x = -\frac{a}{b}\sqrt{b^2 + y^2}$ during the second execution.

3700 - Translates x 127 to the right so that O is central.
3705 - Jumps to the end of the loop if x is off the screen.
3710 - Plots S (labelled A on the screen) and draws PS.
3720 - Plots N and draws PN.
3730 - Halts the demonstraion while the 'm' key is pressed.
3740 - Plots S and erases PS.
3750 - Plots N and erases PN.
3760 - Plots P.
3765 - Halts the demonstration while the 'n' key is pressed.

```
3780 IF j=1 THEN GO SUB 900: LET
 j=-1: GO TO 3590
```

3780 - Repeats 3690 - 3770 with j = -1, i.e. draws the other part of the hyperbola.

```
3790 IF INKEY$="" THEN GO TO 379
0
3800 IF INKEY$="r" THEN CLS : GO
 TO 3650
3810 IF INKEY$="a" THEN GO TO 38
40
3820 IF INKEY$<>"" THEN GO TO 38
20
3830 GO TO 10
```

3790 - 3830 Pause until a key is pressed and released.
'r' repeats the drawing of the locus.
'a' draws the asymptotes (from line 3840).
Any other key returns to the initial option display.

```
3840 IF INKEY$<>"" THEN GO TO 38
40
```

3840 - Pause until the key is released.

```
3850 IF b/a>87/127 THEN GO TO 39
00
```

3850 - The asymptotes have equations $y = \pm \frac{b}{a} x$.

If the gradient b/a is greater than 87/127 then the asymptotes disappear off the top and bottom of the screen rather than off the sides, and are therefore drawn as in lines 3900 - 3910.

```
3860 PLOT 0,88-b/a*127: DRAW 254
,b/a*254
3870 PLOT 0,88+b/a*127: DRAW 254
,-b/a*254
```

3860 - 3870 Draws the asymptotes when they disappear off the sides of the screen.

```
3890 GO TO 3920
```

3890 - Avoids 3900 - 3910.

```
3900 PLOT 127-87*a/b,1: DRAW 174
*a/b,174
3910 PLOT 127-87*a/b,175: DRAW 1
74*a/b,-174
```

3900 - 3910 Draws the asymptotes when they disappear off the top and bottom of the screen.

```
3920 LET rep=3660: GO SUB 950
3930 GO TO 10
```

3920 - 3930 Press the 'r' key to repeat the locus. Press any other key to return to the initial option display.

```
4000 IF INKEY$<>"" THEN GO TO 40
00
4005 CLS : PRINT "A and B are fi
xed"'"P moves in such a way that
 "''''"A ...PB rotates anticlock
wise          at the same rate as
PA"''"B ...PB rotates anticlockw
ise at        twice the rate of PA"
''"C ...Other relations"
4010 IF INKEY$="" THEN GO TO 401
0
4020 IF INKEY$="c" THEN GO TO 46
00
4030 IF INKEY$="b" THEN GO TO 43
00
4040 IF INKEY$<>"a" THEN GO TO 4
010
4045 IF INKEY$<>"" THEN GO TO 40
45
4050 PRINT '"Option A - PA is in
itially                  horizont
al.          Input the initial a
ngle, in      degrees, which PB m
akes with thepositive x directio
n"
4060 INPUT r
4065 IF r>=180 THEN LET r=r-180:
 GO TO 4065
4066 IF r<0 THEN LET r=r+180: GO
 TO 4066
4070 LET ax=127-87*SIN (PI/180*r
): LET bx=127+87*SIN (PI/180*r)
4080 LET ay=87-87*COS (PI/180*r)
```

4000 - 6150 Locus of the point of intersection of two rotating lines through A and through B.

4005 - Prints the option message as in figure 3.59.

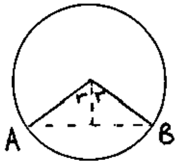
4010 - 4045 Jumps to the line number corresponding to options A, B or C when the appropriate key is pressed.

4050 - Option A - The two lines are rotating at the same rate. The line through A is initially horizontal. The line through B is initially at an angle r°, where r is to be input.

4060 - Input r.

4065 - 4066 Adjusts the value of r so that 0 ≤ r < 180.

4070 - 4080 The locus is a circle. AB is a chord of the circle which subtends an angle r at the circumference.



To draw the circle so that its diameter equals the screen height, ax = distance of A from the left of the screen, bx = distance of B from the left of the screen, ay = the height of AB from the bottom of the screen.

```
4090 IF INKEY$<>"" THEN GO TO 40
90
4100 CLS
4110 PRINT AT 21-ay/8,ax/8+1;"A"
;TAB bx/8-1;"B": PLOT ax,ay: PLO
T bx,ay
```

4090 - Pause until the option key (a) is released.

4100 - Clears the screen.

4110 - Plots and labels A and B.

```
4120 LET k=0: GO SUB 5000
4130 LET j$="k+r": GO SUB 6000
```

4120 - SUB 5000 calculates (p,q) and (s,t), the coordinates of the endpoints on the edge of the screen of the line through A.

4130 - SUB 6000 calculates (u,v) and (w,z), the coordinates of the endpoints on the edge of the screen of the line through B.

```
4140 OVER 1: PLOT p,q: DRAW s-p,
t-q: PLOT u,v: DRAW w-u,z-v
4150 GO SUB 900
```

4140 - Draws the initial lines through A and B.

4150 - Pause until a key is pressed and released.

4160 - 4250 k is the angle which PA makes with the horizontal. j~~$~~ is the angle which PB makes with the horizontal, as a function of k. In this option j = k + r.

```
4160 FOR k=3 TO 180 STEP 3
4170 LET pp=p: LET qq=q: LET ss=
s-p: LET tt=t-q: LET uu=u: LET v
v=v: LET ww=w-u: LET zz=z-v
```

4170 - Allows the next positions of the lines PA and PB, and the coordinates of P, to be calculated before PA and PB are erased. This is so that the time between drawing the lines and erasing them is longer than the time between erasing the lines and drawing the lines for the next position of P.

```
4180 LET x=127+87*COS ((2*k+r-96
)*PI/180): LET y=87+87*SIN ((2*k
+r-96)*PI/180)
4190 GO SUB 5005
4200 GO SUB 6000
4210 IF INKEY$="m" THEN GO TO 42
10
4220 PLOT pp,qq: DRAW ss,tt: PLO
T uu,vv: DRAW ww,zz
4230 OVER 0: PLOT x,y: OVER 1
4235 IF INKEY$="n" THEN GO TO 42
35
4240 PLOT p,q: DRAW s-p,t-q: PLO
T u,v: DRAW w-u,z-v
4250 NEXT k
4260 LET rep=4090: GO SUB 950
4270 GO TO 10
```

4180 - Evaluates the x and y coordinates for P, the intersection of the two lines.

4190 - Calculates the endpoints of the next line through A.

4200 - Calculates the endpoints of the next line through B.

4210 - Halts the demonstration while the 'm' key is pressed.

4220 - Erases the lines AP and BP, previously drawn.

4230 - Plots P.

4235 - Halts the demonstration while the 'n' key is pressed.

4240 - Draws the next lines through A and B.

4260 - 4270 Pressing the 'r' key repeats the demonstraion. Any other key returns to the initial option display.

```
4300 LET ax=40: LET bx=127: LET
ay=88
4310 IF INKEY$<>"" THEN GO TO 43
10
4320 CLS
```

4300 - 4480 Option B - the locus of P is a circle centre B, radius AB. ax, bx and ay are as before, but with values so that B is at the centre of the screen and the radius is 87 pixels.

```
4330 PRINT AT 21-ay/8,ax/8+1;"A"
;TAB bx/8-1;"B": PLOT ax,ay: PLO
T bx,ay
4340 LET k=0: GO SUB 5000
4350 LET j$="2*k": GO SUB 6000
4360 OVER 1: PLOT p,q: DRAW s-p,
t-q
4370 GO SUB 900
4375 OVER 1: PLOT p,q: DRAW s-p,
t-q
4380 FOR k=3 TO 180 STEP 3
4390 LET pp=p: LET qq=q: LET ss=
s-p: LET tt=t-q: LET uu=u: LET v
v=v: LET ww=w-u: LET zz=z-v
4400 LET x=127+87*COS (k*PI/90):
 LET y=88+87*SIN (k*PI/90)
4410 GO SUB 5005
4420 GO SUB 6000
4430 IF INKEY$="" THEN GO TO 44
30
4440 PLOT pp,qq: DRAW ss,tt: PLO
T uu,vv: DRAW ww,zz
4445 IF INKEY$="" THEN GO TO 44
45
4450 OVER 0: PLOT x,y: OVER 1
4460 PLOT p,q: DRAW s-p,t-q: PLO
T u,v: DRAW w-u,z-v
4470 NEXT k
4475 PLOT p,q: DRAW s-p,t-q
4480 LET rep=4310: GO SUB 950
4485 GO TO 10
```

4330 - Plots and labels A and B.

4340 - 4350 Evaluate the endpoints of the lines through A and through B.

4360 - Draws a horizontal line through A and B, as the lines PA and PB are coincident.

4370 - 4375 Erases the line above when a key is pressed and released.

4380 - 4470 As lines 4160 - 4250 except that j = 2k,
x = 127 + 87cos2k, y = 88 + 87sin2k.
The locus is a circle, centre (127,88), radius 87 pixels.

4475 - Draws a horizontal line through A and B.

4480 - Pressing the 'r' key repeats the demonstration.
Pressing any other key returns to the initial options.

```
4600 IF INKEY$<>"" THEN GO TO 46
00
4605 PRINT ''"Option c - PA is i
nitially                 horizon
tal."''"PA rotates anticlockwise
to makeangle k degrees with the
       positive x direction.
       Input the relation betwee
n the  angle j, which PB makes w
ith thepositive x direction, and
 k"
4610 INPUT "j = ";j$
4620 LET ax=111: LET bx=144: LET
 ay=88
```

4600 - Pause until the option key (c) is released.

4605 - Prints the message as in figure 3.69.

4610 - Prints at the bottom of the screen - j = "?", which awaits an expression in k to be input.

4620 - Since the locus of P is not predictable, these positions of A and B were found to suitable for the general case.

```
4625 CLS
4630 PRINT AT 21-ay/8,ax/8;"A";T
AB bx/8-1;"B": PLOT ax,ay: PLOT
bx,ay
4640 LET k=0: GO SUB 5000
4650 GO SUB 6000
4655 OVER 1
4660 PLOT p,q: DRAW s-p,t-q
4661 LET f=0
4665 IF j<>0 THEN PLOT u,v: DRAW
 w-u,z-v: LET f=1
4670 GO SUB 900
4675 LET y=88: LET x=144
4680 LET k=k+3: LET e=0
4690 LET pp=p: LET qq=q: LET ss=
s-p: LET tt=t-q: LET uu=u: LET v
v=v: LET ww=w-u: LET zz=z-v
4700 GO SUB 5000
4710 GO SUB 6000
```

4625 - Clears the screen.

4630 - Plots and labels A and B.

4640 - 4650 Finds the coordinates of the endpoints of the lines through A and through B, having set k = 0.

4660 - Draws the line through A (initially horizontal).

4661 - f = 0 when both lines are horizontal.

4665 - Draws the line through B, unless it is horizontal. If it is not horizontal, f is set to 1.

4675 - Coordinates of the first position of P, i.e. at B.

4680 - increments k by 3 and sets e = 0.

4690 - As 4170.

4700 - Finds the endpoints of the line through A.

4710 - Finds the endpoints of the line through B.

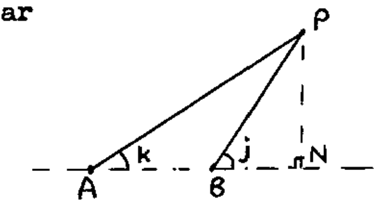
```
4720 IF k=j THEN LET e=1: GO TO
4770
4740 IF j=90 THEN LET x=bx: LET
y=88+33*TAN (PI/180*k): GO TO 47
70
4750 IF k=90 THEN LET x=ax: LET
y=88-33*TAN (PI/180*j): GO TO 47
70
4760 LET x=33*TAN (PI/180*k)/(TA
N (PI/180*j)-TAN (PI/180*k)): LE
T y=88+x*TAN (PI/180*j): LET x=x
+bx
```

4720 - If k = j then the two lines are parallel and so do not intersect, in which case e is set to 1 and 4740 - 4760 omitted.

4740 - 4760

If N is the foot of the perpendicular from P to AB, PN = ANtank = BNtanj, i.e. (AB + BN)tank = BNtanj, and so

$$BN = \frac{ABtank}{tanj - tank}, \quad PN = BNtanj.$$



4740 - If j = 90, BN = 0 and PN = ABtank.

4750 - If k = 90, BN = -AB and PN = -ABtanj. The jump to line 4770 avoids an error when tan90° is required.

4740 - 4760 x is BN, y is PN + 88 pixels.

```
4770 IF INKEY$="m" THEN GO TO 47
70
4780 PLOT pp,qq: DRAW ss,tt
4781 IF f=0 THEN LET f=1: GO TO
4790
```

4770 - Pause in the demonstration while the 'm' key is pressed.

4780 - Erases the line through A.

4781 - If f = 0 then both lines are horizontal initially, in which case only one line was drawn and so only one needs to be erased.(line 4785 is omitted).

```
4785 PLOT uu,vv: DRAW ww,zz
4786 IF INKEY$="n" THEN GO TO 47
86
4787 IF INKEY$="x" THEN GO TO 10
4790 OVER 0: IF y>=0 AND y<=175
AND x>=0 AND x<=255 AND e=0 THEN
 PLOT x,y
4800 OVER 1
4810 PLOT p,q: DRAW s-p,t-q
4820 IF k=0 AND j=0 THEN LET f=0
: GO TO 4840
4830 PLOT u,v: DRAW w-u,z-v
4840 GO TO 4680
```

4785 - Erases the line through B.

4786 - The demonstration is halted while the 'n' key is pressed.

4787 - Pressing the 'x' key returns to the initial options.

4790 - Plots (x,y), the position of P, only if it fits within the screen and the lines are not parallel (indicated by e = 0).

4810 - Draws the line through A.

4820 - Sets f = 0 if both lines are horizontal.

4830 - Draws the line through B if the lines are not both horizontal.

```
5000 IF k>=180 THEN LET k=k-180:
 GO TO 5000
5005 IF k>90 THEN GO TO 5080

5010 IF k=90 THEN LET p=ax: LET
q=0: LET s=ax: LET t=175: RETURN

5020 IF k=0 OR k=180 THEN LET p=
0: LET q=ay: LET s=255: LET t=ay
: RETURN

5030 IF ay/ax<TAN (PI/180*k) THE
N LET q=0: LET p=ax-ay/TAN (PI/1
80*k): GO TO 5050

5040 LET p=0: LET q=ay-ax*TAN (P
I/180*k)

5050 IF (175-ay)/(255-ax)<TAN (P
I/180*k) THEN LET t=175: LET s=a
x+(175-ay)/TAN (PI/180*k): RETUR
N
5060 LET s=255: LET t=ay+(255-ax
)*TAN (PI/180*k)
5070 RETURN
5080 IF ay/(ax-255)>TAN (PI/180*
k) THEN LET q=0: LET p=ax-ay/TAN
 (PI/180*k): GO TO 5100
5090 LET p=255: LET q=ay+(255-ax
)*TAN (PI/180*k)
5100 IF (ay-175)/ax>TAN (PI/180*
k) THEN LET t=175: LET s=ax+(175
-ay)/TAN (PI/180*k): RETURN
5110 LET s=0: LET t=ay-ax*TAN (P
I/180*k)
5120 RETURN
```

Subroutine 5000 - 5120 finds where the line through A meets the edges of the screen. A has coordinates (ax,ay).

5000 - Sets k = k (modulo 180).

5005 - If k is obtuse then the gradient of PA is negative, in which case jump to 5080.

5010 - If k = 90, then PA is vertical and so the coordinates (ax,0) and (ax,175) are returned.

5020 - If k = 0 or 180 (k = 180 is possible when the subroutine is called from line 5005) then PA is horizontal and so the coordinates (0,ay) and (255,ay) are returned.

5030 - If $\frac{ay}{ax}<$ tank, i.e. the gradient OA is less than the gradient of PA, then PA meets the bottom of the screen at the point with coordinates (ax - ay/tank,0).

5040 - If $\frac{ay}{ax}>$ tank, i.e. the gradient of OA is greater than the gradient of PA, then PA meets the left side of the screen at (0,ay - ax tank)

5050 - If the gradient of AS, where S is the top right corner of the screen, is less than the gradient of PA, then PA meets the top of the screen at (ax + (175 - ay)tank,175).

5060 - If the gradient of AS is less than the gradient of PA, then PA meets the right side of the screen at (255, ay+(255-ax)tank).

5080 - 5120 The coordinates of the points where PA meets the edges of the screen are evaluated in a similar way to that above, but for obtuse values of k.

On RETURN (p,q) and (s,t) are the coordinates for the endpoints of the line to be drawn through A.

```
6000 LET j=VAL j$
6010 IF j>=180 THEN LET j=j-180:
 GO TO 6010
6020 IF j<0 THEN LET j=j+180: GO
 TO 6020
6030 IF j>90 THEN GO TO 6110
6040 IF j=90 THEN LET u=bx: LET
v=0: LET w=bx: LET z=175: RETURN

6050 IF j=0 THEN LET u=0: LET v=
ay: LET w=255: LET z=ay: RETURN
6060 IF ay/bx<TAN (PI/180*j) THE
N LET v=0: LET u=bx-ay/TAN (PI/1
80*j): GO TO 6080
6070 LET u=0: LET v=ay-bx*TAN (P
I/180*j)
6080 IF (175-ay)/(255-bx)<TAN (P
I/180*j) THEN LET z=175: LET w=b
x+(175-ay)/TAN (PI/180*j): RETUR
N
6090 LET w=255: LET z=ay+(255-bx
)*TAN (PI/180*j)
6100 RETURN
6110 IF ay/(bx-255)>TAN (PI/180*
j) THEN LET v=0: LET u=bx-ay/TAN
 (PI/180*j): GO TO 6130
6120 LET u=255: LET v=ay+(255-bx
)*TAN (PI/180*j)
6130 IF (ay-175)/bx>TAN (PI/180*
j) THEN LET z=175: LET w=bx+(175
-ay)/TAN (PI/180*j): RETURN
6140 LET w=0: LET z=ay-bx*TAN (P
I/180*j)
6150 RETURN
```

6000 - j = the angle which PB makes with the horizontal.

Subroutine 6000 - 6150 is similar to subroutine 5000 - 5120, except that on RETURN (u,v) and (w,z) are the coordinates of the endpoints of the line to be drawn through B.