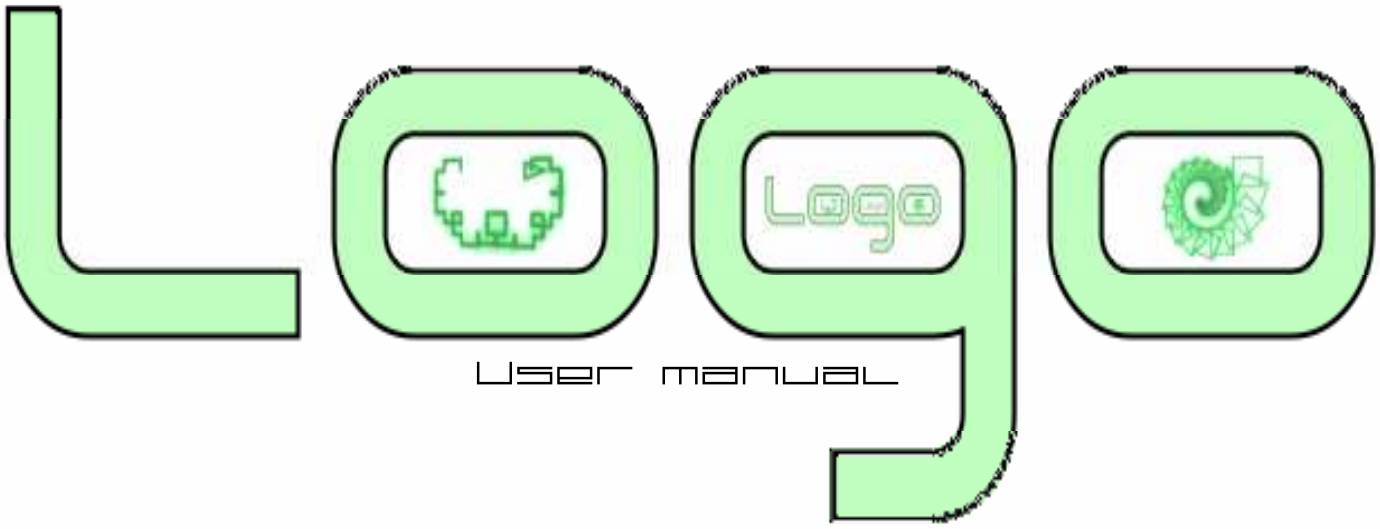


User manual

from Vectorzoa.com

Introduction

Welcome to Vectrex Logo. Logo was a popular learning tool for children in the 80's and was most popular on the BBC Microcomputer in UK schools. It is available in many forms, but this is the first time it has been written for the Vectrex.

To my knowledge this is the first "game" for the Vectrex that is not a regular game, it is a programming language, but I hope it is a game you will find as enjoyable, if not more so that the traditional ones.

It is important to emphasise that this game / programming language is based upon Logo but has been modified to optimise it for the Vectrex. The most obvious difference is the Glyph scripting rather than conventional text. I have borrowed simple concepts from Logo, Pascal, C and.. Basic. Hopefully this will make for an accessible, creative and most of all, fun programming experience.

Whilst Logo can be very simple it can also grow complex in its concepts. I have therefore divided the manual into three sections Beginner, Intermediate, and Advanced. I would suggest even the most adventurous of users fully explore the functionality explained in each section by trying it out on the Vectrex before progressing to the next section.

Logo has been written to allow you, the user the most creativity. You will be able to think of ideas that I have not anticipated, to facilitate this I have put as few safeguards as possible into the system. This prevents my inability to anticipate your creativity from being a hurdle to your programs. The downside is that like any programming language you can abuse it! Infinite loops etc are all possible and like any machine they will crash it. Logo has been implemented on unique hardware which means if you hit the reset button your program *should* survive! (unless you have been really naughty.)

A final word.. as with all Vectrex programming, (and yes you are about to be a real Vectrex programmer) is it very hard to damage your Vectrex and its screen, so feel free to experiment. If something looks deeply disturbing then just hit reset. You will be surprised what your Vectrex can do without damage however to ensure minimum risk to your CRT observe the following.

1. Avoid leaving the Vectrex displaying the same thing motionless for long periods of time, (like over an hour or so.)
2. Drawing a lot of very, very small things can be a strain. You will know what I mean as you will see a **very** bright area on your screen, this simply means the CRT is making a very intense image and reduces the time required before possible screen damage / burn in. Either draw bigger so you can see it properly, or move it about a bit!

OK, enough waffle, be creative and have fun!!

Using Vectrex Logo

There are 3 main screen modes,

- Edit - Shows and edits the program listing
- Draw – Sets the turtle on his way
- Menu – Configuration

You can navigate between these screen modes simply.
Button 3 takes you in and out of menus.

Original Screen Mode	Button	New Screen Mode
Edit	3	Menu
Draw	3	Menu
Menu	3	Edit

Button 4 takes you in and out of Draw mode.

Original Screen Mode	Button	New Screen Mode
Edit	4	Draw
Draw	4	Edit
Menu	4	Menu (select option)

The menus enable you to customise your program control and Turtle behaviour but for the moment lets just accept the defaults and start coding!

Beginner

Logo is all about you and your new best friend – the turtle. You can tell the turtle to move around the screen and leave a trail behind him. Traditionally this was with primitive robots that could hold a *Pen* and trundle around on a paper mat. (This could still happen on Vectrex but not in this release ☺) So our turtle is now shown on the screen and can "draw" by leaving a light trail.

The turtle will do as it is told (by you.) Commands are presented as Glyphs (as in Egyptian hieroglyphics) and are processed in lines from left to right, then down the screen (just like how you read.)

Remember the Vectrex doesn't have a keyboard so we must use the joystick to enter everything. It seems alien but you will find it simple in time. (The overlay is especially helpful as a guide.) The standard controller is fine, but any Vectrex compatible 4 button controller is ok since we are not using the analogue features. I would recommend either a Vectorcade or a converted Sega (Megadrive) Arcade Power Stick.

By now you have probably seen the Logo titles program, and are ready to have a go. So first we must wipe that program to begin with a simpler starting point. (The title program is quite large and flickers a little, your initial programs will be smaller. The Vectrex works hard to try to give you a good picture at all times however I would recommend using the overlay provided. You may find that doubling this up over a minestorm overlay gives the best results for long term use.) So how do we wipe? It seems a little long winded but trust me you don't want to go wiping your own programs by accident! ☺

To clear a program and start with a single glyph..

- Hit Button 3 to bring up the "**Main Menu**"
- Move the joystick to highlight "**Program I/O Menu**"
- Hit Button 4 to select it
- Move the joystick to highlight "**Load from ROM**"
- Hit Button 4 to select it
- Move the joystick to highlight "**5 Clear + Load FD10**"
- Hit Button 4 to select it

You will now return to the Edit screen with a single glyph, hit Button 3 to switch between this and the Draw screen. (you will see the turtle with a very short trail behind it.) So back to the Edit screen…

When you first turn on the Vectrex it will load the Logo title program. Once you have been using Logo Vectrex the machine will attempt to retain your current program through a reboot. (It checks if it still has it, and loads the Logo titles if it has been lost.) If you want to force the Logo title program to be restored on reboot simply hold down Button 4 as it powers up.

Most commands take a single parameter and are created in the following way.

1. Ensure you are in Edit mode
2. Move the cursor (a square) to where you would like to place the command. Top left is a good start!
3. **Hold** Button 1 down
4. Still holding Button 1 move the large crosshairs that will appear **across the screen** (the X axis direction) so the command you require is shown in the top right corner of the screen. The overlay should help you find it. (Some Vectrex screens vary so the overlay is only a guide.)
5. Still holding Button 1 move the large crosshairs that will appear **down the screen** (the Y axis direction) so the parameter you require is also selected. (Ensure you have not moved across from your original command.) Values range from 127 down to 0 (top half), then down to -64 (third quarter), and finally VAR1 to VAR6 (fourth quarter.) For the moment stick to the positive numbers (top half) Note the cursor move upwards slightly slower than it moves down, this is to enable you to select exact figures more easily.
6. Still holding Button 1 move the crosshairs until you are happy with your combined selection. (Both X and Y axis match your choices.)
7. Release button 1
8. Your new Glyph will appear where your original square was positioned.

By passing the square cursor over any glyph you will highlight it (glows) and be reminded of its name and parameters.

We will now explore the commands in detail:

	Forward (FD)

Forward is the most fundamental and useful of all commands. It moves the Turtle forward a distance indicated by the parameter specified. The maximum distance you can travel in one go is 127 (which conveniently takes you to the edge of the screen if you were at the centre.)

If the Pen is down (default) then the Turtle will leave a trail.
If the Pen is up then the Turtle will still move, but will not leave a trail.

	Backward (BK)

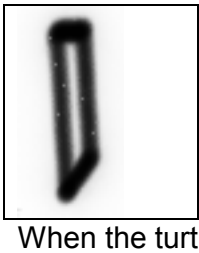
Backward (as you might expect) is just like Forward but moves backward the distance given in the parameter in the opposite direction to the turtle in facing. It does not turn the turtle, but just makes it reverse. Like FD it will leave a trail if the Pen is down.


Left (LT), Right (RT)

Left and Right turn the turtle by the amount given in the parameter. They only change the direction the turtle faces and do not produce any lines. They do not move the turtle sideways! (hey that's classic logo, trust me you'll like it.)

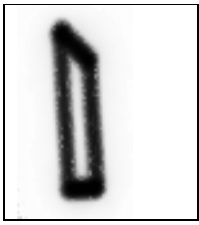
There is a complication however. I haven't used degrees! (it's ok I haven't used radians either.) Since the Vectrex is an 8 bit machine and I want to reduce any unnecessary processing (so there is more time to run *your* programs) I have implemented a compromise. We will call our new unit of angle a "Vegree" there are 64 *Vegrees* in a right angle. If you want to turn right by 90 degrees you just choose 64 Vegrees as the parameter. Here's a quick table for you for reference:

Degrees	**=**	**Vegrees**	**Degrees**	**=**	**Vegrees**
0	=	0	60	=	43
3	=	2	**90**	**=**	**64**
22.5	=	16	120	=	85
30	=	21	179	=	127
45	**=**	**32**	-30	=	-21



Pen Down (PD)

When the turtle moves it will leave a trail. No parameters are required for this command so will be ignored. The default is the Pen is down



Pen Up (PU)

When the turtle moves it will **not** leave a trail. No parameters are required for this command so will be ignored.

Note: If you find setting the values too fiddly then you can set JM Reflex Mode to On to make upward movement on parameter setting much slower.

Still reading? Go have a play now!
OK lets resume, just one more command in the beginner section, but it's a good one!

Repeat

This command is the most complex yet, but you will grow to love it. The first parameter is the number of times you want to repeat some preceding commands. Remember they will have already performed once so this is the number of times to repeat them *again*. By default the Repeat will jump back to the start of its own line. (i.e. to its left.) This will be illustrated by a line from the Repeat command to its destination.

This command uses a second parameter. This is to tell the Repeat command how far to loop back to repeat from. This introduces button 2 to set it as follows:

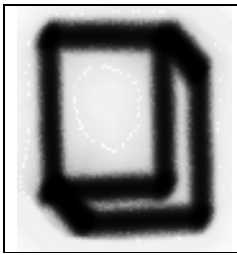
1. Ensure you are in Edit mode
2. Move the cursor to the Repeat where you would like to place the command. Top left is a good start!
3. **Hold** Button 2 down.
4. Still holding Button 2 "drag" the line that is forming at the Repeat command to your loop destination.
5. Release Button 2.
6. A line should now be drawn from the bottom of your Repeat glyph to the bottom of your destination glyph.
7. Repeat from 2 if you are not happy with the loop destination.

Bonus feature: In Draw mode if you can see the Turtle you can use buttons 1 and 2 to change the size of the display. Note it is only a kind of magnify feature and does not make the area the Turtle can reach any larger.

Here's a simple program to enter if you need further explanation of the repeat command.

100 64 3

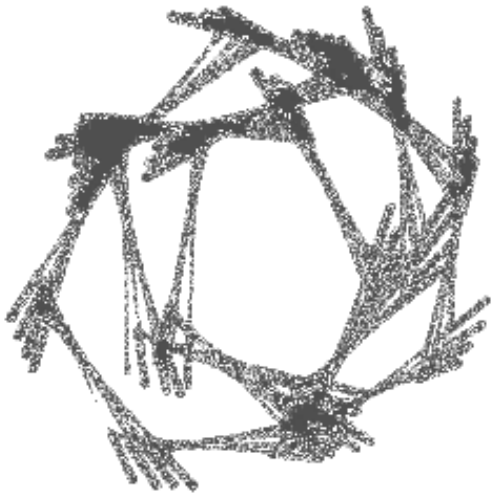
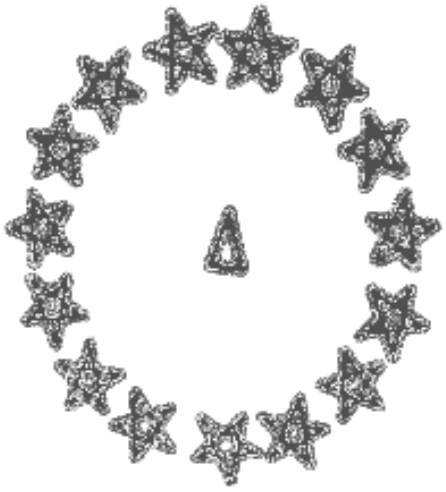
Note: On a few Vectrex's tested the Repeat line can be hard to see. Adjust the brightness very slightly if this proves to be a problem.

	Rubout / Null

This is not a real command but simply removes any glyph previously occupying the position. Use it to clean up your programs.

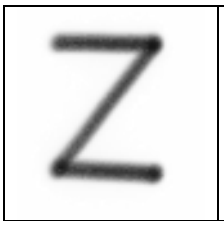
..and that's the end of the beginner section!

Congratulations! With these commands and some practice should now be able to produce images like these (they were made with only the commands we have met so far.)

Intermediate

We now introduce some more complex topics such as using Variables and Jumps.



Set (PD)

Vectrex Logo comes with 6 Variables that you can access and modify. The Set command can be used to initialise them. When a Set glyph is created by default it refers to VAR1 (variable1) and will Set VAR1 to the value in the parameter. If you wish to change this, place the cursor over the Set glyph and hold down button 2. Drag up and down until the VAR you require is shown. To reset the parameter you must replace the Set glyph.

Variables can be viewed and accessed in the Menus. Use Button 3 to access the main menu and then use up and down to highlight the Variables Menu. Select it with Button 4. Each time the Turtle visits the Draw screen the variables will be reset to the values shown in the Initialise Variables screen. By moving the highlighted line up and down you can choose the variable to modify, Buttons 1 and 2 decrement and increment the variable respectively. The advantage of this method is it saves you a glyph on the Edit screen and allows you convenient control of all 6 variables at the same time. The Set command still has its uses though, especially as it is possible to copy variables by selecting another variable as the parameter (hence copying the value of one variable to another)

If you wish to see the current value of the variable (after or during a draw) then select the Actual Values option using Button 4 from the Variables Menu.

Quit the menus using button 3.

Check out the explanation of Advanced Edit Mode in the Menus section for extra flexibility.

Calc

This enables you to perform a calculation resulting in a change to the value of a variable. By default the equation created will be of the form: (let us use a parameter of value 1 for this example.)

VAR1 = VAR1 + <parameter>

VAR1 = VAR1 + 1

Note you cannot add one variable to another variable. This is a limitation, but not one that imposes too much of a restriction. Remember you can copy variables with the set command.

This is our basic equation. Logo has defaulted in the + operator and VAR1. We can modify this with Button 2. By selecting the Calc glyph and dragging whilst holding button 2 we can modify:

The *operator* by dragging left and right. Possible operators are:

- + Add
- - Subtract
- < Left shift (Quick multiply by 2)
- > Right Shift (Quick divide by 2)
- * Multiply
- / Divide (see note)
- & Bitwise And
- ^ Bitwise Or

or The Variable by dragging up and down

- VAR1 through 6

Note if you have enabled Advanced Edit Mode, Button 2 will behave differently, it is up to the user to determine which mode you are comfortable with.

Note regarding Divide. Division and the Vectrex do not make good bedfellows. I have implemented a bit of a cheat for divide (again to keep speed up.) If you wish to divide perform the following calculation to determine the value of the parameter.

To divide by X
Parameter = 128 ÷ X
That is 128 ÷ (your number) = (the Logo number)

Divide By	=	Parameter	Divide By	=	Parameter
2	=	Use > 1	7	=	18
3	=	42	8	=	Use > 3
4	=	Use > 2	9	=	14
5	=	26	10	=	13
6	=	21	20	=	6

So to divide Var1 by 3.

Choose *Calc*, *Var1*, "/" and *42* (128÷3)

If..Jump

This can be considered to be a variation of either

IF (condition) THEN GOTO <destination>

Or

DO <destination> <glyph1> <glyph2> .. WHILE (condition)

In practice it is very simple.

Create the If..Jump glyph. By default the condition created will be of the form: (let us use a parameter of value 1 for this example.)

IF (VAR1 = <parameter>) JUMP <destination>

IF (VAR1 = 1) JUMP <start of line>

By default VAR 1 will be tested and the If..Jump will jump back to the start of its own line. (i.e. to its left.) This will be illustrated by a line from the If..Jump command to its destination.

We can override these with the use of Button 2.

To change the variable:

- Place the cursor over the If..Jump glyph.
- Press Button 2 once (do not hold down) Do not move the cursor from the glyph.
- The Variable chosen will cycle from VAR1 to VAR6.

To change the destination:

- Place the cursor over the If..Jump glyph.
- **Hold** Button 2 down.
- Still holding Button 2 “drag” the line that is forming at the If..Jump command to your Jump destination.
- Release Button 2.

Hopefully this is all starting to make some sense. By mastering the If..Jump command you can control the Turtle and make decisions.

See if you can predict what this program might do:

- CALC VAR1 + 1
- FORWARD 10
- IF..JUMP VAR1 & 004 (points to REPEAT)
- LEFT 16
- REPEAT 31 (points to CALC)

The & is a bitwise AND comparison, currently we can only use the = sign, but we later we will see how to change the operator. (Simply enable Advanced Edit Mode, I suggest for the moment we stick with = though.

Call Proc

This is very similar to the If..Jump above and is akin to the BASIC command GOSUB (goto subroutine) or the ASM command JSR (jump to subroutine) A subroutine is a piece of code that you can invoke but then return to the original place of invocation. In Vectrex logo we will refer to it as a Procedure.

This is non conditional i.e. will always Jump to the destination given.
No numerical parameters are required for this command so the parameter will be ignored.

Call Proc will jump back to the start of its own line. (i.e. to its left.) This will be illustrated by a line from the Call Proc command to its destination. To change the destination Drag with button 2 as explained in If..Jump.

Please do not use this command before reading the next command. (Return)

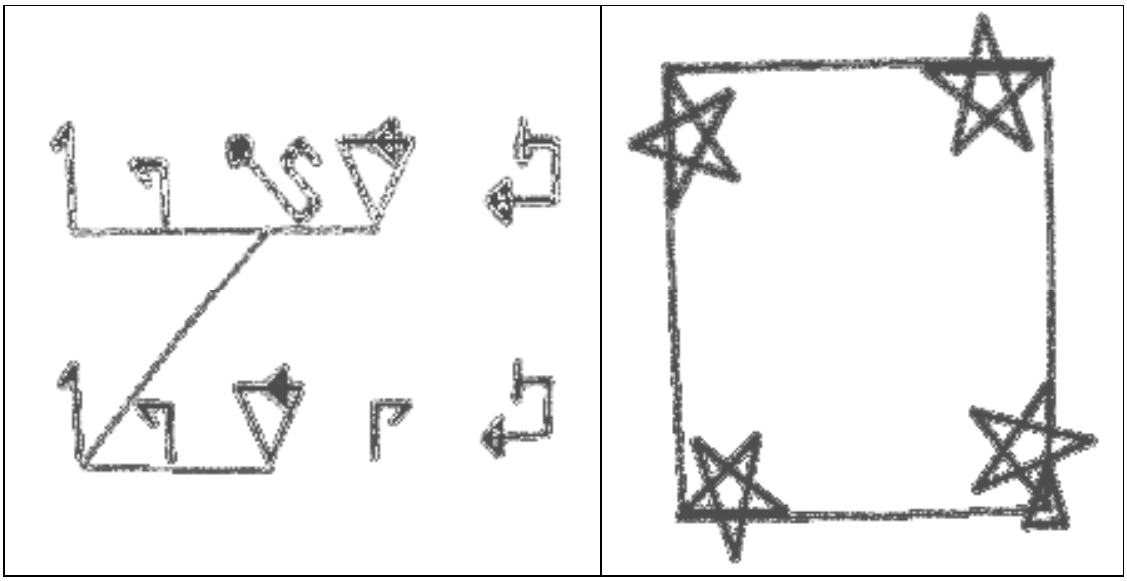
Return

After a sequence of commands have been invoked by the Call Proc command it is necessary to indicate they are complete and that we should return control back to the original position of the invoking Call Proc.

We might create a sequence of glyphs to draw a star. If we wish to draw several stars it is inefficient to repeat our commands. We can refer to them as a Procedure and use the Return command to Jump back to the originating Call Proc.

Logo should look after you and set the parameter to 0 (zero) for now as we will revisit Return in the Advanced section.

Return can also be used to terminate your program execution before its natural end at the bottom rightmost glyph postion. This is especially useful when we have defined a procedure and wish to prevent it being processed accidentally. If you use procedures and forget to “end” your program before them then execution will proceed through them and give extra results. An example program layout is shown below.
To summarise, The top line draws the first line of the square by moving forward, turning 64 Vegrees (90 degrees) and then calling the lower subroutine to draw a star (Note the trailing Right to correct the orientation, it is good practice to make sure your subroutines leave the Turtle as you would hope to find it ☺) Finally the Repeat ensures we do the same for the remaining 3 sides.



	Home

This command sends your Turtle home! i.e. back to the centre of the screen (facing up)

When you create the glyph the parameter will default to zero. There are a few extra options covered in the advanced Section

	Clear Screen

This command clears the screen! i.e. wipes the trail. It is pointless to use in Light speed mode except in conjunction with the parameters below.

When you create the glyph the parameter will default to zero. There are a few extra options covered in the advanced Section

The Turtle Menu

The Turtle menu (accessed using Button 3) allows us to control how the turtle behaves. Options are:

Turtle Light Speed [On/Off]
By default the Turtle moves processes one command at a time, and you get to see the results. By activating Light Speed On, we get to see how fast our turtle can really move. In fact the Turtle can move so fast that when we visit the Draw screen we only see the final result. This is best for the impatient and also a useful tool for advanced topics like animation. It is best used with care. If you have written a complex piece of code or one with hundreds of resultant lines it is best to watch your turtles progress to help with debugging.

Turtle Delay [001-127]
When Light Speed is Off we can see the turtle moving, this sets how long the Turtle pauses between executing each command. Values between 1 and 10 tend to be most useful. Remember that EACH program command causes a delay so if you program is well spaced out this can mean big delays. Use this to your advantage.

Turtle InVisible [On/Off]
This reflects the current Turtle status and does not reset at each visit to the Draw screen. It is traditional to show the Turtle especially with young users, but you may find the turtles presence mars your complex visual masterpieces, in which case make him invisible!

Credits
Rather self explanatory.

The Edit Menu

The Edit menu (accessed using Button 3) allows us to control how the Vectrex behaves. Options are:

Show Parameters [On/Off]
This is applies to the Edit Screen and is normally set to Off to save your Vectrex straining, however it can be useful for small programs. If it is enabled then a vertical line is draw next to each glyph indicating the magnitude of the associated parameter.

Show Loops [On/Off]
This is applies to the Edit Screen and is normally set to On. The Repeat, If..Jump and Call Proc commands each produce a connecting line to another glyph, this can remove the line to save your Vectrex straining with large programs.

Advanced Edit Mode [On/Off]
This allows you a little more control in your programs. It affects how most commands respond to button 2, but is most useful on If..Jump, Set, and Calc. If this mode is Off then you simply drag and tap button 2 to set values as normal. If it is set to On then whilst dragging works as normal, to set loops and jumps, a tap without moving the stick will bring up a small menu. This will be pretty self explanatory but in the case of If..Jump it enables you to easily change the Variable, and Constant at the same time but also allows you to modify the condition operator. Introducing:

- = Equal to
- ! Not equal to
- > Greater than
- < Less than
- & Bitwise And
- ^ Bitwise Or (of dubious value here)

Force Lean Glyphs [On/Off]
Vectrex Logo pushes the hardware somewhat, especially on the Edit screen of large programs. When you exceed around 20 Glyphs you will notice a flicker as the Vectrex “changes gear.” This should not deteriorate further. If you create a huge program the Vectrex knows it is in for a hard time and simplifies the glyphs it draws. Function is unchanged, but they just look a little simpler. They are still labelled and created exactly as normal. It is best to let the Vectrex handle this itself, but if you are comfortable with the shorthand glyphs then you can force them to be used with this option. It will reduce the flicker somewhat on moderate sized programs. Note once this has been activated the mode will remain until a new program is loaded regardless of you changing it to Off. Programs can be saved and dumped as normal independently of the Glyphs’ appearance.

JM Reflex Mode [On/Off]
This simply slows the upward speed of the cursor when setting parameter values. Activate it if your original Vectrex joystick is a little tired.

The Program I/O Menu

Proceed with caution this menu can wipe your program in a flash
(if you ask it to!)

Save to RAM

You have 4 banks in which to Save your programs. These are very useful if you have a complex program and you want to try variation out. Just save your program, continue editing it and then Load it back in again! These should survive a reboot but not the Vectrex being turned off for long.

Load from RAM

Pretty easy, you choose a bank and ***your current program will be overwritten by the contents of the bank!*** If that's what you wanted great, if it was an accident then be more careful! ☺

Load from ROM

This lets you access the logo programs that come with the cart. Just select one of the 4 programs or option 5 to start with a relatively clean edit screen. This was the first thing we did in the Beginner section. The programs are discussed in detail later in an appendix to this manual.

Hex Dump Program

This is an innovative feature that enables you to email your programs to other Vectrex Logo users. Simply select the option and read the numbers off the screen, you can either type them into an email or write them down for your own use. The numbers are in hexadecimal which is a common computer notation. There are six numbers per line and each has 2 digits which can be 0123456789ABCDEF only. 0A can be written as just A but not A0. There is no need to write any zeros if they occur in the far right pairs such as columns 3, 4,5 and 6. You can use the joystick to move up and down. You can stop writing when you reach the word END. Make sure you copy these down correctly, one wrong number will spell trouble!

Hex Enter Program

This will corrupt your current program if you proceed so be sure you have cleared you program or are prepared to. This is a little like a giant highscore chart. You need to use the joystick and buttons to enter the codes that your buddy has emailed you. I have changed the way the buttons function to make it as easy as possible. It does not take more than a few minutes to enter most programs with a little practice.

You will see there are 6 pairs of 2 digit numbers (probably all zeros right now)

- Button 1 increments the left digit. (0123456789ABCDEF and back to 0)
- Button 2 increments the right digit. (as above)
- Button 3 moves to the next pair along (and cycles back to the left eventually)
- Button 4 says "I'm done" and returns you to the edit screen. If it was premature you can re-enter the Hex Enter Program option and resume.

- The joystick moving up or down commits your line to memory. Think of it as the handle on a slot machine. If you do not move the joystick your line will not be saved! Remember this on the last line.

You can move up and down to select the line to edit and hit button 3 to move to a specific pair if you only need a small change.

Remember the small program listed in If..Jump? If you want to enter it now using the Hex Entry this is the code:

```
00 0B 01 2B 00 01
01 01 0A
02 08 04 26 04 01
03 03 10
04 07 1F 1F
```

(remember there is no need to set trailing zeros)

Did you guess right?

This is also a good time to check the health of your Vectrex Beam, if your shape is not a smooth line then reduce the Beam Drift (as explained in the Menu section above) to a lower value (say 004.) If it already looks good then congratulate yourself on having a healthy, well cared for machine!

Now there is no excuse not to share your creativity!

The Run Time Menu

Beam Drift Control [1 to 127]
Your Vectrex is a peculiar and complex device. Unlike modern computers it features some analogue components and these can behave in peculiar ways. The beam that creates the display on your Vectrex tends to lose it's way after a while. The extent of this is down to your individual Vectrex specifically its age/usage and condition. The concept is called Beam Drift and is the reason why some things wobble in some games. It is necessary to reset the beam every now and again to stop it going too far astray. (This does not cause any harm.) By default Logo attempts to fix the beam every 12 lines drawn. You can change the 12 to any values between 1 and 127. A value of 4 would give you a rock solid display, a value of 127 would give you (were you to draw enough lines) a very wobbly display. By synchronising the Beam drift count with the number of lines your geometrical shapes use you can create some very pleasing visual effects. It is best to try it and see. (Light speed On will save you some time.)

Var Watch Window [On/Off]
Displays all 6 variables (first generation only) in real time in the Draw mode. This is only of practical use with Lightspeed off and a Large Turtle Delay (20) however it can be very useful for debugging. Note due to the methods used internally if you need the Turtle trails to be accurate in this mode it is best if the Beam Drift Value is set to a high number.

Advanced

We now introduce some more complex topics such as using Variables and Jumps.

Local

This is the final and most impressive command so let me big it up with some fancy quotations.

"To iterate is human, to recurse, divine."
- Peter L. Deutsch

"To understand recursion, you must first understand recursion."
- unknown (but clearly wise)

Logo is a good language to implement fractal style images through recursion. Each of the 6 variables can be declared as a "local" variable using this command. In Vectrex logo declaring a variable as Local makes a copy of it for use within the current *procedure*. If the procedure calls itself then a further copy can be made, each "level" of recursion holds its own discrete values.

Those which are not "local" are "global" and the values are shared regardless of the level of depth of recursion. A bitmask is used to declare which variables are local, although you need not worry since a menu popup will appear when you create the Local glyph, (hey you're advanced users, you can cope!)

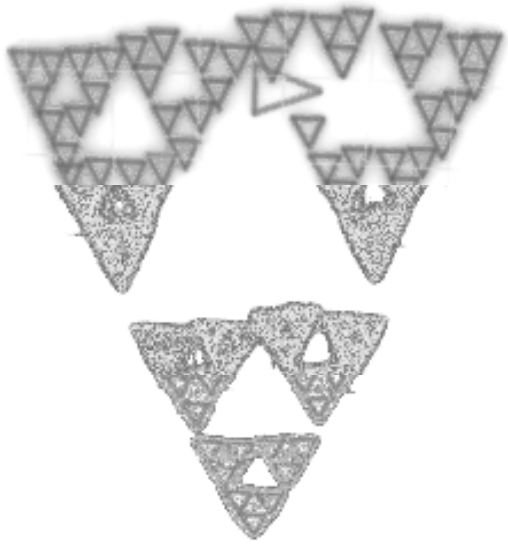
Variables can be independently declared as local at any time.
The following table shows how logo derives the parameter to declare which variables to make local, you don't need to know this but it makes a nice sanity check.

Variable	Parameter
1	1
2	2
3	4
4	8
5	16
6	32

Thus to declare 1, 2 and 4 as local variables we would use a parameter value of 11. Local is the only command that automatically brings up a menu to allow you to declare which Variables to declare as Local. Remember subroutines do

not need a Local command unless you need local variables. (i.e. the regular global ones are not up to the job.)

Here is an example of Vectrex Logo drawing Sierpinski's Gasket (first introduced in the 2005 Snowflake Vectrex coding challenge.)



A tiny word of warning. If you design a program that uses a Repeat statement to recurse into a procedure it may surprise you in its behaviour. This is because Repeats use self sufficient little variables that are global. If you really want to loop into procedures then build your own local loop with Set, Calc, and If..Jump wrapped around the Call..Proc instead.

	Return

If you have created Local variables Return will attempt to match up the closing of these variables by looking at the previous Local command. If you are sneaky and modify the Local command after creating the Return glyph it will not update until you invite it to by tapping Button 2. This is deliberate as it enables a clever programmer to access additional variables, although it gets freakishly complex; abuse it at your peril.

	Clear Screen

Button 2 will activate a parameter selection menu.

The parameter can be used to set his exact behaviour.

Parameter Value	**Behaviour**
0	Clears screen, Light speed off
1	Clears screen, Light speed on
2	Doesn't clear screen, Light speed off
3	Doesn't clear screen, Light speed on

You can create impressive animations by performing

1. CLS (Clear screen, and Lightspeed On)
2. moving the turtle
3. drawing an object
4. CLS (Don't clear screen, and LIghtspeed Off)
5. Loop (Repeat) back to step 1

Remember to set Turtle Delay to very low values.

	Home

Button 2 will activate a parameter selection menu.

The parameter can be used to set his exact behaviour.

Parameter Value	**Behaviour**
0	Centre of screen, turns visible, faces up
1	Stays put, turns visible, same direction
2	Centre of screen, turns visible, faces up
3	Stays put, turns visible, same direction
4	Centre of screen, turns invisible, faces up
5	Stays put, turns invisible, same direction
6	Centre of screen, turns invisible, faces up
7	Stays put, turns invisible, same direction

You now know all you need to know, go work your turtles!

Credits and Acknowledgments.

Vectrex Logo design and coding	Alex Nicholson
Original Logo concept	Seymour Papert
Hardware PCB design	John Maccallan
Overlay design and manufacture by	Kirsty Fletcher
Packaging manufacture by	Kirsty Fletcher
Cart casings by	Sean Kelly & Mark Shaker

Logo program listings, resources, downloads, a Vectrex logo art gallery and details of forthcoming releases can be found at:

Vectorzoa.com

ROM Included Logo Programs

1 Vectrex Logo Title

Not the most elegant of programs, but signwriting is not a turtle's forte. This program contains 2 procedures. One to draw a curved corner, and the other to combine 4 corners to make a ring.
The program runs as follows:

- The first 4 glyphs draw the tail of the "ᑫ" using the corner bend subroutine.
- Next 4 glyphs face left and with the pen up move back to the rightmost edge.
- The entire second line makes use of the pleasing common rings of the "ᗝᑫᗝ" The subroutine to draw a ring is called, and the turtle moves with the pen up to the next ring position. The repeat is used to creat a total of 3 rings.
- The first 6 glyphs of the third line position the turtle to begin the top of the "ᒷ"
- The next 3 glyphs draw the "ᒷ"
- Finally we use the Home glyph to return the turtle to the centre of the screen (carefully preserving its current orientation)
- We face the turtle back upon its self, since it looks nice on the "ᑫ" tail
- The blank spaces are left to provide a natural pause
- A final left to "mostly" move face the turtle back to its original angle ready for the next draw. The rotation is not exact to provide us with a pleasing spin effect.
- Finally a clear screen, and a Repeat to take us back to the start.
- Note the Return after the Repeat to ensure the program does not continue into the subroutines

Something to try (do both these steps at the same time)

- Modify the Clear Screen to be
  - Don't clear screen On
  - Light speed Off
- Add a second Clear Screen right after it (in the convenient gap)
  - Don't clear screen Off
  - Light speed On

2 Square Whirl

A short and elegant program that gives good value in the final result.
24 squares, spiralling out, each of increasing size and spacing. (Note the angle of rotation remains constant.)

The program runs as follows:

- (Upon loading all initialize VARs will be set to zero)
- An initial right to make sure the final result has the best orientation
- The first Calc adds 2 to VAR1
- The second Calc adds 1 to VAR2
- 3 glyphs to draw a square of side VAR1
- A left to offset the turtle angle by just the right amount
- Move forward VAR2 (note VAR2 is always half of VAR1) to give a nice overlap effect
- Repeat from the Calcs; 23 more times

Things to try.

- Use the menu to turn Lightspeed Mode On
- Hit buttons 1 and 2 on the Draw screen.
- Play around with the Left 18 to try different angles.
- For the really advanced, try to write a similar program from scratch that draws pentagrams instead of squares.

3 Fractal Recusion

This program shows recusive techniques. Note it could be written much more concisely, but I have left it in this form to allow maximum flexibility and play. Note how with most recursive programs the work is done by the procedure and not the main program.

The program runs as follows:

- Re-Set VAR2 to 64 (default sizing)
- The rest of the line clears the screen and moves the turtle to the best position to launch the subroutine from.
- Note the final Calc, this increments VAR1 (which started at 1) VAR1 controls the depth of Fractal descent.
- On the third line we check to see if our detail is less than 8, and if so we repeat the whole thing. (With a bigger VAR1)
- Finally we move the turtle and end the program.

The subroutine is a little more complex:

- We declare the first 3 variables as local
- As with all recursion we must know when to stop! VAR1 is decremented (locally)
- This If..Jump is key to recusion. If VAR1 = 0, i.e. we have recursed enough then jump to the portion that draws the lines.
- If VAR1 > 0 then scale VAR2 down and proceed.
- The bottom 4 lines are very similar
    - Move left, right, left calling the subroutine twice
    - Move right,left,right calling the subroutine twice
    - Draw left, right
    - Draw right,left
- Note each of the 4 lines swops the value of VAR3 to ensure alternating directions and then Returns.

Things to try:

- The setting of VAR3 at the end of each of the final four lines in the subroutine is not reallynecessary for the C-curve that is drawn. By modifying their usage you should be able to draw a dragon curve.
- Modify the local to make VAR3 remain global. (Be sure to reset the 4 Returns to match it, by tapping Button 2 over each of them)

4 Star Anim

This program first draws a spinning pentagram and then a (slightly sinister) smiley. It is a good introduction to animation.

This is how it works:

- The first line moves us up and to the left, facing right.
- The second line performs the entire animation here it is in detail
    - Clear screen (using special parameters)
        - Don't clear screen Off
        - Light speed On
    - Put the pen down to do some drawing
    - 3 glyphs draw the pentagram
    - Pen up and move very slightly to create overall movement
    - Clear screen (using special parameters)
        - Don't clear screen On
        - Light speed Off
- The Repeat on the third line repeats the Pentagram cycle 127 times.

Note how the two Clear Screen commands compliment each other and control the speed of the turtle.

The second part of the program is rather simple and utilises a small subroutine that draws a semi-circle. Note how the size of the semi-circle is controlled by setting VAR1 before it is called. This enables us to pass parameters into the subroutine.

- Draw the head by calling the semi-circle subroutine twice with size set to 9
- Draw the eyes by calling the semi-circle subroutine twice with size set to 2.
- Sneaky repeat loop reuses the same code to draw the second eye
- Finally move the turtle and draw the smile with size set to 3.
- Note the Return ends the main program just before the subroutine starts.

Things to try:

- The smiley looks particularly sinister due to the imperfections in the head and eyes (depending on the health of your Vectrex) Adjust
    - The beam drift to 033 (my favourite)
    - The beam drift to and 004 for differing effects. (default is 12)

  The beam drift option can be found in the Real Time Menu