VMOS Beginner's Guide

Contents

Welcome 1
1. Starting VMOS 2
2. Your first program 5
3. Loops 6
4. Decisions 8
5. Your own commands 9
6. Drawing 11
7. Moving things 13
8. Sound and music 16
9. A game 18
10. Going further 20
Appendix A: Editor keys 20
Appendix B: The sixteen colors 21
Appendix C: Common messages 21

Welcome

VMOS is the programming language of the VIXEN-32. You write your program in an editor on the machine itself, press one key, and it runs – not slowly, line by line, the way old BASICs did, but as real machine code, at the full speed of the VIXEN's 100 MHz processor.

If you have used BASIC on a home computer, VMOS will feel familiar: there are `Print`, `For ... Next`, `If ... Then` and their friends. If you have used AMOS on the Amiga, even more so: VMOS borrows many of its words. And if you have never programmed at all, this guide is for you most of all.

This guide is written in the spirit of the computer manuals of the 1980s: it assumes nothing, it keeps every program short, and it gets you moving a spaceship around the screen before the end. Type the programs in yourself – that is how programming is learned – then change them and see what happens. Nothing you type can harm the machine.

The sample programs

Every program in this guide is also on disk, in the folder **Users_Guide_Samples** next to this guide, with the pictures and sounds they use. Copy the whole folder onto your SD card as `/`

`VIXEN-32/VMOS/GUIDE`, and you can load any of them instead of typing it. The name of each program is in its first line, for example `' HELLO - your first VMOS program`.

Chapter	Programs
2 Your first program	HELLO, PAINT, NAME
3 Loops	COUNT, COLORS, TABLE, WAITING
4 Decisions	GUESS, KEYS
5 Your own commands	PROC, FRUIT
6 Drawing	HOUSE, CIRCLES
7 Moving things	BOUNCE, BOBS, SHIP, MOUSE
8 Sound and music	SOUNDS
9 A game	CATCH

The pictures and sounds are BALLS.BMP, SHIP.BMP, BOOM.WAV and YMSONG.VGM. A program that uses one finds it in its own folder.

1. Starting VMOS

From BASIC to VMOS

Switch your VIXEN-32 on. When BASIC says **Ready**, type

```
VMOS
```

and press RETURN. The machine asks **ARE YOU SURE (Y/N)?** – because VMOS takes over the memory BASIC uses, a BASIC program you had typed in is lost. Press **Y**, and a second later you are in the VMOS editor.

To go back to BASIC later, press the **reset button** on the side of the machine.

The editor

The editor shows a small welcome program. Press **F1**: VMOS compiles the program and runs it. It prints a greeting in fifteen colors and waits for a key. Press one, and you are back in the editor.

That is the rhythm of VMOS: type, **F1**, look, change, **F1** again.

A few keys to know from the start:

Key	What it does
F1	Run the program
RUN/STOP	Stop a running program

Key	What it does
F3	Load a program
F4	Save the program
F5	Start a new program
F7	40, 63 or 80 columns
F8	Help: every command on one page
CTRL+Z	Undo

Appendix A lists all the editor's keys.

40, 63 or 80 columns

The top line shows the program's name and how many lines it has, the second line the main keys. VMOS's messages appear on the bottom line, and the bars on the right and at the bottom show where you are in a long program.

The editor starts in 40 columns, with big letters that are easy to read. **F7** switches to 63 columns, then to 80, then back to 40. The program stays the same; you just see more of each line. In 40 columns a long line runs off the right edge, and the editor scrolls sideways when the cursor goes there. Here is CATCH, the game from chapter 9, in all three:



Figure 1: The editor in 40 columns

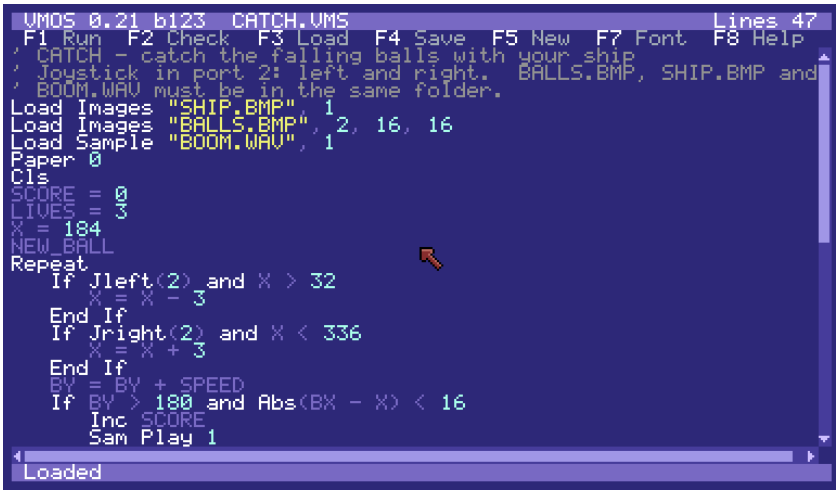


Figure 2: The editor in 63 columns (F7)

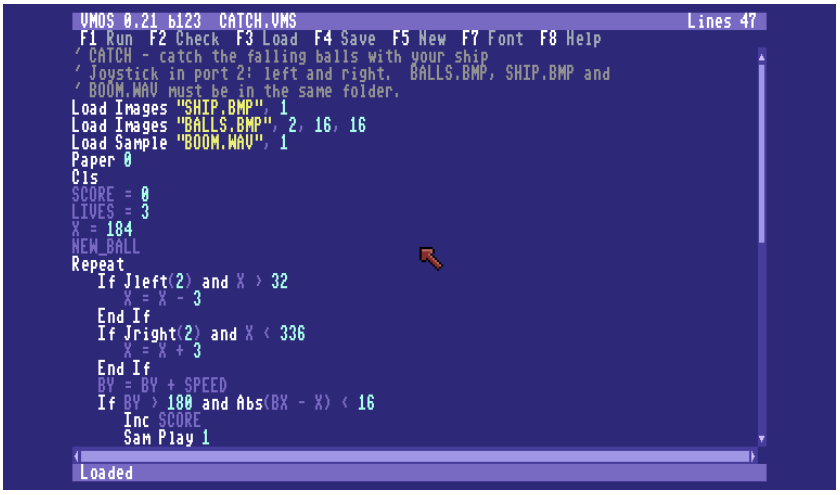


Figure 3: The editor in 80 columns (F7 again)

Typing

Type as on any modern computer: the cursor keys move, DEL deletes to the left, RETURN starts a new line. You can type commands in small letters; when you leave the line, VMOS writes them the way the guide does – `print` becomes `Print`.

TAB indents a line. Indenting the lines inside a loop or an `If`, as the programs here do, makes a program much easier to read.

A line that starts with `'` is a **comment**: VMOS ignores it. It is for you, to remember what the program does.

When something is wrong

If VMOS cannot understand a line, it does not run the program. It puts the cursor on the line and says what it found, for example `For without Next in line 7`. Fix the line and press **F1** again.

A program can also stop while it runs, for example with `Division by zero in line 12`. Again the cursor shows you where.

2. Your first program

Hello

Press **F5** for a new, empty program, and type:

```
' HELLO - your first VMOS program
Cls
Print "Hello! I am a VIXEN-32."
Print "Two and two make "; 2 + 2
Wait Key
```

Press **F1**.

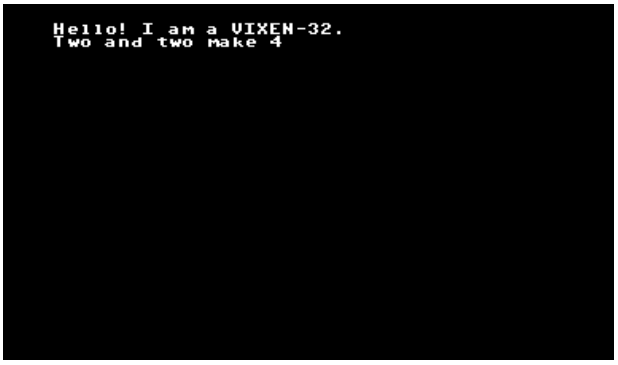


Figure 4: HELLO: your first program

`Cls` clears the screen. `Print` writes on it: text in quotes exactly as it is, and a sum like `2 + 2` worked out first. The `;` puts the next thing straight after the last one.

Why `Wait Key`? When a program ends, VMOS goes back to the editor at once – so fast that you would not see what it printed. `Wait Key` waits until you press a key. Most programs in this guide end with it.

Save your program now: press **F4**, type the name `HELLO` and press RETURN. VMOS adds `.VMS` to the name. Next time you change it, **CTRL+S** saves it under the same name without asking.

Colors

The screen has sixteen colors, numbered 0 to 15 (Appendix B names them). `Pen` sets the color of the text, `Paper` the color behind it, and `Border` the color of the frame around the screen:

```
' PAINT - pens, papers and the border
Border 6
Paper 14
Cls
Pen 1
Print "White on light blue"
Pen 2
Print "Red"
Pen 7
Print "Yellow"
Paper 0
Pen 5
Print "Green on black"
Wait Key
```

`Cls` fills the whole screen with the Paper color, so set the Paper before you clear.

Remembering things

A **variable** is a name that holds a value. `SCORE = 10` stores 10 in `SCORE`; `Print SCORE` shows it; `SCORE = SCORE + 1` makes it 11.

A name ending in `$` holds **text** (called a *string*): `N$ = "Anna"`.

`Input` asks the person at the keyboard and stores the answer:

```
' NAME - variables and Input
Cls
Input "What is your name? "; N$
Input "How old are you? "; AGE
Print
Print "Hello, "; N$; "!"
Print "In ten years you will be "; AGE + 10
Wait Key
```

Type an answer and press RETURN. The question is printed exactly as you wrote it, so end it with a space.

Whole numbers. A plain variable like `AGE` holds whole numbers: `7 / 2` gives 3. That is fast and enough for most programs. (VMOS has numbers with fractions too, `A#` and `A!` – the Command Reference tells you about them when you need them.)

Where the text goes

The screen holds 40 columns and 25 rows of text. `Locate 10, 5` moves the place where the next `Print` writes to column 10, row 5; both count from 0. Try adding `Locate 15, 12` before the first `Print` in HELLO.

3. Loops

Doing things again: For … Next

```
' COUNT - a For loop
Cls
For I = 1 To 10
    Print "Counting: "; I
Next I
Print "Done!"
Wait Key
```

The lines between `For` and `Next` run ten times, with `I` going 1, 2, 3 … 10. Change `10` to `100`, or `1` to `50`. Add `Step 2` after `To 10` to count in twos, or `For I = 10 To 1 Step -1` to count down.

A loop can use its counter for anything – here, to try every color:

```
' COLORS - all sixteen colors, with a loop
Paper 0
Cls
For C = 0 To 15
    Pen C
    Print "This is color "; C
Next C
Pen 1
Wait Key
```



Figure 5: COLORS: color 0 is black on black, so the first line is invisible

And a loop with a question before it:

```
' TABLE - a times table for any number
Cls
Input "Which times table? "; N
For I = 1 To 10
    Print I; " x "; N; " = "; I * N
Next I
Wait Key
```

`*` multiplies; `+`, `-` and `/` do what you expect.

Until something happens: Repeat ... Until

`For` is for a loop that knows how many times it runs. `Repeat ... Until` runs until something becomes true:

```
' WAITING - Repeat ... Until, and the Timer
Cls
Print "Press any key when you are ready."
Timer = 0
Repeat
    Wait Vbl
Until Inkey$ <> ""
Print "You waited "; Timer / 50; " seconds."
Wait Key
```

Three new words:

- `Inkey$` is the key pressed just now, or `""` (nothing) if none. `<>` means *is not*.
- `Wait Vbl` waits for the next picture. The screen is redrawn 50 times a second, so `Wait Vbl` waits a fiftieth of a second. `Wait 50` waits a whole second.
- `Timer` counts those fiftieths from the moment it was set to 0.

There are two more loops. `While ... Wend` tests first and may not run at all. `Do ... Loop` runs for ever, until an `Exit` inside it leaves – the next chapter uses one.

4. Decisions

If ... Then

`If` runs a command only when something is true:

```
If SCORE > 100 Then Print "Well done!"
```

When more than one line depends on it, write the `If` alone and close it with `End If`. `Else` gives what happens otherwise:

```
If AGE < 18
    Print "Young!"
Else
    Print "Grown up!"
End If
```

The comparisons are `=` `<>` `<` `>` `<=` `>=`, and you can join them with `and` and `or`: `If X > 0 and X < 10 Then ...`

A game: guess the number

`Rnd(99)` gives a random whole number from 0 to 99, so `Rnd(99) + 1` is 1 to 100:

```
' GUESS - guess the number
Cls
```

```
N = Rnd(99) + 1
TRIES = 0
Print "I am thinking of a number"
Print "from 1 to 100."
Repeat
    Input "Your guess? "; G
    Inc TRIES
    If G < N
        Print "Too low!"
    End If
    If G > N
        Print "Too high!"
    End If
Until G = N
Print "Right! It was "; N; "."
Print "You needed "; TRIES; " tries."
Wait Key
```

`Inc TRIES` adds 1 to `TRIES`. When it works, change it: guess up to 1000, or give the player only ten tries.

Keys while the program runs

`Input` stops the program until RETURN. A game must keep moving, so it looks at `Inkey$` instead, many times a second:

```
' KEYS - read the keyboard while the program runs
Cls
Print "Press keys. Q stops."
Do
    K$ = Upper$(Inkey$)
    If K$ = "Q" Then Exit
    If K$ <> ""
        Print "You pressed "; K$
    End If
    Wait Vbl
Loop
Print "Bye!"
Wait Key
```

`Upper$` turns small letters into capitals, so `q` and `Q` both stop it. `Exit` leaves the `Do ... Loop`.

5. Your own commands

Procedures

When a program does the same thing in several places, give that thing a name. A **procedure** is a command you make yourself:

```
' PROC - your own commands: procedures
Cls
GREET["Anna"]
GREET["Bob"]
LINE[20]
GREET["everybody"]
Wait Key

Procedure GREET[N$]
    Print "Hello, "; N$; "!"
End Proc

Procedure LINE[L]
    Print String$("-", L)
End Proc
```

`GREET["Anna"]` runs the lines of `Procedure GREET` with `N$` set to `"Anna"`. A procedure without values is written and called without the brackets: `Procedure SCREAM` … `SCREAM`. Procedures go at the end of the program; the program never runs into them by itself.

The variables inside a procedure are its own: a `SCORE` inside is not the `SCORE` of the main program. To use the main program's, name them on a `Shared` line at the start of the procedure, as CATCH does in Chapter 9.

Lists: arrays, Read and Data

An **array** is a numbered list of variables. `Dim F$(4)` makes five of them, `F$(0)` to `F$(4)`. `Data` lines hold values inside the program, and `Read` takes the next one:

```
' FRUIT - arrays, Read and Data
Dim F$(4)
For I = 0 To 4
    Read F$(I)
Next I
Cls
Print "Today's fruit is "; F$(Rnd(4)); "."
Print
Print "All the fruit:"
For I = 0 To 4
    Print I; ": "; F$(I)
Next I
Wait Key
Data "apple", "banana", "cherry", "grape", "melon"
```

6. Drawing

The screen is made of pixels

Besides text, you can draw. The screen is a grid of dots, **pixels**, and every pixel has an *x* (across) and a *y* (down). On the VIXEN, as on the Commodore 64, the count starts behind the border: the top-left pixel you can see is **x 32, y 12**, and the bottom-right one **x 351, y 211**. The picture inside the border is 320 pixels wide and 200 high.

Command	Draws
`Ink c`	sets the color for drawing
`Plot x, y`	one pixel
`Draw x1, y1 To x2, y2`	a line
`Box x1, y1 To x2, y2`	a rectangle's outline
`Bar x1, y1 To x2, y2`	a filled rectangle
`Circle x, y, r`	a circle, r pixels round its middle
`Text x, y, "text"`	text at any pixel, in Pen on Paper

A house

```
' HOUSE - drawing with Ink, Bar, Box, Draw and Circle
' The screen starts at 32,12 and ends at 351,211
Paper 6
Cls
Ink 7
Circle 300, 50, 20
Ink 5
Bar 32, 170 To 351, 211
Ink 2
Bar 110, 110 To 250, 170
Ink 10
Draw 100, 110 To 180, 60
Draw 180, 60 To 260, 110
Draw 260, 110 To 100, 110
Ink 7
Bar 130, 125 To 160, 150
Bar 200, 125 To 230, 150
Ink 9
Bar 170, 135 To 190, 170
Pen 1
Paper 5
Text 160, 188, "MY HOUSE"
Wait Key
```

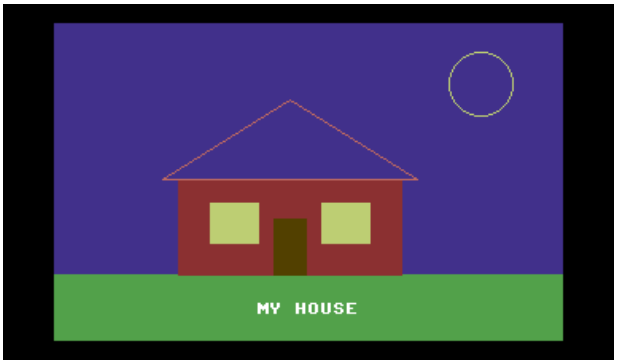


Figure 6: HOUSE: drawn with Bar, Draw and Circle

Move the sun, add a chimney with another `Bar`, put a tree in the garden.

Random art

Drawing in a loop with random numbers makes patterns that never repeat:

```
' CIRCLES - random circles until you press a key
Paper 0
Cls
Repeat
    Ink Rnd(14) + 1
    Circle 32 + Rnd(319), 12 + Rnd(199), Rnd(40)
    Wait Vbl
Until Inkey$ <> ""
```

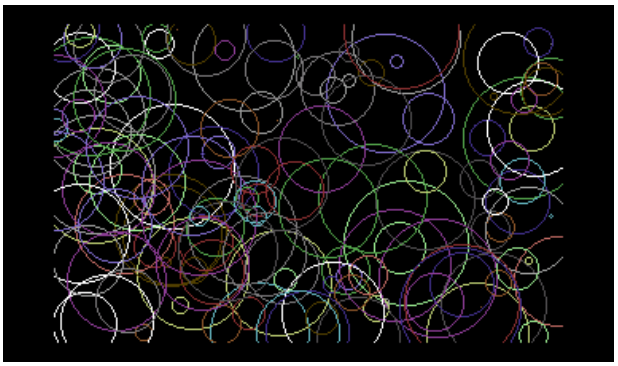


Figure 7: CIRCLES after a few seconds

Take the `Wait Vbl` out and see how fast VMOS really is.

7. Moving things

Sprites

A **sprite** is a small picture that the VIXEN's hardware shows anywhere on the screen, in front of everything else. Moving it costs your program nothing: you only say where it is. The VIXEN has 15 sprites for your programs, numbered 0 to 14.

The pictures come from a BMP file (a 256-color picture; the paint program VPaint makes them). `Load Images "BALLS.BMP", 1, 16, 16` cuts BALLS.BMP into pictures of 16 x 16 pixels, numbered from 1. `Sprite 0, X, Y, 1` then shows picture 1 as sprite 0 with its top-left corner at X, Y:

```
' BOUNCE - a sprite that bounces around the screen
' BALLS.BMP must be in the same folder
Load Images "BALLS.BMP", 1, 16, 16
Paper 0
Cls
Print "Press a key to stop."
X = 100
Y = 50
DX = 2
DY = 1
Repeat
    X = X + DX
    Y = Y + DY
    If X < 32 or X > 336
        DX = -DX
    End If
    If Y < 12 or Y > 196
        DY = -DY
    End If
    Sprite 0, X, Y, 1
    Wait Vbl
Until Inkey$ <> ""
```

`DX` and `DY` are how far the ball moves each picture. When the ball reaches an edge, the program turns its direction round. Sprites use the same grid as drawing; the ball is 16 pixels wide, so it turns at 336, not 351. Change `DX = 2` to `DX = 5` for a faster ball.

`Sprite Off` takes all sprites away again.

Bobs

A **bob** is a picture too, but the VIXEN's blitter draws it into the screen itself, over what you have drawn; when it moves, the drawing behind it comes back. There are 64 bobs, numbered 0 to 63, and they use the same pictures as sprites: `Bob 1, X, Y, 1` works just like `Sprite`. A

bob can also be drawn bigger or smaller: `Bob 0, X, Y, 1, 64, 64` draws picture 1 at 64 x 64 pixels.

```
' BOBS - four balls fly over a big one that grows and shrinks
' BALLS.BMP must be in the same folder
Load Images "BALLS.BMP", 1, 16, 16
Paper 0
Cls
Print "Press a key to stop."
Ink 6
Bar 32, 160 To 351, 211
Dim X(4), Y(4), DX(4), DY(4)
For I = 1 To 4
    X(I) = 30 + I * 60
    Y(I) = 20 + I * 30
    DX(I) = I
    DY(I) = 2
Next I
S = 16
DS = 1
Repeat
    S = S + DS
    If S = 16 or S = 96
        DS = -DS
    End If
    Bob 0, 192 - S / 2, 112 - S / 2, 1, S, S
    For I = 1 To 4
        X(I) = X(I) + DX(I)
        Y(I) = Y(I) + DY(I)
        If X(I) < 32 or X(I) > 336
            DX(I) = -DX(I)
        End If
        If Y(I) < 12 or Y(I) > 196
            DY(I) = -DY(I)
        End If
        Bob I, X(I), Y(I), I
    Next I
    Wait Vbl
Until Inkey$ <> ""
```

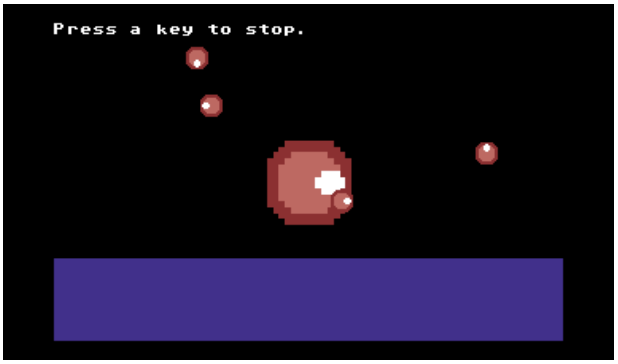


Figure 8: BOBS: the small balls fly over the big one and the blue bar

Bobs are drawn at every `Wait Vbl`, higher numbers over lower ones, so the four small balls (bobs 1 to 4) fly over the big one (bob 0). `S` is the big ball's size, from 16 to 96 pixels and back; `192 - S / 2` keeps its middle in the same place. Each small ball uses its own picture of BALL-S.BMP. `Bob Off` takes all bobs away again.

Pictures bigger than 64 x 64 go in images 241 to 1024; the Command Reference explains them under `Bob`, `Load Images` and `Get Image`.

The joystick

Plug a joystick into **port 2**. `Jleft(2)`, `Jright(2)`, `Jup(2)` and `Jdown(2)` are true while it is pushed that way, `Fire(2)` while the button is held:

```
' SHIP - fly a ship with the joystick in port 2
' SHIP.BMP and BOOM.WAV must be in the same folder
Load Images "SHIP.BMP", 1
Load Sample "BOOM.WAV", 1
Paper 0
Cls
Print "Joystick: fly. FIRE: boom. A key: stop."
X = 184
Y = 150
Repeat
    If Jleft(2) and X > 32
        X = X - 2
    End If
    If Jright(2) and X < 336
        X = X + 2
    End If
    If Jup(2) and Y > 30
        Y = Y - 2
    End If
    If Jdown(2) and Y < 196
        Y = Y + 2
    End If
    If Fire(2)
        Sam Play 1
```

```
    End If
    Sprite 0, X, Y, 1
    Wait Vbl
Until Inkey$ <> ""
```

`Load Images "SHIP.BMP", 1` without a size takes the whole picture as one image. The sound is explained in the next chapter.

The mouse

A mouse goes in **port 1**. `Show` shows its pointer, `X Mouse` and `Y Mouse` are where it points, `Mouse Key` is 1 while the left button is held and 2 for the right one. Here is a tiny paint program:

```
' MOUSE - paint with the mouse in port 1
' Hold the left button to paint, the right one to clear. A key: stop.
Paper 0
Cls
Show
Ink 7
Repeat
    If Mouse Key = 1
        Bar X Mouse, Y Mouse To X Mouse + 2, Y Mouse + 2
    End If
    If Mouse Key = 2
        Cls
    End If
    Wait Vbl
Until Inkey$ <> ""
```

Add keys that change the `Ink`: 1 for white, 2 for red ...

8. Sound and music

The VIXEN has three kinds of sound, and VMOS plays all of them at once:

	What it is	In VMOS
Samples	recorded sounds, WAV files	`Load Sample`, `Sam Play`
The SID	the Commodore 64's sound chip	`Play` voices 1-3, `Sound`
The YM2151	an arcade FM sound chip	`Play` voices 8-15, songs

Samples

`Load Sample "BOOM.WAV", 1` loads a WAV file as sample 1; `Sam Play 1` plays it. The program goes on at once – the sound plays by itself.

Playing notes

`Play` plays a tune written as a string of notes, in the background:

```
Play "O4QCDEFGABO5HC"
```

In the string	Means
`C D E F G A B`	the notes
`#` or `$` before a note	sharp or flat
`R`	a rest
`O4`	octave 4 (middle C) from here on
`W H Q I S`	whole, half, quarter, eighth, sixteenth notes from here on
`T0` to `T9`	the instrument: piano, accordion, calliope, drum, flute, guitar, harpsichord, organ, trumpet, xylophone

A plain `Play "..."` plays on the SID. `Play 8, "..."` plays the same string on the YM2151, the FM chip: its sounds are rounder and richer. `Tempo` makes all tunes faster or slower.

Songs

`Track Load` loads a whole song – a MOD file, a song from the VIXEN's own trackers (.P-ST, .PSY), or a YM2151 song (.VGM) – and `Track Play` plays it in the background until `Track Stop`.

All of it in one program:

```
' SOUNDS - samples, the SID, the YM2151 and a whole song
' BOOM.WAV and YMSONG.VGM must be in the same folder
Load Sample "BOOM.WAV", 1
Track Load "YMSONG.VGM"
Cls
Print "B  a sample (BOOM.WAV)"
Print "S  a tune on the SID"
Print "F  a tune on the YM2151 (FM)"
Print "M  music on, N  music off"
Print "Q  quit"
Do
    K$ = Upper$(Inkey$)
    If K$ = "B"
        Sam Play 1
    End If
    If K$ = "S"
        Play "O4T0QCDEFGABO5HC"
    End If
    If K$ = "F"
        Track Stop
        Play 8, "O4T8QCDEFGABO5HC"
    End If
```

```
    If K$ = "M"
        Track Play
    End If
    If K$ = "N"
        Track Stop
    End If
    If K$ = "Q" Then Exit
    Wait Vbl
Loop
```

YMSONG.VGM is a YM2151 song, so while it plays it has the YM2151 to itself – that is why F stops it first. The samples and the SID still play over it: press M, then B and S.

9. A game

Everything so far, in one game: balls fall from the sky, and you catch them with your ship. Miss three and the game is over. You need the joystick in port 2.

```
' CATCH - catch the falling balls with your ship
' Joystick in port 2: left and right.  BALLS.BMP, SHIP.BMP and
' BOOM.WAV must be in the same folder.
Load Images "SHIP.BMP", 1
Load Images "BALLS.BMP", 2, 16, 16
Load Sample "BOOM.WAV", 1
Paper 0
Cls
SCORE = 0
LIVES = 3
X = 184
NEW_BALL
Repeat
    If Jleft(2) and X > 32
        X = X - 3
    End If
    If Jright(2) and X < 336
        X = X + 3
    End If
    BY = BY + SPEED
    If BY > 180 and Abs(BX - X) < 16
        Inc SCORE
        Sam Play 1
        NEW_BALL
    End If
    If BY > 211
        Dec LIVES
        NEW_BALL
    End If
    Sprite 0, X, 190, 1
```

```
    Sprite 1, BX, BY, 2
    Locate 0, 0
    Print "Score "; SCORE; "   Lives "; LIVES; " "
    Wait Vbl
Until LIVES = 0
Sprite Off
Locate 15, 12
Print "GAME OVER"
Wait 100
Wait Key

Procedure NEW_BALL
    Shared BX, BY, SPEED, SCORE
    BX = 32 + Rnd(304)
    BY = 12
    SPEED = 1 + SCORE / 5
End Proc
```



Figure 9: CATCH: one caught, three lives left

How it works:

- The ship is sprite 0, always at y 190; the joystick moves its `X`.
- The ball is sprite 1 at `BX, BY`. Every picture it falls `SPEED` pixels.
- When the ball is low enough and less than 16 pixels from the ship, it is caught: the score goes up, a sound plays, and `NEW_BALL` starts a new one at the top.
- When it falls past the bottom, you lose a life.
- `NEW_BALL` makes the balls faster as the score grows: one pixel faster every five points.
- `Abs` gives a number without its minus sign, so `Abs(BX - X)` is the distance between ball and ship, from either side.

Ideas: a high score, a ball that sometimes falls faster, a second ball, a different sound for a miss.

Sharing your game

When a program is finished, save it and press **CTRL+B**. VMOS makes two files next to it: **CATCH.RVK**, the game as a program of its own that runs without VMOS, and **CATCH.VXA**,

the pictures and sounds it loads. Copy both to the same folder on a friend's card; they start CATCH.RVK like any other program on the machine.

10. Going further

You now know enough to write real programs. VMOS can do much more, and everything is there when you want it:

- **F8** in the editor shows every command on one page.
- The **VMOS Command Reference** (`basic_command.pdf`, next to this guide) explains every command, with examples you can run.
- The **EXAMPLES** folder holds bigger programs: SPRITES, BOBS, STARS, the YM2151 and network examples, and more.

Some of what waits for you: big bobs made from your own drawings, tile maps and smooth scrolling, the copper (colors that change down the screen), 3D objects, the network, and the VIXEN's own 6502 and Z80 processors.

Have fun – and send your friends your games.

Appendix A: Editor keys

Key	What it does
F1	Run the program
F2 (SHIFT+F1)	Check the program without running it
F3	Load a program: RETURN opens a folder or loads, DEL goes up
F4 (SHIFT+F3)	Save: type a name, RETURN
F5	New program (asks first if it is not saved)
F7	40, 63 or 80 columns
F8 (SHIFT+F7)	Help
RUN/STOP	Stop a running program
Cursor keys	Move; with SHIFT, select text
HOME / CTRL+HOME	Start / end of the line
CTRL+UP / CTRL+DOWN	A page up / down
DEL / SHIFT+DEL	Delete left / right
TAB	Indent

Key	What it does
CTRL+C, CTRL+X, CTRL+V	Copy, cut, paste
CTRL+A	Select all
CTRL+Z	Undo
CTRL+F	Find
CTRL+S	Save under the same name
CTRL+D	Delete the line
CTRL+B	Make a release (.RVK) to share
SHIFT+£	The network on or off
Reset button	Back to BASIC

The mouse works too: click to place the cursor, drag to select, and drag the scroll bars.

Appendix B: The sixteen colors

Number	Color	Number	Color
0	black	8	orange
1	white	9	brown
2	red	10	light red
3	cyan	11	dark grey
4	purple	12	grey
5	green	13	light green
6	blue	14	light blue
7	yellow	15	light grey

These are the Commodore 64's colors. `Pen`, `Paper`, `Border` and `Ink` take these numbers. The VIXEN has 256 colors in all; the Command Reference shows how to use the others.

Appendix C: Common messages

The message	What it means
`... without Next` (or `End If`, `Until`, `Loop`)	a loop or `If` is not closed

The message	What it means
`= expected`	often a misspelled command: `Prnt "Hi"` looks to VMOS like a variable called `Prnt`
`Cannot open: FILE`	the file is not in the program's folder
`Division by zero`	a `/` by a number that was 0
`Array index out of range`	`A(10)` in an array made with `Dim A(5)`
`Stopped`	you pressed RUN/STOP
`Network is off (SHIFT+POUND)`	a network command before SHIFT+£

Every message ends with `in line` and the line's number, and the cursor goes to that line.