

# STOS

## Newsletter

Issue 7 • STOS User Club

*Inside:*

**Horizontal scrolling**

**More about Midi**

**Absolute beginners**

**Extending zones**

**STOS 2.5 – STE compatible!**

*Plus:*

- *More Public Domain games and demos*
- *Spektext • 10-Liners • Mixer Intro*
- *Working between resolutions ...and more!*

# BITS AND BOBS

---

## EDITORIAL *by Aaron Fothergill*

Greetings STOS users! Welcome to my first issue of the newsletter, which is packed with usefull programs, hints and tips for just about all of you. Going on the information gained from your re-subscription forms, it would seem that most STOS club members are in the beginner or moderate programmer league. There would also seem to be a lot of members out there who have been programming hard and have done demos or games. More for them later...

Apologies to those of you who have written in with queries. Hopefully a lot of them will be answered in this newsletter, and I will be going through the rest and replying in the next newsletter, or by post (if you sent an SAE). There might be a slight delay though, as I got them all in a rather large bundle at the same time as the re-subscriptions came in!

For those of you who are attempting to get to grips with STOS, there's a new section. Called "Absolute Beginners", it will be aimed at those of you who would like to program a game, but don't know where to begin. It will help users who haven't even programmed in BASIC before, and will also be useful to those converting to STOS from another language.

You will notice the plastic thing that came with your newsletter. Throw it into your disc drive, and you should have STOS Word, a powerful word processor written in STOS - it was used to write all the listings for this article. If your disc is damaged or doesn't load, then send it back to me for replacement, enclosing an SAE.

## Using STOS Word

STOS Word can either be run as a STOS accessory, a STOS program or as a Gem desktop program. Its best use is from STOS where you can use it to edit text files at the same time as you are developing a program. To load STOS Word as an accessory type:

```
ACCLOAD "STOSWORD"
```

Then access it with the Help key as you would with any other accessory. If you want to use STOS Word as a normal STOS program (so that

you can modify it), type:

```
LOAD "STOSWORD.ACB"
```

...and Bob's your uncle (assuming you have an uncle named Bob!). The version on the disc is the full version of STOS Word, the only difference between your version and the release version is that the release version will be fully packaged and will contain a printed manual plus a few tricks. STOS Word is **not** public domain and can be upgraded as new versions are released for a minimal fee. By the way, this newsletter was written using STOS Word on a 520ST (before transferring to an Apple Mac for layout).

## Speaktex!!

Martin Taylor of Essex has managed to come up with a hack for Speaktex, thus enabling it to be used from within STOS. After some horrendously awkward machine-code programming Speaktex can now be loaded in as a memory bank and called from STOS.

## 'Features'

It would seem that there are quite a few undocumented features and bugs in STOS and, as such, they have earned themselves a regular column. Please write in or phone with anything unusual you find in STOS, as it will be passed on to Mandarin so that they can make STOS and AMOS even better. First a bug which came up while I was writing STOS Word, and which was spotted by several STOS users. The MID\$ function doesn't always work correctly. The example that Martin Taylor (again!) sent in explains this bug the simplest way. In the following program:

```
10 input A$: rem Enter say "ABC"  
20 B$=A$  
30 mid$(A$,1,1)="X"  
40 print B$
```

You would expect B\$ still to contain "ABC" as a copy of the original A\$. But it doesn't - it contains "XBC". Both Martin and myself contacted Mandarin, who explained that after INPUT (and Inkey\$) commands STOS still regards

# BITS AND BOBS

B\$ as the same variable as A\$ due to a mild case of bug (the variable pointers are the same). It can be programmed around by using  $A\$ = \text{mid}\$(A\$,1,N-1) + 'X' + \text{mid}\$(A\$,N+1)$  instead of  $\text{mid}\$(A\$,N,1) = 'X'$ . Hopefully this bug will be fixed in STOS Plus.

## Undocumented bits

Unearthed by Martin Taylor and Stephen Hill almost simultaneously is the fact that you can leave out the sprite frame number from the sprite command. For example, when you are moving a sprite around, but not changing it, you only need to do the `SPRITE N,X,Y,P` command once to put the sprite on the screen, then you only need `SPRITE N,X,Y` to move it around!

A few users have noticed the fact that the screen copy command isn't totally limited to multiples of 16 on the X axis. The area to be copied from can be any X and Y co-ordinate, although the area to be copied to must be on a 16 pixel boundary for the X value.

## PROTECT.BAS

After upgrading STOS to V2.4 with the public domain upgrade disc, some users have found difficulty in using the protect program to 'fix' their run-only programs so that STOS cannot be copied. This is because the upgrade changes the name of the main code file for STOS, so to convert `PROTECT.BAS` to work with V2.4 type:

```
LOAD "PROTECT.BAS",
```

```
Change "BASIC.BIN" to "BASIC204.BIN",  
save "PROTECT.BAS" – and it will now work!  
However Version 2.5 is now available – which works with the brand new Atari STE series. So from now on all references will be to V2.5 of STOS as this is the most up-to-date version. Remember, it only costs £2 (or £1 and a disc) to get the upgrade disc from the PD Library!
```

## Music while loading

Quite a few people have noticed that you cannot play music or samples while loading a program or memory bank. This occurs when either RUN-

ning another program from a loader, or `LOADing` a .MBK file. However, music will still play over a `BLOAD`, as will samples. So to get music to play when loading in a memory bank, `B$SAVE` it as raw data, and `BLOAD` it. The switching off is caused when STOS re-calculates the locations of the memory banks when reserving or erasing them.

## .MAP Files

There are still a few people out there stuck with .MAP files and how to load them. If you want to load the raw data for the map i.e. the .MAP file, you must use the `BLOAD "mapname.MAP",addr` to load it in. Although the recommended way of using the maps is to use the map definer program to save the map routine as an Ascii file, which you then merge in with your program. That way, all the data for the map is contained in data statements.

## STOS 3D Extension

As mentioned in the last issue, the 3D extension for STOS is currently being written. Having seen the early spec it looks pretty impressive as the commands are being written to be very similar to the existing sprite commands. So you could define a 3D object, and then move and animate it under interrupts! There will be full details, of course, when I get hold of a copy sometime early next year.

## STOS Book

As mentioned in the last issue "Game Maker's Manual: Atari ST and STOS Basic" by Stephen Hill should be released soon by Sigma Press. It should be available through your local bookshop in January priced £11.95 for 280 or so pages. Hopefully I will be reviewing it for the next issue.

## Raceway

The Raceway Program in the last issue has caused a few problems with 520ST owners, it would seem that it just won't fit. Ouch!

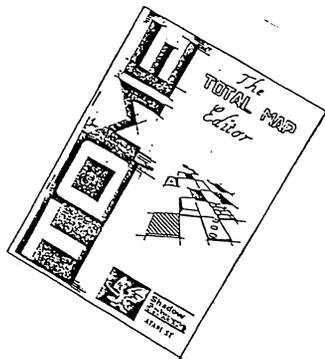
# ADVERTISEMENT

## Create high-speed mapping games with ease!

Yes, you too can write scrolling games like Gauntlet! TOME (The TotalMap Editor) contains everything you need to produce some amazing games – and to show you what's possible TOME comes with a free game, Tin Glove, in a compiled form with sampled sound, and as a Basic listing.

TOME provides a specially-written machine-code extension which adds 10 new commands to STOS. There's also a powerful map editor with many unique features – no wonder it's already being used by some of the top ST programmers to write their latest machine-code games.

TOME comes complete with a 10-page manual which explains how you can make the most of this excellent new package. The manual and two discs come attractively presented in a white A4 vinyl wallet.



**Only £14.95**

(£18.95 for non-STOS Club members)

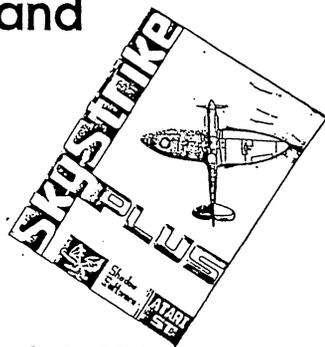
**Available now!**

## Climb into your Spitfire – and it's "chocks away"!

Take to the skies in Skystrike Plus – the deluxe version of the title that's on Games Galore.

56 levels of dogfighting, tank-busting, train-demolishing, battleship-sinking, ground-attacking action over 401 detailed screens.

- 17 frames a second screen update (the same as Xenon II)
- Sampled sound effects
- Loads of hidden features
- Intro demo disc with three-way parallax scrolling, stunning graphics and sampled soundtrack
- Fully compiled – no need to own STOS Maestro



**Only £9.95**

(£11.95 for non-STOS Club members)

**Available now!**

*Both titles written by Aaron Fothergill and work on both 520ST and 1040ST.*

*Send cheque or postal order payable to Shadow Software to:*

*Shadow Software, North Devon.*



# ARTICLE

## Going horizontal alGoing horizont

If I get this one right, it could be the most internationally popular article ever published in a club magazine. So far, we have been asked how do you do horizontal scrolling from members in Australia, New Zealand, Germany, Holland and East Clacton! It would seem that just because some inept programmer wrote in to a magazine and said that horizontal scrolling was impossible on the ST, just about everybody has been trying to do it. Well here it is, not only possible, but in STOS Basic!

**Basic Theory:** The theory behind all scrolling is to grab a section of the screen and move it to a new location, then to put something where the gap caused by the move has appeared. So, if you had two screens in banks 5 and 6, and wanted to scroll horizontally from one to the other, you would use something like this:

```
10 mode 0:key off:reserve as work 5 :
  reserve as work 6:curs off
20 load "Mypic1.Pi1",5 : load
  "Mypic2.Pi1",6 : rem Change the names
  to whatever picture files you want to
  scroll
30 X=0: screen copy 5 to back:screen
  copy 5 to logic
40 screen copy back,16,0,320,200 to
  back,0,0 : rem Move the screen 16
  pixels left
50 screen copy 6,X,0,X+16,200 to
  back,304,0 : rem Cover the gap
60 screen copy back to logic: wait vbl
70 X=X+16 : if X<320 then 40
```

You will note that this screen scroller moves at 16 pixels at a time, and goes very fast! Notice also that I am using the variable X to tell STOS where I want to grab the next chunk of the picture from. So what if you want to move the screen a little smoother than 16 pixels at a go? Okay, you reduce the difference between the area grabbed and the size of the screen, using co-ordinates for instance like 4,0,320,200 instead of 16,0,320,200. However, when you try and stick the gap cover-

ing on the edge, it won't work, as STOS can only put graphics down on a 16 pixel boundary when using screen copy and screen\$. The trick is to only place the 16 pixel wide covering down when there is a gap of 16 pixels at the edge. This is a little messy when you can see the gap, so it is usually done in the background out of sight or even better on a totally different screen. This example uses the lines 10-20 from the last program:

```
30 C=0 : X=0 : rem C counts the number
  of pixels in the gap.
50 reserve as screen 7 : rem Reserve
  ourselves a screen to work on
60 screen copy 5 to 7
70 screen copy 7,1,0,320,200 to 7,0,0 :
  rem Scroll it 1 pixel at a time
80 C=C+1 : rem 1 pixel scroll so add 1 to
  C
90 C=C mod 16 : rem Wrap C round at 16
100 if C=0 then screen copy
  6,X,0,X+16,200 to 7,304,0 : X=X+16 : rem
  If C=0 then there is a 16 pixel gap. Go for
  it!
110 screen copy 7,0,0,304,200 to
  back,0,0:screen copy back to logic:wait
  vbl : rem Show it (it would of course be
  better with screen flip
120 if X<320 then 70
```

You will probably have noticed the 16 pixel border to the right of the screen – this stops you viewing the scrambled section of the scroll, and if you were using this routine in a game, you could cover it up with your scoreboard. These basic routines can be used in almost any form. For instance you could use them to create a scrolling backdrop that is many screens long by putting in a screen number variable (instead of the 6), loading up more than two screens, and when X reaches 320, you set X to 0 and add one to the screen number (that's the New Zealand contingent happy!). The only other problem now is how to place a sprite over this scrolling background without causing all sorts of clashes. Simple: Do

an UPDATE OFF command at the start of your program, and place the sprite just after the area of screen 7 has been copied to the background and logical screens (on line 110) then when all your sprites have been placed, do an UPDATE command to show them. There is of course an easier way to do all of this, just buy the TOME map system (sorry about the shameless plug!) so that you can store your background map as tiles. If you don't want to fork out the measly £14.95 that you need to buy this wonderful program (that's enough of that!), here's a chance to win a copy! This is also the 10-liner competition for this issue. The best horizontally-scrolling game written in 10 lines (yes it is possible) will win a full copy of the TOME system (see the add for details of TOME) including the demo game "Tin Glove", and various programs to show you how to do all sorts of tricks with TOME. As this is quite a difficult challenge, I have wavered the previous rule about having a maximum of 10k of external files.

The rules for this one are :

- *No programs longer than 10 lines.*
- *No machine code (at all!)*
- *Program must be sent in on a disc, with all relevant files.*

As with the previous 10-liner competition, the winning entry, and some of the better losers, will be put together on a compilation disc as one game with each mini game being a level of the main game. Just think - 40+ games on one disc! All games on the compilation disc will earn their authors a cut of the royalties so get programming now! If you send a self addressed stamped envelope (or Jiffy bag) with your disc, we'll send it back to you when the competition has been judged, otherwise we'll wait until you do send us a self addressed stamped envelope! ■

Free with every copy of Games Galore is STOS Squasher, a powerful file compression routine. Simply insert the routine in your STOS folder, then when you boot up STOS you've got two new commands: SQUASH and UNSQUASH so you'll be able to compress your MBK and PRG files by up to 60%!

If you are stuck for space in your games, STOS Squasher could be just what you're looking for.

word  
wrod  
wrdo

## Word Mix

mix  
msi  
xim

This 10-line game has been written by Ralph Effemey. Using the joystick to move a pointer round a grid, try to rearrange the letters so that the key words are displayed correctly. To change the game simply alter the words in the data statements in line 90.

```

10 key off : mode 0 : curs off : flash off :
hide : locate 0,1 : centre "WORD MIX" :
locate 0,23 : centre "PRESS SPACE
FOR NEW WORDS" : on error goto 100
20 P=4 : for l=1 to 4 : read A$ : locate
16,P : print A$ : inc P : next l : locate
15,11 : square 10,6,1 : pen 12 : for l=16
to 23 : for J=4 to 7 : L=scrn(l,J)
30 A=rnd(8)+16 : B=rnd(4)+11 : if
scrn(A,B)<>32 then goto 30 else locate
A,B : print chr$(L) : next J : next l :
X=xgraphic(19) : Y=ygraphic(12) : gr
writing 3
40 X=X+(jleft*8)-(jright*8) : Y=Y+(jup*8)-
(jdown*8) : box X,Y to X+16,Y+16 : wait
20 : box X,Y to X+16,Y+16
50 if X<128 then X=128 else if X>176
then X=176 else if Y<96 then Y=96 else if
Y >112 then Y=112
60 if fire then bell : inc TR : locate 16,20
: print "TRIES=";TR : TX=xtext(X) :
TY=ytext(Y) : T1=scrn(TX,TY) :
T2=scrn(TX+1,TY) : T3=scrn(TX+1,TY+1)
: T4=scrn(TX,TY+1) : locate TX+1,TY :
print chr$(T1) : locate TX+1,TY+1 : print
chr$(T2) : locate TX,TY+1 : print
chr$(T3) : locate TX,TY : print chr$(T4) :
wait 20
70 if inkey$="" then cls : locate 0,1 :
centre "WORD MIX" : locate 0,23 :
centre "PRESS SPACE FOR NEW
WORDS" : TR=0 : goto 20
80 goto 40
90 data "ABSOLUTE","MIRACLES",
"TANGENTS","MANDARIN",
"SOFTWARE","COMPUTER","KEYBOARD",
"JOYSTICK","DATABASE","BOOKWORM",
"WORD MIX","DISC BOX"
100 restore 90 : resume
    
```

# ABSOLUTE BEGINNERS

## The 'Mixer' Intro

Here's another 'Appear' effect – but this time with more documentation! The 'Mixer' takes a screen and separates it into two parts. One part containing all the even lines and the other part all the odd lines. It then scrolls them together to make the whole screen. There are two lots of comments for this program. The first comments are for those of you who have done some BASIC programming and are trying to convert to STOS. The second set is for those of you who have done very little programming at all. It assumes that you know what variables are and how BASIC's line numbering system works. If you don't then see the emergency teaching section below!

### The Program

Boot up your STOS Language disc and type in the following program. It is a good idea to read the comments for each line as you type the line, as this will help you understand better how the commands work. Notice that there are no REMs (Remarks) in this program – this is because they make the lines look much longer and difficult, and nobody types them in anyway! All the comments on the program are listed later.

```
10 mode 0 : key off
20 curs off : hide on
30 reserve as screen 5 : reserve as
screen 6
40 T$="STOS\PIC.PI1"
50 gosub 100
60 rem Your program goes here (O.K so
I put a rem in, so what!)
100 screen copy logic to back : auto
back off
110 load T$,5 : screen copy 5 to 6
120 ink 0 : for a 0 to 198 step 2
130 logic=6
140 draw 0,A to 319,A
```

```
150 logic=5
160 draw 0,A+1 to 319,A+1
170 next A : logic=back
180 get palette(5)
190 for A=0 to 99
200 screen copy 6,0,198-A*2,320,200 to
back,0,0
210 screen$(back,0,198-
A*2)=screen$(5,0,0 to 320,a*2+2)
220 screen swap
230 wait vbl
240 next A
250 logic=physic
260 screen copy logic to back
270 return
```

### Comments

[Note: The text in italics is extra for beginners.]

**10** This sets up the screen in low resolution and removes the function keys from the top of the screen. *The MODE 0 command sets low resolution (mode 1 – sets medium resolution) and the KEY OFF command gets rid of the function keys.*

**20** Here we remove the flashing text cursor and the mouse. *The CURS OFF command removes the text cursor (CURS ON restores it) and HIDE ON hides the mouse pointer. This is required as either would ruin the screen effect.*

**30** Reserves two screens in memory to put the title picture into (one is for the even lines, the other for the odd). *STOS's reserve command can reserve areas of memory for all sort of data: sprites, icons, fonts, machine code programs, pictures and normal data. The RESERVE AS SCREEN command is used to set up a memory bank to store a PI or NEO format screen. We are using it twice here to reserve banks 5 and 6, now referred to as screens 5 and 6.*

**40** The variable T\$ is set to hold the filename of the screen you want to use. It must be a .PI1 or .NEO screen. *This demo is set up to use the title screen from your STOS language disc which has the filename "PIC.PI1" and it is in the folder "STOS", so we have to use the pathname "STOSPIC.PI1"*

**50** calls the picture routine. *The Gosub command is one of BASIC's more useful commands. It allows us to GO to a SUBroutine to do a set of*

# ABSOLUTE BEGINNERS

commands, or mini program, and then to return to the point where we left off. So this program will jump to line 100, and when it gets to the RETURN command on line 270 it will return to just after the GOSUB 100 command (in this case line 60).

60 Ends the program. This is where the rest of your program would go. If you need more space to put your program in, just renumber the subroutine (lines 100-270) so that they are higher up in the program and out of the way (lines 1000-1270 for instance).

## The 'Mixer' Subroutine

100 copies the logical (work) screen to the background screen and sets AUTOBACK to OFF, so that when graphics (in this case lines) are drawn, they are not automatically copied to the background screen. STOS's screen system works like this: You have a PHYSICAL screen, which is the one you see. There is also a BACKGROUND screen, which contains a copy of the PHYSICAL screen without any sprites etc drawn on it. LOGIC (the LOGICAL screen) points to the screen that is going to be drawn upon and is normally set to the PHYSICAL screen.

110 Load our picture into screen 5 and copy it into screen 6. All STOS's graphics commands can be used to work on any screen, including those reserved as memory banks. The picture is copied to screen 6 as we will need two versions of it—one made up of all the even numbered lines, and one with all the Odd lines.

120 Set the drawing colour to 0 (background), and start a count from 0 to 198. The INK 0 command sets the drawing colour to that of the background (Which is see-through in the SCREEN\$( ) function). The FOR...TO command is used when you want to do a function a set number of times. It holds the count in the variable you assigned to it—here we are using A. The STEP function tells it how much to add to the counter on each loop. When the program reaches a connecting NEXT statement, i.e. NEXT A, it will add the step to A, and loop back to just after the FOR...TO..STEP function. In this case line 130. The first FOR...NEXT loop, to divide up the screens. To create the effect of one screen containing all the even num-

bered lines, and the other containing all the odd numbered ones, we will draw lines in the background colour over each screen. Screen 5 will have all the odd lines removed, and screen 6 will have all the even lines removed.

130 set the logical screen pointer to screen 6, so that we are working on that screen. By doing this, we are telling STOS that from now on we want to work on screen 6, so all the graphic output will be drawn on this screen.

140 draws a horizontal line across the screen. Notice that for the Y co-ordinate of the command (the second of each set), the variable A (which is being used as the counter in the FOR..TO..NEXT loop) is used. This means that as the counter goes up, the line is drawn further down the screen. As the counter is going up in steps of 2, this will only draw on even numbered lines.

150 tells STOS we want to work on screen 5 now. See comments for line 130.

160 draws a horizontal line across the screen. As with line 140, we are using the variable A for the Y co-ordinate, but here we are using A+1, so the line is being drawn one line further down, thus drawing only on odd numbered lines.

170 The NEXT A command tells STOS to do the loop again until A is bigger than the second value in the FOR..TO.. statement, in this case 198. By doing logic=back, we are telling STOS to send all the graphics to the background screen, out of sight, but able to be instantly moved into view (see the SCREEN SWAP command in line 220). The first loop ends on line 170.

180 set the screen colours to those used in the picture.

190 Another FOR..NEXT loop, again using A. This time there is no STEP command, so the default of 1 is used for the step.

200 Here we are copying a section from the bottom upwards of screen 6, to the top of the background screen. So that as the loop goes on, more of screen 6 appears at the top of the screen, making it look as if it is scrolling downwards. Note that SCREEN COPY overwrites the screen it is drawing to, so that anything that was below it is removed.

210 This time, we are copying a section of screen

# ABSOLUTE BEGINNERS

---

5, from the top downwards, to the bottom of the background screen. So that it appears to scroll upwards. *The SCREEN\$( ) function always works in what is known as OR mode, so that whatever is copied to a screen is mixed with whatever was underneath it. So when the two areas of screen meet in the middle, the lines will appear to mix.*

220 The SCREEN SWAP command swaps over whatever is on the BACKground and PHYSICAL screens, so that our graphics which are hidden away on the BACKground screen, are now instantly visible. This is useful because you can draw things out of sight and make them visible only when they are finished, thus doing flicker-free animation.

230 The WAIT VBL command tells STOS to stop until the screen has finished updating. This means that each screen gets at least one TV frame (called a Vertical BLank) to be seen in, and thus there is less flicker.

240 Another NEXT A, so that the program goes back to line 200 to continue scrolling, until A is greater than 99. The second loop has now ended.

250 We use LOGIC=PHYSIC to tell STOS that we want to draw on the visible PHYSICAL screen again.

260 And to make sure the background is the same as we are seeing, we copy the PHYSICAL screen to it (remember LOGIC now = PHYSIC).

270 RETURN jumps the subroutine back to where it came from. Okay, now go for it!

## *Emergency Teaching Session*

The programming above assumes that you know what a variable is, and how BASIC's line numbering system works. Here's an explanation of both.

## VARIABLES

The best way of thinking of variables is as little boxes on a blackboard in which numbers are written. Each variable has a name. In STOS this name can be up to 31 characters long, and must begin with a letter. As a number is written into one of these boxes, the previous one is rubbed out, so each variable only remembers the last number that was entered into it. Whenever the

variables name is mentioned in a program statement, it is replaced by its value, so when the command PRINT A is issued, BASIC is really thinking "PRINT whatever is in variable A".

## LINE NUMBERS

All the old versions of BASIC used line numbers to tell the computer where it was up to in a program. Most modern basics now use a method where lines that the computer needs to remember have LABELs on them. However STOS was written with the idea that most people programming in it would be upgrading from 8-bit machines, so line numbers were kept. Each line of commands for the program must have a line number in front of it. The computer then executes these lines in ascending order until it either runs out of lines, or is stopped, either by an error, or a STOP or END command. Some commands such as GOTO and GOSUB (as used in this program) make the computer move to another line. So the command GOTO 50 would make the computer move to line 50 and continue its work from there.

## STOS Hotline

Don't forget that membership to the STOS Club entitles you to ring Aaron Fothergill for assistance on any STOS-related subject.



Ring him on:

or write to:

## From high to low

"Hi-res, who needs it?" would seem to be quite a popular quote by programmers nowadays. However the number of ST owners who only have high resolution monitors and a 1040STF is quite high, especially in Germany, where it is estimated that 90% or so of ST owners work in high resolution. Whether this is because DTP and music applications on the ST are so popular (the high res monitor is really useful for both) or for some other reason, I have no idea. But there is a sizable market out there that is not being reached by games programmers.

In the good old days, programs like Starglider and so on worked in high resolution as well as low resolution. Unfortunately, 99.9% of today's games will only work in low resolution, losing the writers valuable plugs on Saturday morning TV (Climie Fisher mentioned on Going Live that 1/2 the time spent recording their album, they were playing Starglider – a major plug directly to your target market!).

Just think of all those uninspired musicians and desktop publishers who can do nothing but play old games and play with their four-in-a-row desktop accessory! Just think of the advertising potential of inspiring a hit album by your game (I can hear the titles now... "Alien Blasting Blues", "My love was an end of level guardian"...). So how do you go about making your game compatible with high resolution? Read on.

Access to a high resolution monitor is vital. You can develop software using the various mono emulator programs on a TV, but this leads to two things: Severe eye strain and your graphic artist trying to strangle you with the mouse lead (Just try drawing something in Degas Elite using a mono emulator!).

It is usually okay to write your software on a TV and then take it over your friend's house to test it on a mono monitor every weekend. The other way around isn't advisable, I've seen plenty of games with people running around with green/

## Working between resolutions

purple/blue faces, and the sort of colour co-ordination that would make Cyndi Lauper look monochrome!

The only other things you must remember are that all your graphics functions should use some sort of factor to size them for the resolution you are working in, and check at the start of the program for what mode the program is running in.

### Major Differences

The major differences between low and high resolution are these:

- a) Only 2 colours in High res (black and white).
- b) The high resolution screen size is 640x400 pixels, this is where DIVX and DIVY come in useful!
- c) The hi-res screen is 80x25 text characters in size, low res is 40x25 (The high resolution character set is twice as high as the low and medium res ones)
- d) You will get an error if you use a mode 2 command. However you can check for high resolution, because mode=2!

The first thing your program should do is to check for the resolution by looking at the reserved variable MODE. If it equals 2, then you are in high resolution. If it is 0 or 1 then you are in low or medium resolution respectively. Then set up some sizing variables. You already have DIVX and DIVY – you might also want MULX and MULY and MAXX,MAXY (with the maximum X and Y co-ordinates for that screen size). The formulae you would require (aren't I nice?) are:

```
MAXX=640/divx
MAXY=400/divy
MULX=2/divx
MULY=2/divy
```

Another useful idea is to reference all your colours in an array. Instead of doing commands

# ARTICLE

like `inkn` which will return an error if `n>1` in high resolution. You could use `ink CL(n)` where the array `CL(15)` holds the colour numbers usable in that resolution. For Low res it would hold the numbers 0-15. In medium res 0-3 repeating and in high res it would simply hold 0's and 1's, so you won't get an illegal quantity error in high resolution with a command like `ink CL(12)` because `CL(12)=0` in high res and 12 in low res!

Other things to watch out for are your sprites and block graphics. If you are using sprites in a multi resolution program, then you have to define the high resolution ones as well. If you are loading screens you must also do a high resolution version of each screen.

This is where many games writers decide that the extra memory usage and development time needed aren't really worth doing for the extra sales.

If you have enough spare memory in your game, and a bit of time spare in your ever so hurried development schedule then it is quite easy to directly convert your low-res pictures to high-res and convert your sprites (remember: the `sprites2.acb` program on the accessories disk works in all three resolutions) and get yourself some very grateful extra sales. If you are writing utility programs it is almost vital that you get them to work in high-res also.

I have made sure that both `STOS Word` and `TOME` both work in all three resolutions. `STOS Word` is the only word processor that can change between medium and low resolution without going back to the desktop, and `TOME's` commands work automatically in all three resolutions – all you need to do is convert your tile screen and adjust the window size with `divx,divy`. The end result being that anyone can use it, no matter what monitor they are using, unless it is a broken one, or an IBM (Irish Business Machines) paper white screen monitor with the white text option.

So get converting, and send in any ideas you might have for a computer game inspired song title (“You're always on my high score table” by PSB). Make a high resolution user happy today – buy him a TV!

## Hi Happy STOS Users!

Yesterday night, at 6.30 (French time), Richard Vanner asked me, François Lionet, to write a few words for the `STOS Newsletter` (a few words in his mind means four or five pages). I don't personally mind, I'm quite honoured, but what do you think of that? You will have to read it! As Chris Payne always tells me that I'm better at programming than writing (maybe he has read the original `STOS User's Manual` that I wrote – my God), I have made a program for you (I'll have some more in the next issue). HeHeHe!

The first one can be quite a good laugh! It is a fake virus! No, don't call the police yet, I said **fake**: It does not copy itself, and just affects `STOS`. In fact, it is a small extension called `VIRUS.EXF` that you can put into a friend's `STOS` folder. He will not see it unless he goes every hour into the `STOS` folder to see that everything is okay! When `STOS` is loaded, the `VIRUS.EXF` is loaded as a normal extension. It stays hidden for one minute, and then starts to erase the screen from the top to the bottom! It does not crash anything, so you can go on `STOSing` – if you can! I bet the guy will think he has a virus in his machine! Please be gentle, don't let him re-format all his discs and hard drive, stop him before! It is not dangerous, to stop it, just remove `VIRUS.EXF` from `STOS` folder before booting! How to type it in?

- 1 Load the `INPDATA.ACB` accessory
- 2 Type the datas
- 3 Save the bank as `VIRUS.MBK`
- 4 Load the bank: `LOAD "VIRUS.MBK",10`
- 5 Save the program: `SAVE "VIRUS.EXF", START(10) TO START(10)+252`

Now you have your magnificent little virus, ready to frighten friends! Turn to page 17 for the data table – and you'll find the assembly language listing in the next issue of the `STOS Newsletter`!

# Extending zones

by Keith Thomasson

STOS provides simple to use routines to define rectangular zones on the screen and to test whether or not a particular sprite's hot spot is within such a zone, namely "SET ZONE" and "=ZONE(n)". However there are limitations. The first is that the zones must be rectangular, and the second is the limit of 128 zones.

It is possible in some cases to work round the rectangle problem by using the "=DETECT(n)" command, which allows you to identify the colour of the point beneath a sprite's hot spot. But you can't always guarantee that a zone's colour will be unique.

I came across both these problems while working on a project involving a large grid of hexes. I wanted to accurately identify which hex the mouse pointer was in when a mouse button was pressed.

My solution was to create a data bank in which each byte was directly related to a point on the display. The first byte of the data bank identified the top left point of the screen display, the next byte the next point along the top line, and so on. In the program the following code was used:

```
1000 A=start(6) : rem Get start of data bank
...
2000 repeat : until mouse key=0 : rem Wait for no mouse key
2100 repeat : until mouse key : rem Wait for mouse key
2200 gosub 3000 : Call checking routine
```

```
...
3000 X=x mouse : Y=y mouse : rem Get mouse coordinates
3100 B=X+(Y*320) : rem Calculate offset : Note - constant of 320 is for low resolution
3200 Z=peek(A+B) : rem Extract zone number
3300 return
```

Following a call to line 3000, the variable Z contains a number which can be used by the rest of the program to refer to the zone containing the position of the mouse pointer. The use of the X and Y variables in the above example is not strictly necessary as the calculation in line 3100 could be carried out on x mouse and y mouse directly.

However the size of the data bank for a low resolution display is large - 64,000 bytes. This can be reduced by artificially coarsening the resolution. This can be achieved by including the following line:

```
3050 X=bclr 0,X : Y=bclr 0,Y
```

This has the effect of rounding down odd numbered values of X and Y to even values, so that a 2x2 block on the screen is considered a single point. This reduces the data bank size to 16,000 bytes, at the cost of accuracy. This can be taken further if required. It is possible to have any number of zones using this method, but if more than 255 are needed (as I found with the largest

hex grid I used) then the size of the data bank and the calculation constant must be doubled and all access carried out using DEEK and DOKE, giving up to 65535 discreet zones! For

Resolution	Zones	Data Bank (bytes)	Constant	Commands
Low	1-255	64,000	320	peek/poke
Low	>255	128,000	640	deek/doke
Medium	1-255	128,000	640	peek/poke
Medium	>255	256,000	1280	deek/doke
High	1-255	256,000	640	peek/poke
High	>255	512,000	1280	deek/doke

# ARTICLE

reference, the basic data bank sizes and calculation constants required for different resolutions and numbers of zones are as detailed in the table below.

The data bank takes some work to create – particularly if you are going to define zones of differing shapes and sizes. I wrote a special program to create the data bank, set up the values using poke or doke, and saved it to disc. The advantage here was that if something changed, or I wanted to add new zones, it was relatively easy to modify the program and recreate the data bank.

The beauty of this method is the speed with which the value is returned – one short calculation followed by one peek or deek, and the zone number is yours.

The accuracy of the method depends on any artificial coarseness applied. High resolution will almost certainly need some coarsening unless you have memory to burn! The example shown above is for use with the mouse, but changing line 3000 to use x sprite and y sprite gives you the ability to check the location of any active sprite on the screen.

I hope you find this useful if you ever come across the limitations of zone shape or numbers of zones.

(Right, I'm off to write a game that uses one zone per pixel! – Aaron.) ■

## *Drop us a line!*

Whether you've got a problem – or just got something to say, we'd like to hear from you.

We're on the lookout for letters, articles or suggestions to make the Newsletter even better. You can use old-fashioned pen and ink, but we'd be delighted to receive Ascii or STOS Word files together with a printout.

Hope to hear from you soon!

*Aaron*

# Drawing numbers

Do you remember the good old days when arcade games were real arcade games like Asteroids and Space Wars? If so, do you remember the way all the graphics were drawn as vectors, including the scores? More recently, this was revived with the Star Wars games although this time colours were used! On some more ancient computers like the good old Apple [(happy sighs), the only way you could plot sprite-like shapes to the screen was by defining them as a series of vectors (although the screen was bitmapped so people soon started providing utilities to let you use normal raster graphics).

Anyway I digress! Sometimes vector drawn scores, numbers and lettering can be quite useful – for instance where you want your mega game to work in high resolution as well as low resolution with the minimum of fuss, and without having to have two different fonts in memory – vector numbers can be easily re-sized. Because they use the normal drawing routines they can be plotted anywhere on the screen (So you don't have to use sprites to draw numbers). They do however have the disadvantage of being a little bit slower – although not by too much.

The following program defines the vectors needed for the digits 0-9, and displays a counting number at the mouse position. The numbers are created from a grid of seven lines, like in a calculator. These lines go thus:



The arrays VX() and VY() contain the start and end co-ordinates of each of these lines from the start of the digit. These co-ordinates are multiplied by the variables SX and SY, so the numbers can easily be re-sized.

The array N() contains the data for each number, telling the routine which of the seven lines to use.

# LISTING

The subroutine at line 500 draws a single digit – first erasing the background by drawing a block behind it in colour 0, then drawing the digit in colour 1. The variables TX and TY hold the top left co-ordinate of the digit.

The routine at line 510 draws out the number contained in the variable N\$ at graphic co-ordinates TX,TY. It uses the routine at line 500 to draw the individual numbers and is shown as an example of modular programming with subroutines. If I wanted to change the way the individual digits were drawn, all I would have to do would be to change the routine at line 500-509 without having to worry about the string drawing routine at line 510. Similarly if I wanted to change the way the strings were drawn I wouldn't have to change the digit drawing routine. Notice also that I have turned autoback off. Try turning it on to see how much it slows the graphics down. Also try changing the values of the variables SX and SY as these contain the size (in pixels) of the lines used to draw the digits. The drawing routines used in this program are not perfect and it would be interesting to see what you can come up with in terms of faster, more efficient routines.

1 key off : curs off : cls

10 dim N(9,6),VX(6,1),VY(6,1) : for A=0 to 9 : for B=0 to 6 : read N(A,B) : next B : next A

19 rem Data for numbers

20 data 1,1,1,0,1,1,1

21 data 0,0,1,0,0,1,0

22 data 1,0,1,1,1,0,1

23 data 1,0,1,1,0,1,1

24 data 0,1,1,1,0,1,0

25 data 1,1,0,1,0,1,1

26 data 1,1,0,1,1,1,1

27 data 1,0,1,0,0,1,0

28 data 1,1,1,1,1,1,1

29 data 1,1,1,1,0,1,1

30 for A=0 to 6 : read

VX(A,0),VY(A,0),VX(A,1),VY(A,1) : next A

39 rem data for vectors

40 data 0,0,1,0

41 data 0,0,0,1

42 data 1,0,0,1

43 data 0,1,1,0

44 data 0,1,0,1

45 data 1,1,0,1

46 data 0,2,1,0

49 hide on

50 auto back off : ink 1 : SX=2 : SY=2 :

TX=10 : TY=16 : SCORE=0 60 inc SCORE :

N\$=right\$("00000"+str\$(SCORE),5) : TX=x

mouse : TY=y mouse : gosub 510 : goto

60

499 end

500 rem display number N, Xsize SX,

Ysize SY, Topleft TX,TY

501 ink 0 : bar TX,TY to TX+SX,TY+SY\*2 :

ink 1 : A=0 : repeat : if N(N,A)=1 then

draw TX+VX(A,0)\*SX,TY+VY(A,0)\*SY to

TX+(VX(A,0)+VX(A,1))\*SX,TY+(VY(A,0)+VY(A,1)

)\*SY

502 inc A : until A>6 : return

510 rem print string of numbers N\$ at co-

ordinates TX,TY

511 P=1 : repeat : N=val(mid\$(N\$,P,1)) :

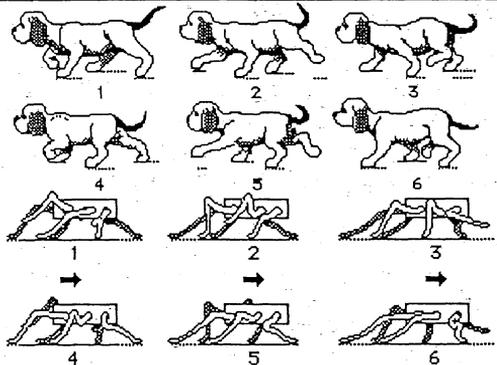
gosub 500 : inc P : TX=TX+SX\*2 : until

P>len(N\$) : return

## Walkies!

For those of you trying desperately to work out how to make creatures move in a realistic way, here are two examples which should be of use – and you'll find a ladybird on the next page. It is important to match feet between frames in order to convey the illusion of movement realistically.

If you decide to use these animations – or would like to see more, please drop us a line.



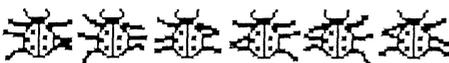
## More on Midi



As promised in the last issue of the Newsletter, here are some details on MIDI's controller commands. These are the commands MIDI uses for control of things like volume, sustain, modulation and so on. The command format (remember the last article) is \$Cn as the command byte and two data bytes. The first of the data bytes is the controller number and the second byte is the new value. Thus to set the volume of a synth on channel 3 to 1/2 normal, we would send the command \$C2 (Controller Command Channel 3), \$07 (Volume controller number), \$40 (range is 0-\$7F). Different synths have different controllers, and there are some really weird ones out there! Here are some of the most commonly used:

Controller	Value
Modulation Wheel	1
Breath Controller	2
Foot Modulation	4
<i>(assuming you want to modulate your foot)</i>	
Portamento Time	5 <i>(this is a fun one!)</i>
Data entry Slider	6
Volume	7
Pan	10
Sustain pedal	64
<i>(controllers 64-127 are on/off controllers)</i>	
Portamento on/off	65
All notes off	123
<i>(try this one to shut your synth up!)</i>	

A nice program to try might be one that sends a series of notes to a synth, along with random pan settings if your synth uses the pan controller, or for really freaky effects, random volume or modulation settings!



### *Communicating between games*

Multi-player games are great fun when they are played using a datalink between two computers. Currently I think Stunt Car Racer and F16 Combat Pilotrate as the best two data-linkable games. RVF Honda would be up there with them but it uses a completely non-standard cable which is not worth the hassle!

But what if you want your STOS game to be able to work multi-player? You could use the RS232 routines if you want (thus needing an expensive or awkward to make up Null Modem lead) or you could use the MIDI ports and STOS's MIDI routines to send the data between the two computers.

This has the advantage of being quicker than RS232, more convenient (MIDI cables are quite cheap), and MIDI cables can be run up to 15 metres without loss of data.

I showed you in the last issue how to communicate between two ST's, so all it needs are some minor modifications to send the data needed for a game to work on a MIDI link. If you think about it, what you need to send to the other computer is all the data that relates to what your computer is doing.

So in a maze game where you and another player are trying to reach the centre, each computer needs to know where both players are, so they can be displayed. So each computer must send the position of its player to the other. The difficult bit is synchronising the data transfer. So it is usual for both computers to send a series of values (1,2,3,4,5,6,7 etc) at the same time as checking MIDI In for a series of rising values. When each computer detects the series of values, it stops sending them and clears its MIDI buffer. Thus both computers are now ready to start the game. Every frame (after the moves have been made) each ST sends the co-ordinates of its player using something like:

# PUBLIC DOMAIN

open #3,"MIDI" Print#3,MYX :  
Print#3,MY Y

It then reads the data for the other computer's player from MIDI in by:

Input #3,HISX : Input #3,HIS Y

Note the lack of a semi-colon after the PRINTs - this is where STOS's MIDI annoyance is actually quite useful.

That's basically it, so try and convert one of your STOS games to multi player. Remember, start with a simple one and work your way up! ■

## FAKE VIRUS DATA

(see page 12)

Bank number : 10      Length : 256

Adrs (bold numbers) | Datas | Check (italic numbers)

```
0000 | 601A 0000 00B0 *0016
6000 0016 8000 *0012 | 34108
0034 | 41F9 0000 00B0 43F9
0000 0026 4E75 23C8 | 0F905
0044 | 0000 000C 33FC 0888
0000 0010 3F3C 0002 | 07F0E
0054 | 4E4E 548F 23C0 0000
0012 4279 0000 0016 | 1093E
0064 | 2079 0000 0456 4A98
66FC 217C 0000 0074 | 0F853
0074 | FFFC 41F9 0000 0008
43F9 0000 0072 45F9 | 1CC61
0084 | 0000 0005 47F9 0000
0006 4E75 4A79 0000 | 0E0F2
0094 | 0010 6808 5379 0000
0010 4E75 303A FF90 | 23CED
00A4 | 207A FF88 41F0 0000
7227 4298 51C9 FFFC | 36876
00B4 | 0640 00A0 B07C 7000
6502 7000 33C0 0000 | 23D1E
00C4 | 0016 4E75 *0006 001A
0608 080E 0610 0806 | 16A07
0008 | 0606 0808 2800 *0022
0000 0000 0000 0000 | 13630
```

## STOS PD Library

Listed on the next page you'll find a good selection of interesting programs with something for everyone. Each single-sided disc (SPD prefix) costs just £2 each - or £1 if you supply your own disc, and each double-sided disc (DPD prefix) costs £2.50. If you order three discs or more, deduct 20p per disc (including the first three). For example four single-sided discs will cost £1.80 each - £7.20 in total.

STOS Paint is shareware: £3 goes to the programmer (deduct £1 if you supply your own disc). Each disc has a specially-printed STOS Public Domain purple sticker kindly produced free-of-charge by Mandarin Software so your PD discs will match the rest of your STOS master discs.

In most cases the programs require you to boot your copy of STOS first - that way you can have a good look at the listings.

Many of the Swedish demos are rather lacking in documentation - some of the instructions are in Swedish! However you should be able to work out how to play the games.

Ring Sandra on      to find out about the latest public domain titles to be added to the library.

If you have any programs which may be suitable please send them along. You can also get something back for your efforts by making your submission shareware.

Send cheques, postal orders or stamps to:

Sandra Sharkey,  
Highfield,  
Wigan

This STOS Newsletter was edited by Aaron Fothergill. All articles are by Aaron unless otherwise stated. Layout by Chris Payne. Printing by Sandra Sharkey. (Contact Sandra if you could use her services).



## Choose from the following:

**SPD1:** *1st Serve Tennis* – Wimbledon revisited in this one-player against the computer tennis game.

**SPD2:** *STOS demo I* – The first demo created for use in shops – no great shakes but includes a nice tune you can grab.

**SPD3:** *STOS demo III* – Shows off the Compiler, Maestro and Sprites 600 using some excellent sampled sound. You do not need the Maestro extension to run this.

**SPD4:** *STOS listings* – All the programs and routines from issues 4, 5 and 6

**SPD5:** *New STOS upgrade*: Update your copy of STOS to version 2.5 so it works with TOS 1.6 and the new Atari STE.

**SPD6:** *Fun School II* (2 discs) – See two programs from each of the three age groups (Under 6s, 6-8s and Over 8s). Two single-sided discs.

**SPD7:** *Swedish I* – Two games submitted in the Swedish competition. Yatzy is a cleanly-designed version of the dice game Yahtzee with a good tune, and Virus is a flip screen maze game in which you are a cute-looking alien searching for missing 3.5" discs! Some great sprites for you to grab.

**SPD8:** *Swedish 2* – Fia is a version of Ludo controlled with the mouse pointer – good fun for kids. Mario is an infuriating platform game, and Rush is a 'shove-the-bricks' the solve the puzzle.

**SPD9:** *Swedish 3* – Saga Castle is an ambitious multi-screen platform game with lots of puzzles to solve. Stratego is a simple war game.

**SPD10:** *Swedish 4* – Frog Race is a complex game in which you bet on a number of excellently animated jumping frogs which race towards the finishing line. Upstart is a nicely put together shoot-'em-up.

**SPD11:** *Swedish 5* – Acid Remix is a spruced-up version of the old standard Snake game with some clever sampled sound at the beginning.

**SPD12:** *Give Us a Break* – An excellent version of the popular pub game based on DLT's radio show. Try to pot all the snooker balls by answer-

ing multiple-choice questions with difficulty related to the value of the ball you choose.

**SPD13:** *Home Finance* – Quite a good financial program.

**SPD14:** *Samples 1* – A marvellous collection of samples put together by Martin Walker, author of a number of commercial C64 games including Chameleon, Hunter's Moon and Citadel. He also did the music for Armalyte – and the samples include effects used in some of these games plus some newly-created ones.

**SPD15:** *Samples 2* – A wide collection of useful sounds for use in your games, recorded at 6Khz. Includes laser bolts, alarm, pulse, scream, warble, warp, take-off and tinkle – 20 in all.

**SPD16:** *Samples 3* – Eleven more samples including alarms, docking, fountain, hum, shoot, tanktrax, up/down, alarm and more.

**SPD17:** *Typing Tutor* – The winning entry in Issue 4's competition. Richard Gale's excellent program has 50+ exercises and a built-in typing game.

**SPD18:** *Hero* – A STOSified version of the 8-bit and VCS game from Activision. Fly round the caverns in your Hoverpack, dropping bombs to blow up walls.

**SPD19:** *Orbit II* – An updated, compiled version with sampled sound.

**SPD20:** *Zoltar II* – Shows what an improvement can be made by adding sampled sound and compiling.

**SPD21:** *New Atari User programs (Disc 1)* – A collection of listings from Peter Hickman's STOS Column, including Input Data and Output Data.

**SPD22:** *Samples 6* – Porsche Carrerra starting up and assorted sound effects from Dave Lee Travis' radio show (bugle, crash dive, train, revving etc)

**SPD23:** *STOS Paint* – A feature-packed art program written by Ralph Effemey which loads as an accessory so you can flip to it with ease. A unique feature allows you to paint with sprites from the sprite bank. Also on the disc is the source code for SPD3 (requires Maestro extension and one meg of memory). [Shareware: £5]

**SPD24:** *Tome demo* – Try out for yourself this sophisticated multi-screen map editor with built-

# Public Domain Library



in machine-code scrolling routine.

**SPD25:** *Skystrike demo* – Take to the skies in Aaron's action-packed Spitfire game.

**SPD26:** *PCP and Skidpan* – The full-screen editor which allows you to forget about line numbers and add labels and procedures to STOS. Includes Skidpan game.

**SPD27:** *Shipwreck* – An educational program which asks maths questions as the intrepid explorer rows, skis, climbs mountains in a sea lift and more.

**SPD28:** *Hot Dog Demo* – A giant juicy hot dog flies over a scrolling landscape with scrolling message – from Peter Hickman using TOME.

**SPD29:** *Raceway* – The full version of the vertically-scrolling car racing game featured in Issues 5/6. Requires 1Mb of ram.

**SPD30:** *Lines and Spirals* – Two menu-driven programs which draw pretty patterns on the screen.

**SPD31:** *Kiki Dee demo* – A simple demo which shows a picture of Kiki while playing a long sample of one of her songs.

**SPD32:** *Star Trek* – A beautifully-presented demo of the Next Generation series. Move the Treki-fied pointer to one of the many miniaturised Vidi digitised pictures and click to hear sampled speech or sound effects from the program. The excellent sounds can be grabbed to spruce up any SF-type game.

**SPD33:** *Fire + Xmas* – A hypnotic demo set to music, and a wide selection of Christmas tunes with accompanying graphics.

**SPD34:** *Soko* – A STOSified version of the puzzle game you can buy for the Nintendo and Game-boy. Thought-provoking fun for fans of Tetris and others.

**DPD1:** *Xmas demo* – A well-executed spoof of the nativity story with good animation.

**DPD2:** *Caves of Rigel* – A well-designed game converted from the Atari 8-bit commercial release on Atlantis Software by the original author, Ralph Effemey. The disc also includes another game from Ralph – A Froggy Day in London, based on the classic arcade game Frogger.

**DPD3:** *STOS demo II* – The cycling demo you may have seen at the shows which uses Ian

Waugh's attention-grabbing tunes from the Accessories disc.

**DPD4:** *Fun School II demo* – The full demo on one double-sided disc (see SPD6).

**DPD5:** *Samples 2* – The same as the samples on SPD15 but recorded at a higher sampling rate: 12kHz instead of 6kHz.

**DPD6:** *Samples 3* – Ditto for SPD16.

**DPD7:** *Thunderbirds* – An entertaining homage to the cult TV show with amusing animated sequences, great music and top class Maestro samples.

**DPD8:** *Phantom of the Opera* – A superb music demo which combines Vidi-digitised animated clips from the video of the classic Iron Maiden tune you heard in the last Lucozade ad.

**DPD9:** *Samples 4* – Arrows, fire, neck twist (?), pain, leg sawn off, thunder (from Richard Gale)

**DPD10:** *Samples 5* – Six screams, laughs, phantom, wolf, bats (from Richard Gale)

**DPD11:** *ST Wizard* – Spiralling sprites casting shadows on a checkerboard landscape, a scrolling message and a long sample of the Top of the Pops theme tune – from the keyboard of Richard Gale. Just great!

**DPD12:** *Blood Money* – A new version of Andrew Webb's adaptation of the theme music for this classic ST/Amiga game with scrolling message. 2.57Mb of music crammed into 350k by using clever looping and repeat sequences – four minutes of music in all.

**DPD13:** *Kylie demo* – Dance your heart out to a Maestro sampled version of the complete Locomotion tune which loads in from disc in chunks, uncompresses and plays away – clever machine-code programming by Bobby Earl.

**DPD14:** *Batdance demo* – The complete tune by Prince with excellent graphics, created using Bobby's routine.

**DPD15:** *Treasure Hunt* – Hunt for treasure in this clever educational coordinates game by Peter Hickman. Includes excellent graphics and sampled speech.

**DPD16:** *Daze Aster* – An involved adventure game with Vidi-digitised pictures and sampled sounds.

## **AMOS amazes at Show**

The Computer Shopper Show at Alexandra Palace at the end of November saw the first public showing of AMOS – the version of STOS for the Commodore Amiga.

Available in late February, AMOS contains all the features which made STOS such a winner – plus a whole host of new features. By pushing the blitter chip to its limits François Lionet has managed to allow up to 440 sprites animating on screen at once (though they will be very small)!

It will be possible, using a special PD program, to insert an ST disc containing an Ascii version of a STOS program into the Amiga drive and read it into AMOS via a convert program – to produce an almost instant AMOS game! Programs will, though, require minor modifications to work properly in AMOS.

## **Cartoon fun**

The latest version of Cartoon Capers was on show at Allie Pallie – and very nice it looks too. The game can be played by one or two players on joystick – and the specially written two-joystick routine (written by Bobby Earl, author of STOS Squasher) will be provided free with the package. The game features a full digitised version of the famous Loony Tunes melody, exclusively licenced by Mandarin from Warner Brothers.

Cartoon Capers' programmer, Simon Cook is already working on the Amiga version using an early developer's version of AMOS. The game will go on sale in February for £19.95.

## **Musical package delayed**

STOS Musician, Mandarin's newest STOS add-on has been delayed until March. The package will include a brand new music routine written by Bobby Earl (again!). The clever routine produces higher quality sound out of the ST's notoriously ill-respected sound chip.

STOS Musician will come with an icon-driven music editor and more than 100 tunes – and sell for just £14.95.

## **STOS games on sale**

Games Galore, a collection of STOS games was released by Mandarin in November – and the company is delighted with the response from retailers and the public. The four games boot straight from disc (STOS is not required) and really show off the power of STOS.

If you have a double-sided drive you can click on the Flip icon to load the Basic, sprite and music files stored on side two (single-sided disc drive owners can send for a disc with the files). This way you can see how the games were put together, modify them if you wish, grab the sprites or background graphics for use in your own games – and also grab the music files – Skate Tribe alone features 25 different tunes, and every one is a veritable gem!

This is a package that's not to be missed: Four great games for £19.95 – and all the other benefits as well!

## **Video times**

Visitors to the Computer Shopper Show also got a glimpse of another new release from Mandarin. STOS Vidi Digitiser consists of Rombo's acclaimed hardware, a STOS extension file which adds a series of new commands to STOS, and a special editor program written in STOS (of course).

Using the package you'll be able to grab graphics in realtime from video or TV. Grab the castle from Trap Door to produce a creepy title page, grab the howling wolf from the Lamot advert to introduce a werewolf game.

Spruce up an educational program by adding Winnie the Pooh jumping up and down if your child gets the right answers. Grab a running character and load him into the sprite editor to add some realistic movement into your games... All this is possible and more. Watch this space!

STOS Video Digitiser should be on sale by March for £99.95. If you already own Rombo's Vidi Digitiser you will be able to obtain the software from Mandarin Software in March (but get in contact now with your wish list of features!)