STOS Newsletter

Issue 4 • STOS User Club

Serious STOS

The May 11 issue of The Guardian included a feature titled Playing at the Dentist by computer journalist Mike Gerrard looking at how STOS had been put to serious use. Darryl Ithell, a talented games writer, had used STOS to write ***Dental Practisioner***, a database for dentists, and Keith Russell-Smith had written ***The InveSTor*** to keep track of investments. The article also discussed the use of STOS in writing the Atari ST version of Fun School 2 from Database Educational Software. Drop us a line if you have put STOS to serious use.

STOS on MicroLink

Keen STOS owners can now exchange hints and tips on the STOS closed user group on MicroLink which is being run by Dave McLachlan, the artist responsible for the graphics in STOS, STOS Compiler, STOS Sprites 600, Starquake (ST) and Fun School 2 (ST). All you need to participate is a modem, comms software and membership to MicroLink. You can get further information by writing to MicroLink at Europa House, Adlington Park,

STOS on PD

Goodman's PDL has collected together a number of public domain STOS programs including a collection of Christmas carols with simply-animated graphics and ***Fire*** (an interesting and varied graphical demo with music). Contact Mike Goodman

STOS for the Amiga

You may be interested to know that AMOS, the Amiga version of STOS, is on target for a late September launch. François Lionet will be sticking pretty closely to the STOS standard but he has decided to dispense with line numbers and make AMOS more structured.

STOS goes abroad

The French and German versions of STOS will soon be available making the package available to a further 200,000 ST owners. Ubisoft will launch STOS in France and Ariolasoft will handle STOS in Germany where the Mandarin Software logo will be replaced by the Database Software logo – that's the Germans for you! The packaging will emphasise that STOS will work in all three modes – this is especially important as 90% of STs in Germany are sold with a monochrome monitor!

£5,000 awaits!

June 13 is the closing date for the STOS Gameswriter of the Year Award. More than 100 entries have been received so far with games ranging from arcade to Defender of the Crown and Dungeon Master types. The winning entry will receive £5,000 as an advance on Mandarin's standard royalty rate. Some of the other games may appear on compilations, or the public domain. Do send your games along – it doesn't matter what state the game is in: What Mandarin are looking for is good gameplay – graphics will be sorted out later. Many of the entries will be shown at the Atari User Show at Alexandra Palace, London.

LETTERS

MIDI PORTS

I have a query regarding the midi ports. Input is very easy using the PORT(#N) function. For output with ST Basic, OUT(3),N used to work quite well but PRINT #N,X doesn't seem to do anything with STOS. Please can you tell me what I am doing wrong?
DANIEL SHAW, GUERNSEY.

Version 2.3 of STOS did have a problem with outputting characters using the PRINT #N,X command. It has been fixed in version 2.4 – this update is included on the compiler disc.

PRICE PUZZLE

Having looked through the magazine I noticed the product news sheet, in particular the STOS Compiler. Will this compile programs completely into machine code and will the finished program still require the STOS folder to run?

I also noticed the STOS sound sampler. On the front of the magazine it says that the software and hardware will each cost £24.95 at a bundled price of £49.90. However on the product news sheet it says the price will be approximately £70. I find this very puzzling and so I would be grateful if you could send me full details of price and facts about the sampler and software.
COLIN STRICKLAND, WARRINGTON

The compiler can produce machine code programs runnable either with the STOS folder (handy if you have lots of short programs on one disc) or as a straight GEM program which incorporates the portion of STOS needed to run.

The final prices for the various STOS add-ons are as follows:

STOS SPRITES 600	*£14.95*
STOS MAESTRO	*£24.95 (software only)*
STOS MAESTRO PLUS	*£69.95 (software & hardware)*
STOS COMPILER	*£19.95.*

All are available now and the best discounts I have seen are from Special Reserve (Tel: 0279 600204) whose prices for the above are respectively £8.49, £12.99, £33.99 and £10.99. To qualify for these prices (which don't include postage) you need to be a member of Special Reserve which costs £4 for the year, and please mention that I sent you – Pat

The price of £49.90 was a very early guess at the price of Maestro Plus – until we did our costings properly, taking into account that we have to give a whopping discount to distributors! – Chris Payne, Mandarin

GIANT SPRITES

Is it possible to use larger sprites if accessing the sprite commands direct from machine code? I would also like some advice about including STOS 'trap' calls in machine-code programs.
S H POSKITT, BASILDON

It is possible to merge sprites together to form larger sprites – you will find the listing in another part of this issue.

See elsewhere in the newsletter for 'trap' information.

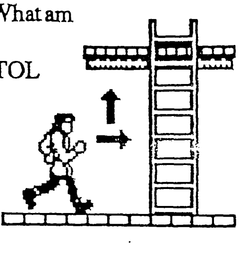
NEVERENDING STORY

I am designing a simple 'multiple choice'-type adventure but having problems due to limited memory. The game needs lots of 64*64 sprites. I think a 1040STFM is going to be my next buy, then perhaps a printer (when will it stop?).
C LLOYD,BATH

FAST AS SNAILS

How can you read the mouse port as though a joystick is plugged in it? In the User Guide it states the sprites run under interrupt therefore not affecting your program speed. As soon as I put a few sprites on the screen my program slows down – even 15 16*16 pixel sized ones slow my program down to a crawl. What am I doing wrong?
MARK HULA, BRISTOL

At present STOS cannot use both joysticks. The best way to attempt it is to set up your own keyboard interrupt



routine.

Moving 15 sprites on the screen at once will slow your program down because each successive sprite generates longer interrupts. Either limit the number you have moving at one time, make your control algorithm more efficient or try using the STOS Compiler.

CONTACTS WANTED

As I got into computing via music and what it can do I would hope that the club will be able to help me in getting further into programming, then one day...! In future, might it be possible to send out a regional list of members as at the moment I do not know anyone with STOS or an ST?

MICHAEL BUSH,
SURREY,

If anyone wants to contact other STOS users, either locally or nationally, I'll be happy to publish their details as space permits.

SPRITE TIPS

I would like to submit the following tips based on my early experiences with the sprite editor (low res).

I loaded up STOS on day one and didn't like the white text on black background so promptly altered this to the converse using the "config.bas" file as described on page 21 of the manual. Great! – black text on a white background. Next I loaded the sprite editor (low res) and found to my dismay that the foreground was dark grey while the background was black. Hmmm...

Then I loaded up "sprite.acb" and listed the file. Line 35 is the key, so edit the info after "palette". May I suggest the following: $677, $0, $531, $700, $740, $770, $60, $74, $7, $47, $77, $707, $747, $444, $555, $333.

This gives a palette with a nice range of colours that I have found adequate for the (very limited) sprites I have created so far. As long as there is good contrast between colours 0 & 1, 0 & 3, and 15 & 16 then the sprite editor will always be clearly visible.

A final tip: If you have a multi-coloured sprite in the editor, select RGB and then as you move the mouse over the sprite the colours will be indicated in the palette box.

ALAN WOODS, WARRINGTON

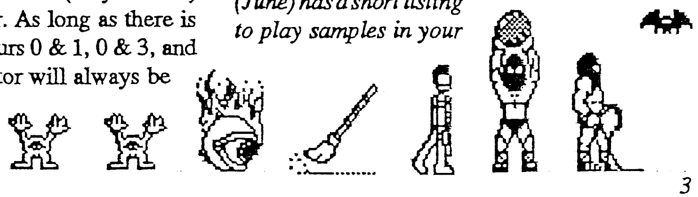
FLASH OFF!

Here's a good one for you. Right now I'm doing a version of Q*Bert – you know, the one where you bounce around a pyramid made of cubes? Well I'm using the PUT SPRITE command to change the colour of the sprite that has just been bounced on. With me so far? Now, here's the crunch *(shouldn't that be boing? Pat)*. Sometimes, the cube that was previously bounced on reverts back to its original (un-bounced) status. Why does this happen? I thought that PUT SPRITE simply puts an immovable copy of the graphic information of that sprite at its present position. So why does it move and what am I doing wrong? And why is Mandarin being so bitchy towards the programmers of Microdeal's ST Replay 4? I know that STOS has its own sampling package now but what about us poor sods who've already got a sampler? I'm not going to spend about £70 plus on another sampler! Sheesh! Just one more thing: The cursor. If there is ever a revised version of STOS then please get it into the programmer's head not to put a cute multi-coloured strobing cursor in the editor. It might look nice, but it gets to be a bit of a pain in the butt when it starts affecting palettes. Hmph!

ALAN HOLDING, MANCHESTER

The PUT SPRITE command works as long as the following set of instructions are used with it:
sprite n,x,y,frame : update : put sprite n

The Maestro software can handle samples produced by Prosound, IS and Replay. The Replay cartridge will even work directly with Maestro software. ST/AMIGA FORMAT (June) has a short listing to play samples in your



own programs.

SCROLLING MESSAGES

I would like to know how to construct a scrolling message. Would I need to construct a message from sprites or will a text string be needed?

I have just started to program with STOS and I have used a scrolling message routine in Fast Basic but I haven't a clue how to go round to converting it.

ALASTAIR CRAIG, THAMES, READING.

I've never used Fast Basic. Can anyone give Alastair a hand to convert his work?

WINDOWS

In Newsletter 3 (page 4) "Looping the Loop" seemed the answer to a problem I have in a shareware program I'm writing. Having tried it as printed it doesn't actually work, the window being a complete screen clear still happens. So back to the drawing board.

I came up with the listing below which treats a block of the screen as a window without using the window command. Providing there is judicious use of LOCATE and semi-colons at the end of print strings, together with space filling for empty portions of the "window", it works wonderfully. (Well I think so, and I'm going to use it in my program.)

```
10 clw:curs off:flash off
40 locate 10,5
50 print "";
60 load "ntrek.neo",back
70 limit mouse 0,148 to 319,199
80 hide on
90 B$=screen$(physic,55,30 to 260,150)
100 show on
110 for I=1 to 8
120 locate 10,5+I
130 print "klingons";
140 next I
150 repeat: until mouse key <>0
160 locate 10,5: print "";
170 screen$(logic,55,30)=B$
180 wait key
190 goto 110
```

To P.J.T Butler who says you can't use multi-condition UNTILs: They do work – well, mine do!

Les Barclay: Printer 'on-line check'... Use ON ERROR routine. Send "" (null) to printer trap for error number 10 message to put online, wait key, resume. Simple(?)

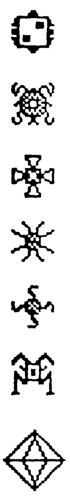
I put a compiled program in an auto folder and watched it fall over on the first move with a bus error on peek or poke (compiler bug?).

A note for the mag about the compiler: The picture compactor needs running through "CONVERT.BAS" to change the real number precision. They don't tell you that in the manual! I think the other STOS software is okay.

I got the STOS Sprites 600 disk, but was a bit disappointed in the size of some of them However, the palettes will come in very useful!

I'm still not sure about the random record editing – my file gets screwed up by the first field being re-written. It could be something to do with adding an extra set of chars (still below Field max size) upsetting the key regions. Or it could be just me doing some daft thing!

ERIC ELSON, GUILDFORD



If you find sprites are the wrong size use the zoom and reduce options in the sprite editor. No idea on the problem with compiled programs in the auto folder. Has anyone else experienced this?

WIGAN V EPSON???

I am writing this on my Hisoft Editor – it's very pleasant to work with, and I trust Ascii. This is a pity as I like 1st Word/Thunder. You will have to check the spelling.

There are two complaints I have about STOS. First it was a pity to have to emphasise the games aspect when it is such an outstanding interpreter in all respects. Secondly the use of line numbers, although I admit to finding them very handy.

When I think of the gallons of ink used by the magazines in the early days of Fast(?) Basic to sort out the problems of menus and windows each of which took dozens of lines – things that STOS accomplishes in a few lines.

One short comment on the wonders of writing and auto-loading accessories in an amazingly simple manner. I use one liners to set my printer for listing, setting time and date etc.

Cannot wait to see the compiler. Hope it handles floats and compiles fairly compact and fast code.

Last ramblings: How good to see the excellent structures of the longer programs already printed in our mag. Despite the critics (and the bad examples set in the STOS handbook) we are not being strangled with GOTOs. Mind you, it's a bit tricky to see the difference between an IF...GOTO line and an IF...LABEL except the line is easier to find.

Congratulations on the last mag – it was great. And didn't Wigan do well with that funny shaped ball? Like a circle dumped on my Epson!

LES BARCLAY, ACTON

Chris Payne from Mandarin comments: We at Mandarin decided to market STOS as a Game Creator to set it apart from the many other Basics already available at the time for the Atari ST. Luckily many ST owners have discovered for themselves that STOS is just as good for writing serious applications too. On the subject of line numbers, François Lionet designed STOS this way as he felt that many ST owners would be upgrading from 8-bit micros and that line numbers were what they were used to.

SPEEDY KEYS

Does anyone out there know how to prevent a key repeat occurring whenever a key is held down for more than about half a second? The usual way to do this would probably be as follows:

```
100 Q$=Inkey$: If Q$<>"" then 100
110 Q$=Inkey$: If Q$="" then 110
```

Line 100 waits for a key to be released and line 110 waits for a key to be pressed. This routine, however, doesn't seem to work using STOS.

The only way so far that I've found to overcome this problem is to alter the key speed setting from within a program with a line such as:

```
100 key speed 100,100: rem Default = 2,15
```

The drawback with this method is during program development when, after the program has run, editing is slowed down unless the key speed is reset every time. Any ideas?

BRIAN PROBYN, BIRMINGHAM

The ST can only produce the current state of a key as it is being pressed. You cannot check for a release!

Using the configuration program, assign one of your function keys as KEY SPEED 2,15 then use this key each time you break out of your program for editing. Why doesn't the DEFAULT command reset the key speed??

SEEING STARS

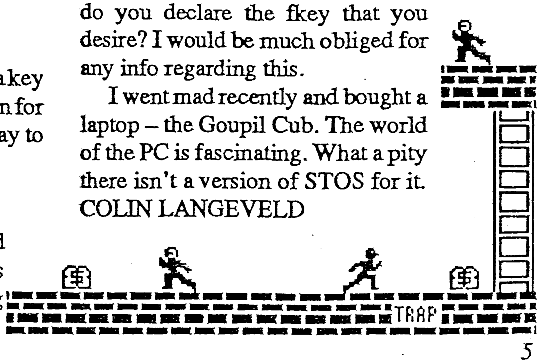
At the moment I am attempting to write an astronomical program. The program opens with a menu. You have the choice of looking at the solar system or some of the stars in the night sky. My main stumbling block is how to jump to the various sections of the program. I had no problem with my old Commodore 64. The Geta$ function worked very well – i.e:

```
20 Geta$: If a$="s" then gosub n
30 If a$<>"s" then 20
```

The trouble is the INKEY$ function does not appear to do the same thing. (Is it supposed to?) As an alternative I turned to the FKEY function: ON FKEY GOSUB etc... but the documentation on this is vague to say the least. For instance how do you declare the fkey that you desire? I would be much obliged for any info regarding this.

I went mad recently and bought a laptop – the Goupil Cub. The world of the PC is fascinating. What a pity there isn't a version of STOS for it.

COLIN LANGEVELD



LETTERS

The following short listings allow any input to send the program to a subroutine. The first can be used for any alphanumeric key while the second uses the "f" keys.

INKEY$ DEMO

```
10 while K$=""
20 K$=inkey$
30 wend
40 if K$="s" then gosub 100
50 if K$="t" then gosub 110
60 K$="" : goto 10
100 print "You pressed 's'" : return
110 print "You pressed 't'" : return
```

FKEY DEMO

```
10 on fkey gosub 100,200,300
20 goto 10
100 print "Fkey 1" : return
200 print "fkey 2" : return
300 print "Fkey 3" : return
```

HELLO TEACHER

Perhaps I may be able to assist you as I noticed from one of your replies in the Newsletter that you are interested in software for young children.

I have similar interests and as a primary school teacher I would like to develop educational software using STOS. You mentioned in the newsletter that you would like to get together with a teacher. I would be happy to help in whatever way possible.

However I am only just finding my way around STOS at the moment and could do with some practical assistance myself. If you feel that I can be of any help please do not hesitate to get in touch with me.

IAN HOLDEN, LANCS

MAESTRO BUG!

There is a small bug in the Maestro compiler extension. To solve it, use the procedure below:

1. Load up STOS Basic
2. Load in the INSTALL.BAS program
3. Type: poke start(11)+$179f,4
4. Now re-save INSTALL.BAS

The bug affected any samspeed commands which used the parameter of 5KHz. The line would work fine in Basic but when compiled it generated an ILLEGAL FUNCTION CALL error. The compiler extension was not checking the parameter range correctly.

Users must re-use the INSTALL program to generate a new Compiler extension.

RICHARD VANNER, MANDARIN SOFTWARE ■

Add speech to STOS!

David Thomson from Paisley in Scotland has successfully managed to get FastBasic's speech module to work in STOS following on from Les Barclay's letter in Issue 3 of the Newsletter.

He had previously written a short assembler routine to access it from C, but discovered that it was incredibly easy to write it for STOS as the machine level interface is so straight forward.

To use the Fast Basic speech system, run SPEAK.PRG as supplied with FastBasic, then STOS and try the following program:

```
1000 SPEECH$ = "Hello world"
1100 SPEECH$=SPEECH$+CHR$(0)
1200 dreg(0)=1
1300 areg(0)=varptr(SPEECH$)
1400 trap 8
1500 input "Enter some text ";
SPEECH$
1600 goto 1100
```

Says David: The Fast Basic speech module can be distributed with any FastBasic program, so I imagine it must be public domain, although I am not sure.

I have SPEAKTEX so I'll have a go at getting that to work. I suspect that all that is required is to LOAD "SPEAKTEX.PRG", bankno and identify where to enter the program. Anybody got any documentation for SPEAKTEX?

Getting the most from STOS

Aaron Fothergill offers some useful hints and tips for all STOS users

Yo STOS people. Greetings from the Shadow Software team. The team are myself, as programmer, musician and general bod, and my brother Adam who does all the graphics and bass playing.

Shado Software specialises in custom-designed synthesiser editors and completely addictive games that we don't release commercially although the STOS ones (ie. most of the stuff we do from now on) will be available through the Club.

First things first though – another bug on the later versions of Bullet Train (at least the version I received). This is the version where all the REMs have been removed to get it into the 520's memory. Someone seems to have goofed up and changed line 12310 which reads..

```
12310 SENS=0:If jup then SENS=1
```

but should read...

```
12310 SENS=0: If jup then SENS=-1
```

... thus helping those of you who wondered why the train wouldn't steer upwards.

There is also a bug with the sprite editor. Anyone who has tried to save a sprite bank with more than 128 sprites in it will have had problems with the editor just not doing it! This is because of a bug in line 8445. I have corrected it to.....

```
8445 If SPRNB then locate
OPXT,OPYT+3: print "Out of";SPRNB;
:for N=0 to SPRNB-1: locate
OPXT,OPYT+2: print "Number ";N+1;
:A$=string$(chr$(N mod 128)+$80), ((len
(S$(N))-8)*5)/4): areg(1)=varptr(S$(N)):
areg(2)=varptr(a$): dreg(0)=4: call 15:
print #1,a$; : next N
```

(*Phew!... Pat)* Pretty horrifying stuff eh! The problem was the A$=string$... etc in the original line. N is the sprite number +1 and a sprite of 128 or greater would cause an illegal value for the chr$() function ($80 being 128 decimal). The bug is fixed by the MOD 128 making it so that the sprite number shoved in the string is never more than 127. This causes no problems with the sprites as far as I can work out – the number is merely used as some sort of reference number and is not relevant to any later use.

I have managed to find a couple of bugs in STOS itself (or at least the manual). The TAB(n) function doesn't work at all like it does in a normal version of Basic. Normally PRINT TAB(n) should put the cursor at the nth column for printing. However in STOS PRINT TAB(n) moves the cursor right by n columns making life awkward for anyone using a printer as you can't normally use the LOCATE x,y function with a printer (or the XCURS variable) so printing accurate columns is difficult. A way around this is by defining a string of 80 spaces (for 80 column printers or screens) and using the MID$(string,n1,n2)="your string" function to place your data into the string at the relevant column positions. Eg:

```
310 L$+space$(80):P$="TITLE":
mid$(L$,4,len(P4))=p$: P$="COST" :
mid$(L$,50,len(P$))=P$: lprint L$
```

This will put the strings "TITLE" and "COST" to the printer at columns 4 and 50 of the same line and is equivalent to the line:

```
lprint tab(4);"TITLE";tab(50);"COST"
```

... in normal Basic.

The other STOS bug I have found concerns random access files. When defining fields using the FIELD #n, 15 AS a$, 23 AS b$ etc command you have to add one to the length of the fields if you intend to use that length. Otherwise the field will not contain the data it is supposed to. Eg:

```
field #1, 15 as A$, 13 as B$
A$=space$(15):B$=space$(13) put #1
```

... will not work, but the following will...

```
Field #1, 16 as A$, 14 as B$
A$=space$(15): B$=space$(13) put #1
```

This is due to an 'off by one bug' (OBOB) a well-known gremlin to most programmers. *(Apt anagram too.... Pat!)*

Good news if you are trying to use two joysticks with STOS. Either use the intelligent keyboard commands and write a machine-code routine to read the second joystick, or use the XMOUSE and YMOUSE variables. Whenever the second joystick is moved these mouse variables are changed. If, for instance, the joystick is moved left the YMOUSE is decremented. When moved right YMOUSE is incremented. XMOUSE reacts similarly to up and down movements. However the variable does not return to normal when the joystick is centered and they do not continuously update if the stick is held at a position. You may find it useful though while I work on the previously-mentioned machine-code routine for the PD library.

Anyone having memory problems with a 520ST raise your hands. Well from my room I can count several thousand hands waving in the air. You can put them down now!

If you don't need to use them you can delete COMPACT.EXA and FLOAT.BIN from your STOS folder (getting about 18k back). However the sprite editor and any program using screen compaction use the COMPACT.EXA file and the FLOAT.BIN floating point routines are useful sometimes so you may need to keep them. <u>Whatever you do, don't remove any files from your master disc</u>–only remove these files from a copy of the master, which you then boot up (saving 32k of memory by ignoring Gem).

Another way of saving memory when you have a big program in memory (apart from removing all REM statements) is to replace any large integer arrays with direct memory access via peek and poke. For instance, if you have a 500 element integer variable array it takes up 2000 bytes (four for each element). If you are only using byte-sized values (ranging 0-255 or -128 to 127) then you can put the array into only 500 bytes. If the values are word size (0-65535) then the array goes in 1000 bytes. Long-word values take up the same space as the normal array (as STOS uses integer variable of long-word length). The only problem with this method is that all references to the variable within the program must be replaced by peeks and pokes. So to replace the A() array in the following program:

```
10 dim A(500)
20 for Z=0 to 500: A(Z)=rnd(25): next Z
30 while inkey$=""
40 inc A(rnd(500))
50 wend:end
```

...you would do ...

```
10 reserve as work 5,500: A9=start(5)
20 for Z=0 to 500: poke A9+Z,rnd(25):
next Z
30 while inkey$=""
40 Z=rnd(500): poke A9+Z,peek(A9+Z)+1
50 wend:end
```

I normally use the 9 after the variable name for a pointer as the 9 is the unshifted '(' which you would normally use with an array element. So the variable A9 points to the start of the memory block containing the A() data array. Multi dimensional arrays can also be done...

```
dim A(24,49)
```

... becomes

```
reserve as work 5,1500: A9=start(5)
```

... (1500 is 30*50, or the size of the array)

...and a reference such as

```
B=A(X,Y)
```

becomes

```
B=peek(A9+X*30+Y)
```

... (30 being the size of the second part of the array).

This can be expanded to encompass arrays of any size and number of dimensions. The direct memory access method can be awkward to program but by applying it to one of my programs I saved 48k!

For Charlie Furness who wanted to do something on his ST that his grandson can't do on the BBC try:

```
10 dim A(255,150)
```

...and run it.

I know it isn't exactly amazing stuff but it does prove a point!

By the way I'm working on a MIDI-to-STOS format sequencer program so that you can enter music in real time (as you play it) from a MIDI keyboard, the ST keyboard or even another MIDI sequencer and convert the music so that it can be used by the STOS music routines. It will be available as soon as I've finished it through the STOS club at a price yet to be decided.

If you need help with your STOS programming (or know any good jokes) write to us at

address removed

Invisible Space Invaders

This is a short game written by Aaron Fothergill which should confuse a few people who you show it to! Type it in first, then look at the explanation.

```
10 rem ****************************
20 rem Invisible Space Invaders
30 rem Aaron Fothergill
40 rem Shadow Software
50 rem ****************************
60 curs off : fade 1
70 volume 15
80 mode 0 : cls back : cls physic
90 hide on
100 reserve as screen 5
110 gosub 370
120 default : mode 0 : cls back
130 def scroll 1,0,0 to 320,160,0,4
140 TT=rnd(1000)+100 : timer=0
150 for A=1 to 100 : ink (rnd(13)+1) : plot
rnd(319),rnd(160),1 : next A
160 X=160 : XO=X
170 gr writing 3 : ink 5 :polygonX-5,190
to X,180 to X+5,190 to X- 5,190 : gr
writing 1
180 scroll 1 : screen copy physic to back
: ink 0 : bar 0,0 to 319,4 : for A=1 to 2 :
ink rnd(14)+1 : plot rnd(319),rnd(4) : next
A
190 ALFIRE=rnd(40)>30
200 gr writing 3 : ink 5:polygon XO-5,190
to XO,180 to XO+5,190 to XO-5,190 :
XO=X
210 polygon X-5,190 to X,180 to X+5,190
to X-5,190 : gr writing 1
220 if jleft and X>9 then X=X-4
230 if jright and X<303 then X=X+4
240 if mouse key=2 then shoot : wait 2 :
gosub 290
250 if timer>=TT then 330
260 if ALFIRE then boom : TT=TT-20
270 play 2+ATG,1 : ATG=10-ATG :
volume 16
280 goto 180
290 if rnd(1) and ALFIRE then boom :
SCRE=SCRE+100
300 if rnd(1) then ALFIRE=1
310 SCRE=SCRE+rnd(20)*abs(ALFIRE)
320 return
330 boom : appear 5,3 : M$="You scored
"+str$(SCRE)
340 paper 2 : pen 1 : locate 0,9 : centre
M$
350 locate 0,18 : centre "Press Anything"
360 while mouse key=0 and inkey$="" :
wend : run
370 curs off : cls physic,2 :
clsback,2:paper 2 : pen 0 : locate 0,8 :
centre "GAME OVER" : screen copy
physic to 5
380 return
```

HOW IT WORKS

Up to line 100 everything is normal, getting rid of the text cursor, fading the screen to black, putting the volume up, setting the screen to low resolution mode, clearing the screens and hiding the mouse.

Line 100 reserves a memory block to store a screen in (the screen with GAME OVER on it) so that we can use it later. At line 110 we jump to a subroutine at line 370 which draws the GAME OVER screen and copies it to the memory bank we reserved in line 100. NB: No-one can see this screen being drawn as all the colours are set to black.

Line 120 resets the viewable screens to black (Note the default command. Doing this and then a mode 0 command saves having to type in a long line of values for a palette command *[...but if like me your STOS is configured as black text on white background you'll get weird colours... Pat]*).

Line 130 sets up the scroll we are going to use to move the star background, while line 140 sets a variable TT to a random amount between 100 and 1100 as well as resetting the timer. TT contains the time in 50ths of a second that the game will play before you get a GAME OVER display and a random score.

Line 150 is quite useful for space games. It plots 100 random stars on the screen (random colours make the stars look more realistic than just white).

Line 160 sets the variables X (horizontal position of the ship) and XO (last position of the ship).

As the ship is going to be plotted in XOR mode line 170 draws the ship ready to be erased as soon as it gets to line 200.

Line 180 is the start of the main program loop.

First it scrolls the star field and copies the physical screen (the one it scrolled) to the background one. It then blanks the top of the screen (where it scrolled down) and bungs in a couple more stars.

Line 190 sets the variable ALFIRE to -1,30 out of 41 times. Line 200 erases the ship from its old position by XORing its shape over the old position and then sets XO to X (as the position in X is now about to become the last plotted position). Line 210 draws the ship in its new position.

Lines 220 and 230 check for the joystick and move the ship accordingly.

Line 240 is a con. If the joystick button is pressed the computer makes a shooting noise and goes to the subroutine at 290 where, if ALFIRE is set and a random 0 or 1 comes up as 1 then you get an explosion sound and the score goes up by 100. Next if another 0 or 1 random comes up as 1 ALFIRE is set to -1 anyway, then 0-20 is added to the score if ALFIRE is set before returning.

Line 250 checks the timer. If it is greater than or equal to TT then the program jumps to line 330 where GAME OVER and the score are displayed. Line 260 is another con. If ALFIRE is set then an explosion is heard and the time you have left is dropped by 2/5ths of a second.

Line 270 makes a menacing alien-type drone using ATG as a toggle value (it constantly alternates between 10 and 0 using ATG=10-ATG).

Line 280 returns the main loop back to line 180.

STOS Hotline

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

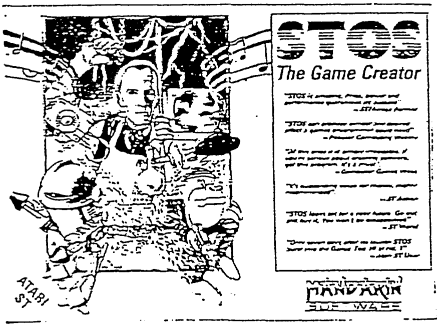
Ring her on

or write to

STOS at Boots

You may be interested to learn that STOS is available in a different box at selected branches of Boots. The box is the same size but the cover is in 'landscape' format as opposed to 'portrait' format and includes a series of quotes from reviews of STOS that appeared since its launch last September.

The reason for the difference in packaging is that Boots wanted software titles to conform to one of a limited range of sizes and they wouldn't accept the portrait format. Since then Boots, Smiths and Menzies have decreed that they won't accept any A5 boxes at all in future. This is why STOS Compiler, Maestro and Maestro Plus are packaged in what are called 'softboxes', and why most other titles from other software houses has also changed. The only problem from Mandarin's point of view is that AMOS will have to be produced in this new smaller-sized box – which will make the User Guide an awful lot thicker, and more expensive to produce. AMOS will be one heavy product!



Music maestro, please!

Pat Winstanley samples the delights of STOS Maestro

At last it's here! The STOS-compatible sampler that everyone has been waiting for. So how does it measure up? Will you be able to get what you want out of it?

OPTIONS

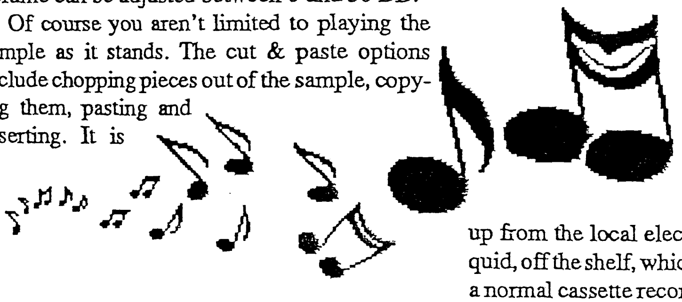
Two versions of Maestro are available. The simpler version consists of software only and costs £24.95 while Maestro Plus consists of both software and the hardware needed to grab your own samples. This costs £69.95.

Looking at the software first, here's what you get: The heart of Maestro is a program run not from STOS but from the Gem desktop. This is the editor which allows you to load in a sample either from disc or (if you have Maestro Plus) from a suitable outside source such as a Walkman or video.

Once loaded the sample can then be manipulated in a variety of different ways. The whole sample is shown on the screen but if you want to concentrate on just a small portion, perhaps to isolate a particular word from a sentence, the portion in question can be magnified to fill the screen.

Having identified the section you want to play about with a wide variety of options are available. The sample can be played forwards or backwards, either single play or looping backwards, forwards or alternately. The playback rate can also be varied from 5 to 32 KHZ, while the volume can be adjusted between 0 and 30 DB.

Of course you aren't limited to playing the sample as it stands. The cut & paste options include chopping pieces out of the sample, copying them, pasting and inserting. It is also possible to mix the sample with itself to produce the N-N-Nineteen effect.



Other manipulative options are available under the FX menu. These are echo, three reverb effects equating to large, small and normal rooms, amplify, fade in and out, soften and filter. A very useful extra is pack which effectively halves the memory taken up by a sample. Since samples, even short ones, devour memory like it's going out of fashion, it's worth packing everything as far as possible.

The Specials menu contains options which have no effect on the sample, but allow you to see what is going on in different display styles.

So much for the editor, but what else comes in the software package?

One of the discs contains samples of ten musical instruments ranging from electric guitars to pan pipes, eighteen spot effects from explosions to helicopters, twelve synthesised sounds ranging from 'weird' to 'clockwork' and eleven voice samples which include "game over", "bonus points" etc. These voices would probably suffice for most simple arcade games without having to resort to using the hardware to sample your own.

For Plus owners there are two programs for use with the sampler hardware. The Disco unit is a "real-time digital delay program" which plays an original sound entering the cartridge then plays it again over the top after a delay. The Phaser is a short program, again for use with the cartridge, which produces sweeping effects through the harmonics of the sample being played.

Maestro owners with a double sided drive can also access a couple of demos which play pretty tunes.

The cartridge which comes with Maestro Plus is a small box which plugs into the cartridge port of your ST. A cable (not provided) is required to link with your sound source. I picked one up from the local elecrical shop for a couple of quid, off the shelf, which connects directly to both a normal cassette recorder and a Walkman. With the right cable you should be able to link up your CD, video or TV too! Using the sampling cartridge is simple. Click once on a button on the Maestro editor screen then press play on the sound source.

The software also includes a couple of STOS Basic programs, one of which is a drum machine while the other is a utility to use the ST as a simple keyboard. Both programs can be fiddled with as desired!

INTERFACING SAMPLES

Having messed about with your sample and saved it to disc, the next thing is to get it into your own program. The Maestro software contains some new commands for STOS which must be installed on your STOS boot disk. A special program makes this a piece of cake.

Once this is done and you have booted your supercharged STOS ready for use, an accessory is available on the disc which allows you to load samples from disc and place them in a memory bank. It's much the same as the sprite editor allowing quit & grab to place the sample bank directly into your program.

With your samples (up to 32 at a time) safely installed in a memory bank you now have around 20 new commands to use. These range from stopping and starting a sample to looping, sweeping and playing samples as musical notes.

DOCUMENTATION

As with other STOS manuals, the Maestro offering is very readable with plenty of worked examples, but a bit obscure when you are trying to find a single thing out and can't remember where you found it lurking the first time. However its 30-odd pages contain information for both the greenhorn and the boffin, with several pages describing what a sample is and the theory behind it while at the other extreme there are several pages of information to help machine coders etc.

SPECIFICATION

For the boffins amongst us I shall quote the following statistics for the sampler cartridge without having a clue what they mean:

Signal to noise ratio	*30.48Dbs*
Maximum frequency	*98.5KHz (not guaranteed)*
Frequency selection	*Software controlled*
Resolution	*8 bits*
Linearity	*+/-0.5LSB*
Maximum input signal	*2.5 V RMS(peak to peak)*
Anti-aliasing filter	*Yes*

Internal successive approximation logic circuitry

Chip conversion activated by ROM-SEL 3 on cartridge port

WHAT DO I THINK???

Having never before used a sampler (software or hardware) on any computer I approached Maestro in complete ignorance. My initial software came without any instructions as the manual was still at the printers, but I found only one or two options which meant nothing to me. As it turned out when I did get the manual, these were the features for use with the cartridge which I didn't have at the time!

I can see myself using Maestro a good deal both for playing about with as a musical instrument and also for the more serious use as a programming aid. Incidentally, the Replay 3 samples on a recent ST/AMIGA FORMAT cover disc loaded straight into the Maestro editor with no modification needed so presumably the Maestro software will be compatible with other standard sampling hardware (but check first!).

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 Protext files together with a printout.

Hope to hear from you soon!

Pat

MACHINE CODE

Machine code from Basic

While getting this newsletter together I managed to corrupt the only (slapped wrists) copy I had on disc of Aaron's rather lengthy hints and tips.

Since it had taken me about three hours to type originally and I didn't fancy having to repeat the exercise I promptly made a working copy (too late! too late!) of the by now rather battered disc which had managed to survive being thrown around the kitchen and stamped on.

The theory was that I would attempt to use a TRAP #4 call, FLOPR (Page 254 of the manual) to copy the affected sectors of the disc to a memory bank, dump them to a fresh disc, then salvage the ascii files via a word processor. (Can you tell I like games that demand lateral thinking?). I'd already checked the disc with a sector editor and knew the text was still there. It was the directory section which had corrupted, not the data.

Unfortunately my efforts drew a blank. I even rang Aaron for help (since I'm a total novice at 68000 m-code) but despite following instructions TRAP #4, FLOPR remained unusable. Then I discovered from Richard Vanner at Mandarin that the TRAP #4 functions (although comprehensively documented in the manual) simply do **not exist in STOS!!!** All the other trap numbers should be okay though.

However, all was not lost. Despite having to type the text all over again I did profit from the exercise in that a manual error was discovered, and in trying to discover whether it was me or STOS that refused to interface correctly I did find out how to use machine code from STOS (well trap calls anyway).

I had much better success with TRAP #3 which covers things like printing to the screen, windows, graphics etc.

To use these calls you don't need to know **anything** about assembler. Instead, STOS has two sets of reserved variables in arrays. These correspond to the A and D registers but are handled in STOS in exactly the same way as normal array elements.

When you use a trap you are actually doing a sort of GOSUB, but instead of going to a subroutine of your own you are actually accessing a library of routines in STOS or the machine itself. After making a trap call, control passes back to the next command in your program, just the same as a gosub.

As with your own subroutines, trap calls are generalised code designed to do a specific job in various ways. To tell the routine exactly what to do you must give it some data to work on (usually) which is where the A and D registers come in. The arrays areg(x) and dreg(x) are copies of the registers which STOS updates in each direction. So if register D0 happened to contain the number 5 and you made a dreg(0)=6 command, the contents of the register D0 would become 6. Similarly, if a trap call changes the contents of a register, the new value will be passed back to the dreg() or areg() array.

Somewhere round here should be six listings. Five of them each deal with one trap 3 call apiece, while the longer one combines two of the smaller ones to make a pretty display. The routines have plenty of rems to help you, and they should be pretty self-explanatory when used in conjunction with the manual (Appendix E).

```
10 rem TRAP 3 CHROUT
15 rem print char in current window
16 for X=65 to 100
20 dreg(0)=X : rem ASCII of letter to
printed is loaded into register D0
25 dreg(7)=0 : rem function number
30 trap 3
40 next X
```

```
10 rem TRAP 3 PRINT STRING
20 rem Prints a string of characters in
window
30 A$="This is a test"
40 areg(0)=varptr(A$) : rem address of
string in memory
50 dreg(7)=1 : rem function number
60 trap 3
```

```
10 rem TRAP 3 LOCATE
20 rem move text cursor
30 dreg(0)=10 : rem x co-ordinate
```

MACHINE CODE

```
40 dreg(1)=15 : rem y co-ordinate
50 dreg(7)=2 : rem function number
60 trap 3

10 rem TRAP 3 SET PAPER
20 rem set paper colour
30 dreg(0)=rnd(777) : rem random colour
generation
40 dreg(7)=3 : rem function number
50 trap 3 : rem call the function

10 rem TRAP 3 SET PEN
20 rem set text colour
30 dreg(0)=rnd(777) : rem random colour
generation
40 dreg(7)=4 : rem function number
50 trap 3 : rem call the function

10 rem multi-coloured messages
15 key off : flash off : cls : for Y=0 to 10
20 rem using trap calls
30 rem TRAP 3 SET PEN
40 rem set text colour
50 dreg(0)=rnd(777) : rem random colour
generation
60 dreg(7)=4 : rem function number
70 trap 3 : rem call the function
100 rem TRAP 3 PRINT STRING
110 rem Prints a string of characters in
window
120 A$="This is a test"
130 areg(0)=varptr(A$) : rem address of
string in memory
140 dreg(7)=1 : rem function number
150 trap 3
160 next Y
```

STOP PRESS!

Update your copy of STOS to V2.4 – the latest version which comes with the STOS Compiler.

For just £2 you will receive a complete, replacement Language disc containing Basic 2.4, the single-precision floating point routine (SIN and COS run 30 times faster) and CONVERT.BAS to modify any programs with floating point to work with the new routine.

Send cheque or PO to Pat Winstanley.

DISC SECTOR SALVAGE

As a result of further work on the 'lost data disaster' I have come up with the following short listing for anyone else who might need to do a similar rescue job. With this I had no difficulty rescuing a couple of pages of Ascii text. Other types of data may prove more challenging, not particularly the retrieval, but piecing together afterwards!

A quick note! Page 249 of the STOS manual is a little unclear about the syntax required for passing words and longwords. The correct syntax is as follows:

trap x,n, .l expression, .w expression, .w expression ....etc.

See line 130 below for an example. Also the parameters should be specified in reverse order compared to the normal listing in reference books such as Atari ST Internals.

```
10 rem getsect4 by a relieved Pat
Winstanley!
20 rem To rescue intact data from a
disc which has lost its directory
30 rem use a disc editor to find the
location of your data
40 rem then define the variable accord-
ingly in lines 60 - 120
50 reserve as work 10,512*9 : rem
9=number of sectors to be read, each
 of which requires 512 (bytes?)
60 BUFF=start(10) : rem address of
buffer
70 SCRT=0 : rem filler - always 0
80 DEVN=0 : rem drive number (0 or 1)
90 STSC=1 : rem start sector
100 TRKN=12 : rem track number
110 SIDN=0 : rem side number (0 or 1)
120 SECN=9 : rem number of sectors to
read
130 trap 14,8, .l BUFF, .l SCRT, .w
DEVN, .w STSC, .w TRKN, .w SIDN,
 .w SECN
140 drive=1 : rem Swap drives before
writing bank to disc
150 save "sector11.mbk",10 : rem Save
bank containg data to disc
```

Sprite merging program

This STOS Basic program – written by the French genius himself, François Lionet – allows you to merge sprites together, to make larger sprites. This means that the maximum size of 64x64 pixels can be exceeded.

Type in the following listing and save it out as SPMERGE.BAS

```
1 rem ********************************
2 rem Big Sprite maker
3 rem By François Lionet 1989
4 rem
5 rem Public Domain !
6 rem ********************************
10 NBLMAX=10 : dim BLOC$(NBLMAX), TX(NBLMAX),TY(NBLMAX)
50 default : erase 1 : erase 2 : NI$=file select$("*.MBK","Load a Sprite bank")
55 if NI$="" then end
60 load NI$,1
65 key off : curs off : mode 0
70 A=hunt(start(1) to start(1)+length(1),"PALT")+4
75 for X=0 to 15 : colour X,deek(A+X*2) : next X
80 NS=1
100 hide on : back=default back : cls logic : cls back
101 sprite 1,x mouse and $FFF0,y mouse,NS
105 K=mouse key : if K=1 then put sprite 1 : while mouse key : wend : goto 101
115 A$=inkey$
120 if A$="+" then inc NS
125 if A$="-" then if NS then dec NS
130 if A$=" " then 100
135 if A$=chr$(13) then 200
140 if A$=chr$(27) then end
145 if A$="G" or A$="g" then 400
150 goto 101
200 sprite off : update : screen copy logic to back : auto back off : X1=0 : Y1= 0 : X2=319 : Y2=199
205 XM=x mouse and $FFF0 : YM=y mouse : K=mouse key
210 if K=1 and XM<X2 then X1=XM
215 if K=1 and YM<Y2 then Y1=YM
220 if K=2 and XM>X1 then X2=XM
225 if K=2 and YM>Y1 then Y2=YM
230 screen copy back to logic : ink ENC : inc ENC : ENC=ENC mod 16 : box X1,Y1 to X2,Y2 : show on : wait vbl : hide on : locate 0,24 : centre " Block #"+str$(NBLOC)+" "
235 A$=inkey$ : if A$="" then 205
240 if A$=" " then 100
245 if A$=chr$(27) then end
250 if A$=chr$(13) then 300
255 if A$="+" and NBLOC<NBLMAX then inc NBLOC
260 if A$="-" and NBLOC<>0 then dec NBLOC
265 if A$="c" or A$="C" then centre "Erasing block"+str$(NBLOC) : BLOC$(NBLOC)=" " : wait 50 : centre space$(30)
290 goto 205
300 centre "Get bloc"+str$(NBLOC)
305  BLOC$(NBLOC)=screen$(back,X1,Y1 to X2,Y2) : TX(NBLOC)=(X2-X1)/16 : TY(NBLOC) =Y2-Y1
310 wait 50 : centre space$(30) : inc NBLOC : if NBLOC>NBLMAX then NBLOC=NBLMAX
315 goto 100
400 NBB=0 : for N=0 to NBLMAX : if BLOC$(N)<>"" then inc NBB
405 next N : if NBB=0 then bell : goto 100
410 default : centre ">>> Bank Creation <<<" : locate 0,5 : erase 2 : reserve as data 2,100000
415 loke start(2),$19861987 : loke start(2)+4,$12 : loke start(2)+8,$12 : loke s tart(2)+12,$12 : doke start(2)+16,NBB : doke start(2)+18,0 : doke start(2)+20,0
420 AA=start(2)+22 : AP=AA+NBB*8
425 for N=0 to NBB-1 : loke AA+N*8,AP-AA : poke AA+N*8+4,TX(N) : poke AA+N*8+5,T Y(N)
430 print "Sprite";N+1;" X size = ";TX(N)*16;" / Y size = ";TY(N);" —> Input t he hot point co-ords:"; : input PX,PY
435 if PX<0 or PY<0 then 430
440 poke AA+N*8+6,PX : poke AA+N*8+7,PY
450 TM=TX(N)*TY(N)*2 : TLM=TX(N)*2 : TLS=TX(N)*8 : AB=varptr(BLOC$(N))+8
455 for Y=0 to TY(N)-1 : for X=0 to TX(N)-1
460 A=AB+Y*TLS+X*8 : M=$FFFF xor
```

```
(deek(A) or deek(A+2) or deek(A+4) or
deek(A+6) )
465 doke AP+Y*TLM+X*2,M : next X :
next Y
470 AP=AP+TM : copy AB,AB+TM*4 to
AP : AP=AP+TM*4
475 next N
480 erase 1 : reserve as data 1,AP-
start(2) : copy start(2),AP to start(1) : era
se 2
485 NS$=file select$("*.mbk","Save the
new Sprite bank")
490 if NS$="" then 50
495 save NS$,1
500 erase 1 : goto 50
```

The sprite merger works in the following manner:

1 Firstly load a sprite bank which contains sprites that will merge together to make the new 'block' sprites

2 Using the mouse and left mouse button, plot the sprites onto the screen, making up the new image.

- The '+' and '-' keys will allow you to select the different sprites within the loaded bank.
- Pressing the spacebar will erase the screen.
- The Escape key will exit from the program

3 When you'r happy with the new sprite image, press RETURN

- A flashing box indicates the size of the new sprite. Using the left and right mouse buttons, define the size box around the sprite.
- Pressing '+' and '-' will select the sprites block postion within the bank
- 'C' erases the current block
- Escape ends

4 Press RETURN to create another bloc
or:

- Press 'G' to enter each of new sprite hot spots and then to save out the new sprite bank

New banks created in this way will work in the sprite editor but you'll find that the big sprites will corrupt the screen.

The maximum size of a large sprite depends on the size of the sprite buffer within STOS. François estimates that the limit will be about 128x128 pixels.

Alert box

Les Barclay has come up with the following useful multi-mode routine which enables you to incorporate 'either or' alert boxes – a feature which is missing from STOS. You can use it as a subroutine in your own programs.

(The PD19 disc contains a more comprehensive listing and tutorial from another member but this is for high resolution only.)

```
10 rem—>Simple two choice alert
"alert.bas". Any mode.
20 rem—>FROM L.Barclay

30 rem*******************************************
40 rem—>set strings and GOSUB.
50 T1$="RUN AGAIN?" : T2$=" YES " :
T3$=" QUIT " : gosub 120
60 rem————————————————
70 if T=1 then A$=" YES " else if T=2
then A$=" QUIT "
80 print "The key selected was ";A$
90 print "ANY KEY " : wait key : cls : end
100 rem
110 rem———————————————
120 rem>>> ALERT BOX <<<
130 rem———————————————
140 windopen 13,20/divx,8,40/divx,9,8 :
clw : hide : curs off
150 locate 6/divx,1 : square 24/divx,3,1 :
centre T1$
160 limit mouse 180/divx,200/divy to 440/
divx,260/divy
170 set zone 1,188/divx,220/divy to 288/
divx,238/divy
180 set zone 2,334/divx,220/divy to 434/
divx,238/divy
190 T=0 : show : repeat : T=zone(0)
200 if T>2 then T=0
210 if T=0 then inverse off : locate 6/divx-
divx,5 : print T2$
220 if T=0 then inverse off : locate 24/
divx-divx,5 print T3$
230 if T=1 then inverse on : locate 6/divx-
divx,5 : print T2$
240 if T=2 then inverse on : locate 24/
divx-divx,5 : print T3$
250 until mouse key and T>0
260 limit mouse : windel 13 : curs on :
return
```

20 things you didn't know about STOS and its creator

1 STOS was written by François Lionet, a 25-year-old Frenchman living in Creil, 60 kilometres from Paris.

2 François is a qualified vetinary surgeon.

3 He gets married on June 24 (the same day as the Atari Show), but is only taking a week-long honeymoon as he has to finish AMOS. His fiancée, Carine, has insisted he leave all his computers at home!

4 François' first computer was a Superboard II from Ohio Scientific, purchased in 1981. He was one of only a few French owners so he had to write his own 6502 assembler in Basic, and then his own games. He even wrote a full Defender game in 8k of memory!

5 François wrote two games for Firebird Software: ***Chicken Chase*** (Oric, CPC and C64 – over 60,000 copies sold) and ***Olé*** (CPC). He also wrote three commercial Oric games for a French software house and a musical puzzle called ***Serenade*** for the Commodore 64.

6 He wrote the ***Captain Blood*** conversions for the PC and C64 – most of the work was done in one month (fast, isn't he?).

7 François programs on almost any microcomputer. But this has its problems: The worst case is programming on an Intel 8088 (IBM PC) and on a Motorola 68000 (Atari ST) – Intel syntax works from right to left, and Motorola syntax from left to right!

8 François has had his fair share of problems with software houses. He converted ***Around the World in 80 Days*** to the Atari ST using STOS, only to discover on the day that he completed it that the French publisher, FIL, had gone bust! Luckily the conversion only took a couple of weeks.

9 STOS took 18 months to write using Kuma's K-Seka.

10 The compiler was written in three months using DevPac II. Extension handling for the compiler was the most difficult part to program.

11 He always programs listening to loud music through his headphones. He had to build a flashing light unit to tell him that the phone was ringing!

12 STOS was first released in France in the spring of 1988 with the multi-mode sprite editor, macro assembler, text-only music editor and example programs. It was not an immediate success.

13 Mandarin bought the rights for STOS, designed the low-res sprite editor, commissioned three games, and asked for other refinements.

14 François wrote the Orbit game in two days flat, and Zoltar within a week.

15 The worst bug was the way STOS handled new extensions.

16 The hardest part was writing the sprite trap routine – which turned out to be a veritable nightmare.

17 François has to do his military sevice next year – but he'll be taking all his computers with him so he can keep programming!

18 François' idols are Jes ***(Starglider)*** San, Bill ***(Sinbad)*** Williams, Jeff ***(Tripatron)*** Minter and all the ***Captain Blood*** team. He really would like to be like them when he greaus (sic!) up.

19 ***Skateball*** from Ubisoft was originally written in STOS before being converted for Amstrad CPC, Spectrum, Commodore 64, PC and Nintendo. With good Nintendo games selling 200,000 copies, Allain Fillion the designer should be getting some impressive royalties soon!

20 Bullet Train is on sale in France at around the £20 mark under the title ***Expressing*** with faster scrolling and improved graphics. ■

This STOS Newsletter was edited by Pat Winstanley. Layout by Chris Payne.

Text parser

Dicon Peeke has developed this parser which can be used as the basis of text input by the player in a variety of different applications. Simply merge it into your own program, renumbering if required. Then add your own vocabulary. It will cope with multiple commands, not just GET ROPE!

```
10 dim NOUN(10) : dim VERB(10) : dim
WD$(10) : dim WD(10) : V$="" : N$=""
20 AN=0 : dim NOUN$(10) : dim
VERB$(10) : AN$=" AND" : THE$="THE"
: THE=0 : T$="
30 for T=1 to 10 : read T$ : T$=T$+" " :
NOUN$=NOUN$+left$(T$,4) : next T
40 for T=1 to 10 : read T$ : T$=T$+" " :
VERB$=VERB$+left$(T$,4) : next T 50
cls
60 rem ****************************************
70 data "XXX ","DROP","EAT
","THROW","GET","UNLOCK","TWIST","OIL
","STEAL","SHO"
80 data "XXXXXX","BOX","HAT",
"SOCK","BOOK","CANDLE","SPLOTCH",
"TORQUE", "KEY","
90 rem ********************** INPUT
******************************************
100 rem
110 rem
120 rem THE I$ HOLDS THE COMMAND
THAT IS TO BE PARSED
130 I$="UNLOCK THE BOX THEN EAT
THE CANDLE AND TWIST KEY"
140 rem:Input ">>>>";I$
150 AN=0
160 THE=0 : THE=instr(I$,THE$,1) : If
THE=0 then goto 180 : rem IS THERE A
"THE?
170 L=len(I$) : I$=left$(I$,THE-
1)+mid$(I$,THE+4,(L-THE)) : rem STRIP
OFF "THE"
180 THN=0 : THN=instr(I$,THN$,1) : If
THN=0 then goto 200 : rem IS THERE A
"THE"
190 L=len(I$) : I$=left$(I$,THN-
1)+"AND"+mid$(I$,THN+4,(L-THN)) : rem
STRIP OFF"
200 AN=instr(I$,AN$,1) : If AN=0 then
goto 240
210 I1$="" : L=len(I$) : I1$=left$(I$,AN-1)
: rem:print "[";I1$;"]"
220 I2$="" : I2$=mid$(I$,AN+5,(L-AN)) :
I$=I1$ : rem:print "[";I2$;"]"
230 rem ************************ SEPARATE
OUT SEPARATE WORDS *******************
240 rem
250 for T=1 to 10 : WD$(T)="" : next T :
WD=1
260 for T=1 to len(I$)
270 T$=mid$(I$,T,1) : if T$<>" " then
WD$(WD)=WD$(WD)+T$
280 if T$=" " then WD$(WD)=WD$(WD) :
inc WD
290 next T
300 for T=1 to WD : WD$(T)=WD$(T)+" " :
WD$(T)=left$(WD$(T),4) : next T
310 rem:for T=1 to WD : print
"[";WD$(T);"]" : next T
320 rem **************************** FIND
VERB ******************************
330 VFLAG=0 : NFLAG=0
340 for T=1 to 10 :
VRB=instr(VERB$,WD$(T),1) : If VRB<>0
then VFLAG=VRB
350 next T : If VFLAG=0 then print "NO
VERB FOUND" : stop
360 for T=1 to 10 :
NN=instr(NOUN$,WD$(T),1) : If NN<>0
then NFLAG=NN
370 next T : If NFLAG=0 then print "NO
NOUN FOUND" : stop
380 print "VERB IS NO. ";(VFLAG-1)/4
390 print "NOUN IS NO. ";(NFLAG-1)/4
400 if AN<>0 then I$=I2$ : print : print
"________________________" : goto 150
```

■

Can you help? 𝄞 ♪ ♫

Mandarin Software will be demonstrating the full range of STOS products at the Atari Computer Show, particularly STOS Maestro.

If you produce any demonstration programs, especially STOSified versions of current PD demos, please send them to Chris Payne or Richard Vanner at Mandarin. All discs will be returned – you may even see your work in all its glory on the big screen, blaring out of the sound system!

Public Domain Library

We are making an extra-special effort to build up a library of STOS programs for everyone to use – so we need your help. Whether you've written a game, a demo or a clever routine, there's over 20,000 STOS owners out there who would love to see what you've created.

Listed alongside you'll find a good selection of interesting programs with something for everyone. Each disc costs just £2 each – or £1 if you supply your own disc. They are supplied in single-sided format on double-sided discs unless stated otherwise. Three of the discs are shareware – the authors will receive payment from every disc that's ordered (deduct £1 if you supply your own disc).

In most cases the programs require you to boot your copy of STOS first – that way you can have a good nosey at the listing!

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

Send cheques, postal orders or stamps to:

Pat Winstanley,

Choose from the following:

SPD15: *Caves of Rigel* – a well-designed arcade-type game converted from the Atari 8-bit commercial release on Atlantis Software by the original author, Ralph Effemey. There's also another game from Ralph: *A Froggy Day in London* based on the classic arcade game Frogger.

SPD16: *STOS Demo* – The first demo created for use in shops – no great shakes, but includes a nice tune you can grab!

SPD17: *STOS Demo2* – The cycling demo you may have seen at the shows, using Ian Waugh's attention-grabbing tunes from STOS Disc 3.

SPD18: *STOS Add-ons Demo* – The latest demo from Mandarin with sampled sound – you'll need the Maestro extension to run this.

PD10: *Xmas Demo* – This double-sided disc contains well-executed and amusing spoof of the nativity.

PD14: *First Serve* – Wimbledon revisited in this one-player against the computer tennis game.

PD19: *STOS listings* – Programs and routines from the Newsletter.

SH5: *Brain Games* – A selection of thinking STOS games (Gridder, Minefield, Rotation, Solitary and Swopper) from Pete Gerrard and Sandra Sharkey. (£4.25)

SH7: *STOS Games* – A selection of simple STOS games by Mike Brown. (£3.00)

SH9: *Kids Games* – Five games for pre-school and primary children by David Alexander. (£3.00)

COMPETITION

£185 of STOS products are up for grabs!

FIRST PRIZE

...is STOS Maestro Plus, STOS Compiler and STOS Sprites 600 (worth over £100!)

SECOND PRIZE

...is STOS Compiler and STOS Sprites 600 (worth £34.90)

THREE RUNNERS-UP PRIZES

...of STOS Sprites (worth £14.95 each)

Here's a fantastic opportunity to win all the latest STOS add-ons for your Atari ST. All you have to do is write a typing tutor in STOS Basic. It's not a difficult task, but we're looking for good presentation, proper error trapping and sufficient addictiveness to keep ST owners coming back to improve their typing skills.

You could start with simple drill and practise then move up to full sentences. Your program could pull in word processor files from disc and ask the typist to copy the document exactly. You could even include simple bar charts to show percentage accuracy, speed and so on – maybe offering a reward for excellent performance. Take a look at programs like Touch Typing and Type Invaders for the Atari 8-bit, or any other similar program you may have on your previous computer.

We're not expecting wonders from you – and there's a good chance of winning a prize as we're not expecting stacks of entries. So give it a try!

The entries will be judged by the staff at Mandarin Software. The winners will be notified by post and their names will be announced in the STOS Newsletter. The winning programs may be put in the public domain for other STOS owners to try out. If the winning entry is of commercial quality Mandarin Software will negotiate further payments with the author.

The closing date is Monday July 30, 1989, so get those fingers tapping! (All discs will be returned.)

Suggestions for future competitions will be welcomed.

Send your entries to:

Touch Typing Competition, Mandarin Software,