

STOS

Newsletter

Issue 10 STOS User Club

Editorial

Issue 10 is here at last, and we have plenty of goodies for you in this issue! The usual series on Absolute Beginners, Learners and Extending STOS, as well as articles by Richard Gale and reviews of both STOS Paint Master and TOME V2.1 (Yes it really is finished!).

Shock Horror TOME V2.1 released!

At last the new version of the Total Map Editor has been released, along with TWO new demo games and a scrolling text demo as part of the package. See Peter Hickman's review on page 21 of this issue.

If you have V2.0 of TOME then you can upgrade to V2.1 for only £1. Simply send your two TOME master disks back to Shadow Software with a cheque for £1 (£3 if overseas), and they will be upgraded and sent to you with the new manual.

New Atari User competition

Judging is now taking place for the New Atari User STOS competition for the best 8-bit Platform game in STOS. The number of entries has been quite good, and the quality of the programming is excellent. The results of the competition will be published in the next New Atari User magazine out in October.

New Budget Software Label

A new software label called **Digital**

Dimension has been set up to market budget ST software, and their first game to be released was written in STOS! Called Jiggers, it will retail at £4.99, and can be bought direct from them for an extra £1 postage and packing. They are also on the lookout for new games to release, the emphasis being on playability and value for money. The address:

**Digital Dimension,
Winterbourne,
Bristol**

Remember to enclose a stamped addressed envelope if you want your disk returned along with an evaluation of the program.

We'll be doing a review of Jiggers in the near future.

Inside this issue:

Ping 10-Liner.....	2
Removals.....	3
Tank Buster 10-Liner.....	4
Extending STOS.....	5
STOS Paint Master Review.....	8
Tiny Pic Decompactor.....	9
Absolute Beginners.....	10
Calculator Routine.....	12
Throwing Things.....	14
Bug Hunting with Professor Speck.....	16
Readers Survey, Win a free subscription!	18
TOME V2.1 Review.....	21
Public Domain.....	23

10-LINER

PING

10-Liner Game

Here's the 10-liner version of one of the world's first ever computer games! If you can remember playing this one you'd better put your walking frame away and get typing!

Ping requires 2 sprites to be defined: These are the bat (used for both players) and a ball. Colour 1 must be used for the bat (as the ball rebounds upon detecting this colour). The larger you design the bat, the easier the game will be for both you and the computer, which plays your opponent (Yes this game has an intelligent opponent in 10 lines!).

```
10 dim D(1) : reserve as screen
5 : mode 0 : key off : curs off
: hide on : locate 0,12 :
inverse on : centre " P I N G "
: locate 0,14 : inverse off :
centre "By Aaron Fothergill" :
locate 0,16 : centre "In 10 line
s of STOS!" : locate 0,23 : c
entre "Press a Key" : wait key
: print 20 input "level of diffi
culty (1-3) ?":LVL : IF LVL<1 or
LVL>3 then 20
30 mode 0 : key off : curs off :
hide on : X=hunt(start(1) to
start(1)+length(1),"PALT")+4 : f
or A=0 to 15 : colour A,desk
(X+A*2) : next A : logic=5 :
auto back off : Y1=100 : Y2=100:
screen copy back to 5 :lnk 4 :
bar 0,0 to 319,9 : bar 0,190 to
319,199 : lnk 5 : bar 158,10 to 1
61,189 : BY=160
40 update off : D(0)=2^LVL : D(
1)=-1 : lnk 2 : bar 0,10 to
4,189 : bar 315,10 to 319,189 : S
C1=0 : SC2=-1 : BX=1-D(0) :
while SC1<5 and SC2<5
50 screen copy 5 to back : logi
c=back : sprite 1,8,Y1,1 : sprite
```

```
2,311,Y2,1 : sprite 3,BX,BY,2 :
update : screen swap : A=0 : r
epeat : P=point(max(0,min(319,BX
+D(0)*(1-A))),max(0,min(199,
BY+D(1)*A))) : If P=1 then D(0)=
(-1-(BX<160)*2)*(2^LVL)
: D(1)=rnd(3+LVL)-rnd(3+LVL)-
DY : A=1 : shoot : goto 80
60 If P=2 then boom : wait 10 :
SC1=SC1-(BX>160) : SC2=SC2-
(BX<160) : BX=7 : BY=100+rnd(60
)-rnd(60) : D(0)=2^LVL : D(1)
=rnd(2)-rnd(2) : logic=5 :
paper 0 : pen 15 : locate 3,3 :
S$=str$(SC1)+" " +str$(SC2)+"
" : print S$ : logic=back
70 If P=4 then ball : D(A)--D(A)
80 inc A : until A=2 : DY=0 : If
Jup and Y1>16 then Y1=Y1-
4 : DY=2*LVL
90 Y2=Y2+sgn(BY-
Y2)*min(LVL*2,abs(BY-Y2)) :
BX=BX+D(0) : BY=BY+D(1) :
If Jdown and Y1<184 then Y1=Y1+
4 : DY=2*LVL
100 wend : logic=physic : screen
copy back to physic : locate
0,23 : centre "Press a Key" :
wait key : goto 30
```

HELP !

If you are stuck and need help with your programming. Just phone the

STOS Helpline on 0271-

It is best if you phone between 1pm and 7pm U.K. Time, as I'm most likely to be in then.

Overseas members please note I Check what time it is in the U.K. before you phone, it isn't funny being woken up at 4a.m!

REMOVALS

REMOVALS

By Richard Gale

As someone who regularly writes large programs in STOS, (like the BIG STOS DEMO), I like to have a few REM's just to see what each chunk of code will do. When the program was finally finished I had to manually remove the REM statements, instead I wrote this short program which did it for me.

Note: To use the REM Remover your program source code must be saved as ASCII text (SAVE"filename.ASC" will do it.).

Run the REM Remover and you will be asked for the 'ASCII program to DE-REM' select your program, to abort the REM Remover

select QUIT. When you have selected your file a check is made to make sure STOS can read it, if STOS cannot find your file an error is reported and the program terminated.

You will then be prompted for the 'ASCII program to create', select a file other than the source file. If you don't have enough free disk space an error will be reported and again the program terminated. The program has specially been coded so that the source and destination programs can be in different directories on different drives.

If the above is correct the program will start processing your program, and report which line is currently being DE-REMOVED. Once the program has finished a report will show the size of the new program as a percentage of the original.

To load the processed file use LOAD"filename.ASC"

Warning: Don't run the REM Remover on itself, because it will remove vital code.

THE PROGRAM

```
100 rem REM Remover
110 ciw : key off
120 locate 0,0
   : under on : centre "REM Remo
   ver v1.0 -
   by Richard Gale" : under off
130 gosub 430 : if ERR=true then
   end
140 gosub 500 : if ERR=true then
   end
150 curs off
160 locate 0,3 : centre "Source
>" + SOURCE$
170 locate 0,4 : centre "Destina
tion >" + DEST$
180 gosub 240
190 locate 0,8 : centre "DE-
REM conversion complete"
200 locate 0,9 : centre "New fil
e is" + str$(PER) + "% of original.
"
210 curs on
220 end
230 rem DE-REM Routine
240 open in #1, SOURCE$ : open
out #2, DEST$
250 repeat 260 if eof(#1) then g
oto 380
270 line input #1, TEXT$
280 TEMP = instr(TEXT$, " ")
290 TEMP$ = left$(TEXT$, TEMP -
1) : locate 0,6 : centre "Line >
" + TEMP$
300 if upper$(mid$(TEXT$, TEMP + 1
, 3)) = "REM" then TEXT$ = ""
310 repeat 320 POS = instr(TEXT$,
" rem ") : if POS = 0 then POS =
instr(TEXT$, " REM ")
330 if POS then TEXT$ = left$(TE
XT$, POS - 1)
340 if POS > 8 then if mid$(TEXT$,
, POS - 1, 1) = ":" then TEXT$
= left$(TEXT$, POS - 8)
350 until POS = 0
360 if TEXT$ = "" then TEXT$ = ""
then TEXT$ = ""
```

10-LINER

```
370 if TEXT$="" && <> ""
  then print #2,TEXT$
380 until eof(#1)
390 SIZE#:=SIZE : PER=lof(#2)/
  SIZE#*100.0
400 close #1 : close #2
410 return
420 rem Select Source program
430 SOURCE$=file select$("*.AS
C"). Select ASCII Program to
DE-REM")
440 if SOURCE$="" then ERR=tr
ue : goto 480
450 SOURCE$=drive$+"-"+dir$+"\
"+SOURCE$
460 if dir first$(SOURCE$,-
1)="" then ERR=true : locate 0,
2 : centre "File does not exist"
: goto 480
470 open in #1,SOURCE$ : SIZE=
lof(#1) : close #1
480 return
490 rem Select Destination
program
500 DEST$=file select$("*.ASC",
" Select ASCII program to cre
ate")
510 if DEST$="" then ERR=true :
goto 540
520 DEST$=drive$+"-"+dir$+"\
"+DEST$
530 if dfree<SIZE then ERR=true
: locate 0,2 : centre "Disk to
o full."
540 return
```

Page 6 New Atari User

Page 6 magazine (also known as New Atari User) regularly run a STOS feature in their ST section. Pete Hickman, the journo hack who writes their column has reviewed STOS products, P.D and just about always has some sort of program listing in his column. Well worth

TANK BUSTER

The object of this game is to destroy the enemy tank that is attacking you, by firing shells at it. Your tank can be moved left and right by the joystick. The Angle of your gun is adjusted by Joystick up/down, and is shown as a mark on the left of the screen. On each level you must hit the tank twice per level number (twice on level one, four times level two and so on.). If the tank reaches you, it's game over.

You will need 5 sprites for this program, these are:

- 1- Your tank/gun, 32x16 pixels, pointing right, hotspot at bottom centre.
 - 2- Enemy tank, 32x16 pixels, pointing left, hotspot at bottom centre.
 - 3- Shell, 3x3 pixels, simple circle, hotspot in centre
 - 4- Explosion, 1st animation, 16x16 pixels, hot spot in centre 5- Explosion, 2nd animation, 16x16 pixels, hot spot in centre
- ```
10 mode 0 : key off : curs off :
 hide on : auto back off :
A=hunt(start(1) to start(1)+leng
th(1),"PALT")+4 : for B=0 to 1
5 : colour B,deek(A+B*2) : next
 B : LVL=1 : DEAD=0 : while
DEAD=0 : X=20 : CBX=-1 : BX=-
1 : EX=-10 : EL=45 : CX=300-
rnd(20) : DEAD=0 : KLL=0 : ink
 3 : sprite off : update : logic
=back : bar 0,190 to 319,199 :
logic=physic : screen copy
back to logic
20 LX=280 : CDX=-1 : locate
 0,0 : paper 0 : pen 15 :
 centre "Level"+str$(LVL) :
 while DEAD=0 and KLL<LVL*2
 : sprite 1,X,190,1 : sprite 2,CX,
 190,2 : sprite 3,EX,190,4+EF :
```
- (Continued on page 7)

# MACHINE CODE

## EXTENDING STOS

• #3 of 3

In this issue, the last of the current series, we will be looking at how to create a compiler extension library, so that you can compile programs written using a custom extension.

The STOS Compiler extensions are the most awkward of the lot, so expect pain and suffering when writing one. Strange bugs occur and wierd things happen, without any expllcable cause. Programmers HATE doing the compiler extension!

Now that's out of my system, we'll get on with it. The basic routines of your Interpreter extension can be used with the compiler version, but the header and parameter routines are completely different. Make a copy of your Interpreter extension and list it to the printer, then rip out everything from the copy except for the actual routines themselves.

You then need the compiler extension header. This has offsets to the parameter lists for the routines, the initialisation routine and the first library routine.

```
START: dc.l PARA-START
 dc.l INIT-START
 dc.l LIB1-START
```

Then you need to put in a library catalog, which contains (in word data) the lengths of your library routines.

```
CATALOG dc.w LIB2-LIB1
* length of routine 1
 dc.w NDPRG-LIB2 ; length
* of routine 2
```

Next up comes the parameter lists for each routine. First you need to tell the compiler the number of Library routines and the number of new commands:

```
PARA: dc.w 2,2
```

Then you must list offsets to the parameter lists:

```
dc.w PPRNTER-PARA
```

## dc.w PRANGE-PARA

All you need now are the lists themselves. These are made up of byte values in the order that parameters are expected. Values of **\$0** are used where an Integer is expected; **\$40** when a Floating Point number is expected and **\$80** when a string is expected. The first byte represents the value expected to be returned if the instruction is a function (note that commands must also have a zero byte inserted here, even though they will not return anything). It is best to define the four values required, to make life easier:

```
e.g
I equ $0
F equ $40
S equ $80
C equ ""
```

If the instruction can only accept one type of parameter, then you stick a **1,1,0** on the end of the list. Otherwise, if you want different permutations of parameters for the command (Most of the maths functions can handle both Integer and floating point values) then simply end each parameter list with a **1** and add **1,0** to the last one. e.g for the definition of **ABS(n)** where n can be either Integer or floating point

```
PABS: dc.b F,F,1 ; FP version
 dc.b I,I,1,0 ; Integer version
```

So for the parameter list of our Useful extension, we would have

```
PPRNTER: dc.b 0,I,I,1,0
* prntr command uses 1 Integer
* parameter
PRANGE: dc.b I,I,C,I,C,I,1,1,0
* range uses 3 integers and
* returns and Integer value
```

Now the instructions have been defined, we have to initialise the routine. The simplest thing to do here is to jump to the coldstart routine. i.e:

```
INIT: bra COLDST
```

# MACHINE CODE

Then we do our coldstart routine...

```
COLDST: lea END(pc),a2
END: rts
```

In this routine, you must load the end address of the extension into the **A2** register. As we only need to do an **RTS** for this extension, simply define the next line as **END:** (thus saving about 4 bytes).

All that's left now is to add our library routines. Note that these routines must be completely independent of position and each other, as the compiler only uses the routines used by the compiled program. It is theoretically possible to call another library routine from within one library routine, however, as this is almost impossible due to the nature of the compiler (i.e. bugs) it is not worth the nervous breakdown you would get if you tried to use them.

One thing you must add to the start of your routines is a **dc.w 0** command as a sort of "I promise not to call any other library routines". Then follows your routine. All parameters are passed on a stack referenced by **A6** and **D0** contains the parameter list being used (if the command only uses one parameter list then ignore **D0**). It is wise not to use Address registers 4-7 (**A4-A7**) as **STOS** uses them, although you can usually safely use data registers 0-7. One thing found while working on the **TOME** extension was that **D0** can sometimes be affected by the joystick interrupt! Thus making it change sometimes when the joystick is moved, so try not to use **D0**.

To pull a parameter off the stack (they are put on in **REVERSE** order), simply do:

```
move.l (a6)+,D1 for an integer value
move.l (a6)+,A0 for the address of a string value
move.l (a6)+,D1 for the top half
```

```
move.l (a6)+,D2 and the bottom half of a floating point value
```

the reverse is required to return a parameter at the end of the routine, although you should always use the **ASK** routine before returning a string value. This has the format

```
ASK equ 70
jsr ASK
```

where **D3.L** is the number of bytes needed for the string. It will return the address of the returned string space in **D3.L**.

O.K now for our first routine the **PRINTER** library (also labelled **LIB1**)

```
PRINTER:
LIB1: dc.w 0 ; no library calls
 move.l (a6)+,d3 ; get an integer value
 movem.l a0-a6,-(a7) ; save all registers
 move.w d3,-(a7)
 move.w #33,-(a7)
 trap #14 ; call the printer settings routine
 addq.l #4,a7 ; tidy up
 movem.l (a7)+,a0-a6 ; restore registers
 rts ; that's all
```

and our **RANGE** function (also labelled **LIB2**)

```
RANGE:
LIB2: dc.w 0
 move.l (a6)+,d1 ; get the last parameter
 move.l (a6)+,d2 ; get the middle parameter
 move.l (a6)+,d3 ; get the first parameter
 cmp.l d2,d3
 bit TOOLOW
 TLBK cmp.l d3,d1
 bit TOOHIGH
 THBK move.l d3,-(a6) ; return an integer
```

# MACHINE CODE

```
rts ; end the routine
TOOLOW move.w d2,d3
bra TLBK
TOOHIGH move.w d1,d3
bra THBK
NDPRG dc.w 0 ; mark the end of
the extension
```

This program should be assembled as USEFULLE.CZ and stored in your compiler folder. You should now have two extra commands in STOS, which can also be compiled. If you have any ideas for extra commands that would be handy, then either write them and send them in, or write in with them, so we can make a new extension (hopefully some STE functions as well). Have fun!

## Upgrade now !

If you are producing PRG versions of your games or demos, it is essential that you upgrade to the latest version of STOS, otherwise they won't work on the newest Ataris. Version 2.5 works with TOS 1.6 and the Atari STE which is now on sale.

Simply send £2 or £1 plus a disk to Sandra Sharkey at the STOS Public Domain Library.

## P.D Upgraded

The STOS Public Domain Library will no longer accept any PRG versions of programs that have not been compiled with version 2.5 of STOS (i.e. any that won't work on the STE). Most of the current P.D library has now been re-compiled to run on the STE, and now I've actually got one, I'll be doing some extension programming for it soon!

Tank 10-Liner

(Cont. From page 4)

```
sprite 4,BX,BY,3 : ink 0 : draw
0,190-EL0 to 2,190-EL0 :
EL0=EL : ink 15 : draw 0,190-
EL to 2,190-EL
30 X=max(5,min(84,X+|left-
|right)) : If fire and BX=-1
then BX=X : BY-180 : BDX#=(90-
EL)/9 : BDY#=-EL/9 : shoot
40 EL=max(0,min(90,EL-
|up+|down)) : EF=1-EF : If BX>-
1 then BX=BX+BDX# :
BY=BY+BDY# : BDY#=min(4,BDY
#+0.2) : If BY>189 then
gosub 100
50 CX=CX+CDX : If CX<X then
boom : DEAD=1
60 If CX<(LX) or CX>300 then
CDX=-CDX : LX=LX-LVL*4
80 If collide(4,16,8) then boom :
BY=199
90 wend : inc LVL : wend :
paper 0 : pen 15 : locate 0,10 :
centre "Game Over" : wait key :
run
100 KLL=KLL-(BX>CX-
16 and BX<CX+16) : EX=BX : EF
=0 : BX=-1 : boom : return
```

*Have Fun !*

## The Games Maker's Manual

The Games Maker's Manual by Stephen Hill is no longer available through the STOS Club as we are now out of stock. We will be trying to obtain books for the few remaining orders, but no further orders can be taken. The Listings disk is now available through the STOS P.D Library.

# REVIEW

## STOS PAINT

**MASTER** £14.95

Stallion Software,  
Plymouth,  
Devon.

### Review by Aaron Fothergill

When I was told that someone was working on a STOS based art package that would allow you to use more than 16 colours sometime last year, I expected it to give you four, sixteen colour palettes on screen at once. Having seen the resultant program I'm impressed! With STOS Paint Master, you can have 49 (Yes Forty Nine) different palette bands on screen at the same time, which lets you use all of the ST's 512 available colours at once. Not to mention the fact that you can do this from within a STOS program, as included in the package is a machine code program to include in your STOS programs. This program sets up a little interrupt driven program which gives you 49 palettes on the screen at once.

The actual Paint Master program itself is supplied as a .BAS file and a .PRG program on one disk (which can be backed up).

The .BAS file can be renamed as .ACB so that it can be loaded as a STOS accessory.

In general layout, the program is very Neochrome based, in that you can see half of the picture, and a bar of icons. You can flip to full screen drawing, or simply flip between the two halves. This system is easy enough to use and the program runs fast enough to make drawing accurate and pleasant. There are plenty of features for a package at this price, including mirroring, cut/paste, text and other features.

The ultimate test was when I let my

graphic artist brother Adam loose on it. Adam tends not to pull punches when he doesn't like an art package (R.I.P Flair Paint and STOS paint) but he got on well with Paint Master, and found it easy to use, as well as being fast enough to work with.

A couple of points were brought up when I was testing the program, these being that you can only work on one picture at a time, and that cut and paste works only as one operation (you can't cut out a block, and then paste it in several positions), also the text function works only to STOS text co-ordinates. However, Stallion Software are working on a new version of the program, which will edit more than one picture at a time, have proper cut and paste and also has pixel accurate text drawing (thanks to Richard Gale).

One of Paint Master's good points is the number of picture file formats it can load. Including Degas Elite, Neochrome, STOS Packed and it's own format (for 512 colour pictures). There will also be a Spectrum 512 converter available on P.D for it soon, so that you can use all these pictures also.

The manual for STOS Paint Master is very short, although fortunately the program is very simple to use, so this shouldn't be much of a problem. It would have been nice to have had some details of using the 512 colour viewer routine however, but again, the viewer program is well REM'ed.

Overall, this is a useful and powerful art package, well suited for use with STOS. I have been using it in the development of TOME V2.1 and other software developers I know have been using it for 512 colour title screens. The package is well supported by the programmers and deserves to do well.

# TINYPIC DECOMPACT

## TINY PICTURE UN-COMPRESSOR

By Richard Gale

Simply type the following STOS Basic program. Load one of your favourite TINY pictures into bank 10 and call the routine ( an example can be seen at the beginning of the program ), and you TINY picture will be decompresses in a couple of seconds.

### TINY PICTURE FORMAT

TNY - any resolution

TN1 - low resolution

TN2 - medium resolution

TN3 - high resolution

1 byte resolution ( 0 low resolution

1 med resolution

2 high resolution

note: the same as MODE )

[ If resolution > 2 ( colour rotation )

1 byte left and right colour animation limits. The high four bits hold left ( start ) limit, and low four bits hold the right ( end ) limit.

1 byte direction and speed of colour animation ( negative value indicates left rotation, positive values indicate right. Absolute value is the colour delay in 1/60's of a second.

1 word colour rotation duration ( number of iterations ) ]

16 words colour palette ( store as 1 word per colour )

1 word number of control bytes

1 word number of data words

3-10667 bytes control bytes

1-16000 words data words

42-32044 bytes total

### Control byte meanings:-

x < 0 absolute value specifies the number of unique words to take from the data section ( from 1 to 127 )

x = 0 1 word is taken from the control bytes section which specifies the number of times to repeat the next data word ( from 128 to 32767 )

x = 1 1 word is taken from the control bytes section which specifies the number of unique words to be taken from the data section ( from 128 to 32767 )

x > 1 specifies the number of times to repeat the next word taken from the data section ( from 2 to 127 ).

### PROGRAM LISTING

```
100 key off : curs off : flash
 off : hide on 110 reserve as
 data 10,32256 : blood "pic.tny"
 ,10
120 SCR=physic : STRT=start(10)
130 gosub 1010
140 end
1000 rem TINY Compact
1010 REZ=peek(STRT) : If REZ>5
 then print "incorrect screen
 mode" : end
1020 If REZ<3 then CBYTES=peek
 (STRT+33)*256+peek(STRT+34)
 : TEMP=STRT+1 : STRT=STRT+
 37
1030 If REZ>2 then CBYTES=peek
 (STRT+37)*256+peek(STRT+38)
 : TEMP=STRT+5 : STRT=STRT+
 41 : REZ=REZ-3
1040 If (REZ=0 or REZ=1) and
 mode<>2 then mode REZ
1050 If mode<>REZ then print
 "incorrect screen mode" : end
1060 for I=0 to 15 : colour I,
 peek(TEMP)*256+peek(TEMP+1) :
 TEMP=TEMP+2 : next I
1070 MEM=STRT+CBYTES :
 SCRMEM=SCR
(Cont. Page 15)
```

# ABSOLUTE BEGINNERS

## Absolute Beginners

- #4 in a series of 1,000,000

In the last issue we programmed a very simple dice game, using some straightforward Basic commands. This issue, we will be going over some new commands, the **FOR.TO.NEXT** loop and the **GOSUB..RETURN** commands.

### □ FOR.TO.NEXT

If you wanted your program to do something several times over, you wouldn't want to type in the commands several times over, would you? This is where the concept of **LOOPS** comes in. The idea of using a Loop is to make the program repeat a command or series of commands a number of times. For instance, if we wanted to count from 1 to 10 on the screen, we would want to do this in pseudo-code:

- 1) Start count at 1
- 2) Print count to screen
- 3) add one to count
- 4) If count is less than or equal to ten go back to (2)
- 5) otherwise stop

If we wanted, we could use a variable and an **IF.THEN** statement to do the above. e.g

```
10 COUNT=1 : Rem start COUNT
 at 1
20 Print COUNT
30 COUNT=COUNT+1 : Rem adds
 one to COUNT
40 If COUNT<=10 then goto 20
50 end
```

Note the use of the **<=** symbols in line 40. In Basic, the symbols **<** and **>** refer to less than and greater than, **<=** refers to Less than or equal to, **>=** means greater than or equal to and **<>** means not equal to.

The above program is not very structured or easy to read, and it could be shorter. This is where the **FOR.TO.NEXT** loop comes in.

**FOR.TO.NEXT** is actually made up of 2 commands, the **FOR.TO** command tells STOS what to count from and to, and the **NEXT** command adds the correct step to the counter and if it is still within the defined range will jump back to just after the **FOR.TO** statement.

```
So for our above program we could do
10 For COUNT = 1 To 10 : Rem
 count from 1 to 10 (the number
 is stored in variable COUNT)
20 Print COUNT
30 Next COUNT : Rem add one
 to count and jump back to line
 20 if it is <=10
40 end
```

You can use any variable instead of **COUNT**, and you can also **NEST** loops within each other. e.g

```
10 For A=1 to 10
20 For B=1 to 8
30 Locate A,B 40 Print "*";
50 Next B
60 Next A
```

The above program counts from 1 to 10 in variable **A**, every step it does for **A**, it counts from 1 to 8 in **B**. The **Locate** command positions the text cursor at **A** columns and **B** rows, and then an asterisk is printed (The semi colon after the print statement stops a return character being printed). The end result is a square of asterisks on the screen.

Note that you cannot do the following nest:

```
10 For A=1 to 10
20 For B=1 to 8
50 Next A
60 Next B
```

as this would give you an error (Work

# ABSOLUTE BEGINNERS

---

your way through the program and see for yourself what is happening).

Another useful trick with **FOR TO NEXT** is that you can change the amount it counts in, with the optional **STEP** parameter. If **STEP** is omitted, then the count will be in ones. The **STEP** parameter comes after the **TO** parameter, i.e

```
10 For A=1 to 10 step 2
```

The above line will count from one to ten in steps of 2. You can also use **STEP** for counting backwards. i.e

```
10 For A=10 to 1 step -1
20 Print A
30 Wait 50
40 Next A
50 Print "Blastoff !"
```

note the use of the step -1 to count backwards in steps of 1. The **Wait 50** pauses the program for one second (50 50ths of a second).

## □ GOSUB.RETURN

If you remember the last issue of Absolute Beginners, you'll remember that we used a **GOTO** statement to jump to a previous line in the program to do a loop. **Goto** can be used easily to get to other parts of the program. For instance:

```
10 Input "Type something in and
press RETURN :";A$
20 goto 100
30 end
100 For A=1 to 3
110 Print A$
120 Next A
130 goto 30
```

If you trace what the above program is doing, you will see that it lets you enter some text into the string variable **A\$**, it then goes to line 100 where it does a count from 1 to 3, Prints **A\$** out and then goes back to line 30 where the program ends.

But what if we wanted to use the

routine at line 100 more than once ?

```
10 Input "Enter a string ";A$
20 Goto 100
30 Goto 100
40 end
100 For A=1 to 3
110 Print A$
120 Next A
130 goto 30
```

If you ran the above program, it would keep looping back to line 30 and you would have to use **ctrl-C** to stop it. What we need is some way of going to our subroutine, and then returning back to just after the jump. This is what the two commands **GOSUB** and **RETURN** are for:

```
10 Input "Enter a string ";A$
20 Gosub 100
30 Gosub 100
40 end
100 For A=1 to 3
110 Print A$
120 Next A
130 Return
```

In this program, line 20 goes to the subroutine at line 100, which will execute until it gets to the **RETURN** statement, which tells the program to go back to just after where the subroutine was called, i.e line 30, where it goes to the subroutine again, and returns to the next line, which is line 40. You must make sure with subroutines however, that the program never gets to them without a **GOSUB** or you will get a **RETURN WITHOUT GOSUB** error (Try removing line 40 to see what I mean).

Until the next issue, experiment with using **FOR NEXT** and **GOSUB RETURN** statements. Have fun!

# CALCULATOR ROUTINE

## Calculator Routine

The following is a very simple calculator routine, as requested by Richard Gale, in return for his articles for the newsletter. All you do is type in a mathematical function, as if you were using a calculator, but inserting spaces between values and functions e.g

1+2^=

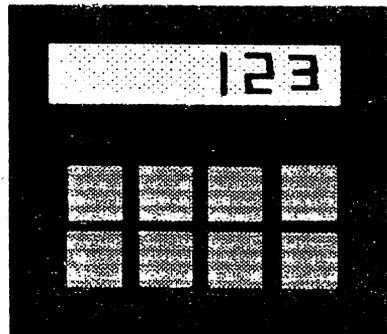
where the ^ character shows the spaces. The program will accept +,-,/ and \* as well as negative numbers. Although it is only set up to do integer maths, it would be very simple to change it to floating point maths.

The calculator also accepts **M** (Memorise current value), **R** (Recall memory) and **S** (Swap Memory with current value).

You could use this routine in a calculator accessory (graphically driven of course!) or in an educational maths game.

```
10 rem very simple calculator routine by Aaron Fothergill
20 rem calculation string in C$, works with +,-,* & /
30 rem includes M (Memorise) and R (Recall) also S (Swap mem and current value)
40 rem each number or function must be seperated by spaces
50 line input C$
60 MEM=0 : rem memory
70 P=1 : rem pointer in string
80 repeat 90 S=instr(C$,"P) :
 if S=0 then S=len(C$)+1
100 F$=mid$(C$,P,S-P) : if F$="+" then gosub 220 : goto 200
110 if F$="-" then gosub 250 : goto 200
120 if F$="/" then gosub 280 : goto 200
130 if F$="*" then gosub 330 : goto 200
140 if F$="=" then gosub 350 :
```

```
goto 200
150 if F$="M" then STRE=MEM : goto 200
160 if F$="R" then MEM=STRE : goto 200
170 if F$="S" then swap MEM,STRE : goto 200
180 if MDE>0 then on MDE gosub 240,270,300,340 : MDE=0 : goto 200
190 MEM=val(F$)
200 P=S+1 : until P>len(C$)
210 goto 50
220 MDE=1 : rem mode 1=add
230 return
240 MEM=MEM+val(F$) : return
250 MDE=2 : rem mode 2= subtract
260 return
270 MEM=MEM-val(F$) : return
280 MDE=3 : rem mode 3= divide
290 return
300 if val(F$)=0 then print "Division by Zero Error !" : goto 50
310 MEM=MEM/val(F$) : return
320 MDE=4 : rem mode 4= multiply
330 return
340 MEM=MEM*val(F$) : return
350 print "Result =";MEM : return
```



V  
2  
1

# TOME

V  
2  
1

## TOTAL MAP EDITOR V2.1

Do you want to write professional map based games, like Gauntlet, New Zealand Story, Rainbow Islands etc. ?

Did you know that it's possible to do so using STOS ! The TOME (Total Map Editor) system gives you 18 extra commands for STOS that enable you to write high speed, pixel accurate scrolling routines just like those used in all the major games ! Over 90% of games written on the ST are map based for their backgrounds or play area, and TOME gives you the power to use maps in your games, saving you huge amounts of memory and giving you a major speed advantage.

The brand new version of the Total Map Editor has now been released, including an even more powerful editor, extra commands and two new demo games, including sourcecode and data files. Also included are example programs and a scrolling demo written using TOME.

TOME's features are:

- Single Pixel Scrolling in all directions (Includes simple STOS example to show you how it's done !)
- Over 240 screens can be stored in 64K of memory !
- Block Animation within your maps (So you can do HUGE animations !)
- The MOST POWERFULL MAP EDITOR AVAILABLE on the ST !
- Full 28 Page manual included, and example files on disk, not to mention the two demo games CUTE CUDDLY PURPLE BABY DRAGON GOES FLOWER ARRANGING, and JITTERBUGS ][.

*"Both Games Galore and TOME deserve to become an integral part of any STOS Programmer's library."* Atari ST user March 1990

*"If you don't buy it (TOME) you must be raving mad !"* Peter Hickman STOS Newsletter Issue 10

**Only £14.95 to STOS Club Members** (£19.95 to Non Members) Add £1 for P&P if ordering from overseas.

TOME was written by Aaron Fothergill. Send U.K cheque, Postal Order or International Money Order to:

Shadow Software, N.Devon.



# LEARNERS



## THROWING THINGS

If you can remember back as far as issue 8 of the newsletter, you'll remember Professor Speck's article on the use of pseudophysics in games writing. Well this issue, you'll actually get to use some!! I will be showing you how to use gravity effects on a game object, whether it is the player, a cannon shell, or a parachuting cute cuddly purple baby dragon (who's just been flower arranging).

The first thing to get straight, is what gravity is. Gravity makes things go down (most of the time). Things like rocket engines, lift from wings and jumping do a very good job of opposing gravity in their different ways, and things like parachutes help to slow down or control gravity's effects (assuming you have an atmosphere or you are cheating).

O.k, for this lesson, you'll need some sprites, using the default palette to make life easier.

**1= a ball**

**2= a ball with a parachute out of the top.**

the hotspots for both sprites should be in the centre of the ball, which should be approx 4x4 pixels in size.

### □ Program 1

#### Jumping Ball

In this program, we shall make the ball jump from the ground, and then fall back down again with gravity. We will be using 4 variables, **X** & **Y** to store the position of the ball and **DX#**, **DY#** to store the vectors of the ball. The variable **DY#** will change as gravity pulls on the ball.

**10 Mode 0 : key off : curs off : hide on**

**20 X=20 : Y=190 : DX#=4 : DY#=-8**

**30 Ink 2 : bar 0,190 to 319,199 :**

```
rem draw the ground
40 sprite 1,X,Y,1 : Update : wait
vbl : rem slow it down !
50 X=X+DX# : rem add X vector
60 Y=Y+DY# : rem add Y vector
70 DY#=DY#+0.2 : rem effect Y
Vector by gravity
80 DY#=min(8,DY#) : rem make
sure DY# is below terminal
velocity
90 If Y<190 then 40
```

Note that we have started the **Y** vector **DY#** off with a value of -8, which will cause the ball to go into the air, as gravity moves **DY#** towards the positive, the ball will slow down, and then start to fall, until **DY#** reaches its maximum value of 8 (done by line 80) which is the ball's **TERMINAL VELOCITY** which means that the ball isn't going to get any faster because the air pushing against it is equal to the pull of gravity. If the ball was thrown on the moon, where there is no air, there would be no terminal velocity, and you could remove line 80.

Things like parachutes lower the terminal velocity, so try the following modifications. We will be using the variable **S** to contain the sprite image number used (1 when the parachute is closed and 2 when open). The chute will open when **DY#** gets greater than 1, and the variable **PD** contains the amount of drag the parachute is causing.

**15 S=1**

```
40 Sprite 1,X,Y,S : Update : Wait
Vbl
```

```
75 If DY#>=1 then S=2 : PD=6
```

```
80 DY#=-min(8-PD,dy#)
```

Once you have tried the above modification, you can vary the speed at which the chute will open, and the amount that it will affect the terminal velocity by changing line 75.



## □ Program 2

### The Trained Ball

Next, we shall make the ball jump on command, when the fire button is pressed. By setting not using the X vector, the ball will jump on the spot.

```

10 Mode 0 : key off : Curs off :
 hide on
20 X=160 : Y=190 : DY#=0 : rem
 note that DY#=0 so the ball
 isn't moving
30 ink 2 : Bar 0,190 to 319,199
40 sprite 1,X,Y,1 : update : wait
 vbl
50 Y=Y+DY# : Y=min(190,Y) : rem
 force ball to stop when it gets
 to the ground
60 IF Y<190 then DY#=DY#+0.2
70 DY#=min(8,DY#) : If Y=190
 then DY#=0
80 If Fire and Y=190 then DY#=-8
90 Goto 40

```

Note that we've only allowed the jump button to work, if the ball is on the ground (when Y=190). To make the ball jump, DY# is set to -8 (i.e. up rapidly). Line 60 checks to see if the ball is in the air, if so gravity modifies DY# (you can adjust the amount of gravity of course). Line 70 does our Terminal Velocity fix and checks to see if the ball is on the ground. If it is, then DY# is set to zero.

O.k, now it's your turn. First modify program 2 so that the ball uses a parachute, then try modifying it so that the ball bounces upon hitting the ground. Answer to the bouncing problem next issue!

**For More Details on Pseudophysics read Professor Speck's Article in Issue 8 of the Newsletter.**

```

1080 XPOS=0 : OFFSET=0 :
 SCANLINE=1 : COUNT=0
Tlajpic Decompactor
(Cont. From Page 9)
1090 repeat
1100 BYTE=peek(STRT+COUNT) :
 inc COUNT
1110 If BYTE=0 or BYTE=1 then
 TEMP=peek(STRT+COUNT)*256+
 peek(STRT+COUNT+1) : COUNT
 =COUNT+2 : FLAG=(BYTE=0)
1120 If BYTE>=2 and BYTE<=127
 then TEMP=BYTE : FLAG=true
1130 If BYTE>=129 and BYTE<=255
 then TEMP=256-BYTE :
 FLAG=false
1140 If FLAG then BYTE=peek(
 MEM)*256+peek(MEM+1) : MEM=
 MEM+2
1150 for I=1 to TEMP
1160 If not(FLAG) then BYTE=
 peek(MEM)*256+peek(MEM+1) :
 MEM=MEM+2
1170 doke SCRMEM,BYTE
1180 inc SCANLINE : If
 SCANLINE<201 then SCRMEM=
 SCRMEM+160 : goto 1210
1190 inc XPOS : If XPOS=20 then
 XPOS=0 : OFFSET=OFFSET+2
1200 SCANLINE=1 : SCRMEM=
 SCR+XPOS*8+OFFSET
1210 next I
1220 until COUNT=CBYTES
1230 return

```

## ***MONEY!!!***

Yes, we are actually paying for articles and programs for the newsletter. If your article or program is published in the newsletter, you will be paid £5 per printed page!

# PROF SPECK O.D.D

---

## **BUG HUNTING**

**This issue, the Professor Speck interviews the famous bug hunter Sir Hugh St Axe, who has just returned from a programming trip to the Congo.**

**Sir Hugh, what do you do as a bug hunter ?**

*Well Prof, I mostly tend to do Interviews. Although from time to time I search through programs for interesting new bugs.*

**What ones have you found ?**

*Oh all sorts ! I normally find the usual everyday ones caused by dodgy programming, especially the dreaded OBOB (Off By One Bug) and quite a few OBSB (Off By Several Bug), but I've found some very rare ones in my time.*

**They named an error message after you didn't they ?**

*Yes, but they spelt it wrong !*

**Shame ! Could you tell us about some of the rarer bugs you have encountered ?**

*No !*

**Oh go on !**

*Only kidding ! As you pointed out, I have just returned from an expedition to the Congo, where I was helping to hunt down a bug a missionary was having with his STOS program. He was trying to write a program that called a few of the custom trap routines in the back of the STOS manual, but was getting address and bus errors all over the place. After a few minutes of investigation and several cups of tea, I traced the bug.*

**What was it ?** *It was the fairly common, "Oh by the way, there's a mistake in the manual that we didn't tell you about" bug, where the STOS manual lists traps 4*

*& 6, which got removed about 10 minutes after the manual was completed.*

**Any other rare ones ?**

*The worst one I encountered recently was the Phone Bug. This bug is caused by trying to dictate a program to someone over the telephone. The typist at the other end tends to mis-hear you, and thus make mistakes, which you have to go back and correct. Thus this little devil strikes in two ways: First it causes other bugs in your program, then it causes you to get monstrous phone bills. Very nasty bug that one ! If you have to work over the phone, or dictate to a typist, then I suggest that you make clear on some commands such as INC and INK which sound the same, which one you want. Also make sure that the typist knows the programming language and its St Axe to a reasonable degree.*

**How to you go about hunting down bugs then ?**

*This is a long one, can I stop talking in Italics ?*

O.k !

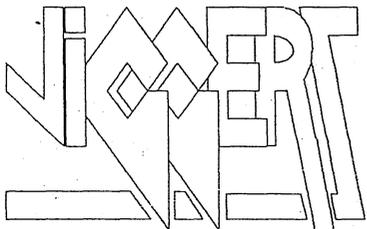
Thanks ! The job of bug hunting isn't one for people with nervous disorders, as the little blighters can sometimes stay hidden for weeks, suddenly popping out at you when you least expect them (usually just minutes before a deadline). Normally though, you can use just a few techniques to catch them.

**1) The "See what it's doing" technique.**

This is a trick used when things in your program seem to be doing strange things, or errors like illegal quantity errors keep occurring. If you are getting a definite error (Not a Bug, bugs make the program go wrong, they aren't specific errors) on a particular line, then check all the values **(Continued on Page 19)**

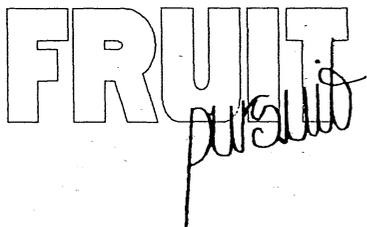
## DIGITAL DIMENSION ANNOUNCE

### THE LAUNCH OF A RANGE OF BUDGET STOSWARE



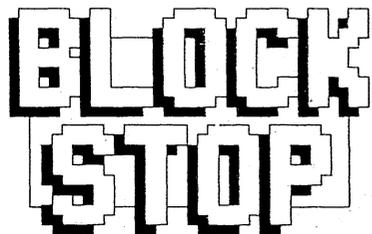
OUT NOW. JIGGERS is a cunning puzzle game where you have to get all the blocks the same colour. Sounds easy doesn't it ? But JIGGERS will blow your brain.

The game features 10 levels of play against the clock, sampled sound and bags of mind bending fun!



OUT NOW. FRUIT PURSUIT, a game that combines the addictive qualities of a fully featured fruit machine with the strategy of a traditional board game.

Packed with features, FRUIT PURSUIT has nudge, hold, autonudge, fruit stop, fruit advance, trail stop plus many more. Addictive play for 1-4 people.



Coming soon! BLOCK STOP is a cross between the old classic - Breakout and the even older classic - Blitz ! All played on a 3D grid. Sounds interesting?

Featuring multiple levels of play, bonus blocks, limited fuel and limited ammunition.

**ALL GAMES ONLY £4.99 + £1 P&P**

EXCLUSIVE TO STOS CLUB MEMBERS : FREE SOURCE  
CODE SUPPLIED WITH EVERY GAME.

*Digital  
Dimension*

For more information, or to place an order write to  
DIGITAL DIMENSION, WINTERBOURNE, BRISTOL

# COMPETITION

## The Great STOS Newsletter Reader Survey

The STOS Newsletter is about to go into its third year of publication, and we thought that you might like to put forward some of your ideas and views. Simply complete this survey and send it to the STOS Club at the usual address. All the forms returned will be put in a hat, and the first one drawn out will win a free subscription for Issues 13-18!

How do you rate the following series of articles?

|                    | Dislike                  | Like                     | More Please!             |
|--------------------|--------------------------|--------------------------|--------------------------|
| Absolute Beginners | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Learners           | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Extending STOS     | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Professor Speck    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 10-Liners          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Reviews            | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

What memory size is your ST?

1/2 meg  1-2 meg  even more meg

Do you have a second floppy drive?

No  Yes

Do you have a hard disk (and what size)?

No  Yes.....Megs

What STOS Products do you own?

|                   |                          |                  |                          |
|-------------------|--------------------------|------------------|--------------------------|
| STOS Compiler     | <input type="checkbox"/> | STOS Maestro     | <input type="checkbox"/> |
| STOS Maestro Plus | <input type="checkbox"/> | STOS Sprites 600 | <input type="checkbox"/> |
| TOME              | <input type="checkbox"/> | Games Galore     | <input type="checkbox"/> |
| Cartoon Capers    | <input type="checkbox"/> |                  |                          |

What resolution do you normally work in?

Low  Medium  High

Please give ratings of your own skills in the following

|                   | Bad                      | Good                     | Excellent                |
|-------------------|--------------------------|--------------------------|--------------------------|
| STOS Programming  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 68000 Programming | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Graphics          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Music             | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Games Design      | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

If you answered Excellent to any of the above, would you wish your name and address to be put on a contact list, so that we can put you in touch with other programmers/graphic artists etc when they require your skills for a project?  No  Yes

And finally, what is your name and address?

.....  
.....  
.....  
.....

# PROF SPECK O.D.D

(Cont. from page 16)

of the parameters used on that line. For instance, in the line

```
10 Draw X,Y to X2,Y2*Z
```

You would check all the variables **X,Y,X2,Y2** and **Z** for their values, and also **Y2\*Z** to make sure that they all come within the boundaries of the command (in this case, **X2** and **X** should be from **0-319** or **0-639**, **Y** and **Y2\*Z** should be between **0-199** or **0-399** depending upon the resolution you are running in). The bug isn't always on this particular line though, so you may have to hunt back several lines to find out what the variables are doing. You will, however, find out which variable is causing the error, so that you can check up on it. This is where the next bug hunting technique comes in:

## 2) The "Follow that variable" technique.

This is where you get output the contents of the variable at strategic points in the program (like just before the error is occurring). The **Follow** command itself isn't much use (I know very few programmers who actually use it), as it doesn't care where it outputs the values. If you are working on a game for instance, where you are using screen flipping, the only way you can display a number is to print it to the background screen and then copy it to the physical screen (otherwise it flashes on and off). Also the bug may be occurring only when the variable changes to a funny value for just one frame. Trying to spot a value that changes every 50th of a second is tricky. There are only 2 ways of checking the variable. 1) Put a **HUGE For...Next** loop in the program to slow it down to a watchable speed, and wait for about 2 hours for the bug to occur, or 2) Output all the values of the variable to a large array, which you can then print at your

leisure. e.g

```
10 Dim T(5000) : TP=0 : rem TP is
the pointer for the T() array
```

```
:-
:-
```

```
50 T=T*X+12 : Rem this is the line
we suspect of going wrong
```

```
51 T(TP)=T: Inc TP : rem so we
store the value of T at this
point
```

```
:-
:-
```

```
program ends
```

```
9000 For A=0 to TP : Print T(A):
while mouse key<>0: Wend
```

```
9001 Next A : end
```

Now all you need to do is do a **GOTO 9000** after your program ends or is stopped, to print out all the values that **T** has gone through. Another (simpler) way of checking for a variable doing crazy things is to use an **IF...Then** statement that will only output the value if it is out of the expected range. e.g

```
100 If T>1000 then locate 0,0:
Print T;" " : Wait key
```

Note the use of a couple of spaces after the variable. This ensures that if the variable is printed more than once, you don't get problems with digits from the previous display being shown.

3) Stepping. This is a favourite of machine coders, and simply means that you put something like a **WAIT KEY** command in your main loop, so that you can go through the program in single steps, thus spotting any errors (like the wrong sprite on a particular frame). If you want to be flash, you can do a little routine to switch your stepping mode on and off e.g

```
50 A$=inkey$: rem if you already
have an inkey$ function in the
loop, then use the variable it is
stored in instead.
```

# PROF SPECK O.D.D

```
51 if a$=chr$(13) then stp=i-stp :
 rem RETURN toggles step mode
52 if stp then wait key
```

4) Using **On Error Goto**. When a STOS program is compiled, if an error occurs, it will display an error number and then return to the desktop after a key is pressed. Another problem is that you can't print out variable values, or the line number it stopped on. However, if you use an **ON ERROR GOTO** statement, upon an error occurring, the program will jump to your routine, and both **ERRN** (error number) and **ERRL** (Error Line) variables work, plus of course your error trapping routine can print out any other variable values you want. e.g

```
1 On Error Goto 2000
:
:
2000 default : Print "error
#";errn;" in line ";errl
2001 Print "X=";X;" Y=";Y;"
Z=";Z
2002 Print "Press a key":clear
key : wait key
2003 end
```

This error routine will display the error number and line, plus the values of the variables X, Y and Z before going back to the desktop after waiting for a key. This

can be very useful when writing compiled programs.

That's all from me for now. I have to go in search of the Little Green Demon bug, which was previously thought by experts to be a mythical beast that was blamed by just about every programmer for mysterious bugs. However, I have been given photographic evidence to prove that Little Green Demons are currently thriving in one of the Macintosh computers at Mandarin Software's office, having caused it to blow up! Although it might also be a case of the "I think we bought the wrong computer" bug, which happened to NASA's Space Shuttle program. In a well documented case, 5 identical computers on board the shuttle, which were supposed to monitor each other for faults, and take over if one shut down, all shut down at once!

*Sir Hugh St. Axe is a full time bug hunter. He is currently writing a follow up to his first book "Error messages of the Bratislavian XZ18 2 bit micro computer" which sold all of 3 copies. His new book will be titled "Understanding and debugging Freestyle Spagetti code" which will include several example programs, be 2304 pages long and probably be plugged on Wogan.*

## STOS Paint Master

• **The Most powerfull art package ever written for STOS Basic**

**512 Colours on screen from within your STOS Programs !**

**Easy to use icons, pixel cut and paste, text, boxes, round-boxes, circles, ellipses, brushes, lines, colour ramp etc.**

**Load Neo,PI7,PC7 and PAC files as well as Spectrum 512 pictures (Converter coming soon).**

**Send £14.95 (Overseas add £3) to Stallion Software, Plymouth, Devon,**

**to Stallion Software.**

# REVIEW

## T O M E V2.1

### Review by Peter Hickman

Hi there, It's Peter Hickman here, no you haven't got the wrong magazine!! I'm doing a little guest spot in this issue of the newsletter to review a new (ish) product for STOS called TOME. The simple reason for me doing this rather than the ED is 'cos he wrote it!!!

TOME isn't exactly new, I have been using it for nearly a year, and in that time it has advanced from little more than an enhanced version of the STOS map editor (written by Stephen Hill) to the most comprehensive map creation/editing system available for almost any computer (yes, even the Amiga!). This review is for version 2.1, which includes an enhanced editor, two new demo games and a myriad of new commands to make games writing easy (well, less painful anyway!).

Before you can use TOME the new commands must first be installed, this simple procedure is controlled by a small program found on the TOME boot disk. The process is entirely automatic and all you are required to do is insert the appropriate disk at the correct time.

The main part of TOME is the editor which uses pre-drawn picture tiles pasted together to construct large "maps", this type of game construction is quite memory efficient and can be seen to good effect in games like Gouls & Ghosts, Rainbow Islands or even Yomo (see the Games Galore review).

To create the tiles Degas Elite, Neochrome, STOS Paintmaster or another similar art package must be used to draw small blocks of 16x16 pixels (low res.),

32x16 pixels (Medium res.) or 32x32 pixels (high res.). There are functions to "touch up" tiles from within TOME but these are not very advanced and should really be used only in an emergency. Previous versions of TOME supported 'BLOCK' commands, these allowed whole blocks of tiles to be grabbed and saved, nothing has changed for the update and those mega-blocks are still there, although a few new uses have been thought up for them including a pseudo sprite effect (look out for the new Hot-Dog demo coming soon!!).

Once you have got past the graphic design stage you have to tackle the editor, the creator of TOME (the prolific Aaron Fothergill) has opted for an icon driven menu system for all the major (and minor) functions, the reason for this (according to the manual) is that it saves memory and works in all three resolutions, but then again it looks pretty nifty too. Everything is icon driven, and the overall system it is quite similar to the STOS Sprite Editor, only much better.

The editor acts as an advanced drawing package, the only difference between it and something like Degas Elite is that you can only use pre-drawn tiles as your brushes. Most of the features you find in art packages are also found in TOME including the ability to draw circles, fill areas and block cut & paste. The interesting thing about any blocks you may cut out is that they can be saved and used within any programs you write to paint directly to the map, so if a player blew up a building in your latest hit game you could alter the map accordingly by plotting a block containing an explosion picture in its place.

The editor is one of the most well thought out pieces of serious software I have seen in ages, it even includes a locator mode where you can store a

# REVIEW

---

point in the map you are currently editing, move away from it, and instantly move back to that point with the press of a key. You can use a joystick, the cursor keys or some pointer arrows located on the top right hand corner of the screen to move around any maps larger than the screen. Probably the most outstanding point of the editor is it's "NICENESS" mode. This allows you to change the colours of the menus and text that TOME uses, so if the colour scheme does not suit your taste or your tile palette blanks out most of the screen all you have to do is use "NICENESS", other software developers please incorporate this feature into your future graphic utilities and art packages. Probably the most major changes to the editor since the last version has to be the ability to load and save in a variety of graphic formats, not just DEGAS and NEO but TINY and DEGAS COMPRESSED are now supported making the whole map creation business a whole lot easier for users, who like me, only have a floppy drive to work from.

The most significant feature of TOME V2.1 over previous versions is the amount of new commands, Aaron has listened to his users and included suggestions from people all over the globe. I haven't got the room to go into a long list so you will have to trust me when I say that all of the commands are adequately documented with a few examples in the manual and on the two discs that come with the package.

The discs contain two new demo games, Cute Cuddly Purple Baby Dragon Goes Flower Arranging and Jitterbugs ][. CCPBDGFA (guess what that's an acronym for!) can only be described as the cutest game to hit the shops since New Zealand Story, I won't go into the logistics of the games, suffice it to say that OCEAN had better watch out.....

Apart from the two new games (which replaced the quite brilliant - but slightly bugged - game TIN GLOVE) the two discs contain example programs for low and hi res, small routines to smoothly scroll a TOME map in eight directions (even the fabled AMOS has trouble doing really smooth scrolling this easily) and a few other bits and pieces. All of this adds up to a very comprehensive package which is well worth your hard earned dosh. About the only thing that lets the package down is the rather poor spelling, although Aaron assures me that a recent purchase of First Word Plus will allow him to spell check things in the future.

TOME is an essential purchase to anybody who is going to consider writing a scrolly game, it's fast, uses hardly any memory and comes complete with access to the programmer 'cos you know who wrote it (that's where the STOS helpline comes in!). What can I say, if you don't buy it you must be raving mad, of course if you had entered my STOS programming competition in **NEW ATARI USER** (put those words in bold please Aaron!) - which just happens to run a really good regular STOS column filled with programs and tips

every two months - you could have won a copy. Oh and before I pop off back into the land of the rising AMOS Aaron has just informed me that he is just putting the finishing touches to a TOME PD utility disc, it will be filled with loads of goodies like a program to grab maps from commercial games (I can't wait, drool drool).

**TOME V2.1 is available from**

**Shadow Software, Barnstaple,  
N.Devon. and costs £14.95**

for STOS Club members (£19.95 for non members). If ordering from overseas, please add £1 for the extra P&P.

# PUBLIC DOMAIN

## HOW IT WORKS

Listed on this page you'll find the very latest additions to the STOS Public Domain Library. See Issues 7, 8 and 9 for other titles in the library. SPD discs cost £2 UK, £2.25 Europe and £2.50 rest of world for airmail. DPD discs cost £2.50 UK, £2.75 Europe and £3 rest of world for airmail. Shareware prices are shown after the disc descriptions, Europe add 25p to the price and Overseas add 50p per Shareware disc. Deduct £1 if you supply your own VIRUS FREE discs. If you order three discs or more, deduct 20p per disc including the first three, a saving of at least 60p. Each disc has a specially printed STOS Public Domain label so your PD discs will match the rest of your STOS master discs.

If you have any programs which may be suitable please send them along and don't forget to choose a disc from the library in exchange for yours. Please make sure that all contributions are STE compatible. STE owners look for "STE" in brackets after the disc description.

Please note the new address and telephone number.

STOS Public Domain Library  
c/o Sandra Sharkey  
WIGAN

C h e q u e s /  
PO's should be crossed and made payable to STOS Public Domain Library. We cannot take credit cards or payments made out in

other currencies.

## Choose from the following:

**SPD-25:** New Atari User Listings + Skystrike Plus playable demo. - The Skystrike Plus playable demo has been added to this disk of Peter Hickman's listings from his STOS column in New Atari User. **(BAS. PRG. SAM. STE.)**

**SPD56:** Zoo-Maker by Eric Elson -

This program builds up a list of animals and questions for you to tell the difference between the animals. You can even draw a picture of the animal to go in the picture gallery. Great fun for kids. **(PRG. STE.)**

**SPD57:** Mix 'n' Match by Keith Feeney - Test your memory in this two player game with three levels of difficulty. Using the mouse click on two squares to get matching pictures.

Good use of sound and excellent graphics. Highly recommended.

**(BAS. PRG. STE.)**

**SPD58:** Plumb Crazy by Keith Feeney - Similar to the popular "Pipemania". Join the pipe pieces together to form a continuous pipe. Another well produced game from Keith. Highly recommended. **(BAS. PRG. STE.)**

**SPD59:** Red War by Stephen Brennan - A one or two player Battleship type game. The disc contains excellent documentation to set the scene and explain the game. Good use of sound effects and a nice screen display.

**(PRG. BAS. STE.)**

**SPD60:** TEU Utility & Slideshow by Chris Marett - Contains three programs. A 50/60 Hz screen switcher which gives you a bigger screen area and claims to give better colour and pixel definition (Only on a colour MONITOR). TEU Slide>Show picks up and displays Neo, Pi1, Pi2 and MBK files. You are given the option of choosing the screen effect and the delay between pictures. TEU Program takes over some of the jobs from GEM and other utilities, 14 in all, e.g. amount of free disc space, list directory, make folder, Degas to Neo converter to name just four. **(PRG.)**

**SPD61:** Heatseeker, Starwars and

# PUBLIC DOMAIN

Reken Accessory by Peter van den Houten- Heatseeker is a 2 player game. Starwars is a shoot'em up (push the joystick left for shield). Reken is a calculator accessory for STOS.

**(BAS. PRG. STE.)**

SPD62: Memory, Lingo and Risk by Peter van den Houten- Memory for 1-4 players, find the pairs. Lingo for 1-4 players, guess the 5 letter word. Risk is a 2 player strategy game, put simply the one who has 35 American states wins. **(PRG. BAS. STE.)**

SPD63: RMS by Bruno Azarri - A two player game from the author of Silder. Based on one of the great Mr. Rubik's puzzles. **(BAS. PRG. STE.)**

SPD64: STOS Paint Master Demo from Stallon Software- Working demo of this excellent paint package with save and print features disabled. Now you can try before you buy. Also on the disc is Reels of Fortune written by Terry Mancey and Keith Hearson. A one-armed bandit game with basic file. Highly recommended.

**(PRG. BAS. STE.)**

SPD65: Pile Up by Russell Moll- A Tetris clone with two skill levels. There are a couple of surprises if you finish level 6 and if you win the game by completing level 10. A good version of Tetris but not as easy as the original. There is a pause feature and you can choose the speed of the joystick. Recommended for Tetris fans. **(PRG.)**

DPD22: Predator Demo by The Islander (P. Smart) - If you've seen the film then you'll really appreciate the demo. Recommended. **(PRG.)**

DPD23: Mars Maze by M. Tomlinson Cute game, reminded me a little of Pacman. The disc contains the full listing so you can see how it was done. **(BAS. STE.)**

DPD24: Dungeon Quest by Frank Carr Explore the dungeons in your quest to save the beautiful Sateen.

**(BAS. PRG. STE.)**

DPD25: Demos from France - 6 demos which were originally listings in the French Atari Magazine.

**(BAS. PRG. STE.)**

DPD26: House Music Demo from France - Three tunes **(PRG.)**

DPD27: Warning Sign 2 Demo from France - **(PRG.) - 2 discs** £4.50.

DPD28: Megamix Demo from France - **(PRG.)**

DPD29: Warning Sign 3 Demo from France **(PRG. STE.) - 2 discs** £4.50

DPD30: Metallica-The Call of Ktulhu by Tony Longworth from the author of Phantom of the Opera. Highly recommended.

**(Imb. PRG. STE.)**

SH11: Antimatta by Bob Spreadborough Arcade shoot'em up. Shoot the balls of Antimatter to get shields, lasers, guns, fuel etc. but don't let them touch you! 9 level joystick controlled game. Single sided disc. **(PRG.) - £2.95.**

SH12: Xalaracx by D. Kiley- Play against the computer and the clock. Licensed to Goodmans and STOS PD Library. Single sided disc. **(PRG. STE.) - £4.00.**

SH13: Dizzy Lizzle from Budgie UK- Excellent graphics in this Boulderdash type game. Lots of humour and very addictive. Highly recommended. Single sided disc. **(PRG. STE.) - £2.85.**

SH14: Bounty Hunter from Budgie UK - Role playing game set in the wild west. Ride from town to town searching out the 20 most wanted men to make the west a safer place to live. Single sided disc. **(PRG. STE.) - £2.95.**

SH15: Big STOS Demo by Richard Gale This disc contains the sourcecode for DPD20. Double sided disc. **(PRG. SAM. BAS. STE.) - £3.45.**

SH16: NEW TOME DEMO V2.1 by Aaron Fothergill- Licensed exclusively to the STOS PD Library. Includes Demo version of the TOME editor (Save disabled, and times out after 3 minutes) as well as a playable demo version of Jitterbugs [[ and a Scrolling text demo with huge text. Highly Recommended **(PRG. STE.) - £3.00**

This Newsletter was compiled and edited by Aaron Fothergill using Timeworks DTP on the ST. All articles are by him unless stated. Printing and P.D Library by Sandra Sharkey.