



DRAG'N'DROP

RISC OS **Pi** and all RISC OS 5 machines

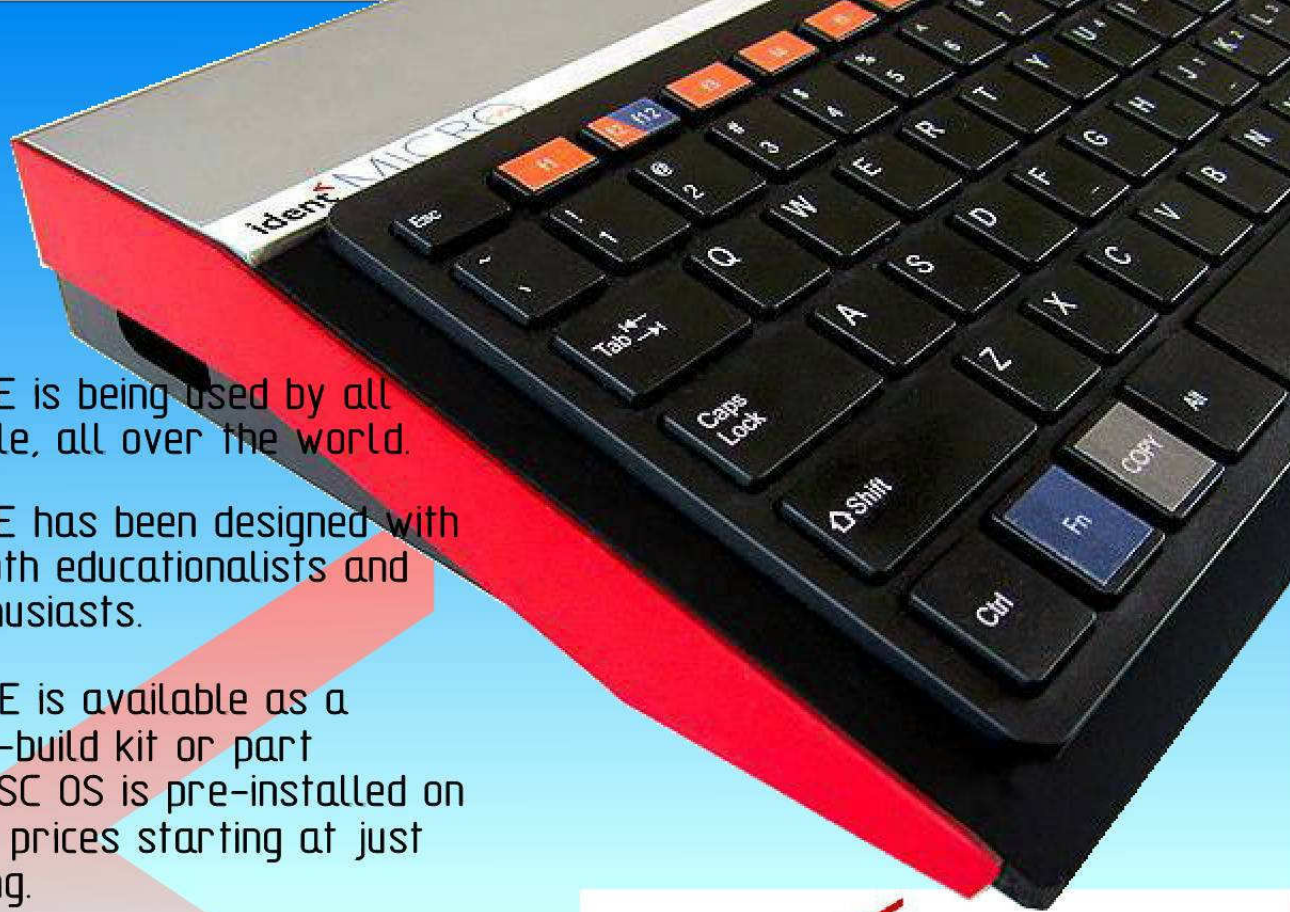
April 2017
Volume 8 Issue 3
£3.50

RiscDSP

Wimp programming
Desktop OXO
SWILister
Python

Type 'em all in!





The MICRO ONE is being used by all sorts of people, all over the world.

The MICRO ONE has been designed with the help of both educationalists and computer enthusiasts.

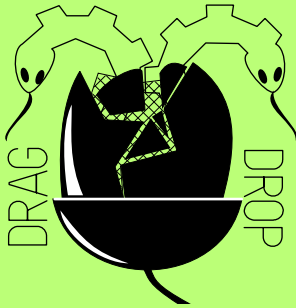
The MICRO ONE is available as a complete self-build kit or part assembled, RISC OS is pre-installed on SD card, with prices starting at just £125 + handling.

The MICRO ONE is being used by all sorts of people, all over the world. Will you join them?

ident  broadcasting & communications

www.ident-online.co.uk/computer

ident 
MICRO  **one** mk4
RISC OS Kit Computer for Raspberry Pi®



Copyright © Drag 'N Drop 2017
Produced on RISC OS computers

This issue has been blessed with contributions from the following people:
Jon Robinson (Wimp programming)
Paul Dunnington (Python Primary School)
Christopher Dewhurst (everything else)

The views expressed in this magazine are not necessarily those of the editor.

Alternative views are always welcome and can be expressed by either writing an article or a short editorial.

All articles and advertisements are published in good faith. No materials in this publication are meant to be offensive or misleading. If you come across something you believe is either of the above please contact the editor using the details below.

Contact Information
Editor: Christopher Dewhurst
Email: editor@dragdrop.co.uk
www.dragdrop.co.uk

EDITORIAL

Welcome to the Spring edition of *Drag 'N Drop*.

By the time you read this you may well have attended the Wakefield show to buy your copy of *Drag 'N Drop* at the special show price plus other goodies on offer such as the back issues stick and perhaps our new publication, *20th Century Fonts*.

By all accounts Wakefield 2017 is going to be an exciting event with rumours of the RISC OS 'big boys' announcing significant software projects. The smaller developers will be out in force: they form the backbone of the RISC OS economy so please come along and show your support and buy their products!

Chris.
Christopher Dewhurst

| | |
|------------------------------------|-----------|
| Editorial | 2 |
| How do I....? | 3 |
| News | 4 |
| Using RDSP | 6 |
| Iconbar animation | 10 |
| Python Primary School | 14 |
| Wimps | 21 |
| Desktop Noughts and Crosses | 27 |
| SWILister | 31 |

How do I...?

...get the BBC Basic prompt?



To get the BBC Basic prompt press F12 and type *BASIC and press Return. You can change the screen mode with MODE n where n is a number e.g. MODE 7 or MODE 0. Type AUTO for automatic line numbering. Press Escape to stop and type *SAVE "myprog"* followed by Return to store *myprog* on hard disc.

To return to the desktop type *QUIT. Programs listed in *Drag 'N Drop* are assumed to work on all machines with RISC OS 5 e.g. Raspberry Pi, unless otherwise stated.

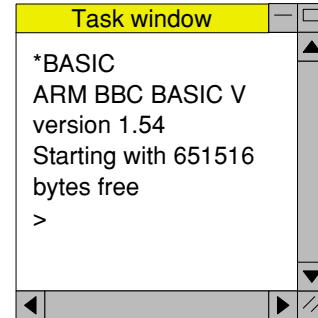
...open a Task window?

Menu click over the Raspberry icon on the right side of the iconbar and select click on Task window. Or press Ctrl + F12.



You may need to reserve more memory for the task in which case adjust-click on the Raspberry icon and under Application tasks click and drag the Next slide bar out to the right.

You can also type programs in a task window, hold down Ctrl and press F12. You can't use the cursor editing facility or change MODE, however.



You can also program and run Basic programs from the desktop. Double-clicking on the filer icon runs it, holding down Shift and double clicking loads it into your text editor.



...select the currently selected directory?

Articles may tell you to set the CSD (currently selected directory). Just click menu over filer window and choose Set directory ^W or you can use the IEasyCSD application presented in *Drag N Drop* 6i1.

...open an Application Directory?

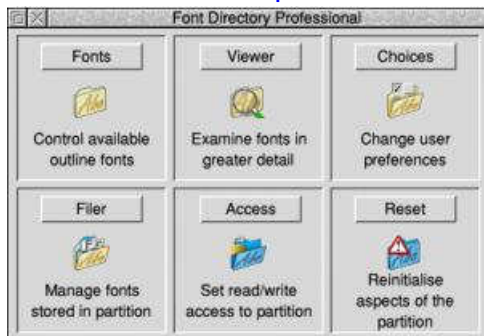
Application directories begin with a ! called 'pling'. Hold down shift and double click select to open the directory.

News and app updates

Recursion '17

The all-format computer fair takes place on 1st July 2017 at KES in Stratford-upon-Avon. RISC OS exhibitors will be there among programmers and users showing off their latest projects using BBC micros, Spectrums, Amigas, Raspberry Pis, robots and more. Doors open 11am and entry is free. More details at

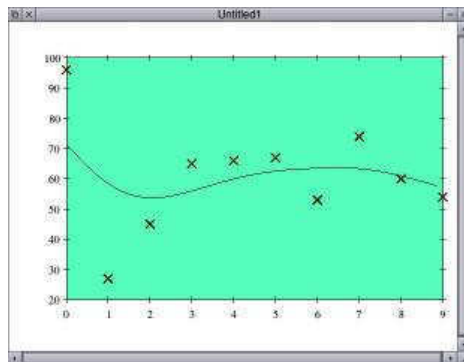
www.recursioncomputerfair.co.uk/



Font Directory Pro

If you work extensively with fonts then Font Directory Pro will help : preview fonts, activate and deactivate groups, scan documents for fonts used and lots more with this package which

costs £22.50 including printed manual and can be purchased from shop.elesar.co.uk and click Accessories > Software



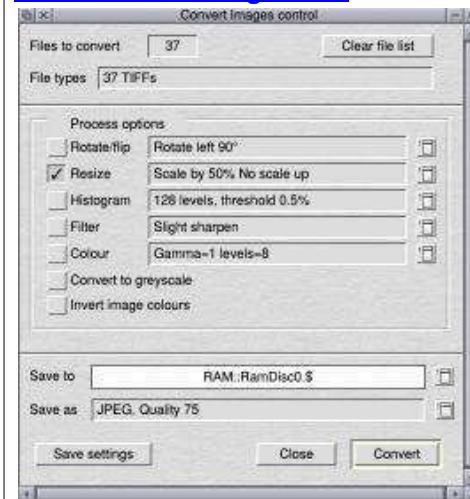
GraphDraw 3.02

The graph plotting program has had a bug fix (where Drawfile export could cause memory corruption). GraphDraw allows input of X and Y data points with a variety of plotting styles plus curve fitting, transform and sorting options. Download for free from www.chrisjohnson.plus.com/software/graphdraw.html

Convimgs

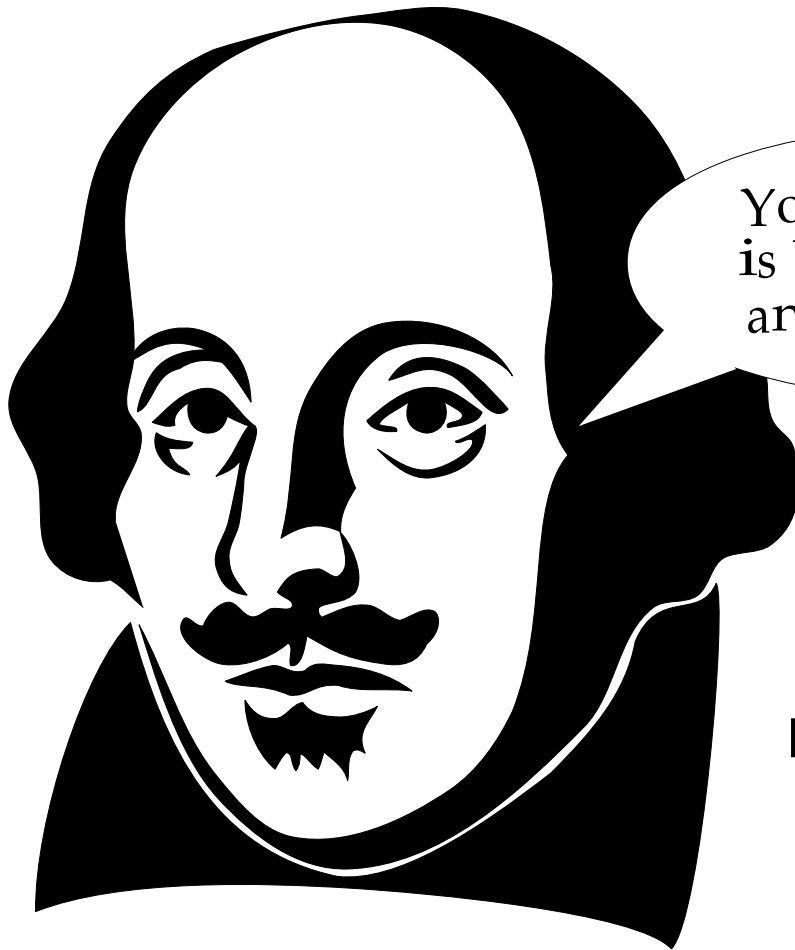
Version 1.11 of the batch conversion of images (PNG, JPEG etc.) with simple

transformations (rotations, flipping, scaling) can be downloaded from www.chrisjohnson.plus.com/software/convimgs.html

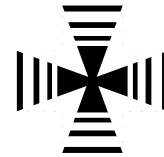


Wispy

And finally . . . we've heard rumours of an exciting new product which will give wireless internet access to your PiTop plus access to those sites which are problematic with Netsurf. We hope to bring full lowdown in the summer edition of *Drag 'N Drop*.



Your date
is better in your Pi
and your porridge.



Recursion Computer Fair
Saturday 1st July 2017
KES, Stratford-upon-Avon

A fun day dedicated to computer science
in industry and leisure.

• Free entry • Doors open 11am

www.recursioncomputerfair.co.uk

Using RDSP

RDSP stands for RISC Digital Sound Processor, a synthesiser module for RISC OS. In this article we'll explore the new possibilities with Basic's SOUND and ENVELOPE statements.

The latest release can be downloaded from www.amcog-games.co.uk/downloads/rdsp-beta2-0_22.zip.

If you follow the installation instructions, RDSP will be available but needs to be invoked with two star commands:

```
*rmload system:modules.audio.sound
chip.rdsp
*restart
```

The first command loads the RDSP module into the relocatable module area and the second tells RISC OS to make SOUND commands come through the synthesizer instead of the normal RISC OS sound handler.

In case you are not clear: press Ctrl+F12 typing *BASIC,

Return key, then the two commands above, pressing Return after each.

We'll look at playing notes first, then sound effects (noises) and finally delve into the ENVELOPE command. RDSP can do lots more than we'll cover here so read the release notes file if you're burning to know more at the end.

To play middle C in a BBC-micro style tone use:

```
SOUND 1,&1F80,172,10
```

The first parameter in the SOUND statement is the channel, as usual, and the third is the pitch, except with RDSP middle C is 172 and not 52 as in conventional RISC OS sounds. This gives a good compass of notes from F four octaves below middle C to C four octaves above as in Fig.1.

The fourth is the duration in 20ths of a second.

The second parameter needs some explanation.

The low byte (&80) is the volume, with 1 being the quietest and &FF loudest.

| Octave | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|--------|----|----|-----|-----|-----|-----|-----|-----|-----|
| C | | 28 | 76 | 124 | 172 | 220 | 268 | 316 | 364 |
| C# | | 32 | 80 | 128 | 176 | 224 | 272 | 320 | |
| D | | 36 | 84 | 132 | 180 | 228 | 276 | 324 | |
| D# | | 40 | 88 | 136 | 184 | 232 | 280 | 328 | |
| E | | 44 | 92 | 140 | 188 | 236 | 284 | 332 | |
| F | 0 | 48 | 96 | 144 | 192 | 240 | 288 | 336 | |
| F# | 4 | 52 | 100 | 148 | 196 | 244 | 292 | 340 | |
| G | 8 | 56 | 104 | 152 | 200 | 248 | 296 | 344 | |
| G# | 12 | 60 | 108 | 156 | 204 | 252 | 300 | 348 | |
| A | 16 | 64 | 112 | 160 | 208 | 256 | 304 | 352 | |
| A# | 20 | 68 | 116 | 164 | 212 | 260 | 308 | 356 | |
| B | 24 | 72 | 120 | 168 | 216 | 264 | 312 | 360 | |

Figure 1.

Iconbar Animation

Have you seen those applications which animate their iconbar icon and wondered how it's done? This article shows how.

Type in Listing 1 which creates



the application's sprite, directory boot and run files.

Then type in Listing 2 which creates the frames of a rotating shape and saves it as a sprite file called *Shapes*, which should be inside the !Spinner application directory.

PROCpoly is a general purpose polygon plotting procedure. It puts a filled polygon on the screen with *sides%* number of sides, *radius%* graphics units big, centred at *xpos%,ypos%* and starting at an initial rotation of *angle* degrees. I've used it to plot a pentagon in a random colour with eight frames.

PROCinitsprites sets up a user sprite area and after each pentagon is plotted OS_SpriteOp 16 is called. (&10 in hex, with &

100 added to signify the operation is to be carried out with the user sprite area). It 'grabs' a the area of the screen occupied by the pentagon and add it to the sprite area.

The sprites are just given names in numerical sequence - "1", "2", etc.

The area of the screen to be grabbed is a square with side length *pix%* (set in PROCinitsprites). The program runs in Mode 21 where there are two graphics units per pixel and the radius of the pentagon is 36 graphics units, or diameter 72 graphics units which is 36 pixels.

Listing 1

```
10REM Make files for Iconbar Sp
inner
20REM (c) Drag 'N Drop 2017
30:
40app$="!Spinner"
50OSCLI"CDIR "+app$
60PROCcreatesprites
70PROCcreatefiles
80END
90:
100DEF PROCcreatesprites
110DIM H% 500:!H%=500:H%!8=&10
```

```
120op$="OS_SpriteOp"
130SYS op$,&109,H%
140READ n$,W%,D%
150SYS op$,&10F,H%,n$,,W%,D%,9:R
EM create sprite
160SYS op$,&11D,H%,n$:REM create
mask
170FOR Y%=D%-1 TO 0 STEP -1
180READ a$
190FOR X%=1 TO W%
200b$=MID$(a$,X%,1):IF b$="." TH
EN
210SYS op$,&12C,H%,n$,X%-1,Y%,0:
REM set pixel in mask
220ELSE
230SYS op$,&12A,H%,n$,X%-1,Y%,EV
AL("&"+b$):REM set pixel in sprite
240ENDIF
250NEXT,
260SYS op$,&10C,H%,app$+"!Sprit
es":REM Save out sprite file
270ENDPROC
280:
290DEF PROCcreatefiles
300REPEAT
310READ file$
320IF file$<>"" THEN PROClines
330UNTIL file$=""
340ENDPROC
350DEF PROClines
360READ type$
370out%=OPENOUT (app$+"."+file$)
380REPEAT
390READ line$
400IF line$<>"" BPUT#out%,line$
410UNTIL line$=""
420CLOSE#out%
430OSCLI"SetType "+app$+"."+file
```


Python Primary School

Last time at Python Primary School we started to convert a Wimp program written in Basic to Python.

The program is **page_043** from Martyn Fox's book *A Beginners Guide to Wimp Programming* and if you have bought your Raspberry Pi's SD card from RISC OS open the book can be found as **Guide** in PDF format in the **Documents.Books.Beginners Guide Wimp** directory with **page_043** in the **Apps.1 Test.Steps** subdirectory.



create_window

This is the largest function which sets up a block defining the window we want to create before asking the Wimp to create it for us.

Basic

```
DEFPROC create_window
REM sets up window data block and creates window
b%!0=168:REM visible area minimum x
b%!4=364:REM visible area minimum y
b%!8=804:REM visible area maximum x
b%!12=872:REM visible area maximum y
b%!16=0:REM scroll x offset relative to work area
origin
b%!20=0:REM scroll y offset relative to work area
origin
b%!24=-1:REM handle to open window behind (-1 means
top, -2 means bottom)
```

```
b%!28=&FF030012:REM window flags
b?!32=7:REM Title foreground and window frame
colour
b?!33=2:REM Title background colour
b?!34=7:REM Work area foreground colour
b?!35=1:REM Work area background colour
b?!36=3:REM Scroll bar outer colour
b?!37=1:REM Scroll bar inner (slider) colour
b?!38=12:REM Title background colour when
highlighted
b?!39=0:REM Reserved - must be 0
b?!40=0:REM Work area minimum x coordinate
b?!44=-700:REM Work area minimum y coordinate
b?!48=1000:REM Work area maximum x coordinate
b?!52=0:REM Work area maximum y coordinate
b?!56=&3D:REM Title bar icon flags
b?!60=&3000::REM Work area flags giving button type
b?!64=1:REM Sprite area control block pointer (1
for Wimp sprites)
b?!68=0:REM Minimum width and height of window
$(b?!72)="Test Window":REM Title data
b?!84=0:REM Number of icons
SYS "Wimp_CreateWindow",b%! TO main%
ENDPROC
```

Python

```
def create_window():
# sets up window data block and creates window
b[0] = 168 # visible area minimum x
b[1] = 364 # visible area minimum y
b[2] = 804 # visible area maximum x
b[3] = 872 # visible area maximum y
b[4] = 0 # scroll x offset relative to
work area origin
b[5] = 0 # scroll y offset relative to
work area origin
b[6] = -1 # handle to open window behind
```

The 20th Century was a golden era in typeface development with classic and eye-catching fonts being designed and used in the world of printed media.

With the advent of digital publishing many of these have since passed into the public domain, collected for this album of over 700 beautiful fonts in RISC OS (Acorn outline) format and also supplied in 'Type 1' for conversion to other platforms e.g. Windows.

Just another collection?

OTHER FONT COLLECTIONS

- Claims 10,000s of fonts but in reality they're just tedious variations on a few fonts.

- Windows only

- Websites with downloadable fonts are transitory or become inaccessible as time goes by

- Contain lots of 'grunge' and 'distressed' fonts.

20TH CENTURY FONTS

- Over 700 unique and beautiful fonts, high quality even for PD standard.

- RISC OS format and Type 1 (PFA) for conversion to other platforms.

- Absolutely no 'grunge' or 'distressed' fonts in this collection!

20th CENTURY FONTS

- Several classic typefaces not available on the internet specially digitised in-house at Drag 'N Drop!
- Printable catalogue (PDF), gazetteer, hints and tips on using fonts on RISC OS.

Price still only £13.00*

+£1 p&p if ordered online





RISC OS Programming

10. More Options on Screen

In the last part of this tutorial we built up a command line using the input path, established when the user dragged the file onto the program's iconbar, and determined the output path when the text file icon was dragged to an open directory window.

This was used to call a command-line utility to extract the text and then modified slightly, to call another utility, which extracted the pictures from the specified PDF file.

In this instalment, we are going to add some extra icons to the program screen, which will make it more useful, see how the Toggle icon works, and start to look at how drop-down menus work.

Adding New Icons

Load the program's Templates file into !WinEd and double click on

Save.




You can see that the window is actually larger than the area that is initially displayed - drag out the size icon at the bottom right of the editing window.

If you click on the Work Area icon (1st icon on the 2nd row) of the !WinEd icon bar, you will see what the width and height of the screen, X1 and Y0, are currently set to. These will probably be the default values, set when you originally created the Save window in Part 4 of the series. That is why the full screen will probably be so large.

What we need to do is to set the full size of the Save screen to something more reasonable. Changing X1 and Y0 to 348 and minus 440 respectively, then click on Update, which reduces the

size of the full window.

Now click on Visible area (2nd icon, 2nd row), change the Height of the window to 248, and click on Move, to shrink the size of the screen that is initially displayed, to just the original, three icons.

Save the Templates file and run !PDFText. You'll see that clicking on the Toggle icon, on the top right-hand corner of the program window, toggles back and forth between the original screen, and one that is a bit longer. 

What we need to do now is to use the extra space to create writable icons for the first and last page to be converted, and also a drop-down menu for the text encoding.

Back in !WinEd and with the Save window open, pull the window out to its full size, and drag six new icons onto the bottom part of the window, from the Icon Picker, to form a row of four and a row of two as in Figure 1.

Desktop OXO

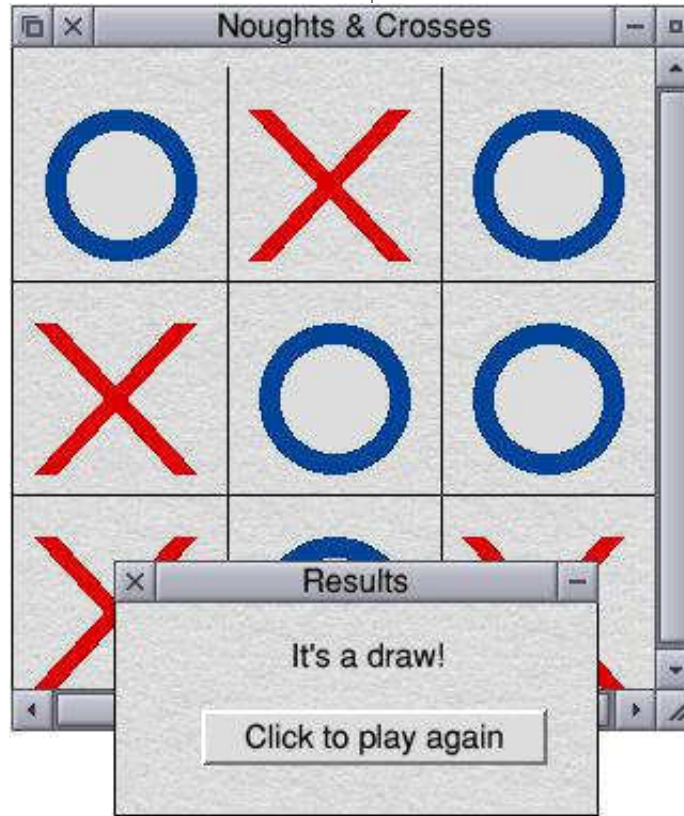
This is a multi-tasking Wimp version of Noughts & Crosses (sometimes called OXO).

It also demonstrates how Wimp programs don't need to occupy reams of code.

The first listing sets up the application directory !OXO and places inside the !Boot, !Run and !Sprites file.

You should then type in the second listing and save it as !OXO.!RunImage.

Finally, double click !OXO to play. Click the mouse in the grid to place your O (the computer plays X). The results window pops up when the game is finished and click on the button for a new game.



!OXO file maker

```
100REM Make files for Desktop OX
0
200REM (c) Drag 'N Drop 2017
30:
40app$="!OXO"
500SCLI"CDIR "+app$
60PROCcreatesprites
70PROCcreatefiles
80END
```

```
90:
100DEF PROCcreatesprites
110DIM H% 500:!H%=500:H%!8=&10
120op$="OS_SpriteOp"
130SYS op$,&109,H%
140READ n$,W%,D%
150SYS op$,&10F,H%,n$,,W%,D%,9:R
EM create sprite
160SYS op$,&11D,H%,n$:REM create
```

```
mask
170FOR Y%=D%-1 TO 0 STEP -1
180READ a$
190FOR X%=1 TO W%
200b$=MID$(a$,X%,1):IF b$="." TH
EN
210SYS op$,&12C,H%,n$,X%-1,Y%,0:
REM set pixel in mask
220ELSE
```



```

230SYS op$,&12A,H%,n$,X%-1,Y%,EV
AL("&"+b$):REM set pixel in sprite
240ENDIF
250NEXT,
260SYS op$,&10C,H%,app$+"!$prite
es":REM Save out sprite file
270ENDPROC
280:
290DEF PROCcreatefiles
300REPEAT
310READ file$
320IF file$<="" THEN PROCLines
330UNTIL file$=""
340ENDPROC
350DEF PROCLines
360READ type$
370out%=OPENOUT (app$+"."+file$)
380REPEAT
390READ line$
400IF line$<="" BPUT#out%,line$
410UNTIL line$=""
420CLOSE#out%
430OSCLI"SetType "+app$+"."+file
$+" "+type$
440ENDPROC
450:
460DATA !oxo,17,17
470DATA .8..5.....5.B.B.
480DATA .8.8.5.....5.B.B.
490DATA .8.8.5.....5..B..
500DATA .8.8.5.....5.B.B.
510DATA .8..5.....5.B.B.
520DATA A5555555555555555555
530DATA .....5.B.B.5..8..
540DATA .....5.B.B.5.8.8.
550DATA .....5..B..5.8.8.
560DATA .....5.B.B.5.8.8.
570DATA .....5.B.B.5..8..
580DATA A5555555555555555555
590DATA .8..5.....5.B.B.
600DATA .8.8.5.....5.B.B.
610DATA .8.8.5.....5..B..

```

```

620DATA .8.8.5.....5.B.B.
630DATA .8..5.....5.B.B.
640:
650DATA !Boot,FEB
660DATA IconSprites <Obey$Dir>.!
Sprites
670DATA ""
680:
690DATA !Run,FEB
700DATA ! |Run file for Desktop
OXO
710DATA | (c) Drag N Drop 2017
720DATA Set OXO$Dir <Obey$Dir>
730DATA WimpSlot -min 32k -max 3
2k
740DATA Run <OXO$Dir>.!RunImage
750DATA ""
760DATA ""

```

!OXO.!RunImage

```

10REM Noughts & Crosses
20REM !RunImage file
30REM (c) Drag 'N Drop 2017
40SYS "Wimp_Initialise",200,&4B
534154,"Noughts & Crosses"
500N ERROR PROCerror
60Q%=200:REM square size
70DIM B% 256,title% 128,itext%
256,U% 10
80DIM board%(8),winning%(7)
90winning%()=012,345,678,036,14
7,258,048,246
100turn%=1
110finish%=FALSE
120
130main%=FNwindow("Noughts & Cro
sses")
140result%=FNwindow("Results")
150i1%=FNicon(result%,116,-76,24
0,52,STRING$(20,CHR$32),&17000119,
"4")

```

```

160i2%=FNicon(result%,80,-152,32
4,52,"Click to play again",&170031
30,"R5,3")
170
180!B%=main%
190SYS "Wimp_GetWindowState",,B%
200SYS "Wimp_OpenWindow",,B%
210exit%=FALSE
220REPEAT
230IF finish%=FALSE AND turn%=-1
PROCcomputer
240SYS "Wimp_Poll",0,B% TO reaso
n%
250CASE reason% OF
260WHEN 1:PROCredraw(!B%)
270WHEN 2:SYS "Wimp_OpenWindow",
,B%
280WHEN 3:SYS "Wimp_CloseWindow"
,,B%:exit%=TRUE
290WHEN 6:PROCclick
300WHEN 17,18:PROCmessage
310ENDCASE
320UNTIL exit%
330SYS "Wimp_CloseDown"
340END
350
360DEF PROCredraw(H%)
370!B%=H%
380SYS "Wimp_RedrawWindow",,B% T
0 more%
390PROCgetorigin(B%,x0%,y0%)
400WHILE more%
410PROCdrawboard(x0%,y0%-620)
420SYS "Wimp_GetRectangle",,B% T
0 more%
430ENDWHILE
440ENDPROC
450
460DEF PROCgetorigin(B%,RETURN x
0%,RETURN y0%)
470x0%=B%!4-B%!20
480y0%=B%!16-B%!24

```

SWILister

SWILister is a short relocatable module which allows you to list the SWIs (SYS commands in Basic) which may be provided by a particular module installed on your machine.

Type in the listing which assembles the code and saves the module as *SWIList*. To install double click *SWIList*.

The module provides one star command *SWIList which must be followed by the module name for example

```
*swilist drawfile
```

and the computer will respond:

```
drawfile swilist
```

```
DrawFile_Render  
DrawFile_BBox  
DrawFile_DeclareFonts
```

The program works by firstly calling OS_Module 18, to look up the module information (given the name typed after *swilist by the user.

The call returns with R1 containing the module number,

R3 the address of the start of the module code. All modules have a standard header and the information about SWIs starts at offset 28 (SWI base number), 32 (SWI entry code), 36 (SWI names table).

The exception to this rule seems to be the Utility Module which provides the OS calls, e.g. OS_WriteC. For some reason the SWI table starts with "OS"+CHR\$0 then the "WriteC"+CHR\$0, "Write\$\$+CHR\$0 etc. The section of code at *.utils* searches through the utility module for this table and deals with it separately.

SWILister

```
10REM SWIList source  
20REM Drag 'N Drop 2017  
30  
40DIM code% 2048  
50FOR pass=4 TO 7 STEP 3  
60%=code%  
70P%=0  
80COPT pass  
90FNzw(4):REM no start, init, o  
r final code or service handler  
100EQUd title ;offset to title  
string of module  
110EQUd help ;offset to help st
```

```
ring of module  
120EQUd comtable ;offset to com  
mand table of module  
130FNzw(5):REM no SWIs  
140EQUd modflags  
150.modflags EQUd 1  
160  
170.title  
180EQUs "SWIList"+CHR$0  
190.help  
200EQUs "SWIList"+CHR$9+"1.00 "+  
MID$(TIME$,5,11)+" Drag 'N Drop"+  
CHR$0  
210ALIGN  
220.listsyntax  
230EQUs "Syntax: *SWILIST <Modul  
e Title>"+CHR$0:ALIGN  
240.listhelp  
250EQUs "Lists the swi commands  
available from the given module."+  
CHR$0:ALIGN  
260.b EQUd 0  
270  
280.comtable  
290EQUs "SWILIST"+CHR$(0) ;comm  
and name  
300ALIGN  
310EQUd listcode ;command code  
320EQUd &10101 ;command flags  
330EQUd listsyntax ;syntax stri  
ng  
340EQUd listhelp ;command help  
string  
350EQUd 0 ;no more commands  
360  
370.listcode  
380STMFd sp!,{r0-r12,lr}  
390MOV r10,r0 ;store pointer to
```

```

command tail
400
410;claim 255 bytes workspace
420MOV r0,#6
430MOV r3,#&FF
440SWI "OS_Module"
450
460MOV r8,r2 ;store pointer to
start of workspace in r8
470MVN r1,#0 ;set pointer into
command tail to -1
480
490.crloop
500ADD r1,r1,#1 ;add 1 to posit
ion in command
510LDRB r2,[r10,r1] ;load byte
520CMP r2,#13 ; 13 (end of comm
and)?
530BNE crloop ;no - loop back
540
550MOV r2,#0 ;indicate end of
560STRB r2,[r10,r1] ;command wi
th zero
570
580;set pointer to module name g
iven in comand
590;and call extract module SWI
600MOV r1,r10
610MOV r0,#18
620SWI "OS_Module"
630
640MOV r11,r3 ;store start of m
odule for later
650CMP r1,#0 ;is module the uti
litymodule?
660BEQ utils ;yes, branch to sp
ecial handler
670LDR r7,[r11,#28] ;load the s
wi base chunk number from module
680CMP r7,#0 ;= 0?
690BEQ exit1 ;yes, exit (no swi
s)

```

```

700MOV r0,r7 ;no, place swi bas
e chunk number in r0
710MOV r1,r0 ;pointer to worksp
ace (for swi string)
720MOV r2,#&FF ;length of works
pace
730;decode swi base no. into swi
string
740SWI "OS_SWINumberToString"
750;r2 now equals the end of the
swi string
760
770LDRB r0,[r8] ;get first byte
of swi string
780CMP r0,#ASC "X" ;is it 'X'?
790BEQ exit1 ;yes, exit (no swi
s)
800LDR r0,[r8] ;no, get first w
ord of swi string
810LDR r1,user ;load r1 with th
e word 'User'
820CMP r0,r1 ;is the swi string
= 'User'
830BEQ exit1 ;yes, exit (no swi
s)
840
850.checkloop1
860SUB r2,r2,#1 ;subtract 1 fro
m the position in swi string
870LDRB r3,[r8,r2] ;load the by
te at this address
880CMP r3,#ASC "_" ;is it equal
to "_"?
890BNE checkloop1 ;no, then con
tinue searching
900
910ADD r2,r2,#1 ;set pointer to
after the '_'
920LDRB r3,[r8,r2] ;get the byt
e at this location
930CMP r3,#ASC "0" ;is it equal
to "0"?

```

```

940BNE getswis ;yes, then say t
hat swis have no names - eg 'OS_0'
950SWI "OS_WriteS" ;print the f
ollowing text
960EQUUS "The swis in the module
"+CHR$0
970MOV r0,r10 ;get pointer to c
ommand string (module name)
980SWI "OS_Write0" ;print comma
nd string
990SWI "OS_WriteS" ;print the f
ollowing text
1000EQUUS " do not have names - th
ey are referenced by number"+CHR$1
0+CHR$13+CHR$0:ALIGN
1010B exit1 ;exit
1020.getswis
1030LDR r9,[r11,#36] ;get offset
to swi decoding table
1040ADD r9,r9,r11 ;get actual me
mory address
1050
1060.printswis
1070MOV r0,r10
1080SWI "OS_Write0" ;print modul
e name
1090SWI "OS_WriteS" ;print follo
wing text
1100EQUUS "swilist"+CHR$13+CHR$10+
CHR$13+CHR$10+CHR$14+CHR$0
1110ALIGN
1120
1130.printswis2
1140MVN r6,#0 ;set swi table poi
nter to -1
1150.swiprintloop
1160ADD r6,r6,#1;add 1 to swi tab
le pointer
1170LDRB r2,[r9,r6] ;get byte at
this location (pointer+table addr
)
1180CMP r2,#0 ;is it equal to 0?

```