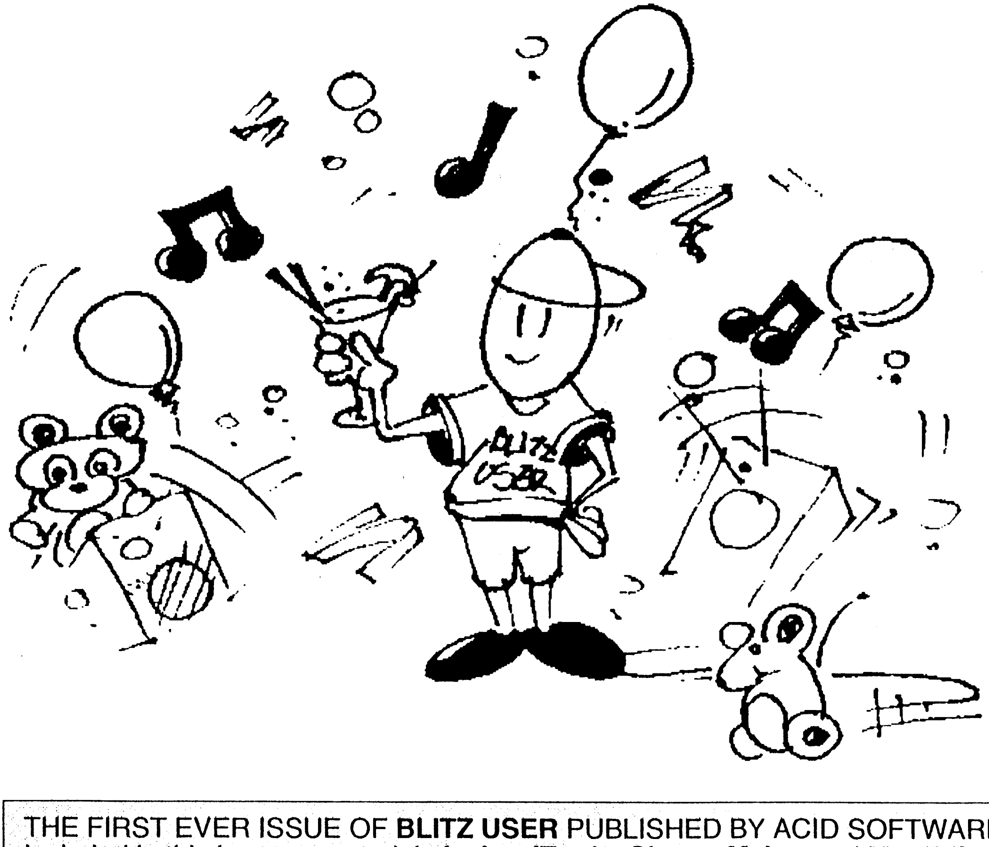
INFORMATION AND SUPPORT FOR BLITZ BASIC 2 USERS WORLD WIDE

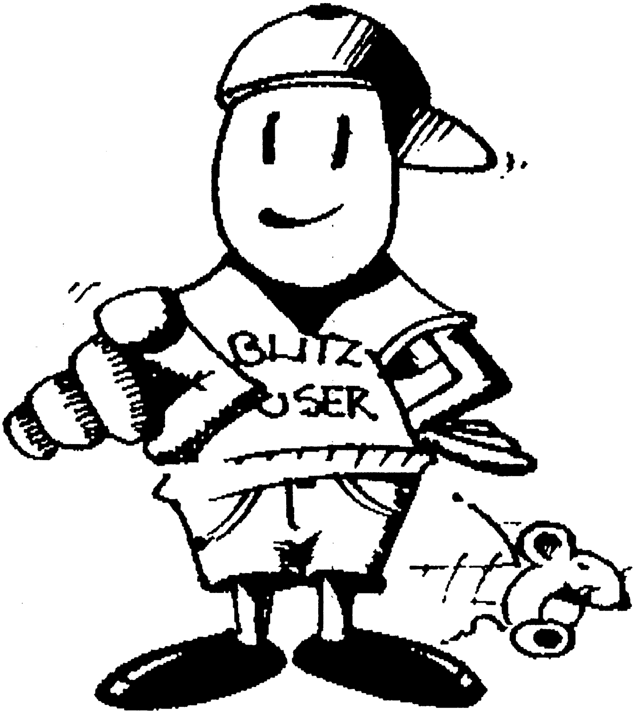
BLITZ USER

ISSUE 1 June 1992



THE FIRST EVER ISSUE OF **BLITZ USER** PUBLISHED BY ACID SOFTWARE. Included in this issue are tutorials for **IntuiTools**, **ShapesMaker** and **MapEditor**. Also two new libraries to drive the **serial port** and use the Amiga's **speech** capabilities. Part 1 of a **hot game** that will have you creating all sorts of weird and wonderful aliens. Some hot machine code **polar co-ordinate routines**, a **phonebook** program and heaps of **hints and tips** for the Blitz 2 programmer.

WANTED!



BLITZMAN NEEDS YOU!

BECOME PART OF THE TEAM. FAME, FORTUNE AND INTERNATIONAL RECOGNITION AWAIT YOU!

CONTACT BLITZ MAN TODAY. PRIZES FOR PD LIBRARY CONTRIBUTIONS TO BE ANNOUNCED SOON.

CONTENTS

No. 1

Editorial 5
Sermon on the mount from the minister of finance himself.

Letters 6
Public opinion at it's finest from Blitz 2 users world wide.

Manual Muckups 8
Get out your pen and turn to page dum de dum of your manual.

Buggy Booboos 10
Reports on Blitz 2's slight imperfections and news on program and library updates.

Using Map Edit 11
Description of the MapEdit program on the ProgramDisk.

Serial LIbrary 12
High speed serial port driver commands for Blitz 2.

SpeakLib 14
Commands for driving the Amiga's speech synthesiser.

PhoneBook 17
Tutorial on using IntuiTools and a simple database type program.

ShapesMaker 19
Spinning logos in DPaint 3.

Project BuzzBar 20
First installment of action packed game.

BLITZ USER

Blitz User is a publication of Acid Software.

Duplication of this magazine is prohibited however all ideas and programs included in this magazine may be used in any size, shape or form.

Acid Software takes no responsibility for the reliability of programs published in this magazine.

Editor

Simon Armstrong

Art Director

Rod Smith

Forward all contributions, advertising and correspondence to:

ACID SOFTWARE
10 StKevins
Arcade
Karangahape Rd
Auckland
New Zealand

fax:64-9-358-1658

ACTION PACKED HIGH QUALITY PD SOFTWARE AVAILABLE NOW

*ACIDSOFTWARE PROUDLY ANNOUNCE THE RELEASE OF BLITZ PD DISKS 1 & 2. FOUR ACTION PACKED GAMES AVAILABLE NOW FROM YOUR LOCAL PD*LIBRARY

BLITZ PD DISK 1

Featuring the ultimate version of the arcade classic. Converted by Blitz author Mark Sibly.

Also includes Zombie Apocalypse from Vision Software. Get even with the crew from Acid software as you blow them away in the most blood and guts game ever seen on the Amiga.

BLITZ PD DISK 2

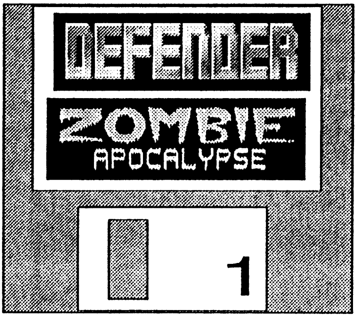
Two great examples of standard arcade formats developed in Blitz 2 by Vision Software.

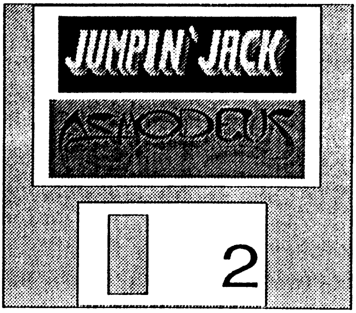
THE BLITZ PD PHILOSOPHY

We at Acid Software aim to set a new standard in Amiga PD Software.

All disks released in the Blitz PD Library will contain Blitz 2 source code to the games and applications included.

Disks 3&4 will be available soon featuring another 2 arcade games plus a disk of applications developed for the Amiga graphics user.





PD Library enquiries welcome. Fax or write to Acid Software for more details:

Acid Software
10 St Kevins Arcade, K Rd.
Auckland, New Zealand
fx: 64-9-358-1658

EDITORIAL

Welcome to the very first Blitz User magazine. As Blitz becomes a more established product we hope to do great things with this publication.

A colour cover would be nice, a cover disk is a definate possibility, heaps of freebies, more user contributions would be most excellent, but hey, you got to start somewhere.

First up, there's a lot of topics to be covered, and a wide range of programming skills to be catered for.

When you send in your **subscription** please let us know what **you** want, how much, why, when, what for...

When you send in your **contributions** remember there's all sorts using Blitz BASIC. I'd like to include a lot more stuff for beginners and maybe some decent educational type programs. My apologies if this issue is a little too advanced, next issue we'll have heaps more for the beginner. Any contributions in this area would be more than appreciated.

In this first issue we've covered using the tools that come on the program disk in some depth. That's the main reason you get the first issue free in the box.

IntuiTools lets you create user interfaces for your applications, the tutorial takes you through using IntuiTools to develop a simple Phone Book type database program.

ShapesMaker is for laying out your animated objects and then converting them to Blitz 2 shapes files. The tutorial delves into animation using Dpaint 3 and then taking that animation and running it in your Blitz program. More sophisticated Anim-Brush support is on its way to simplify this process.

Then there's a couple of type-in libraries. This material is aimed at expanding your Blitz. The speech library uses the standard Amiga message IO to let you drive the synthesiser without the overhead of the DOS speak: device. The serial port library is for Blitz mode type applications where inefficient system drivers just can't deliver. Ever tried getting the serial port working in AmigaBasic? Enough said...

Part 1 of an arcade game is included, in less than 200 lines of code you'll be flying around, firing and chasing nasty aliens around the place. Time to ring up your graphic artist for some better looking aliens though...

Then there's heaps of hints and tips for getting on top of different aspects of programming in Blitz BASIC.

Over the next several issues we've got a lot of material to cover regarding all the new features we've added to Blitz 2.

So, hopefully after viewing the amount of useful information we've been able to include in this first Blitz User you'll want more and subscribe. The more subscriptions we get the more we will be able to do with this magazine.

As far as rumours go, yes there is a 3D library and yes I think its pretty fast. However due to commitments in getting Blitz 2 into production the 3D launch has been delayed for another couple of months. However by then you can expect a real slick package that delivers more than some ever dared to promise. Hold onto your hats!

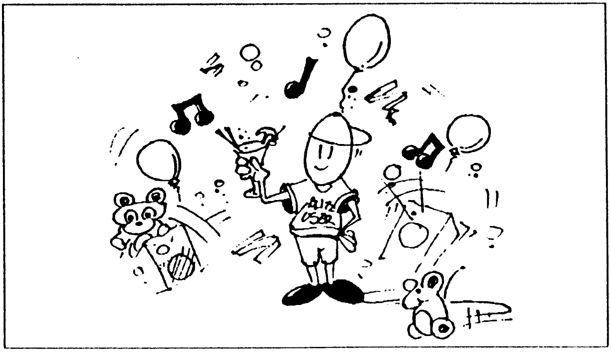
A full Arexx command set is also near release so your Blitz 2 applications can have full Arexx handles attached.

As I write this, the first 1000 Blitz BASICs wait quietly beside the door. Waiting to be exported around the world. Not only is there 1.5 kgs of manuals in each box but 2 years of hard work from Mark. It's up to me to see that they are properly marketed and yes, I need **your** help.

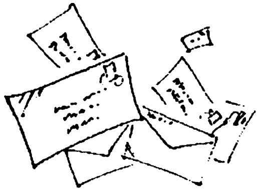
So, subscribe to Blitz User, let your friends and the magazines know what you think of Blitz 2, but most importantly get creative and enjoy your programming.

Cheers,

Simon.



LETTERS



The following section is for publishing any correspondence received at Acid Software in New Zealand. To write to us please use the following address:

Acid Software
10 St Kevins Arcade
183 Karangahape Road
Auckland
New Zealand

If you have a fax our number is:

64-9-358-1658

The leading 64 is the international area code for New Zealand.

Seeing as this issue has been compiled before the official launch of Blitz 2 we thought we'd write to ourselves to get this section started...

Dear BlitzMan,

After receiving my new Blitz 2 I just couldn't beleive what seriously heavy software you guys have come up with. Blitz 2 has got to be the heaviest software available for the Amiga! I had to get my sister to help me get it up the stairs and Dad has reinforced my computer desk so it could hold the weight.

So, how do you get something so heavy to go so fast? I asked my physics teacher and he said it is not weight but mass that is important when delving into such physical relationships.

I got this nerdy friend who's into copying software and he couldn't even pick the box up!

Don't worry he's in hospital now cos he dropped it on his foot and is now suffering massive brain damage.

Signed,

REAL HEAVY!

Yo Dudes,

Totally awesome! Mind expanding. Wow wow wowwow. But hey, what do I do? I can't make up my mind whether to write a kill the mutant drug addicts from the intergalactic scum bucket or develop this program for my Mum's dating agency.

Signed,

Seriously Undecided.

Dear Acid Software,

I don't usually write to companies with such unreputable names such as yours but after using Blitz 2 I felt compelled to write a letter of congratulations.

I have an Amiga 3000 with an 040 card, 16 megabytes of ram and 4 gigabytes of disk storage.

Now I've got Blitz 2 I can make do with a plain old Amiga 500 so I'm selling off the hardware and using the money to finance a certain chemistry experiment you boys might be very interested in.

Signed,

Bubbling Over.

Dear Sir,

May I congratulate you on the great centre-fold in last months issue of your fine magazine. She was so hot I was unable to program my home computer for more than 10 minutes without stopping to admire her amazing bod.

Enclosed is a subscription to your great magazine for the next 15 years (don't forget the plain brown envelope) .

Please keep it up,

Signed

Wrong Mag.

Dear Blitzman,

I'm having a big problem with life. Living in the 90's just doesn't seem to be that cool anymore, facists are everywhere, there's no work and everybody is totally negative.

I wonder if other Blitz users out there are interested in forming some sort of international be-cool and totally-together club.

I'd like to here from others in different places about what can be done with the world and some of the uglier people in it.

Signed

Lessthan Hopeful

Dear Sir,

It was only a freak accident that I came across your package Blitz BASIC 2. I hope you have some mega promotional material planned to make Blitz known to all Amiga users around the place.

I am sure some of my ideas could well realise the international acclaim Blitz deserves.

Signed,

Gizza Marketing Jobby

Dear Sir,

It was with great displeasure that the rumoured 3D capabilities of Blitz 2 have not yet been delivered. I have nearly finished a large model of an interglacatic space station which I plan to turn into a serious piece of arcade software with your planned 3D library. So, where is it and how fast is it now?

Signed,

Hurryup Andeliver

Hopefully next issue we can have some serious letters from some actual Blitz users, so any problems, complaints, thoughts or silliness, put pen to paper and send them in.

Hey, we might even have some totally cool Blitz 2 T-shirts to give away in the next issue.

DISCLAIMER: The views, attitudes and ideas expressed in this section are not necesarily in agreement with the editors of this magazine.

MANUAL MUCKUPS

The following are a list of relatively serious errors in the Blitz 2 manuals. This list will be added to as they are reported, spelling mistakes and typos shall be accepted as part of life and left but you should turn to the pages following and make the necesary changes...

REFERENCE MANUAL

PAGE 19-3

Slice flag table, woza, where did the tabs go? Max Bitplanes should be in their own box to the right 6, 4, 6, 4 & 6.

OK not that it's disasterous but a little disconcerting.

PAGE 24-1

What extended form of the SCREEN command? Whats Viewmode? Infact not much discussion here at all really...

X,Y are offsets from the preferences screen position settings so leave 0,0 unless you want your screen to open in some weird place on the monitor.

Width, Height & Depth: yup obvious. An example? how about 320,200,5.

ViewMode: alrighty just like BPLCon0, hmm just use the following (add them together for legal combinations), more info is available in the graphics library includes and the like.

```
lores=0
genlock=2
interlace=4
*superhire=$20
*pfba=$40
extra halfbrite=$80
genlock audio=$100
dualplayfield=$400
ham=$800
hires=$8000
```

Note: * means ECS Amigas only and because of the extended OS 2.0 structures not yet supported in Blitz 2. If you want them please ask us nicely.

PAGE A4-6

Wo here's a serious one that stuffed me up for a day. Here I was thinking it would be nice for a comprehensive list of hardware registers in the back and I stuff up BLTCON0 and BLTCON1!

In most Amigas you will find them located at $40 and $42 NOT at $100 and $102.

USER GUIDE

PAGE 1-5

The following screen should appear? Oops, not if you just run Ted as you don't get the OkeeDokee requester but just the TED text editor screen... (getting a little picky here perhaps?).

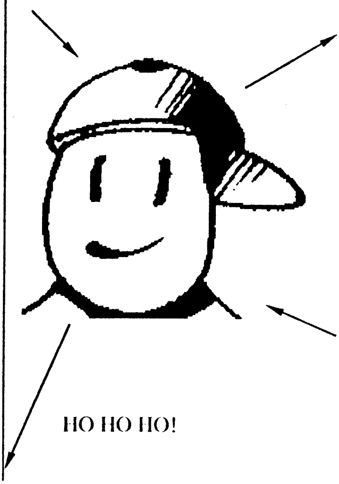
PAGE 1-46

OK this might have happened a few times, seems that my Bold keyword generator for Pagestream missed 9 lines down this page and instead of GOTO being bold its got an @$ before it, weird huh?

PAGE 1-65

Now this one is slightly embarrasing, Mark tells me that the header node does not look anything like the first box in the diagram, however due to lack of space we leave it up to the user to correct the diagram (if this is too much try colouring it in).

Well maybe there is enough room for a header node...



HO HO HO!

UNDOCUMENTED COMMANDS

Statement: **Block** (add to Chapter 21 of the Reference manual)

Syntax: **Block** Shape#,X,Y

Modes: Amiga/Blitz

Description:

Block is an extremely fast version of the **Blit** command with some restrictions.

Block should only be used with shapes that are 16,32,48,64... pixels wide and that are being blitted to an x position of 0,16,32,48,64...

Block is intended for use with map type displays.

The height and y destination of the shape are not limited by the Block command.

Statement: **LoadFont** (add to chapter 25 should be with WindowFont command)

Syntax: LoadFont IntuiFont#,Fontname.font$,Y Size

Modes: Amiga

Description:

LoadFont is used to load a font from the fonts: directory. Unlike BlitzFonts any size IntuiFont can be used. ~~Well, seeing as this newsletter came out the same time we launched Blitz 2 it would be rather silly to think we would be reporting bugs in this issue.~~ See also: Window Font

Function: **VPos** (add to chapter 5)

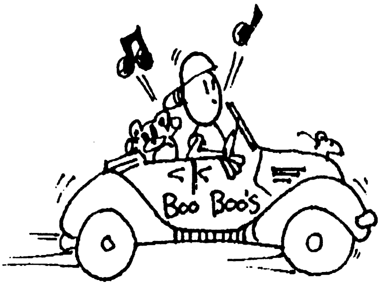
Syntax: **VPos**

Modes: Amiga/Blitz

Description:

VPos returns the video's beam vertical position. Useful in both highspeed animation where screen update may need to be synced to a certain video beam position (not just the top of frame as with **VWait**) and for a fast random nember generator in non frame-synced applications.

BUGGY BOOBOOS



Being the first issue of Blitz User it would be a bit silly if we were already reporting bugs in this section. However their has been a few problems with evaluation inside square brackets...

SQUARING UP

If an evaluation such as:

```
jim\son[5-bob\numkids]="BILLY"
```

gives spurious results try:

```
  num=5-bob\numkids
jim\son[num]="BILLY"
```

We are currently doing more tests on such evaluations.

Rounding off..

A few mathmatical issues will now be addressed here. This discussion is based around the complex issue of rounding numbers not about some 'bug' in the Blitz 2 quick maths library or the Amiga's Fast Floating Point library (which we use for all floating point calculations).

The only reason this discussion is in the Buggy Booboos section is because this section needs fleshing out a bit...

Anyway try the following...

```
DEFTYPE .f
While Joyb(0)=0
  NPrint a
  a+.1
Wend
```

So where do all the weird numbers come from? Try it in another BASIC. Now try changing the deftype back to quick (thats .q). Same problem again.

This is because computers are not decimal (base 10) but binary (base 2) and have big trouble accurately defining a number such as 0.1, this is of course the same as 1/10.

To see the binary equivalent of .1 try the following:

```
a.q=.1
NPrint Bin$(Peek.l(&a))
MouseWait
```

This prints in binary the contents of the memory location where the variable a is kept.

As you can see .1 is a very complex number in base 2.

Whats more when using quicks (fixed point maths) multiply increases the rounding error dramatically:

```
a.q=.1
Print 500*a
MouseWait
```

You can now see why rounding errors are such a big deal in computer science.

The above example remains accurate with floating point because of the nature of floating point (amount of error is absorbed by the change in exponent) but rounding still increases with sufficient initial error.

Confused?

Lets just say that if you want to do an accounting type package in Blitz 2 don't use fixed point (quick) types.

Fast floating point is also probably not a good idea as when you hit the $260,000 anything less than a dollar will disapear.

So what should you do?

Wait for a DoublePrecision maths library to be released?

Don't be silly, just use long integers and treat a dollar as 100 cents. Ha, not only do you get to account for numbers up to 20 million but you don't loose a cent of accuracy either.

Don't forget to physically insert the decimal point two digits from the right when you print figures.

Also divide by whole numbers rather than multiplying by decimal points whenever possible.

As if somebody is going to write huge financial programs in Blitz 2 anyway...

Found a bug in Blitz?

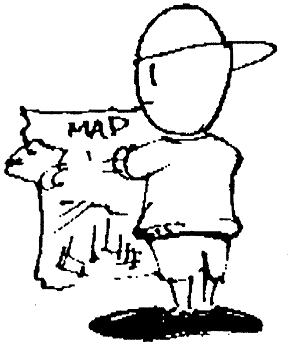
Having problems with some of the documentation?

Does something just not seem to work the way you think it should?

If so, let us know.

If you are reporting a bug, please remove the offending piece of code from your program, add a small 'shell' to it so it is functional and send it in.

USING MAP EDIT



To create a map file using the MapEdit program, you will require at least 1 source file - a shapes file created using the 'SaveShapes' command.

This file should contain all the 'blocks' you wish to use for the map.

The 'Shapesmaker' program may also be used to create a shapes file.

You may also want an IFF ILBM file to pick up the palette information.

When run, MapEdit will bring up a requester asking for block and map sizes. MapEdit supports block sizes of 4,8,16,32 or 64 pixels.

Clicking on the 'FROM FILE' gadget will allow you to specify a map file from which the above information may be gleaned.

Once in MapEdit, you will need to load in a shapes file. The 'BLOCKS' menu allows you to load either shapes, or a colour palette.

The shapes picked up from the shapes file will appear as a set of gadgets at the top left of the screen. The left and right arrows may be used to move around all available shapes.

Clicking on an actual block will allow you to place the block anywhere on-screen using the left mouse button.

Doing so will add that block to the map. Clicking on-screen using the right mouse button will erase a block from the map. If the map is larger than the screen, the cursor arrow keys may be used to scroll around the map. The 'GROUP' menu allows you to create or delete a 'group' of blocks.

To create a group, first paste down the approprate blocks onto the map, and then select 'CREATE' from the 'GROUP' menu. You may then select any on-screen blocks using the left mouse button to make up a group of blocks.

Once done, close the miniature CREATE GROUPS window at the top of the screen. A group may be pasted down or deleted in the same way as individual blocks.

Miniatures of any defined groups appear at the top right of the screen, and may be selected at any time by clicking on the appropriate group with the left mouse button.

When completed, save your map by selecting 'SAVE' from the 'PROJECT' menu.

Maps are saved row by row, using 1 byte per map block. A map block of 0 indicates an 'empty' map block, while map blocks from 1 onwards indicate a shape to use for that map block.

Listing 1 below is an example of reading a map file into your own program. This map has a block size of 16 by 16 pixels, and a map size of 20 by 12 blocks. It dimensions a two dimensional array and reads the block information into the array byte by byte.

Listing 2 draws up the map onto the current bitmap.

The next issue of Blitz User will contain details on adding a map based background to a game and some hints and tips on designing the graphics for map type blocks.

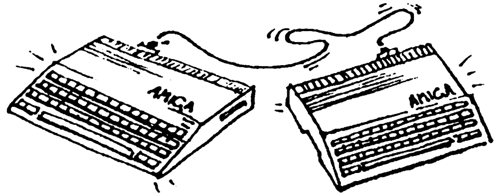
```
Dim m.b(19,11)              ;dim an array for the map
LoadShapes 1,"Map.shapes"   ;load  blocks to be used for the map
If ReadFile(0,"Map1")       ;try to open map file
   FileInput 0              ;get input from file
   For y=0 To 11            ;columns...
   For x=0 To 19            ;rows...
   m(x,y)=Asc(Inkey$)       ;pick up byte from map file
   Next x,y                 ;
   CloseFile 0              ;close map file
   DefaultInput             ;restore input channel
Else
                            ;Error reading map file!
EndIf
```

Listing 1

```
For y=0 To 11
For x=0 To 19
   If m(x,y) Then Block m(x,y),x*16,y*16
Next x,y
```

Listing 2

SERIAL LIBRARY



The following is a listing for a serial port library. Before proceeding you should be familiar with the Blitz 2 Libraries chapter in the User Guide and have already tried the simple Library on page 1-53.

As an overview, the seriallib is a high speed serial port driver intended for use in Blitz mode. Adding resource allocation to the **_openserial** routine is needed for it to properly behave in Amiga mode applications.

Extra commands that do block reads and writes will be added in the next issue as well as information for syncing null-modem type games.

Driving the Commodore multi serial port card will also be addressed in the near future.

Circular buffers are used for both read and write buffers. A cirular buffer simply 'wraps round' at the end and can be thought of as a never ending loop. Two pointers are required, beginning and end, when they are equal the buffer is empty. When either pointer is incremented it is logically 'anded' with 255 (511 in the write buffer) which has the effect of wrapping it round to the beginning.

To access the buffer PC with displacement addressing is used, 'rend(PC,d0)' generates the address in memory at rend + the displacement in register d0.

Both buffers are LIFO- last in first out, to add a byte, write it to the location pointed to by the end pointer and increment the pointer, to read, read from the beginning pointer and then increment it. Simple and efficient.

_setbaud: This routine calculates the baudrate for the

```
#serlib=49

!libheader {#serlib,0,0,finit,0}

!astatement
!args {#long}
!libs
!subs {_setbaud,0,0}
!name {"SetBaud","rate"}

!astatement
!args
!libs
!subs {_openserial,0,0}
!name {"OpenSerial",""}

!astatement
!args
!libs
!subs {_closeserial,0,0}
!name {"CloseSerial",""}

!afunction {#long}
!args
!libs
!subs {_readser,0,0}
!name {"ReadSer"," returns next byte -1 if empty "}

!astatement
!args {#word}
!libs
!subs {_writeser,0,0}
!name {"WriteSer","byte"}

finit:!nullsub{_libfinit,0,0}

!libfin

#serdat=$dff030
#serdatr=$dff018
#serper=$dff032

#intena=$dff09a
#intenar=$dff01c
#intreq=$dff09c
#intreqr=$dff01e

_setbaud:
 MOVE.l#3546895,d1:EXG d0,d1:ALibJsr $cb01
 SUBQ #1,d0:BCLR #15,d0:MOVE.w d0,serper:RTS

_openserial:
 TST.w openflag:BNE okok
 MOVE.l$74,OldInt5+2:MOVE.l$64,OldInt1+2
 MOVE.l#ReadInt,$74:MOVE.l#WriteInt,$64
 CLR.lrbig:CLR.lwbig:MOVE.b#$3f,$bfd000
 MOVE.w intenar,OldInt:MOVE.w#$c801,intena
 MOVE.w #-1,openflag:okok:RTS

openflag:Dc.w 0

_libfinit:

_closeserial:
```

hardware register serper. The **Alib.Jsr $cb01** calls the internal Blitz long divide command. The constant #3546895 is the PAL constant as found in the Amiga Hardware Manual (page 251). NTSC machines should use #3579545.

_openserial: This routine saves the old interupt vectors and pokes in the new ones. Writing #$c801 to intena (interrupt enable) enables both the read and write serial interupts.

_closeserial: If the serial lib has been opened (openflag set) the old interupts are restored and the interupt register is restored to its previous state.

_readser: Checks the read buffer, if there's a byte waiting it returns it else returns a -1.

ReadInt: When ever the serial port is full, ReadInt reads the byte from the serdatr hardware register and adds it to the read buffer.

_writeser: Adds the byte sent by the user to the write buffer, if the write buffer is empty, generates an interupt so the byte is immediately outputted to the serial port.

WriteInt: When ever the serial port empties it's write register this interupt is generated, if there is another byte in the buffer then that gets written to the serdat hardware register.

Note: this library is set up as 8 bit no parity, the write buffer uses words not bytes so the 1 data bit can be generated outside the interupt.

As you can see in Listing 1, I've used more than 1 machine code instruction per line. I'm use to it, as you can see, you can fit heaps more on the screen at once.

When compiling don't forget to set Compiler Options:

NO ERROR CHECKING
& MAKE SMALLEST

Good luck,

Simon.

```
 TST.w openflag:BNE poo:RTS
 poo:MOVE.l OldInt5+2,$74:MOVE.lOldInt1+2,$64
 MOVE.w #$7fff,intena:MOVE.w OldInt,d0
 MOVE.b#-1,$bfd000
 OR.w #$c000,d0:MOVE.w d0,intena:CLR.w openflag:RTS

OldInt:Dc.w 0
OldInt1:JMP -1
OldInt5:JMP -1

_readser:
 MOVE.wrbig(pc),d0:CMP.wrend(pc),d0:BNE NotEmpt
 MOVEQ #-1,d0:RTS
 NotEmpt:MOVE.b readbuff(pc,d0.w),d0:ADD.b#1,rbig+1:RTS

ReadInt:
 BTST.b#3,intreqr:BEQ OldInt5
 MOVEM.ld0/a0,-(a7):MOVE.w rend(pc),d0:LEA readbuff(pc,d0.w),a0
 MOVE.b serdatr+1,(a0):ADD.b#1,rend+1
 MOVEM.l(a7)+,d0/a0:MOVE.w#$0800,intreq:RTE

rbig:Dc.w0
rend:Dc.w0
readbuff:Dcb.b 256,0

_writeser:
 AND.w#$ff,d0:OR.w#$100,d0
 MOVE.wwrend(pc),d1:LEA writebuff(pc,d1.w),a0
 MOVE.wd0,(a0):ADD.b#2,wrend+1
 CMP.wwbig(pc),d1:BNE buffed:MOVE.w#$8001,intreq
 buffed:RTS

WriteInt:
 BTST.b#0,intreqr+1:BEQ OldInt1
 MOVE.w d0,-(a7):MOVE.wwbig(pc),d0:CMP.wwrend(pc),d0:BEQ
NoMore
 wem:BTST.b#5,serdatr:BEQ wem
 MOVE.w writebuff(pc,d0.w),serdat:ADD.b#2,wbig+1
 NoMore:MOVE.w (a7)+,d0:MOVE.w #$0001,intreq:RTE

wbig:Dc.w0
wrend:Dc.w0
writebuff:Dcb.b 256,0

numbuffed:Dc.w0
```

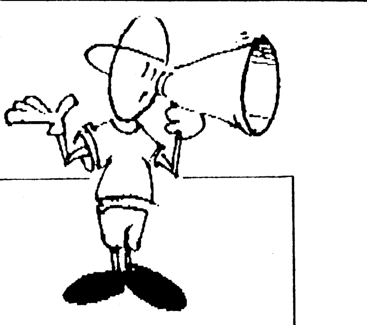
LISTING 1

```
OpenSerial:SetBaud 2400
FindScreen 0
w$="SERIAL PORT TEST ATDT number to auto dial"
SizeLimits 32,32,-1,-1:Window ←(no new line here)
0,0,17,640,128,$100F,w$,1,2:WindowInput 0

Repeat
  a$=Inkey$
  If a$<>"" Then WriteSer Asc(a$)
  x=WCursX
  b=ReadSer
  If b=8 AND x>7                 ;backspace
    WLocate x-8,WCursY:Print " "
    WLocate x-8,WCursY
  EndIf
  If b=13 Then NPrint ""         ;return
  If b>13 Then Print Chr$(b)     ;legal?
Until Event=$200                 ;close window
```

LISTING 2

SPEAKLIB



The speaklib opens the narrator.device and the translator library, adding commands to Blitz that allow you to drive the Amiga's speech synthesiser.

As with the serial port library you should familiarise yourself with the Libraries section of the User Guide before attempting to get speaklib up and running.

The Translator Library

The translator.library is a standard Amiga library containing the one command Translate (EnglStr, EngLen, (APTR)&PhonBuffer, BUFLEN). **_libinit**: calls the Exec routine OpenLibrary with a pointer to the name of the library (in lower case) and a version number. Exec returns the Library base which we store in the loacation TransBase.

To call the translate function in the **_speak** and **_translate** routines we put TransBase in register a6, string length in d0, bufferlength in d1, string in a0, and buffer location in a1. Then we use JSR -30(a6) to call the translate function in the translator.library.

The Narrator Device

The narrator.device is a standard Amiga exec level device (not to be confused with a DOS device).

The Exec call _OpenDevice is used to set up the device for our use. To communicate with a device we need to set up an IORequest structure. The way I have done it in this program is to set out all the values needed (IOReq..IOEnd) and then poke the necesary values such as the Task, an allocated signal bit etc. into this memory.

More information about devices and IORequests, as well as better ways to set them

```
#speaklib=50

!libheader {#speaklib,init,0,finit,0}

!astatement
!args {#string}
!libs
!subs {_speak,0,0}
!name {"Speak","string"}

!astatement
!args {#word,#word,#word,#word,#word,#word}
!libs
!subs {_setvoice,0,0}
!name (no new line)
{"SetVoice","rate(150),pitch(110),expression(1),sex,volume(64),freq(22200)"}

!afunction {#string}
!args {#string}
!libs
!subs {_translate,0,0}
!name {"Translate$","returns phoneme translation of string"}

!astatement
!args {#string}
!libs
!subs {_phospeak,0,0}
!name {"PhoneticSpeak","phonetic string"}

init:!nullsub{libinit,0,0}
finit:!nullsub{libfinit,0,0}

!libfin

_setvoice:  CMP     #40,d0          ;clip all parameters
            BPL     n1
            MOVE    #40,d0
n1:         CMP     #400,d0
            BMI     n2
            MOVE    #400,d0
n2:         CMP     #65,d1
            BPL     n3
            MOVE    #65,d1
n3:         CMP     #320,d1
            BMI     n4
            MOVE    #320,d1
n4:         AND     #1,d2
            AND     #1,d3
            CMP     #64,d4
            BPL     n5
            MOVE    #64,d4
n5:         CMP     #5000,d5
            BPL     n6
            MOVE    #5000,d5
n6:         CMP     #28000,d5
            BMI     n7
            MOVE    #28000,d5
n7:         MOVEM   d0-d3,rate
            MOVEM   d4-d5,vol
            RTS

_speak:     MOVE.l  d0,a0           ;d0=string
            MOVE.l  -(a2),d0        ;-(a2)=string length
test:       MOVE.l  a6,-(a7)        ;save a6
            LEA     spkbuff(pc),a1
            MOVE.l  #1024,d1
            MOVE.l  TransBase(pc),a6
```

Don't forget to make resident libmacs.res and blitzoffs.res up will be tackled in the next issue of Blitz User.

If you want more info now, the Amiga ROM KERNEL manual covers devices pretty thorougly if not a little confusingly. Better still, ask a C programmer.

_setvoice: The 5 parameters rate (words perminute), pitch, expression, sex, volume and frequency can be changed with this command. One thing I haven't been able to do is get speak to sing, can anyone help?

_speak: The main command. The string parameter passed is first sent to the translator library to be converted into phonemes, the result is then passed to the narrator.device in a standard IO packet.

_translate: Converts the string parameter to phonemes and returns the result without passing it to the narrator.device.

_phospeak: Sends the string directly to the narrator device (parameter should just be made up of phonemes).

_libinit: Opens both the narrator.device and translator library.

_libfinit: Closes both the narrator.device and translator library.

If libinit fails to open either device or library it uses the TRAP #0 system to report the error. If runtime errors is enabled Blitz will intercept any Trap#0's and print the message pointed to by D0.

I have had a few problems with this library in that sometimes it does not close properly and the machine must be rebooted before the narrator.device can be successfully re-opened. Hopefully someone can help out.

Also set tab to 12 for setting out the listing as printed.

```
            JSR         -30(a6)             ;translate text to buffer
            MOVEQ       #0,d0
            LEA         spkbuff(pc),a0
srchnull:   ADDQ        #1,d0
            TST.b       (a0)+
            BNE         srchnull
            MOVE.w      #3,_SCommand        ;send message to narrator
            MOVE.l      #0,_Offset
            MOVE.l      #spkbuff,_SData
            MOVE.l      d0,_SLen
            MOVE.l      4,a6
            LEA         IOReq(pc),a1
            JSR         _SendIO(a6)
            MOVE.l      (a7)+,a6            ;restore a6
            RTS

_translate: MOVE.l      d0,a3               ;take string
            MOVE.l      d0,a0
            MOVE.l      -(a2),d0
            MOVE.l      a6,-(a7)
            LEA         spkbuff(pc),a1
            MOVE.l      #1024,d1
            MOVE.l      TransBase(pc),a6
            JSR         -30(a6)
            MOVE.l      (a7)+,a6
            MOVEQ       #0,d0
            LEA         spkbuff(pc),a0
srchnull2:  ADDQ        #1,d0
            MOVE.b      (a0)+,(a3)+         ;add result to string buffer
            BNE         srchnull2
            RTS

_phospeak:  MOVE.l      a6,-(a7)            ;send string straight
            MOVE.l      d0,a3               ;to narrator
            MOVE.w      #3,_SCommand
            MOVE.l      #0,_Offset
            MOVE.l      d0,_SData
            MOVE.l      -(a2),_SLen
            MOVE.l      4,a6
            LEA         IOReq(pc),a1
            JSR         _SendIO(a6)
            MOVE.l      (a7)+,a6
            RTS

libinit:    LEA         Translator(pc),a1   ;open Translator library
            MOVE.l      4,a6
            JSR         _OpenLibrary(a6)
            MOVE.l      d0,TransBase
            BEQ         failed

            LEA         Narrator(pc),a0     ;open Narrator device
            MOVEQ       #0,d0
            LEA         IOReq(pc),a1
            MOVEQ       #0,d1
            JSR         _OpenDevice(a6)
            TST.l       d0
            BNE         failed
            MOVEQ       #-1,d0              ;allocate a signal
            JSR         _AllocSignal(a6)
            MOVE.b      d0,_SigBit
            TST.l       d0
            BMI         failed
            SUB.l       a1,a1               ;a1=0
            JSR         _FindTask(a6)
            MOVE.l      d0,_Task
            MOVE.l      #audchan,ch_masks
            MOVE.w      #4,nm_masks
            MOVE.b      #0,mouths
            RTS
```

```
failed:      MOVE.l    #report,d0
             TRAP      #0
report:      Dc.b      "Failed To Open Speech Library",0

libfinit:    MOVE.l    a6,-(a7)
             MOVE.l    4,a6
             LEA       IOReq(pc),a1
             JSR       _AbortIO(a6)
             LEA       IOReq(pc),a1         ;close Narrator device
             JSR       _CloseDevice(a6)
             MOVEQ     #0,d0                ;free signal bit
             MOVE.b    _SigBit,d0
             JSR       _FreeSignal(a6)
             MOVE.l    TransBase,a1         ;close translator library
             JSR       _CloseLibrary(a6)
             MOVE.l    (a7)+,a6
             RTS

Translator:  Dc.b      "translator.library",0
             Even
Narrator:    Dc.b      "narrator.device",0
             Even
TransBase:   Dc.l      0


IOReq:       Dc.l      0,0                  ;Message
             Dc.b      5,0
             Dc.l      0
             Dc.l      StdMsgPort
MsgLen:      Dc.w      IOEnd-IOReq

_SDev:       Dc.l      0,0                  ;device,unit
_SCommand:             Dc.w                 0
             Dc.b      0,0                  ;flags,error
             Dc.l      0                    ;actual
_SLen:       Dc.l      0                    ;length
_SData:      Dc.l      0
_Offset:     Dc.l      0

rate:        Dc.w      0
pitch:       Dc.w      0
mode:        Dc.w      0
sex:         Dc.w      0
ch_masks:    Dc.l      0
nm_masks:    Dc.w      0
vol:         Dc.w      0
sampfreq:    Dc.w      0
mouths:      Dc.b      0
chanmask:    Dc.b      0
numchan:     Dc.b      0
pad:         Dc.b      0
IOEnd:

audchan:     Dc.b      3,5,10,12

StdMsgPort:  Dc.l      0,0                  ;Node
             Dc.b      4,0                  ;type,pri
             Dc.l      0                    ;sname
_Flags:      Dc.b      0
_SigBit:     Dc.b      0
_Task:       Dc.l      0
_List:       Dc.l      _List+4,0,_List
             Dc.b      0,0                  ;type,pad

             Even

spkbuff:     Dcb.b     4096,0
```

Listing 1

```
;
;BlitzSay program to test speaklib
;

FindScreen 0

PropGadget 0,83,15,320,1,196,9
SetHProp 0,1,0,.0625
PropGadget 0,83,26,320,2,196,9
SetHProp 0,2,0,.0625
PropGadget 0,83,37,320,3,196,9
SetHProp 0,3,0,.0625
PropGadget 0,83,49,320,4,196,9
SetHProp 0,4,0,.0625
Borders On:BorderPens 1,2:Borders 4,2
StringGadget 0,8,82,0,7,256,328
GadgetJam 0:GadgetPens 1,0
TextGadget 0,9,65,1,5,"EXPRESSION"
TextGadget 0,105,65,1,6,"FEMALE"
TextGadget 0,227,65,0,8,"SAY IT AGAIN!"

SizeLimits 32,32,-1,-1
Window 0,80,64,352,99,$100E,"BLITZ SPEAK",1,2,0
WLocate 32,5:WJam 0:WColour 1,0
Print "RATE:"
WLocate 26,16
Print "PITCH:"
WLocate 17,27
Print "VOLUME:"
WLocate -2,38
Print "FREQUENCY:"

DEFTYPE .w
r=150:p=110:v=64:f=22200

Repeat              ;main loop
 SetVoice r,p,c,s,v,f
 WJam 1
 WLocate 280,5:Print r,"  "
 WLocate 280,16:Print p,"  "
 WLocate 280,27:Print v,"  "
 WLocate 280,38:Print f,"  "

 ActivateString 0,7
 VWait 5
 ev.l=WaitEvent           ;wait for a window event
 If ev=$40
  Select GadgetHit
  Case 1:r=40+360*HPropPot(0,1)    ;rate
  Case 2:p=65+255*HPropPot(0,2)    ;pitch
  Case 3:v=64*HPropPot(0,3)        ;volume
  Case 4:f=5000+23000*HPropPot(0,4)  ;frequency
  Case 5:c=1-c          ;toggle expression
  Case 6:s=1-s          ;toggle female
  Case 7              ;string gadget
   a$=StringText$(0,7)
   Speak a$
   ResetString 0,7
  Case 8
   Speak a$

  End Select
 EndIf

 If ev=$200 Then End        ;close Window gadget

Forever
```

Listing 2

PHONE BOOK

The following is a description of IntuiTools the user interface editor that comes with Blitz 2 and then a tutorial to set up a simple phonebook data base system.

IntuiTools is a utility for designing and creating user interfaces for Blitz 2 applications quickly and easily. Full control of Screen, Window and Gadget parameters let you interactively build and test front ends (user interfaces) as complex as you wish.

To begin with we shall run through each of the menu items and their use, then cover some of IntuiTools quirks and lastly, a tutorial which will lead you through the steps needed to create an interface with IntuiTools.

The Menus

PROJECT MENU

LOAD lets you edit previously saved interfaces.
SAVE stores the interface to disk so you can redesign/ edit it later.
CREATE SOURCE generates Blitz 2 code to be inserted into your program.
QUIT is something you contemplate when unhappy with your job.

SCREEN MENU

PALETTE lets you adjust the screen colours used by your program
OPTIONS enables you to change the resolution and other screen attributes

WINDOW MENU

ADD TEXT lets you place text anywhere in your window.
TEST lets you play with your interface just as if your program was running, click in the top left window to exit test mode.
OPTIONS lets you adjust window options as detailed in the Blitz 2 reference manual.

GADGETS MENU

TEXT creates a button with writing on it
STRING creates a string gadget which enables the user to entere text via the keyboard.
PROP is a slider gadget that enables the user to adjust variables accurately.
SHAPE creates a button with a graphic on it, the graphic should be an IFF brush (as created by DPaint etc) and be in the same resolution as your screen.

Positioning Gadgets

After you have entered the relevant information for a gadget a second window appears. You may select whether the gadget's position is is relative to the top or bottom and left or right of the window. This is only important when the window is sizable.

If the gadget is relative to the bottom right, sizing the window will mean the gadget is repositioned to the new bottom right of the window.

After clicking on MAKE you must position the gadget in your window with the left mouse button.

Gadget Numbering

Each gadget you add to your window needs a specific number known as it's ID.

No two gadgets in a window can share the same ID.

If your application will have more than one window it is a good idea to use different ID numbers for each window's gadgets also. However EventWindow can be used to differentiate gadgets if each window has a gadget with the same ID.

Sometimes leaving some spare ID numbers between groups of gadgets is a good idea if you are to update the interface later. If you have 4 prop gadgets and then a set of 6 text gadgets start the text IDs at 10 so that if you ever need to add another prop gadget to the set you can number them 5,6..

Confused?

Don't worry read this section again after you've done the tutorial.

Paste Mode

Once you have added some gadgets to your window you may wish to copy, delete, move or duplicate them. By clicking on the gadget once you will 'pick it up'.

To delete the gadget just pick it up and then click the right button, the gadget will disappear.

To move the gadget pick it up, move the mouse to the desired position and click the left button to paste it. Then click the right button to exit paste mode.

To duplicate a gadget pick it up, position and click the left button as many times as you want and then hit the right button to exit paste mode.

Tutorial

Before trying this tutorial it's a good idea to re-familiarise yourself with screens, windows and gadgets by reading these chapters in the Blitz reference manual.

First, a hires screen...

Select the Screen/Options menu. Click in the gadget to the right of TITLE:. This is a string gadget. Delete the contents by selecting Amiga X which is a keyboard shortcut to empty the contents of a string gadget. Now type in the name of your application and hit return. PHONE BOOK will do for this application.

```
;
; phone book program by simon
;

FindScreen 0

;the following should be imported from the file ram:t as created in the tutorial

Borders On:BorderPens 1,2:Borders 4,2
StringGadget 0,72,12,0,1,40,239
StringGadget 0,72,27,0,2,40,239
StringGadget 0,72,43,0,3,40,239
StringGadget 0,72,59,0,4,40,239
GadgetJam 0:GadgetPens 1,0
TextGadget 0,8,75,0,10,"NEW ENTRY"
TextGadget 0,97,75,0,11,"| <"
TextGadget 0,129,75,0,12,"<<"
TextGadget 0,161,75,0,13,">>"
TextGadget 0,193,75,0,14,">| "
TextGadget 0,226,75,0,15,"DIAL"
TextGadget 0,270,75,0,16,"LABEL"
```

Click on the gadget to the right of mode until it reads HiresNonInterlace. This is the typical resolution for an Amiga application. A depth of 2 gives your screen 4 colours. Height should be set to 256 for PAL/European markets and 200 for NTSC/American markets. Because this application will be using the workbench screen we need to set hires with a depth of 2.

When you select DONE the IntuiTools will reconfigure the display for the new settings.

Next, a draggable window and some gadgets...

Now drag the window so it takes up about 1/4 of the whole screen.

Note: to use the sizing gadget in IntuiTools you need to click on the very bottom right of the window.

First we need to add some text to our window that will explain what our string gadgets represent. Select the Window/Add Text menu and type Name then click on MAKE. Position the text in the top left. Add 'Address' and 'Phone' under name.

Now select the Gadget/String menu. Click on Make then Make. Position the box to the right of the 'Name:' text and holding the left mouse button drag the box across to just before the right edge of the window. We now have a string gadget that the user can type in their name, it's ID should be 1.

Using paste mode we need to add 3 more gadgets the same as our name string gadget. Pick it up with the left mousebutton, paste it straight back down again with the left button then paste 3 more times below. Then click on the right mouse button to exit paste mode.

Four gadgets and 3 titles? Thats so we can have two lines for address, using paste mode move the text blocks so they align properly with the gadgets. Click once to pick them up, click again to put them down in their new position and then click the right button to exit paste mode.

Right now for some control gadgets for our phone book. Using the Gadgets/Text menu option type '<<' in the top field. Then click on MAKE.Type 10 into the gadget ID. Starting our control gadgets at 10 separates them from the string gadgets. Now click on MAKE and add our rewind button to the bottom left.

Use the same procedure as above to add a forward gadget, a 'PRINT' gadget and a 'DIAL' gadget. Now select SAVE to save our interface as "phonebook.int", and then select CREATE SOURCE to generate some Blitz 2 source code use a filename such as "ram:temp". The Listing is the data base program that uses the interface we have just designed.

```
SizeLimits 32.32,-1,-1
Window 0,0.24,331,91,$100E,"MY PHONE BOOK",1,2,0
WLocate 2,19:WJam 0:WColour 1,0
Print "Address"
WLocate 19,50
Print "Phone"
WLocate 27,3
Print "Name"

; and nowwe start typing...

#num=4     ;4 strings for each person

NEWTYPE .person
 info$[#num]
End NEWTYPE

Dim List people.person(200)

USEPATH people()

If ReadFile (0,"phonebook.data")  ;read in names etc from sequential file
     FileInput 0
     While NOT Eof(0)
          If AddItem(people())
               For i=0 To #num-1:\info[i]=Edit$(128):Next
          EndIf
     Wend
EndIf

ResetList people()

If NOT NextItem(people()) Then AddItem people() ;if empty add blank record

refresh:
     ref=0
     For i=0 To #num-1:SetString 0,i+1,\info[i]:Redraw 0,i+1:Next
     ActivateString 0,1:VWait 5
     Repeat
          ev.l=WaitEvent
          ;
          If ev=$200 ;close window event
               Gosub update
               If WriteFile (0,"phonebook.data")  ;save data to file
                    FileOutput 0
                    ResetList people()
                    While NextItem(people())
                    For i=0 To #num-1:NPrint \info[i]:Next
                    Wend
                    CloseFile 0
               EndIf
               End
          EndIf
          ;
          If ev=64
               If GadgetHit=#num Then ActivateString 0,1
               If GadgetHit<#num Then ActivateString 0,GadgetHit+1
               Select GadgetHit
                    Case 10:Gosub update:If AddItem(people()) Then ref=1
                    Case 11:Gosub update:If FirstItem(people()) Then ref=1
                    Case 12:Gosub update:If PrevItem(people()) Then ref=1
                    Case 13:Gosub update:If NextItem(people()) Then ref=1
                    Case 14:Gosub update:If LastItem(people()) Then ref=1
               End Select
          EndIf
     Until ref=1
     Goto refresh

 update:
     For i=0 To #num-1:\info[i]=StringText$(0,i+1):Next
     Return

                                                            ;PhoneBook Listing
```

ShapesMaker

ShapesMaker is a utitility for importing groups of brushes such as animations into Blitz 2 quickly and easily.

The following is a tutorial that quickly covers generating an animbrush in Dpaint 3, laying it out on a page for ShapesMaker to use and then converting it to a shapes file that Blitz 2 can load using ShapesMaker.

For this tutorial you will need to own a version of Dpaint3 or higher.

A Spinning Animbrush

Selecting lo-res 8 colour mode when you first run DPaint, design a logo or shape that will look good spinning, don't make it to big.

Once you are happy with it save it as a brush just in case we run into difficulties later on.

With your logo positioned in the middle of the screen cut it out with the right button, this will make a brush but also erase the image from the screen.

Select the Anim/Frames/Set# menu item and type 16 into the string gadget. This creates 16 frames for your animation to be 'sequenced' through.

Now select the Anim/Move menu. Set the Y rotation to 360 thats the second box along, second down. When you click on preview DPaint should indicate the area and perspective the shape is going to rotate through.

If all looks OK select draw. Dpaint should then draw each frame of the spin on the 16 consecutive frames and leave you back with the numbers 1/16 at the top left. This is which frame you are currently viewing. Now select the Anim/Anim Brush/PickUp menu.

Just as if you had selected cut out brush cut out the shape. When you release the mouse button DPaint stores each frame in an 'anim-brush'. (Blitz 2 will soon support this format).

Laying Out the AnimBrush

After doing a pickup anim brush you can clear the screen. Because there is more than 1 frame DPaint will ask which frame, select current. Now move to the top left and click the button. The first frame of the brush will be put on the screen. moving to the right click again, the second frame will be pasted to the screen. When you get to the right of the screen move down and start at the left again. **Note:** always make sure that a vertical and horizontal strip separates each frame.

If all is correct you should have the 16 frames of the anim on the one screen. This is the format ShapesMaker requires. Select Picture/Save and save the frame.

If you did not have enough room, save the anim brush, set up a bigger screen, hires for example, reload the animbrush and paiste the 16 frames onto the new screen.

Running ShapesMaker.

After saving the screen with the 16 frames layed out with gaps between all shapes quit out of DPaint and run ShapesMaker. Select the Load menu and select the picture file that you created in DPaint. Because it is a spinning logo we will want the handle to always be in the centre so select this option from the menu.

Now select Convert. Enter a filename, such as logo.shapes and hit return. If the converter does not generate 16 frames you must not have had a suitable gap between shapes.

Thats It! Try the listing below to get your shape spinning on an Amiga screen.

```
Screen 0,3
ScreensBitMap 0,0

Queue 0,1

LoadShapes 0,"h:test.shapes"

Repeat
    For i=0 To 15
        VWait 4
        UnQueue 0
        QBlit 0,i,160,100
        If Joyb(0)>0 Then End
    Next
Forever
```

PROJECT BUZZBAR

This is the first installment of what will hopefully be a commercial quality game. It's a rotating shoot'em up using some very fast polar coordinate type algorithms to get some smooth, curving alien flight paths.

The playfield is 1024x1024 (4 screens wide, 4 high). It's both keyboard and mouse control. A few ideas might work out quite well, like firing gives you negative inertia and aliens rotate and thrust at different speeds giving them each some pretty unique characteristics.

Polar Coordinates

For those that haven't done much maths, there are two main types of coordinate systems, cartesian and polar. Cartesian is simply the x axis, y axis system. The polar system uses rotation and length to work out position i.e turn NorthEast and walk thirty feet.

The main mathematics involved when converting cartesian to polar are:

The angle formed when you move x steps across and y steps up is given byArcTangent(y/x)

The distance moved when you move x steps across and y steps up is the Sqrt(x*x+y*y)

Listings 1 & 2 combine to give you the above two functions. I developed them for my 3D library. If these techniques have not been used before I would appreciate credit if they end up in commercial software.

Hmmm, now, how do they work.... The first listing generates the hex look up tables needed, (if you want speed use tables). The second are inline machine code functions that calculate (using the tables) the functions angle and length as detailed above.

The properties of a right angle triangle is the basis of most trigonometry.

For the Mathematicians:

If a=adjacent, h=hypotenuse and o=opposite we know that

(i) Sin(angle)=o/h
(ii) Cos(angle)=a/h
(iii)Tan(angle)=o/a

From (iii) we get
(iv)angle=Atan(o/a)
and with (ii) we can get
h=a/Cos(angle) which with(iv)
h=a/Cos(o/a)
which with a little machine code magic gives us our two functions with 1 divide as the maximum.

Enough! On with the game..

```
;
; table generator
;
DEFTYPE .f a

If WriteFile(0,"arc.inc")
      FileOutput 0
      For i=0 To 511
            a=Int(ATan(i/512)*8191/ATan(1))
            Print Mki$(a)
      Next
      CloseFile 0
EndIf

If WriteFile(0,"len.inc")
      FileOutput 0
      For i=0 To 511
            a=Int(Cos(ATán(i/512))*65535)
            Print Mki$(a)
      Next
      CloseFile 0
EndIf
```

Listing I

```
Function.w distance{x1.w,y1.w,x2.w,y2.w}
    UNLK a4                                        ;unlink (no recursion)
    SUB d2,d0:BPL xpos:NEG d0:xpos                 ;d0=width
    SUB d3,d1:BPL ypos:NEG d1:ypos                 ;d1=height
    CMP d0,d1:BEQ kludge                            ;kludge if equal
    BMI ygtx:EXG d0,d1:ygtx                        ;d0=greater side
    TST d1:BNE yne:RTS:yne                         ;if short side 0 len=other
    SWAP d1:CLR d1:DIVU d0,d1:LSR#7,d1             ;look up=short/long
    ADD d1,d1:SWAP d0
    DIVU lvals(pc,d1.w),d0:RTS
    kludge:MULU #27146,d0:SWAP d0:ADDd1,d0:RTS ; multiply bysqrt(2)
    lvals:IncBin "len.inc"
End Function

Function.w angle{x1.w,y1.w}
    UNLK a4                                           ;unlink
    MOVEQ#0,d2                                        ;d2=quadrant
    TST d1:BPL hpos:MOVEQ#16,d2:NEG d1:hpos           ;y positive
    TST d0:BPL wpos:EOR#8,d2:NEG d0:wpos              ;x positive
    CMP d1,d0:BMI notsteep:BNE neq
    MOVF#$2000,d1:BRA flow:neq
    EOR #4,d2:EXG d1,d0:notsteep
    TST d1:BNE noflow:MOVEQ#0,d1:BRA flow:noflow
    EXT.ld0:SWAP d0:DIVU d1,d0:LSR#6,d0:AND#1022,d0
    MOVE arc(pc,d0),d1
    flow:MOVE.l oct(pc,d2),d0:EOR d0,d1:SWAP d0:ADD d1,d0:RTS
    oct:Dc.w 0,0,$4000,-1,0,-1,$c000,0
    Dc.w $8000,-1,$4000,0,$8000,0,$C000,-1
    arc:IncBin "arc.inc"
End Function
```

Listing 2

Line by Line

The first line inserts the program qfuncs.bb from the previous page, so make sure you use the same name other wise Blitz won't be able to include our qfuncs.bb. Don't forget that qfuncs also includes the tables generated by listing 1 so keep everything in the same directory.

Double Buffering

To acheive a smooth flicker freegame we need to use a doublebuffered display. This means we have two displays, after drawing up a display we show it and draw on the other one. This means we are never drawing to the same display that is being shown on the monitor. So, we need two bitmaps and two queues (see QBlit explanation in the reference guide).

Stars

The .star type holds coordinates and colour information for the stars. Because we are using a double buffered display the drawstars routine needs to know their position from two frames ago so it can 'unplot' them (same reason we need two queues) so px,py hold last frame's screen coordinates and ppx,ppy holds the frame before.

Playfield

As mentioned the playfield is 1024x1024. Instead of scrolling the playfield around using a large bitmap we calculate alien and star positions relative to the player who stays in the middle of the screen.
However we use a 32 pixel nonvisible 'frame' around the bitmap so that aliens can enter and exit the display smoothly. To do this we use a slice that displays 320x200 pixels and

```
INCLUDE qfuncs.bb

NEWTYPE .star
        x.w:y:c:px:py:ppx:ppy
End NEWTYPE

NEWTYPE .ship
        x.w:y:rot:thrust:rspeed:id:xv:yv:px:py
End NEWTYPE

Dim List nme.ship(50) ;enemy
Dim List bul.ship(50)  ;bullets
Dim s.star(50)           ;stars
Dim qsin.q(1023)        ;look up tables
Dim qcos.q(1023)

me.ship\x=0,0,0,0,0,0,0

BitMap 0,320+64,200+64,3 ;doublebuffered display
BitMap 1,320+64,200+64,3
BitMap 2,640,36,3        ;hires 8 color score+scanner
Queue 0,50
Queue 1,50

BLITZ                    ;setup blitz display
Mouse On
BlitzKeys On:BitMapInput
Slice 0,46,320,200,$fff8,3,8,8,320+64,320+64
Slice 1,248,640,36,$fff9,3,8,8,640,640
Use Slice 0

Gosub setupships         ;setup everything
Gosub setupstars
Gosub setupsincos
Gosub setupnme
Gosub setupdisplay

While NOT RawStatus($45) ;main loop (exit on esc key)
        VWait
        Show db,32,32
        db=1-db
        Use BitMap db
        UnQueue db
        Gosub drawstars
        Gosub drawnme
        Gosub drawbullets
        Gosub moveship
Wend

End

.setupdisplay
        Use Slice 1:Show 2
        Use BitMap 2:BitMapOutput 2:Cls
        Locate 0,0:Colour 3:Print "Buzz Bar Blitz"
        Box 320-32,0,320+33,34,3
        Use Slice 0
        Return

.moveship
        If RawStatus($31) Then me\rot-1024
        If RawStatus($32) Then me\rot+1024
        me\rot+(MouseXSpeed*200)
        If RawStatus($38) OR Joyb(0)&2;thrust
                me\xv+qsin((me\rot LSR 6)&1023) ASL 4
                me\yv-qcos((me\rot LSR 6)&1023) ASL 4
        EndIf
        If RawStatus($39) OR Joyb(0)&1;fire
                If AddItem(bul()) AND rl=0
```

bitmaps that are 320+64, 200+64 large. The **Show** db,32,32 command in the main loop positions the larger BitMap correctly.

Aliens

Each alien has an x,y poition as well as a rotation, thrust, rotspeed (turning speed), id, as well as x&y speed, and scanner position.
When we move the aliens we calculate the angle you are to them, they then turn at their rotspeed towards you and thrust in the direction they are pointing. We then calculate their position relative to the player and if inside the Bitmap, **QBlit** them and plot them on the scanner.

Numeric Handling

All x,y coordinates are .w (word) types. A word is 16 bits and so we use the lower 6 as a fixed point fraction. This is quicker than using the quick type. Wo, lets not get confused, a simpler way of looking at it is to say the playfield is actually 65536 wide & each pixel on the screen is 64 units wide. To calculate pixel positions we just **ASR** 6 (quick divide by 64). Don't forget to turn overflow checking off in runtime errors as we are using overflow to our advantage.

Keyboard Handling

After entering blitz mode **BlitzKeys On** turns on the keyboard handler. Use the RawKey table at the end of the reference manual to change keys used, **(ZX,.)** are the keys used at present.

Reference Guides

Setupshapes is covered under the Rotate command 14-8 of the reference manual. The

```
            bul()\x=me\x,me\y,me\rot
            bul()\xv=qsin((me\rot LSR 6)&1023) ASL 9+me\xv
            bul()\yv=-qcos((me\rot LSR 6)&1023) ASL 9+me\yv
            rl=8
            me\xv-qsin((me\rot LSR 6)&1023) ASL 5
            me\yv+qcos((me\rot LSR 6)&1023) ASL 5
        EndIf
    Else
        rl=0
    EndIf
    me\xv-me\xv ASR 5 ;drag
    me\yv-me\yv ASR 5
    me\x+me\xv
    me\y+me\yv
    QBlit db,(me\rot LSR 12)&15,160+32,100+32
    If rl>0 Then rl-1
    Return

.drawnme
    ResetList nme()
    USEPATH nme()
    While NextItem(nme())
        ang.w=32768-angle{me\x-\x,me\y-\y}-\rot
        If ang<0
            \rot-\rspeed;rotate towards me
        Else
            \rot+\rspeed
        EndIf
        \xv+qsin((\rot LSR 6)&1023) * \thrust;thrust
        \yv-qcos((\rot LSR 6)&1023) * \thrust
        \xv-\xv ASR 6;drag
        \yv-\yv ASR 6
        \x+\xv        ;speed
        \y+\yv
        Use BitMap 2:Plot \px LSR 4+309,\py LSR 5+13,0:Use BitMap db
        \px=((\x-me\x) ASR 6)+160+32
        \py=((\y-me\y) ASR 6)+100+32
        If RectsHit(\px,\py,1,1,12,12,320+32,200+32)
            QBlit db,(\rot LSR 12)&15,\px,\py
        EndIf
        Use BitMap 2:Plot \px LSR 4+309,\py LSR 5+13,4:Use BitMap db
    Wend
    Return

.drawbullets
    ResetList bul()
    USEPATH bul()
    While NextItem(bul())
        \x+\xv        ;speed
        \y+\yv
        px=((\x-me\x) ASR 6)+160+32
        py=((\y-me\y) ASR 6)+100+32
        If RectsHit(px,py,1,1,12,12,320+32,200+32)
            QBlit db,(\rot LSR 12)&15,px,py
        Else
            KillItem bul()
        EndIf
    Wend
    Return

.drawstars
    For i=0 To 50
        USEPATH s(i)
        Plot \ppx,\ppy,0
        \ppx=\px:\ppy=\py
        \px=(\x-(me\x LSR 6))&511
        \py=(\y-(me\y LSR 6))&255
        Plot \px,\py,\c
    Next
```

quick sin/cos look up arrays are covered in the starfield tutorial of the user guide.

Things To Try

Increase the number in the for..next loop of setupnme for more aliens. How many before the frame rate drops?

Design your aliens in DPaint (lo-res 8 colour) and load them in instead of the ugly ones generated in setupships. Don't forget to be in Amiga mode before you do a LoadShapes.

Design the palette so your fire can be a 1 colour shape, so when you blit it (without exess) things should be faster.
Hint: make colour 4=white

Add a sound effect for the fire.

Next Issue

Coming up in the next installment we'll add collisions, put a null-modem link in and drop those horrible stars for an 8 colour scrolling backdrop using Mark's map editor. Hmmm, 1 Meg only from now on perhaps.

```
        Return

.setupships:
        Cls:Circlef 8,6,6,6,4 ;draw ship
        Line 8,0,4,10,2:Line 8,0,12,10,2
        Plot 10,20,2:Boxf 0,0,1,10,2
        Boxf 14,0,15,10,2:Boxf 5,8,11,10,3
        GetaShape 0,0,0,16,12:MidHandle 0
        For i=1 To 15
                CopyShape 0,i;generate rotations
                Rotate i,i/16
                MidHandle i
        Next
        Return

.setupstars:
        For i=0 To 50
                s(i)\x=Rnd(1024),Rnd(1024),Rnd(6)+1
        Next
        Return

.setupnme:
        USEPATH nme()
        For i=1 To 5
                AddItem nme()
                \x=Rnd(65535),Rnd(65535),Rnd(65535)
                \thrust=Rnd(12)+3,Rnd(256)+256
        Next
        Return

.setupsincos
        For i=0 To 1023
                r.f=i*Pi/512
                qsin(i)=Sin(r)
                qcos(i)=Cos(r)
        Next
        Return
```

* once everything is working take runtime-errors off for a decent speed increase.

* change setup stars to 20 instead of 50 for more speed.

* change ASL 4 on lines 5&6 of moveship to ASL 5 for more thrust.

* more speed increases next issue!

Simon.