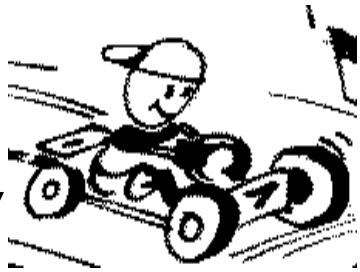
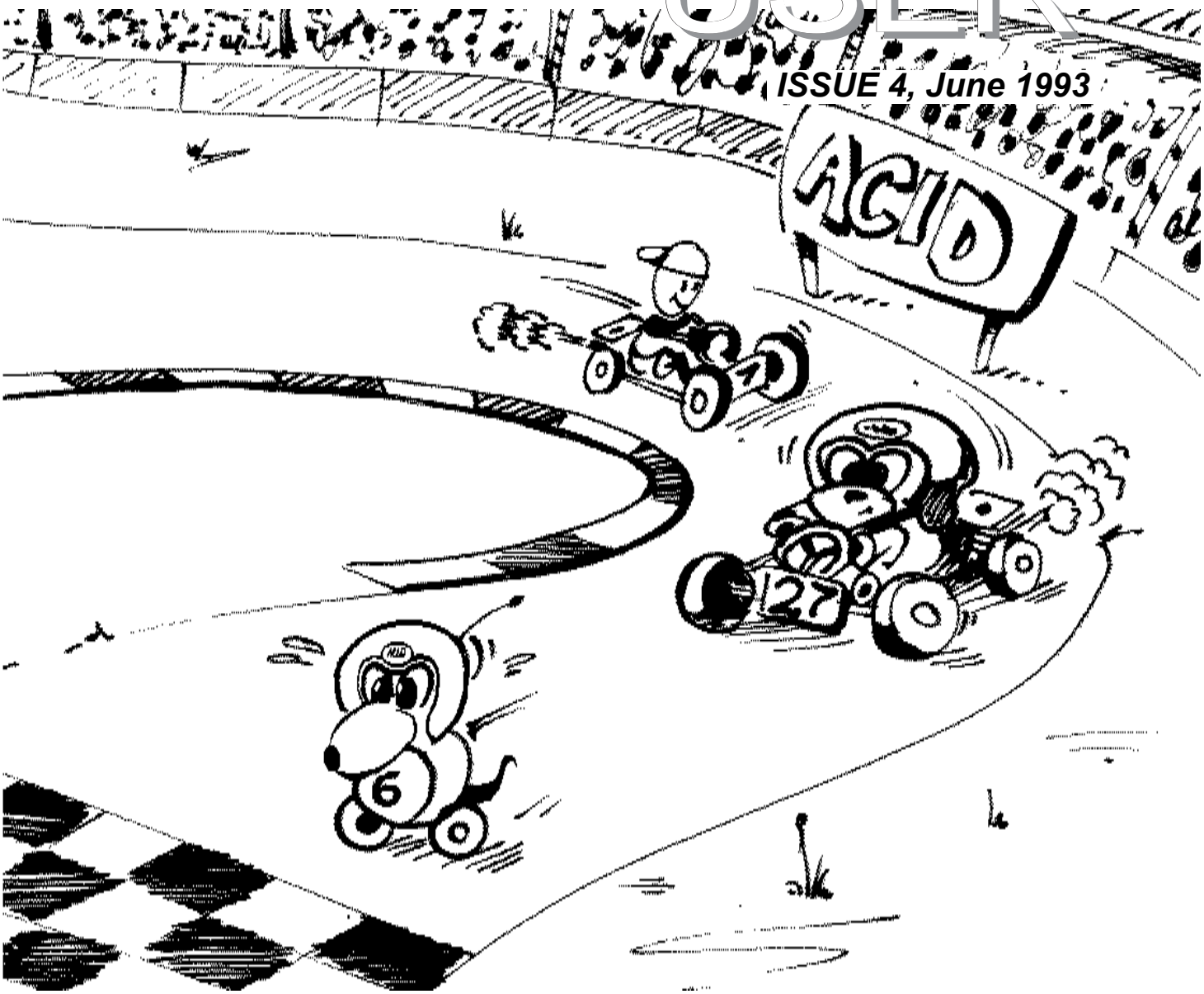
INFORMATION AND SUPPORT FOR BLITZ BASIC 2 USERS WORLD WIDE

BLITZ USER



ISSUE 4, June 1993



New commands for cycling gadgets, radio buttons, AGA screens, filled polygons and more. Plus free demo of Skidmarks our latest inhouse Blitz2 project which will knock your socks off. Beta version of our new Intuitools to plan your applications with and heaps of hints and tips for programming in Blitz2. And a compilation of the Australasian questionaires that have been returned by Down Under Blitz2 Users, Thanks Guys!
Simon :)

BACK IN THE OFFICE...

More gossip from the St Kevins Arcade office of Acid Software & Vision Software...

Severe hardware problems have been encountered over the last few weeks, suprisingly we were to find out, all caused by the office mouse! Gee all those computers and only one mouse you ask, well this mouse is kind of different, he seems to like crawling inside computers via rear-expansion slots and bedding down for the night, and what really sucks, is that he has been having quite a few wet dreams!!! These adolescent rodent fantasies have already been responsible for taking out an entire A2000 mother board (oh dear, now we're getting into incest), and levels of corrosion on two other machines are severe.

Struth cobbers! It seems that every time we get any good press in the English Amiga mags they flatter us buy calling us AUSTRALIANS! Now come on fellas, just because we're on the other side of the world doesn't mean you can lob insults at us with out fear of reprimand! So be on your guard! And speaking of Australia, Acid Software will be represented in Sydney first-third of July at the World Of Commodore show. So if you're coming, don't forget to bring along your latest Blitz2 programs to show us what you're up to.

Skidmarks? What kind of a name for a game is that you ask... A pretty catchy one if you ask me! Anyway, it is likely to be the first software developed in Blitz2 to be released on a commercial basis. Hopefully in a software shop near you mid July, featuring 6 tracks, better computer competition etc. Congratulations to Andrew for prooving you can write such addictive software in Blitz2, hopefully it will spur other Blitz Users to reach greater heights.

Anyway, besides the loud 4 player skidmark competetions (thats the computer version) its been heads down and hard at work. A new game is on it's way from Vision, this time its a rotating helicopter blaster that has to be seen to be beleived (nothing cute about this hell raiser). CD of the month goes to Aerosmith (even though nobody here likes it), drug of the month goes to Nicobrevin which is aiding me in another attempt to kick the cancer sticks....

No. 4

BLITZ USER

Blitz User is a publication of Acid Software.

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

CONTENTS

Back in the office... 2
Gossip from the ever messy hive of activity in St Kevins Arcade.

Editorial 4
Latest state of play in the world of Blitz2 development and marketing...

Letters 5
Yup, original correspondence for once!

Questionaire Results 7
Highly professional statistical analysis of the 1000s of returned questionaires, this month, from our faithful Australasian users.

Buggy Booboos 8
Back by popular demand, an assortment of faux-pars or is that botch ups...

NEW COMMANDS 9
AGA palette control 9
Screen & Bitmap extensions 11
New gadget handling for Blitz2 12
Date & Time commands 14
Environment functions 16
Powerful, new drawing commands 17

Tri Analyse 18
For all the budding Blitz2 Triathletes out there keeping fit (i.e. not for the smoking, drinking, slothful layabout types). Alcohol, Nicotine intake management extension next issue...

Roger's Beginners Column 20
Hand's on Blitz2 for the non-advanced...

Useful OS Calls 24

EDITORIAL

Happy Happy, Joy Joy, it's another issue of Blitz User, woza, infact it's issue 4! The good news is that we now have 100% German manuals available, are back in stock of English packages and are looking forward to selling heaps of Blitz BASIC's in the next few months.

Those interested in developing applications in Blitz2 will appreciate the new features this month. We have extended the gadgets library to handle cycling text gadgets and exclusive gadgets (radio buttons). Plus there's support for some AGA screen modes as well as 24 bit AGA palette handling.

On my side I've extended the commands in the 2D drawing library to handle bitmaps larger than 1024x1024 and also added polygon drawing commands.

Everything should be stable on accelerated Amiga's now that we have sorted out vector base relocation stuff. The debugger and hi-frequency AGA screen modes are still not compatible. We will have a display independent debugger option available soon, until then, you'll need to stick with 50-60Hz displays while developing stuff in Blitz2.

We had excellent responses from the questionaire that went out with issue 3. This month features a compilation of results from our Australasian Blitz Users. Next issue we will cover the German and U.S. responses.

What we would dearly like to see now from Blitz Users are programs written in Blitz2. It's all very well to harp on about support for AGA and 3.0, but surely the primary objective is to develop software that will run on ALL Amigas.

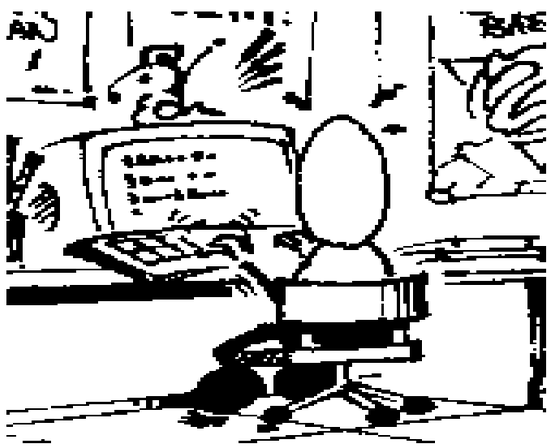
Besides the software we have developed here in NewZealand with Blitz2 I have not seen or heard of anything thats being developed by Blitz Users. Sure we receive lots of messages regarding stuff you guys want added but it would certainly help the cause if we got a disk every now and then with programs people are working on. Otherwise it starts to feel as though Blitz2 is just our in-house development language...

Hmmm, what else is in this issue. Well I could go on about Skidmarks, it's certainly waisted enough of my time play testing it for Andrew :) With the new tracks added I'm really looking forward to showing that not only can you write commercial quality software in Blitz2 but you can write chart-topping software in Blitz2 as well.

O.K. so whats on the top of our list of things to do now that issue 4 is finished? For starters we need AGA support in Blitz mode so we can get a 256 colour version of skidmarks released. Hi-frequency debugging is high on the list.

I'll be polishing off the new Intuitools program and hopefully get it to generate most of the code for the user interface so programmers can concentrate on the core routines etc.

Mark has been working on his own 3D library and has developed a rather amazing 3D shape editor. Hopefully that and the library will be ready for next issue. In the



LETTERS

Dear BlitzMan

I am a registered owner of both the original Blitz Basic and Blitz2. Using Blitz I wrote "Sword of the Warlock" a Bard's Tale type game. I've sold a small number of copies and received a lot of compliments about it. Now I'd like to upgrade Warlock using Blitz2.

I sent Acid Software in Mesa Arizona my $15.00 for the annual newsletter sub and after several months still have no newsletter or disk updates. I can't tell you how important it is that I get some help. I want to begin programming but can't really start unless I have all the commands I need.

Thanks for your time and help.

Alan Broz

Alan, you could have sent a copy of Sword of Warlock with the letter! Anyway sorry about the U.S. support for Blitz2. Hopefully things are now straightened out and you have got hold of the previous upgrades and can start porting the game to Blitz2, look forward to hearing how you get on.

Dear BlitzMan

Blitz2.0 is working fine, however I have a few questions:

* When compiling a custom library do I use the Create File or Create Resident option?

* When compiling the custom library I continue to get "Wrong Number of Parameters" on the !subs line

* How do I change the timing of the soundtracker player for proper replay on NTSC machines

* Do you have a BBS where registered users can get access to update files.

Mike Hurt

Mike, thanks for your letter. You need to "Create File" when compiling a custom library such as that found in issue #1 of BUM. Make sure you are using curly brackets when calling macros such as !subs, also you need to supply 3 parameters. (More info in BUM#2)

At the moment we do not have support for correct soundtracker replay on NTSC machines, we'll get onto it. As for the BBS, we should have one up and running in the U.S.

Call Dave Maziarka on (608)257-1975 for more information.

Dear Blitzman,

I have been using Blitz2 for a couple of

weeks and beleive it to be a pretty good language with a lot of potential, although there are quite a few places where BBS could stand a great deal of improvement, namely the editor:
1st up the Language in General:

1. access to system date
2. ability to access bold, italics and underlining when printing to both the screen and especially the printer 3. access to speech synthesiser

2nd the editor:

1. use of any font rather than the two supplied

2. when you hit return the editor should open a new line at the current cursor position

3. backspace should delete end of line characters

4. auto indenting of structure code

If these few changes were implemented there is no telling how far this language could go in the Amiga community. I for one would recommend your language to my friends and fellow Amiga users.

As it is now I have a slight problem recommending your program because of these slight imperfections...

Thanks for your kind attention...
Dewey A Fish

Hi Dewey, thanks for your comments. We have had LOTS of complaints about TED, and hopefully we will upgrade him again soon including most of the features you mention.

As for the compiler, we have support for speech and system date/time stuff already, print codes etc. hopefully next month. Stay Tuned!

Dear Blitzers,

A friend of mine uses Blitz2 and swears that it is the best thing he has ever seen. I saw some of the stuff he has done and I was really impressed.

I just recently bought an A1200. Do you have a version of Blitz2 which supports the AGA modes as well as the standard ones? If so when will it be available?

Hope to hear from you soon

Kevan Stannard.

Yo Kevan, If you're reading this at your friends place then you'll be pleased to see the inclusion of some of the AGA stuff you requested in this issue. If you've gone and bought Blitz2 and registered then I suppose you'll be even happier that you can start messing with 24 bit color etc. right away.

If you have any comments about Blitz2 or just want to say hi and let us know what you are working on at the moment then please, drop us a line!

Send mail to:

Blitz 2 Questionaire Australasian Results

The following is the results from the 40 or so questionaires returned from Australasian Blitz Users. German and U.S. results will be published in issue 5 of Blitz User Magazine.

Hardware

On the hardware front I was quite suprised to see how many people have upgraded to the new A1200 machine. We've got two in the office and although we still haven't added any fastmem to them due to the stupid prices, they seem to be a fast reliable machine.

So stats for the different Amigas are:

A500 users = 44%
A1200 users = 18%
A2000 users = 18%
A1000 users = 7%
A3000 users = 7%
A4000 users = 7%
A600 users = 0% (wo, popular machine!)

Also 56% of Blitz Users have modems.

Languages

As was suspected Amos and Arexx come in tops, to be expected I suppose, quite a few people have been messing around with E which we have also had a dabble with.

AMOS	25%
AREXX	25%
AMIGABASIC	8%
CANDO	8%
E	5%
C	5%
ASSEMBLER	5%
GFA	3%
PASCAL	0% (for A600 owners only!)

Not as many assembler programmers as I expected, maybe I'm the only one who regards Blitz2 as the best assembler around, oh well...

Using Blitz2

Well, you couldn't get a more mixed bag than asking what type of programs people wanted to develop in Blitz2. This of course doesn't make our job any easier having to cater for every man and his dog!

The main things that stood out were perhaps a need for better audio control and some discussion on calling DOS commands from Blitz2.

The results for types of programs users were keen on developing were:

Utilities	50%
Games	41%
Strategy	12%
Music	12%
Educational	10%
Graphics	8%

Conclusion

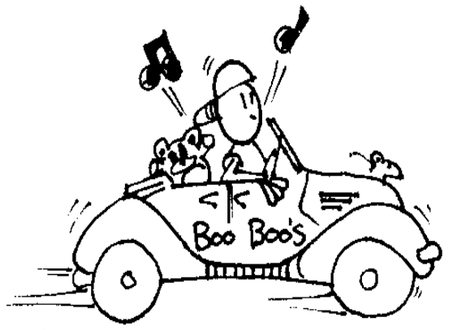
Overall, the general consensus was that people were happy with Blitz2, a lot were having trouble getting to grips with the documentation (which is an ongoing job for us) and we were definately not supporting the beginners enough.

There is a new User Guide that users can upgrade to (included in German manual), and we will be developing a series of beginner tutorials to coincide with a special magazine promotion that we will be starting in the U.K. shortly.

Anyway, if you haven't returned your questionaire please do so today, we will be doing another at the end of the year to try and gauge where we are heading and to keep up to date with what you the user need from Blitz2, the language that just keeps on growing!

German & U.S. scores next month...

BUGGY BOOBOOS



This column has been reinstated after several issues which, yes OK we did have lots of bug reports, well anyway this will be a regular feature.

Once again, if you find a bug in Blitz2, have some serious problems with the documentation please drop us a line, otherwise, how do we know to fix them????

Compiled list of bugs fixed since release...

Accelerated Amigas have the ability to relocate trap and interupt vectors in 32 bit memory, Blitz2 now handles this minor inconvenience.

All bitmap based drawing commands can now be used on bitmaps larger than 1024x1024.

Program exit from Blitz mode was slightly hairy due to Blitz2 not waiting for a BlitFinished at the right time, all fixed now.

Rastport and ViewPort were returning wrong results in the first release, not anymore.

"Variable Already Shared" compile time messages were appearing for no logical reason, its been sussed.

Multiple byte fields in NewTypes were causing odd address problems until v1.5 or so, v1.6 has no such problems.

Ted lost the German chars and went NTSC uncompatible in v1.5, he's recovered and is now in a stable condition.

Slices on AGA machines were crapping out until AGA arrived in NewZealand, as soon as this happened (6 months after the rest of the world) it was sorted out.

The optimiser was removed due to it's ability to generate non 680x0 compatible code, may reappear in the near future.

Square brackets were causing a few problems in NewTypes this was fixed around v1.3.

The .info of source and object code is not overwritten by Ted if it already exists so image and position are not stuffed up by resaving/creating your program.

In version 1.5 or so, the system libraries were overhauled to handle the release of 3.0. This meant a LOT of the tokens were changed and source code was NOT backward compatible! SORRY. Anyway, to get round this problem, programs that use operating system calls should be loaded into the older version of Blitz2, saved as ascii (using Write-Block), and THEN loaded into new version of Blitz2. This problem is applicable to the QUICK-ASM example on the examples disk.

AGA PALETTE HANDLING

Blitz 2's palette object has (again) changed. Palette objects are now capable of containing AGA compatible 24 bit colours.

The new palette objects look likethis:

```
NEWTYPE.rgbcomp
_red.l                  ;left justified red component.
_green.l                ;left justified green component.
_blue.l                 ;left justified blue component.
End NEWTYPE

NEWTYPE.palettedata
_numcols.w              ;same as palette/_numcols.
_zero.w                 ;for compatibility with graphics lib LoadRGB32.
_rgbs.rgbcomp[256] ;256 is the max... the amount will actually
                        ;depend upon the highest accessed palette entry.
_zero2.l                ;for graphics lib too.
End NEWTYPE
```

This is the actual object return by Addr Palette(n):

```
NEWTYPE.palette
_*data.palettedata          ;00: NULL if no palette present, else a pointer to
                        ;    palettedata.
_numcols.w              ;04: number of colours present in palettedata.
                        ;below is colour cycling info.
_lowcol.w               ;06: low colour for cycle range.
_hicol.w                ;08: high colour for cycle range.
_speed.w                ;10: speed of cycle : 16384 = max speed, sign of speed
                        ;    indicates cycling direction.
_var.w                  ;12: cvariable speed is added to.
                        ;
                        ; more possible cycling entries....
                        ;
                        ;128: sizeof.
End NEWTYPE
```

Now for the new AGA functions added to Blitz 2...these will all generate a runtime error if used on a non-AGA Amiga:

Statement: **AGARGB**

Syntax: **AGARGB** *Colour Register,Red,Green,Blue*

Modes: Amiga

Description:

The AGARGB command is the AGA equivalent of the RGB command.

The 'Red', 'Green' and 'Blue' parameters must be in the range 0 through 255, while 'Colour Register' is limited to the number of colours available on the currently used screen.
Example:

```
;
; AGA test
;

Screen 0,0,0,1280,512,8,$8024,"SUPER HIRES LACED 256 COLOR SCREEN",1,2

ScreensBitMap 0,0

For i=0 To 255
    AGARGB i,i/2,i/3,i      ;shades of purple
    Circle 640,256,i*2,i,i  ;big SMOOTH circles
Next

MouseWait
```

Statement: **AGAPalRGB**

Syntax: **AGAPalRGB** *Palette#,Colour Register,Red,Green,Blue*

Modes: Amiga

Description:

The AGAPalRGB command is the AGA equivalent of the PalRGB command. AGAPalRGB allows you to set an individual colour register within a palette object. This command only sets up an entry in a palette object, and will not alter the actual screen palette until a 'Use Palette' is executed.

Function: **AGARed**

Syntax: **AGARed***(colour register)*

Modes: Amiga

Description:

The AGARed function returns the red component of the specified colour register within the currently used screen. The returned value will be within the range 0 (being no red) through 255 (being full red).

Function: **AGAGreen**

Syntax: **AGAGreen***(colour register)*

Modes: Amiga

Description:

The AGAGreen function returns the green component of the specified colour register within the currently used screen. The returned value will be within the range 0 (being no green) through 255 (being full green).

Function: **AGABlue**

Syntax: **AGABlue***(colour register)*

Modes: Amiga

Description:

The AGABlue function returns the blue component of the specified colour register within the currently used screen. The returned value will be within the range 0 (being no blue) through 255 (being full blue).

NEW SCREEN FLAGS

The superhires viewmode flag $20 is now acceptable, but should always be used in conjunction with the standard hires flag of $8000.

The depth of a screen may now be specified up to 8 bitplanes (256 colours) deep (if you've got an AGA machine!). Here's how you would go about opening a superhires, 256 colour screen:

```
Screen 0,0,0,1280,256,8,$8020,"MyScreen",1,0
```

3.0 BITMAP HANDLING

Blitz 2's Bitmap object has been upgraded to allow for interleaved bitmaps:

```
NewType.Bitmap
_   ebwidth[0]          ;00: for compatability.
_   linemod.w           ;00: value to get from one scanline to next.
_   height.w            ;02: currently pixel height - but open to
commodore 'enhancement'.
_   depth.w             ;04: number of bitplanes.
_   pad.b[2]            ;06: nothing.
_   data.l[8]           ;08: actual bitplane pointers.
_   pad2.b[12]          ;40: zilch.
_   flags.w             ;0=normal bitmap, <0=interleaved.
_   bitplanemod.w           ;value to get from one bitplane to next. MAY BE 0!
_   xclip.w             ;pixel width for render clipping
_   yclip.w             ;pixel height for render clipping
```

_ cclip.w ;number of colours available on bitmap (= 2^_depth)
_ isreal.w ;0=no bitmap here, <0=blitz created bitmap, >0=borrowed
;64: sizeof
End NEWTYPE

Also, many Blitz2 bitmap related commands have been altered to take this new object into account.

NEW GADGET HANDLING

A new bit, bit 9, in the 'Flags' parameter of
the 'TextGadget' and 'ShapeGadget' commands allow you to create mutually exclusive radio button type gadgets. These gadgets DO NOT require Kickstart 2.0 to operate!

Here is an example of setting up some radio button style text gadgets:

TextGadget 0,16,16,512,1,"OPTION 1":**Toggle** 0,1,On
TextGadget 0,16,32,512,2,"OPTION 2"
TextGadget 0,16,48,512,3,"OPTION 3"

The new 'ButtonGroup' command allows you to specify which 'group' a series of button gadgets belong to. See 'ButtonGadget' below.

Note that if you are using button gadgets, you SHOULD really toggle ONE of the gadgets 'On' before giving the gadgetlist to a window - as in the example above.

Text Gadgets may now be used to create 'cycling' gadgets. Again, these gadgets DO NOT require kickstart 2.0 to work.

If you create a text gadget which contains the '|' character in the gadget's text, Blitz 2 will recognize this as a 'cycling' gadget, using the '|' character to separate the options - like this:

TextGadget 0,16,16,0,1," HELLO |GOODBYE| SEEYA |"

Now, each time this gadget is clicked on, the gadgets text will cycle
through 'Hello', 'GOODBYE' and 'SEEYA'. Note that each option is spaced out to be of equal length. This feature should not be used with a GadgetJam mode of 0.

Now for the new gadget commands:

Function: **GadgetStatus**

Syntax: **GadgetStatus***(GadgetList#,Id)*

Modes: Amiga

Description:

GadgetStatus may be used to determine the status of the specified gadget. In the case of 'toggle' type gadget, GadgetStatus will return true (-1) if the gadget is currently on, or

false (0) if the gadget is currently off.

In the case of a cycling text gadget, GadgetStatus will return a value of 1 or greater representing the currently displayed text within the gadget.

Statement: **ButtonGroup**

Syntax: **ButtonGroup** *Group*

Modes: Amiga

Description:

ButtonGroup allows you to determine which 'group' a number of button type gadgets belong to. Following the execution of ButtonGroup, any button gadgets created will be identified as belonging to the specified group. The upshot of all this is that button gadgets are only mutually exclusive to other button gadgets within the same group.

'Group' must be a positive number greater than 0. Any button gadgets created before a 'ButtonGroup' command is executed will belong to group 1.

Function: **ButtonId**

Syntax: **ButtonId***(GadgetList#,ButtonGroup)*

Modes: Amiga

Description:

ButtonId may be used to determine which gadget within a group of button type gadgets is currently selected. The value returned will be the GadgetId of the button gadget currently selected.

Statements: **Enable & Disable**

Syntax: **Enable** *GadgetList#,Id* & **Disable** *GadgetList#,Id*

Modes: Amiga

Description:

A gadget when disabled is covered by a "mesh" and can not be accessed by the user. The commands Enable and Disable allow the programmer to access this feature of Intuition.

Statement: **SetGadgetStatus**

Syntax: **SetGadgetStatus** *GadgetList#,Id,Value*

Modes: Amiga

Description:

SetGadgetStatus is used to set a cycling text gadget to a particular value, once set **ReDraw** should be used to refresh the gadget to reflect it's new value.

NEW GADGETS EXAMPLE:

```
;
; new gadget types
;
WBStartup:FindScreen 0     ;open on workbench

TextGadget 0,32,14,0,0,"CYCLE 1|CYCLE 2|CYCLE 3"  ;cycling gadget

ButtonGroup 1  ;first group of radio buttons follows
For i=1 To 5:TextGadget 0,32,14+i*14,512,i,"CHANNEL #"+Str$(i):Next

ButtonGroup 2  ;second group of radio buttons follows
For i=6 To 10:TextGadget 0,32,14+i*14,512,i,"BAND #"+Str$(i):Next

Window 0,20,20,160,180,$1008,"GADGET TEST",1,2,0  ;open window with gadgets

Repeat              ;wait until close window gadget hit
     ev.l=WaitEvent
Until ev=$200
```

DATE & TIME COMMANDS

Function: **SystemDate**

Syntax: **SystemDate**

Modes: Amiga

Description:

SystemDate returns the system date as the number of days passed since 1/1/1978.

Example:

```
;
; date/time test
;
Dim d$(6):Restore daynames:For i=0 To 6:Read d$(i):Next
Dim m$(12):Restore monthnames:For i=1 To 12:Read m$(i):Next

NPrint Date$(SystemDate)
NPrint d$(WeekDay)," ",Days," ",m$(Months)," ",Years
NPrint Hours,":",Mins,":",Secs
NPrint "press mouse to quit"
MouseWait

daynames:
Data$
SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY
monthnames:
Data$ JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
```

Function: **Date$**

Syntax: **Date$** (days)

Modes: Amiga

Description:

Date$ converts the format returned by SystemDate (days passed since 1/1/1978) into a string format of dd/mm/yyyy or mm/dd/yyyy depending on the dateformat (defaults to 0).

Function: **NumDays**

Syntax: **NumDays** (date$)

Modes: Amiga

Description:

Numdays converts a Date$ in the above format to the day count format, where numdays is the number of days since 1/1/1978.

Statement: **DateFormat**

Syntax: **DateFormat** format# ; 0 or 1

Modes: Amiga

Description:

DateFormat configures the way both date$ and numdays treat a string representation of the date: 0=dd/mm/yyyy and 1=mm/dd/yyyy

Functions: **Days Months Years & WeekDay**

Syntax: **Days Months Years & WeekDay**

Modes: Amiga

Description:

Days Months and Years each return the particular value relevant to the last call to **SystemDate**. They are most useful for when the program needs to format the output of the date other than that produced by date$. WeekDay returns which day of the week it is with Sunday=0 through to Saturday=6.

Functions: **Hours Mins & Secs**

Syntax: **Hours Mins & Secs**

Modes: Amiga

Description:

Hours, Mins and Secs return the time of day when **SystemDate** was last called.

ENVIRONMENT COMMANDS

Functions: **WBWidth Height Depth & ViewMode**

Syntax: **WBWidth, WBHeight, WBDepth & WBViewMode**

Modes: Amiga

Description:

The functions WBWidth, WBHeight, WBDepth & WBViewMode return the width, height, depth & viewmode of the current WorkBench screen as configured by preferences.

Functions: **Processor & ExecVersion**

Syntax: **Processor & ExecVersion**

Modes: Amiga

Description:

The two functions Processor & ExecVersion return the relevant information about the system the program is running on. The values returned are as follows:

ExecVersion	OperatingSystem Release	Processor	Part#
33	1.2	0	68000
34?	1.3	1	68010
36	2.0	2	68020
39	3.0	3	68030
		4	68040

NEW DRAWING COMMANDS

Statement: **Poly & Polyf**

Syntax: **Poly** numpoints,*coords.w,color & **Polyf** numpoints,*coords.w,color[,color2]

Modes: Amiga/Blitz

Description:

Poly & **Polyf** are bitmap based commands such as **Box** and **Line**. They draw polygons (unfilled and filled respectively) using coordinates from an array or newtype of words. Polyf has an optional parameter color2, if used this colour will be used if the coordinates are listed in anti-clockwise order, useful for 3D type applications. If color2= -1 then the polygon is not drawn if the verticies are listed in anti-clockwise order.

Example:

```
;
; polygon test using triangle newtype
;
NEWTYPE .tri:x0.w:y0:x1:y1:x2:y2:End NEWTYPE
BLITZ
BitMap 0,320,256,3
Slice 0,44,3:Show 0
While Joyb(0)=0
      a.tri\x0=Rnd(320),Rnd(256),Rnd(320),Rnd(256),Rnd(320),Rnd(256)
      Polyf 3,a,1+Rnd(7)
Wend
```

Statement: **BitPlanesBitMap**

Syntax: **BitPlanesBitMap** *SrcBitMap, DestBitMap, PlanePick*

Modes: Amiga/Blitz

Description:

Tri Analyse

Hi again to all you Blitz people out there. Well its early June and here in New Zealand that means cold miserable winter. Perfect for programming computers but terrible for triathlon training.

Triathlon training? Well the reason I brought that up is that is what the accompanying programme is all about, It's a triathlon training database that allows you to enter you various types of training info into a data file and retrieve them with ease. It will also perform some graphing functions which I'll go into later.

This programme shows you that Blitz is not just for writing games (though it certainly is good at it) but can quite happily generate applications software *Tri Analyse* is reasonably complex and demonstrates how to do the following things.

Set up screens & windows

Set up gadgets including textgadgets, string gadgets, and showing some of the new gadget commands presented elsewhere in this issue like mutual exclude gadgets, and disabled gadgets

It also shows how convenient Newtypes are for storing records how creating functions and statements can tidy up your code, and save you typing!

How to use the Amiga's inbuilt library functions from within Blitz

And much more!

As this is Blitz User Magazine and not a fitness publication I have put a ascii file called TriReadMe.Doc in the Tri directory on the cover disk, for details of how to use the programme please refer to that, If you are a triathlete or cyclist or runner, I would hope that you find it useful, drop us a line and let us know about your training.

Right, on with a description of the programming for all you code heads out there.

I split Tri Analyse into two separate listings, the first (Tri.inc) contains all the function and statement definitions used by the main programme, the function is a Yes No requester which returns a true or false value for the main programme, and the statement is a simple alert requester to bring the users attention to something. It then defines all the Newtypes (structures) for the records to be stored in. When writing a database type programme such as this you need to work backwards, the first thing you have to do is decide on what you want the programme to output, my requirements were as follows.

1. Each diskfile had to contain 1 months worth of training
2. Each day could handle information for all 3 disciplines of the triathlon, Swimming, Cycling, and Running.
3. Each discipline had to hold the resting heartrate, what type of training was actually done, and a string that described the training for the day.

With this in mind I ended up with the following newtypes and arrays.

```
NEWTYPE.data
 spare.w        ;just in case
 heartrate.w    ;heartrate in BPM
 train.w                ;0 - endurance 1 - aerobic
                        ;2 - sprint (speed training)
 text.s                 ;editor generated text.
End NEWTYPE

NEWTYPE.file    ;this is the record for a DAY
 dis.data[3]    ;include the data newtype 3 times for the 3
;disc 0 1               ;2 swimming 0, cycling 1, running 2
End NEWTYPE
```

```
Dim current.file(32)   ;dim a 32 day month 1-31 for convenience
```

So we have a array called current, who has a subset of the newtype file, which has 3 subsets of the type data, In this way we can easily reference data for any discipline on any day example:

To access the text about my cycle training on the 13th of a month would be

```
thetext$ = current(13)\dis[1]\text
```

This is how newtypes and nested newtypes really come into their own. Right now that we have the data structure we needed to design the programme to fill it.

First up I did the User Interface. Created your standard WBStyle 640x256 PAL Screen (sorry American users use one of those PAL utils :)) Then a window was generated, non draggable etc, simply for its ability to allow us to attach a gadgetlist, and use different sized fonts etc.

Then the gadgets, these had to be put in before the window was opened obviously. I used GadgetBorder to divide the screen into logical segments for groups of different operations, GadgetBorder is a purely cosmetic function that does not affect gadgets at all but I think you will agree that it can make a display look a whole heap better. I set my gadgetpens to give the buttons (texgadgets) a 'raised look' and designed all those, then swapped the gadgetpens to give a 'reccessed' look and did all the StringGadgets. The last two Gadgets were designed with a bigger font and the GadgetPens were set for a raised look again.

Also at relevant points through-out the code, various gadgets were assigned buttongroups (for the ones that were set up for mutual exclude) and certain gadgets were preset on or disabled completely (see later in this issue of Blitz User #4 for details on the new Gadget commands.)

Right now the interface was out the way. All that remained was the main gadget handling loop (the *event* loop), And the coding of all the subroutines to deal to the entered gadget data in a meaningful way.

Ok, the way I had designed this programme in my mind the piece of data on which all others relied was the entering of a valid date, after all all the various bits of the newtype structures were reference from the date so if a valid date was not there then no entered data would be valid either.

So the first subroutine I did was the checkdate one.

The checkdate routine does quite a bit, the first thing it does is check the length of the string we entered since it wants DD/MM/YY format if the string is not the correct length (8) we can assume the date is invalid and tell the user if the length is correct we extract two sets of data the individual day month and year are stored as strings for use in the filesave and load routines, and are also converted to numeric variables for use in the various offsets in the arrays and newtypes. We then check the individual values for BASIC validity (we assume that every month could theoretically be up to 31 days long as it does not affect the file, saves a lot of checking and at worst the user could create something like the 30th of Feb -- A bit lazy on my part but no big deal.) If the date syntax is invalid we tell the user via a nifty alert function and set the date flag to false, at the end of this if the date is false, we clear the datefield and wait for the user to enter another date.

Right, now that we've established that we have a valid date we set the date flag to true, we then check that the month we have asked for is not an already open file if we have asked for a new month, we save the old month (if there is no old month a dummy yearfile will be written but this once again is a few bytes and not a problem. Now that we have saved the oldmonth we make the oldmonth the currentmonth and go and try and read our current month. If we successfully read it, we reset the fileread flag to false (so its rechecked nextime) otherwise, the read of the month must have failed so we have to create a new one, first of all we zero all the data fields so our new month does not get the last months data, and then we create the file.

If way back at the first step the month of the date we entered was the SAME as the last entry we do NO file-handling, as it is already open, This routine is quite a groovy way of handling records as it eliminates excessive disk access and shields the user from having to load files.

Now that the checkdate routine has been completed we then check to see

if a valid date was the result again (by the date flag) and if so refresh our mutual exclude gadgets so that they take note of any changes in training type on our new day, this has the advantage that if there is no training currently entered for that day the two banks of gadgets are set to a known state (ie SWIMMING and ENDURANCE).

All the other gadget events are to do with handling all the other bits of the programme like changing disciplines and training types, editing data, graphing it, saving data (as you need to do this for the last month you do before quitting the programme) setting our default editor type and quitting the prog, lets take a brief look at a few of these routines.

1. Selecting your own text editor

I am too lazy to write my own text editor, also they are rather a beast to write so rather than re-invent the wheel I've decided to take advantage of Blitz 2's ability to utilise the Amigas operating system calls. The call I am using here is the 'Execute_' function, which goes away and executes a programme. One annoying thing about the Execute_ function is that it requires the Run command to be present in the C: directory but that is the norm rather than the exception so as long as the user is told about this in the documentation its OK.

The other thing to be wary about the execute command is that some programmes run as a detached process, this means when you execute them they run, but detach themselves from your programme and return control to your programme, Cygnus-Ed is an example of a editor that does this. This 'feature' sucks as far as we are concerned so use a nice ATTACHED editor such as TED or Ed or something.

So to implement a Editor in our programme we have done two things

1. Given the user the ability to enter his own command string for his editor of choice and save it in a preferences or config file so that its all set up for him next time (Case 3 in the main loop)

2. When the person goes to edit some text we once again check for a valid date, then call the Execute) function with the pathname and the command for the editor, and two nulls for the other arguments. We return a result that we check for success, but I have since found out that Execute returns a true result even if the execution of the command was not successful. GO FIGURE? This is not to much of a prob cos the average clued up user is going to go 'Hmmm no editor here, perhaps my command string is wrong' and go and fix the problem.

I am also making the rather naughty assumption that everyone is going to have a ram: disk as I originally wrote this programme for my own use only and only decided to put it on blitz user at the last minute. So if you don't have a ram disk, you can substitute your own desired disk device and recompile the source, (if its someone that does not own Blitz Basic -- Tuff luck :-)

the user is then to enter the text which gets saved in a file called ram:temp which upon exiting the editor we store in the current days and disciplines text field, VIOLA

2. Altering our training types and disciplines

This is a piece of cake, if the gadget is clicked on by you (or it is altered in the refresh process, we simply have two variables curdis and curtype which we set to the various flag values for inputting data into our newtype.

3. The Heartrate

The heartrate, simply type in a number we go away and convert this string to a numerical value and store it in all 3 disciplines for the current day, (This is because you only have one resting heart rate a day, not one for each discipline). Note how I've set the stringgadget entry length to 3 chars max, this is good cos your never going to have a heartrate of 999, and it means we don't have to check that a number to big to be a WORD value is going to be entered.

4. Read todays Training

The read todays training simply grabs the days training, dumps it back out to the ram:temp file (so that it can be re-edited as well) and views it using a little text viewer.

5. Refreshing the Gadgets via software for a new date or discipline

When we do the refresh gadgets for a new date or discipline, you will notice that there is little border sometimes left round the 'unselected' gadgets, this is something to do with the way Mark handles the gadgets in gadgetlib and I have not worked a way round it, I will ask mark if it can be changed, or maybe even (SHOCK HORROR) I'm just programming it wrong :-)

Notice that the way the gadget refresh is programmed is modular ie: When there is a new date you have to refresh both the heartrate and the training type. So you have two refresh routines and you then call both those with another routine called refreshall. However if its just a discipline that's changed all you do is call the refreshdis routine, this is an example of tidy modular code.

6. The Graph Routine

The graph routine simply takes the data for the current month loads a IFF backdrop of a graph (which was created with a separate little blitz prog), and then if the heartrate falls between certain values it uses some math to work out how high and where and what colour to plot the heartrate.

7. Miscellaneous

This programme is far from perfect, it assumes too much and the way the editor is called is a trifle crusty, however it suites my purpose fine and it is a good example of how to work round problems if time or ability stops you from doing something (try writing a text editor one day and you'll see what I mean :-)

I have put the little graph maker programme in the directory as well, all this did was draw out my graph and print all the numbers in the right place and save it as an IFF, this is a good example of little pieces of utility code that can save you time, it would have taken heaps longer to design the graph and put all the little numbers down as brushes than it did to plot them out using bit of math, the results are accurate too.

One interesting aside about the Graph creator is that it shows a way of cheating when you want to use non 8x8 fonts on a bitmap, simply open a full screen invisible window to do your font drawing on and then convert to a bitmap afterwards.

8. Functions and Statements

The two requesters designed in tri.inc are good examples of how to make your code, smaller, more reliable and easy to understand and develop

The yes no function just opens a window with a gadget list, waits for a gadget event from that window , if its a yes the function returns true otherwise if its a no it returns false. This enables you to act on the result of the requester in your code.

The alert function opens a window with a gadget list, waits for a gadget event from that window, if the ok gadget is pressed then the window closes and your programme is able to continue.

Both the function and statement allow the programmer to display a line of centred text and allow the user to answer the requester using ESC=NO RETURN=YES keys thus making the programme more versitile.

These are probably one of the best examples of tidy modular coding in the whole project, so now that you have seen just what sort of things you can do with user defined functions/statements... Use them yourself.

Well that's it basically, as I said for the athletes out there of you the actual documentation for USING Tri-Analyse is on the disk, hope this gives you all some inspiration to start developing in Blitz.

Seeya

Roger.

BitPlanesBitMap creates a `dummy' bitmap from the SrcBitMap with only the bitplanes specified by the PlanePick mask. This is useful for shadow effects etc. where blitting speed can be speed up because of the fewer bitplanes involved

Statement: **ClipBlit**

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

Modes: Amiga/Blitz

Description:

ClipBlit is the same as the Blit command except ClipBlit will clip the shape to the inside of the used bitmap, all blit commands in Blitz2 are due to be expanded with this feature.

SKIDMARKS INSTRUCTIONS

At all times, "F10" will advance you through the game, and "ESC" will quit you back out.

Press ESC from the TitleScreen to quit back to WorkBench. Select F10 to start the game.

Click on Link to connect two Amigas via null modem cable or modem.

To connect via modme one user should click on box marked "ATDT" and enter phone number to dial, the other should select "ANSWER" when the call comes through.

After Connecting & Linking the bottom panel dissapears, you may now chat with the other machine.

After starting the game with F10, change the control method of car with the labelled function keys.

Pressing "F8" toggles between "Race" and "Champ" modes. While in "Champ" mode, you may only choose the white car. The aim is to beat your best lap time (the red car), effectively racing against yourself.

Control Methods:

KEYS: Use the arrow keys, up, to accelerate, down to reverse and left/right to steer.
CTRL: keys on the left of the keyboard, CTRL to accelerate, shift to reverse, and Alt/Amiga to steer.
JOY1: Use a joystick plugged into the second joystick port.
JOY0: Use a joystick plugged into the mouse port.
COMP: The computer will take control of this car.
LINK: The other computer (modem or cable) controls this car.

Champ Mode:

BEST: The best lap-time your car has done since entering Champ Mode.
GOOD: The second-best lap-time your car has acheived.
LAST: Repeats your last lap, exactly as you drove it.