AMOS Newsletter

VOLUME 9, NOVEMBER 1991

WELCOME AND MERRY CHRISTMAS

Welcome to issue 9 of the AUSSIE AMOS NEWSLETTER! And before I forget, MERRY CHRISTMAS to you all! This edition is running a bit later than I had hoped but as yet I have not been able to find a way to write these things underwater! Well as you can see there are a number a changes in this issue, such as - more pages, more articles, more programs, larger print and a separate PD listing/order page. I think that putting those two questions on the bottom of the order form was one of the best things I have done for a while! It gives you no reason not to give me some feedback on what you like and dislike about the newsletter, and believe me the response has been very good. Remember, the more feedback I get, the better I can make the newsletter.

While we are on the subject of the newsletter, a small number of you wanted to see the newsletter become bigger or have more in it. There has to be a limit on the size of the newsletter, don't expect to see a newsletter which is 20 pages in size because it just won't happen at this stage. On a direct comparison with the British newsletter, number 4 has just come out, and as you can see we are up to number 9. If the British newsletter was produced in a similar format to this one then the newsletter would be about two thirds the size of the Aussie version! The Aussie version has never become smaller, it has grown with every new one and this issue is no exception! It takes, as I said before, about 100 to 200 hours (After Hours not Business Hours) to produce each newsletter, I am not complaining, If I didn't like writing them I wouldn't do it! But there has to be a limit so that I have some time to pursue my own AMOS interests, so before you complain that there is not enough in it then spare a thought for the person who spends the time writing and putting it together and then get off your butts and write an article yourselves. Then there would be more in it and you would have no reason to complain.

One big change that you should have noticed is the fact that the text size has been increased in all the articles, and also in the program listings. Both have been increased by one point. This should make it easier to read without a magnifying glass! What I am looking for at the moment is someone who would like to write a series of articles or even just a single article on any subject to do with AMOS. If you think that you can put some words onto a disk that make some sort of sense, then give me a call during business hours on (02)748 4700 and we will discuss your ideas. You will be able to choose disk(s) from the PD library in return for your efforts, but we will discuss this when you give me a call! Keep your eyes out in Australian Commodore Review for the new AMOS column which is being written by Sausage, it should first appear in the December edition.

If you live in the ACT, or you don't mind travelling, then the following is going to be of great interest to you indeed. CAUSe, Canberra Amiga Users Society have started an AMOS Special Interest Group. There has already been at least one meeting of this SIG which was very successful with quite a number of people attending, and this was just from word of mouth. The whole idea for this SIG is so that AMOS users can get together and meet other AMOS users, discuss problems and programming techniques as well as maybe even having a good time! For more information see the article on page 2.

On page 5 there is an interesting article, complete with sample source code on using the new Serial Extension. There is a lengthy article on using the various commands as well as a simple terminal program complete with an explanation. So if the Serial Extension is driving you around the bend, or even worse to the GURU then you simply must read this article! Also on page 5 I have written an article on how to check if a disk is write protected, or locked in the case of a hard drive. This involves the calling of internal libary functions to read the status of a disk/drive. Being able to check the write protect status of a disk before attempting to write to it is of great benefit to people working with files, well at least you can stop the workbench requester from popping up!

Also in this issue is the final instalment of the Adventure series, about time you all say! There's heaps more as well so I will leave you to it! Christmas is just around the corner, so from all of us to all of you have a MERRY CHRISTMAS and an AMOS NEW YEAR!

Neil (TEX) Miller, Wayne (SAUSAGE) Johnson.

TABLE OF CONTENTS

TRACK'N ON!

Hello once again to everyone in sequencerland. Got a bit more in this edition now that the newsletter is 4 times a year. Let's get straight into it with something I forgot last edition - details of a new Tracker. Yes, a new tracker has emerged that is heads above Noisetracker 1.0. Yes we may have a lot of better trackers around such as the rest of the Noisetracker series (1.1 and 1.2) and the Startrekkers. However, these trackers went away from being keyboard friendly and the file formats were not compatible enough for AMOS.

But now we have PROTRACKER 1.1b. This is now the tracker to go for because it is fully compatible with Noisetracker 1.0, has the same keyboard commands but most importantly, it is compatible with the Soundtracker2_1 converter program in AMOS. It also boasts many incredibly powerful tools such as:

SAMPLER - A lot of trackers seem to have these now but this one is nearly gadget for gadget with Audiomaster. Samples can be manipulated with ECHO, MOD, UP & DOWN sampling, and much more.

PREFS - Colours, file paths, multitasking and much more can be set in this area. You can even change the colour and look of your VU Bars.These prefs can then be saved to disk becoming the defaults upon loading.

This is just a small sample of what's available. Where can you get it? Well, a music disk of all the examples in the Trackin' Articles, some example tunes and PROTRACKER V1.1b will be available on disk AA-??. I encourage you to order this disk as it will help you through the more complicated techniques in becoming editions.

FINAL EFFECTS COMMANDS

Here are the last of the basic set of effects commands.

3 - Tuning Slide. This allows you to slide automatically to the next note: e.g..

```
A#301000   ;Plays note A#3 with sample 1
---00000
---00305   ;Begins to slide towards A-3 with a speed
A -301305  ;of 5 and then reaches A-3
---00000
---00000
G-301000   ;Plays note G-3
---00000
A -301305  ;And slide to A-3;
---00000
```

This command is useful for puting expression and variation into your melody and tunes.

Another technique to add expression to a tune is to simulate an echo by playing a tune and in it's sister channel (0 & 3 are sister channels - 1 & 2 are sister channels) copy the same tune at a lower volume and with an offset of one note like so:

Chn 0	Chn 1	Chn 2	Chn3
	A#301000	---00000	
	---00000	A#301C15	
	---00000	---00000	
	A - 301000	---00000	
	---00000	A- 301C15	
	---00000	---00000	
	G - 3 01000	---00000	
	---00000	G- 301C15	
	A - 3 01000	---00000	
	---00000	A -301C15	

OVERALL LIST OF BASIC COMMANDS

0 - Arpeggio (synthetic chords) or plain notes
1 - Slide note up
2 - Slide note down
3 - Slide to next note
4 - Vibrato (wobble note effect)
A - Volume slide
B - Break to pattern **
C - Sample volume
D - End current Pattern
E - Filter on/off (not for Amiga 1000's)
F - Tempo (speed setting)

GLOSSARY

Below is a glossary for all terms used in this article series and Soundtracking...

OCTAVE	- Set of 8 notes from C to C
CHANNEL	- Window holding notes. Four channels are at the bottom half of the Tracker
VOICE	- Same as CHANNEL
SAMPLE	- Piece of sound that the tracker plays when it reads notes in the channels
INSTRUMENT	- Same as SAMPLE
TRACK	- Set of notes in one channel
PATTERN	- Set of notes in all four channels
PLAY	- Play song from current selected position onwards
PAT.PLAY	- Play pattern over and over
RECORD	- Play pattern over and over and accept any new note inputs
MODULE	- Complete song ready to port to AMOS
SONG	- Song data only. NO INSTRUMENTS. This save config is not to be used with AMOS!

LAST WORD

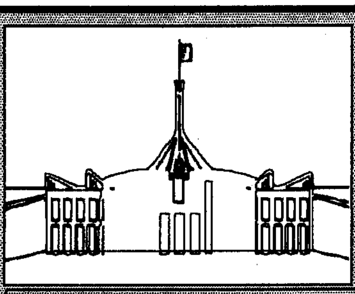
Next issue I will be doing a thorough review of Protracker 1.1b - it's functions, tools. I will also be giving details on new music extensions for AMOS as they become available. I will be going through the examples on the Tracker tutorial disk (check order page) as well as listing more techniques and ideas for those on Noisetracker1.0 and those without the disk. However I do strongly advise people using this series to purchase the disk from the library as it will benefit you in further issues.

Francois is planning a new music extension for AMOS soon. Apparently, it is to be based around Protracker 1.1b, so as it becomes available I will be doing tutorials on the supported new effects commands already in Protracker.

(He is also making it compatible with MED as well! - ED)

Finally, as usual, the Tracker Helpline is open 9.00pm - 9.30pm Tuesday to Thursday nights on (02) 545-3190. I will help you with any enquires or problems.
Till then.. Keep on Trackin'

Sausage



ACT AMOS
SPECIAL INTEREST GROUP

If you live in or around the ACT, then this is the event you have been praying for?? The Canberra Amiga Users Society Inc (CAUSe) is/has starting/ed an AMOS Special Interest Group in amongst all their other SIG's.

There is a large number of AMOS users down in the ACT so you would be mad to miss out on the first official meeting, which is to take place very soon!

The AMOS SIG will have meetings monthly and will allow you to meet with other people who have the same programming interests, namely AMOS. They have already had their first meeting and they ended up with more people than could fit in Bernies House! What or who is a Bernie? Well I had better tell you as it was mostly all his idea anyway. Bernard Weimers, Bernie to his mates, decided that AMOS user should be able to get together and discuss programming techniques and problems. If you want more information then call Bernie after hours on (06) 241 6737 and he will tell you when the first official meeting is to take place. Who knows, I may even make an incognito appearance at one of the meetings - just to see what's going on.

CAUSe has a yearly membership fee of $20.00 and have monthly meetings at 8pm on the second Thursday of each month in either the Chifly Room or the Auditorium at the Canberra Worker's Club. If you wish to get more details about CAUSe itself, then contact Paul Martin on (06) 253 2121. You would of course have to join CAUSe to attend the AMOS meetings. But you can enquire about all that at one of the meetings. So take my advice and JOIN! It may just save your sanity!

QUICKIE-1: IF YOU ARE ANNOYED BY THE LEADING SPACE AT THE BEGINNING OF A NUMBER AFTER BEING CONVERTED TO A STRING USING THE STR$( ) FUNCTION THEN TRY THE FOLLOWING INSTEAD!

N$=STR$(N)-" "

SIMULATING A STRING GADGET WITH AMOS - BY TEX

One thing that AMOS lacks is Intuitions String Gadgets, A String Gadget allows you to enter text/numbers easily with a bit of simple editing as well. You are able to delete part of the text, delete all text easily with one key press and go back through the text making changes. When setting up String Gadgets you were able to specify whether the text was to be Centred or Left Justified, you could also specify how many characters were allowed to be entered. C and ML Programmers can easily set up these gadgets, but in AMOS we really don't have anything that really comes close. Well I have written a pretty close version of the String Gadget, it is not quite as flash as the real thing but it does the job quite welll

This small routine does have the following features:

AUTO Centre Justification Of Text.
Right AMIGA X Clearing Of All Text Entered.
Deleting Of Text Using DEL Key.
Set Maximum Number Of Characters To Be Input.
Can Be Set To Only Accept Numeric Or ALPHA Input.
Can Use ANY Size Of AMIGA FONT.

This routine is ideal for entering HighScore Names or just about anywhere you feel you need it. Because of it's versatility, you just simply call it over and over again.

To use it you must first set up a few things, Firstly you must set a Font. In our case we use the internal Fonts, but you can of course use any AMIGA Disk Fonts. Now that we have a font set up, then we can call our Procedure with a call like on line 3. We are passing quite a few parameters to our Input Procedure, in fact we are passing 7! Let's cover each one from left to right.

200 (SX) =Starting X Co-Ordinate. This is the MIDDLE of your Input Text.
100 (SY) =Y Co-Ordinate. This is the Y Distance From the top of the screen.
6 (ML) =Maximum Number of Characters to be Input.
0 (WC) =Clearing colour. Usually the ink colour of the text's background.
5 (TC) =Text Colour. The colour your text will be printed in.
6 (FH) =Font Height. This is the MAX height of the selected Font.
1 (M) =Input Mode. 1= TEXT Only. 2= NUMBERS Only.

OK now have a look at the routine below. Then we will go into some detail about it.

```
1) Get Rom Fonts : Set Font 1
2) Cls 0
3) IP[200,100,6,0,5,6,1]
4) Procedure IP[SX,SY,ML,WC,TC,FH,M]
5)   Gr Writing 0 : MX=Text Length(Space$(ML))
6)   Do
7)     IP$=""
8)     V1:
9)     Repeat : I$=Inkey$ : SC=Key Shift : Until I$<>""
10)      I$=Upper$(I$) : I=Asc(I$) : If(SC=128) and(I$="X")
Then IP$="" : Gosub V3 : Goto V1
11)      Exit If Len(IP$)>0 and I=13
12)      If Len(IP$)>=ML and I<>8 Then Bell : Goto V1
13)      If Len(IP$)>0 and I=8 Then Goto V2
14)      If M=1 and I<65 or I>90 Then Goto V1 Else Goto V4
15)      If M=2 and I<48 or I>57 then Goto V1
16)      V4:
17)        IP$=IP$+I$ : Gosub V3 : Ink TC : TX=Text
Length(IP$) : TX=TX/2 : Text SX-TX,SY,IP$ : I$="" : Goto V1
18)      V2:
19)        IP$=Left$((IP$),Len(IP$)-1) : Gosub V3 : Ink TC :
TX=Text Length(IP$) : TX=TX/2 : Text SX-TX,SY,IP$ : Goto
V1
20)      V3:
21)      Ink WC : Bar SX-(MX/2),SY-FH To SX+(MX/2),SY
22)      Return
23)    Loop
24) End Proc
```

Line 4 of course is the start of the procedure and reads in the values being passed to it into the variables SX,SY,ML,WC,TC,FH and M.

Line 5 sets Graphics Writing Mode to 0 and we then calculate the Maximum length that the input could possibly be. This value is put into MX and is used for wiping the text when deleted and added to.

Line 6 starts the Input Loop.

Line 7 sets the Input Variable IP$ to a null string "".

Line 8 is simply a label.

Line 9 is our Repeat-Until loop that Waits for a key to be pressed, we are looking for two different kinds of input. I$=INKEY$ is looking for standard keyboard input. SC=KEY SHIFT is looking for the Right AMIGA Key. This line repeats until I$<>"".

Line 10 converts I$ to Uppercase and then sets I to the ASCII code of I$. We then check to see if SC=128 (Right AMIGA Key) and I$="X", this means that the user has hit Right AMIGA X, if this is so then we set IP$ to "", branch to V3 to clear the gadget and return back to V1 to start over.

Line 11 Exits the DO LOOP if the Length of IP$ is greater than 0 and I=13. (I=13 means the user has hit Return.)

Line 12 checks firstly to see if the length of IP$ is equal to or is greater than the Maximum length set in ML and then also checks to see if we have not hit the delete key. If this is not found to be true then we do not allow any more characters to be entered and we make a sound using the Bell command, we then goto V1. The reason we check for the DELete Key when IP is at it's maximum length is because if the user is at the Maximum length, then you would not be able to Backspace to change the entered text.

Line 13 makes the check to see if the length of IP$>0 and I=8, this means that we have got text in out IP$ and we have hit DEL. We then jump to V2 to perform our DELeting.

Line 14 checks to see what "MODE" we are inputing in, M=1 means that we are inputing only Text, we then check to see if we have input a valid character in the ASCII range 65-90. If this is found to be out of the range then we jump back to V1 else we jump to V4 to process the Input further.

Line 15 is similar to line 14, except that this time we are checking for Numeric entries only.

Line 16 is just a label.

Line 17 Adds the new character (I$) to our string (IP$), we then Gosub to V3 to clear our text on the screen. We then change the ink to that of the set text colour. We then calculate the new length in pixels of the new IP$ and put the result into TX, TX is then divided into 2 and we then use the TEXT command to put the new text onto the screen nicely centered. We then clear I$ and go back to the beginning to wait for a new input.

Line 18 is yet another label.

Line 19 is the routine which looks after DELeting a character out of our text. This is done using the LEFT$ function by specifying one less character than there is in the string. As with line 17 we then clear the text area, change inks and put the new text on the screen, returning to the beginning again.

Line 20 is another label.

Line 21 is our text clearing routine, we first set the ink to the Clearing Colour which is held in WC. Then we draw a bar to wipe the maximum area that our text can possibly cover.

Line 22 then simply RETURNS back to where it was called from.

Line 23 is the LOOP statement in our DO LOOP.

Line 24 of course signifies the end of our procedure.

If you want to read the result returned then it would be a good idea to have IP$ defined as a GLOBAL variable. I'm sure that this routine can be improved but it workes fine the way I like. If you do improve it then please send the improvements to me and I will put them in the next newsletter.

HIGH SCORE TABLE ROUTINE - 1

One thing that is usually quite boring in most games that I have seen is the HIGHSCORE TABLE, or in a lot of cases - the lack of one. High score tables can add just the right touch to a game, and giving the player the incentive to try harder to be able to beat the "High Score"! I have seen the following effect in a number of commercial games recently and I was reasonably impressed with it. I won't tell you what it looks like until you have typed it in for your selves. But I will tell you this, it works better if you have a LARGE Font at hand to make the text stand out and look more impressive.

CONTINUED ON PAGE 4 ⟶

Something around say 20-30 points.

Type this in, save it before running it, find a disk that has some nice fonts on it, and then make sure that you have ASSIGNED FONTS: to that disk and then run it.

```
1) Dim HI$(10),HI(10)
2) Global HI$(),HI()
3) HI$(1)="TEST1" : HI(1)=453232
4) HI$(2)="TEST33221" : HI(2)=453232
5) HI$(3)="TE1" : HI(3)=453232
6) HI$(4)="TE13ST1" : HI(4)=453232
7) HI$(5)="TE3ST1" : HI(5)=453232
8) HI$(6)="TE3131ST1" : HI(6)=453232
9) HI$(7)="TE33ST1" : HI(7)=453232
10) HI$(8)="T1EST1" : HI(8)=4232
11) HI$(9)="T1" : HI(9)=4532432
12) HI$(10)="T3EST1" : HI(10)=44253232
13) '
14) HIGHSCORES[10,1,20,50,10,200]
15) Procedure HIGHSCORES[NN,SF,FH,DH,DS,YP]
16)    Screen Open 7,320,NN*(FH+10),16,Lowres : Y=FH+2 :
Screen Hide 7
17)    Get Disc Fonts : Cls 0 : Set Font SF : Gr Writing 0 : Ink
1 : Colour 1,$FFF
18)    For X=1 To NN
19)       HS$=HI$(X)+" --"+Str$(HI(X)) : TX=Text Length(HS$)
: TX=TX/2
20)       Text 160-TX,Y,HS$ : Y=Y+FH+2
21)    Next X
22)    Screen Display 7,,YP,,DH : SH=NN*(FH+2)-(DH-10)
23)    Channel 14 To Screen Offset 7 : Screen To Front 7 :
Screen Show 7
24)    A$="L: M 0,"+Mid$(Str$(SH),2)+","+Mid$(Str$(DS),2)+";
M 0,-"+Mid$(Str$(SH),2)+","+Mid$(Str$(DS),2)+"; J L"
25)    Amal 14,A$ : Amal On 14
26)    Do : Loop
27) End Proc
```

If you had a reasonably large font to work with then you should of had a pleasing result. If it was a bit strange then I will explain all about the procedure.

When using this procedure there are a number of guidelines to follow...

1) You MUST have lines 1 & 2 in your program, the procedure needs those variables to work.

2) It runs under interrupt using channel 14 and screen 7, when starting your game stop the Amal Channel and close Screen 7. Then after you have entered the new name and your title page is displayed then just call the procedure and it will look after itself.

You could quite easily use the String Gadget procedure for entering High Score Table Names! Now for a bit of an explanation about how it works.

LINE 1) DIMensions 2 Variables, HI$() and HI(), these variables actually contain the High Score Table data, HI$() holding the names and HI() holding the actual scores. In this example I have DIMensioned them to 10, if you want more names in your High Score Table then DIMension it higher. Keep in mind when doing so, the more names you have the smaller the font should be, otherwise you could find yourself crashing the system when the procedure tries to open a screen 6000 high! or some other lesser height screen but none the less it could be enormous.

LINE 2) See above.

LINE 3-12) are just dummy test data, you don't need them after you have tested the procedure, but it is a good idea to have some defaults in your game but with some more exciting names than I have here!

LINE 13) Just ignore line 13 it's just to complicated to explain with such little space as we have here.

LINE14) Is our CALL to the procedure, passing 7 parameters as we do so, an explanation of each parameter follows. (Refer to line 15 for Variable names)

NN = Number of names in the High Score Table.
SF = Number of the FONT being used.
FH = The actual Font's Height in Pixels.
DH = The DISPLAYHEIGHT of the High Score Table, or how much is shown at once.
DS = DISPLAYSCROLLSPEED, the speed at which the High Score Table scrolls.
YP = High Score Tables Y Position from the top of the screen.

LINE 16) Opens screen 7 for our High Score Table, it does this by calculating the minimum Y Height it will need based on the height of the selected Font and the number of entries in the High Score Table. We also define the variable Y with the following formula...Y=FH+2. Meaning Y = FONT HEIGHT + 2, this gives us a starting position for our first entry in the table. We next hide the screen so we don't see the table creation going on.

LINE 17) Makes a call to Get Disc Fonts, clears the screen to black, sets the font to the requested font (SF) and then sets Graphics Writing Mode to 0. We then select INK 1 as our text colour and in our case we then set colour 1 to pure white. But don't limit yourself to just one boring colour, set up a palette which goes from light to dark and as you are printing your names from 1 to the end change colours!

LINE 18) Is a For Next Loop going from 1 to the total number of names held in our table, we get this information from the variable NN.

LINE 19) We do quite a bit on this line that is important, we first create out dummy text by joining the Name from HI$(x) with " --" and then add the score to finish off from HI(x). This is put into a string variable called HS$. We also now work out how wide the string is in pixels and then divide it by 2 to get our centring working correctly.

LINE 20) We now put our dummy name onto the screen using the text command, next we add the Fonts Height to Y and add 2 to be sure there will be a gap between each name in the table.

LINE 21) Completes the For Next Loop.

LINE 22) Is where we start to get ready to make our High Score Table ready for display. We first change the Screens Display paramaters to suit what we need by using two of the variables that were passed to this procedure with those being YP (Screen's Y Position) and DH (Screens Display Height). We then create a new variable which will hold the total distance we are going to scroll our table. We do this in much the same way we calculated the total screen height earlier, except this time we have to minus a bit off so we don't corrupt the display.

LINE 23) Assigns channel 14 to our screens offset, (See page 198). We then bring our screen to the front and make it visable.

LINE 24) On this line we generate our AMAL string from all our relevant Variables, this just moves the screens Y Offset up and down continuously.

LINE 25) We now turn our AMAL channel on and activate it.

LINE 26) Is only here to allow you to see it working without going to direct mode. Remove it once you are happy with it.

LINE 27) Ends our procedure.

So there you have it, an idea for a high score table. It can be improved and added onto, but that's another story. One thing you could do though is to create a fancy Alphabet as an IFF, (Just like in the scrolling text demo that came with AMOS) and screen copy them to your output screen. This would make it look really flash indeed! If you have an idea for a High Score table that is flash then send it in and if we have the room then I will print it. Next edition we will have another flash high score routine as well as routines for loading and saving the high scores to disk!

CHECKING TO SEE IF A DISK/HARD DRIVE IS WRITEPROTECTED USING THE DOSCALL FUNCTION

One problem that has popped up time and time again is when wanting to write data to a disk, it is usually always write protected or locked in the case of a hard drive. This can be got around by AMOS's Error trapping but AMOS does not trap the error until after the horrible workbench requester has appeared. This can look very unprofessional!

But help is at hand, with a bit of experimenting with the DOSCALL function, I have been able to check the write protect status with no problems. This routine works on any valid drive that can exist on the Amiga. I have also added a bit more to the routine to show relevant data about the disk/drive we are examining. Using these routines will also allow you to check that there is enough free space on the disk before you write to it! Yes I know we already have such a command (DFREE) but what the hell, it does not give you as much information as this does.

All this takes is just 2 calls to functions contained within the DOS.LIBRARY, the first call is to....

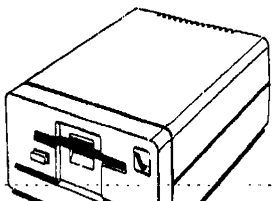
LOCK (Offset -84)
This makes sure that the disk/drive we are wanting to access is available, if it is then a structure is created for it in memory.

The second call is to....

INFO (Offset -114)
This is the function that does all the work, it will return the needed information into the variable that we created - D$. All we then need to do is to look in memory for the relevant information.

For a full explanation of this short piece of code then see after the program itself, but here is the program.

```
1) Dreg(2)=-2
2) T$="dh0:"+Chr$($0)
3) A=Varptr(T$)
4) A=A+(A mod 4)
5) Dreg(1)=A
6) R=Doscall(-84)
7) Dreg(1)=R
8) D$=Space$(32)
9) DA=Varptr(D$)
10) DA=DA+(DA mod 4)
11) Dreg(2)=DA
12) RR=Doscall(-114)
13) WPS=Peek(DA+11)
14) TB=Leek(DA+12)
15) TBU=Leek(DA+16)
16) BPB=Leek(DA+20)
17) NSE=Leek(DA)
18) DN=Leek(DA+4)
19) Screen Open 0,640,256,2,Hires : Colour 1,$FFF
20) Print T$;
21) If WPS=80 Then Print " is Write Protected!"
22) If WPS=81 Then Print " is Being Validated!"
23) If WPS=82 Then Print " is Write Enabled!"
24) Print T$;" Has the following Statistics..."
25) Print "Total Number Of Soft Errors is";NSE
26) Print "It's Logical Drive Number is";DN
27) Print "Total Capacity is";TB;" Blocks or";TB*BPB;" Bytes"
28) Print "Total Blocks Used=";TBU;" - Total Bytes
Used=";TBU*BPB
29) Print "Total Blocks Free=";TB-TBU;" - Total Bytes Free=";
(TB*BPB)-(TBU*BPB)
30) Print "Bytes Per Block=";BPB
```



Line 1 puts the value of -2 into the 68000's Data Register 2. This value of -2 tells the LOCK Function that we wish to READ the drive as opposed to -1 if you wish to WRITE to it.

Line 2 declares T$ to contain the name of the Drive we wish to examine. It also adds a CHR$(0) to the end of the Drive name, this is vitally important, as DOS requires everything to end in a 0.

Line 3 loads the variable A with the START Address in memory of the variable T$.

Line 4 then makes sure that the Start Address of T$ is LONG WORD Aligned.

Line 5 then loads this address into the 68000's Data Register 1.

Line 6 then makes the call to the LOCK function while the returned result is placed into R.

Line 7 then places the value in R into the 68000's Data Register 1.

Line 8 then creates a String Variable called D$ which is 32 bytes long. This is the variable where the important information is placed.

Line 9 loads DA with the START Address in memory of D$.

Line 10 again makes sure that DA is Long Word Aligned!

Line 11 then loads DA into the 68000's Data Register 2

Line 12 calls the INFO Function with the result being returned into RR.

Line 13 commences the first of 6 Peek's and Leeks into the area of memory used by our variable D$. The first we get is the byte containing the Drive's Status and places it into the variable WPS. This is the one which contains the information about wether the drive is Locked/Write Protected, Validating or Write Enabled.

Line 14 Leeks the information about the Total Blocks that the drive is capable of storing. This is placed into TB.

Line 15 Leeks the information about the Total Blocks Used on the specified drive. This is placed into TBU.

Line 16 Leeks the information about how many Bytes make up 1 Block. This information varies from drive to drive. (Floppies-Hard Drives) This is placed into BPB.

Line 17 Leeks the information about the number of Soft Errors that are on the specified drive. This is placed into NSE.

Line 18 Leeks the Drives Logical Device Number. This is placed into DN.

Line 19 simply opens a Hires screen so that screen output is easier to read.

Line 20 prints the Drive Specified's name.

Lines 21 - 23 test WPS (Write Protect Status) and prints the relevant message.

Lines 24 - 30 are self explanatory.

Hope that this routine helps you when you are writing your programs. In the next issue, we will have another interesting article about using DOSCALL, this time it will be written by Evan Thomas who was responsible for the routine that retrieved the system date and time.

TEX

Using The Serial Extension

Neil's price for helping me with some of the small problems with the AMOS 1.3 Serial Extension commands was that I prepare an article for inclusion in the Newsletter, so here's my end of the bargain. Before continuing, I feel that I should explain that I lay absolutely no claim to being an expert in either AMOS or telecommunications (or anything else for that matter). If you read this article and think to yourself "this guy knows less than I do", well you're probably right, and I wouldn't be surprised!

"Welcome to the intriguing world of AMOS communications"

So begins the chapter in the "What's New in 1.3" booklet, and a slightly lukewarm welcome it turned out to be at first. A number of articles in past Newsletters have commented on a few configuration problems with serial communications and with a bit of 'bugginess' in the 1.3 Serial Extension. Like many of you, I suppose, when I first read through the booklet I was impressed with the simplicity of the commands and, in my innocence, entertained visions of churning out rivals to JR-Comm and GP-Term in pretty short order. Needless to say, I was having myself well and truly ON.

CONTINUED ON PAGE 6 ⟶

First stumbling block was the "Serial Bit" (as specified in the booklet) command. This is essential for any serial comms, particularly since AMOS is defaulted to the French Minitel Net comms protocol which is recognised by virtually nobody here. It doesn't work! When you type in the command the editor will treat it as two variable names, not as a command statement. In fact the booklet is misprinted and the actual syntax of the command is "Serial Bits". When entered this way, the command appears to work in precisely the way the booklet describes.

The "bugginess" of the Serial Extension was next on the list, and for this Neil had to send a couple of faxes to Francois (or Daisy?) for clarification. Without going into the why's and wherefore's of shipping AMOS with a Workbench which is incompatible with an AMOS extension, the problem has turned out to be the Serial.Device in the Devs directory of the Workbench. Simply, you will need the Serial Device from Workbench 1.3.3 or later. I have tried installing the 1.3.3 Serial Device on every Workbench I could get my hands on back to 1.2, and the AMOS serial commands have worked like a charm with all of them. Without the 1.3.3 Serial Device, the Serial Close command will not work, and so the Amiga will hang every second time you attempt to run a comms program (you can't reopen a serial port that is already open), or, if its your lucky day, you won't even get that far!

I have included some code which you may like to try out on a BBS or two. Some sample code was supplied to me by Neil, which I then stripped down to the bare essentials, for my own education. It should be fairly clearly remarked. It is basically a dumb terminal - all the program does is transmit everything you type to the modem, and print on the screen everything the modem receives as long as its a printable character. You may, therefore, have to brush up on some of the Hayes commands before you try it. Don't be surprised, of course, if you get some odd ANSI formatting and graphics codes on BBS's if you've been using JR-Comm or GP Term, since you need to program in you're own ANSI interpretation routines if you want to go in that direction, which shouldn't be any great drama. I don't like ANSI colour graphics much anyway, and so I haven't bothered to try. In any case, I think you will find that too many bitplanes will compromise throughput at higher baud rates.

With some BBS's you will need to program in a command to interpret the screen clearing ascii code, or add a Carriage Return when a Line Feed is received, although with Paragon, for one, there is not a problem.

I have not tried baud rates over 1200, because I only have a 1200 modem, but again, things should be reasonably straightforward, at least up to the really fast rates, or comparatively long receives, in which case you will probably have to extend the Serial Buffer ("Serial Buffer" command) or perhaps program some buffering of your own, as well as resorting to some trickiness in order to maintain throughput.

In further programming I have used the AMOS "Wait" command in preference to the Hayes commands for delays in automatic dialling sequences and the like. This allows me to actually test (albeit crudely) the speed of the modem commands and their execution and then optimize the program accordingly. So far I have been a bit surprised how slow my modem (Netcomm Pocket 1200) is to accept and act on commands.

To date I have not seriously tried to program an upload routine and so I will not comment on the "Serial Out" command. Again, it appears to be simple enough in the booklet (famous last words!). Certainly the download equivalent "Serial Get" command works quite happily, and I have been using it in preference to the "Serial Input$" command so that I can view any ANSI code characters, and so that things like Ack/Nak and Xon/Xoff will come through OK.

For binary uploads, I guess my initial attempt will be to set up a sort of bucket brigade by loading predetermined lengths of the appropriate file into two sections of memory, sending one block while topping up the other from the disk, and then swapping etc. I think I will probably use temporary AMOS Memory Banks to start with, just to try to get things working before doing things "properly". Of course this system may not be appropriate for multi user games, but for me it seems to be the simplest way to start, and this seems to be indicated in the booklet entry titled "Sending large strings." I know nothing about file transfer protocols, and would be happy to hear from anybody with access to the actual specifications for X,Y or Z Modem protocols.

BUGS

The above solutions notwithstanding, some apparent bugs have manifested themselves during my brief tussle with the Serial Extension. Some of them have the very unfortunate trait of only manifesting themselves when running programs as executables after compilation rather than as interpreted programs. Some of them have been revealed very late in the piece, so to speak, and have cost me quite a lot of time. I guess the moral is that you should never assume that what works as an interpretive program will work as an executable program - always test your programs in compiled form at regular intervals. Some of you may have found this out already like me - the hard way!

The apparent bugs which have caught me out so far are:-

(a) If you try to use the Set Buffer and Serial Buffer commands in the one program the compiler seems to crash during compilation.

(b) The window command 'Title Bottom "text" ' command seems to produce an error in my compiled program.

I haven't discounted the distinct possibility that the above problems are caused by my usage, not by AMOS, and if anybody can get around them, I would be grateful for some help.

If anybody wants to contact me, you can leave a message for me on Paragon BBS (02 5977477) addressed to "Mike Bancroft." I generally log two or three times a week.

Lastly, I know it's enormous, but I recommend the JR-Comm User Manual for a lot of interesting info, if you can be bothered with the hassle of formatting and printing it.

MIKE BELLAMY

My thanks to Mike for putting these words together for me on his experiences with the Serial Extension. You will by now know that all the problems in using the Serial Extension were not the fault of the Extension itself, but as we have said the fault of Commodores SERIAL.DEVICE driver. I have tested the commands using the 1.3.3 and 2.0 versions of the driver and they work without any problems. I have been able to log onto Predators BBS with the example code below without ony problems as well. To fix the problem simply get your original workbench and either use CLI or some directory utility like DiskMaster, OPUS or any other utility. Then simply copy the Serial.Device driver out of the DEVS directory on your workbench to the DEVS directory on your AMOS/PROGRAM disk.

We have received our first game which uses the Serial Extension, look in the PD newsletter for details. Below is the source code which allows you to log on to a BBS or a friends computer with ease, it's very simple and can easily be built on for a more advanced Terminal Package etc. In the next newsletter I will add A simple Phone Directory and Dialling routines to the code below.

```
1) Screen Open 0,640,256,2,Hires : Cls 0 : Colour 1,$FFF
2) Serial Open 0,0 : Rem open the serial port as 0
3) Serial Speed 0,2400 : Rem set Baud rate at 2400 Baud*
4) Serial Bits 0,8,1 : Rem set 8 data bits, 1 stop bit
5) Serial Parity 0,-1 : Rem set no parity
6) Do
7)    B$=Inkey$
8)    Exit If B$=Chr$(27) : Rem exit if we hit ESC
9)    If A$<>""
10)      Serial Send 0,B$ : Rem send users input
11)      Wait Len(B$) : Rem wait for all of b$ to be sent
12)   End If
13)   B$="" : Rem reset b$ to "" so we don't keep
sending it
14)   R=Serial Get(0) : Rem get input from modem
15)   If R=13 : Print : End If : Rem print return for a new
line
16)   If R>31 : Print Chr$(R); : End If : Rem only print
printable characters
17) Loop
18) Serial Close
```

Line 1 opens a Hi-Res screen so that you can display 80 characters wide, most BBS's require you to be able to show this width.

Line 2 Opens the Serial Port in exclusive mode to channel 0.

Line 3 Sets the speed of the channel opened to 2400 Baud, you can of course set this to any speed that your modem can handle.

Line 5 sets the parity to -1, with -1 being no parity.

Line 6 starts our main loop.

Line 7 reads any input from the keyboard into B$.

Line 8 checks to see if you have hit the Escape key, if you have then

CONTINUED ON PAGE 7 ⟶

we abort the loop, close the serial port and then exit the program.

Line 10 checks B$ to see if it contains anything, if it does then it sends it out the serial port.

Line 13 then sets B$ to a null string "" so that it does not continually send the same data and thus going into a continual loop.

Line 14 reads any incoming data from the Serial Port into B.

Line 15 checks to see if B = 13, if it is then we print a line and continue our loop. 13 is of course the Character code for return

Line 16 then checks the value of B to make sure that it is a printable character, if it is then it is printed with a semi colon after it so that the next character received is placed after it.

Line 17 marks the end of our loop.

Line 18 closes the Serial Port.

WORLD-WIDE SALES OF AMOS REACH..

Unbelieveable But TRUE!

World wide AMOS sales have passed the 5.5 Million Dollar mark! The total unit sales for the world stand at around 40,000 units! That's an impressive track record for a piece of software that is just over a year old, and is even more impressive when you remember that it is a programming language not just a game! German, American and French versions of AMOS have just recently gone on sale and are already taking off. Now with the addition of the Compiler and the 3D extension, AMOS is fast becoming THE most powerful language on the AMIGA today.

A QUICK TYPE-IN-AND-RUN-JUST-FOR-THE-HELL-OF-IT!

The following program I copied out of Issue 4 of the British Newsletter, they just released newsletter 4, and we are already up to newsletter 9! Our newsletters have much more in them and are twice the size, and if I may quote a certain frenchman, "Your Newsletters are far superior than the British ones." What can I say with a complement like that? Anyway, this program shows you how to use multiple bullets in your programs as well as showing you how things can be affected while under gravity. With some refining it could be made quite flash. It would look quite good as an explosion of your space ship when hit with bits flying off in all directions and then falling to the ground, picking up speed as they go. Well you will see what I mean once you have run it. Most of the commands are pretty much self explanatory and since I did not write it, AAron Fothergill did then I'm sorry there are no explanations on how it works.

Type it in and have some fun experimenting with it, as it stands it would make a nice fireworks display, but you really would have to COMPILE it to get any real speed.

```
1) Dim BX(20),BY(20),BDY(20),BS(20)
2) Curs Off : Flash Off : Cls 0 : Paper 0 : Pen 1 : Ink 2,2
3) Bar 0,0 To 7,15 : Hide On
4) Get Bob 1,0,0 To 16,16 : Hot Spot 1,4,0
5) For A=0 To 9 : Cls 0 : Locate 0,0 : Pen A+2 : Print "."
6) Get Bob 2+A,0,0 To 16,8 : Hot Spot 2+A,4,4 : Next A
7) X=160 : Y=180
8) Cls 0 : Double Buffer : Colour 2,$FFF : Colour 1,$FF0
9) Fade1,$0,$FFF,$F00,$F0,$F,$FF0,$FF,$F0F,$F80,
$8F0,$F8,$8F,$80F,$F08
10) While Mouse Key<2
11)   Bob 1,X,Y,1
12)   If NB>0 Then Gosub BULLETS
13)   GTG=1-GTG : X=X Screen(X Mouse)
14)  If Mouse Key=1 Then Gosub FREBULLET
15) Wend : End
16) BULLETS:
17) For A=0 To NB-1
18)   Bob 2+A,BX(A),BY(A),2+BS(A)
19)   BX(A)=BX(A)+1 : BY(A)=BY(A)+BDY(A)
20)   If GTG=0 Then BDY(A)=Min(8,BDY(A)+1)
21) Next A
22) NB2=NB
23) For A=NB-1 To 0 Step -1
24)   If BY(A)>199 or BX(A)<0 or BX(A)>319
25)     Bob Off 1+NB2
26)     Swap BX(A),BX(NB2-1) : Swap BY(A),BY(NB2-1)
27)     Swap BDY(A),BDY(NB2-1) : Swap BS(A),BS(NB2-1)
28)     Dec NB2
29)   End If
30) Next A
31) NB=NB2
32) Return
33) FREBULLET:
34) If NB<20
35)   BX(NB)=X : BY(NB)=Y
36)   BS(NB)=Rnd(9)
37)   BDY(NB)=Rnd(2)-12
38)   Inc NB
39) End If
40) Return
```

The AMAL Series
By SAUSAGE

Well, the Amal editor tutorial is back after a month's absence. The reason for this is that there simply wasn't enough space. Also, a correction... The AMOS system does not destroy header blocks when Amal Files are loaded in. Therefore an Amal file can be "ripped" from a program by saving it out and loading it into the Amal editor.

This time round we are looking at the Amal_example_file.abk on the extras disk. So load up AMOS and start the Amal Editor (A.E. from now on). Load the example file and put in the data disk when prompted. Run the program. We are going to alter it so that we have a custom attack wave.

Go to channel 03 and clear the channel by placing the cursor line 1 and pressing return switch to channel 04 and back to 03 again. The channel will have disappeared. Repeat this for channels 04,05,06 & 07. Back to channel 03 again and enter the following:

```
For R0=1 To 10;   } This makes the program
Next R0;          } wait for 10 VBL's.
L X=0;            } Position X at far left.
L Y=0;            } Position Y at far top.
PL 1;             } Play movement 1
M 0,0,0;          } Re-initialise play buffer
                    for further play movements.
```

Copy this channel into 04,05,06 and 07.

We have to change the amount of waiting for each channel. Therefore, go to channel 04, line 1 and change the 10 to a 20. Go to Channel 05 and change the 10 to a 30. In Ch 06, make it a 40. And in Ch 07, make it a 50. Press F5 to go to the play editor. Select 01 and place the mouse pointer in the top left hand of the screen. Press F1, then return and move your mouse around. Not too quick or slowly but at a steady pace. Click left mouse when finished.

Press Esc to go back to the main menu and press F1 to run. You should have a custom attack wave moving smoothly around. Looks good huh? Remember to put in a M 0;0;0; after each PL in your own programs.

JOYSTICK ROUTINES

A lot of people when writing programs, seem to use basic to control joystick movement. Amal is a better medium because of the machine code speed of detection. Calling a standard joystick position is done by using:

```
If J1=? Jump
```

J1 is the variable that contains the value of joystick port 1. The full list of possible values for J1 are as follows:

CONTINUED ON PAGE 8 →

1 - UP	17 - UP/FIRE
2 - DOWN	18 - DOWN/FIRE
4 - LEFT	20 - LEFT/FIRE
8 - RIGHT	24 - RIGHT/FIRE
5 - UP/LEFT	21 - UP/LEFT/FIRE
9 - UP/RIGHT	25 - UP/RIGHT/FIRE
6 - DOWN/LEFT	22 - DOWN/LEFT/FIRE
10 - DOWN/RIGHT	26 - DOWN/RIGHT/FIRE
16 - FIRE	

So to create a routine to detect the joystick in four directions:

```
A:
If J1=1 Jump B;  }
If J1=2 Jump C;  } Labels B,C,D,E are routines
If J1=4 Jump D;  } to move an object, etc.
If J1=8 Jump E;  }
Jump A;
B: Let Y = Y - 1; Jump A;
C: Let Y = Y + 1; Jump A;
D: Let X = X - 1; Jump A;
E: Let X = X + 1; Jump A;
```

Make sure that you type this in exactly as it is here, complete with CAPITALS, as the proper Syntax is required when using AMAL. AMOS can not correct these problems because all AMAL strings are in inverted commas. In this simple routine there is no checking for the object going out of view, you can simply do this by checking to see if X or Y is greater than or less than the Maximum and Minimum Values for each one respectively.

Only a short one this time but the techniques here should be enough to spark off some decent Shoot'em'ups. If you write something you consider to be half decent, why not send it in?

Cheers till the next issue.

Sausage.

Writing Adventures In AMOS
The END!

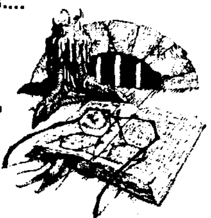
Well we have come to the last installment in this LONG series on writing Adventure games. Hooray, some of you may shout! But I can only say this in my defence, At least it was thorough! We have covered quite a few subjects in this series and there are only a couple more left to cover, most of these are implementing problems into your adventure games. I am a bit disappointed that no body bothered to write and suggest a new problem to put in the article. But we have received our second Adventure game, which is now in the PD library.

One of the topics I promised to cover in this article was the implementing of pictures and sound effects into your games. Well I did the picture one, but since the theory behind the sound one is the same (Bar the AMAL) then you should be able to work it out. OK we have got to the stage in our adventure where we can move around, pick things up and even solve a few simple puzzles, we implemented large descriptions and now we want to make it LOOK flasher. So below is a simple picture routine, it can be implemented pretty simply into your game with the edition of a few variables and a single line in the look routine.

First goto the very top of your program and DIM PIC$(CL), this sets up an array for the total number of locations that are in our adventure. Next, Declare PIC$() as a global variable, while your at it declare PICON as a global as well. On the line that reads something like NL=0: SPIN=1 .... add PICON=1. Now go to the LOOK procedure and at the very end before the End Proc add the line If PICON=1 Then Proc _SHOWPIC

Now to allow our Picture routine to work, we must add the following lines into our SETUP procedure, go to the very last line before the End Proc and add the following lines....

```
If PICON=1
  Limit Mouse 0,20 To 10,280
  Channel 1 To Screen Display 3
  A$="L:Let Y = YM-256; L RA=Y; J L"
  Amal 1,A$
  Amal On
End If
```



Now there are only two more things to do, type in the actual procedure below and then draw your pictures, pack them down, put them all into the same directory, create a bunch of data statements that contain the name of the picture that corresponds to the location. These must then be read into the array PIC$() with a For Next Loop. If you don't have a picture for every location then leave the data statement blank like Data "", because the routine knows that if it strikes a blank entry then there is no pic available for that location. OK, now type away and then I will give a brief explanation.

```
1) Procedure _SHOWPIC
2)   If PIC$(CL) ="" Then Pop Proc Else Amal Freeze 1
3)   If Amreg(0)>-236 Then Gosub RAISEPIC
4)   NEWPIC:
5)    Load PIC$(CL),15 : Unpack 15 To 3 : Screen Display 3,,-236,,
6)   LOWERPIC:
7)   Channel 2 To Screen Display 3 : Amal 2,"M 0,260,125" : Amal On 2
8)   Wait 1 : Repeat : Until Chanmv(2)=0 : Y Mouse=280 : Amal Off 2
9)   Goto SHW99
10)   RAISEPIC:
11)   Wait 1
12)    Y=Amreg(0) : DY=-236-Y/2 : Channel 2 To Screen Display 3
13)    DY$=Mid$(Str$(DY),2) : DYS$=Mid$(Str$(DY),3) : A$="M 0,-"+DY$+","+DYS$
14)    Amal 2,A$ : Amal On 2 : Wait 2 : Do : Exit If Chanmv(2)=0 : Loop : Return
15)   SHW99:
16)   Channel 1 To Screen Display 3
17)   A$="L:Let Y = YM-256; L RA=Y; J L"
18)   Amal 1,A$
19)   Amal On 1
20)   Screen 0
21) End Proc
```

The whole routine uses an AMAL string which keeps the screen at the same level as the mouse pointer, the beauty of it all is that it is running under interrupt, so you don't have to do anything yourself!

Line 1 marks the beginning of the procedure.

Line 2 checks the array PIC$() for a null string, if it finds one then that means that there is no picture for this location and we leave the procedure using the POP PROC command. However if it does find something in the string, then we freeze the Amal channel that controls the screen following the mouse.

Line 3 then checks to see if the picture is exposed, if it is then we branch to a routine which raises the picture for us.

Line 4 starts the NEWPIC routine.

Line 5 loads the file found in the array PIC$(), it is unpacked from bank 15 onto screen3. The screen is immediately positioned out of view, ready to be lowered into view.

Line 6 starts the LOWERPIC routine.

Line 7 assigns Amal channel 2 to screen number 3, we then declare a movement for that screen and turn it on.

Line 8 waits for a short period of time to give the AMAL channel time to start, we then do a Repeat until CHANMV(2)=0. This means that we are waiting for the screen to stop moving before we continue with our program. We then set the mouse pointers position to the same as the bottom of the pictures screen. We then turn AMAL Channel 2 off, just to be sure!

Line 9 then sends the program off to label SHW99, which exits the procedure.

Line 10 starts the RAISEPIC routine.

Line 11 waits for a short period of time, I can't remember why but I must have had a reason for doing it at the time I wrote this.

Line 12 retrieves the Y position of the mouse from the AMAL Registers, we then do some calculations for speed etc. Lastly we set up AMAL channel 2 to screen 3.

Line 13 sets up our AMAL movement from previously defined variables (Line 12).

Line 14 starts the AMAL and then waits for the screen to stop moving, just as we have done before except this time I use a DO LOOP, who knows why I changed from a Repeat Until! We then

CONTINUED ON PAGE 9 ⟶

RETURN to where we called this routine.

Line 15 starts the label SHW99.

Lines 16-19 restart the AMAL routine for the screen to follow the mouse again.

Line 20 makes sure that we are using Screen 0 once again, otherwise all text would be printed to the picture screen!

So that's it for the Picture routine, I probably could be improved but that's up to you.

An old favourite puzzle would have to be the crack that you can't take something through with you, I think nearly every adventure game has had a puzzle of this type somewhere in the game. So we will implement such a problem in our game, we will locate it in our maze at location 24. Because I don't know what your location descriptions say, you will just have to adjust the description for location 24 to reflect the fact that there is a narrow crack leading north. Or you can add a bit of a twist to the plot by adding a bit of fantasy to the game, it's quite OK to do this, you are able to see and do the impossible in adventure games. That's what they are all about! Firstly we will give a location description to paint a picture of the new location, it could go something like this...

You are in a smallish chamber and you have to stoop so as not to hit your head on the very low ceiling, there are exits leading off in numerous directions. The ground here seems to be much smoother than the ground that you have been traveling over and upon close inspection turns out to be composed of a greenish marble. To the North there is a narrowish passage leading into the gloom, on either side of this passage you can see two faces carved into the rock itself, they appear to be looking at each other.

OK, we now know where we stand. What we have to first decide is what is going to be the offending object, what is actually going to stop you from going through. Usually it is a large object, but there is no reason why it can't be something as simple and as small as a ring. Now we have to go and put the code in to check for the three important factors that make up our problem.

1) Correct Location.
2) Correct Direction Chosen.
3) Offending Object Being Carried.

Open the GO Procedure and before the line that reads something like If RS=1 Then Gosub G1. Enter the following....

```
If CL=24 and NO=1 and OB#(34,1)=-1
    Print "As you start to move North, the Carved Faces begin to glow and the earth begins to shake! Both faces  open their eyes and stare at you with horror, a sneer creeps across their lips and just before you are"
    Print "able to leave the chamber, with a loud crash the walls with the faces smash together blocking your movement in that direction. A voice booms loudly - Your not passing these two walls carrying THAT object! You jump back with fright and slowly the walls return to their original positions.": Goto G99
End If
```

Now that is a bit dramatic, but it also stops you from moving in the direction specified. Let's look at the test that does the checking.

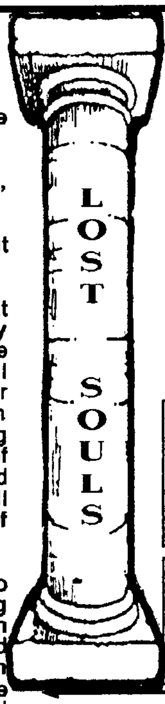
```
If CL=24 and NO=1 and OB#(34,1)=-1
```

Firstly we check what location we are at (CL=24) we then check to see if we want to go NORTH (NO=1) and that we are carrying the offending object (OB#(34,1)=-1) with 34 being the object we are checking on. Now we can also add the following line as well AFTER what we have just typed in.

```
IF CL=24 And NO=1 then Print "As you pass by the stone faces, you hear a hollow laugh and you could swear that the eyes followed you!"
```

There you have it, another puzzle. Remember that you must supply some way to get the object past this obstacle, that is of course providing that you want this to be done. This can be quite simple or it can be a major puzzle in itself!

Well that wraps up our short series on how to start writing adventure games! I hope that the people that have been following this series now know a bit more about writing adventure games and are now able to write their own. If you want my source, complete with accessories then order disk AA100, also includes another adventure that was submitted recently! Bye for now. TEX.



Column Of Lost Souls

Welcome to the new section in the Newsletter, The Lost Souls Column. As we said in the previous Newsletter, this is where you can write in with your Name, Address, Age and Interests and hopefully someone will answer with similar interests. We have a few Lost Souls to choose from this time and hopefully we will have more in the next one. Our only AMOS user in New Caledonia has written in and is desperately looking for someone to write to him, must be pretty lonely being the only person in an entire country with AMOS. So if you feel inclined, then put pen to paper and write away. If I missed you this time then maybe next time.

Scott Le Brun
58 Brentwood Drive
Wantirna, VIC, 3152
CO-Write Games
Music or Business Programs.

Bergade Xavier
12 Rue Lareign Ere Ouetto
Magenta, Noumea
New Caledonia, 2375
Will Answer All Letters!

Nick Ashman
11 Weise St
Oakey, Qld, 4401
Demos, Games.
All Letters Answered!

Neil McKnight: 28 Yrs
551 Albany Hwy
Albany, WA, 6330
Programming, Graphics
Games & Music

It's A Dogs Life!

+ François

Hi all Australian dogs!
Daisy speaking. After a few month of holidays, I have made a new Bone Demo. Bone Demo 5, the Return of the Come Back II. What a title. Have you ever seen a picture moving with a Sinus Wave? Today's demo does the same, not only in the X but also in the Y. Like a flag moving in the wind. This demo moves a whole 200 line picture every 1/50th of a second! More than that, it asks for very little processing time! I must really admit that I'm not too stupid for a 8 years dog.

Here we go for the listing.

```
'---------------------------------
' Bone-Demo number 5
'
' "The return of the Come-Back II"
'
' By Daisy Lionet
' (c) Bone-Productions 1991
'---------------------------------
Screen Close 0
Close Editor
Close Workbench
'
' Defintion of movement
'
NPAS=16
NSCREEN=NPAS : SX=320 : SY=200
AMPX1#=6 : AMPY1#=0.05
AMPY2#=6 : AMPX2#=0.04
'
' Memory reservations
'
' Make sure there is space below the bank
Reserve As Chip Work 9,10000
'
' Reserve big bank
'
SIZE=SX/8*SY
Reserve As Chip Work 10,SIZE*NSCREEN
'
' Free security memory to reserve the screen at the same place
'
Erase 9
'
' Reserve base screen
'
Screen Open 0,SX,SY,2,Lowres
Curs Off : Cls 0
```

CONTINUED ON PAGE 10 ⟶

```
Colour 1,$EEE
'
' Display nice drawings
'
Scroll Off : For X=0 To 80 : Print "Bone Demo 5 "; : Next
BONE[160,100,100,20,25]
'
' Make Sure memory is well allocated
'
If Logbase(0)>Start(10) : Stop : End If
'
' Open two workscreens
'
Screen Open 1,SX,SY,2,Lowres
Curs Off : Screen Hide 1
Screen Open 2,SX,SY,2,Lowres
Curs Off : Screen Hide 2
'
' Sinus step
STP#=2*Pi#/NPAS
'
' Image building loop
For N=0 To NPAS-1
  P#=P#+STP#
  Screen 1 : Cls 0 : Screen 2 : Cls 0
  '
  ' Movement in X, screen 0 -> 1
  For Y=0 To SY-1
    X=AMPX1#*Sin(Y*AMPY1#-P#)
    Screen Copy 0,0,Y,SX,Y+1 To 1,X,Y
  Next
  '
  ' Movement in Y, screen 1 -> 2
  For X=0 To SX-1
    Y=AMPY2#*Sin(X*AMPX2#-P#)
    Screen Copy 1,X,0,X+1,SY To 2,X,Y
  Next
  '
  ' Copy into bank
  Copy Logbase(0),Logbase(0)+SIZE To Start(10)+N*SIZE
Next
'
' Close work screens
Screen Close 1 : Screen Close 2
Screen 0 : Cls 0
'
' Distance between screen and bank
TL=SX/8
D=Start(10)-Logbase(0)
OY=D/TL : OX=(D mod TL)*8
'
' Set up AMAL registers
Amreg(0)=OX : Amreg(1)=OY : Amreg(2)=SY :
Amreg(3)=NPAS
'
' Set up AMAL string
Channel 0 To Screen Offset 0
'
A$=A$+"L: For R0=0 To RD;"
A$=A$+"  Let X=RA;"
A$=A$+"  Let Y=R0*RC+RB;"
A$=A$+"  Next R0;"
A$=A$+"  Jump L;"
'
' Start AMAL, end of program!
Amal 0,A$ : Amal On
Direct
'
Procedure BONE[X,Y,SX,SY,C]
  Bar X-SX,Y-SY To X+SX,Y+SY
  For R=1 To C
    Circle X-SX,Y-SY,R
    Circle X-SX,Y+SY,R
    Circle X+SX,Y-SY,R
    Circle X+SX,Y+SY,R
  Next
End Proc
```

Some Explanations.

To do such an animation, we must make possible to display quickly a lot of pre-calculated pictures in on screen. We could reserve 8 screens, and display them using screen to front. But this would only make 8 frames of animation, not much. We could possibly turn the screens into double buffer, but it would only make 16 frames.

So, as I am a genius dog (Remember it?), I reserve a very big bank, containing enough chip ram to put all the pre-calculated pictures, and I use a special trick to display the bank in a normal screen.

-I first free as much memory as possible to reserve the big bank.

-I define the deformation. NPAS is the number of steps (I'm french, you know!), AMPX and AMPY define the sinus wave. Have fun changing these values!

-One thing is crucial for the display process: the big bank must be higher than the screen, in memory. So I reserve a security bank, then the big bank. The big bank will be reserved at a higher place than the security, of course. Before reserving the screen, I free the security memory, UNDER the bank. Clever dog I tell you!

-Then I open a couple of screens to do the sinus process, and I hide them.

-Comes the calculation loop. I make the X movement from screen 0 to screen1, then the Y movement from screen 1 to screen 2. When screen 2 is ready, I just copy the content of the bitmap into the big bank.

-The display process. I use SCREENOFFSET. You may have already seen that you can display the whole memory of your Amiga by giving high value to the SCREENOFFSET instructions. Here I carefully calculate the values of SCREENOFFSET to display the content of the bank. As simple than that!

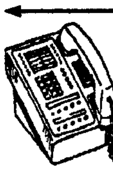
-Now, I can initialise an AMAL string, that will simply modify the Y coordinate of SCREENOFFSET, to display every screen one after each other.

-Everything is ready, I can come back to direct mode. The demo is running, it does not take any processor time at all!

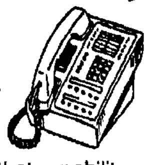
All right! I have to leave you know. Back to work. Lucky guys, you are in spring in Australia, then in summer. We are in Autumn, the weather is cooooooolllIdddddd these days. Anybody wants to have a nice, intelligent, charming black dog for the summer? Hmmmmmmmmm?

Oh, I forgot to tell you. Francois is on USENET, the international computer network and will answer to all questions about AMOS in: "COMP.SYS.AMIGA.PROGRAMMER"!

See you in the next newsletter, Daisy!



UPDATE!



AMOS MODEM SCENE.

There is some great news in the Modeming Areal Predators has now started receiving NETMAIL, WOW you may be thinking, every other BBS already has that capability. BUT, Predators has now started the first AMOS NETMAIL Facility! This means that if you live in WA, or anywhere else for that matter, and you are able to send and receive messages to other AMOS users, including myself, without the cost of a STD call! Think of the savings in phone charges, there is of course only one side effect of this whole thing. It may take a while for a message to get from your local BBS to Predators, and of course a delay in the reply getting back as well. But you will now be able to contact lots of new AMOS users all over Australia! But this of course depends on your local BBS taking up this new NETMAIL section, if your BBS does not then you will just have to hassle the Sysop into picking this NET up.

We are still getting a few people on the boards, but it always seems that it is the same old regulars who are chewing the fat, so lets see some new faces on the wire! New files are appearing all the time and from time to time I put a large batch of new programs up into the AMOS section for all to download. So if you have not been on the wire for quite some time then you should dust off the modem and check out what has been happening lately!

See you on the wire! TEX!

Quickie 2: If You Are Having Problems Trying To Display All 8 Sprites (16 Wide) On The Same Line, Then Try This: Open A 4 Colour Screen, Paste Bob, And Then Sprite Grab Them Back Into Their Original Position In The Bank.