AMOS Newsletter

Volume 7, May 1991

HELLO, HELLO, (OR WE'RE BACK PART 7!)

Welcome to the 7th edition of the Newsletter! Thanks must go to all the people who wrote in and gave their comments about the newsletter in general, some of these suggestions will be acted on over the next couple of editions and some have already been implemented in this one. Enough about the future, lets talk about the present! As you can see this newsletter is bulging at the seams! Yes in this one we have an extra three pages of articles, programs and other assorted bits and pieces.

Quite a bit of interest was shown in Sausages guide to using Sound/Noise trackers, so part two of this series is inside and according to Sausage he has enough information to fill another 3-4 articles! So by the time you have read the entire series you should all be sufficiently proficient to rattle out your own original tunes for your games/educational programs and demos. In this article he delves deeper into the hot-keys and editing keys used in Noisetracker, also he continues writing the "Ultimate" tune. So if you enjoyed the last article don't miss this one.

One of the other articles that attracted a lot of attention was my article on writing adventure games. Boy I didn't realize the job I had ahead of me and the time it would take to write this article. The article in this newsletter had to be scaled down and came out at three full pages of almost pure text! Explaining something as complex as an adventure parser is not a short exercise. As promised in this article is the Parser itself plus a couple of the most commonly used commands. It ended up almost being a mini game in itself. Mainly because I included a couple of problems and surprises for you to discover along the way. This series is not finished here by any means, I have plans for another 2-4 articles (Hopefully not as long as this one!) covering many topics to do with writing adventure games so keep your eyes out for newsletter 8!

Unfortunately Daisy's regular column will not be appearing in this edition, it would seem that Francois has taken her away on holidays with him so I have not been able to Dog-Fax her. I was hoping to have some more information about the compiler in her column but it was just not meant to be. About the time that you are reading this the compiler should be landing here in Australia or even be in the shops already. Fingers crossed that no problems happen between now and then! Unfortunately I cannot give you a price as Mandarin still have no idea themselves yet. But keep your eyes out for the compiler, If you are contemplating buying the compiler then please read the notice on page 9.

But the compiler is not the only good news that I have, TOME, The TOtal Map Extension system is now available, see page 3 for more details. But now for the bad news, the AMOS 3-D Extension will not be available now for at least another 2 months. But hopefully we should have a demo real soon to show the power of this new system. One thing that has disappointed me greatly is the PD or lack of being submitted by Australian Users, we have the same number of users in the club as does England, but when it comes to submitting PD we are falling further and further behind. At one stage we were only about 15-20 disks behind the British, but now we have fallen almost 100 disks behind! I have received 40-50 new disks from them and only about 10-12 from Aussies! What is happening out there? Is AMOS just sitting on the shelf? The British are bagging us Aussies all the time in Magazines as being slack so come on guys lets get stuck into it, it does not have to be a full blown game, even just procedures that do things are great. One disk that springs to mind from the last lot from England was a disk just with procedures etc on it, some were as simple as 2 lines long! Lets get this PD library kicking and hopefully have the total up over 100 by the time the next Newsletter comes along!

You will notice that from now on I will only be publishing the NEW Editions to the PD Library, this saves heaps of space so that I can put more articles in instead. Every third or fourth Newsletter I will publish a full listing, so keep these additions handy with your other original ones.

The BLADE have once again contributed another interesting article on writing arcade games, in this article they show you how to write the frame-work for an "ASTEROIDS" style of game. All that is basically missing is the enemy or asteroids, scoring and collision detection routines. A great simple game could be easily built up from this shell. The Blade are currently working on a 3-D car racing game, I have seen their racing track routines working and this is going to be hot, but then you would come to expect that from them. They are also going to be regular contributors to this Newsletter from now on so you can count on more interesting articles to come!

Some good news, Chris Whale who runs the RPG Club has told me that he was swamped with letters from the last newsletter, so it would seem that it is not going to die, which is great! It is undergoing its name change in this edition, but I will let Chris tell you about that in his article.

I was recently speaking to Nic Wilson and he told me that his NoVirus program has been updated yet again! So if you still have not ordered your copy at the special price then the offer only lasts for the duration of this newsletter. You have until the release of the next newsletter to save yourselves $15.00 off the normal price. For those of you who did take up the offer, Nic thanks you for supporting Aussie Programs and Programmers. You can of course get your copy updated when ever you like, simply send it back to him. Remember, if you want to order a copy you must use the order form from the last Newsletter, otherwise you will be charged the normal price of $49.95.

Well Thats about all from me. Theres plenty to read so get stuck into it! Hope you enjoy this edition because I think that they just keep getting better. Till next time CU Later!

Neil Miller

Newsletter Vol 7, Contents

THIS SPACE INTENTIONALLY LEFT BLANK FOR WRITING DOWN PHONE NUMBERS YOU WOULD NORMALLY FORGET!

HOW TO SERIES:
How To Write An AMOS Extension

How to write an extension for AMOS

Alex J. Grant Apr 1991.

We all know that AMOS is a very powerful language as it stands, with itsadvanced graphics and sound facilities, but what if that is just not enough for your every power hungry programming mind? What if there is some function or command that would (to you) be an obvious improvement, yet does not exist? Fear not! Our dear friend Francois has left us a way to write our own commands for AMOS. Hooray! I hear you all shout. The world is saved, I can write my 18 dimensional time warped graphics commands to add to AMOS... but how is it done?

Let me state right at the beginning (well the second paragraph will do...) That I am NOT going to attempt to explain,teach,tutor etc. ANYTHING to do with 68000 assembly language. That is your responsibility as a programmer to come to terms with. Neither will I replicate word for word what Francois has already written in his documentation to the Music.s file on the 'Extras' disk. (Note that you should study this file closely.) This discussion is purely a 'how-to', the aim of which is to show you from start to finish how to write a simple extension for AMOS.

As mentioned previously, AMOS extensions are machine language 'add ons'. At first sight they may seem complicated, but if this is so, remember that AMOS is a very powerful, and hence complicated language. OK enough of the talk, lets get down to business. I will use as my example a extension to add an extremely useful function that returns a string - 'Hello There"

There are six basic parts to an extension: what follows is an outline of these sections

1. Cold Start - This is a small piece of code that AMOS will call on startup. It needs to initialize anything that you will need, and must pass some information about the extension back to AMOS.

2. Screen Reset routine - This is a routine that AMOS will call every time the screen is reset ie. a Default command. eg. turn off sprites, stop music etc.

3. Quit routine - This code is called as AMOS exits. Here you should close ANYTHING you opened, free anything you grabbed etc.

4. Main code - The main code...

5. Token table - Information about the syntax of your commands

6. Data - Any data you might need.

Cold Start.

This code must be at the very start of your program. AMOS will jump to it when it has finished loading. On entry, AMOS passes to you several important pieces of information:

A0: Branch table (See Music.s for description - note that some of the instructions about parameters etc. are WRONG. The best way to be sure is to see where Francois uses them himself and to note what he does.)
A1: intuition.library base address
A2: graphic.library base address
A3: diskfont.library base address
A5: AMOS data zone address.

If you need any of this you will need to store it straight away.

You need to pass some information back to AMOS:

A0: Token table address
A1: Welcome message address
A2: Address of screen reset routine
A3: Address of quit routine
D0: 0/1 -> OK/ERROR
D1: Extension number. This is NOT the same as the number of the 'Loaded extension' in the list in the Config.AMOS program. It is in fact one less. Ie. if you put your extension as the fourth entry in the loaded extension list, D1=3.
D2: Address of a routine to be called when memory banks are changed, A 0 if no routine.

All that above information can be found in the Music.s file, but note where I have made corrections. Always check that 'Francois do what Francois say'

Ok, lets do our first bit of code.

```
; hello.s - a test extension for AMOS
;
; Cold Start part.

 movem.l a4-a6,-(sp) ; you must preserve these registers always!
 movem.l #0,d2  ; no change bank routine
 lea Tk(pc),a0 ; Token table address
 lea HelWel(pc),a1 ; Welcome message address
 lea HelDef(pc),a2 ; Screen reset routine
 lea HelEnd(pc),a3 ; Quit routine
 moveq #3,d1  ; extension number 3
 moveq #0,d0  ; no errors
 movem.l (sp)+,a4-a6 ; restore registers always!
 rts
```

You will have noted that everything is in relative addressing (pc) mode. This will be required by the compiler.

2. Screen reset routine.

We don't want to do anything on a screen reset so our next bit of code is as follows:

```
HelDef: rts ; do nothing
```

3. Quit routine.

Again, we don't need to do anything in particular.

```
HelEnd: rts
```

4. Main code.

Well, this can be anything you want, but here is what I did.

```
Hello: movem.l a4-a6,-(sp) ; never forget this!

 lea HelTxt(pc),a0 ; the address of the string to be returned.
 move.l a0,d3
 moveq #2,d2  ; #2 means a string
 movem.l (sp)+,a4-a6
 rts
```

That's the main code! Now for some explanation of how the variable is passed back to AMOS.

If you are writing a function or a reserved variable, you will need to return a value. This value is returned in D3, and its type in D2. The types are as follows:

0 - Integer (value in d3)
1 - Float (value in d3)
2 - String (address in d3)

Strings are stored as follows: (This is not part of our code)

```
Test: dc.w 17 <— length
 dc.b "This is some text" <— text
```

5. The Token table.

This is a very important part of an extension. It tells AMOS about the format of your commands. See Music.s for a full description of how to use it.

The entries in the token table are as follows:

```
 dc.w Address of instruction-Tk, Address of function-Tk
 dc.b "instruction nam","e"+$80,"Param list", -1 (or -2)
```

Addresses:
In the first line, you put the address of your code. If you are writing an instruction (no return value) you put it in the left side. If its a function it goes on the right. You put a '1' in the unused field. Note that reserved variables use BOTH fields. (they are made the same)

```
instruction :-> dc.w inst-Tk,1
function   :-> dc.w 1,func-Tk
reserved var:-> dc.w var-Tk,var-Tk
```

Instruction name:
The instruction name must have $80 added to the last letter. You will need to see how your assemble handles this. You may have to do it manually.

Parameter list:
The parameter list works as follows:

The first character determined the type of instruction/function

I - instruction

functions: 0 - integer
1 - float
2 - string

reserved var: V0 - integer
V1 - float
V2 - string

Next you need you list of input parameters which follow the same 0,1,2 type convention. This is best shown by example. If I want my function to be called like so: Doobah(a$,b#,c to d,e,f) returning an integer my list would be so:

```
 dc.b "Dooba","h"+$80,"02,1,0t0,0,0",-1
```

If you have no more input syntaxes for your function, you end with a -1. If you have a variety of input parameter combinations, you list them all separating with -2.

```
eg. dc.w 1,func1a-Tk
 dc.b "xxx","x"+$80,"00",-2 <- one parameter
 dc.w 1,func1b-Tk
 dc.b "xxx","x"+$80,"00,0",-1 <- two parameters.
```

Note that you will have to sort out the unpiling of these different sets of parameters, which brings me to the next point.

If you have parameters passed to you, you will need to unpile and store them. They are all pushed onto a3, so you simply get them in reverse order.

```
eg. move.l (a3)+,a0 ; parameter 2
 move.l (a3)+,a1 ; parameter 1
```

You need to do this as soon as your function is called. Since our function hasn't got any parameters, we don't need to worry.

Continued On Page 3 ⟶

Finally, here is our token table:

```
dc.w 1,0
dc.b $80,-1 ; These two lines must always be first
dc.w 1,Hello-Tk
dc.b "hell",239,"2",-1
dc.w 0 ; You must end in a zero.
```

5. Data

The only data that we need is our welcome message (That is prints on the startup screen) and our hello text.

```
HelTxt: dc.w 11
 dc.b "Hello There"
HelWel: dc.b 27,"Y",48+11,"Hello V1.0",0
dc.l 0
```

Note the "Y",48+11 ... this tells AMOS to write it on the eleventh line down, so "Y",48+15 would be the 15th line - easy.

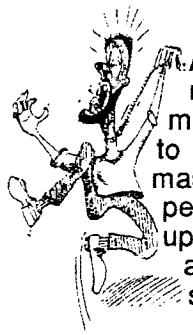
Well, that's about it. Type in the example and assemble and link it. Copy it into the AMOS_System directory of your ** ***BACKUP*** * AMOS disk. The boot up AMOS and load the 'Config.AMOS' program. Choose 'load default configuration' (***ARE YOU USING A BACKUP OF YOUR PROGRAM DISK!!!!!!***) and then (one it has loaded) choose 'Loaded Extensions' from the 'Set' menu. Enter the name of your extension in the list at number 4 (if you made d1=3 on cold start) and then save your configuration. Now reboot AMOS, and if all is well, underneath the Music Player, Compact and Request messages should be 'Hello V1.0'

Go into direct mode and type print hello. If everything is OK, you will see 'Hello There' displayed. If not, you'll probably meet Mr. Guru! (but that is more your problem than mine - my Hello command works fine!)

Look out for my Dump extension - which allows you to do screen dumps to the printer from within AMOS, and coming soon - trackdisk.device commands.
(Available from the PD Library NOW! ED)

Here is the complete code - for completeness.

```
; hello.s - a test ML routine for AMOS

; Cold start part

 movem.l a4-a6,-(sp) ; save registers

 move.l #0,d2  ; No routine for change bank

 lea  Tk(pc),a0  ; Token table
 lea  HelWel(pc),a1  ; Welcome message
 lea  HelDef(pc),a2  ; Screen reset
 lea  HelEnd(pc),a3  ; Quit
 moveq #3,d1  ; number of extension
 movem.l (sp)+,a4-a6
 moveq #0,d0
 rts

HelDef: rts ; do nothing on a screen reset
HelEnd: rts ; do nothing when quitting!

; Now the code!

Hello: movem.l a4-a6,-(sp) ; save registers

 lea HelTxt(pc),a0 ; put the address of the text
    ; note that the first word of a
    ; string is its length!!!!!!!
 move.l a0,d3  ; into d3
 moveq #2,d2  ; #2 means string!!

 movem.l (sp)+,a4-a6

 rts

Tk: dc.w 1,0
 dc.b $80,-1

 dc.w 1,Hello-Tk
 dc.b "hell",239,"2",-1
 dc.w 0

HelTxt: dc.w 11
 dc.b "Hello There",0  ; the text to be returned

HelWel: dc.b 27,"Y",48+11,"Hello v1",0

 dc.l 0
```



Bloopers Corner

As you can see I am hanging myself in this newsletter! Why you may ask, well quite simply I forgot to put The Twilight Zone on the master disk for AA62! A number of people have ordered it and picked up that mistake, if you did order it and haven't had it fixed up yet send it back and I will fix it up!

HANDY KEYBOARD SHORTCUTS

This is a list of handy Keyboard-Shortcuts (for more detail and extra commands refer to the manual from page 24 onwards)

LOAD: AMIGA+L - Load A Program.
SAVE: AMIGA+S - Save A Program.
SAVE AS: SHIFT+AMIGA+S - Save A Program Under A New Name.
INDENT PROGRAM: F3 - Indents And Checks The Syntax Of The Program.
TEST PROGRAM: F2 - Checks The Syntax Of The Program.
RUN PROGRAM: F1 - Runs The Program After First Running A Syntax Test.
DIRECT MODE: ESC - Closes The Editor And Goes To Direct Mode.
UNFOLD/FOLD PROCEDURE: F9 - Folds Or Unfolds A Procedure.
LINE INSERT: CTRL+I - Moves Text Down From The Cursors Position Creating A New Line.
QUIT: SHIFT+F10 - Quits AMOS.
DELETE PART OF LINE: CTRL+Q - Deletes all text to RIGHT of the cursor.
MARK ALL PROGRAM TEXT: CTRL+A - Marks all program Text.
MARK BLOCK BEGINNING: CTRL+B - Marks The Beginning Of A Block.
MARK BLOCK END: CTRL+E - Mark The End Of A Block.
BLOCK CUT: CTRL+C - Cut A Highlighted Block Out.
BLOCK PASTE: CTRL+P - Paste A Highlighted Block At The Cursors Position.
BLOCK STORE: CTRL+S - Saves The Highlighted Block Into Memory, You Can Now Move To Another Program In Memory And Paste The Block.
BLOCK MOVE: CTRL+M - This Erases The Highlighted Block And Moves It To The New Cursors Position.
DESELECT BLOCK: CTRL+H - This Deselects The Highlighted Block.
BLOCK PRINT: CTRL+F10 - This Prints The Highlighted Block On Your Printer.
FIND STRING: CTRL+F - Searches Downwards From The Current Cursor Position For EXACT Matches Of Your Input.
FIND NEXT: CTRL+N - Continues Searching Downwards For The Next EXACT Match.
REPLACE TEXT: CTRL+R - Ability To Replace Single Occurrences of The "Find" Text With New Text.
REPLACE ALL: ALT+F5 - Replaces All Occurrences Of The "Find" Text With The New Text.
OPEN ALL PROCEDURES: ALT+F7 - Opens All Procedures.
CLOSE ALL PROCEDURES: ALT+F8 - ClosesAll Procedures.

TOME

TOtal Map Extension

Thats right, the TOME Extension system is now available through the AUSSIE AMOS USERS CLUB. This extension includes 27 new commands for drawing and scrolling large maps/playing areas. You can store up to 960 screens of information in just 64K of memory! What does that mean? Well quite simply it means that you can have a horizontal/vertical scrolling shoot-em-up that is 960 screens long/high! Or a large Faery Tale style scrolling area around 31x31 screens. These can all be scrolled effortlessly! Tome includes the most advanced MAP Editor found on any computer! + A free game to show you the power of the TOME System!

You can also order a demo of the TOME System from the PD Library, this is a game written using the TOME System. You can find it on the British PD Disk -BA123. This game is called DEADLINE. It is a large scrolling game, a bit like Time Bandit, Gauntlet, Boulderdash and Loderunner all mixed into one!

TOME will cost you just $59.95 + 2.05 for postage. But before you rush off and order TOME, You require AMOS1.23 & 1 Meg of memory to be able to use the Map Editor (TOME itself will work on a half meg machine but the editor needs 1 meg) to use it you must first update to the latest version of AMOS, V1.23. TOME cannot and will not work with any other version of AMOS! See the order form at the rear of this newsletter for more details.

One last word, TOME is not PD! Tome is a commercial program and is the result of hundreds of man hours by the creator. This means that it is covered by normal Copyright Laws! So you just can't give a copy to your mates. If you wish to see further extensions released for AMOS then do the right thing and

BUY IT IF YOU WANT IT!

Writing Adventure Games. Part 2

Welcome to the second installment in a series that may go on forever and ever! This installment is short and sweet at just 3 pages! This article has taken me many hours of late nights pounding away at the keyboard and I truly hope that you learn something from this series! So if I don't see a big influx of Adventure games being submitted in the next couple of months, then I will be very dissapointed indeed. But then judging from the letters that I received since the last newsletter, there are some people out there who are desperate enough to actually have read the article and are looking foward to this one. I'm glad that you have started planning your adventures and you have started your Noun, Verb & Object lists, Maps and of course your scenario.

Well as promised, this newsletter I am publishing the Parser and a couple of the commands. (Depends on how much room it takes up.) To get to this stage I had to search frantically high and low for ages for a long abandoned Amiga Basic Disk. (Sorry, I swore!) After finding it, I had the wonderful task of converting that code to AMOS, and then making it work properly. That code is below, there is a bit to type in, but I will explain what each line does. (It is pretty simple really!)

But before we do that, type in this small listing below, check your typing and then save it. This is a little added bonus program I wrote in 5 minutes to make the job of designing your maps a little less painful. It simply allows you to enter your map data directly, short room description first then your map data for that location. You all of course wrote out your map data like I did in the last newsletter didn't you!? When you have entered your last location, simply enter a "Q" as a location description and you will be asked for a file name. The program will then write your map data directly into the required DATA Statements. This can then be MERGED directly into your/my adventure. (You will have to do this anyway for this example.) One last thing, if you examine the line that DIM's the variables you will notice that there are two 100's. This dictates the maximum number of locations to be held in the arrays. If you have more locations then simply change them to what ever you like!

```
Set Buffer 50
Rem                ADVENTURE MAP EDITOR
Rem                  (c) OSCARSoft
Rem                     by TEX
Dim OB(100,10),OB$(100),M$(10) : Screen Open 1,640,290,2,Hires : CL=1
Data "N","S","E","W","NE","SE","SW","NW","U","D"
For X=1 To 10 : Read M$(X) : Next X
Do
  Cls 0 : Colour 1,$FFF
  Centre "Location"+Str$(CL) : Cdown : Print
  Input "Loc Name-";O$ : OB$(CL)=Upper$(O$)
  Exit If OB$(CL)="Q"
  For X=1 To 10
    Print "Map-(";M$(X);") "; : Input OB(CL,X)
  Next X
  Inc CL : Cls : Locate 0,0
Loop
F$=Fsel$("*.ASC","ADVENTURE-MAP.ASC","SELECT PATH","")
If F$="" Then End
Open Out 1,F$
For X=1 To CL-1
  T$="Data "+Chr$(34)+OB$(X)+Chr$(34)+","
  For Z=1 To 10
    T$=T$+Str$(OB(X,Z)) : If Z<=9 Then T$=T$+","
  Next Z
  Print #1,T$
  T$=""
Next X
```

That was painless now wasn't it!? This is going to be a BIG article this edition, the Code below is complete except for 21 out of the 22 of the required locations map data. I needed the space, and it gives you a good reason to go back and type in the program above and then use it!

Once you have the map editor up and running simply grab newsletter 6 (You of course have kept it in a safe place), and enter the map data from location 2 down. Keep your location descriptions brief as they only serve to tell you where you are, not what is around you. Save the data, don't forget to put a .ASC on the end or just select your path and then hit return.

Now for the complete program. (Leave line numbers off!) Please be careful when typing the program in, this font that I use makes it hard to tell the difference between a Zero "0" and the letter O "O". There is a lot to type in, don't be tempted to run it before it is all typed in or you will just get error messages or even worse a GURU!

A SIMPLE ADVENTURE

```
1) Set Buffer 100
2) Screen Open 0,640,238,2,Hires : Screen Display 0,,68,, : Cls 0 :Colour1,
$FFF : Screen Open 1,640,25,2,Hires : Cls 0 : Colour 1,$FFF : ScreenDisplay
 1,,42,,
3) P=22 : LO=29 : NN=LO : NV=13 : SW=3 :  CL=2:NW=6
4) DimVB$(NV),OB1$(LO),OB#(LO,2),NO$(NN),P$(P),P(P,10),AQ(LO+1),
5) GlobalVB$(),OB1$(),OB#(),NO$(),P(),P$(),NV,NO,LO,CL,NN,P,AQ(),VB,CPI,
VB$,NO$,AZ,SD,NL,BW
6) NL=0
7) SETUP
8) Do
9)   Proc PARSER
10)  Proc BRANCHER
11)  If SD=12 Then Exit
12)  If NL>0 Then Inc NL : If NL=4 Then OB#(25,1)=0 : Print "The newsletter
finally burns into ashes and falls to the ground."
13) Loop
14) Print "You Died!" : Wait Key : End
15)  Procedure SETUP
16)  Data "IN A SECRET ALCOVE",0,3,0,0,0,0,0,0,0,0
17)  Data"INVENTORY","I","LOOK","L","QUIT","Q","GO","GET","DROP",
"EAT","READ","LIGHT","BURN"
18)  Data"NORTH","SOUTH","EAST","WEST","NORTHEAST",
"SOUTHEAST","SOUTHWEST","NORTHWEST","UP","DOWN"
19)  Data "N","S","E","W","NE","SE","SW","NW","U","D"
20)  Data "ALL",0,0,0,""
21)  Data "PILLOW",7,0.9,0,"An Embroidered Silk Pillow","CRYSTAL",
14,0.4,0,"A Shimmering Crystal"
22)  Data "MATCHES",20,0.2,0,"A Box Of Matches","NEWSLETTER",5,0.4,0,
"An Old AMOS Newsletter"
23)  Data "PARCHMENT",10,0.2,0,"A Crumbling Parchment","DONUT",4,0.1,0,
"A Donut"
24)  Data "WEB",18,0,1,"A Sticky Spiders Web","BOX",0,0.1,0,"An Empty
Match Box"
25)  For X=1 To P : Read P$(X)
26)    For Z=1 To 10 : Read P(X,Z) : Next Z
27)  Next X
28)  For X=1 To NV : Read VB$(X) : Next X
29)  For X=1 To 20 : Read NO$(X) : Next X
30)  For X=21 To NN : Read NO$(X),OB#(X,1),OB#(X,2),OB#(X,0),OB1$(X) :
Next X
31) End Proc
32) The Main Code Starts From Here On In, Be Careful!
33) Rem
34) Rem
35) Procedure PARSER
36)  If CPI=1 Then Goto P1 Else Proc LOOK
37)  P1:
38)  Print "> ";
39)  CM1$="" : AA$="" : CM$=""
40)  P2:
41)  Repeat : AA$=Inkey$ : Until AA$<>""
42)  AA$=Upper$(AA$)
43)  Z=Asc(AA$) : If Z>90 Then AA$="" : Goto P2
44)  ZL=Len(CM$)
45)  If Z>31 Then CM$=CM$+AA$ : Print AA$; : Goto P2
46)  If Z=13 Then Print : Goto P3
47)  If Z=8 and ZL Then CM$=Left$(CM$,ZL-1) : Cleft : Dec ZL : Cline(1) : Goto
P2
48)  P3:
49)  N1$="" : V1$="" : VB$="" : NO$="" : VB=0 : NO=0 : LC=Len(CM$)
50)  If LC=0 Then Print "Hey I can't do this all myself! Try typing something in
and just maybe somethingwill actually happen around here!" : Goto P6
51)  P=Instr(CM$," ") : If P=0 Then VB$=CM$ Else VB$=Left$(CM$,P-1)
52)  V1$=VB$
53)  For I=1 To NV : If VB$(I)=VB$ Then VB=I : Goto P4
54)  Next
55)  VB=NW+1 : NO$=VB$ : Goto P5
56)  P4:
57)  If P=0 Then NO=0 : Goto P6
58)  NO$=Mid$(CM$,P+1)
59)  P5:
60)  N1$=NO$
61)  For I=1 To NN
62)    If NO$=NO$(I) Then NO=I : Goto P6
63)  Next I
64)  NO=0
65)  P6:
66) End Proc
67) Procedure BRANCHER
68)  REM
69)  If(VB>NW) and(NO$="") Then Print "To really get anywhere you must use
a direct object!" : Goto B1
70)    If VB>NW and NO=0 Then Print "I'm not sure that ";NO$;" is in my
vocabulary! Hang on and I'll double check." : Curs Off : Wait 100 : Print "Yep
I'm sure, I really don't know the word ";NO$;"!" : Curs On : Goto B1
71)  On VB Proc INVENTORY,INVENTORY,LOOK,LOOK,QUIT,QUIT,GO,_GET
,DROP,EAT,_READ,LIGHT,BURN
72)  B1:
73) End Proc
74) Procedure GO
75)    If NO>20 or NO=0 Then Print "I'm not quite sure I understand, Try
re-phrasing it!" : Goto G1
76)  If NO>10 Then NO=NO-10
77)  If P(CL,NO)=0 Then Print "All progress in that direction is not possible.
Please think again." : Goto G1
78)  If SD>0 Then Inc SD : Print "You are growing weaker."
79)  CL=P(CL,NO) : CPI=0
80)  G1:
81) End Proc
82) Procedure _GET
83)    If OB#(NO,1)=-1 Then Print "Hang five....Check what your carrying
because I can see you already have that!" : Goto GT6
84)  If NO=21 Then Goto GT1
85)    If OB#(NO,1)<>CL Then Print "As hard as I try, I cannot see it here!
Perhaps it's somewhere else?" : Goto GT6
86)  Print OB1$(NO);
87)  If OB#(NO,0)=1 Then Print " -You cannot pick this up!" : Goto GT6
88)  If OB#(NO,2)>10 Then Print " -You cannot pick this up, it's just too heavy
!" : Goto GT6
89)  If AZ>10 Then Print " -Can't! Drop something first." : Goto GT6
90)  AZ=AZ+OB#(NO,2) : Print " -Taken." : OB#(NO,1)=-1 : Goto GT6
91)  GT1:
92)  X=1
93)  For I=0 To LO : If OB#(I,1)=CL Then Gosub GT3
94)  Next I
95)  If X=1 Then Goto GT5
96)  For I=1 To X : NO=AQ(I) : Print OB1$(NO);
97)    If OB#(NO,2)>10 Then Print " -You cannot pick this up!" : Goto GT2
98)    If OB#(NO,0)=1 Then Print " -You cannot pick this up!" : Goto GT2
99)    If AZ+OB#(NO,2)>10 Then Print " -Can't! Drop something first." : Goto
GT2
100)     If I=X Then Goto GT4 Else Add AZ,OB#(NO,2) : Print " -Taken." :
OB#(NO,1)=-1
101)   GT2:
102)  Next I : Goto GT6
103)  GT3:
104)  AQ(X)=I : Inc X : Return
105)  GT4:
```

Continued On Page 5 ⟶

```
105)  GT4:
106)  For I=0 To X : AQ(I)=0 : Next I : Goto GT6
107)  GT5:
108)  Print "There is nothing here that you can pick up, perhaps its just
decoration?"
109)  GT6:
110) End Proc
111) Procedure DROP
112)  If NO=21 Then Goto DR1
113)  If OB#(NO,1)<>-1 Then Print "Hang five....Check what your carrying
because I don't thinkthat you have that!" : Goto DR99
114)  If NO=23 and OB#(22,1)<>CL Then OB#(23,1)=0 : Print "The crystal
smashes into dust as it hits the ground!" : Goto DR99
115)  Print OB1$(NO);
116)  AZ=AZ-OB#(NO,2) : Print " -Dropped." : OB#(NO,1)=CL : Goto DR99
117)  DR1:
118)  X=1 :
119)  For I=0 To LO : If OB#(I,1)=-1 Then Gosub DR3
120)  Next I
121)  If X=1 Then Print "But you have nothing to drop here!" : Goto DR99
122)  For I=1 To X : NO=AQ(I)
123)    If NO=23 and OB#(22,1)<>CL Then OB#(23,1)=0 : Print "The crystal
smashes into dust as it hits the ground!" : Goto DR99
124)    Print OB1$(NO);
125)    If I=X Then Goto DR4 Else AZ=AZ-OB#(NO,2) : Print " -Dropped." :
OB#(NO,1)=CL
126)    DR2:
127)  Next I : Goto DR99
128)  DR3:
129)  AQ(X)=I : X=X+1 : Return
130)  DR4:
131)  For I=0 To X : AQ(I)=0 : Next I : Goto DR99
132)  DR99:
133) End Proc
134) Procedure LOOK
135)  Screen 1 : Clw : Centre Border$(P$(CL),1) : Curs Off : Screen 0
136)  X=1
137)  For I=1 To LO : If OB#(I,1)=CL Then AQ(X)=I : Inc X
138)  Next I
139)  If X=1 Then Goto L2
140)  Print "You can see "; : SL=12
141)  For I=1 To X : NO=AQ(I) : If SL+Len(OB1$(NO))>80 Then Proc
STRING[OB1$(NO),SL] : Goto L1
142)    Print OB1$(NO); : Add SL,Len(OB1$(NO))
143)    L1:
144)    If I+1=X Then Print "." : Goto L2
145)    If I+2=X Then Print " and "; : Add SL,5
146)    If I<X and I+2<>X Then Print ", "; : Add SL,2
147)  Next I
148)  L2:
149)  CPI=1
150)  For I=1 To X : AQ(I)=0 : Next I
151) End Proc
152) Procedure INVENTORY
153)  Print "You are carrying: ";
154)  X=1 : SL=17
155)  For I=1 To LO : If OB#(I,1)=-1 Then AQ(X)=I : Inc X
156)  Next I
157)  If X=1 Then Print " Nothing but a stupid grin!" : Goto I2
158)  For I=1 To X : NO=AQ(I) : If SL+Len(OB1$(NO))>80 Then Proc
STRING[OB1$(NO),SL] : Goto I1
159)    Print OB1$(NO); : Add SL,Len(OB1$(NO))
160)    I1:
161)    If I+1=X Then Print "." : Goto I2
162)    If I+2=X Then Print " and "; : Add SL,5
163)    If I<X and I+2<>X Then Print ", "; : Add SL,2
164)  Next I
165)  I2:
166)  For I=0 To X : AQ(I)=0 : Next I
167) End Proc
168) Procedure QUIT
169)  End
170) End Proc
171) Procedure STRING[S$,SL]
172)  Dim C(20) : W=1
173)  ML=80-SL
174)  PP=0
175)  Do
176)    P=Instr(S$," ",PP) : C(W)=P : PP=P+1 : Inc W
177)    Exit If P=0
178)  Loop
179)  W=W-2
180)  For L=W To 1 Step -1
181)    CV=C(L)
182)    If CV<=ML Then Print Left$(S$,CV-1) : Print Mid$(S$,CV+1); : Goto
S1
183)  Next
184)  S1:
185) End Proc
186) Procedure EAT
187)  If OB#(NO,1)<>-1 Then Print "Hang five....Check what your carrying
because I don't think that you have that!" : Goto E99
188)  If NO=27 Then Print "You carefully turn the Donut over, take a great
big bite, and another, GULP!" : OB#(27,1)=0 : SD=1 : Goto E99
189)  Print "You try to eat ";OB1$(NO);" But Fail!"
190)  E99:
191) End Proc
192) Procedure _READ
193)  If OB#(NO,1)<>-1 Then Print "Hang five....Check what your carrying
because I don't think you have that!" : Goto R99
194)  If NO=25 and NL=0 Then Print "You read it from cover to cover and
become instantly more intelligent!" : Goto R99
195)  If NO=26 Then Print "It reads... Look for article 3 in this series!" :
Goto R99
196)  Print "You feel silly trying to read ";OB1$(NO);"."
197)  R99:
198) End Proc
199) Procedure LIGHT
200)  If OB#(NO,1)<>-1 Then Print "Hang five....Check what your carrying
because I don't think that you have that!" : Goto R99
201)  If OB#(24,1)<>-1 Then Print "Light ";OB1$(NO);" with what?" : Goto
LI99
202)  If NO=25 Then OB1$(25)="A Burning Newsletter" : Print "The newsletter
catches light and burns with protest!" : OB#(24,1)=0 : OB#(29,1)=-1 : NL=1 :
Goto LI99
203)  LI99:
204) End Proc
205) Procedure BURN
206)  If NL=0 Then Print "You have nothing to burn ";OB1$(NO);" with!" :
Goto BU99
207)  If OB#(25,1)=-1 and NO=28 and BW=0 Then Print "You try to light the
web but it does not catch fire." : BW=1 : Goto BU99
208)  If OB#(25,1)=-1 and NO=28 and BW=1 Then Print "The web finally
catches allght and starts to burns fiercely!" : P(18,8)=14 : P(14,6)=18 :
OB#(28,1)=0 : Goto BU99
209)  Print "You do not succeed in trying to burn ";OB1$(NO);"!"
210)  BU99:
211) End Proc
```

So there you have it, save it, now position the cursor at the beginning of line 17. Hold down The SHIFT Key and hit F5, this will then prompt you for the file that you saved your map data out as, select it. Wait a few seconds while AMOS processes the data and then when it is finished, save it again. Hit F3 to Indent and perform a Syntax Check. Correct any errors, save it if you need to and then RUN IT! have fun playing around there is one puzzle built in plus some hidden tricks as well. (You won't be able to get to all locations until Newsletter 8)

Well we have certainly got allot of program to discuss! Some of the lines are pretty much self explanatory, so I will not explain what things like End Proc mean or do. Well, As they say "I guess we had better take it from the top". (Who the hell are "THEY" anyway?)

Well here we go with the prodigious task of explaining the mess that you just typed in, you did just type that in didn't you?

Ok well Lines 1 - 6 do the initial setting up of the screen, setting the variable buffer to 100K to make sure we don't run out of memory, DIMensioning the Variables and declaring them as GLOBAL Variables. Line 3 sets up some important default Variables, they are out lined below.

P=22 This is the number of locations in the Adventure.
LO=29 This contains the number of Nouns(Objects).
NN=LO Same as above.
NV=13 This contains the number of Verbs.
CL=4 This always contains your current Location.

When writing an adventure always write down each new Variable that you start using and write next to it what it is used for, because after writing 500 lines or more of code, leaving it for a week or more and coming back to keep going you will be hopelessly lost in a maze of mixed up variables.

OK, line 7 calls the procedure SETUP which starts at line 15, lets now examine this procedure. As you can see it contains all of our data for the adventure, first there is our Map Data, then our Verb Data, then our Noun Data. When typing text into data statements you must put "" around each element otherwise AMOS will treat it as program code. If you are going to add more commands, which you will have to, make sure that you place them directly after these ones, start a new line with a Data command and list them after each other. The only exception to this rule is commands that only require a Verb only, Like QUIT or HELP etc. They must get inserted directly after then "Q" in these data statements. If you do this you must also go to line 3 and change NW= to reflect the new amount of Verb only words. So if you add two more Verb only words then you will change NW to 8. If you don't make these changes you will get some unpredictable - But stupid results!

After the Noun Data for the directions you will notice line 20, this is a dummy object that allows you to have ALL in your adventure eg GET ALL, DROP ALL etc. The rest of the Nouns follow, in this adventure I have used the following system for the Nouns/Objects.

Data "PILLOW",7,0.9,0,"An Embroidered Silk Pillow"

First we have the ACTUAL Noun-PILLOW This is of course the word that the adventurer would type in to interact with this object. This is stored in the Array NO$().

Next we have the location that this object is to be found at in our adventure, refer to the last newsletter for more information on this. This is stored in the Array OB#(?,1).

Next we have the WEIGHT of the object, in this case it weighs 0.9 KGs. This is used to place weight restrictions on how much your adventurer can carry. This is stored in the Array OB#(?,2).

next is a 0, this can either be a 0 or a 1, this dictates wether an object can be picked up or not. If it is set to 0 then you can, if it is set to 1 (Like the Spiders Web-Line 24) then you cannot. This is mainly here so that you can have buttons and obstacles in your adventure that cannot be moved, I mean who ever heard of you being able to pick up a button? This is stored in the Array OB#(?,0).

Lastly we have the small description which is displayed when you enter a room or when you do an Inventory listing. This is stored in the Array OB1$().

If you want to add values to objects then simply Dimension OB#(?,?) to another element like so...DIM OB#(LO,3). Then in your read Data statements simply add ,OB#(X,3) after the other Arrays so your line would look like so..Read NO$(X),OB#(X,1),OB#(X,2),OB#(X,0),OB1$(X),OB#(X,3). You must make sure that you then add you new value AFTER the SMALL DESCRIPTION in the data statements, simply add a , and the value like "A Round Donut",5. With 5 being the value, and of course a zero is quit legal.

Lines 8 to 13 contain the main control loop, first we call the PARSER Procedure which waits for your input, processes it and then returns back to line 10. Line 10 then calls the BRANCHER Procedure which contains some error checking and of course sends program control off to the relevant Verb Procedure. Once the called Verb procedure is finished, program control returns to line 11. Lines 11 and 12 contain tests for certain conditions, line 11 tests for the Variable SD=12. If this condition is met then the programs control exits the loop and goes to line 14 where you are told you have died, the program then waits for a keypress and then exits. Line 12 tests to see if the variable NL>0, this means in this case that you have lit the newsletter. If this is true then the variable NL is incremented, another test is then performed to see if NL has reached 4. If so then we make the Newsletter disappear and give a relevant message. NL is then reset to 0 again. Remember all these tests are performed after each action that the player makes.

THE PARSER ROUTINE:

OK now it's time to examine the Parser itself, this is a very simple Parser

Continued On Page 6 ⟶

```
106)  For I=0 To X : AQ(I)=0 : Next I : Goto GT6
107)  GT5:
108)  Print "There is nothing here that you can pick up, perhaps its just
decoration?"
109)  GT6:
110) End Proc
111) Procedure DROP
112)  If NO=21 Then Goto DR1
113)  If OB#(NO,1)<>-1 Then Print "Hang five....Check what your carrying
because I don't thinkthat you have that!" : Goto DR99
114)  If NO=23 and OB#(22,1)<>CL Then OB#(23,1)=0 : Print "The crystal
smashes into dust as it hits the ground!" : Goto DR99
115)  Print OB1$(NO);
116)  AZ=AZ-OB#(NO,2) : Print " -Dropped." : OB#(NO,1)=CL : Goto DR99
117)  DR1:
118)  X=1 :
119)  For I=0 To LO : If OB#(I,1)=-1 Then Gosub DR3
120)  Next I
121)  If X=1 Then Print "But you have nothing to drop here!" : Goto DR99
122)  For I=1 To X : NO=AQ(I)
123)    If NO=23 and OB#(22,1)<>CL Then OB#(23,1)=0 : Print "The crystal
smashes into dust as it hits the ground!" : Goto DR99
124)    Print OB1$(NO);
125)    If I=X Then Goto DR4 Else AZ=AZ-OB#(NO,2) : Print " -Dropped." :
OB#(NO,1)=CL
126)    DR2:
127)  Next I : Goto DR99
128)  DR3:
129)  AQ(X)=I : X=X+1 : Return
130)  DR4:
131)  For I=0 To X : AQ(I)=0 : Next I : Goto DR99
132)  DR99:
133) End Proc
134) Procedure LOOK
135)  Screen 1 : Clw : Centre Border$(P$(CL),1) : Curs Off : Screen 0
136)  X=1
137)  For I=1 To LO : If OB#(I,1)=CL Then AQ(X)=I : Inc X
138)  Next I
139)  If X=1 Then Goto L2
140)  Print "You can see "; : SL=12
141)    For I=1 To X : NO=AQ(I) : If SL+Len(OB1$(NO))>80 Then Proc
STRING[OB1$(NO),SL] : Goto L1
142)    Print OB1$(NO); : Add SL,Len(OB1$(NO))
143)    L1:
144)    If I+1=X Then Print "." : Goto L2
145)    If I+2=X Then Print " and "; : Add SL,5
146)    If I<X and I+2<>X Then Print ", "; : Add SL,2
147)  Next I
148)  L2:
149)  CPI=1
150)  For I=1 To X : AQ(I)=0 : Next I
151) End Proc
152) Procedure INVENTORY
153)  Print "You are carrying: ";
154)  X=1 : SL=17
155)  For I=1 To LO : If OB#(I,1)=-1 Then AQ(X)=I : Inc X
156)  Next I
157)  If X=1 Then Print " Nothing but a stupid grin!" : Goto I2
158)    For I=1 To X : NO=AQ(I) : If SL+Len(OB1$(NO))>80 Then Proc
STRING[OB1$(NO),SL] : Goto I1
159)    Print OB1$(NO); : Add SL,Len(OB1$(NO))
160)    I1:
161)    If I+1=X Then Print "." : Goto I2
162)    If I+2=X Then Print " and "; : Add SL,5
163)    If I<X and I+2<>X Then Print ", "; : Add SL,2
164)  Next I
165)  I2:
166)  For I=0 To X : AQ(I)=0 : Next I
167) End Proc
168) Procedure QUIT
169)  End
170) End Proc
171) Procedure STRING[S$,SL]
172)  Dim C(20) : W=1
173)  ML=80-SL
174)  PP=0
175)  Do
176)    P=Instr(S$," ",PP) : C(W)=P : PP=P+1 : Inc W
177)    Exit If P=0
178)  Loop
179)  W=W-2
180)  For L=W To 1 Step -1
181)    CV=C(L)
182)    If CV<=ML Then Print Left$(S$,CV-1) : Print Mid$(S$,CV+1); : Goto
S1
183)  Next
184)  S1:
185) End Proc
186) Procedure EAT
187)  If OB#(NO,1)<>-1 Then Print "Hang five....Check what your carrying
because I don't think that you have that!" : Goto E99
188)  If NO=27 Then Print "You carefully turn the Donut over, take a great
big bite, and another, GULP!" : OB#(27,1)=0 : SD=1 : Goto E99
189)  Print "You try to eat ";OB1$(NO);" But Fail!"
190)  E99:
191) End Proc
192) Procedure _READ
193)  If OB#(NO,1)<>-1 Then Print "Hang five....Check what your carrying
because I don't think you have that!" : Goto R99
194)  If NO=25 and NL=0 Then Print "You read it from cover to cover and
become instantly more intelligent!" : Goto R99
195)  If NO=26 Then Print "It reads... Look for article 3 in this series!" :
Goto R99
196)  Print "You feel silly trying to read ";OB1$(NO);"."
197)  R99:
198) End Proc
199) Procedure LIGHT
200)  If OB#(NO,1)<>-1 Then Print "Hang five....Check what your carrying
because I don't think that you have that!" : Goto R99
201)  If OB#(24,1)<>-1 Then Print "Light ";OB1$(NO);" with what?" : Goto LI99
202)  If NO=25 Then OB1$(25)="A Burning Newsletter" : Print "The newsletter
catches light and burns with protest!" : OB#(24,1)=0 : OB#(29,1)=-1 : NL=1 :
Goto LI99
203)  LI99:
204) End Proc
205) Procedure BURN
206)  If NL=0 Then Print "You have nothing to burn ";OB1$(NO);" with!" :
Goto BU99
207)  If OB#(25,1)=-1 and NO=28 and BW=0 Then Print "You try to light the
web but it does not catch fire." : BW=1 : Goto BU99
208)  If OB#(25,1)=-1 and NO=28 and BW=1 Then Print "The web finally
catches alight and starts to burns fiercely!" : P(18,8)=14 : P(14,6)=18 :
OB#(28,1)=0 : Goto BU99
209)  Print "You do not succeed in trying to burn ";OB1$(NO);"!"
210)  BU99:
211) End Proc
```

So there you have it, save it, now position the cursor at the beginning of line 17. Hold down The SHIFT Key and hit F5, this will then prompt you for the file that you saved your map data out as, select it. Wait a few seconds while AMOS processes the data and then when it is finished, save it again. Hit F3 to Indent and perform a Syntax Check. Correct any errors, save it if you need to and then RUN IT! have fun playing around there is one puzzle built in plus some hidden tricks as well. (You won't be able to get to all locations until Newsletter 8)

Well we have certainly got allot of program to discuss! Some of the lines are pretty much self explanatory, so I will not explain what things like End Proc mean or do. Well, As they say "I guess we had better take it from the top". (Who the hell are "THEY" anyway?)

Well here we go with the prodigious task of explaining the mess that you just typed in, you did just type that in didn't you?

Ok well Lines 1 - 6 do the initial setting up of the screen, setting the variable buffer to 100K to make sure we don't run out of memory, DIMensioning the Variables and declaring them as GLOBAL Variables. Line 3 sets up some important default Variables, they are out lined below.

P=22 This is the number of locations in the Adventure.
LO=29 This contains the number of Nouns(Objects).
NN=LO Same as above.
NV=13 This contains the number of Verbs.
CL=4 This always contains your current Location.

When writing an adventure always write down each new Variable that you start using and write next to it what it is used for, because after writing 500 lines or more of code, leaving it for a week or more and coming back to keep going you will be hopelessly lost in a maze of mixed up variables.

OK, line 7 calls the procedure SETUP which starts at line 15, lets now examine this procedure. As you can see it contains all of our data for the adventure, first there is our Map Data, then our Verb Data, then our Noun Data. When typing text into data statements you must put "" around each element otherwise AMOS will treat it as program code. If you are going to add more commands, which you will have to, make sure that you place them directly after these ones, start a new line with a Data command and list them after each other. The only exception to this rule is commands that only require a Verb only, Like QUIT or HELP etc. They must get inserted directly after then "Q" in these data statements. If you do this you must also go to line 3 and change NW= to reflect the new amount of Verb only words. So if you add two more Verb only words then you will change NW to 8. If you don't make these changes you will get some unpredictable - But stupid results!

After the Noun Data for the directions you will notice line 20, this is a dummy object that allows you to have ALL in your adventure eg GET ALL, DROP ALL etc. The rest of the Nouns follow, in this adventure I have used the following system for the Nouns/Objects.

Data "PILLOW",7,0.9,0,"An Embroidered Silk Pillow"

First we have the ACTUAL Noun-PILLOW This is of course the word that the adventurer would type in to interact with this object. This is stored in the Array NO$().

Next we have the location that this object is to be found at in our adventure, refer to the last newsletter for more information on this. This is stored in the Array OB#(?,1).

Next we have the WEIGHT of the object, in this case it weighs 0.9 KGs. This is used to place weight restrictions on how much your adventurer can carry. This is stored in the Array OB#(?,2).

next is a 0, this can either be a 0 or a 1, this dictates wether an object can be picked up or not. If it is set to 0 then you can, if it is set to 1 (Like the Spiders Web-Line 24) then you cannot. This is mainly here so that you can have buttons and obstacles in your adventure that cannot be moved, I mean who ever heard of you being able to pick up a button? This is stored in the Array OB#(?,0).

Lastly we have the small description which is displayed when you enter a room or when you do an Inventory listing. This is stored in the Array OB1$().

If you want to add values to objects then simply Dimension OB#(?,?) to another element like so...DIM OB#(LO,3). Then in your read Data statements simply add ,OB#(X,3) after the other Arrays so your line would look like so..Read NO$(X),OB#(X,1),OB#(X,2),OB#(X,0),OB1$(X),OB#(X,3). You must make sure that you then add you new value AFTER the SMALL DESCRIPTION in the data statements, simply add a , and the value like "A Round Donut",5. With 5 being the value, and of course a zero is quit legal.

Lines 8 to 13 contain the main control loop, first we call the PARSER Procedure which waits for your input, processes it and then returns back to line 10. Line 10 then calls the BRANCHER Procedure which contains some error checking and of course sends program control off to the relevant Verb Procedure. Once the called Verb procedure is finished, program control returns to line 11. Lines 11 and 12 contain tests for certain conditions, line 11 tests for the Variable SD=12. If this condition is met then the programs control exits the loop and goes to line 14 where you are told you have died, the program then waits for a keypress and then exits. Line 12 tests to see if the variable NL>0, this means in this case that you have lit the newsletter. If this is true then the variable NL is incremented, another test is then

Screen Effects!

By Brett George

If you are programming anything, be it a game or a demo, one of the key factors is presentation, the user wants to be impressed, and that is what you try to aim at. If you want to get a desired atmosphere into a game first you must choose the music, then the graphics have to be up to scratch as this also set the scene. After you have these two important factors you, the programmer, are in control and must use the Amiga to its full capacity.

Type the program below into Amos -:

```
Close Editor
Screen Open 1,320,266,32,Lowres : Flash Off : Cls 0
Screen Open 0,320,266,32,Lowres : Flash Off : Cls 0
Screen 1 : Load Iff "Amos_Data:Iff/Amospic.Iff",1
Screen 0 : Get palette (1) : Hide
Screen To Front 0 : B=40
AutoBack 0
For A=0 To 176
  If A/2.2=Int(A/2.2) then Dec B
  Get Bob 1,1,0,A To 320,A+1
  Get Bob 1,2,0,A+15 To 320,A+B
  No Mask
  Paste Bob 0,A,1
  Paste Bob 0,A+2,Vrev(2)
Next A
Screen Copy 1 To 0
Screen Close 1 : Erase 1
```

Line 1 : This frees more memory for uses with only 512k
Line 2-3 : Opens two screen. Screen 1 is for the full picture (for grabbing sprites). and screen 0 is for the final output.
Line 9 : This changes the height of the roll every now and then.
Line 10 : Grabs the next line from screen 1
Line 11 : Grabs the roll from screen 1
Line 12 : Makes colour 0 visible rather than transparent
Line 13 : Places the next line in before the roll
Line 14 : Places the vertically reversed roll in front of the latest line.
Line 16 : Places the finished picture in full view.

If you want your own picture to be shown change line 4 to -:
Screen 1 : Load Iff fsel$(""),1 See if you can improve that program or even make it roll back up the screen. With a little imagination nothing is impossible.

The next example is simple but effective -:

```
Screen Open 0,320,256,16,Lowres : Flash Off : Hide
For Z=1 To 15
  Colour Z,273*(Z+1)
Next Z
Cls 0 : X=15
For A = 0 To 20
   For B = 1 To Rnd(60)
      Plot (Cos(B)*X)+150,(Sin(B)*X)+120
   Next B
   If A<15 Then Ink A
   Add X,10
Next A
Shift Up 4,1,15,1
Wait Key
```

Line 2-4 : Sets up a smooth set of colours, from Black to White.
Line 8 : Draws one dot of the circle, placing is set by variable B and X determines the size.
Line 10 : Changes to next colour
Line 11 : Adds ten to variable X
Line 13 : Rotates the colour

If you would like to travel backwards change line 13 to - Shift Down 4,1,15,1
For a different effect Altogether insert a new line at the start and type - Degree

The final example is a very simple trick but looks very good. Type this into Amos -:

```
Screen Open 0,320,256,32,Lowres : Flash off : Auto View Off : Hide
Load Iff "AMOS_DATA:IFF/Amospic.iff"
For A=0 To 255
   Get Bob A+1,0,A to 320,A+1
Next A : No Mask : Cls 0 : View
For A=1 To 255
   Inc B
   For C=B To 256
      Paste Bob 0,C,A
   Next C
Next A
Erase 1
```

Line 3-5 : Grabs every line of the picture as a Bob.
Line 10 : Places the same bob from variable B to the bottom of the screen.
Variable B has one added to it at the start of the loop so the picture will form rather than just seeing all the line. Once again if you would like to see your own picture like this change line 2 to - Load Iff Fsel$("")

If you have any interesting effects that you can think of don't just think about them make them possible via Amos. I think the best way to learn about Amos is to just experiment with the commands. If you make anything interesting put them into your next Amos project, as it is the little touches which makes a game or a demo really sparkle.

Thanks go to Brett for sharing this interesting article with us. As you can see, anyone can submit an article, it need not be long (Like mine!) But as long as the information is correct and any examples are bug free then give me a call!

MUSIC MAYHEM

With SAUSAGE

Hello Again.

Thanks to everyone who wrote and commented on the article. Due to the popularity of the article, it will be continuing on as a series to help beginners to get to grips with the effects and techniques in Noisetracker. (By the way, Noisetracker is available on AA-21). This month we will be looking at starting a tune and using the position gadgets to produce a decent sized song. Also we will begin looking at the effects labels.

THE BIRTH OF A TUNE

To start a song, it is advisable to create what is commonly called a 'click track'. This is beat of straight crotchets that just simply provide a guide to the speed (tempo). This allows you to be kept in time with the beat. Firstly, let's load in a hihat from a st-01 disk. Press the space bar to enter the edit mode. Go to channel 1, press F2 to select the middle and high octaves, and place a B-3 note (key U) on every fourth line, ie. 0,4,8,12,16,20,....,60.

Press right Amiga to hear the click track. Not very impressive but it is only for getting another rhythm channel set up and not getting out of time. At this stage, I usually like to set up a drum track or bass line while the click track is running. To do this, Load in the required instruments, whether it be drums or basses, and press the right shift key to start record mode. Let the pattern scroll once and be ready to start playing when the record goes back to line 0. The click track will guide you. Once you have your next channel set up, you can erase your click track because the drum sequence or bassline will act as a click track from here on.

PUTTING TOGETHER A SONG

Looking at the song we made from last time (single pattern), we can make changes and save it as a new pattern. Using the Function keys F3 to F5, we can copy individual channels or patterns to reduce the labor of writing every track ourselves ie. copying the drum sequence to every pattern in our song to save ourselves re-writing it every time.

Firstly, here is the entire set of block editing functions:

Shift + F3 - Cut the current track (channel)
Shift + F4 - Copy the current track
Shift + F5 - Paste the current track

Left Alt + F3 - Cut whole pattern
Left Alt + F4 - Copy whole pattern
Left Alt + F5 - Paste whole pattern

To change the current pattern,

Left Alt and Right Cursor - Choose the next pattern
Left Alt and Left Cursor - Choose the last pattern

So lets load in the tune we made from last time and we will make a song with a length of four.

1) Load in the tune.
2) Press Left Alt + F4 to copy the pattern
3) Press left Alt + Right cursor to go to the next pattern
4) Press left Alt + F5 to paste down the pattern
5) Make any changes you wish, to the new pattern
6) Go to step 2 until you have a fourth pattern

Now to make a song....

1) Go to the POSITION gadgets
2) Select 4 with the LENGTH up gadget
3) Select POSITION 1 and PATTERN 1
4) Select POSITION 2 and PATTERN 2
5) Select POSITION 3 and PATTERN 3

We now have a four patterned song (positions 0 to 3). Click on play or press the right Alt key to play your song.

THE EFFECT PARAMETERS

Let's look at a standard tracker line.

```
D-206000
```

Breaking it up into parts:
D-2 is note D of the second octave.
06 is sample (instrument) 6.
0 THIS IS THE EFFECTS NUMBER!!
00 Value of Effect

LETS DO IT - VOLUME:

First effect and probably the most popular effect is the 'C' parameter. This is VOLUME. Here is an example of it's use...

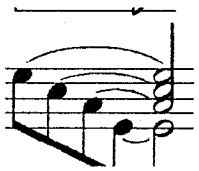
```
D-2 4000
--- 0000
--- 0000
F#2 4000      (Using instrument 4)
--- 0000
G-2 4C20
F#2 4000
```

In this example, the notes D and F# are played at default volume set by the up and down volume gadgets. But the note G is played at volume 20 (hex) which is around half the usual volume of an instrument (default is hex 40). When the next note, F# is played it returns to the default volume set with the gadgets. Yes that's great but what are the practical uses of changing instrument volumes? Below are three examples...

```
C-3 2C40    C-3 2C02    G-2 2C40
--- 0C35    --- 0C05    --- 0000
```

Continued On Page 8 ⟶

--- 2C30	--- 0C10	F-2 2C30
--- 0C25	--- 0C15	G-2 2C30
--- 2C20	--- 0C20	--- 0000
--- 0C15	--- 0C25	F-2 2C20
--- 2C10	--- 0C30	G-2 2C20
--- 0C05	--- 0C35	--- 0000
--- 2C02	--- 0C40	F-2 2C10
		G-2 2C10
FADE OUT	FADE IN	ECHO



You can easily achieve other great FX using the volume command 'C' (we'll call them commands from here on).

TEMPO:

This command (F) selects from speed 0 to F. It simply goes like this...

G-2 7F06

This plays note G in octave 2 using instrument 7 and sets the music speed at 6. The tempo (speed) can be changed at any time during a song.

LED FILTER:

We all know what that is don't we? For those not in the know, the filter cuts out high frequencies in samples (instruments), causing sounds like cymbals, tinny sounds and so forth to be blunt or flat. This is indicated by your AMIGA power light being on. This is usually left off.

--- 0E01

Turns the filter and light off.

--- 0E00

Turns the filter and light on.

Note... Commands don't have to be accompanied by a note or instrument number. They can be used at any time or any point. That's it for this issue. Next time we will be looking at the following commands:

0 - Arpeggio
1 - Slide up
2 - Slide down
3 - Slide to next note
4 - Vibrato
A - Slide Volume
B - Jump Tracks
D - Pattern Break

So experiment with the volume , filter and speeds when starting your own tunes and remember to create a click track to help you along. By the way, Anyone who is desperate for extra help may call me between 9.00pm - 9.30pm weeknights on (02) 545 3190.

Just a final note on musical programs in general. Many people buy programs such as Deluxe music or Sonix. Now, while these programs are very good for coding in sheet music, or for the more classic of us, they don't really keep up with modern day sequencers like Noisetracker or Mugician. The reason is that sequencers are very low level. You can manipulate the notes and add all sorts of effects. Another major difference is the musical score itself. With Sonix and the like, the music is represented with the classic staff,crotchets,semiquavers and so forth, whereas sequencers simply show the note and what the octave is, plus a few other details like the current effect. It's allot easier to read and is a breeze to edit. One final advantage is the ability to record your notes from the keyboard without the need for midi.

Keep Trackin'
Sausage.

SPITFIRE-A 10 Liner

Well unfortunately both Sausage and I were just too busy writing other articles to write another 10 liner for this newsletter but we were saved by Nigel Cawthorne! Yes Nigel has taken up the rewarding challenge of trying to write an entire game in just 10 lines and I might add he has done quite well! In this game you must simply put the gun sights over the enemy spitfires zooming past at high speed and blow them away! This sounds easy but believe me at the speed they are zooming past, its really not easy at all! Now that you have seen that anyone can do it, then how about some more 10 liners being written and submitted? Take up the challenge and write something interesting! I may even write an adventure in 10 lines!

Well here it is, be careful when typing this in.

```
1) Dir$="Extras:Sprite_600" : Load "FLIGHT/SPITFIRE.ABK" :
Load "SPACE/XPLOSION.ABK",6 : Get Sprite Palette : Double
Buffer : Curs Off : Hide : Flash Off : Cls 6
2) Get Bob 5,100,100 To 105,105 : Cls 0 : Ink 15 : Draw 105,75 To
265,75 : Draw 185,0 To 185,150 : Circle 185,75,40 : Ink 2 : Circle
185,75,20 : Ink 0 : Paint 185,75,1
3) Get Bob 6,105,0 To 265,150 : Cls 0 : Gr Writing 0 : Ink 6 : Text
7,7,"SCORE:"+Str$(H) : For B=0 To 4 : Bob B,-100,-100,5 :
Channel B To Bob B : Next
4) A$="L: I K1=0 J L; L X=RX-50; L Y=RY; M 50,-5,10; L X=-200; L
Y=-200 J L;" : B$="L: I K1=0 J L; L X=RX+50; L Y=RY; M -50,-5,10;
L X=-200; L Y=-200; J L;"
5) C$="L A=6; L X=160; Let Y=100; L RX=X; L RY=Y; B: L X=XM;
L RX=X; L Y=YM; L RY=Y; J B;" : D$="A 0,(3,6)(4,6); B: L X=Z(150);
L Y=220; " : Hot Spot 6,$11
6)  D$=D$+"M 100,-50,25;M -50,-100,10;M 100,-100,25;J B;" :
E$="A 0,(3,6)(4,6);B : L X=Z(170)+150;L Y=220;M -100,-50,25;M
50,-100,10;M -100,-100,25;J B;"
7) Limit Mouse 40,40 To 279,159 : Make Mask : Amal 0,A$ : Amal
1,B$ : Amal 2,C$ : Amal 3,D$ : Amal 4,E$ : Amal On : Wait 100 :
For T=1 To 1500 : For B=3 To 4
8)  If Bob Col(B,0 To 1)=-1 Then Boom : Amal Freeze B : Anim
B,"(7,5)(8,5)(9,5)(10,5)(11,5)(12,5)(13,5)" : Anim On : Wait 50 : Amal
9) On B : H=H+10 : Cls 0 : Text 7,7,"SCORE:"+Str$(H)
   If Mouse Key=1 Then Shoot
10) Next : Next : Bob Off : Sprite Off : Amal Off : Ink 11 : Text
120,100,"TIME UP-TRY, TRY AGAIN!" : Edit
```

Writing Games In AMOS
Part 2

WRITING GAMES
BY
The BLADE
Paul Ryman & Mark Stephens.

G'day all, we must apologize for the lack of new AMOS games lately. We have been too busy setting up our new BBS ... called what else but ... BLADE - BBS Supporting AMIGA, IBM And ..... AMOS !!! Now ONLINE 24 hours a day - up to 2400 baud Giving FULL UP/DOWNLOAD ACCESS to ALL USERS. Also... 2 minutes extra time for every minute you upload! (02) 957-3050 CALL NOW !!! Well, enough of the advertising (as if there isn't enough already in this world); Back to AMOS...

Last newsletter we discussed how to get started in writing your own simple arcade style game. In this article, we have given you some AMOS code demonstrating a rotating shooter (as seen in the classic arcade game, ASTEROIDS) controlled using AMAL. The shooter is created using trigonometric functions to draw up POLYGONS which are then 'cut out' and put in the sprite bank using the GET BOB command. There are 36 sprites making up the shooter, each being 10 degrees rotated relative to the previous shape. This number of sprites ensures that the animation is smooth without making it rotate too slow.

The program also uses 12 bobs as guided shots (sprite bank object number 37) which creates some interesting firepower effects. You may notice the similar structure of the game loop to what we described in the last newsletter. We use this structure for all our games; it seems to be fairly efficient and easy to use.

NOTE: This is not a complete game, but it does run. You can make this into a game if you like. It is merely a demonstration of a good way to use AMAL.

This program rotates an 'ASTEROIDS' type shooter controlled using the joystick. There is nothing to shoot, but it will shoot up to 12 shots at one time.

```
'
'    === ROTATING SHOOTER ===
'
'        By the BLADE
'
Degree : Hide On
Screen Open 1,320,255,8,Lowres
Screen To Back
'
'    CREATE ROTATING SHOOTER SPRITES
'
For I=0 To 350 Step 10
  Cls 0
  X1=Sin(I)*10 : Y1=Cos(I)*10
  X2=20+Sin(I-100)*8 : Y2=20+Cos(I-100)*8
  X3=20+Sin(I+100)*8 : Y3=20+Cos(I+100)*8
  Ink 1 : Polygon 20+X1*2,20+Y1*2 To X2,Y2 To 20,20 To
X3,Y3
  Ink 2 : Draw 20+X1*2,20+Y1*2 To 20,20
  Inc SHAPE
  Get Bob SHAPE,0,0 To 41,41
  Hot Spot SHAPE,20+X1,20+Y1
Next I
'
'    CREATE SHOT SPRITE & STARS
'
Ink 7 : Draw 50,50 To 52,50 : Draw 51,49 To 51,51
Get Bob 37,50,49 To 53,52
Cls 0 : Palette 0,$930,$F30,$FFF,$CCC,$AAA,$888,$D7E
For I=1 To 90
  Ink Rnd(2)+4 : Plot Rnd(320),Rnd(255)
Next I
'
'    SET UP BOBS & AMAL
'
Double Buffer
Screen To Front
Synchro Off : Bob Update Off
Bob 13,160,127,1
Channel 1 To Bob 13
Amal 1,"Anim 0,(R0,1); Beg: Pause; Jump Beg"
AML$="Beg: Pause; If RA Jump Shot; Jump Beg; Shot:
Let RA=0;"
AML$=AML$+"Let R1=RX/2; Let R2=RY/2; Move R1,R2,10;
Move RX,RY,20;"
AML$=AML$+"Let X=159; Let Y=126; Jump Beg"
For I=1 To 12
  Bob I,159,126,37
  Channel I+1 To Bob I
  Amal I+1,AML$
Next I
Amal On : Amreg(1,0)=1
'
'    THE MAIN GAME LOOP
'
Do
  Bob Clear
  Synchro
  Bob Draw
```

Continued On Page 9 ⟶

```
If Jright(1)
        If I Bob(13)>1 : Amreg(1,0)=Amreg(1,0)-1
        Else
          Amreg(1,0)=36
        End If
      End If
      If Jleft(1)
        If I Bob(13)<36 : Amreg(1,0)=Amreg(1,0)+1
        Else
          Amreg(1,0)=1
        End If
      End If
      Amreg(0)=Fire(1)
      Amreg(23)=Sin(Amreg(1,0)*10-10)*165
      Amreg(24)=Cos(Amreg(1,0)*10-10)*120
      Screen Swap : Wait Vbl
    Loop
```

This program communicates to the AMAL procedures through the AMREG (AMal REGister) command. For example, AMREG(1,0) [ie. Local Register R0 for AMAL channel 1 - Control of shooter] is used to change the rotated position of the shooter; this may take a value from 1 to 36 which refers to the corresponding sprite bank object number.

Also, AMREG(0) (Global Register RA) is a test for the joystick button press. AMREG(23) and AMREG(24) (Global Registers RX and RY) is used to set the initial direction of the guided shots. Basically the main advantage of using AMAL registers controlled from outside the actual AMAL procedures is that you can do more. It is easier to test Joystick controls from outside of AMAL, for instance (As a lot of AMOS user we know have already found out). Also, to do complicated calculations using SIN and COS functions, for example, is a thousand times easier.

They say you can write an entire arcade game in AMAL, but this would be a living nightmare. Oh well, don't believe everything you here... All our games use AMAL but only to a certain degree. It is extremely fast and easy to use, but lacks in versatility. That is why you will probably never see a fully functional game written in 'Pure' AMAL. Anyway if you do write one, we would love to see it.

Anyway, keep those AMOS games rolling into the PD library (and please, if you are a proud owner of a modem, please don't forget to upload it to our BBS). See ya....

...The BLADE...

Important Note About AMOS V1.23

Well AMOS has been updated yet again! This time we have seen addition of some new commands plus the fixing of some bugs! AMOS 1.22 seems to have disappeared into the black nothingness. First of all let me make one thing perfectly clear, If you wish to buy the compiler, the AMOS 3-D extension or the TOME extension you have to have version 1.23! None of these accessories will work on any of the previous versions. So you must update to V1.23 before you can consider using any of these three new extensions.

Ok, lets talk about new commands first. We have seen the inclusion of INTERLACE Mode! Yeah, I hear all those Video Titlers out there saying! (Hey guys, so far I haven't seen a single video titler yet! Where are they?) Yes, thats correct you can now invoke the jitters, er sorry interlace mode when opening screens. But thats not all, other new commands allow you to detect whether your program is running on a true NTSC(200 lines) or a true PAL(256 line) machine. This is important for screen opening etc!

Not only do we get more new commands, but we get some fixups as well. Firstly we see the INPUT Bug fixed. You may have come across the problem, input works fine until you put it into a loop, then AMOS locks up. Well thats been fixed. Also the Sequential & Random filing routines have had a facelift and fixup.

Some of you who have been ordering BA36 will have already received V1.23, If you want to get V1.23 simply order BA36 from the library. This is the last update till after the compiler is released and update is needed. Just a quick note about the updater, there is actually two updaters on the disk...V1.21 & V1.23. You must first update to V1.21 by booting the updater disk and following the prompts. Then once updated, load the newly created AMOS V1.21, load and run the V1.23 Updater AMOS program off the updater disk. When you run this updater it sits there for about 1-2 minutes looking as if it is not doing anything at all. This is not the case, it is simply clearing out about 145k of memory, once this is done it will continue the updating process.

AMOS BBS LISTING

PREDATORS BBS - AMOS HOME BASE 604-6644
24 Hours. 2 Lines. 2400 Baud. BIG Amos Downloads.
BLADE BBS - Run By The BLADE 957 3050
24 Hours. 1 Line. 2400 Baud. BIG Amos Downloads.
FUTURE DIMENSIONS BBS - QLD AMOS BBS 07 208 5004
24 Hours. 1 Line. 2400 Baud. Large Amos Section
ISLAND SOFTWARE CLUB BBS - VIC AMOS BBS 0374239931
24 Hours. 1 Line. 2400 Baud. $10.00 Mem. NEW AMOS section

The ARCH-*Into The Unknown*

Welcome to the Arch,

Well, there really are amosites (sorry Paul) out there who are interested in Adventure and roleplaying! Thank you to those of you who wrote to me with suggestions and ideas for this club. I will try to cover as many of them as space permits, but keep them coming! I can't emphasize enough how much this club will rely upon your input.

The first thanks go to Paul Carpenter. Several people had good suggestions for the name of the AMOS RPG and Adventure Club, but I thought Paul's was the pick of the bunch. (It's the shortest anyway!) ARCH stands for "AMOS Roleplaying Club Headquarters".

Just because Adventure has been dropped from the title doesn't mean we don't cover text adventures any more. In fact As you can see, a large part of this newsletter has Neil Miller's second instalment (or should I say volume?) on writing text adventure games. This is a huge effort and not only does his article give you the basis for writing your first adventures, it is also a good example of how proceduralise your programs to make them neater and more efficient.

Neil must have put in a lot of late nights on this so let's see some response in the form of feature rooms and actual adventures. Start small and work your way up. Send in your early efforts for people to comment on and test.

On playtesting, several people have now shown an interest in playtesting other peoples prototype games. What we now need are a few more games to be tested! What I would like to do would be to take a few peoples prototypes and place them on one disk which could be distributed to playtesters for comment. I would ask people to send me blank disks (or disks with their prototypes) in a postpak and I would retum the disk with a compilation of prototypes to be tested. WARNING: I do not have enough games to make up such a disk at this time so if you send me a blank disk I cannot guarantee it will retum soon. It is probably best to wait until next newsletter unless you have something for me to test.

There was considerable discussion about the SCF (Standard character format) I discussed in the last newsletter which is a healthy thing. Please remember this is only designed as a tool to allow you to transfer characters from one RPG to another. The last thing I want to do is restrict all those vivid imaginations, by accepting the SCF as the format they have to use inside their own programs.

While most of us were just thinking about this, Jonathan Corbett has actually done something! He has sent me an example of a character generator using some of the ideas in the last newsletter. Thanks Jonathan and keep up the good work!

The area of SCF which generated the most interest (aside from me forgetting sex!) was skills. There seem to be two schools of thought in this area. One is that the skills should be stored as a text string, Eg "pickpockets/85". This would mean that any skill could be added to the original ones, we would not be restricting ourselves to a fixed set at the start. Also with AMOS's string handling functions searching and manipulating these strings would be a breeze.

The second school of thought is that the skills should be stored as numbers only, either in a fixed size array with an entry for each skill or as a two number entry, eg: 10,60, where the first number refers to the skill and the second refers to the skill level. This would restrict the number of skills that could be passed to each game and would require a table defining each skill beforehand. It would how ever avoid things like spelling mistakes or ambiguities, eg "pickpockets", "pickpocket", "pick-pocket" or "pick pockets"? All mean the same thing and all are quite legitimate spellings.

After initially voting for the string format I think I have been swayed to opt for the numbers. There is no way we can predict all the skills people will devise so lets not try. If we have a base of general skill areas rather than individual skills we might find a more usable format. Ie Skill 12 might be "Slight of hand", this wold cover such things as "pickpockets", "conceal item" and "card sharp", while "Manipulate Mechanical" would cover such things as "Pick locks", "Set and Disarm traps" and "Phaser repair". Please let me know your opinions.

Just to wrap up and answer a few questions:

There are non-programmers out there who want to contribute ideas story lines etc. to programmers.

There is no official PD for the ARCH. Anything I receive, excluding playtesting material I will forward to the Aussie PD.

Some people seem keen to start up the open ended role playing system. (That is the system where we have the 'rules' on one disk and people design their own scenarios and swap them with other players.) Any ideas you heave on that front would be welcomed.

Paul Carpenter made another suggestion to design a type of 'Gladiatorial Competition" where anyone could enter a character (in SCF?) to a type of all in brawl where the survivors are rewarded and fight again next week/month. This could become a regular feature but we require some volunteers to do come coding!

There were more suggestions and ideas proposed, several people wrote to me telling me about ideas they have for different RPG and Adventure games and scenarios. Keep the ideas coming and lets start to see some more actual programs, both prototypes and completed ones appear, both here and on the public domain.

Contact: The ARCH C/O Chris Whale 35 Union St. Dulwich Hill. 2033

AMAL EDITOR TUTORIAL PART-1

It always amazes me to see just how many programmers write strings and strings of amal. It's a cluttered mess and looks awful. There is a much better way of creating and testing amal programs, and that's using the AMAL EDITOR.

So often you see something like the following in a program...

```
screen open 0,320,200,16,lowres
bob 0,160,100,1
a$="A: L R2=5; M 100,0,100; B: I X<> R2 J C; P; J B; "
a$=a$+"C: L X=Z(320); P; J B; "
channel 0 to bob 0
amal 0,0
amal on
Do: Loop
```

In the editor, writing the strings is just the same except that you don't need a$,a$+,a$+ in the beginning of each line, you can look at each channel separately, there's a step by step function so your programs can be debugged easily, and you can even create the wavy patterns for bobs and sprites with the path definer. All these things can give you the ability to write larger and better code. In FLAME (available on disk AA-24), the amal programs work out to be around 12 printed pages in length. To do this in AMOS would be an incredibly laborious task and debugging would have been almost impossible.

In this tutorial, I will guide you into starting with the editor and getting comfortable with all the functions. This is not a necessarily a tutorial for beginners but it is for those who can program fairly comfortably in AMAL.

FIRST PROGRAMS.

Most people's first reaction to seeing the editor is - 'how can I have my AMOS code running at the same time?'. The answer is you don't need to. To start writing an AMAL program, you have to write a small piece of AMOS code in the ENVIRONMENT channel. This is to set up screens, bobs, sprites and to set up channel configurations (screen offset, bob, sprite, etc).

This is located as a gadget left of the channel 00 gadget. Clicking on this will make that area active.

ENVIRONMENT PROGRAM

Grab your extras manual and turn to page 26. Here is listed the entire set of AMOS commands that can be used in the environment channel. However, the command CLS has been omitted from the manual but can be used. Once your setup program has been written in the environment channel, you can then write your amal programs to each channel, making sure that each channel you use is assigned to an object. ie. Channel To ***** 0,1 Commands such as Amal 1,A$ or Amal 5,5 are not needed as the editor will do this automatically.

Your environment program will firstly set up the screen and then the editor will begin to play the amal channels.

LOADING FILES

All the most common types of files can be loaded in. These are...

IFF pictures
SPACKed picture banks
BOBs/SPRITEs
AMAL code (only the if made in the editor! *)

* NOTE: The code written in the editor still contains the Environment program code when loaded into AMOS despite the fact that it is not needed. Once you load this AMAL bank file into AMOS, it will destroy The Environment section meaning that you can't rip the bank out and load it back into the Amal Editor. ALWAYS KEEP A BACKUP OF YOUR AMAL FILES!!!

The order to load files into the editor is this:

SPACKed Picture banks,
your BOBs,
and finally your amal code (if any).

A better method is to do to direct mode before starting any type of work with AMOS and erase all in memory. Load in your picture that you will need, and SPACK it into a bank. Then load in your BOBs. Save the Banks in one file. This way, you can load in your picture and bob banks in one go, then your Amal programs.

Try this method, and then load in your banks. Click on the EE button (the environment editor) and type the following example.

```
unpack * to 0 '* being the no# of bank where you stored your pic.
double buffer
bob 0,50,100,1
bob 1,100,100,2
channel to bob 0,0  'Note that the commands here are
channel to bob 1,1  'used differently
Click on 00  (channel 0) and type the following.
A:
Move 200,0,100;
Move -200,0,100;
Pause;
Jump A;
```

The next channel will contain basically the same code as in channel 0, however we don't need to re-write the text. Move the cursor to the second line (A:) and press F6. This places the beginning of the text we wish to grab. Move the cursor down 5 places and push F7. The text will become highlighted. Press F9 to cut and immediately press F10 to stamp it back down.

Now you may either click on the 01 gadget to get to channel 1 or you can press the left Amiga key and right cursor. Now we are in channel 1. Press F10 to stamp down the text again. We will have two identical copies only this time change lines 2 & 3 from:

```
Move 200,0,100;
```

To:

```
Move 200,100,50;
Move -200,-100,50;
```

Press F1 to view the program.

This is just a brief intro into the Amal Editor. Next edition, I'll be going into problems with long code and using the other functions of the editor such as the play editor, where we will be creating a 7 bob attack wave, and the debugger, the function that lets us look at our programs frame-by-frame and the variables.

Till then...

Cheers,
Sausage

Whats New In PD!

Well as you can see we have changed to the new format of listing PD disks. From now on we will be only listing the new additions to the libraries, but every 3 editions I will publish a full listing. Reason for this? Yes there is, mainly because the listings are becoming to long and taking up valuable space that could be taken up by articles. Just like this newsletter, I'm sure you will agree that more articles makes more sense. I will now also try and keep the newsletter at an ODD number of pages, this way the order form is on a page by itself and you don't loose anything on the reverse side when you cut it out. But then thats progress for you!

Well I won't dwell on the fact that the number of Aussie submissions is at an all-time low, but it is pretty disappointing. End of subject.

Well look at the number of new disks that have arrived from Britain! Heaps of new disks to choose from with heaps of games and source code to look at! The Aussie submissions have got some good stuff in there as well with my favorite being the Mandlebrot Generator by Bob Devries, well done Bob! Of course the games that were added are of top quality as well, there is of course a new game from DCAT, which is a version of Blackjack. Nothing from the Blade this time but they are working on a new car racing game which I saw a preview of just the track moving and it is going to be hot! So watch out for it in the next listing. Another busy beaver was Peter Craven, Peter has submitted before but he is going at it like crazy and has two submissions in this listing. They include Roulette AA69 and Poker Machine AA74.

One of the best to come out of Britain in this batch is on disk BA115-Balloonacy. This is my favorite out of the British collection. Benson has been busy writing demos and you can find his latest effort on BA125. Disk BA123 of course contains the TOME Extension demo game, TOME should be available soon! Another favorite disk is BA149, this contains a variation on the game QBert called Gobbit as well as a neat version of Pontoon. Theres heaps to look at so get stuck into it!

Not much room left so I had better go. Cu all later and keep the Submissions coming in!

AUSSIEDISK PD LISTING

AA69...ROULETTE. Excellent version of Roulette.
AA70...CORRUPTION DEMO. Another great demo.
AA71...DCAT-BLACKJACK. Another excellent game from DCAT.
AA72...TRONICS 2091. Great Lightcycles game with plenty of different backdrops.
AA73...ELECTRIC FILES DATABASE. NINO CAD. Small database program that can be password protected. Drawing program.
AA74...POKER MACHINE. Great electronic poker machine, you can edit the odds etc.
AA75...MANDLEBROT GENERATOR. Great Mandlebrot generator with plenty of example pictures. Saves variables with picture.
AA76...DEMOS-GAMES. Heaps of games and a demo. Includes Rally-X, Paranoia RPG Character generator + HEAPS More!
AA77...ASSTD GAMES. Lots of programs, games etc.
AA78...ASSTD PROGRAMS-GAMES. Includes level 1 of Starstruck + other interesting programs. Includes original music out of Road-Hog.
AA79...MUSIC PLAYERS+MUSIC. 2 different Music players + 5 songs.
AA80...ROADHOG. VERY HARD car dodge-em game, very simple! All you have to do is miss the cars!, good hard addictive fun!

BRITISH PD LISTING

BA111...BLADERUNNERS MUSIC DISK 1.
BA112...PANTHORUS MEGA DEMO 1-DISK1. Great two disk demo.
BA113...PANTHORUS MEGA DEMO 1-DISK2. Part two too above.
BA114...PREDATORS DEMO. Excellent demo.
BA115...BALLOONACY. Just like the old game Bomber but with plenty of twists and great graphics.
BA116...J.P.M. SOUND DISK. Heaps of sound samples.
BA117...MUSIC + PROGRAMS. Asstd programs + some music.
BA118...FUN SCHOOL 3 SPRITE SETS. All the Bobs/Sprites from the popular educational series Fun School 3.
BA119...E. CAD DEMO V1.2. Working version of a Printed Circuit Board editor/designer.
BA120...AMOS MUSIC PLAYER. Yet another music player!
BA121...NIK WILLIAMS DEMO. Another demo.
BA122...AMOS JUKEBOX. Music demo.
BA123...DEADLINE. The first demo of the soon to be released TOME Map extension. Fully playable game.
BA124...BOB MANIACS SYNTEX DEMO. Superb Demo with COMPLEX mathematical patterns using bobs.
BA125...BENSON DEMO II. Better than Bensons first demo!
BA126...DREAMERS 3. Mag-On-A-Disk. Text not too good, but the Routines/Graphics/Icons are very good.
BA127...CYBORNETICS DEMO II. Another great demo!
BA128...MACC COMPUTER CLUB DEMO DISK. Demo for a club with animation sequences.
BA129...SYNTEX MUSIC DEMO DISK 1. Musical Demo.
BA131...ARMAGEDDON SYNTEX DEMO. Another demo.
BA132...FAMILY HISTORY DATABASE. Geneology Program.

Continued On Page 11

BA133...MANDLEBROT GENERATOR. British Mandlebrot Generator. (Ours is better).
BA134...SPECTRAPAINT V2.0. Excellent Painting Program.
BA135...SIMON SAYS. SIMON MATHS. Educational programs-Very good.
BA136...MATHS CALCULATOR. VENDRED 1. Educational program. Vendredi is an excellent appointment scheduler.
BA137...TILE TRIAL. Good Sliding Tile Puzzle.
BA138...SPANISH TUTOR. Teaches you Spanish.
BA139...MASTER RACE. Excellent Quiz-Space Game.
BA140...AMOS PROGRAMS #7. Full of programs.
BA141...AMOS GAMES #1. Numerous games.
BA142...PAIR CRAZY. Card Matching Games. Like Concentration.
BA143...CYBERNETICS MUSIC DEMO #1. Music Demo.
BA144...ZENN SOUND BOX. Music Demo.
BA145...DREZ 4.7. Light cycles game with 4 players. Some computer controlled players.
BA146...FRUIT MACHINE/FRUIT CRUSH. Poker Machine + Arcade Game.
BA147...AMOS GAMES #2. More Games.
BA148...DEMOLITION MISSION. Variation on the game "Bomber".
BA149...GOBBIT/PONTOON. Version of Q-Bert and Pontoon.
BA150...AMOS PROGRAMS #8. Heaps of procedures/programs.
BA151...UNDERSTANDING AMOS/BOBALIZER. Tutorial on AMOS & Music Program.
BA152...AMOS PROGRAMS #9. More asstd programs.
BA153...MISSILE COMMAND. EXCELLENT Version of Missile Command.
BA154...AMOS PROGRAMS #10. More and More Programs/procedures.
BA155...PANTHORUS DEMO #2. Another great Panthorus demo!
BA156...RIPPERS DISK. Jam-Packed full of ST/NT Module rippers.
BA157...MUSIC DISK #25. 9 Modules.
BA158...MUSIC DISK #26. 10 Modules.
BA159...SLIDESHOW PROGRAM. Half Decent Slide Show Generator.
BA160...AMOS QUIZMASTER. Quiz generating/playing program.

Ordering/Submitting PD Disks

Some new rules for submitting PD, if you buy the compiler and you are going to submit your program to the library, if you send a compiled version then you must also include the UNLOCKED original source code as well. This still allows people to examine and learn from your programming efforts. Because thats what the PD libraries are all about-Learning from other peoples examples.

The editor of the British newsletter was commenting in their latest newsletter about the popularity of the different "Categories" or PD in his collection. He put foward the opinion that the most popular "Category" was Games and Utilities, with Demos coming dead last. Aaron keeps track of the more popular disks and does a top 10 according to the orders for them. The Amos updater is number 1, and the highest a demo ever got on his list is position 7! While demos can be impressive, they serve no real purpose other than to impress and maybe learn a thing or two from. Now games have a long lasting appeal and of course Utilities speak for themselves as long as they are useful.

When deciding what to send into the library please keep in mind that we really don't need any more music, there are nearly 30 disks in the British collection and there are about 15 in ours. We really need some more games to show the British that we can write bigger and better games than they can!

So keep the submissions coming in, hopefully in the next newsletter we will have some more to list than we have at the moment. Thanks go to those who submitted over the last two months!

AUSSIE AMOS CLUB STATISTICS		
NSW---575	NT---11	500'S---1000
QLD---178	TAS---40	1000'S---99
VIC---261	NZ---1	2000'S---125
SA---127	USA---1	2500'S---18
WA---91		3000'S---5

AMOS AUSSIEDisk & British PD Disk Order Form

NAME:. . . . . .
ADDRESS:. . . . . .
CITY:. . . . . .
STATE:. . . . . POSTCODE:. . . . .
AMOS REGO NUMBER:. . . . . .
DATE ORDER PLACED:. . /. . /. . . .

AUSSIEDisk& British PD Pricing
1 DISK - $7.00
2 DISKS - &13.00
3 DISKS - $18.00
4 or More DISKS $5.00 Each

I Enclose $. being in the form of
Money Order( ) Personal/Bank Cheque( )
Made Payable To "AUSSIE AMOS Users CLUB"

CODE	DESCRIPTION	CODE	DESCRIPTION

TOME EXTENSION/AMOS 1.23 UPDATER/DOS-PRINTER EXTENSION ORDER SECTION

CROSS OUT
☐ Please Add A Tome Extension To My PD Order @$59.95 ea
☐ Please Also Add BA36-AMOS 1.21/1.23 Updater @$5.00 (You Need This!)
☐ Alex Grants DOS/Printer Extension @$8.00)

Note: $5.00 for BA36 only Applies If ordering TOME or 4 or more PD Disks from the library. It is not available for $5.00 by itself.

Send Order To: AMOS PD ORDER P.O. Box 253 Rydalmere, NSW 2116