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, let's talk about the present! As you can see this newsletter is bring 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 2-4 articles! So by the time you have read the entire series you should all be sufficiently proficient to rattle out your own fun and creative tunes for your games/educational programs and demos. In this article he delves deeper into the hot-keys and editing keys used in NickelTracker, 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 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 come 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. 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 have not heard from Mandarin yet so I 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 Library. It is tome that is 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 in - ASTEROIDS’ style of game. All that is basically missing is the enemy or asteroids, scoring and collision detection routines. A great simple game could not going to die, which is great! It is undergoing its name change 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 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 over 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

- PAGE 1...Welcome to The Newsletter
- PAGE 2...Contents
- PAGE 3...How To Write An AMOS Extension
- PAGE 4...Bloopers Corner
- PAGE 5...Handy Keyboard Shortcuts
- PAGE 6...About TOME-TOTAL Map Extension
- PAGE 7...Writing Adventures-Part 2
- PAGE 8...Screen Effects
- PAGE 9...Music Mayhem - Part 2
- PAGE 10...Spitfire-A Submitted 10 Liner
- PAGE 11...Writing Games In AMOS-Part 2
- PAGE 12...Important Note About AMOS V1.23
- PAGE 13...AMOS BBS Listing
- PAGE 14...The ARCH-Formerly The AMOS RPG Club
- PAGE 15...Tutorial On The AMAL Editor
- PAGE 16...AMOS PD Listing Updates
- PAGE 17...AMOS AUSSIE Statistics
- PAGE 18...PD/TOMe Order Form

THIS SPACE INTENTIONALLY LEFT BLANK FOR WRITING DOWN PHONE NUMBERS YOU WOULD NORMALLY FORGET!
**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 its advanced graphics and sound facilities, but what if that is 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 words 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. A Default command.

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 are about program etc. as follows:
A1: Intuition library base address
A2: Graphic library base address
A3: Assistant library base address
A4: 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: Veloce message address
A2: Address of screen reset routine
A3: Address of quit routine
D0: SH -> OKERROR
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. 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 none.

All that above information can be found in the Musica file, but note where I have made corrections. Always check that Francois do what Francois say. OK, lets do our first bit of code:

```assembly
Hello: movem.l a4-a6,-(sp) ; never forget this!
moveq #3,d1 ; extension number 3
move.l #0,d4 ; no change bank routine
lea Tk(pc),a0 ; Token table address
leaHelDef(pc),a2 ; Welcome message address
leaHelWel(pc),a1 ; Welcome message address
leaIntRef(pc),a3 ; Intuition library base address
leaHelEnd(pc),a4 ; Screen reset routine
leaHelEnd(pc),a3 ; Quit routine
move.w #3,d1 ; extension number 3
move.w #0,d0 ; no errors
move.l (sp)+,a4-a6 ; feature registers always set
```

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

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

```assembly
Hello: ; do nothing
```

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

```assembly
Hello:
```

**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:

```assembly
dc.w Address of instruction-Tk, Address of function-Tk
dc.w "Instruction name","A"+$80,Var list,-1 (or -2)
```

**Addresses:**
These are 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 func-Tk,func-Tk

**Reservered var:**
V0 - Integer
V1 - Float
V2 - String

**Next you need:**
You need a 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:

```assembly
dc.w "Doobah","A"+$80,02,01,00,00,00,-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.

gc.w 1,func1a-Tk
dc.w "XXX","X"+$80,00,00,-2 <- one parameter
dc.w 1,func1b-Tk
dc.w "XXX","X"+$80,00,00,-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 d3, so you simply get them in reverse order.

eg. move.l (a3)+,d0 ; parameter 2

```assembly
move.l (a3)+,d1 ; 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:

<table>
<thead>
<tr>
<th>dc.w 1,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc.b 380,-1</td>
</tr>
<tr>
<td>dc.b Hello-Tk</td>
</tr>
<tr>
<td>dc.b &quot;Hello&quot;,239,&quot;2&quot;,-1</td>
</tr>
<tr>
<td>dc.w 0</td>
</tr>
<tr>
<td>; You must end in a zero.</td>
</tr>
</tbody>
</table>

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

He
tf: dc.w 11
dc.b "Hello There"
HM
W: 5c 27,"Y",48-11,"Hello V1.0",0
dj: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 "ARE YOU USING A BACKUP OF YOUR PROGRAM DISK"

V1.0"

and then load it. Now you have loaded the "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. Gurri (but that is more your problem than mine - my Hello constructor 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.

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

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

He
tf: dc.w 11
dc.b "Hello There"
HM
W: 5c 27,"Y",48-11,"Hello V1.0",0
dj: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 "ARE YOU USING A BACKUP OF YOUR PROGRAM DISK"

V1.0"

and then load it. Now you have loaded the "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. Gurri (but that is more your problem than mine - my Hello constructor 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.

Finally, here is our token table:

<table>
<thead>
<tr>
<th>dc.w 1,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc.b 380,-1</td>
</tr>
<tr>
<td>dc.b Hello-Tk</td>
</tr>
<tr>
<td>dc.b &quot;Hello&quot;,239,&quot;2&quot;,-1</td>
</tr>
<tr>
<td>dc.w 0</td>
</tr>
<tr>
<td>; You must end in a zero.</td>
</tr>
</tbody>
</table>

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

He
tf: dc.w 11
dc.b "Hello There"
HM
W: 5c 27,"Y",48-11,"Hello V1.0",0
dj: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 "ARE YOU USING A BACKUP OF YOUR PROGRAM DISK"

V1.0"

and then load it. Now you have loaded the "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. Gurri (but that is more your problem than mine - my Hello constructor 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.
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 applications for Adventure games being submitted in the next couple of months, then I will be very disappointed 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 need the article and are looking forward 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.

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 lack of converting that code to AMOS, and then making it work properly. That code below 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 the spelling. 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 “O” 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 AMOS adventure. (You 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 array. If you have more locations then simply change them to whatever you like

Set Buffer 50

1) Set Buffer 50

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem

Rem
105) GT4:
106) For i=1 To X: AQ(i)=0: Next I: Goto GT6
107) GT6:
108) Print "There is nothing here that you can pick up, perhaps its just decoration?"
109) GT7:
110) End Proc
111) Print "You feel silly trying to read;" "OB1$(NO);."
112) End Proc
113) Goto R99
114) Procedure DROP
115) Dim C(20) : W=1
116) Procedure STRING[S$,SL]
117) End Proc
118) Procedure QUIT
119) End Proc
120) GT8:
121) For I=1 To X: AQ(I)=0: Next I: Goto GT6
122) GT9:
123) For I=1 To X: AO(I)=0: Next I: Gala 13R99
124) DR4:
125) If X=1 Then Goto DR4 Else A-AZ-OBH(NO2); Print " -Dropped;": OBH(NO1)-CL: Goto DR99
126) DR5:
127) Next I: Goto DR99
128) DR6:
129) AOQ(I)=: X=1: Return
130) DR7:
131) Next I: Goto DR99
132) DR8:
133) End Proc
134) Procedure LOOK
135) Screen 1: Clw: Centre Border$(PS$CL): Curs Off: Screen 0
136) X=1
137) Print "You try to eat OB1$(NO); But Fail!"
138) If NO=2 Then Goto DR11
139) Print "But you have nothing to drop here!" : Goto DR99
140) Print "You do not succeed in trying to burn OB1$(NO);" : Goto DR99
141) For I=1 To X: AO(I)=0: Next I: Gala 13R99
142) Print "You try to light the OB1$(NO); BUT you have nothing to burn with!"
143) If NO=2 Then Goto DR11
144) Print "You have nothing to burn OB1$(NO); with!"
145) Procedure BURN
146) If I<X and I+20X Then Print ",;" : Add SL,2
147) STRING[OB1$(NO),SL] : Goto LI99
148) L1:
149) Print OB1$(NO);
150) If NO=23 and OB#(22,1)<>CL Then OB#(23,1)=0 : Print "The crystal is cooled by the water"
151) If X=1 Then Print 'But you have nothing to drop here!' : Goto DR99
152) Print "You try to use OB1$(NO);"
153) Next I: Goto DR99
154) Print "You try to destroy OB1$(NO);"
155) Next I: Goto DR99
156) Next I: Goto DR99
157) Next I: Goto DR99
158) Next I: Goto DR99
159) Next I: Goto DR99
160) Next I: Goto DR99
161) Next I: Goto DR99
162) Next I: Goto DR99
163) Next I: Goto DR99
164) Next I: Goto DR99
165) Next I: Goto DR99
166) Next I: Goto DR99
167) Next I: Goto DR99
168) Next I: Goto DR99
169) Next I: Goto DR99
170) Next I: Goto DR99
171) Next I: Goto DR99
172) Next I: Goto DR99
173) Next I: Goto DR99
174) Next I: Goto DR99
175) Next I: Goto DR99
176) Next I: Goto DR99
177) Next I: Goto DR99
178) Next I: Goto DR99
179) Next I: Goto DR99
180) Next I: Goto DR99
181) Next I: Goto DR99
182) Next I: Goto DR99
183) Next I: Goto DR99
184) Next I: Goto DR99
185) Next I: Goto DR99
186) Next I: Goto DR99
187) Next I: Goto DR99
188) Next I: Goto DR99
189) Next I: Goto DR99
190) Next I: Goto DR99
191) Next I: Goto DR99
192) Next I: Goto DR99
193) Next I: Goto DR99
194) Next I: Goto DR99
195) Next I: Goto DR99
196) Next I: Goto DR99
197) Next I: Goto DR99
198) Next I: Goto DR99
199) Procedure LIGHT
200) If OB#(24,1)<>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$(NO1): with what?": Goto L99
203) L99:
204) Procedure BURN
205) Procedure DROP
206) If NLS Then Print 'You have nothing to burn;" "OB1$(NO); with!": Goto B99
207) If OB#(25,1)=1 and NO=26 and BW=0 Then Print "You try to light the OB1$(NO); BUT it does not catch fire": BW=1: Goto B99
208) If OB#(25,1)=1 and NO=28 and BW=1 Then Print "The bomb finally catches aflight and starts to burn fiercely!": P(18,4)=14: P(14,6)=16: Goto D99
209) Print "You do not succeed in trying to burn OB1$(NO);": Goto LI99
210) End Proc
211) End Proc

So there you have it, save 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. If you have any questions then you will need to look at the hidden section below. (You won't be able to get to all locations until Newsletter 8)

Line 12 tests to see if the variable NL>0, this means in this case that you have lit one of the Nouns. There are two more variables that are used, SD and NW. If SD is not equal to 12 then the program will ask you the next move, otherwise the program will go to line 14, and the control of the program will be determined by the Verb that you add in the data statements, if you are a Verb only statement then the program will go to the verb procedure that is called. If SD is equal to 12 however the program will ask you the next move that you wish to perform, and if you say anything other than Verb only statements it will go to line 14. If SD is equal to 12 the program will ask you the Verb only statement, and if you type in a Verb only statement it will go to line 14 of the verb procedure that is called. This is stored in the Array OB#(NO,2).

If you have a Verb only statement that you want to add to the data statements, take it off the Verb only statement list, and insert it directly after the verb that you want to add. This is stored in the Array OB#(NO,3). This contains the number of Verbs. CL=4 This always contains your current Location.

When entering an adventure always write down each variable that you start using and write note of what it is used for, because after a week 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.

CL6=10 This is the number of locations in the Adventure.

NV=13 This contains the number of Nouns/Object.

NL=9 This contains the number of Verbs.

RUN ITI

Continued On Page 6 -
The text seems to be a mix of programming code and English text. It appears to be a part of a computer game or program development guide. The text contains code snippets and explanations for programming tasks, possibly related to adventure games or similar software. The code includes various commands and functions, and the English text provides context and explanations for these code elements. The overall content seems to be a detailed tutorial or reference guide for someone learning to program, potentially for a game development platform or similar software.
MUSIC MAYHEM

By Brett George

Hello Again.

Thanks to everyone who wrote and commented on the article. Due to the popularity of the article, it will be continued 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 a 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 hi-hat from a 512k 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 Amos 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

To change the current pattern,

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

So let's 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 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-2 4000
G-2 2C40
C-3 2C40
CTS 5000

This is YOUR EFFECT NUMBER!
00 Value of Effect

LET'S DO IT - VOLUME:

First effect and probably the most popular effect is the 'C' parameter.

This is VOLUME. Here is an example of its use...

D-2 0000
G-2 0000
F# 4000 (Using instrument 4)
C-3 4C20
F# 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 23 (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 about the practical uses of changing instrument volumes? Below are three examples...

C-3 2C40
C-3 2C02
G-2 2C40
- 0035
- 0035
- 0000

Continued On Page 8
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...

- **Amal** On
- **Amal** I+1,AML$
- **Channel** I+1 To Bob I

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 it 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's power light being on. This is usually left off.

Turns the filter and light off.

- **Amal** On

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 for this issue. Next time we will be looking at the following commands:

- **Amal** On
- **Amal** I+1,AML$
- **Channel** I+1 To Bob I

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 on weeknights on (02) 545 3190.

---

**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 challenge of trying to write an entire game in just 10 lines and I must say he 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 can do it, then how about some more 10 liners being written and submitted?

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 can do it, then how about some more 10 liners being written and submitted?

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

1) Dir$='Extras:Sprite_600' : Load 'FLIGHT/SPITFIREABK'
2) Get Bob 5,100,100 To 265,150 : CIs 0 : Gr Writing 0 : Ink 6 : Text 4) A$='L: 1 K1=0 J L; L X=RX-50; L Y=RY; M 50,-5,10; L X=-200; L

---

**Writing Games In AMOS**

- **THE MAIN GAME LOOP**

Well, enough of the advertising (as if there isn't enough already in this newsletter). Next time we will be looking at the following commands:

- **Amal** On
- **Amal** I+1,AML$
- **Channel** I+1 To Bob I

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

---

**CREATE ROTATING SHOOTER SPRITES**

For I=0 To 350 Step 10

- **Increment** SHAPE
- **Increment** 1

Next I

Next I

1) Dir$='Extras:Sprite_600' : Load 'FLIGHT/SPITFIREABK'
2) Get Bob 5,100,100 To 265,150 : CIs 0 : Gr Writing 0 : Ink 6 : Text 4) A$='L: 1 K1=0 J L; L X=RX-50; L Y=RY; M 50,-5,10; L X=-200; L

---

**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,*Amal 0,(R0),1; Beg: Pause; Jump Beg*
AMLS*-Beg=Pause; If RA Jump Shot; Jump Beg; Shot:
Let RA=0

Next I

**CREATE SHOT SPRITE & STARS**

Amal 1,*Amal 0,(R0),1; Beg: Pause; Jump Beg*
AMLS*-Beg=Pause; If RA Jump Shot; Jump Beg; Shot:
Let RA=0

Next I

**THE MAIN GAME LOOP**

For I=1 To 20

- **Amal** On
- **Amal** On

Bob Clear
Bob Draw

continued on page 9
## Important Note About AMOS V1.23

Well AMOS has been updated yet again! This time we have what many programmers are looking for, 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. One is that the skills should be stored as a list string. Eg 'pickpockets/65'.

This would mean that any skill could be added to the original ones, we would not be restricting ourselves to a fixed skill list at the start. Also with AMOS being used for writing text adventure games, this is a very good example of how proceduralism your programs to make them easier and more efficient.

Well must have put in a lot of late nights on this so let’s see some responses in the form of feature requests and actual adventures. Start emailing 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 the blank disks (or disks with their prototypes) in a postacpe and I would return the disk with a compilation of protoypes to be tested. We do not want to make up such a disk at this time so if you send me a blank disk I cannot guarantee it will return soon. It is probably best to wait until next newsletter unless you have something for me to test.

The area of SCF which generated the most interest (aside from me forgetting what it was called). There were more suggestions and ideas proposed, several people decided so lets not try. If we have a base of general skill areas rather than individual skills we might find a more usable format, eg 'Skill 12 might be "Slight Pickpocket"'.

On playtesting, several people have now shown an interest in playtesting text adventure games. This is a huge effort and not only does his article give the basics for writing text adventure games, it is a good example of how proceduralism your programs to make them easier and more efficient.
AMAL EDITOR TUTORIAL

It always amazes me to see just how many programmers write strings and strings of amal, and that's using the AMAL EDITOR. So often you see something like the following in a program:

```
screen open 0,200,200,16,lowres
brd 0,100,1
```

This is just a brief intro into the Amal Editor. Next edition, I'll be going into programs 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 new additions to the libraries, but it is pretty disappointing. End of subject.

Well look at the number of new disks that have arrived. We have heaps of games and source submissions have got some new artists contributing. We have seen the listings for games from DCAT, which is a version of Blackjack. Nothing from the working on the new car racing game as yet, but I'm sure they will be watch out for it in the next listing. Another busy week has gone by, the Amal Editor has been updated and there is of course a new game of Blackjack. Nothing from the working on the new car racing game as yet, but I'm sure they will be looking out for it in the next listing.

NEW ADDITIONS TO THE PD LIBRARIES.

AA105...A&L: Double Shot. Another excellent game from DCAT.
AA106...DRAWS WELL: Dave's new program, very good.
AA107...THE DESCRIBER: Various programs, some music.
AA108...ANALOGUE: Various programs, some music.
AA109...BLADE GAMES: Various programs, some music.
AA110...MUPYR D: Various programs, some music.
AA111...AMOS MUSIC PLAYER: Yet another music player.
AA112...MR. B: Various programs, some music.
AA113...PREDATORS DEMO: Excellent demo.
AA114...E. CAD DEMO V1.2: Working version of a Printed Circuit Board generator.
AA115...NIK WILLIAMS DEMO: Another demo.
AA116...J. P. M. SOUND DISK: Heaps of sound samples.
AA117...MUSIC + PROGRAMS: Asstd programs + some music.
AA118...J. P. M. SOUND DISK: Heaps of sound samples.
AA119...THE DESCRIBER: Various programs, some music.
AA120...AMOS MUSIC PLAYER: Yet another music player!
AA121...NIK WILLIAMS DEMO: Another demo.
AA122...J. P. M. SOUND DISK: Heaps of sound samples.
AA123...DEADLINE: The first demo of the soon to be released TOME Map Generator. Fully playable but.
AA124...BOB MANIACS SYNTAX DEMO: Super Demo with COMPLEX patterned music and graphics.
AA125...BENSOM DEMO II: Better than Bensom's first demo!
AA126...DREAMERS 3: Mag-On-A-Disk. Text not too good, but the graphics/animation are very good.
AA127...COBAN'S ROMANTIC ROMANTIC: Another great demo!
AA128...MACC COMPUTER CLUB DEMO DISK: Demo for a club with animation every 3 seconds.
AA129...SYNTAX MUSIC DISK 1: Musical Demo.
AA130...AMOS GEMODION SYNTAX DISK: Very good.
AA131...FAMILY HISTORY DATABASE: Genealogy Program.

Continued On Page 11
AMOS AUSSIE Disk & 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"

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

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

[Options for adding Tome Extensions or updates]

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