INSIDE THE ULTIMA ONLINE GOLD DEMO - THE COMMAND LIST – PART 1

GOAL

It's our goal to get a deep understanding of how the Ultima Online Gold Demo works. This demo is a representation of the rule set from the Ultima Online Second Age Era.

There is proof that some people have already reversed this demo partially or as a whole, however so far no tools or knowledge has been published. This project is to overcome does shortcomings.

URL's with some proof for this:
http://www.runuo.com/forums/general-discussion/94767-help-m-files.html
http://azaroth.org/2008/12/31/your-topic/ (posting by Faust)

If we understand the demo there is a big chance we can alter the demo and even create our own demo. By default mounting horses is not possible in the demo, but what if we can alter the demo and unlock horses; can we then see how horses behaved during T2A?

This demo is 10 years old and I do not understand no one published his/her work. Maybe that DMCA thing is in the way?

UTILITIES USED

IDA Pro, a very professional utility, definitely worth buying, Standard version is affordable.
BRAIN, every one has one, use it

ABOUT ME

I'm just a guy who loves the Ultima universe and knows a bit assembler. Why not combine the two? ☺ A nice story from my school period; during the dBase IV classes we had to write some programs as practice. However, I was always the first one to be ready and I got bored while waiting for the others to finish. So I wrote an arkanoid-like game in dBase IV (with mouse support ☺). When the teacher found out, he couldn't say much other than "Please don't do this during class time.".

STUMBLING UPON DATA

While browsing through the disassembly of the UoDemo and while stepping through the code with the debugger I stumbled upon an array which looked like offset data. IDA incorrectly identified this data as bytes.

```
.data:00606EA4 dd offset sub_40D1CE
.data:00606EA8 db    3
.data:00606EA9 db    0
.data:00606EAA db    0
.data:00606EAB db    0
.data:00606EAC off_606EAC dd offset aViI              ; DATA XREF: sub_40CF6E+A0↑r
.data:00606EAC                                        ; "vi|i"
.data:00606EB0 dd offset aTk_else_0                   ; "TK_ELSE"
.data:00606EB4 dd offset sub_40D1F7
.data:00606EB8 db    2
.data:00606EB9 db    0
.data:00606EBA db    0
.data:00606EBB db    0
.data:00606EBC db 0C8h ; +
.data:00606EBD db  9Eh ; ×
.data:00606EBE db  60h ; `
.data:00606EBF db    0
.data:00606EC0 db 0CCh ; ¦
.data:00606EC1 db  9Eh ; ×
.data:00606EC2 db  60h ; `
.data:00606EC3 db    0
.data:00606EC4 db  1Ah
.data:00606EC5 db 0D2h ; Ê
.data:00606EC6 db  40h ; @
.data:00606EC7 db    0
.data:00606EC8 db    0
.data:00606EC9 db    0
.data:00606ECA db    0
.data:00606ECB db    0
.data:00606ECC db 0D8h ; Ï
.data:00606ECD db  9Eh ; ×
.data:00606ECE db  60h ; `
.data:00606ECF db    0
.data:00606ED0 db 0DCh ; _
.data:00606ED1 db  9Eh ; ×
.data:00606ED2 db  60h ; `
.data:00606ED3 db    0
.data:00606ED4 db  9Ch ; £
.data:00606ED5 db 0D2h ; Ê
.data:00606ED6 db  40h ; @
.data:00606ED7 db    0
.data:00606ED8 db    3
.data:00606ED9 db    0
```

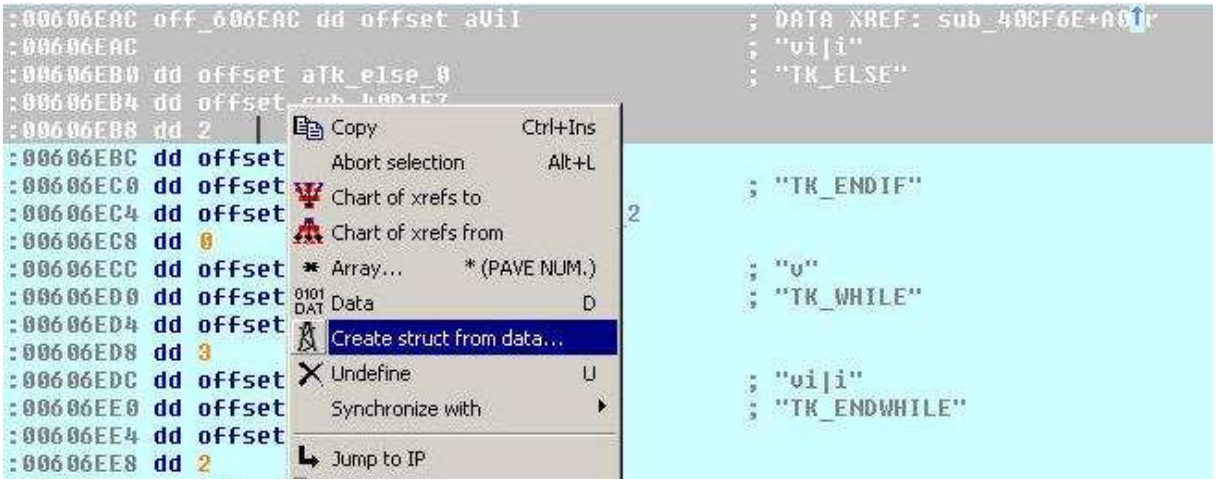
THE STRUCTURE

I started correcting the identification using the data option of IDA, which resulted in the following:

```
:00606EBC dd offset off_609EC8
:00606EC0 dd offset aTk_endif_1                   ; "TK_ENDIF"
:00606EC4 dd offset __initp_misc_winxfltr_2
:00606EC8 dd 0
:00606ECC dd offset aV_22                         ; "v"
:00606ED0 dd offset aTk_while_0                   ; "TK_WHILE"
:00606ED4 dd offset sub_40D29C
:00606ED8 dd 3
:00606EDC dd offset aViI_0                        ; "vi|i"
:00606EE0 dd offset aTk_endwhile_1                ; "TK_ENDWHILE"
:00606EE4 dd offset sub_40D2C5
:00606EE8 dd 2
```

As you can see, lots of interesting stuff is starting to emerge. After my corrections IDA automatically identified pointers to strings and added comments for them. Also, look at the fact that IDA identifies pointers to code labeled sub_XXXXXX. Does this mean that the code for a (scripted) WHILE is available at 0040D29C? Most probably yes. It really looked like I found a command structure array in memory.

So I used IDA to create a structure from this data:



IDA created a structure for me and I renamed the structure members based on what I saw:

```
00000000
00000000 struct_3 struc ; (sizeof=0x10)
00000000 UnknownText dd ?
00000004 Command dd ?
00000008 FunctionAddress dd ?
0000000C UnknownValue dd ?
00000010 struct_3 ends
00000010
```

When you do this at home; if you don't know what a certain member is being used for indicate this in its name. That's why I named 2 members UnknownText and UnknownValue.

You can now apply this structure to the data and IDA will do the rest:

```
00606EAC stru_606EAC struct_3 <offset aVil, offset aTk_else_0, offset sub_40D1F7, 2>
00606EAC                                        ; DATA XREF: sub_40CF6E+A0↑r
00606EAC                                        ; "vi|i"
00606EBC struct_3 <offset off_609EC8, offset aTk_endif_1, \ ; "TK_ENDIF"
00606EBC           offset __initp_misc_winxfltr_2, 0>
00606ECC struct_3 <offset aV_22, offset aTk_while_0, offset sub_40D29C, 3> ; "v"
00606EDC struct_3 <offset aVil_0, offset aTk_endwhile_1, offset sub_40D2C5, 2> ; "vi|i"
00606EEC struct_3 <offset off_609EFC, offset aTk_for_0, offset sub_40D1B2, 3> ; "TK_FOR"
00606EFC struct_3 <offset off_609F08, offset aTk_endfor_0, offset sub_40D1C3, \ ; "TK_ENDFOR"
00606EFC           2>
00606F0C struct_3 <offset off_609F18, offset aTk_continue, offset sub_40D2F2, \
00606F0C           2>
00606F1C db  28h ; (
00606F1D db  9Fh ; ■
00606F1E db  60h ; `
00606F1F db    0
00606F20 db  2Ch ; ,
00606F21 db  9Fh ; ■
00606F22 db  60h ; `
00606F23 db    0
```

Then I discovered I made an error. My brain didn't work well ☺.

```
:00606E70 aOprnot_6 db 'oprnot',0
:00606E77 align 4
:00606E78 aSm_lparen_4 db 'SM_LPAREN',0
:00606E82 align 4
:00606E84 aOprnull_5 db 'oprnull',0
:00606E8C aSm_lbracket_4 db 'SM_LBRACKET',0
:00606E98 aOprlist_10 db 'oprlist',0
:00606EA0 off_606EA0 dd offset aTk_if           ; DATA XREF: sub_40CF6E+22↑r
:00606EA0                                       ; sub_40CF6E+7A↑r ...
:00606EA0                                       ; "TK_IF"
:00606EA4 dd offset sub_40D1CE
:00606EA8 dd 3
:00606EAC dd offset aVil                        ; DATA XREF: sub_40CF6E+A0↑r
:00606EAC                                       ; "vi|i"
:00606EB0 dd offset aTk_else_0                  ; "TK_ELSE"
:00606EB4 dd offset sub_40D1F7
:00606EB8 dd 2
:00606EBC struct_3 <offset off_609EC8, offset aTk_endif_1, \ ; "TK_ENDIF"
:00606EBC           offset __initp_misc_winxfltr_2, 0>
:00606ECC struct_3 <offset aV_22, offset aTk_while_0, offset sub_40D29C, 3> ; "v"
:00606EDC struct_3 <offset aVil_0, offset aTk_endwhile_1, offset sub_40D2C5, 2> ; "vi|i"
:00606EEC struct_3 <offset off_609EFC, offset aTk_for_0, offset sub_40D1B2, 3> ; "TK_FOR"
:00606EFC struct_3 <offset off_609F08, offset aTk_endfor_0, offset sub_40D1C3, \ ; "TK_ENDFOR"
:00606EFC           2>
:00606F0C struct_3 <offset off_609F18, offset aTk_continue, offset sub_40D2F2, \ ; "TK_CONTINUE"
:00606F0C           2>
:00606F1C db  28h ; (
```

ERROR! TK_IF really seems to fit our structure and the rest of the found text. However, we cannot apply the structure to it since the remaining data is only 3 DD's. Something does not compute...

This will still fit in the structure, our structure is 4 DD's

Now, what happened? I started applying the structure in the middle of a bigger something (array?). I wouldn't have encountered this problem if I had started at off_606EA0. It's an array by the way. All the structures follow each other sequentially in memory.

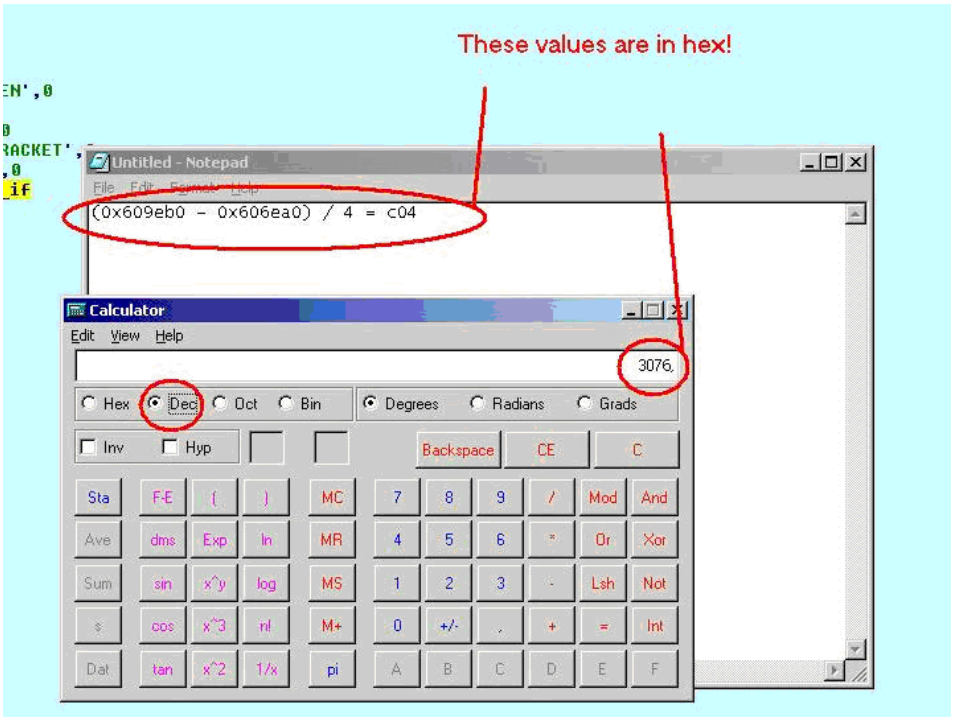
THE ARRAY

Okay, so if offset 00606EA0 is the beginning of the array. Where does it end? What is the size of the array?

To locate the ending I simply scrolled down looking for something that was off:

```
.data:00609E95 db  66h ; f
.data:00609E96 db  41h ; A
.data:00609E97 db    0
.data:00609E98 db  2Ch ; ,
.data:00609E99 db    0
.data:00609E9A db    0
.data:00609E9B db    0
.data:00609E9C db  88h ; ê
.data:00609E9D db 0D9h ; +
.data:00609E9E db  60h ; `
.data:00609E9F db    0
.data:00609EA0 db    0
.data:00609EA1 db    0
.data:00609EA2 db    0
.data:00609EA3 db    0
.data:00609EA4 db    0
.data:00609EA5 db    0
.data:00609EA6 db    0
.data:00609EA7 db    0
.data:00609EA8 db    0
.data:00609EA9 db    0
.data:00609EAA db    0
.data:00609EAB db    0
.data:00609EAC db  8Ch ; î
.data:00609EAD db 0D9h ; +
.data:00609EAE db  60h ; `
.data:00609EAF db    0
.data:00609EB0 aTk_if db 'TK_IF',0                    ; DATA XREF: .data:off_606EA0↑o
.data:00609EB6 align 4
.data:00609EB8 aViI db 'vi|i',0
.data:00609EBD align 10h
.data:00609EC0 aTk_else_0 db 'TK_ELSE',0
.data:00609EC8 dd offset unk_697C76
```

You can now calculate the size of the structure assuming it ends at 00609EB0:

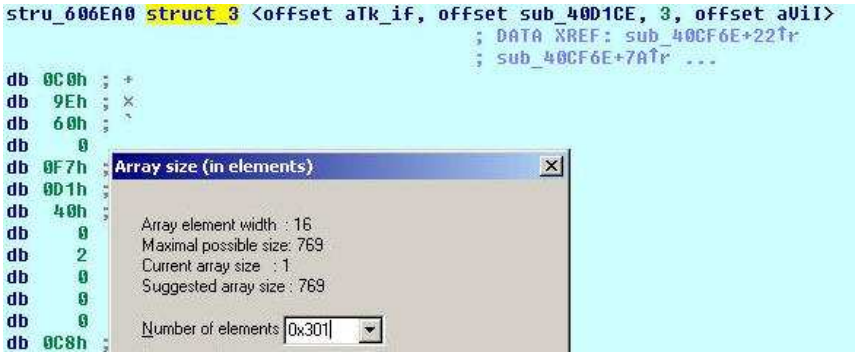


First step, apply the structure at the beginning:

```
:00606EA0 stru_606EA0 struct_3 <offset aTk_if, offset sub_40D1CE, 3, offset aViI>
:00606EA0                                       ; DATA XREF: sub_40CF6E+22↑r
:00606EA0                                       ; sub_40CF6E+7A↑r ...
:00606EA0                                       ; "TK_IF"
:00606EB0 db 0C0h ; +
:00606EB1 db  9Eh ; ×
:00606EB2 db  60h ; `
:00606EB3 db    0
:00606EB4 db 0F7h ; ÷
:00606EB5 db 0D1h ; Ð
:00606EB6 db  40h ; @
:00606EB7 db    0
:00606EB8 db    2
:00606EB9 db    0
:00606EBA db    0
:00606EBB db    0
```

Next step, our calculated array consisted of 3076 DWORD's, each DWORD being 4 bytes. Our structure consists of 4 DWORD's. Therefore to get the number of structures in the array you have to divide 3076 by 4, the result is 769 or 0x301 (hexadecimal).

With this knowledge we have IDA create an array:



Result:

```
:00606EA0 stru_606EA0 struct_3 <offset aTk_if, offset sub_40D1CE, 3, offset aViI>
:00606EA0                                       ; DATA XREF: sub_40CF6E+22↑r
:00606EA0                                       ; sub_40CF6E+7A↑r ...
:00606EA0 struct_3 <offset aTk_else_0, offset sub_40D1F7, 2, offset off_609EC8> ; "TK_IF"
:00606EA0 struct_3 <offset aTk_endif_1, offset __initp_misc_winxfltr_2, 0, \
:00606EA0            offset aV_15>
:00606EA0 struct_3 <offset aTk_while_0, offset sub_40D29C, 3, offset aViI_0>
:00606EA0 struct_3 <offset aTk_endwhile_1, offset sub_40D2C5, 2, \
:00606EA0            offset off_609EFC>
:00606EA0 struct_3 <offset aTk_for_0, offset sub_40D1B2, 3, offset off_609F08>
:00606EA0 struct_3 <offset aTk_endfor_0, offset sub_40D1C3, 2, \
:00606EA0            offset off_609F18>
:00606EA0 struct_3 <offset aTk_continue, offset sub_40D2F2, 2, \
:00606EA0            offset off_609F28>
:00606EA0 struct_3 <offset aTk_break, offset sub_40D31F, 2, offset off_609F38>
:00606EA0 struct_3 <offset aTk_goto_0, offset sub_40D342, 2, offset aVi_3>
:00606EA0 struct_3 <offset aTk_switch, offset sub_40D22E, 26h, offset aViJ>
:00606EA0 struct_3 <offset aTk_endswitch_0, offset __initp_misc_winxfltr_3, 0, \
:00606EA0            offset aV_16>
:00606EA0 struct_3 <offset aTk_case_1, offset __initp_misc_winxfltr_4, 0, \
:00606EA0            offset aV_17>
:00606EA0 struct_3 <offset aTk_default, offset __initp_misc_winxfltr_5, 0, \
:00606EA0            offset aV_18>
:00606EA0 struct_3 <offset aTk_return_0, offset sub_40D509, 0, offset aV_22>
:00606EA0 struct_3 <offset aTk_return_1, offset sub_40D365, 2, offset aVi_4>
```

CLEANING UP – PART 1

Now, what annoys me here is that the array view isn't that easy to read. Sure, if you look closely you can see text hidden in it, but that isn't easy to work with.

Take a look at a DWORD view (by removing the array and marking the data as DD):

```
.data:00606EB4 dd offset sub_40D1F7
.data:00606EB8 dd 2
.data:00606EBC dd offset off_609EC8
.data:00606EC0 dd offset aTk_endif                     ; "TK_ENDIF"
.data:00606EC4 dd offset __initp_misc_winxfltr_2
.data:00606EC8 dd 0
.data:00606ECC dd offset aV                            ; "v"
.data:00606ED0 dd offset aTk_while                     ; "TK_WHILE"
.data:00606ED4 dd offset sub_40D29C
.data:00606ED8 dd 3
.data:00606EDC dd offset aViI_0                        ; "vi|i"
.data:00606EE0 dd offset aTk_endwhile_0                ; "TK_ENDWHILE"
.data:00606EE4 dd offset sub_40D2C5
.data:00606EE8 dd 2
.data:00606EEC dd offset off_609EFC
.data:00606EF0 dd offset aTk_for                       ; "TK_FOR"
.data:00606EF4 dd offset sub_40D1B2
.data:00606EF8 dd 3
.data:00606EFC dd offset off_609F08
.data:00606F00 dd offset aTk_endfor                    ; "TK_ENDFOR"
.data:00606F04 dd offset sub_40D1C3
.data:00606F08 dd 2
.data:00606F0C dd offset off_609F18
.data:00606F10 dd offset aTk_continue                  ; "TK_CONTINUE"
.data:00606F14 dd offset sub_40D2F2
.data:00606F18 dd 2
.data:00606F1C dd offset off_609F28
.data:00606F20 dd offset aTk_break                     ; "TK_BREAK"
.data:00606F24 dd offset sub_40D31F
.data:00606F28 dd 2
.data:00606F2C dd offset off_609F38
.data:00606F30 dd offset aTk_goto_0                    ; "TK_GOTO"
.data:00606F34 dd offset sub_40D342
.data:00606F38 dd 2
.data:00606F3C dd offset aVi_0                         ; "vi"
.data:00606F40 dd offset aTk_switch                    ; "TK_SWITCH"
.data:00606F44 dd offset sub_40D22E
.data:00606F48 dd 26h
.data:00606F4C dd offset aViJ                          ; "vi|j"
.data:00606F50 dd offset aTk_endswitch                 ; "TK_ENDSWITCH"
.data:00606F54 dd offset __initp_misc_winxfltr_3
.data:00606F58 dd 0
.data:00606F5C dd offset aV_0                          ; "v"
.data:00606F60 dd offset aTk_case                      ; "TK_CASE"
.data:00606F64 dd offset __initp_misc_winxfltr_4
.data:00606F68 dd 0
.data:00606F6C dd offset aV_1                          ; "v"
.data:00606F70 dd offset aTk_default                   ; "TK_DEFAULT"
```

I don't know about you, but I find the above view much more interesting. Even though it doesn't show that we are dealing with a structure + array. This situation needs to be solved somehow.

One way of solving this is to modify all the string variable names into capitals, the red square I have already modified, the orange part I didn't do yet, see yourself:

```
.data:00606E70 aOprnot_6 db 'oprnot',0
.data:00606E77 align 4
.data:00606E78 aSm_lparen_4 db 'SM_LPAREN',0
.data:00606E82 align 4
.data:00606E84 aOprnull_5 db 'oprnull',0
.data:00606E8C aSm_lbracket_4 db 'SM_LBRACKET',0
.data:00606E98 aOprlist_10 db 'oprlist',0
.data:00606EA0 stru_606EA0 struct_3 <offset aTK_IF, offset sub_40D1CE, 3, offset aViI>
.data:00606EA0                                         ; DATA XREF: sub_40CF6E+22↑r
.data:00606EA0                                         ; sub_40CF6E+7A↑r ...
.data:00606EA0 struct_3 <offset aTK_ELSE, offset sub_40D1F7, 2, offset off_609EC8> ; "TK_IF"
.data:00606EA0 struct_3 <offset aTK_ENDIF, offset __initp_misc_winxfltr_2, 0, \
.data:00606EA0           offset aV>
.data:00606EA0 struct_3 <offset aTK_WHILE, offset sub_40D29C, 3, offset aViI_0>
.data:00606EA0 struct_3 <offset aTK_ENDWHILE, offset sub_40D2C5, 2, \
.data:00606EA0           offset off_609EFC>
.data:00606EA0 struct_3 <offset aTK_FOR, offset sub_40D1B2, 3, offset off_609F08>
.data:00606EA0 struct_3 <offset aTK_ENDFOR, offset sub_40D1C3, 2, offset off_609F18>
.data:00606EA0 struct_3 <offset aTK_CONTINUE, offset sub_40D2F2, 2, \
.data:00606EA0           offset off_609F28>
.data:00606EA0 struct_3 <offset aTK_BREAK, offset sub_40D31F, 2, offset off_609F38>
.data:00606EA0 struct_3 <offset aTK_GOTO, offset sub_40D342, 2, offset aVi_0>
.data:00606EA0 struct_3 <offset aTK_SWITCH, offset sub_40D22E, 26h, offset aViJ>
.data:00606EA0 struct_3 <offset aTK_ENDSWITCH, offset __initp_misc_winxfltr_3, 0, \
.data:00606EA0           offset aV_0>
.data:00606EA0 struct_3 <offset aTk_case, offset __initp_misc_winxfltr_4, 0, \
.data:00606EA0           offset aV_1>
.data:00606EA0 struct_3 <offset aTk_default, offset __initp_misc_winxfltr_5, 0, \
.data:00606EA0           offset aV_2>
.data:00606EA0 struct_3 <offset aTk_return, offset sub_40D509, 0, offset aV_3>
.data:00606EA0 struct_3 <offset aTk_return_0, offset sub_40D365, 2, offset aVi_1>
.data:00606EA0 struct_3 <offset aTk_return_1, offset sub_40D40D, 56h, offset aVc>
.data:00606EA0 struct_3 <offset aTk_return_2, offset sub_40D3B9, 51h, offset aVo>
.data:00606EA0 struct_3 <offset aTk_return_3, offset sub_40D459, 8, offset aVs>
.data:00606EA0 struct_3 <offset aOprnull, offset sub_40E392, 14h, offset aIi>
.data:00606EA0 struct_3 <offset aOprplus_5, offset sub_40E39A, 15h, \
.data:00606EA0           offset off_609FF0>
.data:00606EA0 struct_3 <offset aOprminus_5, offset sub_40E3A5, 15h, \
.data:00606EA0           offset off_60A000>
.data:00606EA0 struct_3 <offset aOprmult_5, offset sub_40E3B0, 15h, \
.data:00606EA0           offset off_60A00C>
.data:00606EA0 struct_3 <offset aOprdiv_5, offset sub_40E3BC, 15h, offset off_60A018>
.data:00606EA0 struct_3 <offset aOprand_5, offset sub_40E3E0, 15h, offset off_60A024>
.data:00606EA0 struct_3 <offset aOpror_5, offset sub_40E407, 15h, offset off_60A030>
.data:00606EA0 struct_3 <offset aOprxor_5, offset sub_40E42E, 15h, offset off_60A03C>
.data:00606EA0 struct_3 <offset aOprequiv_5, offset sub_40E439, 15h, \
.data:00606EA0           offset off_60A04C>
.data:00606EA0 struct_3 <offset aOprnequiv_5, offset sub_40E44B, 15h, \
```

And there is yet another problem we have to solve, IDA Pro isn't perfect. It's interactive and its interactivity comes in handy now:

```
struct_3 <offset aTK_ELSE, offset sub_40D1F7, 2, offset aVI> ; "TK_IF"
struct_3 <offset aTK_ENDIF, offset __initp_misc_winxfltr_2, 0, \
         offset aV>
struct_3 <offset aTK_WHILE, offset sub_40D29C, 3, offset aViI_0>
struct_3 <offset aTK_ENDWHILE, offset sub_40D2C5, 2, \
         offset off_609EFC>
struct_3 <offset aTK_FOR, offset sub_40D1B2, 3, offset off_609F08>
struct_3 <offset aTK_ENDFOR, offset sub_40D1C3, 2, offset off_609F18>
struct_3 <offset aTK_CONTINUE, offset sub_40D2F2, 2, \
         offset off_609F28>
struct_3 <offset aTK_BREAK, offset sub_40D31F, 2, offset off_609F38>
struct_3 <offset aTK_GOTO, offset sub_40D342, 2, offset aVi_0>
struct_3 <offset aTK_SWITCH, offset sub_40D22E, 26h, offset aViJ>
struct_3 <offset aTK_ENDSWITCH, offset __initp_misc_winxfltr_3, 0, \
         offset aV_0>
struct_3 <offset aTk_case, offset __initp_misc_winxfltr_4, 0, \
         offset aV_1>
struct_3 <offset aTk_default, offset __initp_misc_winxfltr_5, 0, \
         offset aV_2>
```

The red circle is what we will look into next. The orange circles are strings, so I can only assume that the red circle should also be a readable string! Yes! My brain at work again.

Follow the pointer and look at the red circles, the first red circles I have corrected by telling IDA Pro that an ASCII string is at that location, the other red circles I still have to correct and the orange circles are there for comparison with the orange circle above:

```
.data:00609EEF db    0
.data:00609EF0 aTK_ENDWHILE db 'TK_ENDWHILE',0          ; DATA XREF: .data:stru_606EA0↑o
.data:00609EFC off_609EFC dd offset unk_697C76          ; DATA XREF: .data:stru_606EA0↑o
.data:00609F00 aTK_FOR db 'TK_FOR',0                    ; DATA XREF: .data:stru_606EA0↑o
.data:00609F07 db    0
.data:00609F08 aVii db 'vii',0                          ; DATA XREF: .data:stru_606EA0↑o
.data:00609F0C aTK_ENDFOR db 'TK_ENDFOR',0              ; DATA XREF: .data:stru_606EA0↑o
.data:00609F16 db    0
.data:00609F17 db    0
.data:00609F18 off_609F18 dd offset unk_697C76          ; DATA XREF: .data:stru_606EA0↑o
.data:00609F1C aTK_CONTINUE db 'TK_CONTINUE',0          ; DATA XREF: .data:stru_606EA0↑o
.data:00609F28 off_609F28 dd offset unk_697C76          ; DATA XREF: .data:stru_606EA0↑o
.data:00609F2C aTK_BREAK db 'TK_BREAK',0                ; DATA XREF: .data:stru_606EA0↑o
.data:00609F35 db    0
.data:00609F36 db    0
.data:00609F37 db    0
.data:00609F38 off_609F38 dd offset unk_697C76          ; DATA XREF: .data:stru_606EA0↑o
.data:00609F3C aTK_GOTO db 'TK_GOTO',0                  ; DATA XREF: .data:stru_606EA0↑o
.data:00609F44 aVi_0 db 'vi',0                          ; DATA XREF: .data:stru_606EA0↑o
.data:00609F47 db    0
.data:00609F48 aTK_SWITCH db 'TK_SWITCH',0              ; DATA XREF: .data:stru_606EA0↑o
.data:00609F52 db    0
.data:00609F53 db    0
.data:00609F54 aViJ db 'vi|j',0                         ; DATA XREF: .data:stru_606EA0↑o
.data:00609F59 db    0
.data:00609F5A db    0
.data:00609F5B db    0
.data:00609F5C aTK_ENDSWITCH db 'TK_ENDSWITCH',0        ; DATA XREF: .data:stru_606EA0↑o
.data:00609F69 db    0
.data:00609F6A db    0
.data:00609F6B db    0
.data:00609F6C aV_0:                                    ; DATA XREF: .data:stru_606EA0↑o
.data:00609F6C unicode 0, <v>,0
.data:00609F70 aTk_case db 'TK_CASE',0                  ; DATA XREF: .data:stru_606EA0↑o
.data:00609F78 aV_1:                                    ; DATA XREF: .data:stru_606EA0↑o
.data:00609F78 unicode 0, <v>,0
.data:00609F7C aTk_default db 'TK_DEFAULT',0            ; DATA XREF: .data:stru_606EA0↑o
.data:00609F87 db    0
.data:00609F88 aV_2:                                    ; DATA XREF: .data:stru_606EA0↑o
.data:00609F88 unicode 0, <v>,0
.data:00609F8C aTk_return db 'TK_RETURN',0              ; DATA XREF: .data:stru_606EA0↑o
.data:00609F96 db    0
```

To give you a better idea how it works:

```
aGetfreehandslot db 'getFreeHandSlot',0
aIo_61 db 'io',0
db    0
aGetyear db 'getYear',0
aI:
unicode 0, <i>,0
aGetmonth db 'getMonth',0
db    0
db    0
db    0
aI_0:
unicode 0, <i>,0
aGetweek db 'getWeek',0
aI_1:
unicode 0, <i>,0
aGetday db 'getDay',0
db    0
aI_4:
unicode 0, <i>,0
aGethour db 'getHour',0
aI_5:
unicode 0, <i>,0
aGetminute db 'getMinute',0
db    0
db    0
aI_6:
unicode 0, <i>,0
aGetseconds db 'getSeconds',0
db    0
aI_7:
unicode 0, <i>,0
aIsweapon db 'isWeapon',0
db    0
db    0
db    0
aIo_62 db 'io',0
db    0
```

BEFORE

```
aGetfreehandslot db 'getFreeHandSlot',0
aIo_61 db 'io',0
db    0
aGetyear db 'getYear',0
aI db 'i',0
db    0
db    0
aGetmonth db 'getMonth',0
db    0
db    0
db    0
aI_17 db 'i',0
db    0
db    0
aGetweek db 'getWeek',0
aI_0 db 'i',0
db    0
db    0
aGetday db 'getDay',0
db    0
aI_1 db 'i',0
db    0
db    0
aGethour db 'getHour',0
aI_4 db 'i',0
db    0
db    0
aGetminute db 'getMinute',0
db    0
db    0
aI_5 db 'i',0
db    0
db    0
aGetseconds db 'getSeconds',0
db    0
aI_6 db 'i',0
db    0
```

AFTER

The GLOBAL_CommandList is born

Now, one thing is certain now, the array we are working on contains a list of supported operators and built-in functions, therefore I named that variable GLOBAL_CommandList. When stepping through the code I can now see access to the structure and this will help in analyzing the behavior of the Ultima Online Demo.

```
:00606EA0 GLOBAL_CommandList struct_3 <offset aTK_IF, offset sub_40D1CE, 3, offset aViI>
:00606EA0                                          ; DATA XREF: sub_40CF6E+22↑r
:00606EA0                                          ; sub_40CF6E+7A↑r ...
:00606EA0 struct_3 <offset aTK_ELSE, offset sub_40D1F7, 2, offset aVI> ; "TK_IF"
:00606EA0 struct_3 <offset aTK_ENDIF, offset __initp_misc_winxfltr_2, 0, \
:00606EA0           offset aV>
:00606EA0 struct_3 <offset aTK_WHILE, offset sub_40D29C, 3, offset aViI_0>
:00606EA0 struct_3 <offset aTK_ENDWHILE, offset sub_40D2C5, 2, \
:00606EA0           offset off_609EFC>
:00606EA0 struct_3 <offset aTK_FOR, offset sub_40D1B2, 3, offset aVii>
:00606EA0 struct_3 <offset aTK_ENDFOR, offset sub_40D1C3, 2, offset aVI_0>
:00606EA0 struct_3 <offset aTK_CONTINUE, offset sub_40D2F2, 2, offset aVI_1>
:00606EA0 struct_3 <offset aTK_BREAK, offset sub_40D31F, 2, offset aVI_2>
:00606EA0 struct_3 <offset aTK_GOTO, offset sub_40D342, 2, offset aVi_0>
:00606EA0 struct_3 <offset aTK_SWITCH, offset sub_40D22E, 26h, offset aViJ>
:00606EA0 struct_3 <offset aTK_ENDSWITCH, offset __initp_misc_winxfltr_3, 0, \
:00606EA0           offset aV_8>
```

CLEANING UP – PART 2

No words, just screenshots:

```
.data:00606E8C aSm_lbracket_4 db 'SM_LBRACKET',0
.data:00606E98 aOprlist_10 db 'oprlist',0
.data:00606EA0 GLOBAL_CommandList struct_3 <offset a__TK_IF, offset sub_40D1CE, 3, offset a__vii_i__0>
.data:00606EA0                                         ; DATA XREF: sub_40CF6E+22Tr
.data:00606EA0                                         ; sub_40CF6E+7A↑r ...
.data:00606EA0 struct_3 <offset a__TK_ELSE, offset sub_40D1F7, 2, offset a__v_i__0> ; "TK_IF"
.data:00606EA0 struct_3 <offset a__TK_ENDIF, offset sub_40D21A, 0, offset a__v__0>
.data:00606EA0 struct_3 <offset a__TK_WHILE, offset sub_40D29C, 3, \
.data:00606EA0           offset a__vii_i__1>
.data:00606EA0 struct_3 <offset a__TK_ENDWHILE, offset sub_40D2C5, 2, \
.data:00606EA0           offset a__v_i__1>
.data:00606EA0 struct_3 <offset a__TK_FOR, offset sub_40D1B2, 3, offset a_vii__0>
.data:00606EA0 struct_3 <offset a__TK_ENDFOR, offset sub_40D1C3, 2, offset a__v_i__2>
.data:00606EA0 struct_3 <offset a__TK_CONTINUE, offset sub_40D2F2, 2, \
.data:00606EA0           offset a__v_i__3>
.data:00606EA0 struct_3 <offset a__TK_BREAK, offset sub_40D31F, 2, offset a__v_i__4>
.data:00606EA0 struct_3 <offset a__TK_GOTO, offset unk_40D342, 2, offset a__vi__0>
.data:00606EA0 struct_3 <offset a__TK_SWITCH, offset sub_40D22E, 26h, \
.data:00606EA0           offset a__vi_j__0>
.data:00606EA0 struct_3 <offset a__TK_ENDSWITCH, offset sub_40D21F, 0, \
.data:00606EA0           offset a__v__1>
.data:00606EA0 struct_3 <offset a__TK_CASE, offset sub_40D224, 0, offset a__v__2>
.data:00606EA0 struct_3 <offset a__TK_DEFAULT, offset unk_40D229, 0, offset a__v__3>
.data:00606EA0 struct_3 <offset a__TK_RETURN__0, offset unk_40D509, 0, \
.data:00606EA0           offset a__v__4>
.data:00606EA0 struct_3 <offset a__TK_RETURN__1, offset unk_40D365, 2, \
.data:00606EA0           offset a__vi__1>
.data:00606EA0 struct_3 <offset a__TK_RETURN__2, offset unk_40D40D, 56h, \
.data:00606EA0           offset a__vc__0>
.data:00606EA0 struct_3 <offset a__TK_RETURN__3, offset unk_40D3B9, 51h, \
.data:00606EA0           offset a__vo__0>
.data:00606EA0 struct_3 <offset a__TK_RETURN__4, offset unk_40D459, 8, \
.data:00606EA0           offset a__vs__0>
.data:00606EA0 struct_3 <offset a__oprnull, offset unk_40E392, 14h, offset a__ii__0>
.data:00606EA0 struct_3 <offset a__oprplus, offset unk_40E39A, 15h, offset a__iii__0>
.data:00606EA0 struct_3 <offset a__oprminus, offset unk_40E3A5, 15h, \
.data:00606EA0           offset a__iii__1>
.data:00606EA0 struct_3 <offset a__oprmult, offset unk_40E3B0, 15h, offset a__iii__2>
.data:00606EA0 struct_3 <offset a__oprdiv, offset unk_40E3BC, 15h, offset a__iii__3>
.data:00606EA0 struct_3 <offset a__oprand, offset unk_40E3E0, 15h, offset a__iii__4>
.data:00606EA0 struct_3 <offset a__opror, offset unk_40E407, 15h, offset a__iii__5>
.data:00606EA0 struct_3 <offset a__oprxor, offset unk_40E42E, 15h, offset a__iii__6>
.data:00606EA0 struct_3 <offset a__oprequiv, offset unk_40E439, 15h, \
.data:00606EA0           offset a__iii__7>
.data:00606EA0 struct_3 <offset a__oprnequiv, offset unk_40E44B, 15h, \
.data:00606EA0           offset a__iii__8>
```

```
.data:00606EA0 struct_3 <offset a__getMaxHP, offset unk_413ED6, 2Ch, \
.data:00606EA0           offset a__io__48>
.data:00606EA0 struct_3 <offset a__getCurFatigue, offset unk_413EF1, 2Ch, \
.data:00606EA0           offset a__io__49>
.data:00606EA0 struct_3 <offset a__getMaxFatigue, offset unk_413F0C, 2Ch, \
.data:00606EA0           offset a__io__50>
.data:00606EA0 struct_3 <offset a__getCurMana, offset unk_413F27, 2Ch, \
.data:00606EA0           offset a__io__51>
.data:00606EA0 struct_3 <offset a__getMaxMana, offset unk_413F42, 2Ch, \
.data:00606EA0           offset a__io__52>
.data:00606EA0 struct_3 <offset a__getCanCarry, offset unk_413F5D, 2Ch, \
.data:00606EA0           offset a__io__53>
.data:00606EA0 struct_3 <offset a__getSkillTotal, offset unk_413F78, 2Ch, \
.data:00606EA0           offset a__io__54>
.data:00606EA0 struct_3 <offset a__getCappedSkillTotal, offset unk_413F78, 2Ch, \
.data:00606EA0           offset a__io__55>
.data:00606EA0 struct_3 <offset a__getNaturalAC, offset unk_414347, 2Ch, \
.data:00606EA0           offset a__io__56>
.data:00606EA0 struct_3 <offset a__getHPLevel, offset unk_413FAE, 2Ch, \
.data:00606EA0           offset a__io__57>
.data:00606EA0 struct_3 <offset a__getFatigueLevel, offset unk_413FF2, 2Ch, \
.data:00606EA0           offset a__io__58>
.data:00606EA0 struct_3 <offset a__getManaLevel, offset unk_414036, 2Ch, \
.data:00606EA0           offset a__io__59>
.data:00606EA0 struct_3 <offset a__setCurHP, offset unk_41407A, 32h, \
.data:00606EA0           offset a__voi__19>
.data:00606EA0 struct_3 <offset a__setMaxHP, offset unk_4140B2, 32h, \
.data:00606EA0           offset a__voi__20>
.data:00606EA0 struct_3 <offset a__handleHealthGain, offset unk_4140F6, 51h, \
.data:00606EA0           offset a__vo__7>
.data:00606EA0 struct_3 <offset a__setCurFatigue, offset unk_414128, 32h, \
.data:00606EA0           offset a__voi__21>
.data:00606EA0 struct_3 <offset a__setMaxFatigue, offset unk_41416B, 32h, \
.data:00606EA0           offset a__voi__22>
.data:00606EA0 struct_3 <offset a__setCurMana, offset unk_4141A1, 32h, \
.data:00606EA0           offset a__voi__23>
.data:00606EA0 struct_3 <offset a__setMaxMana, offset unk_4141D7, 32h, \
.data:00606EA0           offset a__voi__24>
.data:00606EA0 struct_3 <offset a__setNaturalAC, offset unk_4142EF, 32h, \
.data:00606EA0           offset a__voi__25>
.data:00606EA0 struct_3 <offset a__addHP, offset unk_41420D, 32h, offset a__voi__26>
.data:00606EA0 struct_3 <offset a__addMana, offset unk_414263, 32h, \
.data:00606EA0           offset a__voi__27>
.data:00606EA0 struct_3 <offset a__addFatigue, offset unk_4142A9, 32h, \
.data:00606EA0           offset a__voi__28>
.data:00606EA0 struct_3 <offset a__doDamageWithWeapon, offset unk_41459F, 0A7h, \
.data:00606EA0           offset a__voooi__0>
.data:00606EA0 struct_3 <offset a__doDamage, offset unk_4145C0, 84h, \
```

FINAL WORD

There is no final word yet since we are just beginning to understand the mysteries of the GLOBAL_CommandList. The next step is most definitely naming all the functions that are being called, which will bring us yet one step closer to fully understanding the inner workings of the Ultima Online demo scripting engine.

While typing this I already have an idea what the strings like “ii”, “voi” and so on mean. This will be covered in Part 2.

In Part 2, I will also add screenshots of some interesting functions inside this command list.

Enjoy waiting.