INSIDE THE ULTIMA ONLINE CLIENT
- ALPHA CLIENT LEFTOVERS, THE CURSORS

GOAL

I'm going to describe and try to understand some unused code found in the Ultima Online 2D clients (and even the Ultima Online Demo).
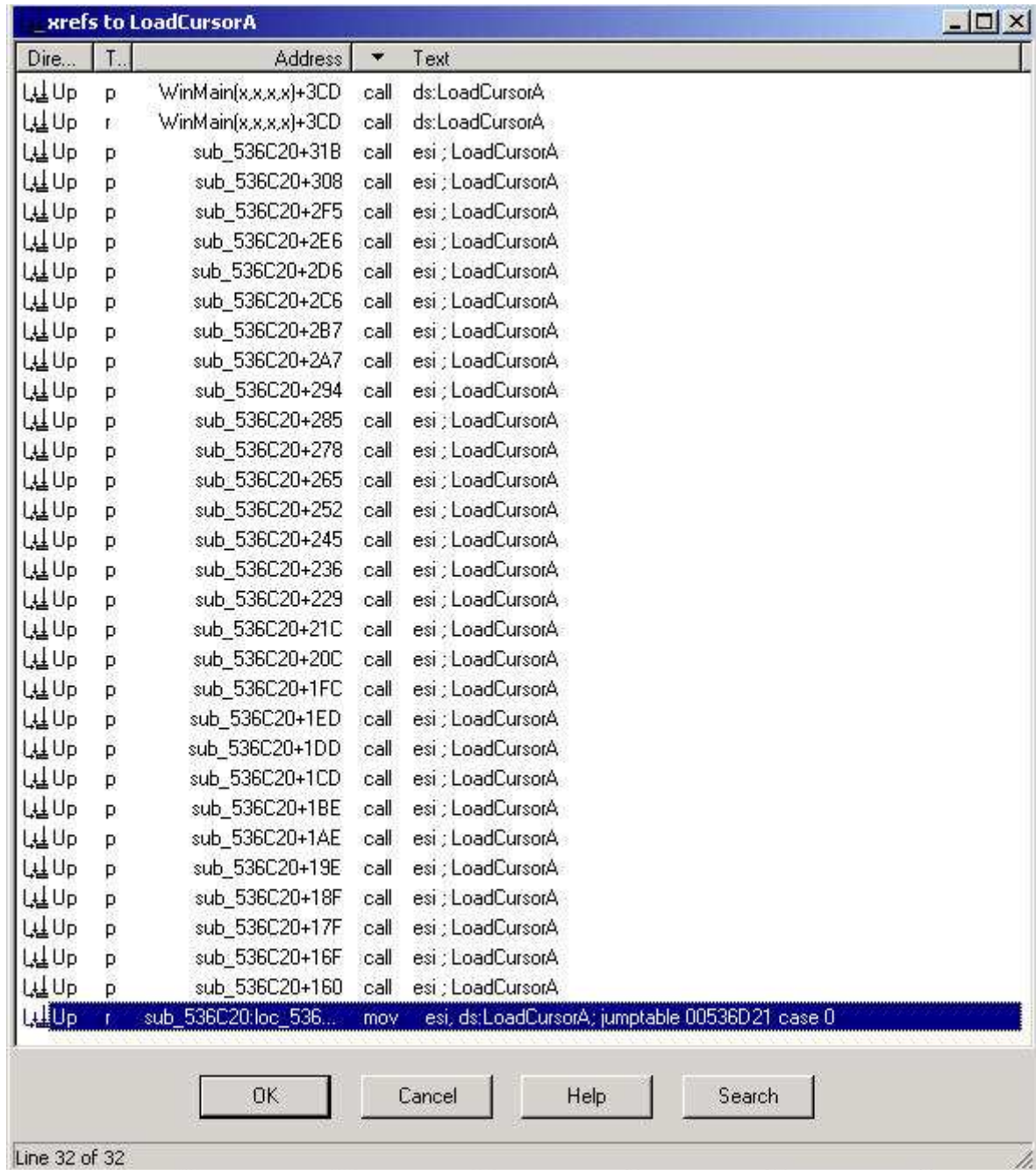
UTILITIES USED

IDA Pro, a very professional utility, definitely worth buying, Standard version is affordable
Resource Hacker, a free utility to fool around with a program's resources

INSIDE THE CLIENT

NOTE: the client analyzed here is version 5.0.8.3

We start the journey by jumping to LoadCursor and opening the cross-references window:

We see that LoadCursor is called a few times but by carefully looking we see that the LoadCursor usage can be divided into two blocks, one is WinMain and the other one is sub_536C20.

Let's look at the first one in WinMain:

```
00535CE5 loc_535CE5:                             ; CODE XREF: WinMain(x,x,x,x)+3AF↑j
00535CE5                 xor     ebp, ebp
00535CE7                 push    7F8Ah            ; lpCursorName
00535CEC                 push    ebp              ; hInstance
00535CED                 call    ds:LoadCursorA
00535CF3                 mov     edi, ds:SetCursor
00535CF9                 push    eax              ; hCursor
00535CFA                 call    edi ; SetCursor
```

Cursor 7F8Ah equals cursor 32650 which in turn equals IDC_APPSTARTING, a default
Windows cursor and thus nothing to be interested in.

Let's look at the second one:

```
00536D66 loc_536D66:                             ; CODE XREF: sub_536C20+101↑j
00536D66                                         ; DATA XREF: .text:off_5379C0↓o
00536D66                 mov     esi, ds:LoadCursorA ; jumptable 00536D21 case 0
00536D6C                 mov     ecx, 40h
00536D71                 xor     eax, eax
00536D73                 mov     edi, offset dword_89AA78
00536D78                 push    7F00h            ; lpCursorName
00536D7D                 push    ebx              ; hInstance
00536D7E                 rep stosd
00536D80                 call    esi ; LoadCursorA
00536D82                 mov     dword_89AA78, eax
00536D87                 mov     eax, hInstance
00536D8C                 push    67h              ; lpCursorName
00536D8E                 push    eax              ; hInstance
00536D8F                 call    esi ; LoadCursorA
00536D91                 mov     ecx, hInstance
00536D97                 push    6Bh              ; lpCursorName
00536D99                 push    ecx              ; hInstance
00536D9A                 mov     dword_89AA7C, eax
00536D9F                 call    esi ; LoadCursorA
```

LoadCursor is being called in series and each returned handle is stored in sequence (an
array). The array itself is initialized with zeroes at the beginning and is 64 (32-bit) handles
long (0x40 dwords). If you count the number of LoadCursor's you will see that only 29
cursors are being loaded.

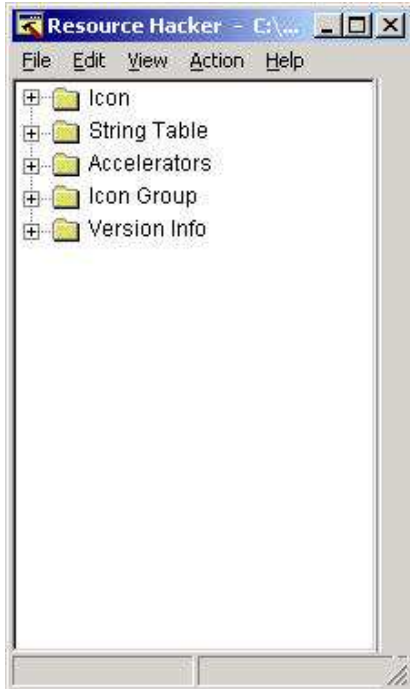We clean up the code a bit, sub_536C20 is the WindowProc:
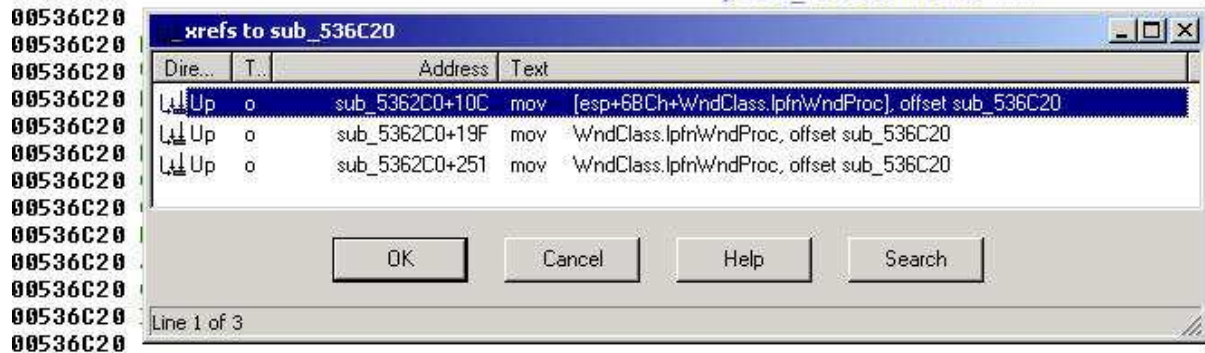
And we convert dword_89AA78 into an array (which I named GLOBAL_CursorArray):
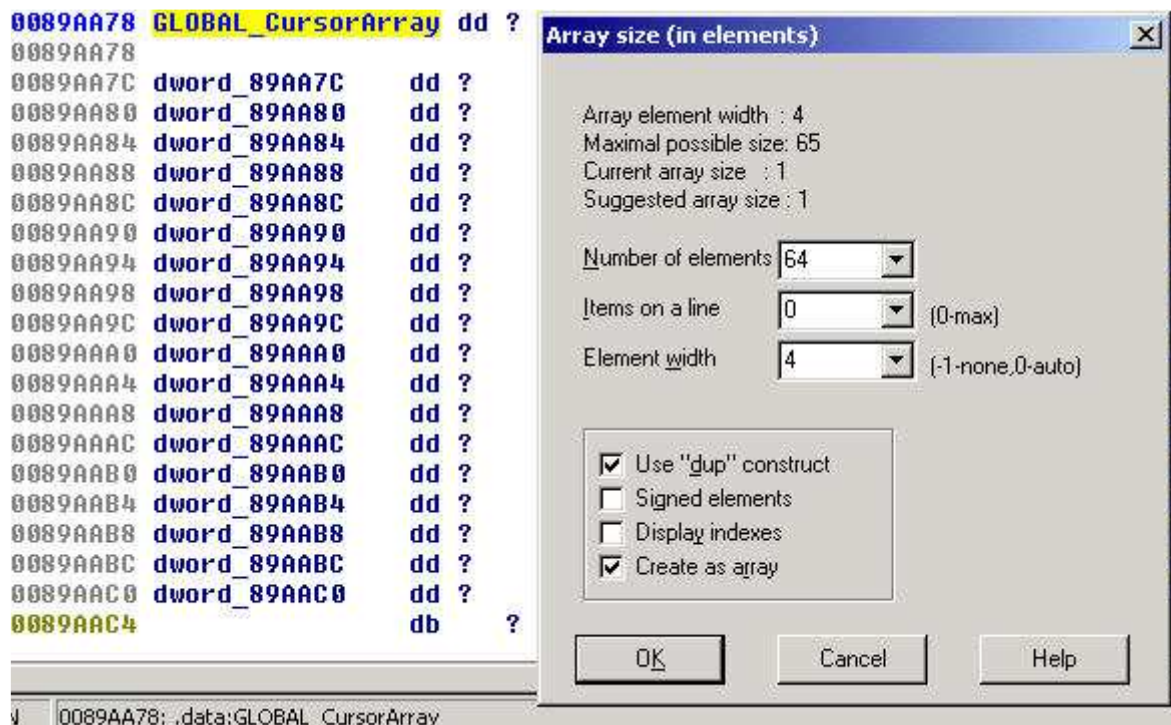
What kinds of cursors are being loaded? Most cursors are application-defined cursors. A few times IDC_ARROW (7F00h) is loaded too.

Let's use a resource editor to peek at the application-defined cursors in the UO client executable:

As you can see (or not see), there are no cursors defined! So all handles returned by LoadCursor will be NULL (except for IDC_ARROW). Either this is some anti-hacking trick or this is bad programming. Considering the unprotected nature of the client I'm going for the second option.

Cross-referencing GLOBAL_CursorArray tells us that the cursors are in fact unused (except for WindowProc), so besides trying to load cursors nothing else is going on with them:

QUESTIONS

Why is the game loading cursors that are not even defined?
Why is the game loading cursors it won't even use?

INSIDE THE UODEMO

The Ultima Online Demo Client is loading cursors too as shown on this screenshot:

```
004FD8F6 mov      [ebp+VAR_Counter], 0
004FD8FD jmp      short LOCAL_DoCursorInitLoop
004FD8FF ; --------------------------------------------
004FD8FF
004FD8FF LOCAL_NextCursorInitLoop:
004FD8FF mov      ecx, [ebp+VAR_Counter]
004FD902 add      ecx, 1
004FD905 mov      [ebp+VAR_Counter], ecx
004FD908
004FD908 LOCAL_DoCursorInitLoop:
004FD908 cmp      [ebp+VAR_Counter], 40h ; '@'
004FD90C jge      short LOCAL_EndCursorInitLoop
004FD90E mov      edx, [ebp+VAR_Counter]
004FD911 mov      GLOBAL_CursorArray[edx*4], 0
004FD91C jmp      short LOCAL_NextCursorInitLoop
004FD91E ; --------------------------------------------
004FD91E
004FD91E LOCAL_EndCursorInitLoop:
004FD91E push     7F00h
004FD923 push     0
004FD925 call     ds:LoadCursorA
004FD92B mov      GLOBAL_CursorArray, eax
004FD930 push     67h ; 'g'
004FD932 mov      eax, GLOBAL_hInstance
004FD937 push     eax
004FD938 call     ds:LoadCursorA
004FD93E mov      GLOBAL_CursorArray+4, eax
004FD943 push     6Bh ; 'k'
004FD945 mov      ecx, GLOBAL_hInstance
```

Also, in the demo the cursors don't exist and are unused. Again, why is the game trying to load non-existing cursors which it won't use, even if they would be available...?

Side Note:
   There is something interesting though which teaches us a bit about the compilation process of the demo: lack of optimization! In client 5.0.8.3 the array is initialized using "rep stosd", in the demo the array is initialized handle by handle. The (total) lack of compiler optimization teaches us that the demo has been compiled without any optimization settings.

   This is a good thing for us, the reversers.

INSIDE THE ALPHA CLIENT

Recently I acquired the UO alpha client (through betaarchive) and I loaded it into IDA for analysis. Strangely enough, the same cursor loading routine exists in this 1996 client ···:

```
00423CCB loc_423CCB:                             ; CODE XREF: FUNC_WindowProc+21↑j
00423CCB                 mov     edi, offset GLOBAL_CursorArray
00423CD0                 xor     eax, eax
00423CD2                 mov     ecx, 40h
00423CD7                 push    7F00h           ; lpCursorName
00423CDC                 rep stosd
00423CDE                 push    eax             ; hInstance
00423CDF                 mov     esi, ds:LoadCursorA
00423CE5                 call    esi ; LoadCursorA
00423CE7                 push    67h             ; lpCursorName
00423CE9                 mov     GLOBAL_CursorArray, eax
00423CEE                 mov     eax, hInstance
00423CF3                 push    eax             ; hInstance
00423CF4                 call    esi ; LoadCursorA
00423CF6                 push    6Bh             ; lpCursorName
00423CF8                 mov     GLOBAL_CursorArray+4, eax
00423CFD                 mov     eax, hInstance
00423D02                 push    eax             ; hInstance
00423D03                 call    esi ; LoadCursorA
00423D05                 push    69h             ; lpCursorName
00423D07                 mov     GLOBAL_CursorArray+8, eax
00423D0C                 mov     eax, hInstance
00423D11                 push    eax             ; hInstance
00423D12                 call    esi ; LoadCursorA
00423D14                 push    6Eh             ; lpCursorName
00423D16                 mov     GLOBAL_CursorArray+0Ch, eax
00423D1B                 mov     eax, hInstance
00423D20                 push    eax             ; hInstance
00423D21                 call    esi ; LoadCursorA
```

NOTE: the alpha client has been compiled with optimizations enabled too (notice the usage of "rep stosd"), this sort of makes the uodemo client very unique in terms of readability

··· with a subtle difference! The resources actually exist:
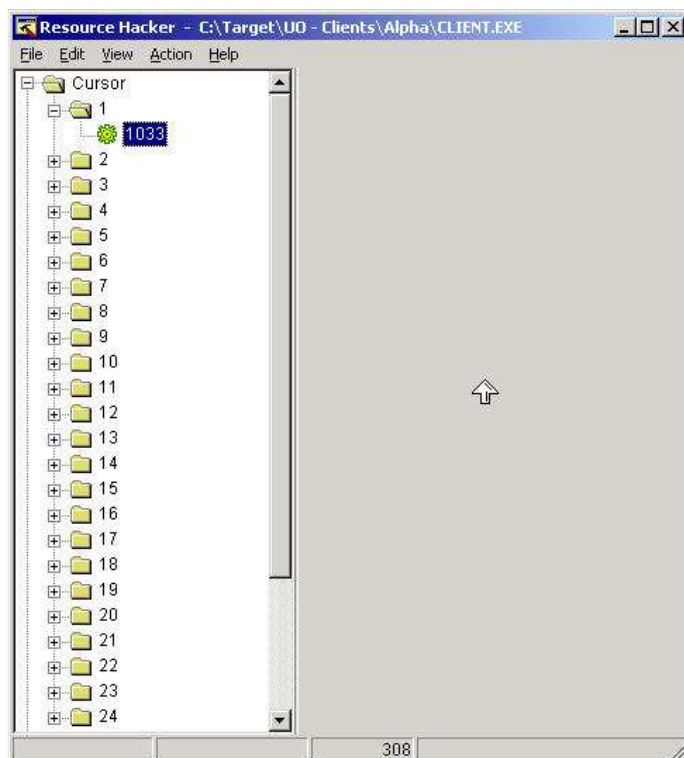
There is another more than subtle difference; the loaded resources are actually being used:


```
_xrefs to GLOBAL_CursorArray                                        _ □ ×

Dire...   T..        Address    Text

Up      r        sub_409130+14   mov   ecx, GLOBAL_CursorArray
Up      r        sub_409160+96   mov   edx, GLOBAL_CursorArray
Up      r         sub_40A400+1   mov   eax, GLOBAL_CursorArray
Up      r         sub_40A420+1   mov   eax, GLOBAL_CursorArray
Up      r         sub_40B590+1   mov   eax, GLOBAL_CursorArray
Up      r         sub_40C380+1   mov   eax, GLOBAL_CursorArray
Up      r         sub_40C3A0+1   mov   eax, GLOBAL_CursorArray
Up      r         sub_40D1E0+1   mov   eax, GLOBAL_CursorArray
Up      r        sub_413700+166  mov   eax, GLOBAL_CursorArray
Up      r         sub_41BA50+1   mov   eax, GLOBAL_CursorArray
Up      r         sub_41BE50+1   mov   eax, GLOBAL_CursorArray
Up      r         sub_41FDC0+1   mov   eax, GLOBAL_CursorArray
Up      o    FUNC_WindowProc:l...  mov   edi, offset GLOBAL_CursorArray
        w    FUNC_WindowProc+...   mov   GLOBAL_CursorArray, eax
D...    r         sub_426200+1   mov   eax, GLOBAL_CursorArray
D...    r         sub_426C10+1   mov   eax, GLOBAL_CursorArray
Up      r         sub_41C4B0+8A  mov   eax, [GLOBAL_CursorArray+4][eax*4]
D...    w    FUNC_WindowProc+...   mov   GLOBAL_CursorArray+4, eax
D...    w    FUNC_WindowProc+...   mov   GLOBAL_CursorArray+8, eax
D...    w    FUNC_WindowProc+...   mov   GLOBAL_CursorArray+0Ch, eax

            OK        Cancel        Help        Search

Line 17 of 20
```

INSIDE THE GOD CLIENT

During the UO:Renaissance period a GOD client was leaked. This GOD client (2.0.8n) also loads the cursors and just like the alpha client uses them:

```
0064D378                    mov      [ebp+VAR_Counter], 0
0064D37F                    jmp      short LOCAL_DoInitLoop
0064D381 ; ------------------------------------------------------------------
0064D381
0064D381 LOCAL_NextInitLoop:                      ; CODE XREF: FUNC_WindowProc+3CE↓j
0064D381                    mov      ecx, [ebp+VAR_Counter]
0064D384                    add      ecx, 1
0064D387                    mov      [ebp+VAR_Counter], ecx
0064D38A
0064D38A LOCAL_DoInitLoop:                        ; CODE XREF: FUNC_WindowProc+3AF↑j
0064D38A                    cmp      [ebp+VAR_Counter], 40h
0064D38E                    jge      short LOCAL_EndInitLoop
0064D390                    mov      edx, [ebp+VAR_Counter]
0064D393                    mov      GLOBAL_CursorArray[edx*4], 0
0064D39E                    jmp      short LOCAL_NextInitLoop
0064D3A0 ; ------------------------------------------------------------------
0064D3A0
0064D3A0 LOCAL_EndInitLoop:                       ; CODE XREF: FUNC_WindowProc+3BE↑j
0064D3A0                    push     7F00h           ; lpCursorName
0064D3A5                    push     0               ; hInstance
0064D3A7                    call     ds:LoadCursorA
0064D3AD                    mov      GLOBAL_CursorArray, eax
0064D3B2                    push     67h             ; lpCursorName
0064D3B4                    mov      eax, GLOBAL_hInstance
0064D3B9                    push     eax             ; hInstance
0064D3BA                    call     ds:LoadCursorA
0064D3C0                    mov      GLOBAL_CursorArray+4, eax
0064D3C5                    push     6Bh             ; lpCursorName
0064D3C7                    mov      ecx, GLOBAL_hInstance
0064D3CD                    push     ecx             ; hInstance
0064D3CE                    call     ds:LoadCursorA
```
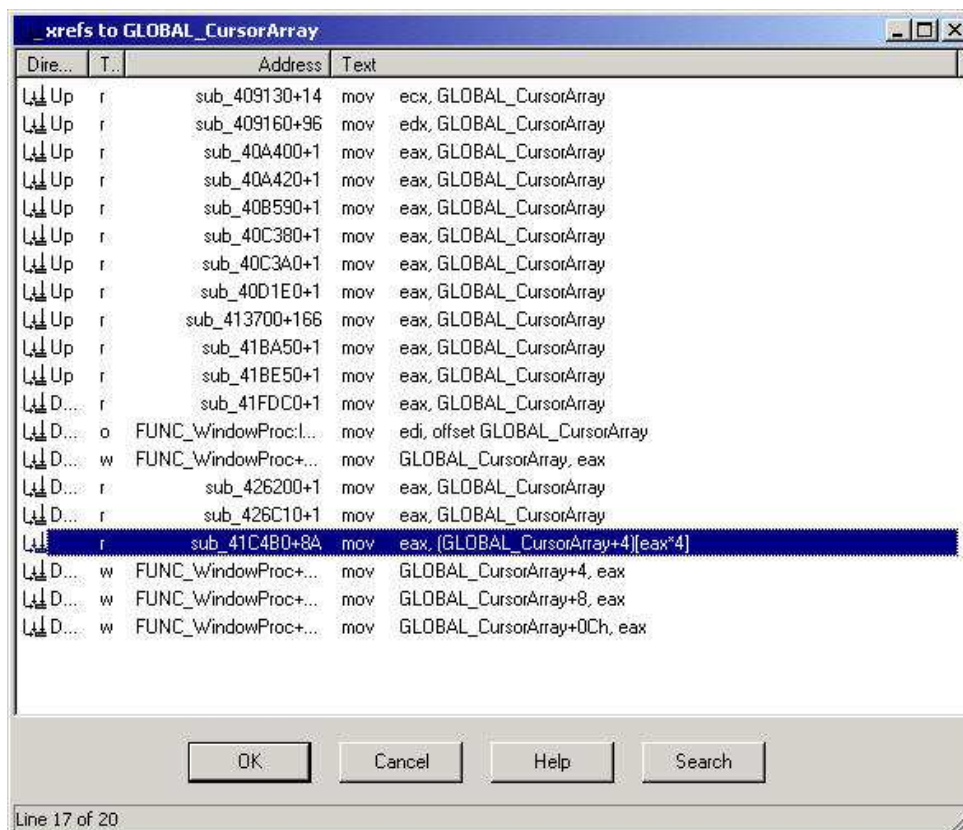
ANSWERS

Why is the game loading cursors that are not even defined?
Why is the game loading cursors it won't even use?

Let's review the time-frame:

1996	Alpha Client	Working Cursors
1998	UO Demo	Cursor Leftovers
2000	GOD Client	Working Cursors
2007	Client 5.0.8.3	Cursor Leftovers

The cursor loading in the 2007 client is clearly a leftover from the 1996 alpha client. The fact that the GOD client of the year 2000 is using the cursors makes me believe that the cursors from the alpha client are still being used by OSI for their modern GOD clients up to at least 2007 and probably even beyond.

The cursor loading routines should not be compiled in their public build of the client. This in my point of view an error, but not a fatal one.