INSIDE THE ULTIMA ONLINE CLIENT - PRE-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

<u>IDA Pro</u>, a very professional utility, definitely worth buying, Standard version is affordable <u>Resource Hacker</u>, 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:

xref	s to Lo	oadCursorA		
Dire	T.,	Address	*	Text
L <u>ul</u> Up	P	WinMain(x,x,x,x)+3CD	call	ds:LoadCursorA
<u>L₊∔</u> Up	E2	WinMain(x,x,x,x)+3CD	call	ds:LoadCursorA
<u>L,</u> LUp	P	sub_536C20+31B	call	esi ; LoadCursorA
<u>L,⊥</u> Up	Р	sub_536C20+308	call	esi ; LoadCursorA
<u>Lµ</u> Up	P	sub_536C20+2F5	call	esi ; LoadCursorA
<u>L₊⊥</u> Up	P	sub_536C20+2E6	call	esi ; LoadCursorA
<u>Lµ</u> LUp	P	sub_536C20+2D6	call	esi ; LoadCursorA
<u>Լ₊∔</u> Up	P	sub_536C20+2C6	call	esi ; LoadCursorA
<u>Լ₊∔</u> Up	P	sub_536C20+2B7	call	esi ; LoadCursorA
<u>Լ₊∔</u> Up	Р	sub_536C20+2A7	call	esi ; LoadCursorA
<u>Ļ</u> uup	Р	sub_536C20+294	call	esi ; LoadCursorA
<u>Լվ</u> Up	Р	sub_536C20+285	call	esi ; LoadCursorA
<u>Lŧ</u> LUp	Р	sub_536C20+278	call	esi ; LoadCursorA
<u>Li</u> ∐Up	Р	sub_536C20+265	call	esi ; LoadCursorA
<u>Lų</u> LUp	Ρ	sub_536C20+252	call	esi ; LoadCursorA
Li∐Up	P	sub_536C20+245	call	esi ; LoadCursorA
Ļ <u>⊎</u> Uр	P.	sub_536C20+236	call	esi ; LoadCursorA
<u>Li≟</u> Up	Р	sub_536C20+229	call	esi ; LoadCursorA
Ļ <u></u> ЦUр	Р	sub_536C20+21C	call	esi ; LoadCursorA
<u>Li</u> ∐Up	Р	sub_536C20+20C	call	esi ; LoadCursorA
<u>Lŧ</u> LUp	P	sub_536C20+1FC	call	esi ; LoadCursorA
<u>L,</u> Up	P	sub_536C20+1ED	call	esi ; LoadCursorA
<u>Lµ</u> LUp	Ρ	sub_536C20+1DD	call	esi ; LoadCursorA
<u>Lt</u> ∐Up	P	sub_536C20+1CD	call	esi ; LoadCursorA
Lu⊒Up	P	sub_536C20+1BE	call	esi ; LoadCursorA
Li₫Up	Ρ	sub_536C20+1AE	call	esi ; LoadCursorA
LitiUp	Р	sub_536C20+19E	call	esi ; LoadCursorA
L <u>i</u> ∐Up	Р	sub_536C20+18F	call	esi ; LoadCursorA
<u>Lŧ</u> LUp	Р	sub_536C20+17F	call	esi ; LoadCursorA
<u>Li⊒</u> Up	Р	sub_536C20+16F	call	esi ; LoadCursorA
L <u>LL</u> Up	P	sub_536C20+160	call	esi ; LoadCursorA
ЦЦUр	1	sub_536C20:loc_536	mov	esi, ds:LoadCursorA; jumptable 00536D21 case 0
000				
		Ωκ]	1	Cancel Help Search
			<u> </u>	
Line 32 o	f 32			7

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	1	LpCursorName
00535CEC	push	ebp	1	Instance
00535CED	call	ds:LoadCursorA		
00535CF3	mov	edi, ds:SetCurson		
00535CF9	push	eax	Ĩ	Cursor
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 536	D66:		; CODE XREF: sub 536C20+1011j
00536066			; DATA XREF: .text:off 5379C010
00536D66	mov	esi, ds:Lo	oadCursorA ; jumptable 00536D21 case 0
00536D6C	mov	ecx, 40h	and the second
00536D71	xor	eax, eax	
00536073	mov	edi, offse	et dword 89AA78
00536D78	push	7F 00h	; 1pCursorName
00536D7D	push	ebx	; hInstance
00536D7E	rep st	osd	and a second s
00536080	call	esi ; Load	dCursorA
00536082	mov	dword 89Af	A78, eax
00536D87	mov	eax, hInst	tance
00536D8C	push	67h	; 1pCursorName
00536D8E	push	eax	; hInstance
00536D8F	call	esi ; Load	dCursorA
00536D91	mov	ecx, hInst	tance
00536D97	push	óBh	; 1pCursorName
00536D99	push	ecx	; hInstance
00536D9A	mov	dword 89Af	A7C, eax
00536D9F	call	esi ; Load	dCursorA

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:

xref	s to sub	_536C20		
Dire	T.	Address	Text	
Lul Up	0	sub_5362C0+10C	mov	[esp+6BCh+WndClass.lpfnWndProc], offset sub_536C20
Lu⊒Up	o	sub_5362C0+19F	mov	WndClass.lpfnWndProc, offset sub_536C20
Լ <u>₊∔</u> Սр	o	sub_5362C0+251	mov	WndClass.lpfnWndProc, offset sub_536C20

And we convert dword_89AA78 into an array (which I named GLOBAL_CursorArray):

0089AA78	GLOBAL_CursorA	rray dd	?	Array size (in elements)	×I
0089AA78					-10
0089AA7C	dword_89AA7C	dd ?			
00894480	dword_89AA80	dd ?		Array element width : 4	
0089AA84	dword_89AA84	dd ?		Maximal possible size: 65	
0089AA88	dword 89AA88	dd ?		Current array size 1	
0089AA8C	dword 89AA8C	dd ?		Suggested array size : 1	
00898890	dword 89AA90	dd ?		and a second	
0089AA94	dword 89AA94	dd ?		Number of elements 64	
00898898	dword 89AA98	dd ?		have been for	
00898890	dword 89AA9C	dd ?		Items on a line U (0-max)	
00894440	dword 89AAA0	dd ?		Element width	
0089AAA4	dword 89AAA4	dd ?		(-1-none,0-auto)	
00898888	dword 89AAA8	dd ?			
0089AAAC	dword 89AAAC	dd ?			
0089AAB0	dword 89AAB0	dd ?		I ve "dup" construct	
0089AAB4	dword 89AAB4	dd ?		🔲 🔲 Signed elements	
0089AAB8	dword 89AAB8	dd ?		🗖 Display indexes	
0089AABC	dword 89AABC	dd ?		Create as array	
0089AAC0	dword 89AAC0	dd ?			
0089AAC4	11 H 1 8 20 1 1 1 1 1 1	db	?		
		1.		OK Cancel Help	
0089AA7	78: .data:GLOBAL Curso	rArray	8		-

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:

R	esou	rce Ha	cker -	C:\	×
File	Edit	⊻iew	Action	Help	
± (🗋 loo	in			
÷-(🗋 Str	ing Ta	ble		
Ð-(🗋 Ac	celerat	tors		
± (in Grou	цр		
<u>ا</u>	🗋 Ve	rsion I	nfo		
			901S		1

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:

	s to I	GLOBAL_CursorArray		
Dire	Τ.,	Address	Text	
나보 Up 나보 나보 D 나보 D 나보 D	0 W W W	FUNC_WindowProc+ FUNC_WindowProc+ FUNC_WindowProc+ FUNC_WindowProc+ FUNC_WindowProc+	mov edi, offset GLOBAL_CursorArray mov GLOBAL_CursorArray, eax mov GLOBAL_CursorArray+4, eax mov GLOBAL_CursorArray+8, eax mov GLOBAL_CursorArray+8, eax mov GLOBAL_CursorArray+8, eax	
Line 5 of	5	ОК	Cancel Help Search	

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], 8
                  short LOCAL DoCursorInitLoop
004FD8FD jmp
004FD8FF
004FD8FF
004FD8FF LOCAL_NextCursorInitLoop:
                  ecx, [ebp+VAR Counter]
004FD8FF mov
004FD902 add
                  ecx,
                  [ebp+VAR Counter], ecx
004FD905 mov
004FD908
004FD908 LOCAL DoCursorInitLoop:
                  [ebp+VAR Counter], 40h ; '@'
004FD908 cmp
004FD90C jge
                  short LOCAL EndCursorInitLoop
                  edx, [ebp+VAR_Counter]
004FD90E mov
004FD911 mov
                  GLOBAL CursorArray[edx*4], 8
004FD91C jmp
                  short LOCAL NextCursorInitLoop
004FD91E ;
004FD91E
004FD91E LOCAL EndCursorInitLoop:
004FD91E push
                 7F 00h
004FD923 push
                  8
                  ds:LoadCursorA
004FD925 call
                 GLOBAL_CursorArray, eax
67h ; 'g'
004FD928 mov
004FD930 push
                  eax, GLOBAL_hInstance
004FD932 mov
004FD937 push
                  eax
                  ds:LoadCursorA
004FD938 call
004FD93E mov
                  GLOBAL CursorArray+4, eax
004FD943 push
                 6Bh ; 'k'
ecx, GLOBAL_hInstance
004FD945 mov
```

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 PRE-ALPHA CLIENT

Recently I acquired the UO Pre-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+211j
00423CCB	mov	edi, offset GLOBAL CursorArray
00423CD0	xor	eax, eax
00423CD2	mov	ecx, 40h
00423CD7	push	7F00h ; 1pCursorName
00423CDC	rep st	osd
00423CDE	push	eax ; hInstance
00423CDF	mov	esi, ds:LoadCursorA
00423CE5	call	esi ; LoadCursorA
00423CE7	push	67h ; 1pCursorName
00423CE9	mov	GLOBAL CursprArray, eax
00423CEE	mov	eax, hInstance
00423CF3	push	eax ; hInstance
00423CF4	call	esi ; LoadCursorA
00423CF6	push	6Bh ; 1pCursorName
00423CF8	mov	GLOBAL CursorArray+4, eax
00423CFD	mov	eax, hInstance
00423D02	push	eax ; hInstance
00423D03	call	esi ; LoadCursorA
00423D05	push	69h ; 1pCursorName
00423D07	MOV	GLOBAL_CursorArray+8, eax
00423D0C	mov	eax, hInstance
00423D11	push	eax ; hInstance
00423D12	call	esi ; LoadCursorA
00423D14	push	6Eh ; 1pCursorName
00423D16	mov	GLOBAL CursorArray+OCh, eax
00423D1B	mov	eax, hInstance
00423D20	push	eax ; hInstance
00423D21	call	esi ; LoadCursorA

NOTE: the pre-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:

Resou	rce Ha	cker -	C:\Targ	jet\UO	- Clients\Al	pha\CLI	ENT.EXE	<u>- 0 ×</u>
jie <u>E</u> dit	⊻iew	Action	Help					
3 🔄 Ci	ursor			_				
ē 🖨	1							
	8 1	033						
÷ -	2							
	14							
	15							
÷.	6							
÷	7							
÷	8							
÷-	9							
÷ 🚞	10						5	
÷	111						Ŷ	
+ · · · · ·	12							
	114							
÷	115							
÷.	16							
÷-	17							
÷ 🗀	18							
÷	19							
÷ 🤶	20							
Ð-	21							
±-	22							
±	23							
	1.47					0		
		ale			30	8]		24

Dire	T.,	Address	Text	
<u>կ⊥</u> Up	r	sub_409130+14	mov	ecx, GLOBAL_CursorArray
L <u>ul</u> Up	ſ	sub_409160+96	mov	edx, GLOBAL_CursorArray
<u>Լվ</u> Up	r.	sub_40A400+1	mov	eax, GLOBAL_CursorArray
L <u>µ⊥</u> Up	r	sub_40A420+1	mov	eax, GLOBAL_CursorArray
Up <u>Li</u>	r	sub_40B590+1	mov	eax, GLOBAL_CursorArray
<u>u</u> ⊥Up	1	sub_40C380+1	mov	eax, GLOBAL_CursorArray
Lt Up	E2	sub_40C3A0+1	mov	eax, GLOBAL_CursorArray
<u>կվ</u> Սթ	ſ	sub_40D1E0+1	mov	eax, GLOBAL_CursorArray
Lil Up	Ê.	sub_413700+166	mov	eax, GLOBAL_CursorArray
L <u>ul</u> Up	ſ	sub_41BA50+1	mov	eax, GLOBAL_CursorArray
<u>կ</u> ∎Up	r	sub_41BE50+1	mov	eax, GLOBAL_CursorArray
L <u>ul</u> Up	r	sub_41FDC0+1	mov	eax, GLOBAL_CursorArray
L <u>ul</u> Up	o	FUNC_WindowProcil	mov	edi, offset GLOBAL_CursorArray
L <u>H</u>	W	FUNC_WindowProc+	mov	GLOBAL_CursorArray, eax
<u>ці</u> D	12	sub_426200+1	mov	eax, GLOBAL_CursorArray
나 <u>네</u> D	ſ	sub_426C10+1	mov	eax, GLOBAL_CursorArray
Lul Up	1	sub_41C480+8A	mov	eax, (GLOBAL_CursorArray+4)[eax*4]
<u>Ļ</u> ∎D	W	FUNC_WindowProc+	mov	GLOBAL_CursorArray+4, eax
<u>↓</u> D	w	FUNC_WindowProc+	moγ	GLOBAL_CursorArray+8, eax
<u>↓</u>	w	FUNC_WindowProc+	mov	GLOBAL_CursorArray+0Ch, eax
		ОК		Cancel Help Search
	6.00	117	÷ 74	- 77 W W 77 221 U

There is another more than subtle difference; the loaded resources are actually being used:

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 pre-alpha client uses them:

0064D378	mov	[ebp+VAR_Counter], 0
0004037F		SHOLE FORME DOTHER FOR
00040381	8	
0064D381	LOCAL NextInitLoop:	: CODE XREF: FUNC WindowProc+3CE1i
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	СПр	[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		5. 5
0064D3A0	LOCAL_EndInitLoop:	; CODE XREF: FUNC_WindowProc+3BE [†] j
0064D3A0	push	7F00h ; 1pCursorName
0064D3A5	push	0 ; hInstance
0064D3A7	call	ds:LoadCursorA
0064D3AD	mov	GLOBAL_CursorArray, eax
0064D3B2	push	67h ; 1pCursorName
0064D3B4	mov	eax, GLOBAL_hInstance
0064D3B9	push	eax ; hInstance
0064D3BA	call	ds:LoadCursorA
0064D3C0	mov	GLOBAL_CursorArray+4, eax
0064D3C5	push	6Bh ; 1pCursorName
0064D3C7	mov	ecx, GLOBAL_hInstance
0064D3CD	push	ecx ; hInstance
0064D3CE	call	ds:LoadCursorA

	s to I	GLOBAL_CursorArray			- O ×
Dire	T.,	Address	Text		
<u>L,⊥</u> Up	r	sub_409130+14	mov	ecx, GLOBAL_CursorArray	
<u>L,⊥</u> Up	ſ	sub_409160+96	mov	edx, GLOBAL_CursorArray	
<u>L,⊥i</u> Up	r	sub_40A400+1	mov	eax, GLOBAL_CursorArray	
<u>L,⊥</u> Up	r	sub_404420+1	mov	eax, GLOBAL_CursorArray	
<u>Lu</u> LUp	ſ	sub_40B590+1	mov	eax, GLOBAL_CursorArray	
<u>L,⊥</u> Up	f 2	sub_40C380+1	mov	eax, GLOBAL_CursorArray	
L <u>u</u> LUp	E	sub_40C3A0+1	mov	eax, GLOBAL_CursorArray	
<u>L,</u> ⊥Up	r:	sub_40D1E0+1	mov	eax, GLOBAL_CursorArray	
<u>L,⊥</u> Up	f (sub_413700+166	mov	eax, GLOBAL_CursorArray	
<u>L,</u> ⊥Up	ſ	sub_41BA50+1	mov	eax, GLOBAL_CursorArray	
<u>L,⊥i</u> Up	r	sub_41BE50+1	mov	eax, GLOBAL_CursorArray	
<u>t,</u> ∎D	r	sub_41FDC0+1	mov	eax, GLOBAL_CursorArray	
<u>ці</u> D	o	FUNC_WindowProcil	mov	edi, offset GLOBAL_CursorArray	
<u>ці</u> D	W	FUNC_WindowProc+	mov	GLOBAL_CursorArray, eax	
ι <u>,</u> μ <u>i</u> D	E	sub_426200+1	MOV	eax, GLOBAL_CursorArray	
(<u>네</u> D	1	sub_426C10+1	mov	eax, GLOBAL_CursorArray	
LLL	1	sub_41C4B0+8A	mov	eax, (GLOBAL_CursorArray+4)[eax*4]	
<u>Ļ</u> ∎D	W	FUNC_WindowProc+	mov	GLOBAL_CursorArray+4, eax	
<u>L,</u> ⊒D	w	FUNC_WindowProc+	mov	GLOBAL_CursorArray+8, eax	
<u>L,</u> ∎D	w	FUNC_WindowProc+	mov	GLOBAL_CursorArray+0Ch, eax	
		OK.	C	ancel Help Search	
Line 17 o	f 20				/ii

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	Pre-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 pre-alpha client. The fact that the GOD client of the year 2000 is using the cursors makes me believe that the cursors from the pre-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.