Gamefest

MICROSOFT GAME TECHNOLOGY CONFERENCE 2011

Microsoft

How Valve Makes Games Better with Xperf

Bruce Dawson
Senior Engineer
Valve

Introduction

- Xperf is awesome (see last year's Gamefest talk)
- Xperf has a "challenging" learning curve

- Talk goals:
  - Pass on the Xperf lessons learned at Valve
  - Pass on the techniques Valve uses (including sample code)
  - Encourage a common perf-interchange format
  - Force me to learn ETW/Xperf more thoroughly

What is Xperf?

- Free, whole system ETW profiling tool
  - ETW stands for Event Tracing for Windows
  - Disk, CPU, GPU, processes, threads, etc.
- Includes sampling profiler
- Used extensively by Microsoft
- Profiling without Xperf is also known as “guessing”
  - “I think our level loads are bound by I/O time” (there was none)
  - “I think our level loads are bound by CPU time” (wrong again)

Stuff Xperf Found

- 400 ms startup hang on Portal 2 and Dota 2
- 10 s of static lighting initialization on map load
  - On a game that didn't use static lighting
- 3 s of wasted time during map load
- 100,000 unintentional memory allocations
- Conditional breakpoint accidentally left enabled
- Excessive assert cost in debug builds
- Many, many, more

How Valve Uses Xperf

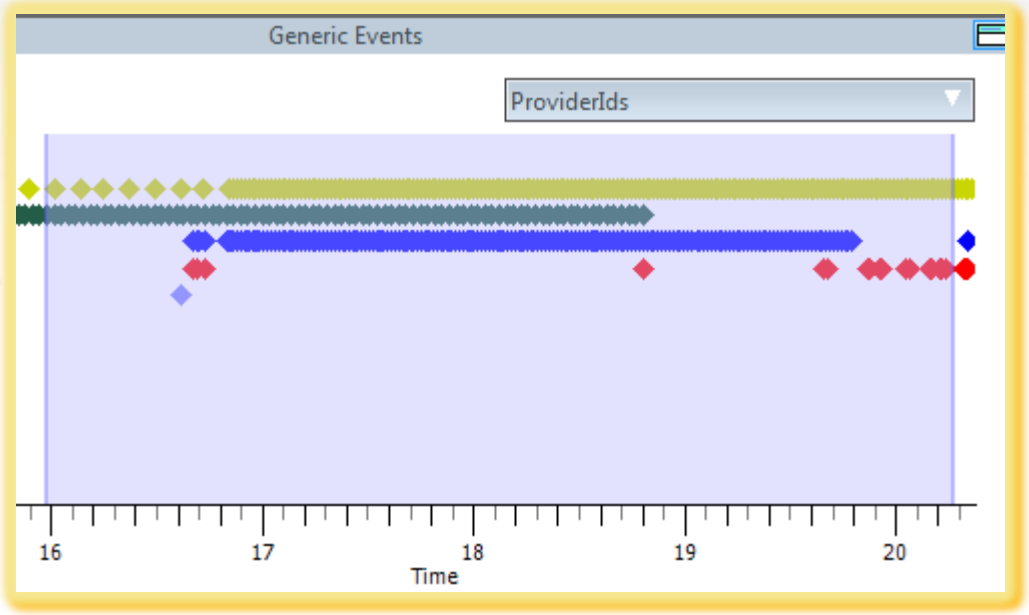
- You can record just system data
  - Sampling profiler, Disk I/O, page faults, context switches, DirectX information, memory allocations, etc.
- But, system data is much more valuable with context:
  - Frame start/frame rate
  - Key events (begin/end task, etc.)
  - Network traffic
  - User input
  - Etc.
- User providers let you provide this context

Context: Graph View

- Some things Valve's user providers tell us include:
  - User input
  - Frame boundaries
  - Network traffic

Rows of diamonds represent:
- Server frame rate
- Client frame rate
- Network events
- Start/stop events and markers
- User input

- One line per provider



Common Timeline

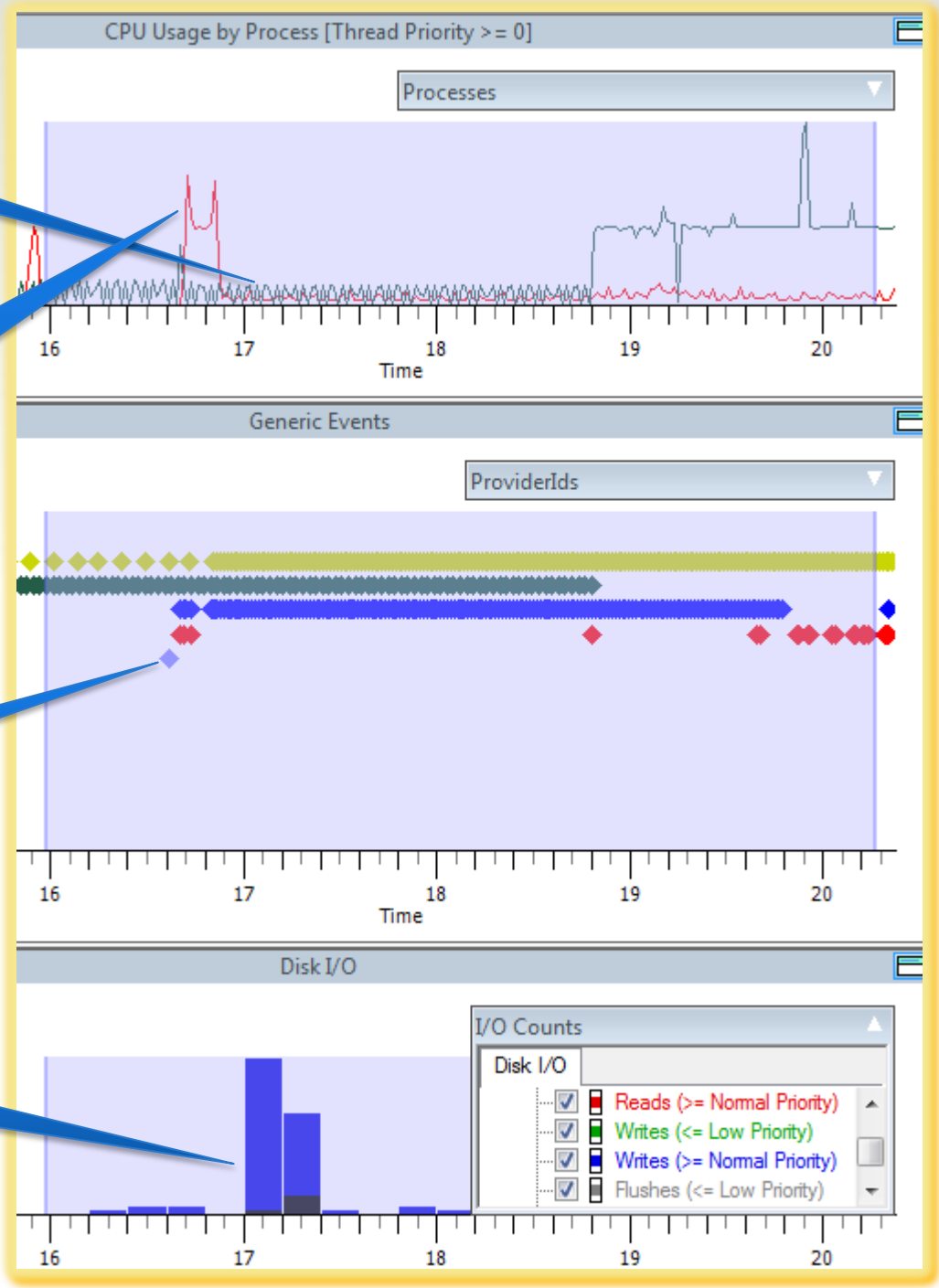
- The real power is when user events are in the same view as system events

Idle CPU

Server process CPU time

Map load request

Minimal disk I/O (all writes)



Digging Deeper: Summary Tables

- Grouping is powerful
- Time is recorded for all events
- Input events are awesome

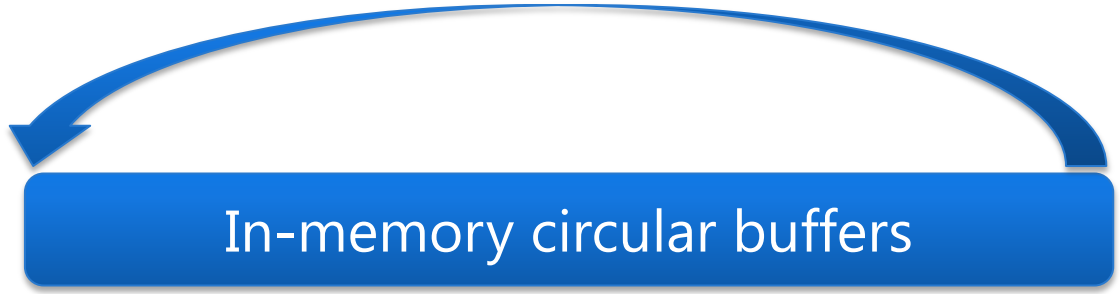
Line	Path Name	Process Name	Service Time (us)	Size	Write Size	Flush Cou...
1	Unknown	System	256,243.178	0	0	6
2	C:\kernel.etl	System	103,603.023	22,020,096	22,020,096	0
3	C:\$LogFile		4,029.507	401,408	401,408	0
4		BuildService.exe	2,489.030	294,912	294,912	0
5		System	1,540.477	106,496	106,496	0
6	C:\$Mft	System	1,413.732	286,720	286,720	0
7	C:\Windows\Prefetch\DOTA.EXE-...	System	1,322.419	261,632	261,632	0
8	C:\Users\bruced\NTUSER.DAT	System	1,142.031	54,272	54,272	0
9	C:\Users\bruced\NTUSER.DAT.LO...	System	1,122.553	55,808	55,808	0
10	C:\$BitMap	System	868.095	49,152	49,152	0
11	D:\$LogFile	System	541.801	57,344	57,344	0
12	C:\user.etl	System	387.913	65,536	65,536	0

Process	Provider Name	Opcode Name	P			
srcds.exe (7216)						
	Valve-Network					
	Valve-ServerFrameRate					
dota.exe (8148)						
	Valve-FrameRate					
	Valve-Main					
	Valve-Network					
	Valve-Input					
		KeyDown	d			
		KeyDown	d			
		KeyDown	dota.exe	"u"	14.582 353 331	
		KeyDown	dota.exe	"c"	14.705 349 189	
		KeyDown	dota.exe	"e"	14.879 358 763	
		KeyDown	dota.exe	"d"	15.048 781 056	
		KeyDown	dota.exe	"ENTER"	16.610 457 776	

80 wpm

How Valve Uses Xperf

- Tracing is always on
- Kernel data goes to a 600 MB circular buffer
  - ~2 minutes of data on a busy 12-proc machine
- User data goes to a 100 MB circular buffer
  - ~2-100 minutes of data depending on what providers are active
- Buffers can be saved to disk after a performance problem is noticed
  - Retroactive profiling



Demo

- Load various Xperf traces and show axtual isues found at Valve

Xperf Compared to Bracketed Event Profilers

- PIXBegin/EndNamedEvent style profilers coexist well with xperf
- Bracketed Event Profilers:
  - Have lower data rate, for faster data manipulation
  - Make slow frames easier to see
- Xperf:
  - Shows OS details (other processes, disk, locks, etc.)
  - Shows what happens between the bracketed events
  - Works when there are no events

ETW User Provider Definition (Manifest File)

```
<provider
  guid="{2B25961D-BA6E-4D79-BEC7-3605366E2E09}"
  name="Multi-FrameRate"
  symbol="MULTI_FRAMERATE"
  messageFileName="%temp%\MultiProvider.exe"
  resourceFileName="%temp%\MultiProvider.exe“
  >
 <templates>
  <template tid="T_FrameMark">
   <data inType="win:Int32" name="Frame number" />
   <data inType="win:Float" name="Duration (ms)" />
  </template>
 </templates>
<opcodes>
  <opcode name="FrameMark" symbol="_FrameMarkOpcode" value="10"/>  </opcodes>
 <tasks>
  <task name="Frame" symbol="Frame_Task" value="1“
                eventGUID="{43DADA85-49B6-4438-83D6-931477635DE3}"/>
 </tasks>
 <events>
  <event symbol="FrameMark" template="T_FrameMark" value="200" task="Frame" opcode="FrameMark" />
 </events>
</provider>
```

Provider definition

- Name, location, and GUID
- Can h
  in on

Event payloads. Available types include:

- Signed/unsigned 8-bit, 16-bit, 32-bit, and 64-bit integers
- ANSI and Unicode strings
- Float and Double
- Boolean, Binary, GUID, Pointer, FILETIME, SYSTEMTI

Static event data

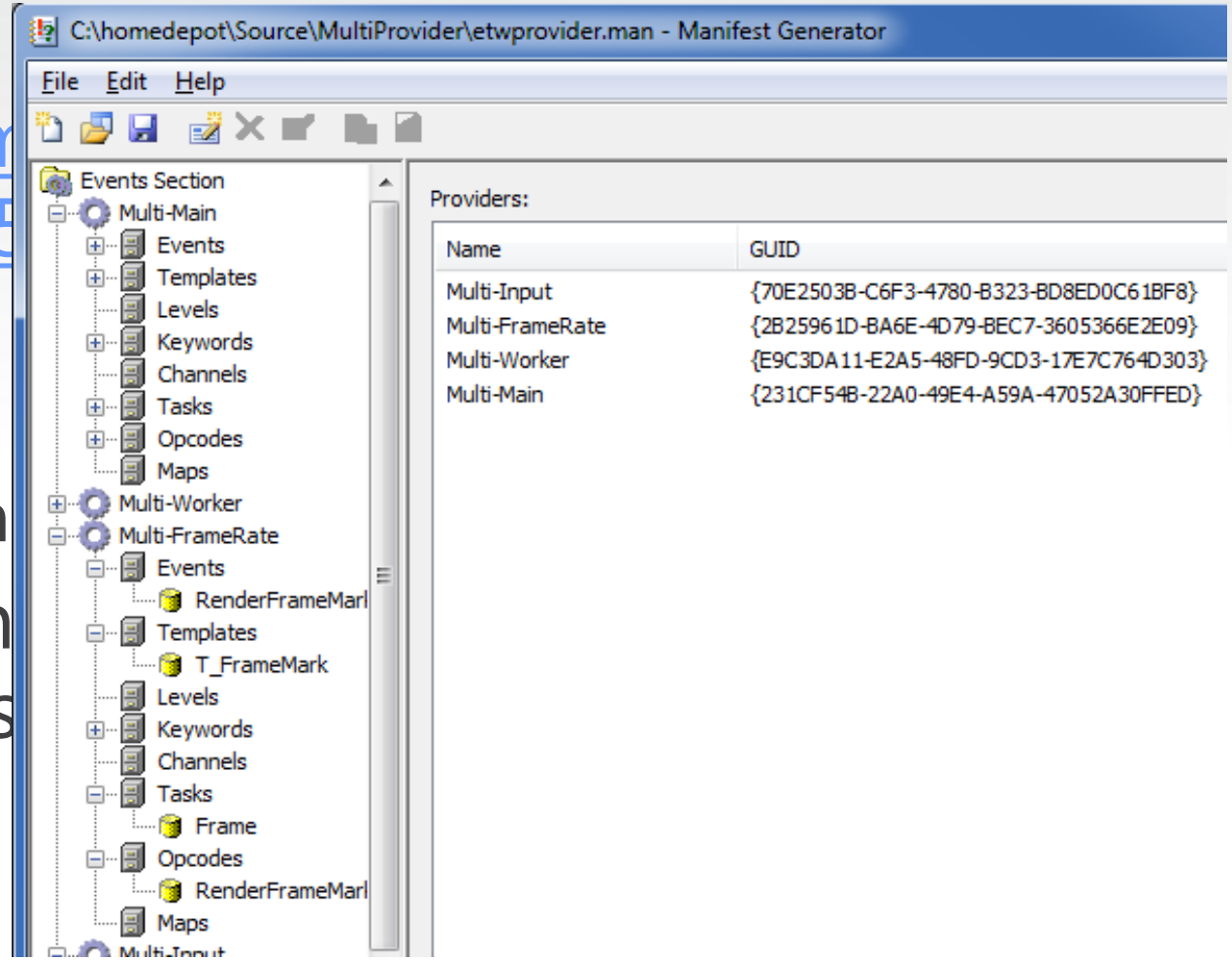
- Used to aid in interpreting, sorting and grouping

Event definitions

- Ties together payload and static data
- Your code emits events

Writing an Instrumentation Manifest

- http://msdn.microsoft.com/en-us/library/dd996930(VS.85
- You can use Visual Studio
  - Get \Include\Eventman.xsd XML->Schemas menu to in
- You can use Manifest_Gen ECManGen.exe) to edit ins
  - From the Platform SDK
- You can go old school
  - Notepad!



Compile Manifest

- mc.exe -um %(Filename)%(Extension) -z %(Filename)Generated
- Generates:
  - %(Filename)Generated.h
  - %(Filename)Generated.rc
  - %(Filename)Generated_MSG00001.bin (compiled into resource file)
  - %(Filename)GeneratedTEMP.bin (compiled into resource file)
- Don’t check in the generated files
- Don’t forget to build the resource file into your program

Valve's ETW API

```
#ifdef WIN32
PLATFORM_INTERFACE int64 ETWMark( const char *pMessage );
PLATFORM_INTERFACE int64 ETWMarkPrintf( const char *pMessage, ... );

PLATFORM_INTERFACE int64 ETWBegin( const char *pMessage );

PLATFORM_INTERFACE int64 ETWEnd( const char *pMessage, int64 nStartTime );

PLATFORM_INTERFACE void ETWRenderFrameMark();
PLATFORM_INTERFACE void ETWSimFrameMark();

PLATFORM_INTERFACE void ETWMouseDown( int nWhichButton, int nX, int nY );
PLATFORM_INTERFACE void ETWMouseUp( int nWhichButton, int nX, int nY );
PLATFORM_INTERFACE void ETWKeyDown( int nScanCode, int nVirtualCode, const char *pChar );

PLATFORM_INTERFACE void ETWSendPacket( const char *pTo, int nWireSize, int nOutSequenceNR, int nOutSequenceNrAck );
PLATFORM_INTERFACE void ETWThrottled();
PLATFORM_INTERFACE void ETWReadPacket( const char *pFrom, int nWireSize, int nInSequenceNR, int nOutSequenceNRAck );
#else
// Inline NOP functions for cross-platform compatibility
#endif
```

Valve's ETW API Implementation

- Startup/shutdown:

```
#include <ETWProviderGenerated.h>
EventRegisterValve_Network(); // Call this at process startup for each provider
EventUnregisterValve_Network(); // Call this at process shutdown for each provider
```

- Implementation

```
void ETWSendPacket( const char *pTo, int nWireSize, int nOutSequenceNR, int nOutSequenceNrAck )
{
    static int s_nCumulativeWireSize;
    s_nCumulativeWireSize += nWireSize;
    // EventWriteSendPacket is a macro in the generated header file
    EventWriteSendPacket( pTo, nWireSize, nOutSequenceNR, nOutSequenceNrAck, s_nCumulativeWireSize );
}
```

- XP compatibility thunks:

```
#define EVNTAPI __stdcall
#include "ETWProviderGenerated.h"
ULONG EVNTAPI EventWrite( REGHANDLE RegHandle, PCEVENT_DESCRIPTOR EventDescriptor, ULONG UserDataCount, PEVENT_DATA_DESCRIPTOR UserData )
{
    if ( g_ETWRegister.m_pEventWrite )
        return g_ETWRegister.m_pEventWrite( RegHandle, EventDescriptor, UserDataCount, UserData );
    return 0;
}
```

Demo

- Run Visual Studio, load the sample, and build it
- Register the providers
  - xcopy /y yourgame.exe %temp%
  - wevtutil um etwmanifest.man
  - wetvutil im etwmanifest.man
- Run the sample
- Record a trace, analyze it
  - Etwrecord.bat myfirsttrace.etl

Necessary Sample Customizations

- Replace all GUIDs in ETWProvider.man to avoid conflicts
- Rename provide ‘name’ and ‘symbol’ in ETWProvider.man
  - Also update etwcommonsettings.bat and etwprof.cpp to match new names/symbols
- Adjust ‘messageFileName’ and ‘ResourceFileName’ in etwprovider.man, and DLLFileMain and DLLFileAlternate in etwregister.bat

Technical Challenges

- 32-bit stack walking is buggy on 64-bit Windows Vista
  - Sampling profiler becomes useless
  - Luckily we use 64-bit Windows 7
- Xperf/ETW work on Windows XP (with many limitations), but won't install on Windows XP
  - Find 32-bit Windows Vista or Windows 7 machine, install there. Copy the install image
- Running applications off of non-system drive is busted
  - Use mklink/junction to hack around this
  - mklink /j c:\dota d:\dota
- No easy way to record traces on customer machines
  - Working on an installer and Xperf wrapper

Junction junction, what's your function…

Process Simplifications

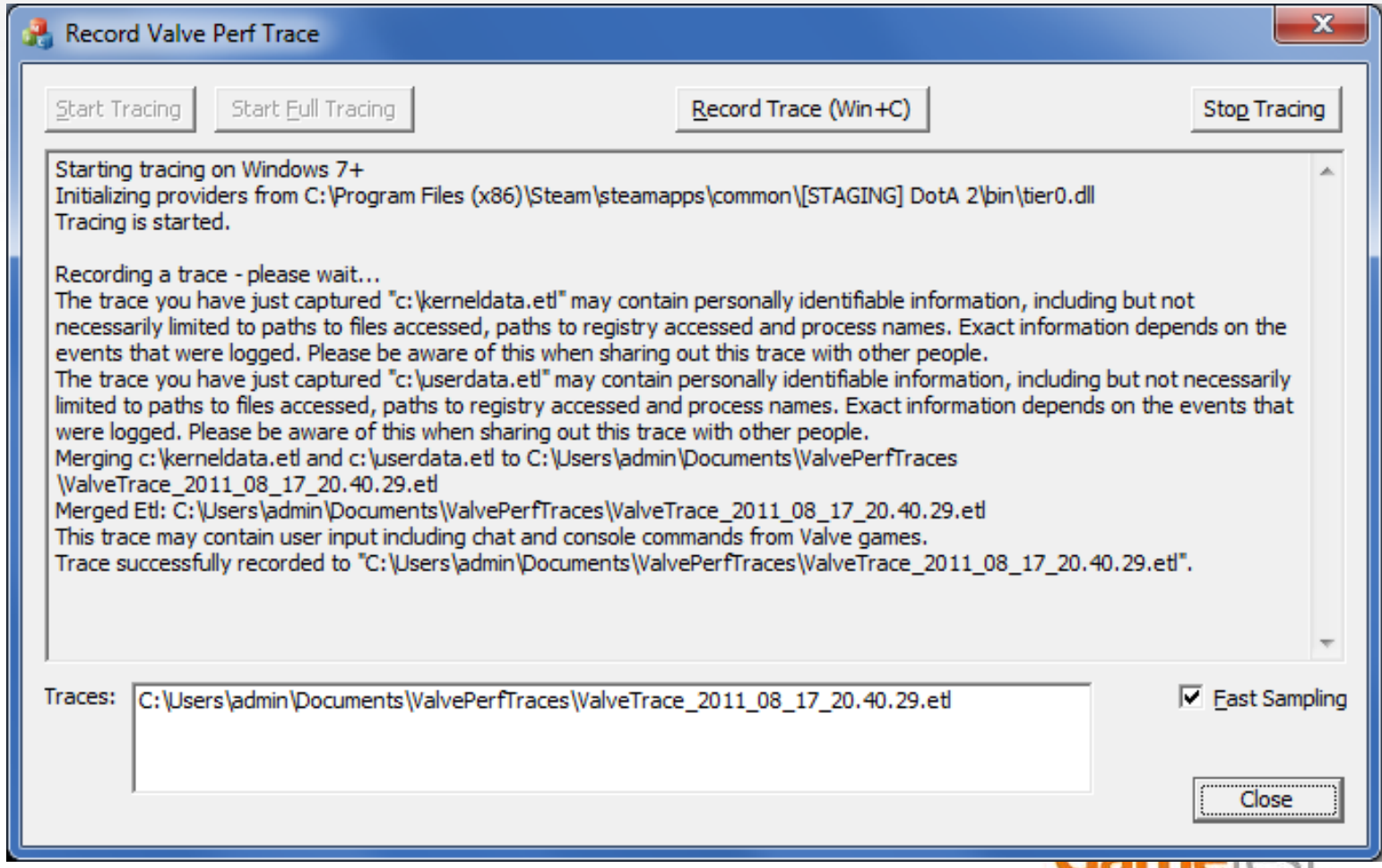
- Installation from Windows SDK is tedious
  - Make local distribution directory for xcopy install
- Syntax for recording tracing is byzantine
  - Make bullet proof batch files, put in distribution directory
- xcopy install is too much work (???)
  - Allow running batch file from network drive
- Typing batch file parameters is hard
  - Make all parameters optional
- Users might not register or have user providers
  - Make batch files fallback gracefully if providers aren't registered

More Process Simplifications

- 64-bit stack walking requires setting reg key, rebooting
  - Set reg-key in batch files
  - Get IT to deploy the reg key using group policy
- Frame Pointer Omission (/Oy) breaks 32-bit stack walks
  - Change all project files to /Oy-, default with VS 2010
- Developers won't run circular buffer recording
  - Automatically start/stop it when recording traces
- Creating junctions is annoying
  - Modify batch files to do it automatically
- Not everyone has _NT_SYMBOL_PATH set
  - Have batch files set it if not already set

Ultimate Simplification

- Redistributable Xperf wrapper is possible
- About three days work
- Installs xperf
- Buttons for starting and stopping circular buffer tracing, recording traces
- Also registers providers
- Adds global hot key for recording traces

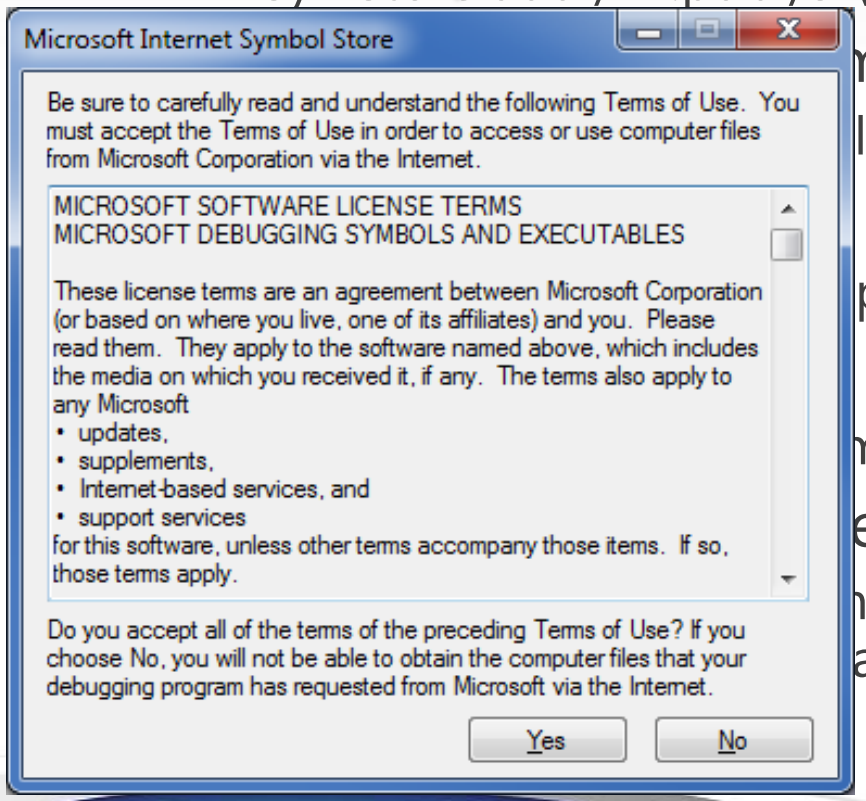


Sym

If this keeps showing up, then create a symsrv.yes file in the Xperf install directory

81% of CPU time in this region is in code I don't have symbols for

- Mic… …Server
  - http://m… …rosoft.com/download/symbols
- Your sym… …Server
  - You do h… …ve a symbol server, don't you?
  - symstore add /f *.pdb /s \\OurSymbolServer\s…



Module	Function	Weight
		6,470.785 163
[redacted].dll	Unknown	5,288.319 367
shaderapidx9.dll		609.482 736
	CTransitionTable::Find...	563.478 381
	LzmaDecode	22.001 903
	CTransitionTable::Crea...	8.000 924
	CTransitionTable::Crea...	5.999 416
	write_string	2.000 869
	__set_flsgetvalue	1.000 595
	BlitSurfaceBits	1.000 594
	_output_l	1.000 275

…mbols

…l server

…p!

…m, so add build directories to symbol path

…erf, windbg, VS, etc.):

…nygame\bin;SRV*c:\symbols*\\OurSymbolServer\symbols*http://ms…ad/symbols

Xperf as Perf-interchange Format

- No company is an island
  - Valve uses projects and source from other companies
  - Valve uses DLLs from other companies
  - We need them built with /Oy- and symbols so we can profile our game with this foreign code inside it
- When we hit problems in *your* code, we want to send you an ETW file
- If you hit problems in *our* code, we want you to send us an ETW file
- Common toolset allows sharing techniques and skills
- Common toolset allows reporting perf-bugs in others' code

Xperf for Other People's Software

- Recording graphics performance problems, sending traces to IHVs
- Found and reported opportunities for improved performance in PowerPoint, Visual Studio, and Windows Live Photo Gallery
- Used Xperf to profile third-party profiler, and itself
- Server performance problem due to Windows bug
- Profiling Valve's games before starting at Valve
- Poor network perf caused by network driver DPC time

Platform SDK

- Make sure you are getting the latest version
  - 7.1 as of this writing, available at http://msdn.microsoft.com/en-us/windows/bb980924.aspx
- Contains Xperf, and Xperf installers
- Also contains Manifest_Generator, GUID Generator, eventman.xsd
  - Plus other goodies like Application Verifier, debuggers, etc.
- Note that with Visual Studio 2010 SP1 installed, the Platform SDK will fail to fully install
  - To avoid this failure, don't install the SDK compilers
  - If they're needed, you can install them afterward from http://go.microsoft.com/fwlink/?LinkID=212355

Corrections

- The undocumented “-capturestate” option to xperf does not work on Vista
  - Sorry – you’ll have to fix the batch files
  - But tracing works better on Windows 7 anyway
  - And it’s not documented

Other Xperf Stuff I'd Cover if I Had Time

- Heap profiling
- GPUView
- Finding UI hangs
- Advanced threading analysis
- Summary tables, summary tables, summary tables
- Python script for packaging up .symcache files used by a trace

Resources

- http://randomascii.wordpress.com/category/xperf/
  - Resource links, slides, and sample project are here. Future updates will go here
- See sample code
  - Uses multiple ETW providers to record game-relevant data
  - Includes batch files for recording traces
  - Readme.txt explains what to do
- Last year's xperf talk
  - http://www.microsoft.com/downloads/details.aspx?familyid=14f10c84-8f31-412d-bcdf-4f1097bd8b5f&displaylang=en
- Platform SDK
  - http://msdn.microsoft.com/en-us/windows/bb980924.aspx
- Writing an instrumentation manifest
  - http://msdn.microsoft.com/en-us/library/dd996930(VS.85).aspx
- Documentation of all event payload template types
  - http://msdn.microsoft.com/en-us/library/aa382774(v=VS.85).aspx
- http://www.bing.com/search?q=xperfview
- http://www.bing.com/search?q=Event+Tracing+for+Windows

Microsoft®

www.microsoft.com