

## Forensic Debugging

Getting the truth by inference, autopsy, and putting the squeeze on informants

Elan Ruskin Valve Corporation GDC 2011

slides available at: **bit.ly/hPCmVW** 

VALVE

### VALVE

## Good Afternoon!

slides available at: **bit.ly/hPCmVW** 

Good morning everyone! Please turn off cell phones, fill out eval forms, etc.

### VALVE

### Who this talk is for:

CS grads

- (who know C++ and how a call stack works)

- Gameplay Programmers
- System Programmers
- Artists

- (try room 303?)

slides available at: **bit.ly/hPCmVW** 

This talk is for programmers of all levels:

Beginners will learn why the debugger sometimes lies to them and how to beat the truth out of it

Intermediates will learn techniques for dealing with core dumps of all kinds and divining the causes of really weird problems

Experts may be interested by our system for aggregating stability data during testing and even after release, and how we act upon that in support QA will learn all they need to know about what "crash dumps" are and how to collect them in a way that gets them fixed fast

Artists are in the wrong room.

Game Developers Conference <sup>®</sup> 2011		VALVE
	Today's Show	
What is a crash?	Basic Anatomy	The lying debugger
Typical Fingerprints	Collecting Evidence	Data After Shipping

Today's program, in six parts.

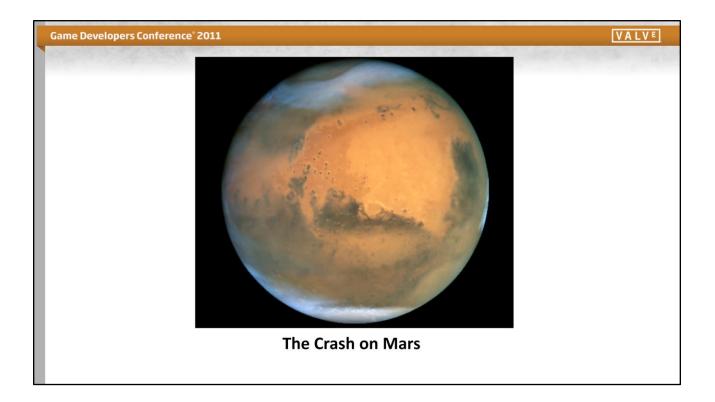
First, what is a crash? What exactly happens in the app and the operating system when one is encountered? Second, some basic anatomy of crash dumps and what they contain. Third, why the debugger is so often unreliable when dealing with crash data or optimized executables. Fourth, typical patterns of common crashes, and how to identify them. Fifth, how to collect all your evidence in one place, and deal with it efficiently. Sixth, how we collect data and act on stability data from our customers even after we've shipped to them.

This talk is about: Crashes	
	Collecting crashes
	Testing crashes
	Analyzing crashes
	Common causes of crashes
	Fixing crashes
	A crash course
The myst	terious "Disassembly" window in your debugger and why it is your best friend
Debuggir	ng "release" builds generally
	How to do it, why to do it, and why not to be afraid
	How to read a call stack by eye when the tools fail you
	Tips about MSVC and GDB you may not have known
	Techniques for a forensic approach to debugging
Tools for	collecting stability data from a lot of machines
	Best practices for QA while you're still testing internally
	How to get data from your game after you've shipped it
	How Valve gets data from five million testers customers around the world and turns it into
	patches
	And some handy open source tools: breakpad, socorro, and friends
Every pla	Itform Valve supports:
	PC, Mac, 360, PS3, Linux

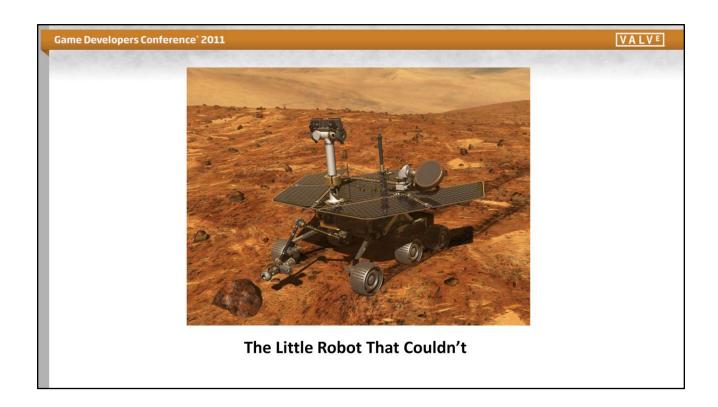
### VALVE

### ONCE UPON A TIME, ON A PLANET FAR FAR AWAY...

Let's start with a story.



A story about the crash on Mars. Not the probe that slammed into it because it mixed up miles and kilometers...



... but the Spirit rover, which went out of commission for many days because its control computer crashed.

### VALVE

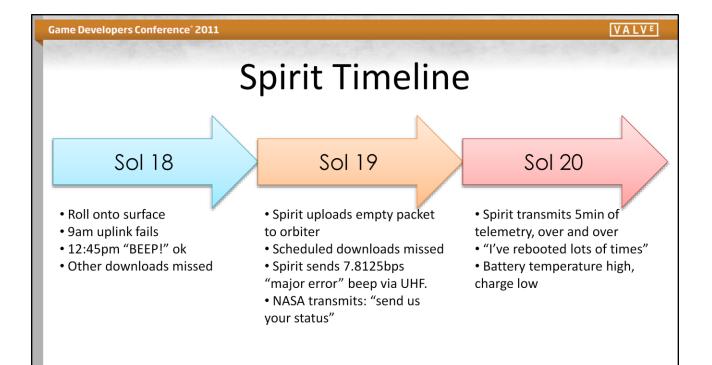
### Mars Spirit Rover

- IBM RAD6000 CPU (PowerPC derivative)
- 128mb DRAM
- 256mb Flash card (DOS filesystem)
- VxWorks OS in ROM

- Two radio antennas
  - Low-gain UHF
    - to Earth or to orbiter
    - <1kbps</li>
    - omnidirectional
  - High-gain X-Band
    - 11kbps to Earth
    - 128kbps to orbiter
    - Needs aiming
- 20 minute ping time!

Here's the specs on the Mars spirit rover's computer system. It has two ways of communicating by radio: a low-gain antenna, which is used for emergencies and during landing, and a high-gain antenna, which is like a satellite dish that communicates at a much higher baud. Either antenna can be used to transmit directly to Earth, or (at a faster rate) to the Mars Global Surveyor orbiting overhead, which can relay data between Earth and the probe.

This information from a great NASA paper: The Mars Rover Spirit FLASH Anomaly Glenn Reeves Tracy Neilson Jet Propulsion Laboratory (JPL) Pasadena, CA 91109 818-393-1051 Glenn.E.Reeves@jpl.nasa.gov, Tracy.A.Neilson@jpl.nasa.gov



## Sol 21

- <u>6</u>!#?@!
- Day 21, "low power fault" signal received
- Dedicated backup circuit detects batteries running low, tries to shut down
  - When shutdown fails, backup backup circuit unplugs the CPU!

### **Key Facts**

- Stuck in reboot loop
- Not shutting down overnight to conserve battery
- Not recording telemetry or science to FLASH

After midnight, the probe successfully turned on its UHF radio and trasnmitted to the orbiter for 2 minutes and 20 seconds, but sent no data. Ie, its radio was on and working properly, but had not been given data to transmit.

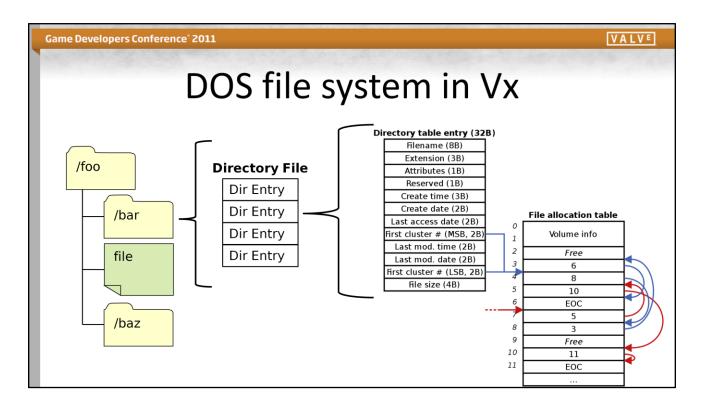
The probe missed its next appointed downloads, and finally sent the Blue Bleep Of Death at its "double backup emergency" super low bitrate.

NASA knew four things could put the probe in that mode: low power, broken system clock, loss of uplink, or an X-band radio fault. They knew the clock wasn't broken because Spirit beeped on time, the "uplink lost" timer hadn't expired yet, there was no indication of low power on day 18, and an X-band fault wouldn't have affected the UHF radio.

VALVE

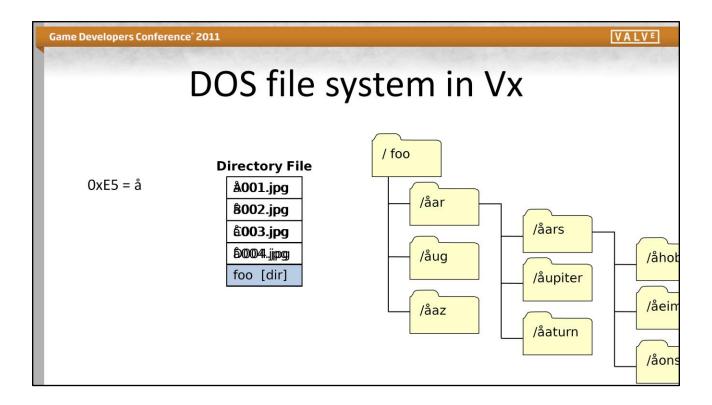
## Workaround

- NASA spams: "go to crippled mode, disable FLASH"
- Batteries recharge over the next day and CPU reboots in crippled mode.
- On day 32, sent "reformat FLASH drive" command
  - fixed problem (temporarily).
  - Why?



The DOS filesystem is a tree of subdirectories.

Each subdir is a file containing a list of filenames and disk addresses Deleting a file means just overwriting the first byte of its name with 0xE5 Mounting the filesystem means loading the directory trees into RAM



But this means that the amount of RAM needed to mount the filesystem increases with the number of deleted files! Even though the space is available "on disk", you still need storage to represent all those deleted filenames.

### VALVE

## DOS file system in Vx

- Two fatal misconfigurations:
  - DOS library allowed to malloc() space for deleted filenames, and run out of room
  - malloc() deadlocked when out of space
- PowerPC watchdog timer (DEC register) detects deadlocks and triggers exception when they occur (in this case reboot)
- CPU would boot, mount filesystem, run out of room, deadlock, reboot, mount filesystem, run out of room, deadlock....
- NASA uploaded a patch in April 2004

Two configuration errors conspired to place the system into a condition where it would reset repeatedly and also prevented the vehicle from autonomously shutting itself off to save power. A configuration error in the DOS Library module allowed the size of the private memory area to expand by allocating additional space from the free system space12. A configuration error in the Mem Library module silently resulted in a suspended task when the request for additional memory could not be satisfied.

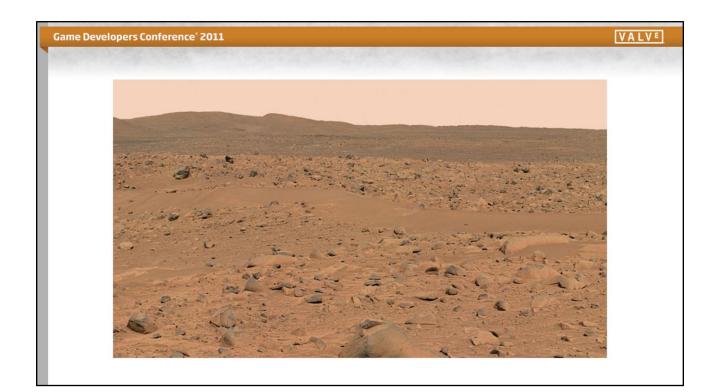
VALVE

## NSALIVE

... in what is probably the first ever example of interplanetary DLC. The patch included:

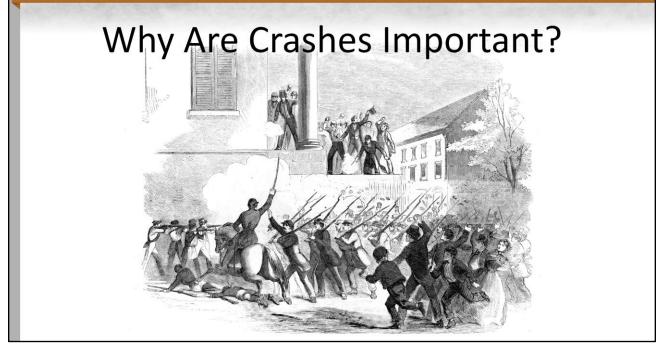
Compaction of subdirectories after files are deleted to yield back the used space Automatically entering crippled mode after repeated resets Using watchdog timer to force overnight shutdown if "normal" shutdown deadlocked Put a timeout on the semaphore that waited for the FLASH library functions to return

Oh, and not using malloc after initialization!



And so they Saved Science.

### VALVE



Why do we care about crashes?

Raise your hand if you've ever had a game crash on you. Keep your hand up if you thought this was really annoying. Keep your hand up if you ever lost progress to a crash. Look around the room. Those people with their hands up are your customers. [display image]

Even if you hate your customers, an unstable game will usually fail console certification.

The question isn't really "why are crashes important" but "why is it important to fix them this way?"

## Why Treat Crashes Specially?

- Extremely annoying to customer
  - The ultimate "showstopper"
- Relatively easy to fix (compared to severity!)
- State snapshot is better than waiting for repro
- A special case of "release mode forensics."

The question is really, why treat crashes differently from other bugs?

Well, they're extremely annoying to the customer, and they can make you fail cert outright more easily than almost any other bug.

But at the same time, once you know how to diagnose them, it's often obvious what caused the crash.

So crashes have a pretty high ratio of how annoying they are over how hard they are to fix.

Also, crashes are unique in ways that makes them debugabble by means not available to most other bugs: in particular, it's often easier \*not\* to wait for a repro, since you have a snapshot of the crash's state.

And really this talk isn't so much about crashes particularly as a forensic approach to debugging: how you can look at the current state of a process, at its memory and stack, and figure out what's gone wrong and what to do about it.

### VALVE

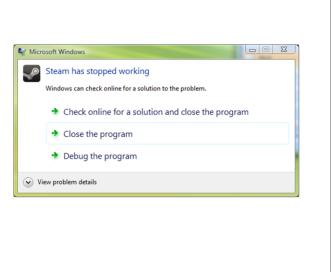
### WHAT IS A CRASH?

And why are they hard to debug?

Unless you're in aerospace, when you say "the computer crashed" you don't mean it actually blew up and died. You mean that your application stopped running because it hit some exceptional condition and went to "plan B."

## A Crash is "Plan B"

- A programmed response to an exceptional condition.
- Protect system from malfunctioning apps
- Allow recovery
- Provide diagnostics
  - The "core dump"



VALVE

A crash is a programmed response to an exceptional condition. It's a Plan B you've filed with the operating system ahead of time, to tell it what to do in the case your program can't continue.

Plan B is designed to:

Protect the computer from a malfunctioning program

Allow the user to recover and regain control of their OS (more so in Win/Mac than consoles)

Provide information to help fix the problem

But sometimes this information is hard to interpret

Debugging a crash is like performing an autopsy: you're looking at the state of the thing after it has died, and trying to figure out the sequence of events that got it there

The default exception handler for the Windows OS is this familiar dialog, which lets you close the program, send back diagnostic information, or open a debugger.

This is the technical description, but I find there's another metaphorical way to look at crashes that's a more helpful mindset for actually fixing them.

### VALVE

### A Crash is a crime scene



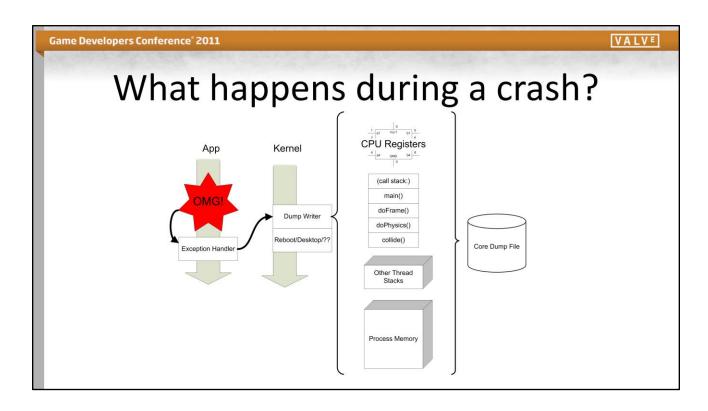
# initial praims exyme seene

... a crash is a crime scene.

Debugging a crash is like performing an autopsy: you're looking at the state of the thing after it has died, and trying to figure out the sequence of events that got it there

You have to think like CSI.

- Sometimes the primary cause is in the past and you infer it from secondary evidence
- Sometimes the state of the victim is a little bit ground-up and you need to piece it back together like a puzzle.



Whatever the cause, the program went into an exception handler -- not a C++ exception, but a "Plan B filed in case of emergencies"

Typically, this halts the program, records some information about the current state of the program, and goes into a recovery state -- shutting down the process in a Windows machine, or simply halting/rebooting a console.

"Recording the current state" can take many different forms, but usually it includes the current state of the CPU, a call stack, and sometimes some of the memory of the current thread stack, other threads, or the process heap.

### VALVE

## Where does the dump go?

### Dump to Disk

vorite Links	Name () fix.bat	Date modified	Type	39	B B 🖗 💭 🖳 R B 🗣 🕶 🕬	0 45 3 × 2 8 5 8				
Dropbox	hl2.dat		Make_360_Config.bat		H Wy Targets	Path: /dev_hdd0/game_debug				
Documents	📄 hl2.map		Make_PS3_Config.bat		a) a 400p test box	Name	Extension	Size	Mode	Modified
Pictures More » ders v portal2stagin * backup content dvd_image +			MakePortal2,360_Increm MakePortal2,360_Langu MakePortal2,360_Perform MakePortal2,360_SyncA MakePortal2,360_SyncA MakePortal2,P53_Increm MakePortal2_P53_Langu	age.bat cePoll.bat d.bat ndBuildBins.b nental.bat	Bank's DECH     Bank's DECH     Bank's DECH     Ban Decg Wood's 1400     Ban DECH     Ban     Ban	Dps3core-1286105685-0x01000500-EBCOT.81N.eff	elf	419 MB	-rw	27/01/2011 00:21
hl2_4345 Crash Du	_crash_2010_9_30T22_50_12C0 mp File	Siz	± 9/30/2010 2:50 PM = 88.9 KB ± 9/30/2010 2:50 PM		Web page Resources E Filk Robson E Filk Robson I do DECR1400 S S Iestyn DECR1000 (Device	Transfers ID % Status Source			Destina	tion

Different things happen after this. Where does the data go?

The simplest possibility is that it just gets written to disk as a big file – either to your local filesystem, or a file server, or wherever. This is convenient for in-house testing, when you can just double click the file and get the debugger to open.

en and a constant						VALVE
Valve - Site Data Last Updated: 16-Jan-2011 Product Hotilat Over The Last 90 Days for product: Stear the fulfit age device the agelieting end stellow.	nere (	Event Details for Event ID: 851866070 and Eve Event Signature Application Name Reader Name Reader Name Control Control Exception Carlo Data Collection and Responses Catistics Data Stepses Catistic	nt Type: Special Exception	On This Page - Fort Line Lineation - Reform Datelli	0x1	
Show filter     Reports by volume		Event Time Trending Details			0x0 0x3iiiiii ci	4000005 4000005
Depart averal         Cable         Response         Arg. Hills         Table Arg. Hills           Special Exception         I </td <td>Aglication Name Aglication Varian Aglication Link Date (free) Maduk Varian Maduk V</td> <td></td> <td>Total HIMS 2116 First Yanne II 2006-01 Last Yanne II 2006-01 Average HIM: 2013 Event Type Name: Good Ecopton</td> <td></td> <td>0x0 d 0x1 d 0x0 d 0x3 d 0x0 d 0x0 d 0x0 d 0x0 d 0x1 d</td> <td></td>	Aglication Name Aglication Varian Aglication Link Date (free) Maduk Varian Maduk V		Total HIMS 2116 First Yanne II 2006-01 Last Yanne II 2006-01 Average HIM: 2013 Event Type Name: Good Ecopton		0x0 d 0x1 d 0x0 d 0x3 d 0x0 d 0x0 d 0x0 d 0x0 d 0x1 d	
Nong         G3         W         J, J0         Z/J           Image         G3         W         J, J0         Z/J         Z/J           Simon Tang         G         P         Z, J0         Z/J         Z/J           Simon Tang         G         P         Z, J0         Z/J         Z/J           Simon Tang         G         P         Z, J0         Z/J         Z/J           Mang         G         P         Z, J0         Z/J         Z/J           Mang         G         P         Z, J0         Z/J         Z/J           Mang         G         P         Z/J         Z/J         Z/J           Secold Exception         G         P         Z/J         Z/J         Z/J           Secold Exception         G         P         Z/J         Z/J         Z/J           Secold Exception         G         P         Z/J         Z/J         Z	<b>?</b> Repret additional cak data	Platform Details - 90 days Operating System Breakest: Woolses V days (S2) 2024 Woolses V days (S2) 2024 Woolses V days (S2) 2024 Woolses V days (S2) 2024 Norther V days	Language Broaknet Popular London Dana Ogeneration Commony Rights - London Common Rassien - Rassie Pranse - Pranse Tanlans - Ray Baselant - Sandre Common - Autoin Common - Autoin Common - Autoin	14.39%	0x1 ct	

In Windows, it can get sent back to Microsoft via Windows Error Reporting (aka Watson), and you as a developer can have Microsoft send that along to you via Winqual.

ame Developers Conference <sup>®</sup> 2011		VALVE
Where	does the	dump go?
Image: Contract to the service servic	Thread 0 Crashed: 0 ??? 1 com.apple.CoreFoundation 2 com.apple.HIToolbox 4 com.apple.HIToolbox 5 com.apple.HIToolbox 6 com.apple.AppKit 7 com.apple.AppKit 8 com.apple.AppKit 9 com.apple.AppKit 10 com.apple.TextEdit	0000000000 0 + 0 0x942cf0fe CFRunLoopRunSpecific + 18 0x942cfd38 CFRunLoopRunInMode + 88 0x919e58a4 RunCurrentEventLoopInMode 0x919e550b ReceiveNexEventCommon + 0x919e5531 BlockUntilNexEventMatchi 0x9390bd5b _DFSNexEEvent + 657 0x9390bd5b _DFSNexEEvent + 657 0x9390bd6d -[NSApplication nexEVent 0x939046d1 -[NSApplication nextEvent 0x938d19ba NSApplicationKain + 574 0x00001df6 0x1000 + 3574
Cooption Type: DCLAGACCESS (10000)         Require into access to the model of		

On the Mac platform, app crashes automatically go into MacOS's built in crash reporter app. This collects a call stack (not the entire stack) and writes it to a text file. The customer can push "Report" to upload this text file to Apple, where it falls beneath an event horizon and is never heard from again. They don't give devs access.

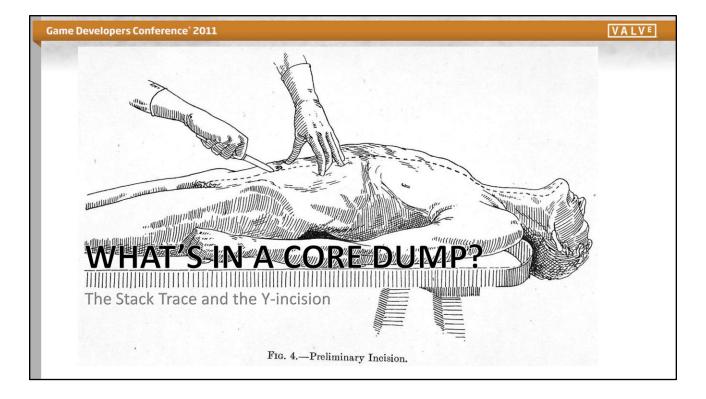
Because Mac CRT libraries are often built with symbols included, you'll usually get useful stacks even if you blew up inside the OS.

http://developer.apple.com/library/mac/#technotes/tn2004/tn2123.html

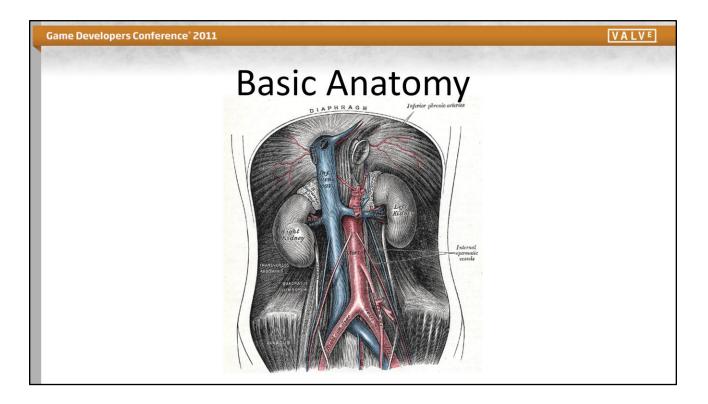
VALVE Game Developers Conference<sup>®</sup> 2011 Where does the dump go? **Build System**  Write your own Breakpad symbo strip debug Google Breakpad Talkback distribute program to users copy symbol file Crashrpt Breakpad Crash! Breakpad client minidump rites minidump... etc and submits it to crash collecto User's System **Crash Collector** 

Or you can always write your own handler and do whatever you want -- including sending this data back to you.

Breakpad is one open source solution, which we use, but there are alternatives. Certain digital distribution services provide this to you



What clues to cause of death can we learn from looking inside the body?



In doing an autopsy, it helps to know where everything is and how it's supposed to be put together generally. In particular, you'll often find yourself manually poking through call stacks to find function parameters. Data has a tendency to ricochet around program memory like a bullet inside the thorax,

	s Conference' 2011	20.0	10.25			- THE
	<b>A A</b>		1 110	•		
	Case St	110	1い 廿(			
		u	<u>лу п</u>	<u> </u>		-
	viewpostprocess.cpp		Watch 1 Find Results 2 Men Addresse 0x7005E050	nory1 × Threads Autos Modules	mes 4	-
	ChartenderContextPtr pRenderContext( materials ); if ( im_pDef->IsScreenSpaceEffect() && im_pDef->IsViewNodelEffect() )	4	0x70052050 7005e210 7005e1b4	00000000 7005elac		*
	{ Vector vecCamera;	· ·		00000000 84e0dfc8 00000000 84e0e3f4		
	pRenderContext->GetWorldSpaceCameraPosition( &vecCamera );	122 122		84d7d9d0 84e0e3f4 00000000 dc0392b0		
	<pre>if ( CalcSqrDistanceToAABB( m_NinBounds, m_NaxBounds, vecCamera ) &gt; ( m_pDef-&gt;m_flMaxDrawDi return 0:</pre>	istance * m	0x7005E0A8 00000000 00000000	84d9b3b4 dc0392b0		
	3			84d9b260 00000000 00000000 7005e23f		
	<pre>if ( m_pDef-&gt;IsViewNodelEffect() )</pre>		9x70055000 00000000 7005e237	00000000 7005e22f		
	(			00000000 84e0dfc8 00000000 84e0e3f4		
	<pre>C_BasePlayer* pPlayer = C_BasePlayer::GetLocalPlayer(); if ( !pPlayer    !pPlayer-&gt;IsAlive()    !pPlayer-&gt;GetRenderedWeaponModel()-&gt;IsViewModel() )</pre>		0x7005E100 00000000 84e0e3fc	84d7c6f8 00000078		
	return 0;			00000000 dc0392b0 84d946b4 84e0e244	0	3
	)			84d9b420 00000001		
	if ( ( flags & STUDIO_TRANSPARENCY )    !IsBatchable()    !m_pDef->IsDrawnThroughLeafSystem() )			84d9aa1c dc0392b0 00000000 410de018		
	<pre>{     C BaseEntity *pCameraObject = GetSplitScreenViewPlayer();</pre>			ffffffff 88db88a5		
	<pre>c_baseintity "pCameraObject = GetSplitScreenvlewPlayer(); if( ((C BasePlayer *)pCameraObject)-&gt;GetViewEntity() )</pre>	C3		0000000 0000000		
	(		ex7005E180 3fd20000 00000000 ex7005E190 40000000 00000000	40540000 00000000 404ca5dc 1a63c1f8		
	<pre>pCameraObject = ((C_BasePlayer *)pCameraObject)-&gt;GetViewEntity(); }</pre>			bffeesee eeeeeee		
				00000000 411707f8 ffffffff 88db88a5		
	C RaseEntity foSkinRenderChiert = / m nDef.am oSkinRenderControlPoint 1= -12 GetControlPoint1	intitu/ m		00000000 000001e0		-
	Cell Stack	• # X Local				×
	Name	Langua + Nar	ne	Value	Туре	
			• this	0x00000000 (m_References=() m_pNext=		
	client 360.pe/C_Portal_Player::RenderScreenSpacePaintEffect()MatRenderContext * pRenderContext=0;dfffffff) line 3794 client 360.pe/C_Portal_Player::RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect)MatRenderContext * pRenderCo		flags instance	0x00000000 (m_nAlpha=??? )	int const RenderableInstance_t &	
	client_360.petDoEnginePostProcessing(int x=0xfffffff, int y=0xffffffff, int w=0xffffffff, int h=0xffffffff, bool bFlashlightlsOn=true, bool bF	C++ (8)	pRenderContext	()	CMatRenderContextPtr	
	client_360.petCViewRender::RenderView(const CViewSetup & view=[], const CViewSetup & hudViewSetup=[], int nClearFlags=0xffff C		VecCamera	(x=-5.11726756e-036 y=-1.48138026e+017		
			vecDiffuseModulation	(x=0.000000000 y=0.00000000 z=-1.#QNA (x=-5.07289810e-036 y=1.6816e-043#DEN z		
	engine_360.pelSCR_UpdateScreen() line 283 0	C++	maxs	{x=0.000000000 y=-5.28676575e-036 z=0.00	0 Vector	
			e center	{x=0.000000000 y=1.65739782e+029 z=-1.#		
		C++ (E)	vecPos	{x=0.000000000 y=1.65740084e+029 z=0.00	ul vector.	
	engine_360.pelCHostState::State_Run(float frameTime=0.000000000) line 557 0	C++				
		C++				
		C++				
	engine_360.pelCAppSystemGroup::Run() line 774 + 0x14 bytes 0					
PPC		0++ 0++				
		C++ *				

Let's warm up with a really simple crash that was sitting on my hard drive.

	Conference <sup>®</sup> 2011							
viewpostprocess.cpp	c_portal_player.cpp 🛍 particles_new.cpp 🛎 🗙 Di	sassembly client_360.dll.dmp		Watch 1 Find I	Results 2 Memory 1 ×	Threads Autos	Modules	-
(Unknown Scope)	•			Address: 0x7005E0	50		- (a) Columns: 4	
if ( lm_pD { Vector pRende if ( (	Contextvrr pkengercontext( materials ); ef>isscreenSpaceEffect() && im_DDef>isviewik vecCamera; rContext-SettiorIdSpaceCameraPosition( &vecCam rCotext-SettiorIdSpaceCameraPosition( &vecCam calcSprDistanceToAABB( m_MinBounds, m_MaxBoun turn 0;	iera );	DrawDistance * m	0x7005E060         fff           0x7005E070         700           0x7005E080         000           0x7005E090         700           0x7005E040         000	5e210 7005e1b4 000000 fffff 84d8ed70 000000 5e110 448eb6bc 0000000 5e100 84e0e3fc 84d7d9d 5e130 84e0e3fc 0000000 00000 00000000 84d9b3b 5e130 84e0e204 84d9b26	0 84e0dfc8 0 84e0e3f4 0 84e0e3f4 0 dc0392b0 4 dc0392b0 0 0000000		
{ C_BasePla	IsViewNodelEffect() ) yer* pPlayer = C_BasePlayer::Get ayer    !pPlayer->IsAlive()    ! n 0;		nModel()->IsV	iewModel() )	00 84e0e204 84d946b 90 84e0e204 84d9b42 b0 00000280 84d9aa1 b0 0000000 0000000	0 7005e22f 0 84e0dfc8 0 84e0e3f4 0 0000078 0 dc0392b0 4 84e0e244 0 0000001 c dc0392b0 0 410de018		0
1 <b>f</b> ( (() {	Entity *pCameraObject = GetSplitScreenViewPlay C_BasePlayer *)pCameraObject)->GetViewEntity() ameraObject = ((C_BasePlayer *)pCameraObject)-	)	C	0x7005E180 3fd 0x7005E190 400	0000000         ffffff           0000000         0000000         0000000           20000         0000000         4034000           00000         0000000         404ca5d           00000         0000000         bf0000	0 00000000 0 00000000 c 1a63c1f8		
100 % a f	Entity =nSkinRenderObject = ( = nDef-ym nSkinR	enderControlPoint != -1? GetControl	PointEntitu( =		00000 0000000 000000 00000 0000000 fffffff	0 411707f8		
100 % • 4	Entity ≞nSkinBenderOhiect <u>= (</u> = nDef-y= nSkinB ∭	enderControlPoint 1= -12 GetControl	PointFotitu(	0x7005E1C0 000		0 411707f8		
100 %	Entity ≛nSkinRenderObject <u>– ( – nDef-sm nSkin</u> R ∭	enderControlPoint_1=12_GetControl	- a Loc	0x7005E1C0 000		0 411707f8	Value	
Call Stack	<i>"</i>		रूम् Loc Langua N	0x7005E1C0 000 als ame		0 411707f8	Value	
100 % ▼ Call Stack Name → client_360.pelCNewPa	Entity_InStinBenderObject(nObf_isssSin 	bleInstance_t & instance={}) line 634 + 0x38 b	रूम् Loc Langua N	0x7005E1C0 000		0 411707f8	Value 0x00000000 (m_Refe	rences={} m_pl
100 % - < Call Stack Name → Client_360.pelCNevPa client_360.pelC_Portal, client_360.pelC_Portal,	" diddffett Dra Modelini flaga Crossoco cont Pender Player:RenderScreenSpacePaintffet(DentScreenSpaceff Payer:RenderCasterenSpaceff	bleInstance_t & instance=[]) line 634 - 0:38 b pRenderContext=0xfffffff) line 3794 ect effect=0xfffffff, MarRenderContext * pRen	↓      ↓	als ame () this		0 411707f8		rences={} m_pl
100 % - < Call Stack Name Client_360.pet(CNewPa client_360.pet(C.Portal client_360.pet(C.Portal	" Plyer:FanderScreenSpacePaintEffect(MatRenderContext * Plyer:FanderScreenSpacePaintEffect(PortalScreenSpaceEffect)PortalScreenSpaceEffectPortalScreenSpaceEffec	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	tangua     tangua     tangua     toytes C++     C++     derCo C++     ool bf C++	als ame this flags		0 411707f8	0x00000000 {m_Refe 0x00000000	rences={} m_pl
100 % - < Call Stack Name Client_360.petCNewPa client_360.petC.portal client_360.petC.portal client_360.petCViewRe	" Interffer:: Dras Mudeliof Bugs Dr0000000, cont Render Player:RenderScreenSpacePaintEffect[MatRenderContext Payer:RenderLocalScreenSpaceEffect[PotalScreenSpaceEff PotBrocessing[int subdittilli, int yubdittilli, int yubdittilli inder:RenderView(sort) Vieworl_1, cont OV	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	← 0     ← 0	als ame this flags instance	00000 0000000 ffffff	0 411707f8	0x00000000 {m_Refe 0x000000000 {m_nAlpha=??? }	rences={} m_pl
100 % - < Call Stack Name Client_360.peiCNewDa client_360.peiC_Portal client_360.peiC.portal client_360.peiCViewRe client_360.peiCViewRe	" Player-Brodie-ScreenSpacePantEffect()MatRenderContext * Player-Brodie-ScreenSpacePantEffect()MatRenderContext * Player-Brodie-ScreenSpaceEffect(PortalScreenSpaceEff edetThrocessign(in: text)Mittill, int yod/Mittill, int wod/Mittill, int yod/Mittill, int yod/M	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b		als ame this flags	00000 0000000 ffffff	0 411707f8	0x00000000 {m_Refe 0x00000000	rences={} m_pl
100 % - + Call Stack Name • client_360 pelC/Portal client_360 pelC/Portal client_360 pelC/Serve client_360 p	W     Construction of the second	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b		als ame this flags flags pRenderC	ooooo ooooooo fffffff	0 411707f8	0x00000000 {m_Refe 0x000000000 {m_nAlpha=??? } {}	
100 % - 4 Call Stack Name Cient, 360 pelCNen02 cient, 360 pelCOrpsin cient, 360 pelCOrpsin cient, 360 pelCVewRe engine, 360 pelCVewRe engine, 360 pelCVewRe	" Player-RenderScreenSpacePaintEffect()MatRenderContext * Player-RenderScreenSpacePaintEffect()MatRenderContext * Player-RenderLocaStreenSpaceEffect(PortaScreen	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b		ane ane this flags (	ooooo ooooooo fiffiff oontext a	0 411707f8	0x00000000 {m_Refe 0x00000000 {m_nAlpha=??? } {} {x=-5.11726756e-036	5 y=-1.48138026e
100 % - < Call Stack Name Providence, 260 performer diener, 360 performer diener, 360 performer client, 360 performer client, 360 performer engine, 360	W     Construction of the second	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b		arrestation of the second sec	ooooo ooooooo fiffiff oontext a	0 411707f8	0x00000000 {m_Refe 0x00000000 {m_nAlpha=??? } {} {x=-5.11726756e-036 {x=0.00000000 y=0.	5 y=-1.48138026e .00000000 z=-1.4
100 % - < Call Stack Name Cient, 360 pelC Portal cient, 360 pelC Portal cient, 360 pelC Portal cient, 360 pelC ViewRe cient, 360 pelC ViewRe c	The second	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b		ane ane this flags (	ooooo ooooooo fiffiff oontext a	0 411707f8	0x00000000 {m_Refe 0x00000000 {m_nAlpha=??? } {} {x=-5.11726756e-036	5 y=-1.48138026e .00000000 z=-1.4
100% - 4 Cal Stack Name client, 360 peC, Portal client, 360 peC, Po	These Draw Multicle Engravity and the second and a second a se	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	Logue         Loc           Langue         N           objete         C++           derCa         C+-           c+-         C+-           C++         C+-	anne anne anne anne anne anne anne ann	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_nAlpha=??? } {} {x=-5.11726756e-036 {x=0.00000000 y=0. {x=-5.07289810e-036	5 y=-1.48138026e .00000000 z=-1.4 5 y=1.6816e-043#
100 % - 4 Call Stack Call St	Physics Book Model (1999) And Parket Model (1999)     Physics Book (1999) Consensity of the Charles of Approximation of Approximatio	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	• 0         • 0           Langua	orressize eee als ame     this     flags     flags     pRenderC     vecCamer     vecCiffuse     mins     maxs	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_Alpha=??? } {} {x=5.11726756e-030 {x=0.00000000 y=0 {x=5.07289810e-030 {x=0.00000000 y=-5	5 y=-1.48138026e .00000000 z=-1.4 5 y=1.6816e-043# 5.28676575e-036 z
100% - 4 Cal Stack Name client, 360 petC, Portal client, 360 petC, Portal engine, 380 petChards engine, 380 petChards engine, 380 petChards engine, 380 petChards engine, 380 petChards engine, 380 petChards engine, 380 petChards	The Advance of the second	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	• • • •         Loc           Langua         N           ysts         C++           C++         •           derCa         •           derCa         •           derCa         •           derCa         •           G++         •           G++         •           G++         •           C++         •	orressize eee als ame     this     flags     flags     pRenderC     vecCamer     vecCiffuse     mins     maxs	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_nAlpha=??? } {} {x=-5.11726756e-036 {x=0.00000000 y=0. {x=-5.07289810e-036	5 y=-1.48138026e .00000000 z=-1.4 5 y=1.6816e-043# 5.28676575e-036 z
100 %	"     "	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	• • •         Loc           Langua         N           ytts         C+-           C+-         •           deCc         •           0dff         C+-           0dff         •           0dff         •           C+-         •	orressize	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_NAlpha=??? } {} {x=0.00000000 y=0. {x=5.07289810e-036 {x=0.00000000 y=-5 {x=0.00000000 y=1.	5 y=-1.48138026e .00000000 z=-1.4 5 y=1.6816e-043# 5.28676575e-036 z .65739782e+029 z
100 %	The set of the se	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	+ 0         Longus         N           Langus         N         N           Syste         C         E           C         O         C           Odd         C         E           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C           C         E         C	orressize	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_Alpha=??? } {} {x=5.11726756e-030 {x=0.00000000 y=0 {x=5.07289810e-030 {x=0.00000000 y=-5	5 y=-1.48138026e 00000000 z=-1.3 5 y=1.6816e-043# 5.28676575e-036 z .65739782e+029 z
100 %	"     "	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	• • •         Locu           Langua         N           Yits         C++         •           C++         •         •           deCo         •         •           0:0ff         C++         •           0:0ff         C++         •           C++         •         •	orressize	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_NAlpha=??? } {} {x=0.00000000 y=0. {x=5.07289810e-036 {x=0.00000000 y=-5 {x=0.00000000 y=1.	5 y=-1.48138026e 00000000 z=-1.3 5 y=1.6816e-043# 5.28676575e-036 z .65739782e+029 z
100 %	The second	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	• • •         Locus           Langus         N           ytts         C           C         E           C         O           deTCa         C           Odfff         C           C         E	orressize	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_NAlpha=??? } {} {x=0.00000000 y=0. {x=5.07289810e-036 {x=0.00000000 y=-5 {x=0.00000000 y=1.	5 y=-1.48138026e .00000000 z=-1.4 5 y=1.6816e-043# 5.28676575e-036 z .65739782e+029 z
100 %	The set of the se	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	• • •         Locu           Langua         N           Yits         C++         •           C++         •         •           deCo         •         •           0:0ff         C++         •           0:0ff         C++         •           C++         •         •	orressize	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_NAlpha=??? } {} {x=0.00000000 y=0. {x=5.07289810e-036 {x=0.00000000 y=-5 {x=0.00000000 y=1.	5 y=-1.48138026e .00000000 z=-1.4 5 y=1.6816e-043# 5.28676575e-036 z .65739782e+029 z
100 %	The second	blehnstance, t & instances j)) Sine 634 - 0-33 b pRenderContext=0xfffffff, Jine 3794 ect effect=0xfffffff, JinderRenderContext * pRen fi, int houfffffff, Bool PflashightSOnstruc, b	• 0 Con- Langue ) N Con- Co	orressize	ooooo ooooooo fiffiff oontext a	0 411707f8	0x0000000 {m_Refe 0x00000000 {m_NAlpha=??? } {} {x=0.00000000 y=0. {x=5.07289810e-036 {x=0.00000000 y=-5 {x=0.00000000 y=1.	5 y=-1.48138026e .00000000 z=-1.4 5 y=1.6816e-043# 5.28676575e-036 z .65739782e+029 z

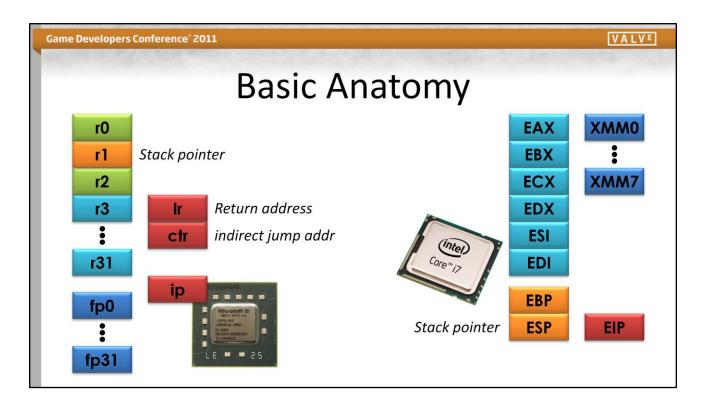
We've crashed somewhere in that block, but where exactly? There's a bunch of places that could go wrong. And, because we've built in release, opening the Locals window doesn't seem to have anything useful for us. So, when the going gets tough, the tough go to the disassembly.

viewpostproce	ss.cpp 🗈 c_portal_player.cpp 🖴 particles_new.cpp 🛍 Disassembly 🗙 client_360.dll.dmp	•	Watch 1	Find Results
Address: CN	ewParticleEffect::DrawModel(int, const RenderableInstance_t &)	-	Address: 0x	7005E050
Viewing O	ptions		0x7005E050	
888E2E20	mr r31.r3		0x7005E060	fffffff
	f ( !pPlayer    !pPlayer->IsAlive()    !pPlayer->GetRenderedWeaponModel()->IsViewModel() )		0x7005E070	
	cmplwi cr6,r3,0		0x7005E080	
	beg cr6,CNewParticleEffect::DrawModel + 0510h (888e3270h)		0x7005E090	
888E2F2C	lwz r11,0(r3)		0x7005E0A0	
888E2F30	lwz r10,240h(r11)		0x7005E0B0	
888E2F34	mtctr r10		0x7005E0C0	
888E2F38	bctrl		0x7005E0D0	
888E2F3C	clrlwi r9,r3,24		0x7005E0E0	
888E2F40	cmplwi cr6,r9,0	E	0x7005E0F0	
	beq cr6,CNewParticleEffect::DrawModel + 0510h (888e3270h)		0x7005E100 0x7005E110	
	lwz r11,0(r31)		0x7005E110	
	mr r3,r31		0x7005E120	
	lwz r10,430h(r11)		0x7005E140	
	mtctr r10		0x7005E150	
888E2F58			0x7005E160	
	lwz r9,0(r3)		0x7005E170	
888E2F60			0x7005E180	
	mtctr r8		0x7005E190	
888E2F68			0x7005E1A0	
	clrlwi r7,r3,24		0x7005E1B0	
888E2F70	cmplwi cr6,r7,0		0x7005E1C0	0000000
00050574	<pre>return 0; beg cr6.CNewParticleEffect::DrawModel + 0510h (888e3270h)</pre>		0x7005E1D0	
000027/4		٣	0x7005E1E0	0000000
4	m	F	0x7005E1E0	0000000

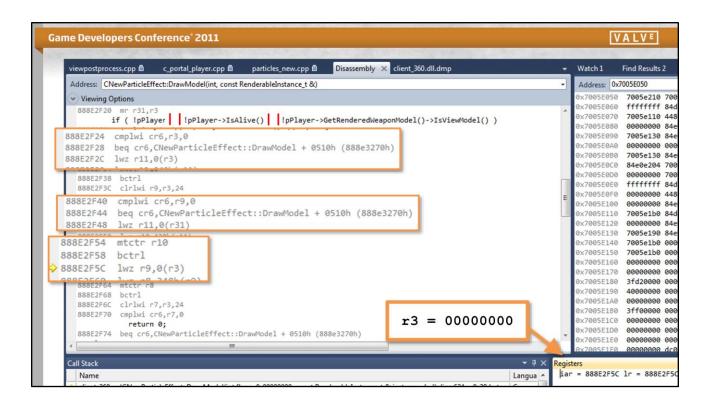
But, as you can see, the disassembly window doesn't always have nice variable names. So to understand the disassembly, you need to understand the CPU's registers. Let's take a quick review.

ne Developers C	onference <sup>®</sup> 2013			VALV
		Desis Au		
		Basic An	atomv	
	SUPERVISOR	MODEL - OEA		
USER MODEL				
UISA	Configuration	CONTRACTOR AND	General-Purpose Registers	0
General-Purpose Registers		Processor Version Register 1 (Read Only)	31	
GPR0 (64/32)	MSR (64/32)	PVR (32) SPR 287		EAX
GPR1 (64/32)	Memory Managen	ent Registers		EBX
	Instruction BAT Registers	Data BAT Registers		
• • • • • • • • • • • • • • • • • • •	IBATOU (64/32) SPR 528	DBATOU SPR 536		ECX
GPR31 (64/32)	IBAT0L (64/32) SPR 529	DBATOL SPR 537		EDX
Floating-Point Registers	IBAT1U (64/32) SPR 530	DBAT1U SPR 538		
FPR0 (64)	IBAT1L (64/32) SPR 531	DBAT1L SPR 539		ESI
FPR1 (64)	IBAT2U (64/32) SPR 532	DBAT2U SPR 540		CD1
	IBAT2L (64/32) SPR 533	DBAT2L SPR 541		EDI
: .	IBAT3U (64/32) SPR 534	DBAT3U SPR 542		EBP
FPR31 (64)	IBAT3L (64/32) SPR 535	DBAT3L SPR 543		
11101(04)		Segment Registers <sup>1, 2</sup>	~	ESP
Condition Register <sup>1</sup>	SDR1	SR0 (32)		
CR (32)	SDR1 (64/32) SPR 25	SR1 (32)	Segment Registr	ers
	Address Space Register <sup>3</sup>		15	0
Floating-Point Status	ASR (64) SPR 280		(intel)	
and Control Register 1		SR15 (32) Microsoft (D)		CS
FPSCR (32)	Exception Handl			DS
XER Register 1	Data Address Register	USISK . D-DBI	LOLD W:	
	DAR (64/32) SPR 19	o reveno		SS
XER (32) SPR 1	SPRGs	Save and Restore Registers		ES
Link Register	SPRG0 (64/32) SPR 272	SRR0 (64/32) SPR 26		
LR (64/32) SPR 8	SPRG1 (64/32) SPR 273	SRR1 (64/32) SPR 27 L E 📟 📟 2 S		FS
Entonise/ Orno	SPRG2 (64/32) SPR 274	Floating-Point Exception		GS
Count Register	SPRG3 (64/32) SPR 275	Cause Register (Optional)		us
CTR (64/32) SPR 9		FPECR SPR 1022		
	Miscellaneous		Program Status and Control Registe	r
	Time Base Facility <sup>1</sup> (For Writing)	Data Address Breakpoint Register (Optional)	31	0
USER MODEL VEA	TBL (32) SPR 284		<u>,</u>	
	TBU (32) SPR 284	DABR (64/32) SPR 1013		EFLAGS
Time Base Facility <sup>1</sup> (For Reading)	Decrementer <sup>1</sup>	External Access Register		
	DEC (32) SPB 22	(Optional) 1	31 Instruction Pointer	0
	a and they are the	EAR (32) SPR 282	21	
TBU (32) TBR 269	Processor Identification Register (Optional)			EIP
	PIR SPR 1023		·	

The two major CPUs game developers deal with are the POWER architecture and the x86. Each contains POWER has a whole bunch of register, of which only those in the left column really matter: the 32 general purpose registers, which store integers and addresses, 32 floating point registers, and the two jump target registers CTR and LR. On the x86, you have eight general purpose registers, of which EBP and ESP are usually reserved for use in managing the stack.



Of the PowerPC's 32 general purpose registers, some have special meanings to the OS, and r1 is almost always reserved for use as the stack pointer. All the stuff I say about the PPC in this talk, by the way, comes from IBM's documentation for their own ABI – the game consoles have slightly different implementations, whose details you can get from your platform documentation, but the operation is similar or analogous.



We can return now to the disassembly window and try to get a better idea of where the game died. Notice that there are two OR operators in that conditional expression. That means the corresponding assembly ought to have two corresponding branch opcodes (because of C++'s early-out shortcut semantics). You can see that the instruction which actually crashed came after both branches, so the crash occurred somewhere in pPlayer->GetRenderedWeaponModel()->IsViewModel(). In particular, the op that died was trying to load a word from the address in register 3, but r3 contained NULL. Why? Let's see what it means for one function to pass data to another in C++.

VALVE

# int foo( int a, int f, int \*pb )

<pre>int foo( int a, int f, int *pb ) {     return a + f + *pb; } void caller() {</pre>	stdu ; funct extsw	tion does its w r3,r3	; move stack for locals work ; return value is on r3 ; restore stack pointer
<pre>int c = 2; printf( "%d\n", foo( globalint, 1, &amp;c ); ); }</pre>	li addic variable bl	r4,1 r5,r1,0x70	at, int*)
РРС			

On the PowerPC, parameters are passed from left to right, on registers r3 through r10. Params too big to go on registers go on the stack.

#### VALVE

# Float CThing::foo( int a, float f, int &pb )

<pre>struct CThing {     int m_n;     CThing( int a ) : m_n( a ) {};     float foo( float f, int &amp;pb )     {         return m_n + pb + f;      } }; void Caller( int a, int c ) {</pre>	CThing::foo:         stdu       r1,-0x40(r1)         lwz       r3,0x0(r3) ; load this->m_n         lwz       r4,0x0(r5) ; load through pb ref         addc       r3,r3,r4 ; m_n + pb         std       r3,0x30(r1) ; move data from an int register         lfd       f2,0x30(r1) ;to a float register via stack         fcfid       f2,f2 ; converts int to float         fadds       f1,f2,f1 ; ( + f )         addi       r1,r1,0x40         blr       ; return	
<pre>int b = c; CThing woo(a); float f = woo.foo( 3.14f, b ); Msg("%f\n", f); } PPC</pre>	Caller:         lis       r4,0x0         stw       r3,0x70(r1)         addic       r5,r1,0x70         addi       r4,r4,0x0         stw       r30,0x74(r1)         addic       r3,r1,0x74         lfs       f1,0x0(r4)         bl       CThing::foo(float, int&)         ; return       value comes back on f1	

C++ member functions have an invisible first parameter "this", which is a pointer to the class instance. Thus, in a C++ function, r3 will always have THIS, and the first formal parameter goes on r4.

#### VALVE

int foo( ir	nt a, int f, int *pb )
<pre>int foo( int a, int f, int *pb ) {    return a + f + *pb; } void caller() {    int c = 2;    printf( "%d\n",       foo( globalint, 1, &amp;c ); ); }</pre>	<pre>foo(int, int, int*): push ebp mov ebp, esp mov eax, DWORD PTR _pb\$[ebp] mov edx, DWORD PTR [eax] add edx, DWORD PTR [ecx] mov DWORD PTR tv134[ebp], edx fild DWORD PTR tv134[ebp] fadd DWORD PTR _f\$[ebp] pop ebp ret 8</pre>
x86	<pre>caller: ;; push params right to left lea eax, DWORD PTR _c\$[ebp] ;; get address to c push eax ; push &amp;c onto stack push l ; push immediate l onto stack push edi ; push globalint onto stack call ?foo@@YAHHHPAH@Z ; foo ;; return value comes back on eax</pre>

On the x86, there's many different calling conventions, but in most, parameters are pushed onto the stack in right to left order.

Msg("%f\n", f);

-)

x86

VALVE

#### Float CThing::foo( int a, float f, int &pb ) CThing::foo: struct CThing ebp push { mov ebp, esp int m\_n; eax, DWORD PTR \_pb\$[ebp] CThing(int a) : m n( a ) {}; mov float foo( float f, int &pb ) edx, DWORD PTR [eax] mov add edx, DWORD PTR [ecx] return m n + pb + f; mov DWORD PTR tv134[ebp], edx DWORD PTR tv134[ebp] fild }; fadd DWORD PTR f\$[ebp] pop ebp void Caller( int a, int c ) ret 8 { int b = c;Caller: CThing woo(a); fld DWORD PTR real@4048f5c3 ; 3.14 const float f = woo.foo( 3.14f, b);

eax

ecx

DWORD PTR b\$[ebp], eax eax, DWORD PTR \_b\$[ebp]

DWORD PTR woo\$[ebp], edi

?foo@CThing@@QAEMMAAH@Z ; return value comes back on x87 fpu stack

DWORD PTR [esp]

ecx, DWORD PTR \_woo\$[ebp] ;; this pointer

mov

lea push

push

lea

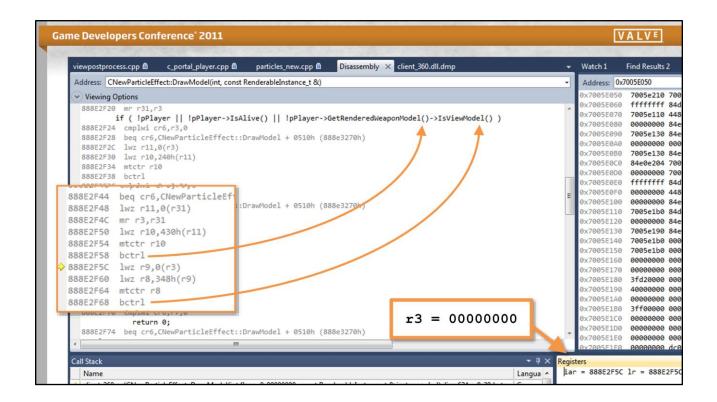
fstp mov

call

There's a few common x86 conventions for dealing with C++ member functions. The more efficient one, THISCALL, passes "this" on the ecx register. Others simply pass it as an implicit first parameter on the stack.

pers Conference <sup>®</sup> 2011			<u>v</u>
Case St	udv #	0	
viewpostprocess.cpp @ c_portal_player.cpp @ particles_new.cpp @ X Disassembly client_360.dll.dmp	Watch 1 Find Results 2     Address: 0x70056050		-
<pre>United end of the set of the</pre>	istance * 0.0005100 FIFFFFF 44.0 0.0005000 000000 000 0.0005000 000000 000 0.0005000 000000 000 0.0005500 000000 000 0.0005500 000000 000 0.0005500 000000 000 0.0005500 000000 000 0.0005500 000000 000 0.0005500 000000 000 0.0005510 000000 0000 0.0005510 000000 00000 000 0.0005510 000000 00000000000000000000000	ella debene 705:11. ella debene 705:11. debene 2012 ella debene	
Call Stack	+ 4 × Locals Langua + Name	Value	- 9 × Type ^
Clent_Bi0.petCNexParticleEffect::DrawModelint flages0x0000000, const RenderableInstance,t & instance.[i] Ine 634 + 0x38 bytes client_360.petC_Portal_Player:RenderScreenSpacePaintEffect(JMatRenderContext * pRenderContext.* 0xfffffff) Ine 3794	C++ (i)	0x00000000 (m_References=() m_pNext=?? 0x0000000	CNewParticleEffect * const int
	C++ 🛛 🕑 pRenderContext		Vector4D
engine_360.petSCR_UpdateScreen() line_283 engine_360.pet_Most_RunFrame_Render() line_2939 engine_360.pet_Most_RunFrame(Float times 0.00000000) line_3880	C++       @ maxs C++       @ center C++    @ vecPos C++	(x=0.00000000 y=-5.28676575e-036 z=0.000 (x=0.00000000 y=1.65739782e-029 z=-1.#Q (x=0.000000000 y=1.65740084e+029 z=0.000)	Vector Vector
engine, 366 getCheinsteine Stand Aurolfteint Ennen 200000000 (ine 537 engine, 366 getCheinsteine Stand Ausoleffent für Eines 20000000 (ine 732 engine, 366 getChippeler Standarssfort für Eines 2000000 (ine 732 engine, 366 getChippeler Standarssfort für Eines 20000) (ine 732 engine, 366 getChippeler Standarssfort für Eines 20000) engine, 356 getChippeler Standarssfort für Eines 2000 engine, 356 getChippeler Standarssfort für Eines 2000 engine 350 getChippeler Standarssfort für Eines 2000 enginter 350 getChippeler Standarssfort für Eines 2000 engine 350	Co+ Co+ Co+ Co+ Co+ Co+ Co+ Co+ Co+		
launcher_360.pelCSteamApplication::Main() line 210	C++ v C++ v B Registers III Locals		-

So, armed with this knowledge, we can return to our case study.



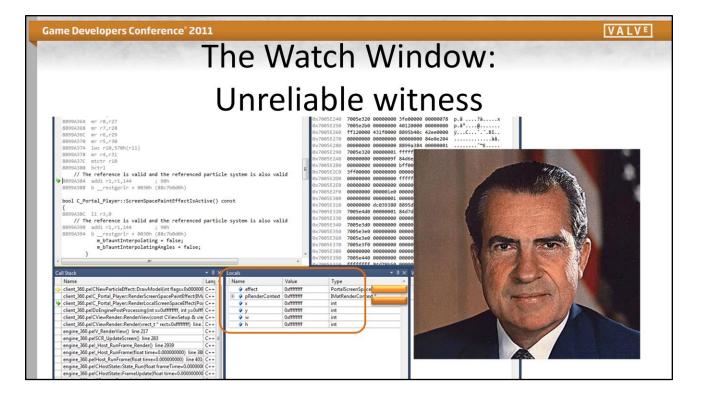
In this case, we've gone past the two conditionals, and died between the bctrls. We know that the first bctrl returned 0x00, and that its return value was to become THIS to the second, so we can look up at the two virtual function calls in the third clause of the if(), and see that what must have happened is that GetRenderedWeaponModel() returned NULL.

# WHY THE DEBUGGER LIES TO YOU

And why forensics is better than relying on eight-bit stoolpigeons

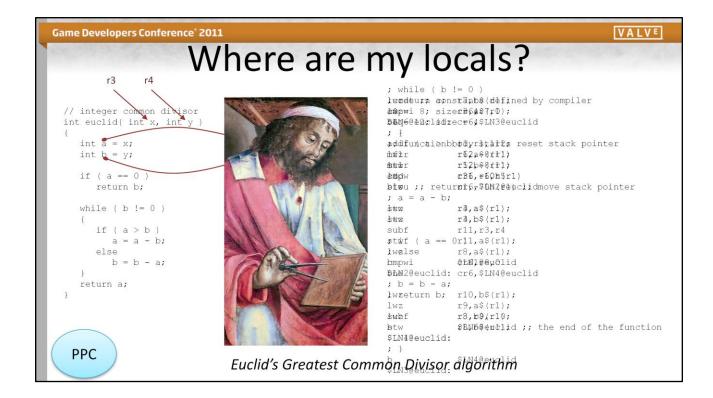
Why the debugger is a filthy lying rat.





As you've noticed, the watch window can be what's technically referred to as a BIG FAT LIAR. To see the reason for this, we can look at what the optimizing compiler does to a function when you compile it in release mode.

Pictured: the "0xfffffffff" lie for an unknown local



Here's the simplest function I could think of, Euclid's Greatest Common Divisor algorithm. Based on the code you would expect to find room for two locals on the stack, and indeed that's what we've got. Looking at the code the "debug" compiler produced, we see pretty much the prolog code we expect: it makes room on the stack and moves the input parameters from r3 and r4 to the locals A and B in memory. When it does the math, you see that it loads A and B from memory into a register before each operation, does the op, and then stores them back. Finally, it retrieves the result from memory, puts it onto r3 for return, and restores the stack.

The compiler does this in "debug" builds in order to preserve a 1:1 mapping between the source code and the machine language. The assembly performs work in the same order as the source, and for every line of source code you can find the corresponding machine op, and vice versa. Also, the debugger can always find local variables in memory, because they are always stored back after being modified, and always to consistent locations.

You might expect that all this round trip traffic to main RAM might be less than performant, and you'd be right. Let's look at that same function compiled in Release.

Game Developers Conference <sup>®</sup> 2011				VALVE
r3 r4 In relea	ase:	; { mr		
<pre>// integer common divisor int euclid( int x, int y) Ox14</pre>	caller frame		cr6, \$LN8@euclid	
Ox10	backchain	mr		
int a = x;	int a	blr; return		
int b = y;	int b	\$LN8@euclid:		
0x04	LR save	; while ( b cmpwi		
if ( a == 0 )			cr6, \$LN3@euclid	
return b; +0x00	+0x10		; Start of loop	
<pre>while ( b != 0 ) {     if ( a &gt; b )         a = a - b;     else         b = b - a; } return a; } </pre>		<pre>subf ; else b \$LN2@euclid: subf \$LN1@euclid: cmpwi bne \$LN3@euclid:</pre>	<pre>cr6,r10,r11 cr6,\$LM2@euclid r10,r11,r10 ; a = a - b; \$LN1@euclid r11,r10,r11 ; b = b - a; cr6,r11,0 cr6,\$LL4@euclid r3,r10 a</pre>	

As you can see, now there is no stack! The function doesn't even have to move the stack pointer since it calls no sub functions, and more importantly, it never stores its locals in memory. They are always in the registers. So, the debugger's watch may not know where to look for these numbers. Also, generally, the compiler can do many kinds of optimization that may cause some intermediate values to vanish altogether. It can order machine code differently from the source, collapse common calculation, all sorts of things that destroy the 1:1 mapping between source and assembly. That's why the instruction arrow jumps around a lot when stepping through release code, and why the debugger has such a hard time finding local data: the data is never in memory in the first place, but stored in registers, or possibly never even stored at all.

#### VALVE

# Who can you trust?

**Unreliable witness?** 

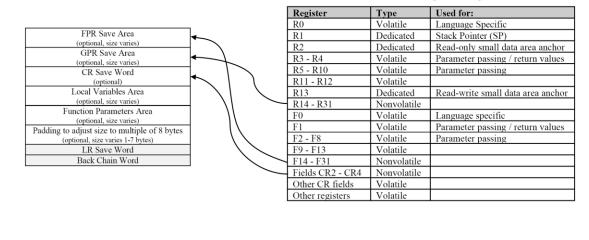
**Try SCIENCE!** 



The watch window has to interpret the compiler results it sees to give you data, and when its interpretations fail, it's an unreliable witness. As any CSI knows, if you've got an unreliable witness, the best place to turn is the physical evidence.

#### VALVE





#### PowerPC EABI register usage

The notion of volatile and nonvolatile registers on the PPC is very useful in debugging analysis. Because a function must restore nonvolatile regs to their initial state before returning to its caller, that means that it must always save them to the stack before modifying them. That means that nonvolatile regs are almost always recoverable so long as you still have the stack. Volatile registers can easily be overwritten.

			Hourse whomese
Come Dec E	// do distance cull check here. We do it here instead of in	particles so we	👻 0x7005E050 7005e210 7005e1b4 00000000 7005e1ac
Game De	// it for root objects, not bothering to cull children indi		Ox7005E060 ffffffff 84d8ed70 0000000 84e0dfc8     VALV <sup>2</sup>
	CMatRenderContextPtr pRenderContext( materials );		0x7005E070 7005e110 448eb6bc 00000000 84e0e3f4
ALC: NOT THE OWNER OF	if ( !m_pDef->IsScreenSpaceEffect() && !m_pDef->IsViewModel	Effect() )	8x7005E080 00000000 84e0e3fc 84d7d9d0 84e0e3f4
	{		0x7005E890 7005e130 84e0e3fc 00000000 dc0392b0
	Vector vecCamera;		0x7005E8A0 00000000 00000000 84d9b3b4 dc0392b0
	pRenderContext->GetWorldSpaceCameraPosition( &vecCamera		0x7005E080 7005e130 84e0e204 84d9b260 00000000 0x7005E0C0 84e0e204 7005e247 00000000 7005e23f
	<pre>if ( ( CalcSqrDistanceToAABB( m_MinBounds, m_MaxBounds,</pre>	vecCamera ) > (	0x7005E0C0 04406204 70052247 00000000 7005231 0x7005E0D0 00000000 7005e237 00000000 7005e23f
	return 0;		0x7005E0E0 fffffff 84d79b30 00000000 84e0dfc8
	}		0x7005E0F0 00000000 448eb708 00000000 84e0e3f4
	if ( = -Def +Telle Hele (Cffeet() )		0x7005E100 00000000 84e0e3fc 84d7c6f8 00000078
	<pre>if ( m_pDef-&gt;IsViewModelEffect() ) </pre>		0x7005E110 7005e1b0 84d42df4 00000000 dc0392b0
	<pre>i C BasePlayer* pPlayer = C BasePlayer::GetLocalPlayer();</pre>		0x7005E120 00000000 84e0e204 84d946b4 84e0e244
	if ( !pPlayer    !pPlayer->IsAlive()    !pPlayer->GetRe		1 0x7005E130 7005e190 84e0e204 84d9b420 00000001
· · · · ·	return 0;	nuer cuncupon loues	0x7005E140 7005e1b0 00000280 84d9aa1c dc0392b0
	}		0x7005E150 7005e1b0 00000000 00000000 410de018
			0x7005E160 00000000 00000000 ffffffff 88db88a5
	if ( ( flags & STUDIO_TRANSPARENCY )    !IsBatchable()    !	m pDef->IsDrawnTh	h 0x7005E170 0000000 0000000 0000000 0000000
	{	-	0x7005E180 3fd20000 0000000 40340000 00000000
	<pre>C_BaseEntity *pCameraObject = GetSplitScreenViewPlayer(</pre>	);	0x7005E190 40000000 00000000 404ca5dc 1a63c1f8
	<pre>if( ((C_BasePlayer *)pCameraObject)-&gt;GetViewEntity() )</pre>		8x7005E1A0 00000000 00000000 bff00000 00000000
	{		0x7005E180 3ff00000 00000000 00000000 411707f8 0x7005E100 00000000 00000000 ffffffff 88db88a5
	pCameraObject = ((C BasePlayer *)pCameraObject)->Ge		
	pcameraobject = ((c_baseriayer ))pcameraobject)->de	tViewEntity();	
	}	tViewEntity();	_ 0x7005E1D0 00000000 00000000 0000001e0
100 %	<pre>plameraobject = ((t_baseriayer ')plameraobject)*&gt;ve </pre>	tViewEntity();	
100 % Call Sta	} • {	,	0x7005E1D0 00000000 00000000 0000001±0 0x7005E1E0 00000000 00000280 00000000 00000000
and the second se	} • « [m] sck	,	0x70955EID0 000908060 00000000 00000000 00000000 0x70055EE0 00000000 00000288 00000000 00000000 0x70055EEC 00000000 00000288 00000000 01000550
Call Sta Nar	} • « [m] sck	→ ♯ × Langua ▲	0x700551D0         00000000         00000000         00000000         00000000           0x700551D0         00000000         00000000         00000000         v           0x700551D0         00000000         00000000         41000550         v           Registers         v 0         x         1         x           iar         88852F5C         1r = 0000000087D1980         msr = 02008030         x
Call Sta Nar	}	→ # × Langua ^ Istance_t C++	0x700551D0         00000000         00000000         00000000         00000000           0x700551D0         00000000         00000000         00000000         00000000           0x700551D0         000000000         00000000         41000500         -           Registers         000000000         000000000         41000500         -           re = 88852F5C lr = 88852F5C ctr = 0000000088701980 msr = 02008030         -         -           r0 = 00000000857785CA0 rl = 0000000070055E050 r2 = 000000000000000000000000000000000
Call Sta Nar Clie	*      /      m  stck  me  nt_360,pelCNewParticleEffect::DrawModel(int flags=0x0000000, const RenderableIn	+ ↓ × Langua ^ Istance_t C++ InderCont C++	0:70955EID0         000000000         000000000         000000000         000000000           0:70055EE0         000000000         000000000         000000000         0         0           0:70055EE0         000000000         000000000         000000000         0         0         0           Register         • 0         • 0         • 0         • 0         • 0         • 0           r0 = 0000000055785CA0         r1 = 0000000070055050         r2 = 00000000000000000         • 0         • 0         • 0           r3 = 00000000000000000         r4 = 00000000000000000000000000000000         • 0         • 0         • 0
Call Sta Nar ➡ clie clie	) act3 me mt_380.pelCNewParticleEffect::DrawModel(int flags=0x0000000, const RenderableIn mt_380.pelCNewParticleEffect::DrawModel(int flags=0x0000000, const RenderableIn t_380.pelCPontal_Playes::RenderScreenSpacePaintEffect(IMatRenderContext * pRen	+ A × Langua ^ Istance_t C++ InderCont C++	0x700551D0         000000000         00000000         00000000         00000000         00000000         0           0x700551D0         000000000         000000000         000000000         000000000         +         +           0x700551D0         000000000         000000000         000000000         +         +         +           Registers         -         -         -         +
Call Sta Nar ➡ clie clie clie clie	m      m      m      m      m      m      m      m      m      m      m      m      gelCNewParticleEffect:DrawModel(int flags=0x00000000, const Renderable      m      m     30, pelC Portal_Playe::RenderScreenSpacePaintEffect(MatRenderContext*pRe-     m     130, pelC Portal_Playe::RenderScreenSpacePaintEffect(PortalScreenSpaceFirst)	Langua A stance_t C++ nderCont C++ t h=0xffff C++	0:7/055ELD0         000000000         000000000         000000000           0:7/055ELE0         000000000         00000000         00000000           0:7/055ELE0         000000000         00000000         00000000           0:7/055ELE0         000000000         00000000         00000000           0:7/055ELE0         000000000         00000000         000000000           0:7/055ELE0         00000000000         0000000000000         00000000000000           0:r         88822F5C         tr =         000000000000000000000000000000000000
Call Sta Nar Clie Clie Clie Clie Clie	*	Langua A stance_t C++ nderCont C++ t h=0xffff C++	0x70055E1D0         000000000         00000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         0000000000         0000000000         0000000000         0000000000         000000000000000000000000000000000000
Call Sta Nar Clie Clie Clie Clie Clie Clie		Langua * stance_t C++ derCont C++ effect=0x C++ tup & hu C++	0:70955EID0         00000000         00000000         00000000         00000000           0:70055EIE0         00000000         00000000         00000000         0         0           Register         •         •         •         •         •         •         •           iar = 88822F5C 1r •         88822F5C ctr •         0000000080701950 msr = 0200000         •         •         •         •           r0 = 00000000055785CA0 r1 =         000000007055550 r2 =         00000000000000         •         •         •         •           r3 = 000000000000000 r4 =         00000000000000 r5 =         000000000000000000000000000000000000
Call Stat Nar ♀ clie clie clie clie clie clie e eng	m      m      m      m      m      m      m      m      m      m      d0.pelCNewParticleEffect::DrawModel(int flags=0x0000000, const RenderableIn      m     d0.pelCPortal_Plays::RenderScreenSpacePaintEffect(IMatRenderContex**pRen      m     d0.pelC_Portal_Plays::RenderColosScreenSpaceEffect      rents.000.pelDoEnginePostProcessing(int x=0dfffffff, int y=0dfffffff, int y=0dffffffff, int y=0dfffffff, int y=0dfffffff, int y=0dfffffff, int y=0dffffffff, int y=0dfffffffff, int y=0dfffffffff, int y=0dffffffff, int y=0dffffffff, int y=0dffffffff, int y=0dffffffff, int y=0dfffffffff, int y=0dffffffffffffff, int y=0dffffffffffffffffffffffffffffffffffff	+ 4 × Langua * Istance_t C++ InderCont C++ Iffect=0x C++ th=0xfff C++ tp & hu C++ C++	0:70955EID0         00000000         00000000         00000000         00000000           0:70055EIE0         000000000         00000000         00000000         0         0           Register
Call Str Nar clie clie clie clie clie clie eng eng		+ Q × Langua * stance_t C++ iffect=0xf C++ t h=0xfff C++ t h=0xfff C++ t b=0xfff C++ C++ C++	0×70955LD0         000000000         000000000         000000000         000000000           0×70055LD0         000000000         000000000         000000000         0         0           20000000         000000000         000000000         000000000         000000000         0         0         0           2000000000000000000         r0         000000000000000000000000000000000000
Call Str. Nar clie clie clie clie clie clie clie eng eng eng			0:7005ELD0         00000000         00000000         00000100           0:7005ELD0         00000000         0000000         00000000           0:7005ELED         000000000         4/01010         00000000           0:7005ELED         000000000         4/01010         00000000           0:7005ELED         000000000         4/01010         00000000         0           0:00000000000000         4/01010         00000000000000         5         000000000000000000000000000000000000
Call Str. Nar ∳ clie clie clie clie clie eq eng eng eng eng eng		+ 1 × Langua * istance_t C++ derCont C++ t h=0xfff C++ t h=0xfff C++ t h=0xfff C++ t C++ C++ C++ C++	0+70955EID0         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         0000000000         0000000000         0000000000         000000000000000000000000000000000000
Call Str Nar ¢ clie clie clie clie clie clie eng eng eng eng eng eng eng eng			0:70055E100         000000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         0000000000000         000000000000000000000000000000000000
Call Str. Nar ∳ chie chie chie chie chie chie chie eng eng eng eng eng eng eng eng	m      m	-         3 ×           Langua •         -           stance_t C++         -           ffectod/C++         -           th=0.6Ht         C++           C++         -           C++         -           C++         -           C++         -           C++         -           C++         -           C++         -	0:70955EID0         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         0<
Call Str Nar ¢ clie clie clie clie clie clie clie clie	(	→ 3.× Langus → stance,1 C++ defCont C++ the0offf C++ C++ C++ C++ C++ C++ C++ C++	0:70955EID0         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         000000000         0<
Call Str Nar		→ 3 × Langua * stance_t C++ ffect=0d C++ th=0.0ff C++ C++ C++ C++ C++ C++ C++ C++	0:70955EID0         000000000         00000000         00000000         00000000         000000000         000000000         000000000         000000000         0
Call Str. Nar ∳ clie clie clie clie clie eng eng eng eng eng eng eng en	) sck me mt. 360.pelCNewParticleEffect:DrawModel(int flags=0x0000000, const Renderabledn ent, 360.pelC, Portal_Player:RenderScreenSpacePaintEffect(I)MatRenderContext * pRer ent, 360.pelC, Portal_Player:RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect ant, 360.pelCNewRender:RenderView(const CVevSetup & viewer_Loc,, const CViewSet int, 360.pelCNewRender:RenderView(const CVevSetup & viewer_Loc,, const CViewSet int, 360.pelCNewRender:RenderView(const CVevSetup & viewer_Loc,, const CViewSet jine_360.pelCNewRender:RenderView(const CVevSetup & viewer_Loc, const CViewSet jine_360.pelCNewRender:RenderView(const CVevSetup & viewer_Loc,, const CViewSet jine_360.pelCNewRender:RenderView(const CVevSetup & viewer_Loc, const jine_360.pelCNEwRenderRenderView(const CVevSetup & viewer_Loc, const jine_360.pelCNEwRenderRenderView(const CVevSetup & viewer_Loc) jine_360.pelCNEwRenderRender(line 2339 jine_360.pelCNExRenFarme(IntelTime=0.000000000) line 5380 jine_360.pelCNExtate:State_State_Render(line time=0.000000000) line 537 jine_360.pelCNextate:State_State_Render(Intel 1) jine_360.pelCRengEnExMericherMetatfameTime=0.000000000) line 712 jine_360.pelCRengEnExMericherMetatfameTimeRender(line time=0.000000000) line 712 jine_360.pelCRengEnExMetate(line time=0.000000000) line 712 jine_360.pelCRengEnExMetate(line time=0.000000000) line 712		0:7005ELD0         000000000         00000000         00000000           0:7005ELD0         000000000         00000000         00000000         0           0:7005ELD0         000000000         00000000         00000000         0         0           1:ar         838E275C         1r         00000000000         000000000000000000000000000000000000
Call Std Nar ¢ clie clie clie clie clie clie clie clie	<pre>} * * * * * * * * * * * * * * * * * * *</pre>		0:70955EID0         000000000         00000000         00000000         00000000         000000000         000000000         000000000         000000000         0

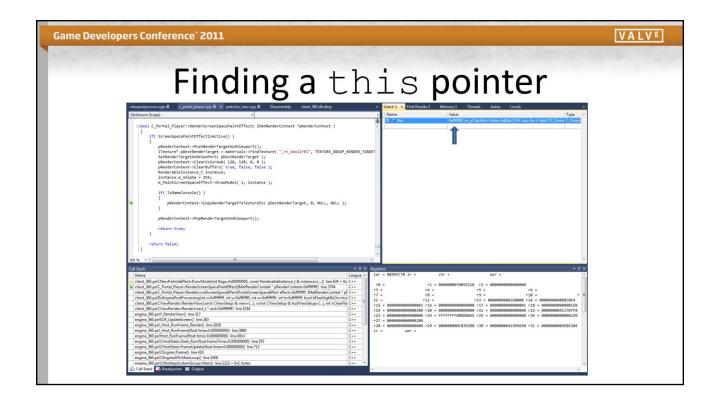
The debugger can show you all the registers in their current state at the bottom frame of the stack, because they were stored in the exception at the moment of crashing..

			Address				
Anna Bar	<pre>m_PaintScreenSpaceEffect-&gt;StopEmission();</pre>		✤ 0x7005E050		1b4 00000000 7005e1ac		A WALVE
Game De	<pre>m_PaintScreenSpaceEffect = NULL;</pre>		* 0x7005E060		d70 00000000 84e0dfc8		VALVE
and the second se	}				6bc 00000000 84e0e3f4		
ALC: NOT THE OWNER OF			0x7005E080		3fc 84d7d9d0 84e0e3f4		
	// commenting out the 3rd person drop effect		0x7005E090		3fc 00000000 dc0392b0		
	/*		0x7005E0A0		000 84d9b3b4 dc0392b0		
	// Remove paint drip effect		0x7005E0B0		204 84d9b260 00000000		
	<pre>if( m_PaintDripEffect.IsValid() )</pre>				247 00000000 7005e23f		
	{		0x7005E0D0		237 00000000 7005e22f		
	<pre>m_PaintDripEffect-&gt;StopEmission();</pre>				b30 0000000 84e0dfc8		
	<pre>m_PaintDripEffect = NULL;</pre>		0x7005E0F0		708 0000000 84e0e3f4		
12					3fc 84d7c6f8 00000078		
*	*   */				df4 00000000 dc0392b0 204 84d946b4 84e0e244		(B)
	13				204 84d94604 84e0e244 204 84d9b420 00000001		
					280 84d9aa1c dc0392b0		
	<pre>_void C Portal Player::ClientPlayerRespawn()</pre>				000 00000000 410de018		
	void C_Portal_Player::ClientPlayerRespawn()		0x7005E160		000 ffffffff 88db88a5		
	if ( IsLocalPlayer( this ) )				00000000 00000000 000		
	T) ( TECCOTLIANCE ( CUTE ) )				000 40340000 00000000		
	<pre>m bGibbed = false;</pre>				000 404ca5dc 1a63c1f8		
	"_bolbbed = folse,		0x7005E1A0	000000 000000	000 bff00000 00000000		
	// Dod called these, not sure why		0x7005E180	3ff00000 00000	000 00000000 411707f8		
	<pre>//MoveToLastReceivedPosition( true );</pre>			000000 0000000	000 ffffffff 88db88a5		
	//ResetLatched():		_ 0x7005E1D0	000000 000000	000 0000000 000001e0		
	<pre>//ResetLatched();</pre>		<ul> <li>0x7005E1D0</li> <li>0x7005E1E0</li> </ul>		000 0000000 000001e0 280 0000000 00000000		
10	//ResetLatched(); 00% - 4	•		0000000 00000			-
		• # ×		0000000 00000	280 0000000 0000000	•	₽×
-	10 % • 4	→ 무 × Langua ^	0x7005E1E0	00000000 00000 00000000 dc030	280 0000000 0000000		
Ca	0 % • 4 II Stack Name	Langua 🔺	Registers iar = 8899FC	00000000 00000 00000000 dr030 78 lr =	280 0000000 00000000 260 0000000 41000650 ctr =	msr =	
Ca	10 %	Langua A	Registers iar = 8899FC r0 =	0000000 00000 0000000 dra30 78 lr = r1 =	280 0000000 0000000 260 0000000 41000550 ctr = 000000007005E210 r2	msr = 0000000000000000	*
Ca •	0 % • 4 III Stack Name client_360.pelCNewParticleEffect::DrawModel[int flags=0.00000000, const RenderableInsta client_360.pelCQ.portal_Playe::RenderScreenSpacePaintEffect[MatRenderContex**pRender	Langua ^ ince_t C++ erCont C++	Registers iar = 8899FC	0000000 00000 0000000 d-000 78 lr = r1 = r4 =	280 0000000 0000000 200 0000000 11000550 ctr = 000000007005E210 r2 r5 =	msr = = 0000000000000000000000000000000000	7 ×
Ca •	0 % • 4 III Stack Name client_360.pe/CNewParticleEffect::DrawModel(int flags=Dx00000000, const Renderableinsta client_360.pe/C_Portal_Playe:::RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect) client_360.pe/C_Portal_Playe:::RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect)	Langua * nce_t C++ erCont C++ ct=0xf C++	Registers iar = 8899FC r0 =	0000000 00000 0000000 dr000 78 lr = r1 = r4 = r7 =	280 0000000 0000000 200 000000 1100050 ctr = 000000007005E210 r2 r5 = r8 =	msr = = 0000000000000000000000000000000000	*
Ca 🌳	0 % • • • III Stack Name Client, 360, pelCNewParticleEffect::DrawModel(int flags::Dx00000000, const RenderableInsta Client, 360, pelC, Portal, Playe::RenderScreenSpaceBartEffect(IDrataScreenSpaceEffect He Client, 360, pelC, Portal, Playe::RenderScreenSpaceEffect(IDrataScreenSpaceEffect He Client, 350, pelC, Portal, Playe::RenderIcaScreenSpaceEffect(IDrataScreenSpaceEffect He Client, 350, pelC, Portal, Playe::RenderIcaScreenSpaceEffect(IDrataScreenSpaceEffect) Client, 350, pelC, Portal, Playe::RenderIcaScreenSpaceEffect(IDrataScreenSpaceEffect) He (InterSpaceEffect) He (InterSpaceE	Langua * ince_t C++ erCont C++ ct=0xf C++ =0xfff C++	Registers iar = 8899FC r0 =	0000000 00000 0000000 dra30 78 lr = r1 = r4 = r7 = r10 =	280 0000000 0000000 200 0000000 41000500 ctr = 000000007005E210 r2 r5 = r8 = r11 =	msr = = 000000000000000 r9 r12 =	*
Ca ••	0 % • 4 III Stack Name client_360.peiCNewParticleEffect::DrawModel(int flags=0x00000000, const RenderableInsta client_360.peiC_Portal_Playe::RenderScreenSpaceBaintEffect(MatRenderContext * pRender client_360.peiC_Portal_Playe::RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect effect client_360.peiCD_EngeborStProcessing(ints-avofffffff, int v=0x0ffffff, int v=0x0ffffff, int v=0x0ffffff, int v=0x0fffffft, int v=0x0fffffft, int v=0x0ffffftf, int v=0x0ffffftf, int v=0x0ffffftf, int v=0x0ffffftf, int v=0x0fffftft, int v=0x0fffftft, int v=0x0fffftft, int v=0x0fffftft, int v=0x0fffftft, int v=0x0ffftft, int v=0x0ffftftft, int v=0x0ffftftft, int v=0x0ffftftft, int v=0x0fftftft, int v=0x0fftftft, int v=0x0fftftft, int v=0x0fftfttt, int v=0x0fftftftt, int v=0x0fftfttt, int v=0x0fftfttt, int v=0x0ffttftt, int v=0x0ffttftt, int v=0x0ffttftt, int v=0x0ffttftt, int v=0x0ffttftt, int v=0x0ffttttt, int v=0x0ffttttt, int v=0x0fftttttt, int v=0x0ffttttt, int v=0x0ffttttt, int v=0x0ffttttt, int v=0x0fftttttttttttttttttttttttttttttttttt	Langua ^ mce_t C++ erCont C++ ct=0xf C++ =0xfff C++ a& hu C++	ex7005E1E0 ex7005E1E0 Registers iar = 8899FC r0 = r3 = =	0000000 00000 0000000 draso 78 lr = r1 = r4 = r10 = r12 = 000000	280 0000000 0000000 200 0000000 41000550 ctr = 0000000070055210 r2 r5 = r8 = r8 = r14 - 00000	msr = = 000000000000000 r9 r12 =	*
Ca ••	0 % • • • • • • • • • • • • • • • • • •	Langua ^ nce_t C++ trCont C++ ct=0xf C++ =0xfff C++ & hu C++ C++	0x7005E1E0 0x7005E1E0 0x7005E1E0 registers iar = 8899FC r0 = r3 = = r15 = 000000	00000000 00000 0000000 dr300 78 lr = r1 = r7 = r10 = r12 = 000000 000000000 r16 =	280 0000000 0000000 ctr = 000000007005210 r2 r5 = r8 = r11 = 0000000000 =11 00000000000000 r17	msr = = 000000000000000 r9 r12 = = 0000000000000000	*
Ca ••	0 % • • • • • • • • • • • • • • • • • •	Langua ^ nce_t C++ rcCont C++ ct=0xf C++ a& hu C++ C++ C++	0x7005E1E0           0x7005E1E0           Registers           iar = 8899FC           r0 =           r3 =           -           r15 = 000000           r18 = 000000	00000000 00000 00000000 dr330 78 lr = r4 = r7 = r10 = r12 = 000000 000000003 r16 = 0000000016 r19 =	280 0000000 0000000 200 000000 41000600 ctr = 000000007005E210 r2 r8 = 00000000001 - 0000 0000000000000000 r17 0000000000000000 r17	= 000000000000000000000000000000000000	*
Ca ••	0 % • • • • • • • • • • • • • • • • • •	Langua ^ nce_t C++ erCont C++ ct=0xff C++ =0xfff C++ - 0x+ C++ C++ C++ C++ E	0x7005E1E0           0x7005E1E0           0x7005E1E0           registers           iar = 8899FC           r0 =           r3 =           -           r15 = 000000           r15 = 000000           r21 = 000000	00000000 00000 00000000 dra30 78 lr = r1 = r10 = r10 = r12 = 000000 00000000150 r19 = 000000000 r22 =	220 0000000 00000000 220 0000000 4100050 ctr = 000000000005210 r2 r5 = r11 = 0000000000000000 r17 000000000000000 r17 0000000000000000 r17	=5r = = 00000000000000000 r9 r12 = = 00000000000001 = 00000000000000	*
Ca ••	0 % * *  III Stack  Name  Liter, 350, pelC, Portal, Player::Render/Screen/SpacePantEffect()MatRenderContext * pRender  Liter, 350, pelC, Portal, Player::RenderLocalScreen/SpacePantEffect()MatRenderContext * pRender  Liter, 350, pelC, Portal, Player::RenderLocalScreen/SpaceFifectHer  Liter, 350, pelC/ViewRenderRenderView()Cont CVIew/Setup & viewr), cont CVIew/Setup  Liters, 350, pelC/ViewRenderRenderViewt, * rect=0/dfffff), int >0/dffffff, int va.0/dffffff, int va.0/dfffffff, int va.0/dfffffff, int va.0/dffffff, int va.0/dfffffff, int va.0/dffffff, int va.0/dfffff, int va.0/dffffff, int va.0/dffffff, int va.0/dfffff, int va.0/dfffff, int va.0/dfffff, int va.0/dffffff, int va.0/dffffff, int va.0/dfffff, int va.0/dffffff,	Langua ∧ nrce_t C++ erCont C++ ct=0xf C++ C++ C++ C++ C++ C++ C++ C++	0x7005E1E0 Gw7005E1E0 Registers iar = 8899FC r0 = r3 = = r15 = 000000 r18 = 000000 r18 = 000000 r21 = 000000 r24 = FFFFFF	00000000 00000 00000000 de000 r78 lr = r7 = r10 = r12 - 000000 000000000 r16 = 000000000 r19 = 000000000 r22 = 000000000 r25 r25 =	280 000000 0000000 200 0000000 11000000 ctr = 000000000000000000 r5 = r8 = r1 = 0000000000000000 r17 000000000000000 r12 00000000000000 r20 00000000000000 r20	msr = = 0000000000000000 r12 = = 000000000000000 = 00000000000000 = 0000000000	*
Ca ••	11 Stack Name Client_360.peiCNewParticleEffect::DrawModel(int flags=0x00000000, const RenderableInsta Client_360.peiC_Portal_Player::RenderScreenSpaceBaintEffect(IDMatRenderContext * pRender Client_380.peiC_Portal_Player::RenderScreenSpaceEffect(PortalScreenSpaceEffect effect Client_380.peiCDEginePostProcessing(int x=0xfffffff, int y=0xfffffff, int x=0xffffffff, int x=0xfffffffff, int x=0xffffffff, int x=0xfffffffff, int x=0xffffffff, int x=0xfffffffff, int x=0xffffffffffffffff, int x=0xffffffffffff, int x=0xfffffffffffffffffffffffffffffffffff	Langua ^ nce_t C++ crCont C++ ct=0xfff C++ C++ C++ C++ C++ C++ C++ C++	0x7005E160 0x7005E160 registers r0 = r3 = - r15 = 0000000 r18 = 0000000 r21 = 0000000 r24 = FFFFFF r27 = 0000000	00000000 00000 0000000 drain r1 = r2 = r2 = r10 = r2 = 000000000 r12 = F88D888A5 r25 = F88D888A5 r28 =	280 0000000 00000000 280 0000000 41000000 ctr = 600000007005E210 r2 r5 = r11 = 00000000000000000000 r20 00000000000000000 r20 00000000000000000 r20	msr = = 000000000000000000000000000000000	*
Ca ••	10 % • • • • • • • • • • • • • • • • • •	Langua ∧ nr.ce_t C++ efCont C++ =0xfff C++ =0xfff C++ C++ C++ C++ C++ C++ C++ C++	0x7005E160 0x7005E160 registers r0 = r3 = - r15 = 0000000 r18 = 0000000 r21 = 0000000 r24 = FFFFFF r27 = 0000000	00000000 00000 0000000 drain r1 = r2 = r2 = r10 = r2 = 000000000 r12 = F88D888A5 r25 = F88D888A5 r28 =	280 000000 0000000 200 0000000 11000000 ctr = 000000000000000000 r5 = r8 = r1 = 0000000000000000 r17 000000000000000 r12 00000000000000 r20 00000000000000 r20	msr = = 000000000000000000000000000000000	*
Ga •	0 % • • • • • • • • • • • • • • • • • •	Langua * nce_t C++ rCont C++ ct=0d C++ 0dff C++ C++ C++ C++ C++ C++ C++ C++	0x7005E160 0x7005E160 registers r0 = r3 = - r15 = 0000000 r18 = 0000000 r21 = 0000000 r24 = FFFFFF r27 = 0000000	00000000 00000 0000000 drain r1 = r2 = r2 = r10 = r2 = 000000000 r12 = F88D888A5 r25 = F88D888A5 r28 =	280 0000000 00000000 280 0000000 41000000 ctr = 600000007005E210 r2 r5 = r11 = 00000000000000000000 r20 00000000000000000 r20 00000000000000000 r20	msr = = 000000000000000000000000000000000	*
Ca ••	0 % • • • • • • • • • • • • • • • • • •	Langua * nce_t C++ cfont C++ ctobd C++ =0xfff C++ & hu C++ C++ C++ C++ C++ C++ C++ C++	0x70055120           cx70055120           r0           r0           r0           r3           -	00000000 00000 0000000 4-010 78 lr = r1 = r10 = r2 = 000000150 r15 = 000000000 r22 = 000000000 r25 r3 = 000000000 r23 = 000000000 r31 =	280 0000000 00000000 200 0000000000000000	EST = = 0000000000000000 r9 r12 = = 0000000000000000 = 00000000000000	*
Ca ••	0 % • < III Stack Name Client, 360, peiC, Portal, Playe:::RenderScreenSpaceBantEffect[MatRenderContext * pRender Client, 360, peiC, Portal, Playe:::RenderScreenSpaceBantEffect[DrataScreenSpaceEffect tfe Client, 360, peiC, Portal, Playe:::RenderView(const CViewSetup & view(), const CViewSetup Client, 360, peiC/ViewRender:RenderView(const CViewSetup & view(), const CViewSetup Client, 360, peiC/NewRender:RenderView(const CVIewSetup & view(), const CViewSetup Client, 360, peiCR, UpdateScreen(), line 217 engine, 360, peiCR, UpdateScreen(), line 238 engine, 360, peiCR, UpdateScreen(), line 239 engine, 360, peiCR, UpdateScreen(), line 239 engine, 360, peiCR, UpdateScreen(), line 239 engine, 360, peiCHost, TuurFarme(Float time=0.000000000), line 3880 engine, 360, peiCHostState:State, Run(float frameTime=0.000000000), line 557 engine, 380, peiCHostState:FrameUpdate(float time=0.000000000), line 712 engine, 380, peiCHostState:FrameUpdate(float time=0.000000000), line 712 engine, 380, peiCHostState:State, Run(float frameTime=0.000000000), line 712 engine, 380, peiChostState:State, Run(float frameState), Run(float frameSta	Langua + nnce_t C++ vecont C++ abdff C++ 8 hu C++ C++ C++ C++ C++ C++ C++ C++	Register           iar         8899FC           r0         -           r0         -           r3         -           r15         000000           r18         000000           r29         -           r30         -           r600000         -           r30         -           r30         -           r10         -           r10         -           r10         -	00000000 00000 0000000 4-010 78 lr = r1 = r2 = r7 = r2 - 000000000 716 = 000000000 r19 = 000000000 r19 = 000000000 r19 = 000000000 r19 = 000000000 r19 = 00000000000 r19 = 000000000000000000000000000000000000	280 0000000 00000000 200 0000000000000000	msr = = 0000000000000000000 r9 r12 = = 00000000000000000 = 0000000000000	*
Ga •	0 % • • • • • • • • • • • • • • • • • •	Langua * nne.t C++ rCont C++ rCont C++ ctody C++ ctody C++	Part 2005Ele0           Aur 2005Ele0           Aur 2005Ele0           Aur 2005Ele0           r0 =           r3 =           -           r15 =           r15 =           000000           r21 =           r20 =           r3 =           -           -           r3 =           -           000000           r24 =           r39 =           r30 =           r17 =           r000000           r10 =	00000000 4-000 0000000 4-000 78 lr = r1 = r1 = r2 = r10 = r2 = 0000000 r2 = r2 = 00000000 r2 = 000000000 r2 = rF88D88A5 r25 = 0000000000 r2 = 000 0000000000 r2 = 000	280 0000000 00000000 200 0000000000000000	EST = = 0000000000000000 r12 = = 00000000000000001 = 0000000000000000 = 0000000000	*
<ul> <li>Ga</li> <li>♦</li> <li>♦</li> </ul>	0 % • < III Stack Name Client, 360, pelC, Portal, Player::Render/Screen/SpacePaintEffect()MatRenderContext * pRender client, 360, pelC, Portal, Player::Render/Screen/SpacePaintEffect()MatRenderContext * pRender client, 360, pelC, Portal, Player::Render/Screen/SpaceFfecttffer lient, 360, pelC, Portal, Player::Render/Screen/SpaceFfecttffer lient, 360, pelC/WeinRender::Render/WeinCont CVIew/Setup client, 360, pelC/WeinRender::Render/WeinCont CVIew/Setup client, 360, pelC/WeinRender::Render/WeinCont CVIew/Setup client, 360, pelC/WeinRender::Render/WeinCont CVIew/Setup client, 360, pelC/Host, Run/Frame(Ploat time=0.00000000) line 380 engine, 360, pelCHostState::State:State, Run(float frame-Time=0.00000000) line 712 engine, 360, pelCHostState::FrameUpdate(float time=0.000000000) line 712 engine, 360, pelCHostState::FrameUpdate(float time=0.000000000) line 712 engine, 360, pelCHostState::FrameUpdate(float time=0.000000000) line 712 engine, 360, pelCEngine::FrameUpdate(float time=0.000000000) line 712 engine, 360, pelCHostState::FrameUpdate(float time=0.000000000) line 712 engine, 360, pelCEngine::FrameUpdate(float time=0.000000000) line 713 engine, 360, pelCEngine::FrameUpdate(float time=0.000000000) line 714 engine, 360, pelCEngine::	Langua + nnce_t C++ r(cont C++ abdff C++ - & hu C++ C++ C++ C++ C++ C++ C++ C++	Processing           Register           iar         8899FC           r0         r3           r3         -           r3         -           r4         FFFFF           r27         000000           r30         -           r4         FFFFF           r27         000000           r40         FFFFF           r27         000000           r60         +0.00           fr0         + +0.00           fr4         +6.23           fr4         +6.23	00000000 40000 000000 4000 78 lr = r1 = r1 = r2 = r10 = r2 = 000000000 r15 = 000000000 r25 = 000000000 r23 = 000000000 r23 = 0000000000 r23 = 00000000000 r23 = 00000000000 r23 = 000000000000 r23 = 000000000000 r23 = 0000000000000000 r23 = 000000000000000000000000000000000000	220 0000000 00000000 200 0000000 41000000 ctr = 00000000000000000 175 r5 = r6 = r11 = 0000000000000000 r20 000000000000000 r20 000000000000000 r20 000000000000000 r20 000000000000000 r20 000000000000000 r20 0000000000000000 r20 00000000000000000 r20 0000000000000000 r20 0000000000000000 r20 0000000000000000 r20 00000000000000000 r20 0000000000000000000 r20 000000000000000000 r20 000000000000000000 r20 000000000000000000 r20 00000000000000000 r20 00000000000000000 r20 0000000000000000 r20 00000000000000000 r20 000000000000000000 r20 0000000000000000000000000 r20 00000000000000000000 r20 0000000000000000000 r20 00000000000000000000000000000000000	msr = = 0000000000000000000 r9 r12 = = 00000000000000000 = 0000000000000	*
<ul> <li>Ga</li> <li>♦</li> <li>♦</li> </ul>	0 % • • • • • • • • • • • • • • • • • •	Langua * nne.t C++ rCont C++ rCont C++ ctody C++ ctody C++	Register           iar         =           r3         =           r3         =           r4         =           r5         =           r5         =           r6         =           r7         =           r8         =           r9         =           r15         =           r200000         =           r30         =           r30         =           r30         =           r4         =           r5         =           r6         =	00000000 00000 0000000 4-000 78 lr = r1 = r2 = r3 = r3 = r3 = r2 = r3	280 0000000 00000000 200 0000000000000000	msr = = 00000000000000000 r9 r12 = 000000000000000000 = 00000000000000 = 00000000	*

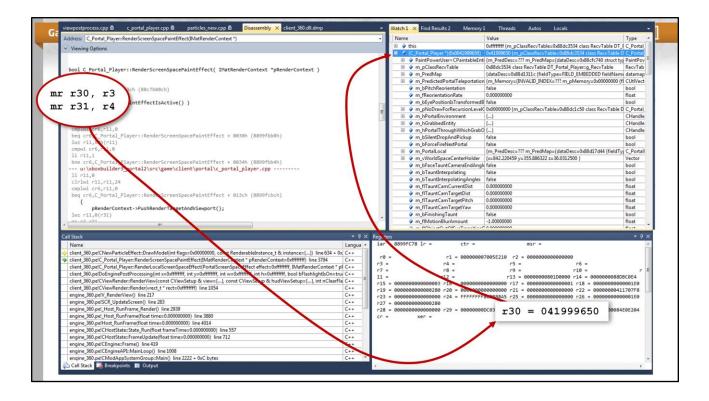
... and is usually pretty good at recovering the nonvolatile registers from the other frames of the stack, again because they must have either been untouched, or stored in memory somewhere.

Concerning and the second	COLUCIE YOLD SCHENDER FOR SCHENDER OF SCHENCE FET		Hudicas.				
And and an other states of the state of the	{	+	0x7005E050		b4 00000000 7005e1ac		
Game De	CMatRenderContextPtr pRenderContext( materials );	-	0x7005E060		70 00000000 84e0dfc8		VALVE
	<pre>pRenderContext-&gt;SetRenderTarget(rt);</pre>		0x7005E070		bc 00000000 84e0e3f4		
0.000	if (rt)		0x7005E080		fc 84d7d9d0 84e0e3f4		
	{		0x7005E090		fc 00000000 dc0392b0		
	pRenderContext->Viewport(0,0,rt->GetActualWidth(),rt->GetActua	lHeight())	0x7005E0A0		00 84d9b3b4 dc0392b0		
	A 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1	1 - B. (1986)	0x7005E0B0		04 84d9b260 00000000		
	}		0x7005E0C0		47 00000000 7005e23f		
			0x7005E0D0		37 00000000 7005e22f		
	ISBN 0110410 CONTRACTOR OF THE RESIDE		0x7005E0E0		30 00000000 84e0dfc8		
	<pre>Dool C_Portal_Player::ScreenSpacePaintEffectIsActive() const</pre>		0x7005E0F0		08 00000000 84e0e3f4		
		and the second second	0x7005E100		fc 84d7c6f8 00000078		
	// The reference is valid and the referenced particle system is al:				f4 00000000 dc0392b0		
	<pre>return m_PaintScreenSpaceEffect.IsValid() &amp;&amp; m_PaintScreenSpaceEffect.IsValid() &amp;&amp; m_PaintScreenSpaceEffect.IsValid()</pre>	ect->IsVal	0x7005E120		04 84d946b4 84e0e244		
	}		0×7005E130		04 84d9b420 00000001		
			0x7005E140		80 84d9aa1c dc0392b0		
			0x7005E150		00 00000000 410de018		
	<pre></pre>	Color1, IM	0x7005E160		00 fffffff 88db88a5		
	{		0×7005E170		00000000 0000000 00		
	<pre>const Color visualColor = MapPowerToVisualColor( m_PortalLocal.m_Pi</pre>		0x7005E180		00 40340000 00000000		
	<pre>Vector vColor1( visualColor.r(), visualColor.g(), visualColor.b()</pre>	);	0x7005E190		00 404ca5dc 1a63c1f8		
	Vector vColor2 = vColor1;		0x7005E1A0		00 bff00000 0000000		
	<pre>for( unsigned i = 0; i &lt; 3; ++i )</pre>	-	0x7005E1B0		00 00000000 411707f8		
	{		0x7005E1C0 0x7005E1D0		00 ffffffff 88db88a5 00 00000000 000001e0		
	vColor2[i] = clamp( vColor2[i] - 15.0f, 0, 255 );						
		*					
	100 % +	*	0x7005E1E0	00000000 000002	80 00000000 00000000 b0 00000000 41000050		-
	100 %	- I X Rec	0x7005E1E0	00000000 000002	80 0000000 0000000		-
	100 % +	+ ₽ × Reg	0x7005E1E0 0x7005E1E0 isters	00000000 000002 00000000 dc0392	80 0000000 0000000	#5C =	+ + ×
	100 % · Call Stack Name	Langua 🔺 ia	0x7005E1E0	00000000 000002 00000000 dc0392	80 0000000 00000000 ba 00000000 11000650	383	
	100 % * Call Stack Name Cill Stack Cilent, 360.pelCNewParticleEffect::DrawModel(int flags=0x00000000, const RenderableInstance, t	Langua ^ ia	0x7005E1E0 0x7005E1E0 isters	00000000 000002 00000000 dr0300 04 lr = 0	80 0000000 00000000 ba 00000000 11000650	msr =	<b>₽</b> ×
	100 % m Cell Stack Name client_360.pelCNewParticleEffect::DrawModel(int flags=0x00000000, const RenderableInstance, t client_360.pelC_Portal_Player=RenderScreenSpacePaintEffect(IMatRenderContext * pRenderCont	Langua ^ ii C++ C++	0x7005E1E0 0x7005E1E0 isters ar = 8899A38	00000000 000002 00000000 dr0300 04 lr = 0	80 0000000 00000000 be 0000000 41000550 ctr =	msr = = 00000000000000000	₹ 7 × 6 E
	100 % " Call Stack Name Call Stack Call Stac	Langua ^ ii C++ C++ C++	0x7005E1E0 0x7005E1E0 isters ar = 8899A38 r0 =	00000000 000002 20000000 dr0302 34 lr = 0 r1 = 0	80 0000000 00000000 ba 0000000 41900650 ctr = 0000000070055290 r2	msr = = 00000000000000000	r6 =
	100 % m Cell Stack Name Client 360.pelC.NewParticleEffect::DrawModel(int flags=0:00000000, const RenderableInstance, t client 360.pelC, Portal Player::RenderScreenSpacePaintEffect(I)MatRenderContex * pRenderCont client_360.pelC, Portal Player::RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect effect=0) client_360.pelC.portal_Player::RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect effect=0) client_360.pelCongineOstroprocessing(int x-dout)fffff, int y=doufffff, int x=doufffff, int x=douffffff, int x=doufffff, int x=douffffff, int x=doufffff, int x=doufffff, int x=douffffff, int x=doufffffff, int x=douffffff, int x=douffffff, int x=doufffffff, int x=doufffffff, int x=doufffffff, int x=doufffffffff, int x=douffffffffffffffffffffffffffffffffffff	Langua / ii C++ C++ C++ C++	0x7005E1E0 0x7005E1E0 isters ar = 8899A38 r0 =	0000000 000002 0000000 dra302 04 lr = 00 r1 = 0 r4 =	80 0000000 00000000 ba 0000000 11000550 ctr = 0000000070055290 r2 r5 =	msr = = 0000000000000000	r6 E
	100 % " Call Stack Name Client, 360, pelCNewParticleEffect::DrawModel(int flags=0x00000000, const RenderableInstance, t Client, 360, pelC, Portal_Player:RenderScreenSpacePaintEffect(IMatRenderContex* pRenderCont @ client, 360, pelC, Portal_Player:RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect)CortalScreenSpaceEffect(PortalScreenSpaceEffect)CortalScreenSpaceE	Langua ^ 14 C++ C++ C++ C++ C++ C++	0x7005E1E0 0x7005E1E0 isters ar = 8899A38 r0 =	0000000 000002 0000000 dr0302 04 lr = r1 = 0 r4 = r7 = r10 =	80 0000000 0000000 b0 0000000 4100055 ctr = 000000007005E290 r2 r5 = r8 =	msr = = 0000000000000000000000000000000000	r6 E
	100 %         m           Call Stack         Name                view Job OpelCNewParticleEffect::DrawModel(int flags=0x00000000, const RenderableInstance, t client, 360, pelC, Portal Player:RenderScreenSpaceParietFfect(MMatRenderContex* * PRenderCont view Job OpelC Dental Player:RenderScreenSpaceFfectPotalScreenSpaceRefectPfectFoot client, 360, pelC/ViewRender:RenderWex(cont CViewScrup & view=[_, cont CViewScrup & hu client, 360, pelCViewRender:RenderWex(cont CViewScrup & view=[_, cont CViewScrup & hu client, 360, pelCViewRender:RenderWixet, ** rect=0xffffff), int 1=034	Langua ^ 14 C++ C++ C++ C++ C++ C++ C++ C++	0x7005E1E0 0x7005E1E0 0x7005E1E0 ar = 8899A38 r0 = 3 =	00000000 000002 00000000 dc0002 14 lr = r1 = 6 r4 = r10 = r12 = 0000000	80 0000000 0000000 50 0000000 41000550 ctr = 0000000070055290 r2 r5 = r8 = r11 =	msr = = 000000000000000 r9 r12 =	r6 E
	100 % m Cell Stack Name client_360,pelCNewParticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t client_360,pelC_Portal_Player:RenderScreenSpacePaintEffect(IMatRenderContext * pRenderCont @ client_360,pelC_Portal_Player:RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect effect=0 client_360,pelCViewRender:RenderView(const CViewSetup & view=(), const CViewSetup & tu client_360,pelCViewRender:RenderView(const CViewSetup & view=(), const CViewSetup & tu client_360,pelCViewRender:RenderView(), tu client client_360,pelCViewRender:RenderView(), tu client client_360,pelCViewRender:RenderViewView(), tu client client_360,pelCViewRender:RenderViewViewTerViewSetupRender:RenderViewViewViewTerViewViewTerViewViewTerViewViewViewTerViewViewViewViewViewV	Langua A 31 C++ C++ C++ C++ C++ C++ C++ C+	0x7005E1E0 0x7005E1E0 15ters ar = 8899A38 r0 = 3 = = 15 = 0000000	00000000 000002 00000000 d-0302 44 lr = r7 = r10 = -12 - 0000000 000000000 r16 = (	80 0000000 0000000 60 0000000 41000550 ctr = 9000000070055290 r2 r5 = r8 = r11 = 900000 r14 = 00000	msr = = 0000000000000000 r9 r12 = = 0000000000000001	r6 E
	100 % m Cell Stack Name client_360,pelCNewParticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t client_360,pelC_Portal_Player:RenderScreenSpacePaintEffect(IMatRenderContext * pRenderCont @ client_360,pelC_Portal_Player:RenderLocalScreenSpaceEffect(PortalScreenSpaceEffect effect=0 client_360,pelCViewRender:RenderView(const CViewSetup & view=(), const CViewSetup & tu client_360,pelCViewRender:RenderView(const CViewSetup & view=(), const CViewSetup & tu client_360,pelCViewRender:Render:RenderView(const CViewSetup & view=(), const CViewSetup & tu client_360,pelCViewRender:RenderView(), tu client_360,pelCViewRender:Render:RenderView(), tu client_360,pelCViewRender:RenderView(), tu client_360,pelCViewRender:RenderViewRender:RenderViewRender:RenderViewRender:RenderViewRender:RenderViewRender:RenderViewRender:RenderViewRender:RenderViewRender:RenderViewRender:RenderViewRender:RenderViewRender:Rend	Langua * 34 C++ C++ C++ C++ C++ C++ C++ C+	0x7005E1E0 0x7005E1E0 isters ar = 8899A38 r0 = 3 = 15 = 0000000 18 = 0000000	00000000 000002 00000000 de0002 r4 lr = r7 = r10 = r10 = 000000003 r16 = ( 000000018 r19 = (	80 0000000 0000000 ba according 11000/50 ctr = 9000000070055290 r2 r5 = r8 = r11 = 90000000000000000 r17	msr = = 0000000000000000 r12 = = 00000000000000000	r6 E
	100 % //////////////////////////////////	Langua A 34 C++ C++ C++ C++ C++ C++ C++ C+	0x7005E1E0 0x7005E1E0	00000000 000002 00000000 de0002 44 lr = r1 = ( r4 = r7 = r10 = r22 - 0000000 000000000 r15 = ( 000000000 r22 = (	88 0000000 0000000 19 0000000 4100055 ctr = 0000000070055290 r2 r5 = r8 = r11 = 000000000000000000 r17 0000000000000000 r17	msr = = 00000000000000000 r12 = = 00000000000000 = 00000000000000 = 00000000	r6 E
	100 % / m Call Stack Name Client, 360, pelC, NewParticleEffect::DrawModel(int flags=0:00000000, const RenderableInstance, t client, 360, pelC, Portal, Player::RenderScreenSpacePaintEffect(I)MatRenderContex * pRenderCont & client, 360, pelC, Portal, Player::RenderLocalScreenSpaceEffect(PlotalScreenSpaceEffect effect=0)d client, 360, pelCViewRender::RenderViceoccontsCViewSetup & viewer[], const CViewSetup & the client, 360, pelCViewRender::RenderViceoccontsCViewSetup & viewer[], const CViewSetup & the client, 360, pelCViewRender::RenderViceoccontsCViewSetup & viewer[], const CViewSetup & the client, 360, pelCViewRender::RenderViceoccontsCViewSetup & viewer[], const CViewSetup & the engine, 360, pelCViewRender::RenderViewDitexCViewSetup & viewer[], const CViewSetup & the engine, 360, pelCViewRender::RenderViewDitexCViewSetup & viewer[], const CViewSetup & the engine, 360, pelCViewRender::RenderViewDitexCViewSetup & viewer[], const CViewSetup & the SetUp = the state of the the state of th	Langua A 14 C++ C++ C++ C++ C++ C++ C++ C+	0x7005E1E0 0x7005E1E0 0x7005E1E0 0x7 = 8899A38 r0 = 3 = 15 = 0000000 18 = 0000000 11 = 0000000 24 = FFFFFF	00000000 000002 00000000 dc0002 r14 lr = r1 = ( r7 = r10 = 22 - 0000000 r15 = ( 000000000 r22 = ( 00000000 r25 = (	88 00000000 0000000 ba aanaaa 11000ta ctr = 900000000000052290 r2 r5 = r11 = 9000000000000000 r17 900000000000000 r17	msr = = 0000000000000000 r12 = = 0000000000000001 = 00000000000000 = 0000000000	r6 E
	100 % m Cell Stack Name client_360.pelCNewParticleEffect::DrawModel(int flags=0x00000000, const RenderableInstance, t client_360.pelC_Portal_Player::RenderScreenSpacePaintEffect(IMatRenderContext * pRenderCont @ client_360.pelCDenginePartProcessing(Imt x=0dfffffff), the v=0dfffffff, int v=0dfffffff, int v=0dfffffff, int v=0dfffffff, int v=0dffffffff, int v=0dffffffff, int v=0dffffffff), int v=0dfffffff), int v=0dffffffff), int v=0dfffffffff), int v=0dfffffffff), int v=0dfffffffff), int v=0dffffffff), int v=0dffffffff), int v=0dffffffff), int v=0dffffffff), int v=0dffffffff), int v=0dfffffffffffffff, int v=0dffffffffffff), int v=0dffffffffffffffffffffff, int v=0dffffffffffffffffffffffffffffffffffff	Langua * 11 C++ C++ C++ C++ C++ C++ C++ C	0x7005E1E0 0x7005E1E0	00000000 000002 r14 lr = r7 = r7 = r10 = 12 00000000 r15 = 0 000000000 r15 = 0 000000000 r22 = 0 F880D88A5 r25 = 0	38 0000000 000000 ba poponon 11000600 ctr = 0000000070055290 r2 r5 = r6 = 00000000000000 r20 00000000000000 r20 0000000000	EST = = 00000000000000000 r12 = 000000000000000 = 00000000000000 = 0000000000	r6 E
	100 %         m           Call Stack         Name                v Client, 360, pelC/NewParticleEffect::DrawModel(int flags=0:00000000, const RenderableInstance, t)                v Client, 360, pelC/NewParticleEffect::DrawModel(int flags=0:00000000, const RenderableInstance, t)                v Client, 360, pelC/NewParticleEffect::DrawModel(int flags=0:00000000, const RenderableInstance, t)                v Client, 360, pelC/NewParticleEffect::DrawTollScreenSpaceEffect effects:DrateGreenSpaceEffect effect:DrateGreenSpaceEffect effect::DrateGreenSpaceEffect effect::DrateGreenSpaceEffect::DrateGreenSpacetEffect::DrateGreenSpaceEffect::DrateGreenSpaceEffect::DrateGr	Langua * 11 C++ C++ C++ C++ C++ C++ C++ C	0x7005E1E0 0x7005E1E0	00000000 000002 r14 lr = r7 = r7 = r10 = 12 00000000 r15 = 0 000000000 r15 = 0 000000000 r22 = 0 F880D88A5 r25 = 0	88 00000000 0000000 ba annana 1100050 ctr = 88000000070052290 r2 r5 = r8 = r11 = 88010000 c14 00000 8000000000000 r10 80000000000000 r20 8000000000000000 r20	EST = = 00000000000000000 r12 = 000000000000000 = 00000000000000 = 0000000000	r6 E
	100 %         m           Cell Stack         Name           Cleint, 360, pelC/NewPanticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t.           Cleint, 360, pelC, Portal_Player::RenderLocalScreenSpacePaintEffect(INtatRenderContext * pRenderCont           Iclient, 360, pelC, Portal_Player::RenderLocalScreenSpaceEffect(PortalScreenSpaceEffectTetTetTiff, int v=0.0dffff, int v=0.0dfffff, int v=0.0dffff, int v=0.0dffff, int v=0.0dfffff, int v=0.0dffff, int v=0.0dffff, int v=0.0dffff, int v=0.0dfffff, int v=0.0dfffff, int v=0.0dffff, int v=0.0dfff, int v=0.0dff, int v=0.0d	Langua * di C++ C++ C++ C++ C++ C++ C++ C++ C++ C+	0x7005E1E0 0x7005E1E0	00000000 000002 r14 lr = r7 = r7 = r10 = 12 00000000 r15 = 0 000000000 r15 = 0 000000000 r22 = 0 F880D88A5 r25 = 0	88 00000000 0000000 ba annana 1100050 ctr = 88000000070052290 r2 r5 = r8 = r11 = 88010000 c14 00000 8000000000000 r10 80000000000000 r20 8000000000000000 r20	EST = = 00000000000000000 r12 = 000000000000000 = 00000000000000 = 0000000000	r6 E
	100 %         m           Call Stack         Name $\varphi$ client, 360, pelC/NewParticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t client, 360, pelC, Portal Player:RenderScreenSpaceParietFect/MatRenderContex* * PRenderCont $\varphi$ client, 360, pelC/Portal Player:RenderScreenSpacePficte(ProtalScreenSpaceRfecte(PfictaDid client, 360, pelC/ViewRender:RenderView(const CViewSetup & view=[], const CViewSetup & hu client, 360, pelC/ViewRender:RenderView(const CViewSetup & view=[], const CViewSetup & hu client, 360, pelC/ViewRender:RenderView(const CViewSetup & view=[], const CViewSetup & hu client, 360, pelC/ViewRender:RenderView(Did Retz)           engine, 380, pel/Sct, Puptatscreen(D line 283 engine, 380, pel/Host, RunFrame(Float time=0.000000000) line 3800 engine, 380, pel/Host, RunFrame(Float time=0.000000000) line 551 engine, 380, pel/Host, RunFrame(Float time=0.000000000) line 551 engine, 380, pel/Host, RunFrame/Parder[Did time=0.000000000) line 551 engine, 380, pel/Host, RunFrame/Parder[Did time=0.00000000) line 551 engine, 380, pel/Host, RunFrame/Parder[Did time=0.00000000) line 551	Langua * 31 C++ C++ C++ C++ C++ C++ C++ C+	0x7005E1E0 0x7005	00000000 00002 011	88 00000000 0000000 ba annana 1100050 ctr = 88000000070052290 r2 r5 = r8 = r11 = 88010000 c14 00000 8000000000000 r10 80000000000000 r20 8000000000000000 r20	#SF = = 0000000000000000 r12 = 000000000000000 = 00000000000000 = 0000000000	r6 E
	100 % / m Call Stack Name Cellist 360, pelC/NewParticleEffect::DrawModel(int flags=0:00000000, const RenderableInstance, t client, 360, pelC, Portal, Player::RenderScreenSpacePaintEffect(I)MatRenderContex * pRenderCont & client, 360, pelC, Portal, Player::RenderLocalScreenSpaceEffect(PlotalScreenSpaceEffect effect=0)d client, 360, pelC/ViewRender::RenderViceoctory & views:[	Langua * 34 C++ C++ C++ C++ C++ C++ C++ C+	0x7005E1E0 0x7005E1E0 1sters 1r = 8899A38 r0 = 3 = 15 = 0000000 18 = 0000000 21 = 00000000 21 = 00000000 21 = 0000000 21 = 00000000 21 = 0000000 21 = 00000000 21 = 00000000 21 = 00000000 21 = 00000000 21 = 0000000 21 = 0000000 21 = 00000000 21 = 000000000 21 = 000000000 21 = 00000000 21 = 000000000 21 = 000000000 21 = 000000000000 21 = 00000000000000000000000000000000000	00000000 00002 0000000 4/0002 4/1 r = r1 = r1 = r1 = r1 = r1 = 000000000 r15 = 000000000 r15 = 000000000 r22 = 000000000 r23 = 000000000 r23 = 000000000 r23 = 000000000 r23 = 000000000 r23 = 000000000 r23 = 0000000000 r23 = 000000000 r23 = 0000000000 r23 = 0000000000 r23 = 00000000000 r23 = 000000000000 r23 = 000000000000 r23 = 0000000000000 r20 = 0000000000000000000000000000000000	38 0000000 0000000 ba aaaaaa 1100affa ctr = 0000000070055290 r2 r5 = r8 = 000000000 r17 00000000000 r17 000000000000 r17 000000000000 r26 0000000000000 r26 0000000000000 r26 00000000000000 r29 000000000000000 r29	EST = = 0000000000000000 r12 = = 0000000000000000 = 00000000000000	r6 E
	100 %         m           Call Stack         Name           v Client, 360, pel:CNewParticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t)         client, 360, pel:CNewParticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t)           client, 360, pel:CNewParticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t)         client, 360, pel:CNewParticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t)           client, 360, pel:CNewParticleEffect::DrawModel(int this - 0.00000000, inclient, 360, pel:CNewBarter:Render(vect, t): rect=0.00000000, inclient, 360, pel:CNewBarter:Render(Vect, t): rect=0.00000000, inclient, 360, pel:CNewBarter:Render()           engine, 380, pel:CNewBarter:Render()         Incli 233           engine, 380, pel:CNewBarter:Render()         Incli 233           engine, 380, pel:CNewBarter:Render()         Incli 233           engine, 380, pel:CNewBrater:Render()         Incli 233           engine, 380, pel:CHostState:State:State:Runf(Incli time=0.000000000)         Incli 557           engine, 380, pel:CHostState:FrameUpdate(Incli time=0.000000000) <td< td=""><td>Langua A C++ C++ C++ C++ C++ C++ C++ C+</td><td>0x7005E180 0x7005E180</td><td>00000000 00002 0000000 4:000 44 lr = r1 = r7 = r3 = 00000010 : 00000000 : r3 = 000000000 : r3 = r3 = 00000000 : r3 = r3 = 00000000 : 000000000 : 0000000000</td><td>38 00000000 0000000 ba aaaaaa 11000cca ctr = 00000000070055290 r2 r5 = r8 = r11 = 000000000000000 r2 00000000000000 r2 0000000000</td><td>#Sr = = 0000000000000000 r12 = = 000000000000000 = 000000000000000</td><td>r6 E</td></td<>	Langua A C++ C++ C++ C++ C++ C++ C++ C+	0x7005E180 0x7005E180	00000000 00002 0000000 4:000 44 lr = r1 = r7 = r3 = 00000010 : 00000000 : r3 = 000000000 : r3 = r3 = 00000000 : r3 = r3 = 00000000 : 000000000 : 0000000000	38 00000000 0000000 ba aaaaaa 11000cca ctr = 00000000070055290 r2 r5 = r8 = r11 = 000000000000000 r2 00000000000000 r2 0000000000	#Sr = = 0000000000000000 r12 = = 000000000000000 = 000000000000000	r6 E
	100 %     m       Call Stack     Name       V Client, 360, pelC/NewParticleEffect::DrawModel(int flags=0.00000000, const RenderableInstance, t       client, 360, pelC, Portal Player::RenderScreenSpacePaintEffect(IMatRenderContext * pRenderCont       ψ client, 360, pelC/Dorbajine/Borbrocessing(int ±-00000000)       client, 360, pelC/ViewRender::RenderVicescrenSpaceEffect(PortalScreenSpaceEffect effect:Dorbajine/Borbrichessing(int ±-00000000)       Lient, 360, pelC/ViewRender::RenderVices(const CViewSetup & views(, const CViewSetup & hu       client, 360, pelC/CViewRender::RenderVices(const CViewSetup & views(, const CViewSetup & hu       client, 360, pelC/CViewRender::RenderVices(const CViewSetup & views(, const CViewSetup & hu       client, 360, pelC/CViewRender::RenderVices(const CViewSetup & views(, const CViewSetup & hu       client, 360, pelC/CViewRender::RenderVices(const CViewSetup & views(, const CViewSetup & hu       client, 360, pelC/CViewRender::RenderVices(const CViewSetup & hu       client, 360, pelC/CViewRender::RenderVices(const CViewSetup & hu       engine, 360, pel/Chot, RunFrame(Ioat time=0.000000000) line 3800       engine, 360, pel/Chot, RunFrame(Ioat time=0.000000000) line 4014       engine, 360, pelCCHot;RunFrame(Ioat time=0.000000000) line 657       engine, 360, pelCCHot;RunFrame(Ioat time=0.000000000) line 557       engine, 360, pelCCHot;RunFrame(Ioat time=0.000000000) line 712       engine, 360, pelCModpRyStater::Frame() line 1008       engine, 360, pelCModpRyStater::Frame() line 1008       engine, 360, pelCMod	Langua - 34 C++	0x70055180 0x70055180 0x70050 0x70050	00000000 00002 0000000 4/0002 4/1 lr = r10 =	88 0000000 0000000 ba aaaaaa 11000000 ctr = 90000000000000000 r5 r8 = r8 = 10000000000000 r17 9000000000000 r15 90000000000000 r25 900000000000000 r25 900000000000000 r25 9000000000000000 r25 90000000000000000 r25 90000000000000000 r25 90000000000000000 r25 900000000000000000 r25 900000000000000000 r25 900000000000000000 r25 900000000000000000 r25 900000000000000000 r25	EST = = 000000000000000 r12 = 0000000000000000 = 00000000000000 = 0000000000	r6 E
	100 %         m           Call Stack         Name           V Clent, 360, pelC/NewParticleEffect::DrawModel(int flags=0:x00000000, const RenderableInstance, t)         client, 360, pelC, Portal, Player::RenderScreenSpaceParticleInstance, t)           v client, 360, pelC/NewParticleEffect::DrawModel(int flags=0:x00000000, const RenderableInstance, t)         client, 360, pelC/NewParticleEffect::DrawModel(int flags=0:x00000000, const RenderableInstance, t)           v client, 360, pelC/NewParticleEffect::DrawModel(int rink v Dofffffff, int v=Dofffffff, int v=Doffffffff, int v=Dofffffff, int v=Doffffffff, int v=Dofffffff, int v=Doffffffff, int v=Dofffffff, int v=Doffffffff, int v=Doffffffff, int v=Dofffffff, int v=Doffffffff, int v=Dofffffffff, int v=Doffffffffff, int v=Dofffffffffff, int v=Doffffffffff, int v=	Langua A C++++ C++++ C++++ C++++ C++++ C++++ C++++++++	0x70055128 x70055128 ar = 8899A38 r0 = 3 3 = - 15 = 0000000 21 = 0000000 21 = 0000000 21 = 0000000 21 = 0000000 21 = 0000000 51 = 00000000 51 = 0000000 51 = 00000000 51 = 000000000 51 = 000000000000 51 = 00000000000000000000000000000000000	00000000 00002 0000000 4:000 44 lr = r1 = r1 = r2 = r3 = 00000016 r13 = 000000016 r13 = 000000008 r28 = 0000000000 r20 = 00000000000 r20 = 000000000000 r20 = 00000000000000000 r20 = 0000000000000000000000000000000000	38 00000000 0000000 ba pononan 1100050 ctr = 2000000007005E200 r2 r5 = r8 = r11 = 2000000000000000 r1 20000000000000 r2 20000000000000 r2 20000000000	EST = = 00000000000000000 r12 = 000000000000000000 = 00000000000000	r6 E

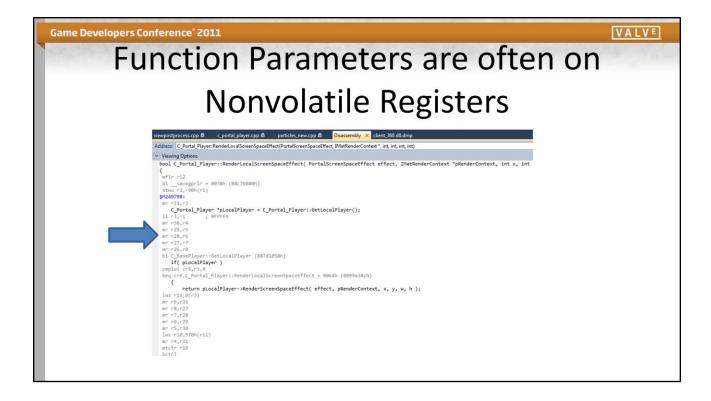
That's what it means to set the "active frame" in the call stack window. It's not just about moving the contents of the source pane to the right place – it tells the debugger to update the stack, local, and register windows to what they would have been in that context. As I move up and down in the call stack pane, it automatically updates r1 in the registers pane to contain the stack base for that function.



So let's say we wanted to recover THIS from one frame up. The watch window lies and says THIS is 0xffffffff.



And that's because the r3 register is volatile and gone by this point. However, if you look carefully at the disasm, you'll see that one of the first things this function did was move the contents of r3 (which you'll recall is where the "this" parameter gets passed) onto r30. R30 is nonvolatile, so if we can find it in the registers pane, it should be possible to punch it into the watch window typecast as the appropriate pointer type, and bang it's like your locals window actually worked properly.



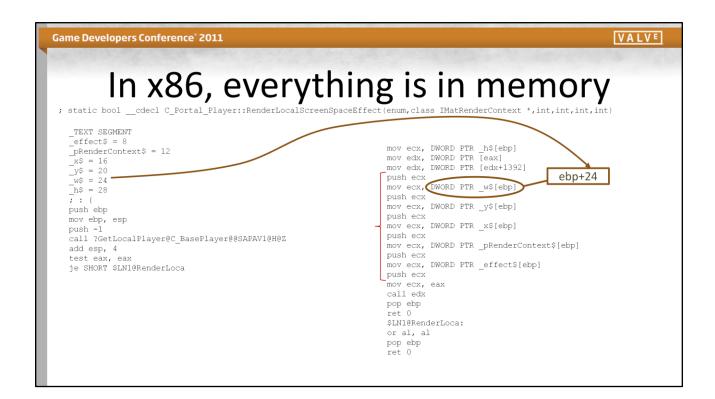
Generally speaking most compilers like to move function params into nonvolatiles. They don't always do it, they don't always do it at the same time or in the same place or in the same order or onto the same registers, so you'll need to look at the disasm to see if it happened in the current frame and if so where the parameters went. However, often you'll get lucky and find your parameters or locals on a nonvolatile somewhere, which means that you can frequently retrieve missing function params in this way.

Game Developers Conference <sup>®</sup> 2011		
SV SV SV SV SV SV SV SV SV SV	Oxfff8         old EBP before function A           Oxfff4         Function B parameter 1           Oxff60         Fetura address to function A           Oxff60         Tetura address to function A           Oxff60         Contine B           Oxff60         Contine B           Oxff60         Contine B           Oxff60         Contine B           Oxff60         Containable           Oxff60         Containable           Oxff60         Containable           Oxff60         Containable           Oxff60         Containable           Oxff60         Containable	<ul> <li>There are many</li> <li>Caller restores ESP <ul> <li>cdecl</li> </ul> </li> <li>Callee restores ESP <ul> <li>stdcall</li> <li>fastcall</li> <li>thiscall (C++ this pointer goes on ECX)</li> </ul> </li> </ul>

The x86 has many different calling conventions, which mostly differ in whether the caller or the callee restores the stack pointer, and what order parameters go onto the stack in. They're numerous and well documented elsewhere, so I'll skip past them.

Diagram credit: Jerry Coffin

http://en.wikibooks.org/wiki/X86 Disassembly/The Stack



On the x86, all parameters always get pushed onto the stack anyway, so they're always in memory. Thus the debugger usually does a pretty good job of retrieving then. Even if it doesn't, you can always go poking around in memory yourself to find them, although your eyes may bleed a bid after combing through the disassembly to try to figure out which offset corresponds with which value.

VALVE

# Debugging live "release builds" is just like debugging a crash

- Watch window lies same way in both cases
- Really the same skills
- Diagnose bugs in situ without needing repro
- Live processes have a little more info
- Connect to QA kits remotely from your desk

Everything I've described just now isn't specific to crashes. All of this is really as much true for debugging any optimized executable, regardless of whether it's dead or alive. So, you can use these same skills to attach to a currently running game – even a release image running on a QA kit – and try to figure out a problem without having to reproduce it on your machine in a debug exe.

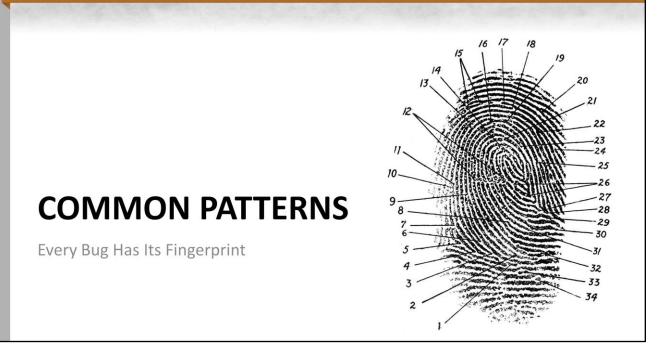
#### VALVE

# "Get this back to my lab"



Also, if you have a live issue on a running kit, but for some reason it's inconvenient for you to debug it at just that moment – maybe you're busy, or QA needs the kit back, or it's going to take a long time – then you can manually trigger a dump to create a state snapshot that you can take back to your workstation and examine at your leisure. There's mechanisms for this on every platform to do this from the debugger, and programmatically from inside the game.

#### VALVE



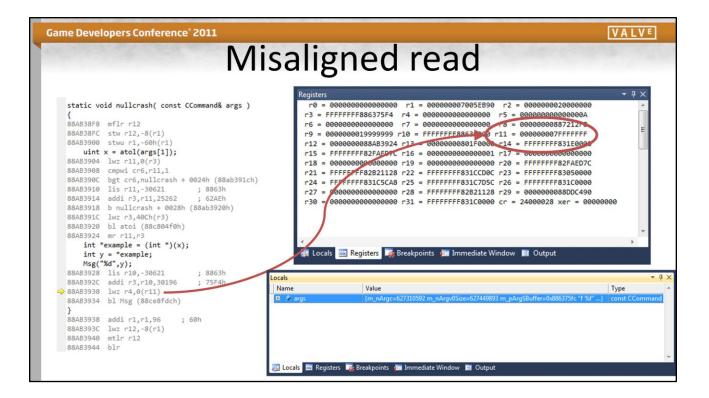
Let's look at some common issues and how to recognize them.

ne Developers Conference' 2011 Misaligr	ned read	VAL
🕴 Normania (1971) 1974 and 👘 🚽 Theorem (1974) 1974 and 👘 💎 🐨 Ball Formary (1984) 2014 (1994) 1974 (1994) 1974 (1994) 1974		

A very simple one: misaligned read – trying to, say, load a 32-bit word from an address that isn't a multiple of four bytes. (That's a crash on PPC.)

ame Developers Conference <sup>®</sup> 2011		VALVE
N/ie	saligned read	
IVII	Saligheu reau	
	0	
<pre>static void nullcrash( const CCommand&amp; args ) {</pre>	Build Debug war Data Tools Test Window Help	
88AB38F8 mflr r12	🔏 🕼 🏝 🔊 - 🕄 - 💭 - 🔍 🕨 Release - Win32 - 🎯 VCTarget.*Path	-   🖏 😤 (
88AB38FC stw r12,-8(r1)	T Streptions	8 23
<pre>88AB3900 stwu r1,-60h(r1) uint x = atol(args[1]);</pre>		
88AB3904 lwz r11,0(r3)	Steel Break when an exception is:	OK
88AB3908 cmpwi cr6,r11,1	ain.cpp Name Thrown User-unhandled	Cruch
88AB390C bgt cr6,nullcrash + 0024h (88ab391ch)	(const C B Win32 Exceptions	Cancel
88AB3910 lis r11,-30621 ; 8863h	- 40010005 Control-C	
88AB3914 addi r3,r11,25262 ; 62AEh 88AB3918 b nullcrash + 0028h (88ab3920h)	40010000 Control Prod	
88AB391C lwz r3,40Ch(r3)	- 8000002 Datatype misalignment	Find
88AB3920 bl atoi (88c804f0h)	- cloucos Access sidedon	Find Next
88AB3924 mr r11,r3	1	rind ivext
<pre>int *example = (int *)(x);</pre>	- c0000008 An invalid handle was specified	
<pre>int y = *example;</pre>	- c0000017 Not Enough Quota	
Msg("%d",y); 88AB3928 lis r1030621 : 8863h	- c000001d Illegal Instruction	Reset All
88AB3928 lis r10,-30621 ; 8863h 88AB392C addi r3,r10,30196 ; 75F4h	- c0000025 Windows cannot continue from this exception	
> 88AB3930 lwz r4,0(r11)	file c0000026 An invalid exception disposition was returned by an e	
88AB3934 sg (88ce8fdch)	- c000008c Array bounds exceeded	Add
}	- c000008d Floating-point denormal operand	Delete
	c00000Re Floating-noint division by zero	
88AB3938 addi,96 ; 60h		
88AB3938 addi 121,96 ; 60n 88AB393C lwz r12, r1) 88AB3940 mtlr r12	Dense ( and framed and )	

Okay, that's pretty easy. MSVC will tell you, misaligned read. Before we move on, take a quick look at what those two numbers mean: one is the instruction of the faulty address, and the other is the exception code. You can map numeric exception codes to strings by looking at the "exceptions" dialog under the debug menu.



Even if you didn't have that, it's still pretty easy to trace. You can see that the faulty instruction here is "load word from address in register 11." Well, register eleven contains an odd number, which can't possibly be word aligned, so boom.

Address: stackfault(const	CCommand &)		<ul> <li>Address: 0x70</li> </ul>	000FBF0			• {\$}
Viewing Options			0x7000	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	*****	******	
88AB4924 lwz r3,40	(h/a2)		0x700/FC00	******	******	******	
88AB4928 bl atoi (			0x7000FC10	******	*****************	******	
88AB492C mr r11,r3			0x7000FC20	******	******	******	
char *foo;			0.7000FC30	*****	******	* *******	
	; i < sentinel ; ++i )		0x7000FC40		*****		
88AB4930 cmpwi cr6			0x7000FC50		******		••••••
88AB4934 ble cr6,s	tackfault + 0058h (88ab4950	3h)	0x7000FC60		*****		
{			0x7000FC70	Second second second second	******		
	<pre>*)_alloca(1024);</pre>		0×7000FC80 0×7000FC90		******		
88AB4938 lwz r10,0			0x7000FC90		>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>		
88AB493C addic. r1			0x7000FCB0		>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>		
	-450h(r1)		0x7000FCC0		*****		
88AB4944 addi r4,r			0x7000FCD0		******		
	faul: + 0040h (88ab4938h) ult + 005ch (88ab4954h)		0x7000FCE0	******	******* ******	******	
88AB4950 lwz r4,50			0x7000FCF0	******	******* ******	******	
3			0x7000FD00	?????????	******	******	
Msg("%p\n",foo)			0x7000FD10		******		
88AB4954 lis r11,			0x7000FD20		******		
88AB4958 addi r3 r	11,-3924 ; 0F0ACh		0x7000FD30		******		•••••
88AB495C bl Msg (8	8ce91a4h)		0x7000FD40		*****		
}		/	<ul> <li>0x7000FD50</li> <li>0x7000FD60</li> </ul>		>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>		
·		•	0.70001000				
Watch 1	/		- 4 × Ca	all Stack			<b>-</b> q
Name	Value	Туре	*	Name			Lan
🥥 r1	0x000000070010040	unsignedint64		client_360.d	Ill!stackfault(const CC	ommand & a	rgs={}) line 742 C++
r1-0x450	0x00000007000fbf0 🥖	unsignedint64					mandTarget_t eTa C++
					dll!Cbuf_Execute() lin		
							08322952e+032) Ii C++
							8322952e+032) lir C++
							neTime=-9.083229 C++

Stack overflow. Well, this one's generally easy. Is it the "update stack" opcode? If so, is that stack pointer being updated to an address that isn't mapped memory? Or, more generally, the CPU exception is "access violation." Is it trying to write to some pointer which is an offset from the stack pointer, and if so, is that offset in unmapped memory?

m	e Developers Conference <sup>®</sup> 2011			VALVE
	Jump throu	igh k	bad	vtable
Ca	ll Stack	14E224A5	movss	<pre>xmm0,dword ptr [real@3f800000 (</pre>
	Name	14E224AD	mov	esi,eax
-	61707320()	14E224AF	mov	eax, dword ptr [esi]
4	Client.dll!oops(const CCommand & args={}) Line 730 + 0x	14E224B1 14E224B4	add	edx,dword ptr [eax+10h]
	Client.dll!ConCommand::Dispatch(const CCommand & con	14E224B4	mov	esp,4 ecx,esi
	engine.dll!Cmd_ExecuteCommand(ECommandTarget_t eTar	14E224B9	movss	dword ptr [esp],xmm0
	engine.dll!Cbuf_Execute() Line 287 + 0x11 bytes	14E224BE	call	edx

Okay, how about a jump through a bad virtual function.

CThingy(int x) : m n(x) {}

virtual int Get() { return m n; }

virtual float Slope( float t );

class CThingy

int m n;

public:

};

delete foo;

virtu

11 . . {

void oop

Polyno

Msg("

};

{

Let's say you have a basic Polynomial class with six virtual functions, which means somewhere in memory is a table with six function pointers in it – its vtable. Then you have your function that calls a couple virtuals on it, and you blow up.

14E224C2 movss

> 14E224C0 mov

Jumping through a bad virtual function pointer will usually have a pretty obvious signature like this, where you jumped to an address that wasn't actually in the code segment – the call stack will contain a frame that's just a number, not corresponding to any legitimate function. But with a little investigation you can usually divine a little more about it. If you look at the code around the last frame on the stack, you can find that the Polynomial, which was being passed as ecx (the "this") pointer to the virtual function, was on esi. So we put esi in our watch window and AHA! Its vtable is not that of a Polynomial at all. It is that of a Thingy, which has only one virtual function. So, our factory function must have returned a pointer to something that wasn't actually a Polynomial.

VALVE

bcc8 {m n=0x00000000 }

1018 const CThingy:: vftable

0x00000003

0000

40036

eax, dword ptr [esi]

adv dward ate fasvill

xmm0,dword ptr [\_\_real@3f800000 (

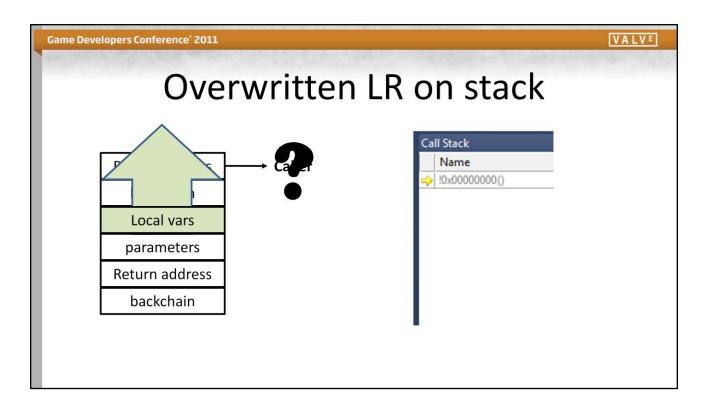
Type

Polynomial \*

unsigned int

CThingy

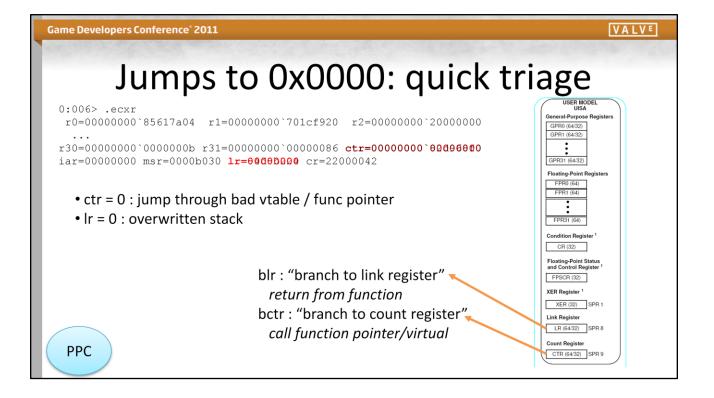
float \*



Bad return addresses – aka the smashed stack. Insidious, annoying, and common. When one function calls another, of course it must store its return address somewhere in memory, usually on the stack. Well, great, except our local data's also on the stack, and thus when you write off the end of an array you can easily overwrite the backchain (making it impossible for the debugger to find parent stack frames) and the return address.

When this happens a function will "return" to an address that's garbage or NULL.

MSVC will usually refuse to give you any kind of call stack when this happens, or even give you a memory window unless you switch to a different thread.



There's exactly two opcodes on the PPC that provide a "jump to address", each of them using a specific register. Looking at which register caused the fault can give you a quick hint as to what went wrong. If it's the count register (used for function pointer calls), then you probably have a bad vpointer, or the like. If the Link Register is NULL, then something overwrite its cell on your stack.



# Thinking Forensically



Ultimately you need to think like a detective at an accident scene. Here you have all these smashed bits and pieces of evidence, and you're trying to work backwards to figure out what caused them to get there like that. And sometimes it can take a little bit of detective work and inference to get to your conclusion.

(I'm not quite sure what that picture is... I typed "forensics" into the NIH's website and that came up.)

VALVE

# Stack Reconstruction

0x7005D8C0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
0x7005D8E0		00000000	00000000	00000000	00000000	00000000	00000000	00000000	
0x7005D96	7005d960	05dbc0	00000000	00000000	7005da80	8281c694	827e5634	dda199a0	
0x7007D920		00000000	0000001	7005dac0	00000000	0000002f	00000002	7005da90	
0x700.D940		7005da10	00000000	00000000	00000000	dd328c30	00000000	00000001	
0x7005D <b>%</b>	0000000	b82eb8	87376b54	00000000	00000000	00000000	fffffff	831e0000	
0x7005D980		831e0000	00000000	dd500640	00000000	0000001	00000000	00000000	
0x7005D9A0	00000000	00000000	00000000	00000001	00000000	dc37e020	82225a78	00000001	
0x7005D9C0	7005da20	dcb82eb8	855fbc9c	00000000	7005da40	84e0bea8	84d7596c	04fd0040	
0x7005D9E0	7005da40	000003d	00000000	00000000	00000000	dd7bb000	87381880	000700ff	
0x7005DA00	7005daa0	000000fe	00000000	8281bdc0	00000000	dc37e020	82226128	0000002c	
0x7005DA20	7005da90	dd328c30	00000000	00000000	7005da90	dd7bb000	87376b54	5f696e6e	
0x7005DA40	7005daa0	00000000	00000002	7005dab0	00000000	dd67e780	00000000	00000000	
0x7005DA60	fffffff	86ba0000	00000000	00000000	7005daf0	84e0dfc8	00000000	00000000	
0x7005DA80	6d6f6465	dc878848	82225e50	00000000	7005db10	dd7bb000	86bfc9d0	5f646573	
0x7005DAA0	7005db50	74696f6e	2f727562	626c655f	77616c6c	5f636569	6c696e67	30310000	
0x7005DAC0	6d6f6465	6c735c70	00000000	00000001	00000000	000001c	00000000	00000001	
0x7005DAE0	00000000	dca6db1c	fffffff	fffffff	00000000	00000079	00000000	00000001	

Sometimes when part of your stack has been trashed, but you still have an r1 pointer and know generally where the bottom frame ought to be, you can make an educated guess about what the actual stack was by looking for consistent chains. Because the stack is so heavily trafficked, old stack frames – those "left over" as traces in memory from previous operations – would tend not to be internally consistent; if you have a chain that forms a complete, consistent link all the way back to main(), it has good odds of being the actual one under execution at the time of death.

Let's say this is a stack, and your r1 is 0x7005D8C0. The bottom frame is all zeroed out, but the damage doesn't seem to go too far. Maybe we can recover something. Any cell containing a number like 0x7005D\*\*\* is probably a pointer to a stack location, since it's so near the value of r1 and we know a priori that's where the stack is.

So, if we were to guess at 0x7005D900 being the bottom of a frame, that means that it links to 7005d960, and... No, that's null.

VALVE

# Stack Reconstruction

0x7005D8C0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
0x7005D8E0	00000000	00000000	00000000	0000000		00000000	00000000	00000000	
0x7005D900	7005d960	7005dbc0	00000000	00000	7005da80	1c694	827e5634	dda199a0	
0x7005D920	7005d9c0	00000000	00000001	7005daco		0000002f	00000002	7005da90	
0x7005D940	00000002	7005da10	00000000	000000000	00000000	dd328c30	00000000	00000001	
0x7005D960	00000000	dcb82eb8	87376b54	0000000	00000000	00000000	fffffff	831e0000	
0x7005D980	fffffff	831e0000	00000000	dd500640	00000000	0000001	00000000	00000000	
0x7005D9A0	00000000	00000000	000000000	00000001	00000000	dc37e020	82225a78	00000001	
0x7005D9C0	7005da20	dcb82eb8	855fbc9c	00000000	7005da40	84e0bea8	84d7596c	04fd0040	
0x7005D9E0	7005da40	000003d	0000000	00000000	00000000	dd7bb000	87381880	000700ff	
0x7005DA00	7005daa0	000000f	00000000	8281bdc0	00000000	dc37e020	82226128	0000002c	
0x7005DA20	7005da90	dd328c30	00000000	00000000	7005da90	dd7bb000	87376b54	5f696e6e	
0x7005DA40	7005daa0	00000000	00000002	7005dab0	00000000	dd67e780	00000000	00000000	
0x7005DA60		6ba0000	00000000	00000000	7005daf0	84e0dfc8	00000000	00000000	
0x7005DA	6d6f6465	5 378848	82225e50	00000000	7005db10	dd7bb000	86bfc9d0	5f646573	
0x7005DAA0		74696f6e	2f727562	626c655f	77616c6c	5f636569	6c696e67	30310000	
0x7005DAC0	6d6f6465	6c735c70	00000000	0000001	00000000	000001c	00000000	00000001	
0x7005DAE0	00000000	dca6db1c	fffffff	fffffff	00000000	00000079	00000000	00000001	

How about this one? No, that's not a legitimate stack address.

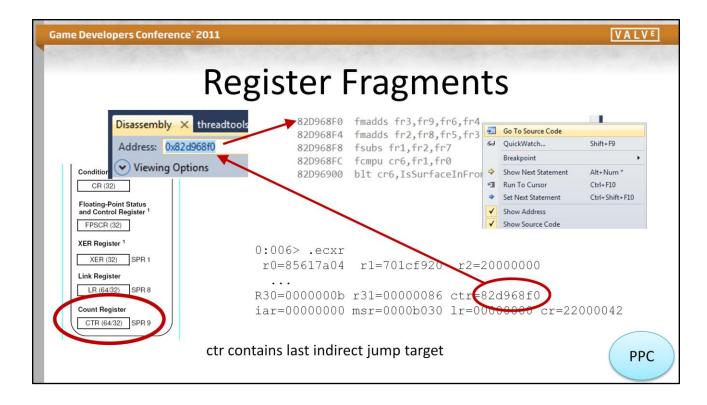
VALVE

# **Stack Reconstruction**

0x7005D8C0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
0x7005D8E0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	
0x7005D900		7005dbc0	00000000	00000000	7005da80	8281c694	827e5634	dda199a0	
0x7005D9	7005d9c0	00000	00000001	7005dac0	00000000	0000002f	00000002	7005da90	
0x7005D940		.005da10	00000000	00000000	00000000	dd328c30	00000000	00000001	
0x7005D960	00000000	dcb82eb8	87376b54	00000000	00000000	00000000	fffffff	831e0000	
0x7005D980	fffffff	831e0000	00000000	dd500640	00000000	00000001	00000000	00000000	
0x700509A0		0000000	00000000	00000001	00000000	dc37e020	82225a78	00000001	
0x7005D9	7005da20	82eb8	855fbc9c	00000000	7005da40	84e0bea8	84d7596c	04fd0040	
0x700bD9E0		000003d	00000000	00000000	00000000	dd7bb000	87381880	000700ff	
0x7005DA00		00000fe	00000000	8281bdc0	00000000	dc37e020	82226128	0000002c	
0x7005	7005da90	28030	00000000	00000000	7005da90	dd7bb000	87376b54	5f696e6e	
0x7005DA40		0000000	00000002	7005dab0	00000000	dd67e780	00000000	00000000	
0x7005DA60	fffffff	86ba0000	00000000	000000		4e0dfc8	00000000	00000000	
0x7005DA80	6d6f6465	dc878848	82225e50	00000	7005db10	000dd	86bfc9d0	5£646573	
0x7005DAA0	7005db50	74696f6e	2f727562	6260655		51636569	6c696e67	30310000	
0x7005DAC0	6d6f6465	6c735c70	00000000	00000001	00000000	000001c	00000000	00000001	
0x7005DAE0	00000000	dca6db1c	fffffff	ffffff		000079	00000000	00000001	

How about d920? Well, that links to d9c0, to da20, and that looks like a consistent chain! And indeed if you follow that up you get back to main, so that looks real.

And of course you can write debugger scripts to do this for you, based on a guess at a stack address.



Sometimes the registers contain trace evidence that can point near the fault. For example, remember that ctr contains the address of the last virtual function called. So, if your crash is due to a bad LR, CTR might still contain a pointer to the most recently called virtual function, perhaps even the function that crashed. Even if not, it might be some function called recently, which can be an important clue.

#### VALVE

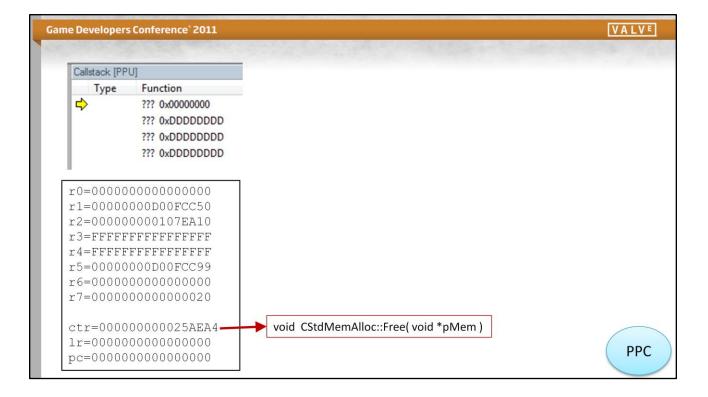
# Working Backwards: Case Study

### BUG #84092: CRASH starting game.

"Happened only on the kits set to Swedish and Danish."

\\dumpserver\Test26\undescriptivename.dmp \\dumpserver\Test27\lessdescriptivename.dmp

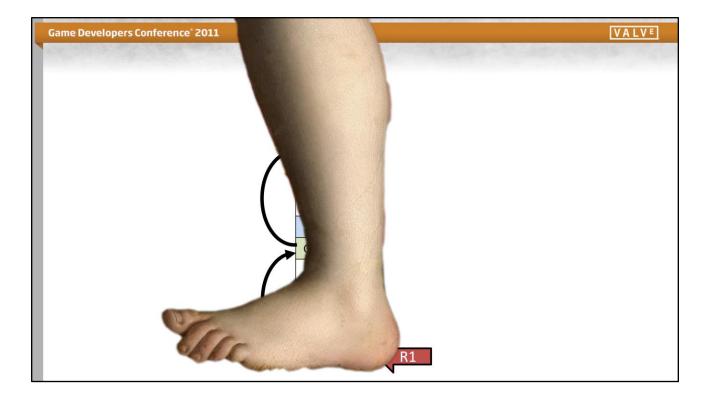
Now let me take you through a fun one that actually came up while I was putting these slides together.



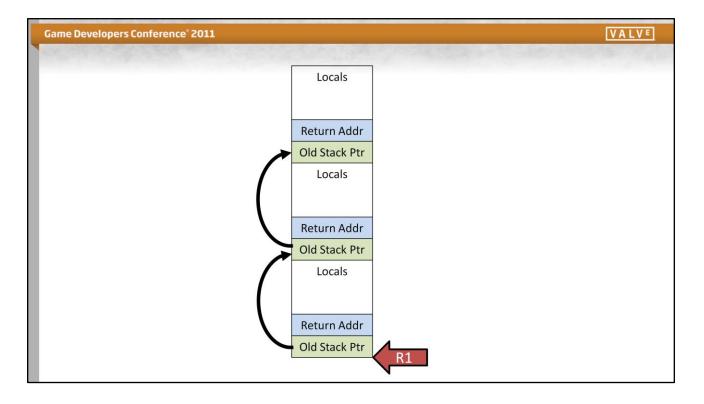
I opened up one of those dumps and got this call stack, so I knew that a) something had trashed my stack and b) I wasn't going to have a good day. The first thing I looked at was the registers pane, and I saw that LR was null, but CTR had a valid code address, so I looked at that, and found it was pointing to... free(). Okay, that's not very specific at all.

Developers						1
		Memory - 4 Byte	Hex [PPU]			0.05
Callstack [PPU]		Address r1		🗸 🖏 🔒	Columns 4	-
Туре	Function					
<₽	??? 0x0000000	D00FCC50	FFFFFF22	00000000	00000000	00000000
	??? 0xDDDDDDDD	D00FCC60	00000000	00000000	00000000	0000000
	??? 0xDDDDDDDD	D00FCC70	00000000	00000000	00000000	0000000
	??? 0xDDDDDDDD	D00FCC80	00000000	00000000	FFFFFF30	FFFFFF
		D00FCC90	FFFFFF9E	FFFFFF1C	FFFFFF3E	FFFFFE
r0=000000000000000 r1=00000000000FCC50 r2=000000000107EA10		D00FCCA0	FFFFFFC3	00000000	00000000	0000000
		D00FCCB0	00000000	00000000	00000000	0000000
		D00FCCC0	00000000	00000000	00000000	0000000
		D00FCCD0	00000000	00000000	FFFFFF7B	FFFFFFF
r3=FFFFFFFFFFFFFFF		D00FCCE0	FFFFFF06	00000000	00000000	FFFFFF6
r4=FFFFFFFFFFFFFFFF r5=0000000000FCC99		D00FCCF0	FFFFFFFF	FFFFFF13	00000000	0000000
	000000000000	D00FCD00	00000000	00000000	00000000	0000000
	000000000000000000000000000000000000000	D00FCD10	00000000	00000000	00000000	00000000
/=00000	0000000020	D00FCD20	00000000	00000000	FFFFFF75	FFFFFFF
	00000025AEA4	D00FCD30	FFFFFF0D	00000000	00000000	FFFFFF7
	00000025AEA4	D00FCD40	FFFFFFD	FFFFFFOE	00000000	00000000
		D00FCD50	00000000	00000000	00000000	00000000
C=00000	0000000000	D00FCD60	00000000	00000000	00000000	00000000

Well, r1 is still legit, so it points at the stack. What's on my stack frame? A bunch of zeroes and FFs, apparently, so I can see my stack has been well and truly trashed for quite a ways.



Remember that stack frames form a linked list in memory, where the "backchain" cell of each frame is the pointer to the previous frame. The way that the debugger reconstructs your call stack pane is by walking that linked list. Previously I showed how to do that by hand by looking upwards a few frames past the damage, but here it looks like the entire stack has been... stepped on.



Except when you "return" from a frame, you don't actually delete anything. You just move the pointer up. The old frames, from the functions you called, stay there in memory like ghosts.

??? 0xDDDDDDDD         D00FCC60         00000000         00000000         00000000         00000000         0000000000000000000000         000000000000000000000000000000000000			Memory - 4 Byte I	Hex [PPU]			1000
Type         Function           ???         0x0000000           ???         0x00DDDDDDD           D00FCC50         FFFFFF22         0000000         0000000         0000000           ???         0x0DDDDDDDD         D00FCC60         0000000         0000000         0000000         0000000           ???         0x0DDDDDDD         D00FCC70         0000000         0000000         0000000         0000000         0000000           ???         0x0DDDDDDDD         D00FCC70         0000000         00000000         00000000         0000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         00000000         0000000000000000         00000000         00000	Callstack [PPU]		Address r1		▼ 🖏 🔒 Columns 4 💌		
??? 0xDDDDDDDD         D00FCC60         00000000         00000000         00000000         0000000000000000000000000000000         000000000000000000000000000000000000	Туре	Function					
??? 0xDDDDDDD         D00FCC70         00000000         00000000         00000000         00000000         00000000           ??? 0xDDDDDDD         D00FCC80         00000000         00000000         FFFFF30         FFFFFF	<₽	??? 0x00000000	D00FCC50	FFFFFF22	00000000	00000000	00000000
Woldstand         D00FCC80         00000000         00000000         FFFFF30         FFFFFF           Go to Address         Cttl-G         Go to Address         Cttl-G         FFFFF9E         FFFFF72         FFFF72         FFF722         FFFF72         FFFF72         FFF722         FFF722         FFF722         FFF722         FFF722         FFF722         FF722         FF7222         FF722         FF7222<		??? 0xDDDDDDDD	D00FCC60	00000000	00000000	00000000	00000000
Go To Address         Ctrl-G           Go to Module         Alt+G           Go to Module         Alt+G           Go to Module         Alt+G           Go to Module         Alt+G           Space         DUUFCEBU           DUUFCEBU         00000000 0000000 0000000 00000000           C=00000         Go to Address           C=000000000000000000000000000000000000		??? 0xDDDDDDDD	D00FCC70	00000000	00000000	00000000	00000000
Go to Module Go to Module 1=000000000C         Alt+G Go to Space DOUFCCBU         FFFFFFC3         00000000         00000000         00000000           1=000000000FCC50         D00FCCC0         D00FCCC0         000000000000000         00000000000000000				00000000	00000000	FFFFFF30	FFFFFFE
CO=00000000C         Go To Source         Space         DOOFCCBU         FFFFFFC3         0000000000000         000000000000000000         000000000000000000000000000000000000				FFFFFF9E	FFFFFF1C	FFFFFF3E	FFFFFFE4
DOUFCESD				FFFFFFC3	00000000	00000000	00000000
22=00000       000000000000000000000000000000000000		00000		00000000	0000000	00000000	00000000
3=FFFFFi       void CFont::GetCharRGBA(         4=FFFFFi       wchar_t ch,         5=000000       int rgbaWide, int rgbaTall,         0:6=000000       unsigned char *pRGBA)         ctr=000000000000000000000000000000000000			D00FCCC0	00000000		00000000	00000000
Lr=000000000000000000000000000000000000	c3=FFFFF c4=FFFFF	voi 1	wchar_t o	ch,			
	6=00000				2		
	6=00000 7=00000 tr=0000	COUDOUZJAEA4	unsigned	char *	pRGBA	)	00000000

So, on the assumption that my function must have previously called some other function in the past, I looked \*down\* the stack – towards where a callee's frame would have been – to look for any frames that pointed at the *current* r1 in the exception. And there is one! Okay, if that's a backchain, then the old return address must be here, try looking that up in the code, and...

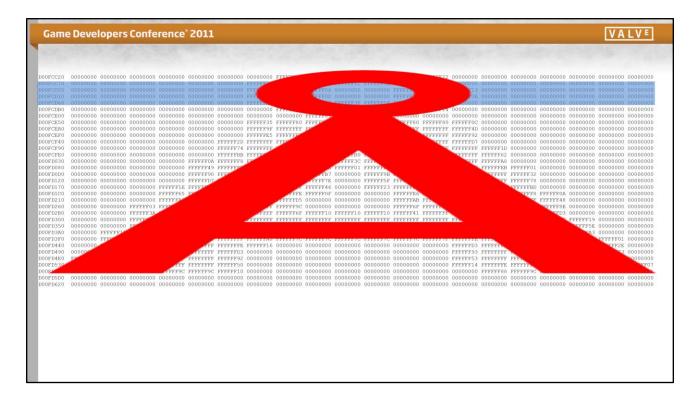
Boom, at some point, we called: void CFont::GetCharRGBA( wchar\_t ch, int rgbaWide, int rgbaTall, unsigned char \*pRGBA )

me De	evelopers Conference° 2011	VALVE
DOOFCCOO	00000000 D00FCCB0 00000000 00299D20 00000000 0120B87C 00000000 00000001	
DOOFCC20	00000000 00000000 00000000 00000000 0000	
000FCC40	FFFFFFA FFFFFFF FFFFFFFF FFFFFD9 FFFFFF22 00000000 00000000 00000000	
000FCC60	00000000 00000000 00000000 00000000 0000	
DOOFCC80	00000000 00000000 FFFFFF30 FFFFFFE FFFFF9E FFFFF1C FFFFFF3E FFFFFE4	
DOOFCCAO	FFFFFC3 0000000 00000000 00000000 0000000 00000	
DOOFCCCO		
D00FCCE0		
DOOFCDOO		
D00FCD20		
DOOFCD40		
D00FCD60		
DOOFCD80		
DOOFCDAO		
DOOFCDCO		
DOOFCDE0 DOOFCE00		
DOOFCE20		
DOOFCE40		
DOOFCE60		
DOOFCE80		
OOFCEAO		
DOOFCECO		
DOOFCEEO		
DOOFCFOO		
DOOFCF20		
DOOFCF40		
D00FCF60	FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFFFFFF	
DOOFCF80	00000000 00000000 00000000 00000000 0000	
D00FCFA0	00000000 FFFFF74 FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF	
DOOFCFCO	FFFFFFF FFFFFFFF FFFFFF1D 00000000 00000000 00000000 00000000 0000	
D00FCFE0		
D00FD000		
000FD020		
D00FD040		
D00FD060		
D00FD080		
DOOFDOA0		
000FD0C0		
00FD0E0		
D00FD100 D00FD120		
D00FD120		
D00FD140		
D00FD180		
DOOFDIAO		
D00FD1C0		
DOOFDIED		
D00FD200		
D00FD220		
D00FD240		

Armed with this knowledge, we can go back to the memory window and make an educated guess about the pattern there. Does it show up with eight column width?

me De	velope	ers Co	nferen	ce° 20	11												VAL	VE
DOOFCBCO	00000000	00000001	00000000	D00FD9A0	00000000	D00FCC50	00000000	0107EA10	00000000	015271A4	00000012	00000000	00000000	00000000	00000000	0107EA10		
DOOFCCOO	00000000	DOOFCCBO	00000000	00299D20	00000000	0120B87C	00000000	00000001	00000000	00000000	00000000	00000000	00000000	00000000	00000000	FFFFFF6C		
DOOFCC40	FFFFFFA	FFFFFFFF	FFFFFFFF	FFFFFFD9	FFFFFF22	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000		
DOOFCC80	00000000	00000000	FFFFFF30	FFFFFFE	FFFFFF9E	FFFFFF1C	FFFFFF3E	FFFFFFE4	FFFFFFC3	00000000	00000000	00000000	00000000	00000000	00000000	00000000		
DOOFCCCO	00000000	00000000	00000000	00000000	00000000	00000000	FFFFFF7B	FFFFFFF8	FFFFFF06	00000000	00000000	FFFFFF68	FFFFFFFF	FFFFFF13	00000000	00000000		
DOOFCDOO	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	FFFFFF75	FFFFFFB	FFFFFF0D	00000000	00000000	FFFFFF73		
DOOFCD40	FFFFFFD	FFFFFF0E	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	FFFFFF23	FFFFFFF9		
									00000000									
									FFFFFF14									
									FFFFFF09									
									00000000									
									00000000									
									00000000									
									FFFFFFFF									
									FFFFFFFF									
									00000000									
									00000000									
									00000000									
									FFFFFFFF									
									FFFFFFEE									
									FFFFFF90									
									00000000									
									00000000									
									FFFFFFFF									
									FFFFFF0F									
									FFFFFFFF 00000000									
									FFFFFF8E									
									FFFFFFFFF									
									FFFFFFFF									
									FFFFFFFF									
									000000000									
									FFFFFFFF									
									FFFFFF7F									
									000000000									
									FFFFFFFF									
									FFFFFF2A									
									FFFFFFFF									
									00000000									
									00000000									
									00000000									
									00000000									
									00000000									
									00000000									
DOOFD6C0									00000000									
D00FD700									00000000									
D00FD740	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000		
D00FD780	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000		
DOOFD7C0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000		
															00000000			

Sixteen column?



Twenty column?

Do you see it?

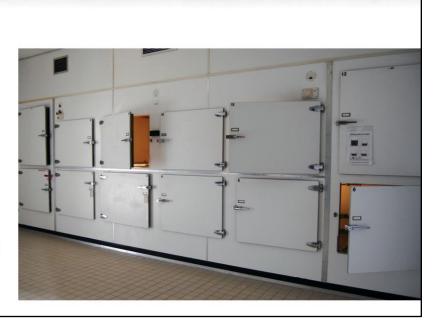
ENHANCE!

And our stack frame – it's too small for you to see – is supposed to be there!

So what happened here is our font had bad metrics for certain diacritics, and in this case when we went to render the Swedish Å, the circle actually poked up out of its memory and into the stack frame!

### COLLECTING EVIDENCE

Making All The Bodies End Up In The Same Place



VALVE

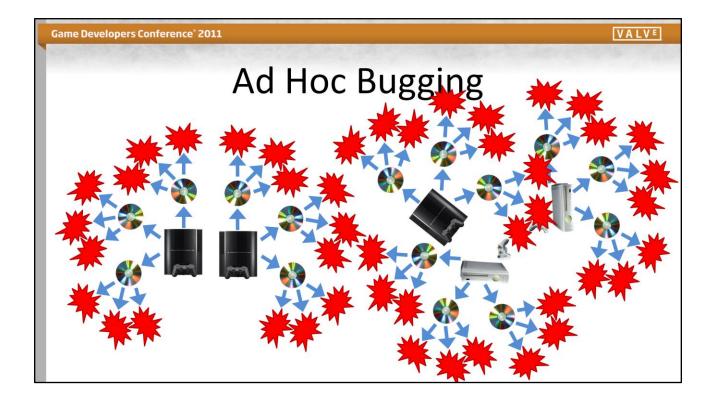
Returning to the subject of core dumps and getting them from QA. You want to have all your core dump files end up in a central location. Consider the alternative: let's say each tester, on encountering a crash, enters a bug. Then to that bug she attaches a dump file, like attaching it to an email. So you have a lot of bugs each with dumps in them. Okay, that's fine when you have one tester...



...playing one image...



But really you have a fresh image every day.

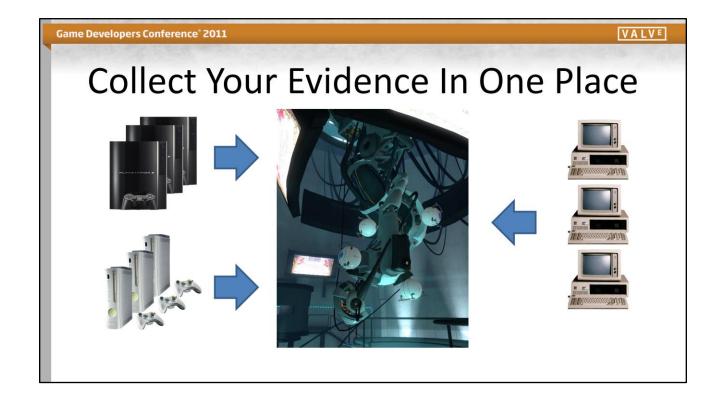


And lots of testers on different platforms.



Lots and lots of different platforms. Maybe PC and Mac too.

Managing each crash individually like this is the road to madness.



So it can help a lot to configure each of your testing endpoints so that they automatically upload all of their dumps to a centralized place, or at least you have some automated process that sweeps them there. Once you've got that level of automation, you can add a bunch of other useful steps to it as well.

# Review: Anatomy of a Core Dump

#### Usually

- Name/timestamp of app
- CPU context (registers)
- Exception data
- List of running threads
  - CPU context for each
  - Their callstacks
- List of loaded modules
- Memory address maps

#### Sometimes

- Entire contents of stack
- OS state
  - Names of other processes
  - Mutexes, etc
- Entire process memory ("full dump")
  - On 360, not "physically mapped" pages
- Custom additions

What's in a dump?

Exception data: what caused the crash. Was it a segfault, an invalid instruction?

On the 360, if you have any memory mapped via "physicalalloc", ie to get a 16mb page, it won't appear in a dump file even if you've configured for "full dump."

VALVE

#### VALVE

## Symbols

- A symbol file maps source-level constructs to machine code and addresses in the executable.
- Produced by compiler/linker along with exe
- Debugger needs matching:
  - Source
  - Executable
  - Symbols
  - Data

Symbol files get invalidated whenever the output binary changes, so

Each platform has its own format of symbol file.

### PDBs

- MSVC and other Microsoft tools
- Proprietary format
- Generated alongside .exe's and .dll's
- Symbols available even for "release" builds

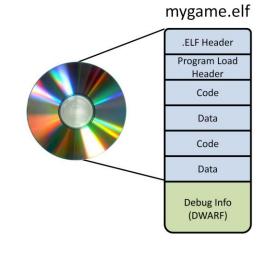
Windows uses the Program DataBase format. The symbols go into separate PDB files that are generated alongside the main executable dlls. One consequence of this is that the "release" configuration doesn't omit symbols from the executable – you always have symbols available for anything you build, so long as you keep the PDBs around. The difficulty of optimizing release Windows builds is mostly due to optimizer rearrangement.

VALVE

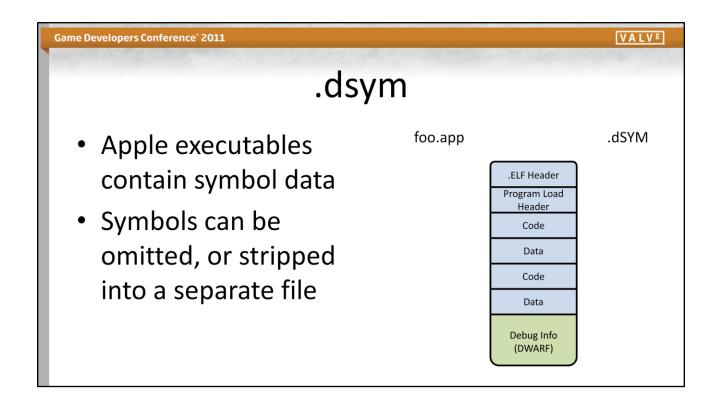
#### VALVE

### **DWARF** format

- GCC derivatives (Linux & PS3)
- Symbol data is part of the .elf
- Debug data "stripped" from retail elves



http://dwarfstd.org/



On Mac, symbol info also gets cooked into the executable, but dsymutil lets you strip it out into its own file. Also, most released executables have some light symbol info, enough to get the function names for a call stack, though not source line numbers.

#### VALVE

# Making a Symbol Server

- Changing the binary changes symbols
- $\forall$  crash, you need the exact same version of:
  - Exe that made it
  - Symbols
  - Source code
- And a way to figure out where to get the info corresponding to a given dump!

# Making a Symbol Server (PC/360)

- .exe/.dlls get signatures from timestamp&size
- which identify PDBs for a given source .exe
- A PDB symbol server is just an ordinary Windows fileshare
- Use symstore.exe to make one

Windows/360

MSFT documentation for symbol store: http://msdn.microsoft.com/en-us/library/ms681417(VS.85).aspx

		SALES STREET, SALES
Matching Dun	nps with Symbols (MS	VC)
Fonts and Colors     Fonts and Colors     Felp     Import and Export Settings     International Settings     Keyboard     Startup     Task List     Web Browser     Projects and Solutions     Source Control     Test Editor     Database Tools     Debugging     General     Edit and Continue     Just-In-Time     Native     Symbols	Symbols Using symbol files from an unknown or untrusted location can be harmful to your computer. Symbol file (.pdb) locations:  Verforce\symbols Cache symbols from symbol servers to this directory: U\SYMBOLSRV Browse Search the above locations only when symbols are loaded manually Load symbols using the updated settings when this dialog is closed  K Cancel	

MSVC can connect a dump with its symbols so long as you tell it where to look for the symbol database.

VALVE

# Making a Symbol Server (PS3)

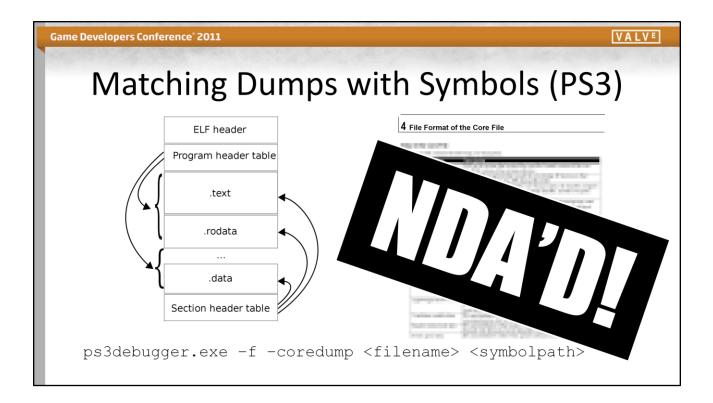
- You have to roll your own. Here's how we did.
- Recent compiler versions can cook a GUID into the .elf/prx by hashing the image:

Doing the same on the PS3 is going to require a little more manual effort. The compiler and linker can cook a GUID into each elf and prx by hashing the binary image (which means the GUID will be the same if you rebuild the image from the same source). You can extract that GUID from the binary with something like ps3bin.exe, and then store the .self, which contains the symbols, in some kind of database.

I find a simple filesystem works pretty well as a database – just put each file under a directory named after the GUID.

Game Developers	Conference <sup>®</sup> 2011		VALVE
ф <b>њ њ</b>	<b>U</b> 1	Options	Symbols (PS3)
· 举 PPU Threads (30)	00, Total Memory Usage = 0x0B21D000 (178.1 MB)	Environment Keyboard	Search Directories
		Display AutoComplete	Search path: PRX Files
E Light Weight Mute	exes (145)	Clipboard	Directories E+ C> A+ C* X >> 1
<ul> <li>⊕ €2. Condition Variable</li> <li>⊕ €2. Light Weight Conce</li> <li>⊕ €4. Reader Writer Lock</li> <li>⊕ €4. Event Queues (11)</li> <li>⊕ ∑6. Modules (48), Men</li> <li>⊕ ⊕ Menory Containes</li> <li>⊕ ⊕ Kennery Containes</li> <li>⊕ ⊕ Kennery Containes</li> </ul>	lition Variables (14) s (6) n Size = 46.9 MB (.text = 0 Bytes, .data = 46.9 MB)	History Source VSI Fonts and Colors Project S Project Options Path Mappings Search Directories Stepping Control	$\int$
Attribute	Value	PS3 PPU Debugging	
Filename	/dev_bdvd/PS3_GAME/USRDIR/EBOOT.BIN	SPU Debugging	
GUID	0000000-0000000-0000000-0000000-0000000	Reset Parameters	
Local Memory	0x07BE0000 (123.9 MB)		Adding subdirectories Max. folder depth 5 A Max entries per level 255
Module Data Size	0x00B50000 (11.3 MB)		Max. folder depth 5 👘 Max entries per level 255 👘
Module Text Size	0x026B0000 (38.7 MB)		
Other	0x0037D000 (3.5 MB)		
Shared Memory	0x00020000 (128 KB)		
Text Size	0x000A0000 (640 KB) 0x0B21D000 (178.1 MB)		
Total Memory Usage			

Once again, it's a little harder with Sony. A crash dump will contain the GUID of the executable that emitted it, but getting that info is tricky. You can load your dump in the Target Manager or the debugger and then look at the Process section of the kernel pane; that will contain the GUID, which you can then use to find wherever you put your symbol files on a server.



The alternative is to use the fact that dump files are also .elfs and Sony's documentation of the core file format to go digging through the dump for the GUID info. Once you have the GUID, again you use that to find your symbols in your server and launch the debugger appropriately.

#### VALVE



As mentioned before, it's easiest if all of your minidump files end up in the same place automatically. This isn't hard; you can either configure your kits to all write their dump files to a shared file server, or you can have each PC and kit emit the dumps to their local hard drives, and then you write some scheduled task or other robot that sweeps them over to your central location. You can get fancy with databases if you like.

VALVE

### Automated Dump Triage

Preliminary, basic info without having to open a debugger

- ie, Call stack, exception type, registers in a .txt file

- Sorting and grouping crashes by cause
- Overview statistics to set priority and spot trends

#### —— See also…

**"Debugging in the (Very) Large: Ten Years of Implementation and Experience"** Microsoft: Kirk Glerum, Kinshuman Kinshumann, Steve Greenberg, Gabriel Aul, Vince Orgovan, Greg Nichols, David Grant, Gretchen Loihle, and Galen Hunt

What works for one dump doesn't work for one hundred – as the incoming storm scales up, you won't have time to open each dump individually in the debugger. Automating part of the process will free up more time to investigate each issue, and psychologically, the easier it is to deal with a stability problem, the more likely a programmer is to deal with it.

One good place to start is to rig a script to automatically get some basic information on every dump, like exception type and a call stack, and put that into a .txt file next to the dump in your repository; thus you can get basic info on each dump in five seconds instead of waiting for ProDG to spool up.

These are individual steps you can take incrementally as your influx scales up – the most important is the at-a-glance readout, then bucketing, etc.

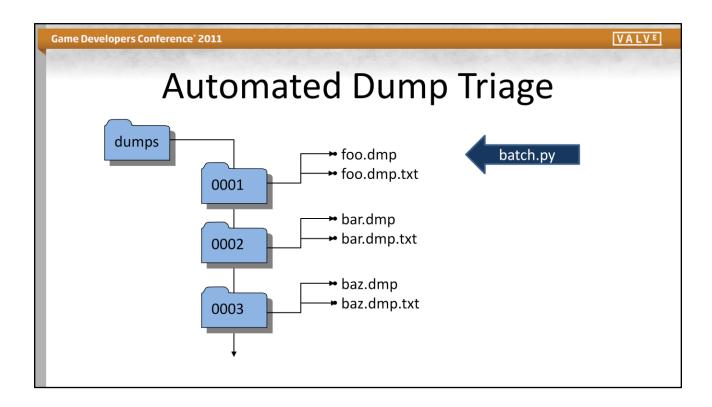
VALVE

# Automated Dump Triage

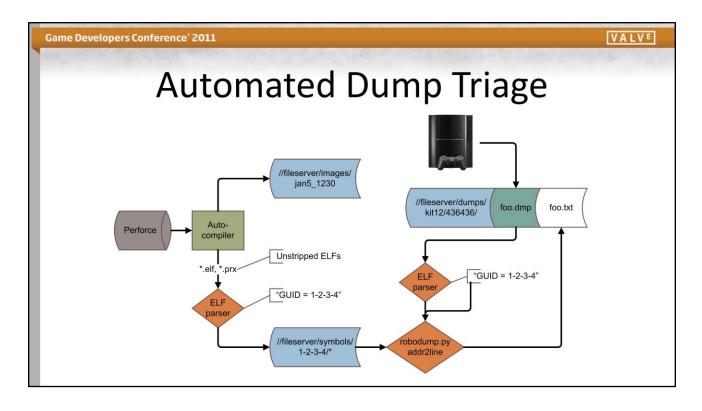
• MSVC (Windows/360): WinDbg/kd (command-line debuggers)

.sympath+ srv*\\myserver\symbols	add symbol path
!reload	reload symbols
ecxr	print exception record
r	print registers
 kb	print call stack
lm lv	print loaded modules
P	quit

Windbg for robo-cracking on 360/windows



Then you can make a scheduled batch file out of this and it can just march through your dump repository generating its report for each one in turn.



On the PS3 again we had to build our own tools. Let's go through the whole chain. Our autobuilder pulls a changelist from perforce and starts building an image from it. Once the executables have been compiled, a script parses through the .ELF for the GUID the linker cooked in there, and then creates a directory on a central fileserver named with that guid. It copies the entire .elf and .prx tree (because the executables *are* the symbols in the .ELF format) onto the fileserver. Later, when a test kit emits a dump, we copy that onto the fileserver as well. We use the ELF parser on the dump file to find the GUID, which gives us the matching directory on the fileserver. Once we have that, I manually walk the stack found in the dump file, correlate each address on the stack against the symbols to come up with a function name, source file, and line number, then emit an exception report onto the fileserver adjacent to the dump.

#### VALVE

### **Evolve Psychic Powers**



analyze all.py > todaysdumps.txt

outlook.exe /c ipm.note
 /m myself@example.com /a todaysdumps.txt

This is Arthur Fellig, a famous news photographer of the early 20<sup>th</sup> century. He was better known under the name Weegee (Oujia), because of his preternatural ability to show up at crime scenes before the police did. (He had one of the first police radio scanners.)

So here is my Magic Batch File that emailed me the stack trace for every 360 crash during Left4Dead QA in real time

hint: you can use this to pretend you are psychic and know about bugs before they are submitted

# Always Do The Autopsy

- Don't assume causes from symptoms
- At least eyeball the call stacks.

When in doubt about a crash, do the autopsy. You might find a surprising cause, or reveal a core bug that hasn't shown elsewhere.

VALVE



### A THOUSAND POINTS OF DATA

Bugs Don't Stop After Shipping

### Gathering Stability Data From Customers

- You can't catch everything in QA
  - $-e^n$  user configurations out there
  - 3<sup>d</sup> parties break your game
- Customers = 1,000,000 testers
- Built into OS:
  - Windows: WinQual, emailing .mdmps
  - Mac: Crash Report dialog, "please copy paste"

You can't catch every issue in testing, especially on PC

umpty-bazillion possible end user configurations means someone out there is going to hit an edge case you didn't think of

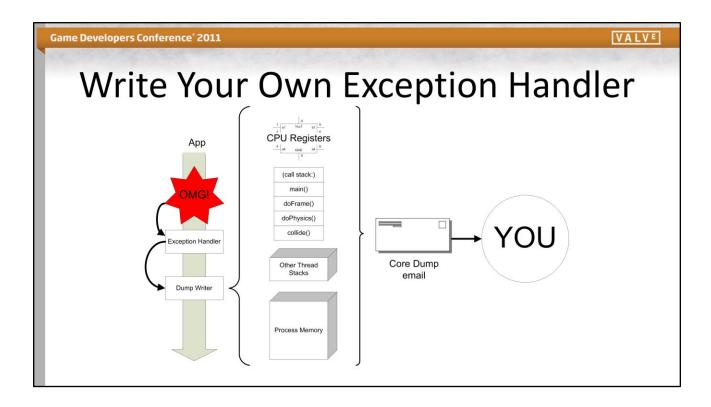
third party software updates can break your game (ie graphics card drivers) via incompatibility

By accumulating data from your customers, you get millions of testers for free and can quickly roll that into updates.

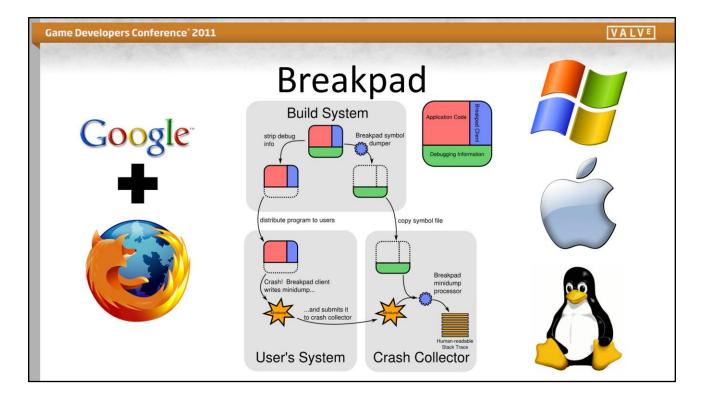
Built in ways:

Windows: Winqual. having the customer email you a .mdmp. Mac: the Crash Report dialog, and asking customers to copy-and-paste the stack from it. That doesn't sound like a good time.

VALVE



Replacing the default exception handler lets you write your game so that it does smart things with crashes, like emailing them to you.



How Valve uses Breakpad to send us data along the steam-tubes (with screenshots each step of the way)

Breakpad is a joint Google/Mozilla effort to create a multiplatform crash-reporting system. As used in Firefox.

We use it on all of our platforms, including our Linux dedicated game servers.

This is Mozilla's public stability site. Anyone can see every crash that Firefox has suffered recently, its call stack, many details. Moz uses this to collect stability data from every Firefox in the world.



In particular, this helps Mozilla track down third party apps that suddenly cause problems for a disproportionate number of their users – because they have data on the actual call stack, and on all the modules loaded at the time, they can immediately pinpoint exactly what caused the crash and take action.

## **Breakpad Files**

### Dumps

- Microsoft .mdmp format
- Contain:
  - Exception cause
  - List of running threads
  - CPU registers and stack data
  - List of loaded .DLLs
  - Processor type, OS version
  - Comment field

### Symbol files

- Breakpad's own (well documented) format
- Processed from PDB/.ELF/DWARF/STABS
- Build your own symbol server

The minidump file format is similar to core files but was developed by Microsoft for its crash-uploading facility. A minidump file contains:

A list of the executable and shared libraries that were loaded in the process at the time the dump was created. This list includes both file names and identifiers for the particular versions of those files that were loaded.

A list of threads present in the process. For each thread, the minidump includes the state of the processor registers, and the contents of the threads' stack memory. These data are uninterpreted byte streams, as the Breakpad client generally has no debugging information available to produce function names or line numbers, or even identify stack frame boundaries.

Other information about the system on which the dump was collected: processor and operating system versions, the reason for the dump, and so on.

VALVE

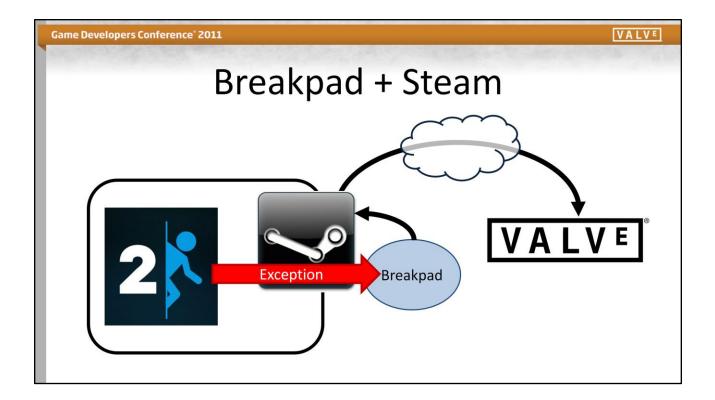
# **Breakpad Client Libraries**

- Exception handler
  - (OS exception, not try/catch)
  - Installed at app launch
- Stackwalker
  - Runs in own thread (ideally process)
- Dump emitter
- HTTP uploader (optional)

Here is how we integrate the Breakpad client libraries into our games that we ship to customers.

The client library consists of an exception handler – an OS-level structured exception handler on Windows, not a try/catch block – and signal handler on Darwin and Linux.

http://code.google.com/p/google-breakpad/wiki/ClientDesign



In the case of our games, since they run inside the Steam process, what actually happens is the game's exception percolates up to Steam, which has Breakpad installed. That exception then goes back to Steam, which uploads it to us!

Game Developers Conference <sup>®</sup> 2011	VALVE
Pro	Crash
	Database
Lots And Lots of Dumps	Upload Server (Apache HTTP) Breakpad Processor
Ŭ	Mozilla Socorro Symbol Server

The dumps all come in (at a very high rate) to a simple Apache server, which just accepts the uploads and queues them up. They are fed at a slower pace to our processor machine, which calls the Breakpad processor encapsulated as a MinidumpProcessor C++ class. Using the filename and timestamp of the modules the user was running, it finds the relevant symbol files on our server (the big flat filesystem again) and produces a call stack, which then gets uploaded to a crash database. This whole backend is actually another open source project from Mozilla called Socorro, which is what they use for their stability website.

ne Develope	ers Conference <sup>®</sup> 2011	ashDB - S	ocorro	VALVE
	mozilla crash r	reports		C Find Crash ID or Signature
	Product: Left 4 Dead 2 (550)	Current Versions	Report: Overview	Advanced Searc
	Product         Counter-Strike: Source (240)         Counter-Strike: Source Beta (260)         Day of Defeat: Source (300)         Dot A 2 Beta (570)         Garry's Mod (4000)         Half-Life 2 (220)         Half-Life 2: Deathmatch (320)         Half-Life 2: Deathmatch (320)         Half-Life 2: Episode One (380)         Half-Life 2: Episode Two (420)         Left 4 Dead (500)         Left 4 Dead 2 (550)         Left 4 Dead 2 Codicated Server (560)         Left 4 Dead Dedicated Server (510)         Portal (400)	Version: AI 550 4391 550 4388 550 4358 550 4346	Coperating System Windows Mac OS X Linux Solaris	Advanced Filters     Filter Crash Reports

Storing hundreds of thousands of customer crash reports in a database. This is what our database looks like. As you can see, we're just using Mozilla's back end, called Socorro.

	uery Results					
Results v	- vithin 1 weeks of 02/16/2011 02:05:03, and the product is one of 550 and the crashing process was a any.					
Rank 🜩	Signature ¢	# \$	Win 🗢	Mac \$	Lin ¢	S
1	CAudioDirectSound::ClearBuffer()	6165	6165	0	0	0
2	(empty signature) Learn More	1962	0	1962	0	0
3	DSP_ClearState	1886	1886	0	0	0
4		1020	1020	0	0	0
5		752	752	0	0	0
6		718	718	0	0	0
7	CShaderAPIDx8::TexLock(int, int, int, int, int, int, CPixelWriter&)	459	459	0	0	C
8	Error [SV_InitGameDLL()	444	444	0	0	0
9		380	380	0	0	0
10	CShaderDeviceDx8::CreateStaticMesh(unsignedint64, char const*, IMaterial*, VertexStreamSpec_t*)	312	312	0	0	0
11	CMatCallQueue::CallQueued()	281	281	0	0	0
12	Error   UTIL SetModel(CBaseEntity*, char const*)	279	279	0	0	0
13		256	256	0	0	0
14		247	247	0	0	0
15	CMeshDX8::Lock(int, bool, VertexDesc_t8)	237	237	0	0	0
16	CMaterial::GetColorModulation(float*, float*)	220	220	0	0	0
17	DecalSurfaceAdd(msurface2_t*, int)	217	217	0	0	0
18	CMeshDX8::SetPrimitiveType(MaterialPrimitiveType_t)	209	209	0	0	0
19	CParticleCollection::LabelTextureUsage()	168	168	0	0	0
20	Sys_Error   Host_ParseConfiguration	161	0	161	0	0
21	ZombieManager::StartFrame	147	0	0	147	0
22		141	141	0	0	0

П

Here's what you get if you click go; you can see here the list of crashes and their causes by count. Notice also that we have some crashes that occur only on Mac and another that's only on Linux.

		NOW SIG C	dual-axis	1-01-18 20:00:00 through 2011-02-15 20:00:00. The report covers 93.90% of all 538 , having Count (Number of Crashes) on the left X axis and Percent of total of Crashes				
				Oth	er Perio	ods:		
				<u>3 Da</u>	ays 7 Days 14 Days			
Rank	Trend	% \$	Diff ≎	Signature \$	Count ¢	Win 🗢	Mac 🜩	Lin \$
1	new	33.40%	0.00%	CAudioDirectSound::ClearBuffer()	1797	1797	0	0
2	new	8.60%	0.00%	(empty signature) Learn More	463	0	463	0
3	new	2.88%	0.00%		155	155	0	0
4	new	2.86%	0.00%		154	154	0	0
5	new	2.27%	0.00%	DSP_ClearState	122	122	0	0
6	new	2.10%	0.00%	CShaderAPIDx8::TexLock(int, int, int, int, int, int, CPixelWriter8)	113	113	0	0
7	new	1.62%	0.00%		87	87	0	0
8	new	1.49%	0.00%	Error [ SV_InitGameDLL()	80	80	0	0
9	new	1.36%	0.00%	CMatCallQueue::CallQueued()	73	73	0	0
10	new	1.34%	0.00%	CShaderDeviceDx8::CreateStaticMesh(unsignedint64, char const*, Material*, VertexStreat	72	72	0	0
11	new	1.30%	0.00%		70	70	0	0

opers (	onfere	nce' 20	11								
Vindo	ws N	Г - Cra	ash Rep	orts for CN	later	ial::GetColorModula	ation(fl	oat*,	float	*, float*	)
Results with	n 2 weeks o	of 02/16/201	1 02:54:35, and t	he product is one of 5	50 and th	e crashing process was a any.					
Graph	Table	Reports	Comments (0)	Correlations							
										501 Crash Re	эро
Date 🗢	Product \$	Version \$	Build \$	os 🜩	CPU ¢	Reason \$	Address 🗢	Hang \$	Uptime 🜩	Comments \$	
Feb 16, 2011 02:43	550	4448	20110111175313	Windows NT 5.1.2600 Service Pack 3	x86	EXCEPTION_ACCESS_VIOLATION_READ	0x10		0		
Feb 16. 2011 01:23	550	4448	20110111175313	Windows NT 6.1.7600	x86	EXCEPTION_ACCESS_VIOLATION_READ	0x10		0		
Feb 15, 2011 14:17	550	4448	20110111175313	Windows NT 6.1.7600	x86	EXCEPTION_ACCESS_VIOLATION_READ	0x10		0		
Feb 15, 2011 13:38	550	4448	20110111175313	Windows NT 6.1.7600	x86	EXCEPTION_ACCESS_VIOLATION_READ	0x10		0		
Feb 15, 2011 13:24	550	4448	20110111175313	Windows NT 6.1.7600	x86	EXCEPTION_ACCESS_VIOLATION_READ	0x10		0		1
Feb 15. 2011 13:06	550	4448	20110111175313	Windows NT 6.1.7600	x86	EXCEPTION_ACCESS_VIOLATION_READ	0x10		0		1
Feb 15. 2011 12:45	550	4448	20110111175313	Windows NT 6.1.7601 Service Pack 1, v.178	x86	EXCEPTION_ACCESS_VIOLATION_READ	0x10		0		
Feb 15. 2011 12:24	550	4448	20110111175313	Windows NT 5.1.2600 Service Pack 3	x86	EXCEPTION_ACCESS_VIOLATION_READ	0x10		0		1

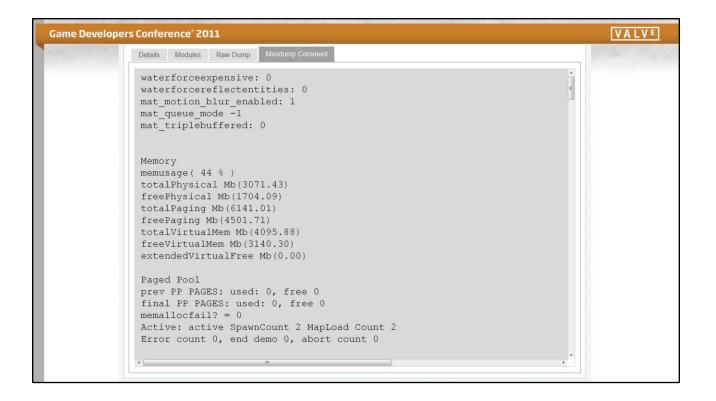
I can drill into a single crash cause, which is all the dumps that share the same call stack. Here you can see all those dumps.

	ec2d6d ure: CMaterial: GetCc	lorModulation(float*, float*, float*)	Search Mozilla Support for Help	
Del		Raw Dump Minidump Comment		
	Signature CMater	ial::GetColorModulation(float*, float*, float	(*)	
	UUID 92ec2d	6d 10 10 10 10 10 10 10 10 10		
	Time 2011-0	2-15 13:38:41.885219		
	Uptime 0			
	Product 550			
	Version 4448			
	Build ID 201101	11175313		
	Branch steam			
	OS Window			
	OS Version 6.1.760	0		
	CPU x86			
		ticAMD family 15 model 75 stepping 2		
C	rash Reason EXCEP	TION_ACCESS_VIOLATION_READ		
Cras	shing Thread			
Fra	me Module	Signature [Expand]	Source	
0	materialsystem.dll	CMaterial::GetColorModulation	//valvegames/rel/left4dead2/src/materialsystem/cmaterial.cpp:2068	
1	shaderapidx9.dll	CBaseMeshDX8::DrawMesh	//valvegames/rel/left4dead2/src/materialsystem/shaderapidx9/meshdx8.cpp:2372	
2	shaderapidx9.dll	CDynamicMeshDX8: DrawInternal	//valvegames/rel/left4dead2/src/materialsystem/shaderapidx9/meshdx8.cpp:4240	
3	shaderapidx9.dll	CDynamicMeshDX8::Draw	//valvegames/rel/left4dead2/src/materialsystem/shaderapidx9/meshdx8.cpp:4253	
4	vguimatsurface.dl	CMatSystemSurface::DrawQuad	//valvegames/rel/left4dead2/src/vguimatsurface/matsystemsurface.cpp:1109	
5	vguimatsurface.dl	CMatSystemSurface::DrawTexturedRec	t //valvegames/rel/left4dead2/src/vguimatsurface/matsystemsurface.cpp:1652	
6	gameui.dll	gameui.dll@0x564fe		

Drill into an individual dump and you can see data on the machine that suffered it, and the whole call stack.

ime Develop	ers Conference <sup>®</sup> 2011				VALV
	Details Modules	Raw Dump Minidu	Imp Comment		
	Filename	Version	Debug Identifier	Debug Filename	
	launcher.dll		F842B628C67B404E9A448A8DC94202073	launcher.pdb	
	tier0.dll		7F9BD272AB8249A991C895508D58C18B1	tier0.pdb	
	filesystem_stdio.dll		F7CFC5A5F29E43819E9788FD2BE1B2863	filesystem_stdio.pdb	
	left4dead2.exe		C2DACBA6EF724787AFFF9DDBEB727BE81	default.pdb	
	vstdlib.dll		443E08D121F14C58B969A57F378831EA2	vstdlib.pdb	
	inputsystem.dll		E68D75B6E7064D4FA3D77C23335189F44	inputsystem.pdb	
	datacache.dll		34C950851BA646F5B1D4CF7FE2F986673	datacache.pdb	
	engine.dll		455D1F2174184D1FB17C263C9204A4E73	engine.pdb	
	materialsystem.dll		2E5AAF69064745ADA379473B108528DF3	materialsystem.pdb	
	studiorender.dll		94E8C64AF7E24FBF9181081652A657313	studiorender.pdb	
	vphysics.dll		2C3C1E15B4AE49239CCBE6FADEE2C4AF3	vphysics.pdb	
	vscript.dll		723763006A594ED09C63EA02B0EDF8883	vscript.pdb	
	valve_avi.dll		983E989E5F114996921E60EF5C953DC43	valve_avi.pdb	
	XInput1_3.dll	9.15.779.0	3482C2EF26164FFC863CC4E4BBA3210A1	XInput1_3.pdb	
	vguimatsurface.dll		1D278BADB24842469B3DB9918D29BDF03	vguimatsurface.pdb	
	vgui2.dll		49D1D62458344D40B4150FA4C92DF4EC3	vgui2.pdb	
	crashhandler.dll	1.1.0.43	6E7B1B193D5843CB8B48D9D1CDE1D7AD1	crashhandler.pdb	
	shaderapidx9.dll		F8848616C14940559282FCDC76073B323	shaderapidx9.pdb	
	stdshader_dbg.dll		6BFEA17519784E80A27DD76B7B67D70B3	stdshader_dbg.pdb	
	unicode.dll		8CE01DA3B5E14340841C651AA1977DB03	unicode.pdb	
	stdshader_dx9.dll		CC80C7E5CC5647A59E55635DBF8C31303	stdshader_dx9.pdb	
	clbcatq.dll	2001.12.8530.16385	00A720C79BAC402295B6EBDC147257182	CLBCatQ.pdb	
	msssrs.flt	7.2.6.0	97F4F28931D94708BDD5ED1C2D51D3751	msssrs.pdb	
	vaudio miles.dll		3528962A7424452F94553269012213C63	vaudio miles.pdb	

You can see all the DLLs that were loaded and their version.



And, if you remember that I mentioned earlier that you can add any blob of "comment" data that you like to a minidump, we use ours to store useful telemetry, like the OS configuration the user had, how much free memory, and so on.

	Olvic			at*, float*, float*) ]				
	ID: 92e Signatu	Search Mozilla Support for Help						
	Deta	IIs Modules	Raw Dump Minidump Comment					
		Signature CMater	ial::GetColorModulation(float*, float*, float	")				
		UUID 92ec2d	6d 10 10 10 10 10 10 10 10 10					
		Time 2011-0	2-15 13:38:41.885219					
		Uptime 0						
		Product 550						
		Version 4448						
		Build ID 201101	11175313					
		Branch steam						
		OS Window						
		OS Version 6.1.760 CPU x86	0					
		CPU x86 CPU Info Authen						
	Cr							
		Crash Reason EXCEPTION_ACCESS_VIOLATION_READ						
	Crast	ing Thread						
	Fram	e Module	Signature [Expand]	Source				
	0		CMaterial::GetColorModulation	//valvegames/rel/left4dead2/src/materialsystem/cmaterial.cpp:2068				
	1		CBaseMeshDX8::DrawMesh	//valvegames/rel/left4dead2/src/materialsystem/shaderapidx9/meshdx8.cpp:2372				
	2	shaderapidx9.dll	CDynamicMeshDX8::DrawInternal	//valvegames/rel/left4dead2/src/materialsystem/shaderapidx9/meshdx8.cpp:4240				
	3		CDynamicMeshDX8::Draw	//valvegames/rel/left4dead2/src/materialsystem/shaderapidx9/meshdx8.cpp.4253				
	4		CMatSystemSurface::DrawQuad	//valvegames/rel/left4dead2/src/vguimatsurface/matsystemsurface.cpp:1109				
	5			//valvegames/rel/left4dead2/src/vguimatsurface/matsystemsurface.cpp:1652				
	6	gameui.dll	gameui.dll@0x564fe					

So let's see what caused this particular issue.

```
VALVE
```

```
2063.
       void CMaterial::GetColorModulation( float *r, float *g, float *b )
2064. {
2065.
               PrecacheVars();
2066.
2067.
              float pColor[3];
2068.
              m_pShaderParams[COLOR]->GetVecValue( pColor, 3 );
2069.
              *r = pColor[0];
2070.
               *g = pColor[1];
2071.
               *b = pColor[2];
2072. }
2073.
```

In this case it looks like the issue is due to missing shader parameters. So we ought to build a machine with that same configuration of graphics card and video driver and see why this shader isn't initializing properly.

## In Conclusion

- Crashes suck.
- Fixing them doesn't have to.
- When everyone else lies, the disassembler is there for you.
- These skills are for debugging live release builds too.
- "The debugger should" ≠ "The debugger **does**".

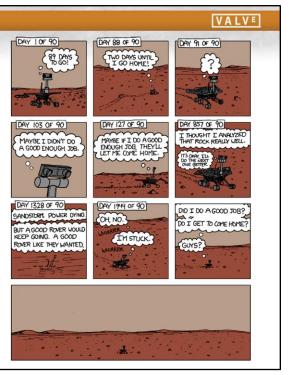
VALVE

### VALVE

## **GO FORTH AND CRASH NO MORE!**

### slides at: bit.ly/hPCmVW

## **Questions?**



XKCD by Randall Munroe

#### VALVE

# Bibliography / Cheat Sheet

- Pietrek, Matt. Just Enough Assembly Language To Get By, I & II. Microsoft Systems Journal, Feb-June 1998. <u>bit.ly/9HKCOk bit.ly/bJf1R0</u>
- (Microsoft) Kirk Glerum, Kinshuman Kinshumann, Steve Greenberg, Gabriel Aul, Vince Orgovan, Greg Nichols, David Grant, Gretchen Loihle, and Galen Hunt. Debugging in the (Very) Large: Ten Years of Implementation and Experience. bit.lv/4lBgkH
- (Jet Propulsion Laboratory) Reeves, Glenn; Neilson, Tracy. *The Mars Rover* Spirit FLASH Anomaly. <u>http://hdl.handle.net/2014/39361</u>
- My blog: Some Assembly Required. <u>http://assemblyrequired.crashworks.org</u> slides for this talk: <u>bit.ly/hPCmVW</u>

#### VALVE

## Bibliography / Cheat Sheet

- IBM. PowerPC Microprocessor Family: Programming Environments Manual for 64 and 32-Bit Microprocessors. <u>http://bit.ly/gHemWI</u>
- Intel. Intel 64 and IA-32 Architectures Software Developer's Manual. <u>http://www.intel.com/products/processor/manuals/</u>
- SunSoft; IBM. SYSTEM V APPLICATION BINARY INTERFACE PowerPC Processor Supplement. http://bit.ly/fRvDe9
- Apple. Introduction to Mac OS X ABI Function Call Guide. http://bit.ly/foRCnE
- Motorola, Inc. PowerPC Embedded Application Binary Interface. http://bit.ly/i9G0yz
- IBM. Developing PowerPC Embedded Application Binary Interface (EABI) Compliant Programs. <u>http://bit.ly/hbhlyk</u>
- Taylor, Ian Lance (Zembu Labs). 64-bit PowerPC ELF Application Binary Interface Supplement. <u>http://bit.ly/hoWPfF</u>
- ELF-64 Object File Format. http://bit.ly/fUel3t
- Wikipedia. Executable and Linkable Format. http://en.wikipedia.org/wiki/Executable and Linkable Format

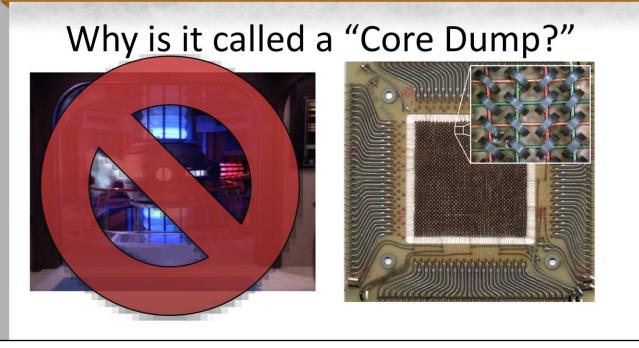
#### VALVE

# Bibliography / Cheat Sheet

- Eager, Michael. Introduction to the DWARF Debugging Format. http://bit.ly/hLBoF2
- Apple. *Debugging and Symbolizing Crash Dumps in Xcode*. <u>http://bit.ly/C2RqC</u>
- Microsoft. Symbol Server and Symbol Stores. <u>http://bit.ly/ecSyzj</u>
- Microsoft. Debugging Tools for Windows. <u>http://bit.ly/eXwcf2</u>
- WinDbg command cheat sheet: <u>http://bit.ly/3Jc0p7</u>
- Sen, Saikat. A WinDbg Tutorial. http://bit.ly/4h4PZx
- Google Breakpad project home page: <u>http://bit.ly/7FfWzQ</u>
- <u>http://www.dumpanalysis.org/</u>



### VALVE



By the way, did you ever wonder why it is called a core dump? I thought it was because the people who invented Unix were big nerds and made some kind of Star Trek reference, but not so. In fact it's because before computers used capacitors for their RAM, they used ferrite cores – little donuts of iron – to store bits magnetically. So, memory was literally called core, and if the computer crashed, it would dump the entire contents of core memory to disk, or the punchcard writer. Hence, core dump.



And if you think that's bad, when the 1951 Whirlwind I computer crashed, it would output the entire core memory to a CRT, in octal. Then an automated camera would take a picture of the CRT on microfilm, which would be developed and sent over to the poor sap debugging it.

### Socorro

- Mozilla's back-end for stability data
- Socorro Server
  - Collects/databases/processes dumps
- Socorro UI
  - The UI you just saw
- Python, Postgres, Hadoop, ...
- You could write your own DB/UI too.

http://code.google.com/p/socorro/

VALVE

A web front-end to your crash database: Socorro.

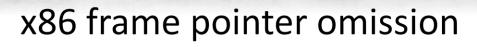
Another open-source tool from Mozilla.

There are pros and cons to using it; you could also write your own front end.

we've made minor modifications to the socorro code - it's PHP + Postgres. We've added things like steam universe so we can filter on it. we're \*NOT\* using the hadoop crap - we tried, and gave up - their current trunk version is overly complex to setup and administer. we backed off to the last "stable" version where they used NFS for dump transmission and didn't rely on hadoop, and the system has been maintenance free (knock on wood) since.

http://code.google.com/p/socorro/wiki/SocorroOverview

VALVE



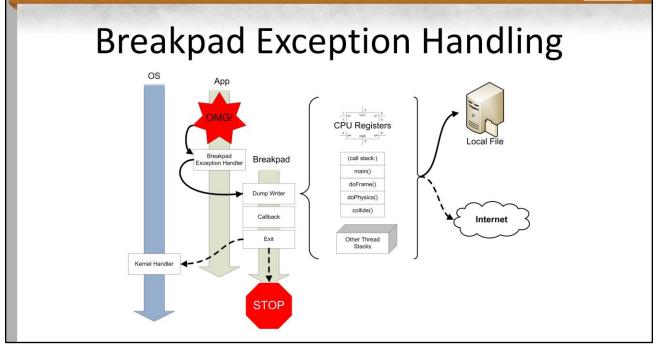
address				void foo()
0x10000	caller EBP	ebp	FDD	int a,b;
0xfffc	int a	esp+42	EBP	float f,g;
0xfff8	int b	esp+38	•	uint64 u;
0xfff4	float f	esp+34		m128 vec;
0xfff0	float g	esp+30		}
0xffec				
0xffe8	uint64 u	esp+24		
0xffe4				mov eax, [ebp+42]; a
0xffe0				add eax, [ebp+38] ; a+b
0xffdc				fld [ebp+32] ; f
0xffd8				fld [ebp+36] ; g
0xffd4				fsub ; f - g
0xffd0	m128 vec	esp+0	ESP	movaps xmm0, [ebp+48];;sseeveclbadd

No nonvolatile registers on x86, but debugger is good about moving the stack pointer around properly. If you see that the EBP pointer here is redundant, you're right: enabling "Frame Pointer Omission" in the compiler does away with it, using just the ESP pointer instead. The trouble with that is you lose your backchain – the compiler has to know how much to move the stack pointer when returning from each function, and the debugger doesn't always have that info – and it complicates finding data since the ESP pointer tends to move about spastically as a consequence of push/pop.

Game Developers Conference <sup>®</sup> 2011	VALVE
Overwritten LR on stack	
Call Stack Name Name No.000000000) Memory1 × Find Results 2 Threads Autos Modules Address: 0x7005EBF0 Unable to evaluate the expression.	
⇒         0x00008600         0x00         ■ Main Thread         MainThrd         >         00000000           0x0000959C         0x00         ■ Worker Thread         IOJob0         ✓         CThreadSync(	Normal Normal

A quirk of MSVC is that when your stack frame thinks it's at address 0x00, it may fail to give you a memory window altogether. The solution is simply to switch to a different thread that still has a valid context, and then memory will come back.

### VALVE



#### VALVE

### **Photo Credits**

NIH / National Library Of Medicine, Visible Proofs exhibit Wikimedia Commons public domain archive Library Of Congress Photo Archive United States Air Force NASA

Chalk outline: Flickr user rbeiber <u>http://www.flickr.com/photos/rbieber/155102957/</u> Lying rat: Flickr user niznoz <u>http://www.flickr.com/photos/niznoz/4233333</u> Memphis sewer: Flickr user mojorider2 / Paul Everett <u>http://www.flickr.com/photos/mojorider2/4670627480/</u>

In 2136(au) In 2136(au) Date United States Volunteers attacked by the mole, corner of Fifth and Walnut Streets, St. Lusis, Missouri / Jestched by M. Hastings, Exp. Date Constraints, States Finge Corned of page 18 stacking them. States (St. States) FAD, Tach, Accidentings Description Digits On D March 1940U S. Navy LL John Smith "Jimmy" Tlach tipped this Brewster FAL 1 Buffalo (Bulko 1333) onto its nose on the flight deck of the circaft carrier USS Saratoga (CV-3) Ensign Edward Buch O'Hare also flew this aircraft several times during the summer and fail of 1940 Digits On D March 1940U S. Navy LL John Smith "Jimmy" Tlach tipped this Brewster FAL 1 Buffalo (Bulko 1333) onto its nose on the flight deck of the circaft carrier USS Saratoga (CV-3) Ensign Edward Buch O'Hare also flew this aircraft several times during the summer and fail of 1940 rch 1940 19 March 1940 Source broadcast.illuminatedtech.comTransferred from en. Author USN; Original uploader was Felix c at en.wikipedia, 25 June 2006 (original upload date) Permission Permission (Reusing this file) PD-USGOV-MILITARY-NAVY. chalk\_outline.jpg Flickr7c-nc user /belber http://www.flickr.com/photos/rbieber/155102957/ nih\_nyc\_crime\_scene.jpg NIH web site NIF web site New York City crime scene, 1914-1918 New York City Municipal Archives http://www.imm.nih.gov/visibleproofs/exhibition/views.html af\_c141\_ramp\_crash USAir Förce http://www.af.mil/photos/media\_search.asp?q=crash&page=14 te\_core\_memory pedia public domain scription JBIN: Random access ferrite core memory (RAM) from 1961. Size of the card: 10.8cm x 10.8cm (6.5 inch), capacity: 1024 bits In: 5 June 2009 roce: Combined from Magnetic core memory card.jpg and Magnetic core.jpg. reference processon grant trans of the set of the set of the set of the corps should be extended and the chin staaded with the left hand. Grasping the post-montem lenfe firmly in the pairs of the right hand and cuting with the bely, not the point, of the kinfe, make a median incluion from the chin to the p the corps of the set of the set of the corps should be extended and the chin staaded with the left hand. Grasping the post-montem lenfe firmly in the pairs of the right hand and cuting with the bely, not the point, of the kinfe, make a median incluion from the chin to the p left behavior of the corps. The set of the corps of the set of the corps should be extended and the chin staaded with the left hand. Grasping the post-montem lenfe firmly in the pairs of the right hand and cuting with the bely, not the point, of the kinfe, make a median incluion from the chin to the p left behavior of the corps of the left behavior. Technique, London the point of the right hand and cuting with the bely. The point, of the kinfe, make a median incluion from the chin to the p left behavior. The point of the point of the corps o lying\_rat flickr user niznoz http://www.flickr.com/photos/niznoz/4233333 Euklid Justus of Ghent, 1474 whirlwind\_control\_room Museum Of Science, Boston, MA credit: Wikinedia 886?-Project-Whirlwind credit: Wikipedia KL\_microsoft\_xbox\_360 credit: Wikipedia forensic anthropology liopia, 1990s. Agg, TBG<sup>2</sup>, TBG<sup>2</sup> New Arran value of the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested the solid around the caskets, and every type of embalming fluid in ~ And receive tested ted positive for Pavulon. The proof was definitive-h micide had taken place. Saldivar had poi f. Annitesen teatrany of Medicine attional Library of Anthony Pidgeon redit: Courtesy of Anthony Pidgeon \*\*\*\*//www.nlm.nih.gov/news/press\_releases/visibleproofphotos.html rensic\_guy Andresen with a sample vial, July 2004. Dr. Andresen had to find some way to detect mir al Ubrary of Anthony Pridgeon Ywww.sim.mla.gov/news/press\_releases/visibleproofphotos.html ute concentrations of Pavulon in long-buried victims-a method of teasing the drug out of decomposed tissue nt\_diagram\_iii\_c\_204 agram, 1940 i diagram", 1940 <sup>—</sup> — Whone, The Finger Print Instructor...Based upon the Sir E. R. Henry System of Classifying and Filing..., New York w.rim.nih.gov/vitibleproofs/galleries/exhibition/views\_image\_11.html phis\_sewer.jpg ive Commmons user mojorider2. / Paul Everett //www.flickr.com/photos/mojorider2/4670627480/ mars spirit rover artists rendition.jgp