## Code Clinic: How to Write Code the Compiler Can Actually Optimize

Mike Acton | Engine Director, Insomniac Games

Mike Acton Macton@ Twitter. acton games.com games.com

# Or...

# Yet another privileged CIS white male in the AAA space talking abut data.

### What is good code?

Our role is not to write "good" code. Our role is to solve our problems well.

With fixed hardware resources, that often means reducing waste or at least having the potential to reduce waste (i.e. optimizable) so that we can solve bigger and more interesting problems in the same space.

"Good" code in that context is the code that was written based on a rational and reasoned analysis of the actual problems that need solving, hardware resources, and available production time.

i.e. At the very least not using the "pull it out your ass" design method combined with a goal to "solve all problems for everyone, everywhere."

# Can't the compiler do it?

A little review...

By Agner Fog. Technical University of Denmark. Copyright © 1996 - 2014. Last updated 2014-02-19.

(AMD Piledriver)

Instruction	Latency 🖵
SQRTSS/PS	13-15
VSQRTPS	14-15
SQRTSD/PD	24-26
VSQRTPD	24-26

http://www.agner.org/optimize/instruction\_tables.pdf

By Agner Fog. Technical University of Denmark. Copyright © 1996 - 2014. Last updated 2014-02-19.

(AMD Piledriver)

Instruction	Latency 🚽
FSIN	60-146
FCOS	~154
FSINCOS	86-141
FPTAN	86-204
FPATAN	60-352

http://www.agner.org/optimize/instruction\_tables.pdf

Tony Albrecht – Technical Consultant Developer Services



http://research.scee.net/files/presentations/gcapaustralia09/Pitfalls\_of\_Object\_Oriented\_Programming\_GCAP\_09.pdf



http://www.gameenginebook.com/SINFO.pdf

## The Battle of North Bridge



#### Andreas Fredriksson

@deplinenoise Follows YOU

Sr Engine Programmer at Insomniac Games. Asm and C. SIMD. Cigars. Dreams of Common Lisp. Slide guitar. Vim. Git. Build Systems. All opinions are my own, etc

San Fernando, CA · deplinenoise.wordpress.com





# L2 cache misses/frame

(Most significant component)

http://deplinenoise.wordpress.com/2013/12/28/optimizable-code/

```
class GameObject {
 float m Pos[2];
 float m Velocity[2];
 char m Name[32];
 Model* m_Model;
 // ... other members ...
  float m Foo;
 void UpdateFoo(float f)
    float mag = sgrtf(
     m_Velocity[0] * m_Velocity[0] +
     m Velocity[1] * m Velocity[1]);
     m Foo += mag * f;
};
```



#### Andreas Fredriksson

@deplinenoise Follows You

Sr Engine Programmer at Insomniac Games. Asm and C. SIMD. Cigars. Dreams of Common Lisp. Slide guitar. Vim. Git. Build Systems. All opinions are my own, etc San Fernando, CA · deplinenoise.wordpress.com

_ZN1ØGar	meObject'	9UpdateFooEf:	#	# @_ZN10GameObject9UpdateFooEf	
	.cfi_st	artproc			
# BB#Ø:					
	pushq	zrbx			
.Ltmp2:					
	.cfi_de	f_cfa_offset 16			
	subq	\$16, %rsp			
.Ltmp3:					
	.cfi_de	f_cfa_offset 32			
.Ltmp4:	~ • · ·	<i></i>			
	.cfi_of:	fset %rbx, -16			
	MOVSS	XXMMU, 12(Xrsp)	Ŧ	# 4-byte Spill	
	movq	Zrdi, Zrbx			
	movss	8(%rbx), %xmml			
	movss	12(%PDX), %XMMU			
	MUISS	XXMMI, XXMMI			
	nuiss	XXMMU, XXMMU			
	auuss	eastf			
	ulse	JYYYAANY YYMMA	#*	t 4-bute Folded Reload	
	addee	184(23by) 2vmm0	•••	a i bycc ioiucu neiouu	
	molies	$2 \times 10^{-1}$ (2 $\times $			
	addu	\$16. Zrsn			
	popq	Zrbx			
	ret				
.Ltmp5:					
-	.size	ZN10GameObject9Up	dateFool	DEF, Ltmp5ZN10GameObject9UpdateFooEf	
	.cfi_en	dproc			

_ZN10Ga	meObject	9UpdateFooEf:	#	e_z	10GameObject9UpdateFooEf
# <b>DD#</b> 0.	.cfi_st	artproc			
н ввню:	nusha	Zehy			
.Ltmp2:	Prond				
_	.cfi_de	f_cfa_offset 16			
T . D.	subq	\$16, %rsp			
.Ltmpj:	of: do	f of a offeet 22			
Ltmn4:	.cr1_ue	I_CIA_UIISEC 52			
- nomp 1-	.cfi_of:	fset %rbx, -16			
	movss	%xmm0, 12(%rsp)	#	4-by	te Spill
_	movq	Zrdi, Zrbx			
	MOVSS Mouss	8(%rbx), %xmm1 12(%rbx), %xmm0			2 x 32bit read; same cache line = ~200
	mulss	Xxmm1, Xxmm1			
	mulss	ххтт0, ххтт0			
	addss	xxmm1, xxmm0			
	callq	sqrtf 12(vaaa) vymmØ	**	1_h.	te Pelded Peleed
	addee	12\/rsp7, //mm0 184(уюру) уумм0	**	4-11	ve rutueu netuau
	MOVSS	$2 \times 10^{-10}$ (21) (21) (21) (21) (21) (21) (21) (21)			
	addq	\$16, %rsp			
	popq	%rbx			
T . E .	ret				
.Ltmp5:		7N10CamaObjectOlludateR		76	ItmpE _ 7N10CameObjectOlludatePeeEf
	.size .cfi en	dneoc	001	، والن	

_ZN10Ga	meObject'	9UpdateFooEf :	#	0_ZN	10GameObject9UpdateFooEf
	.cfi_st	artproc			
# BB#0:	_	_			
	pushq	%rbx			
.Ltmp2:		~ ~ ~ ~			
	.cfi_de	f_cfa_offset lb			
T. D.	subq	516, %rsp			
. цттрз:	- C ± 3-	C - C C C 4 - 7 7			
T 4 4 -	.cr1_ae:	f_cfa_offset J2			
- псыћа-	af: af	faat vuby -16			
	UT:	ISEL ARUX, TIO Yymm@ 19/Yyaan)	##	4_h.	ta Cuill
	mour	Zudi Zuby	**	4-ny	ce spiii
	mouse	R(vaby) vymm1			
	mouse	$12(\gamma_{\rm Pb})$ $\gamma_{\rm Nmm}$			
	mulss	Zxmm1. Zxmm1			Float mul. add = $\sim 10$
	mulss	XxmmØ. XxmmØ			
	addss	Xxmm1. Xxmm0			
_	callg	sgrtf			
	mulss	12(%rsp), %xmm0	#	4-by	te Folded Reload
	addss	184(%rbx), %xmm0		_	
	movss	%xmm0, 184(%rbx)			
	addq	\$16, %rsp			
	popq	%rbx			
	ret				
.Ltmp5:	_				
	.size	ZN10GameObject9UpdateF	'ool	≦ <b>f</b> , .	Ltmp5ZN10GameObject9UpdateFooEf
	.cfi_en	dproc			

_ZN10Gar	neObject? .cfi sta	7UpdateFooEf:	# C_	<b>Q_ZN10</b> GameObject9UpdateFooEf
# BB#Ø:	1	·····		
.Ltmp2:	pushq	ZPDX		
-	.cfi_def subq	f_cfa_offset 16 \$16, %rsp		
.Ltmp3:	.cfi_de	f_cfa_offset 32		
.Ltmp4:	.cfi_off movss	fset zrbx, -16 zxmm0, 12(zrsp)	# 4-	4-byte Spill
	movq movss movss mulee	<pre>%ra1, %rbx 8(%rbx), %xmm1 12(%rbx), %xmm0 %xmm1 %xmm1</pre>		
_	mulss	Xxmm0, Xxmm0 Xxmm1 Xxmm0		
	callq	sqrtf		Let's assume callq is replaced. Sqrt = ~30
	muiss addss mouss	12(%rsp), %хmm0 184(%rbx), %хmm0 %хmm0, 184(%rbx)	# 4-	4-byte Folded Keload
	addq popq	\$16, %rsp %rbx		
.Ltmp5:	ret	7N10CameObject9UvdateR		f ItmpE- 7N10ComeObject9UpdateEcoEf
	.cfi_en	lproc	JUET ,	r, .ncmps2MIUGAMEVDJect70puater00Er

_ZN1@Gai	meObject9	PUpdateFooEf :	#	<b>@_ZN10GameObject9UpdateFooEf</b>
# BB#Ø:	.cfi_sta	artproc		
Ltmn2:	pushq	%rbx		
г пошћа -	.cfi_def	f_cfa_offset 16		
Ltmn3:	subq	\$16, %rsp		
T 4 4 .	.cfi_def	f_cfa_offset 32		
.LTMP4:	ofi of	feat yuhy -16		
	mouss	хжтий. 12(2rsn)	##	4-hute Smill
	movq	%rdi, %rbx		
	movss	8(%rbx), %xmm1		
	movss	12(%rbx), %xmm0		
	mulss	%xmm1, %xmm1		
	mulss	xxmm0, xxmm0		
	addss	Xxmm1, Xxmm0		
	mulee	12(7281) 77mmØ	tt	4-bute Folded Reload
L	addss	184(2°hx). 2xmmM	T	Mul back to same addr: in $11 - 23$
	MOVSS	xxmmØ. 184(%rbx)		White back to same addr, in Er, - 5
	addq	\$16, %rsp		
	popq	%rbx		
	ret			
.Ltmp5:	_		_	
	.size	ZN10GameObject9UpdateFo	oo!	f, .Ltmp5ZN10GameObject90pdateFooEf
	.cfi_end	lproc		

_ZN10Gar	neObject?	UpdateFooEf :	#	<b>@_ZN10</b> GameObject9UpdateFooEf
# BB#Ø:	.cfi_sta	artproc		
Itmn9.	pushq	zrbx		
- ուաիշ -	.cfi def	f cfa offset 16		
<b>.</b> .	subq	\$16, %rsp		
.Ltmpj:	.cfi def	f cfa offset 32		
.Ltmp4:				
	.cfi_off	fset %rbx, -16		4 1
	MOVSS	Zedi Zeby	-##	4-byte Spill
	movss	8(%rbx), %xmm1		
	MOVSS	12(%rbx), %xmm0		
	mulss	Xxmm1, Xxmm1		
	mulss	Xxmm0, Xxmm0 Xxmm1 Xxmm0		
	callo	sartf		
	mulss	12(2rsp) 2xmm0	Ħ	4-byte Folded Reload
	addss	184(%rbx), %xmmØ		Read+add from new line
	MOVSS	2xmm0, 184(2rbx)		= ~200
	auuy nong	210, Arsp Zeby		
	ret			
.Ltmp5:				
	.size	ZN10GameObject9UpdateFo	ooE	f, .Ltmp5ZN10GameObject9UpdateFooEf
	.cr1_end	IPFOC		

\_ZN10GameObject9UpdateFooEf: .cfi\_startproc # BB#Ø: pushg Zrbx .Ltmp2: .cfi\_def\_cfa\_offset 16 subg \$16, %rsp .Ltmp3: .cfi\_def\_cfa\_offset 32 .Ltmp4: .cfi\_offset %rbx, -16 %xmm0, 12(%rsp) MOVSS Zrdi. Zrbx mova 8(%rbx), %xmm1 MOVSS 12(%rbx), %xmm0 MOVSS mulss %xmm1, %xmm1 mulss %xmm0, %xmm0 addss %xmm1, %xmm0 callq sgrtf 12(%rsp), %xmm0 mulss 184(%rbx), %xmm0 addss 2xmm0, 184(2rbx) movss \$16, %rsp addg %rbx popq ret .Ltmp5: \_ZN10GameObject9UpdateFooEf, .Ltmp5-\_ZN10GameObject9UpdateFooEf .size .cfi\_endproc

#### # 0\_ZN10GameObject9UpdateFooEf

#### Time spent waiting for L2 vs. actual work

~10:1

# 4-byte Folded Reload

\_ZN10GameObject9UpdateFooEf: .cfi\_startproc # BB#0: pushg Zrbx .Ltmp2: .cfi\_def\_cfa\_offset 16 subg \$16, %rsp .Ltmp3: .cfi\_def\_cfa\_offset 32 .Ltmp4: .cfi\_offset %rbx, -16 %xmm0, 12(%rsp) MOVSS Zrdi. Zrbx mova 8(%rbx), %xmm1 MOVSS 12(%rbx), %xmm0 MOVSS mulss %xmm1, %xmm1 mulss %xmm0, %xmm0 addss %xmm1, %xmm0 callq sgrtf 12(%rsp), %xmm0 mulss 184(%rbx), %xmm0 addss 2xmm0, 184(2rbx) movss \$16, %rsp addg %rbx popq ret .Ltmp5: \_ZN10GameObject9UpdateFooEf, .Ltmp5-\_ZN10GameObject9UpdateFooEf .size .cfi\_endproc

#### # 0\_ZN10GameObject9UpdateFooEf

Time spent waiting for L2 vs. actual work

~10(1)

This is the compiler's space.

# 4-byte Folded Reload



# COMPLER IS A TOOVI NOT A MAGIC WAND.

# Compiler *cannot* solve the most significant problems.

See also:





@raspofabs FOLLOWS YOU Games Programmer at 22Cans. Writer, Musican, Biker, Artist, fiddler with electronics and data-oriented development guy. Guildford, UK · dataorienteddesign.com



#### Data Oriented Design

Shared publicly - Jan 4, 2012

Things to consider : #08 Bad code is salvagable - bad data can be a much worse spaghetti to untangle

https://plus.google.com/u/0/+Dataorienteddesign/posts

# Today's subject: The 90% of problem space we need to solve that the compiler cannot.

(And how we can help it with the 10% that it can.)

Simple, obvious things to look for + Back of the envelope calculations = Substantial wins

# What's the most common cause of waste?

http://www.insomniacgames.com/three-big-lies-typical-design-failures-in-game-programming-gdc10/

# LIES (1) SOFTWARE IS A PLATFORM 2) CODE DESIGNED AROUND MODEL OF THE WORLD 3) CODE IS MORE IMPORTANT THAN DATA

http://deplinenoise.wordpress.com/2013/12/28/optimizable-code/

```
class GameObject {
 float m Pos[2];
 float m Velocity[2];
 char m Name[32];
 Model* m_Model;
 // ... other members ...
  float m Foo;
 void UpdateFoo(float f)
    float mag = sgrtf(
     m_Velocity[0] * m_Velocity[0] +
     m Velocity[1] * m Velocity[1]);
     m Foo += mag * f;
};
```

**X** 

#### Andreas Fredriksson

@deplinenoise FOLLOWS YOU

Sr Engine Programmer at Insomniac Games. Asm and C. SIMD. Cigars. Dreams of Common Lisp. Slide guitar. Vim. Git. Build Systems. All opinions are my own, etc San Fernando, CA · deplinenoise.wordpress.com

So how do we solve for it?

# L2 cache misses/frame

(Don't waste them!)

_ZN10Ga	meObject <sup>y</sup>	9UpdateFooEf:	#	# @_ZN1@GameObject9UpdateFooEf
# RR#0:	.cfi_sta	artproc		
	pushq	%rbx		
.Ltmp2:	.cfi de:	f cfa offset 16		
142.	subq	\$16, %rsp		
. темћо -	.cfi_de	f_cfa_offset 32		
.Ltmp4:	af; af;	foot why -16		
	.CTI_UT MOVSS MOVQ	xxmmØ, 12(%rsp) %rdi, %rbx	#	# 4-byte Spill
	MOVSS Movss	8(%rbx), %xmm1 12(%rbx), %xmm0		Waste 56 bytes / 64 bytes
_	mulss mulss addss callg	%xmm1, %xmm1 %xmm0, %xmm0 %xmm1, %xmm0 sortf		
	mulss addss movss addq popq ret	12(%rsp), %xmmØ 184(%rbx), %xmmØ %xmmØ, 184(%rbx) \$16, %rsp %rbx	#	# 4-byte Folded Reload
.Ltmp5:	size	ZN1ØGameObject9UndateF	مما	oEfLtmn5- ZN10GameObject9UndateFooF
	.cfi_en	dproc		

_ZN10Gar	neObjectS	UpdateFooEf :	#	# @_ZN10GameObject9UpdateFooEf
	.cfi_sta	artproc		
# BB#0:	_			
T.4	pushq	Zrbx		
.Ltmp2:	of: doi	C of a offect 16		
	suba	516 Zwen		
.Ltmn3:	Sand	tro, wrsb		
	.cfi_def	f_cfa_offset 32		
.Ltmp4:		— —		
	.cfi_off	fset %rbx, -16		
	MOVSS	%xmm0, 12(%rsp)	#	# 4-byte Spill
	movq	Zrdi, Zrbx		
	MOVSS	8(%rbx), %xmml		
	movss	IZ(XrDX), XXMM0 Xxmm1 Xxmm1		
	mulee	ХХШШІ, ХХШШІ Ухород Ухород		
	addee	2xmm1 2xmm0		
	callg	sortf		
_	mulss	12(%rsp). %xmm0	#	# 4-byte Folded Reload
	addss	184(%rbx), %xmm0		
	MOVSS	xxmm0, 184(%rbx)		Waste 60 bytes / 64 bytes
	addq	\$16, %rsp		
	popq	ZPDX		
I.tmn5 •	ret			
. nempo -	eize	ZN10GameObject9Undate	PRAD	oFf Ltmn5- ZN10GameObject9UndateFooFf
	.cfi end			ori intemps _anroadmeonJecciopdatertoori

_ZN10Game # pp#0-	eObject9 .cfi_sta	UpdateFooEf:	# 0_ZN10	GameObject9UpdateFo	ooEf
" DD#8. ] .Ltmp2:	pushq	%rbx			
	.cfi_def subq	<sup>2</sup> _cfa_offset 16 \$16, %rsp			
Ltmp3:	.cfi_def	_cfa_offset 32			
- Demp 1-	.cfi_off movss	set %rbx, -16 %xmm0, 12(%rsp)	1	90% waste!	
ı r	movq movss	%rdi, %rbx 8(%rbx), %xmm1			
	movss mulss mulse	12(%rbx), %xmm0 %xmm1, %xmm1 %xmm0 %xmm0			
i i	addss callq	%xmm1, %xmm0 sgrtf			
Г č	mulss addss	12(%rsp), %xmm0 184(%rbx), %xmm0	# 4-byte	Folded Reload	
i i	movss addq	ХхммИ, 184(Хrbx) \$16, Хrsp Хиру			
Ltmp5:	popų ret	7.FUX			
	.size .cfi_end	_ZN10GameObject9UpdateFo lproc	oEf, .Lt	mp5ZN10GameObject	t9UpdateFooEf

_ZN10GameObject9UpdateFooEf: .cfi_startproc	# @_ZN10GameObject9UpdateFooEf
# BB#0: pushq %rbx	
.Ltmp2:	
.cfi_def_cfa_offset 16	
subq \$16, %rsp	
.Ltmp3:	
.cfi_def_cfa_offset 32	
.LTMP4:	Alternatively,
.CII_UTISEL APUX, TIO mouso (yymm@ 12(yyon)	Only 10% capacity used*
mour Zedi Zeby	
mouss 8(2rbx), 2xmm1	
movss 12(%rbx). %xmmØ	
mulss %xmm1, %xmm1	
mulss %xmm0, %xmm0	
addss %xmm1, %xmm0	
callq sqrtf	
mulss 12(%rsp), %xmmU	# 4-byte Folded Keload
addss 184(%rbx), %xmmØ	
MOVSS XXMMU, 184(XPDX) adda (16 Yuan	
auuq 210, Arsp nong Yuby	
ret	
.Ltmp5:	
.size _ZN10GameObject9	Y NI - I - I
.cfi_endproc	* Not the same as "used well", but we'll start here.
.cfi_endproc	* Not the same as used well, but we li start here.

Ef

```
struct FooUpdateIn {

   float m_Velocity[2];
   float m Foo;
 _};
□ struct FooUpdateOut {
   float m Foo;
 _};

void UpdateFoos(const FooUpdateIn* in, size_t count, FooUpdateOut* out, float f)

 {
   for (size t i = 0; i < count; ++i) {</pre>
     float mag = sqrtf(
       in[i].m_Velocity[0] * in[i].m_Velocity[0] +
       in[i].m_Velocity[1] * in[i].m_Velocity[1]);
       out[i].m_Foo = in[i].m_Foo + mag * f;
   }
```

```
struct FooUpdateIn {

                                               12 bytes x count(5) = 72
   float m_Velocity[2];
    float m Foo;
  5
□ struct FooUpdateOut {
   float m Foo;
 _};
□ void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)
 {
   for (size t i = 0; i < count; ++i) {</pre>
      float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] +
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
       out[i].m Foo = in[i].m Foo + mag * f;
   }
```
```
struct FooUpdateIn {

                                               12 bytes x count(5) = 72
   float m_Velocity[2];
   float m Foo;
                                               4 bytes x count(5) = 20

struct FooUpdateOut {

   float m Foo;
 };
□ void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)
 {
   for (size t i = 0; i < count; ++i) {</pre>
      float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] +
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
        out[i].m Foo = in[i].m Foo + mag * f;
   }
```

```
struct FooUpdateIn {

                                                12 bytes x count(32) = 384 = 64 \times 6
    float m_Velocity[2];
    float m Foo;
                                                 4 bytes x count(32) = 128 = 64 \times 2

struct FooUpdateOut {

    float m Foo;
 };
□ void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)
  {
   for (size t i = 0; i < count; ++i) {</pre>
      float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] +
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
        out[i].m Foo = in[i].m Foo + mag * f;
   }
```

```
struct FooUpdateIn {

                                                12 bytes x count(32) = 384 = 64 \times 6
    float m_Velocity[2];
    float m Foo;
                                                4 bytes x count(32) = 128 = 64 \times 2

struct FooUpdateOut {

    float m Foo;
 };

void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)

  {
   for (size_t i = 0; i < count; ++i) {</pre>
      float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] + (6/32) = ~5.33 loop/cache line
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
        out[i].m Foo = in[i].m Foo + mag * f;
   }
```

```
struct FooUpdateIn {

                                                 12 bytes x count(32) = 384 = 64 \times 6
    float m_Velocity[2];
    float m Foo;
                                                 4 bytes x count(32) = 128 = 64 \times 2
struct FooUpdateOut {
    float m Foo;
 };
□ void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)
 ł
   for (size t i = 0; i < count; ++i) {</pre>
      float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] +
                                                                (6/32) = ~5.33 loop/cache line
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
                                                                Sqrt + math = \sim40 x 5.33 = 213.33 cycles/cache line
        out[i].m Foo = in[i].m Foo + mag * f;
   }
```

```
struct FooUpdateIn {

                                                 12 bytes x count(32) = 384 = 64 \times 6
    float m_Velocity[2];
    float m Foo;
                                                 4 bytes x count(32) = 128 = 64 \times 2
struct FooUpdateOut {
    float m Foo;
 };
□ void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)
 ł
   for (size t i = 0; i < count; ++i) {</pre>
      float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] +
                                                                (6/32) = ~5.33 loop/cache line
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
                                                                Sqrt + math = \sim40 x 5.33 = 213.33 cycles/cache line
        out[i].m Foo = in[i].m Foo + mag * f;
                                                                + streaming prefetch bonus
   }
```



```
In addition...
                                                      1. Code is maintainable

struct FooUpdateIn {

                                                       2. Code is debugable
   float m_Velocity[2];
                                               3. Can REASON about cost of change
   float m Foo;
 _};
struct FooUpdateOut {
   float m Foo;
 };
void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)
 {
   for (size t i = 0; i < count; ++i) {</pre>
     float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] +
                                                              (6/32) = ~5.33 loop/cache line
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
                                                              Sqrt + math = \sim40 x 5.33 = 213.33 cycles/cache line
       out[i].m Foo = in[i].m Foo + mag * f;
                                                              + streaming prefetch bonus
```

```
In addition...
                                                      1. Code is maintainable

    struct FooUpdateIn {

                                                        2. Code is debugable
   float m_Velocity[2];
                                                3. Can REASON about cost of change
   float m Foo;
 _};
                                             Ignoring inconvenient facts is not engineering;
struct FooUpdateOut {
                                                             It's dogma.
   float m Foo;
 };
void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)
 {
   for (size t i = 0; i < count; ++i) {</pre>
      float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] +
                                                               (6/32) = ~5.33 loop/cache line
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
                                                               Sqrt + math = \sim40 x 5.33 = 213.33 cycles/cache line
        out[i].m Foo = in[i].m Foo + mag * f;
                                                               + streaming prefetch bonus
```

# Let's review some code...

### v default - default - ogre / OgreMain / src / OgreNode.cpp

2	569ec69 2013-10-15 -	Full commit
1	/*	
3	This source file is (Object-oriente	a part of OGRE ad Graphics Rendering Engine)
5 6	For the latest info	, see http://www.ogre3d.org/
7	Copyright (c) 2000-	2013 Torus Knot Software Ltd



Mike Acton reviewed the 1.9 version. Perhaps it would've been more interesting to see a review of the 2.0 file which has been refactored to better fit Data Oriented Design principles (and I'm sure there are things I wrote to criticize). Many of the things he criticizes of 1.9 have been fixed. Nevertheless there are things we can learn. Note that if he weren't right, then it would be hard to explain why there was a **5x performance increase** between 1.9 and 2.0.

http://yosoygames.com.ar/wp/2013/11/on-mike-actons-review-of-ogrenode-cpp/

.4.4	
45	namespace Ogre {
46	
47	NameGenerator Node::msNameGenerator("Unnamed_");
48	Node::QueuedUpdates Node::msQueuedUpdates;
49	//
50	Node::Node()
51	:mParent(0),
52	mNeedParentUpdate(false),
53	<pre>mNeedChildUpdate(false),</pre>
54	mParentNotified(false),
55	mQueuedForUpdate(false),
56	mOrientation(Quaternion::IDENTITY),
57	mPosition(Vector3::ZERO),
58	mScale(Vector3::UNIT_SCALE),
59	mInheritOrientation(true),
60	mInheritScale(true),
61	mDerivedOrientation(Quaternion::IDENTITY),
62	mDerivedPosition(Vector3::ZERO),
63	mDerivedScale(Vector3::UNIT_SCALE),
64	mInitialPosition(Vector3::ZERO),
65	mInitialOrientation(Quaternion::IDENTITY),
66	mInitialScale(Vector3::UNIT_SCALE),
67	mCachedTransformOutOfDate(true),
68	mListener(0),
69	mDebug(0)
70	{
71	// Generate a name
72	<pre>mName = msNameGenerator.generate();</pre>
73	
74	needUpdate();
75	
76	}

NameGenerator Node::msNameGenerator("Unnamed_");	(1) Can't re arrange memory (much)
Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can the arrange memory (much)
Node::Node()	
:mParent(0).	
mNeedParentUpdate(false).	Limited by ABI
mNeedChildUpdate(false),	
mParentNotified(false),	
mQueuedForUpdate(false),	Can't limit unused reads
mOrientation(Quaternion::IDENTITY),	
mPosition(Vector3::ZERO),	
mScale(Vector3::UNII_SCALE),	Extra padding
mInneritOrientation(true),	
mDerivedOrientation(Ougternion::IDENTITY)	
mDerivedPosition(Vector3::ZERO).	
mDerivedScale(Vector3::UNIT_SCALE),	
mInitialPosition(Vector3::ZERO),	
mInitialOrientation(Quaternion::IDENTITY),	
mInitialScale(Vector3::UNIT_SCALE),	
mCachedTransformOutOfDate(true),	
mListener(0),	
mDebug(Ø)	
i 11 Conorate a nome	
mName = msNameGenerator generate():	
initiale = insituledener deor .gener dee(),	
needUpdate():	
1	

#### Can a C++ compiler re-order elements in a struct

http://stackoverflow.com/questions/916600/can-a-c-compiler-re-order-elements-in-a-struct

### In theory...

Here's the relevant part of the standard:

Section 9.2.12:

Nonstatic data members of a (non-union) class declared without an intervening access-specifier are allocated so that later members have higher addresses within a class object. The order of allocation of nonstatic data members separated by an access-specifier is unspecified (11.1)"

share | improve this answer

edited May 27 '09 at 16:54

answered May 27 '09 at 16:20 jalf 136k • 22 • 197 • 412

### In practice...

class Foo public: uint8\_t a0; uint16\_t b0; uint8\_t c0; bool dØ; uint8\_t a1; uint16\_t b1; uint8\_t c1; bool d1; uint8\_t a2; uint16\_t b2; uint8\_t c2; bool d2; ;; class Bar public: uint8\_t a0; public: uint16\_t b0; public: uint8\_t c0; public: bool dØ; public: uint8\_t a1; public: uint16\_t b1; public: uint8\_t c1; public: bool d1; public: uint8\_t a2; public: uint16\_t b2; public: uint8\_t c2; public: bool d2; H

## In practice...

lass Foo			
wblic:			
uint8 t	аØ:		
uint16 t	hØ:		
uint8 t	сЙ:		
hool	<b>д</b> й:		
2001	,		
uint8_t	a1;		
uint16_t	b1;		
uint8 t	c1;		
bool	d1:		
uint8_t	a2;		
uint16_t	b2;		
uint8_t	c2;		
bool	d2;		
;			
lass Bar			
public:	uint8_t	aØ;	
public:	uint16_t	bØ;	
public:	uint8_t	сØ;	
public:	bool	dØ;	
public:	uint8_t	a1;	
public:	uint16_t	b1 ;	
public:	uint8_t	c1;	
public:	bool	d1;	
		_	
public:	uint8_t	a2;	
public:	uint16_t	b2;	
public:	uint8_t	c2;	
public:	bool	d2;	
;			



Foo,	aØ	),
Foo,	ЪØ	),
Foo,	сØ	>,
Foo,	dØ	>,
Foo,	a1	>,
Foo,	<b>b1</b>	>,
Foo,	c1	>,
Foo,	d1	>,
Foo,	a2	Σ,
Foo,	b2	Σ.
Foo,	c2	Σ.
Foo,	d2	$\rangle$ ;
	Foo, Foo, Foo, Foo, Foo, Foo, Foo, Foo,	Foo, aØ Foo, bØ Foo, cØ Foo, dØ Foo, a1 Foo, a1 Foo, c1 Foo, c1 Foo, a2 Foo, b2 Foo, c2 Foo, d2

45	namespace Ogre {	
46 47 48 49	<pre>NameGenerator Node::msNameGenerator("Unnamed_");     Node::QueuedUpdates Node::msQueuedUpdates;</pre>	(2) Bools and last-minute decision making
50	Node::Node()	
51	·mParent(0)	
52	mNeedParentUpdate(false),	
53	mNeedChildUpdate(false),	
54	mParentNotified(false),	
55	mQueuedForUpdate(false),	
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	
59	mInheritOrientation(true),	
60	mInheritScale(true),	
61	<pre>mDerivedOrientation(Quaternion::IDENTITY),</pre>	
62	mDerivedPosition(Vector3::ZERO),	
63	mDerivedScale(Vector3::UNIT_SCALE),	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(0)	
70	{	
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

## bools in structs...

lass (	att	tribute	;_((_	_packed_	_>>	Foo	Foo C	
- oublic	=						ัลดี:	И
uint	8 t	аØ:					ью	1
uint	16 t	ЬØ:					cØ.	2
uint	Q +	- GI -					40	- 4
hoo l	_	40:					- 1 - 1	
2001	-	,					al	
uint	Q +	a1 •						. D
uint	16 4	a1,					C1 -	Ö
	10_L	.1.					a1:	÷ Y
uint	8_C	C1;					a2 :	: 1
poo 1		d1;					b2 :	: 1
	_	_					c2:	: 1
uint	8_t	a2;					d2 :	: 1
uint	16_t	b2;					>	
uint	8_t	c2;						
bool		d2;						
<b>};</b>								

Foo ∢	
a0: b0:	0 1 3
dØ: a1:	4 5
b1: c1: d1: a2: b2: c2: d2:	ь 8 9 10 11 13 14
3	

### (3) Extremely low information density

## bools in structs...

	lass	att	ribute	•<<_	_packed_	_>>	Foo	Fe	00
pı	ublio uint uint	c: t8_t t16 t	аØ; hØ:						aØ: bØ:
	uin	+ <u>Q</u> _+	cØ;						dØ:
	poo	1	dØ;						a1:
		-0 -	- 4 •						лι:
	uin	CQ_C	al;						c1:
	uin	t16_t	ы;						d1:
	uin	t8_t	c1;						a2:
	boo	1	d1;						b2:
									c2:
	uint	t8_t	a2;						d2:
	uint	t16_t	b2;					3	
	uint	t8 t	c2;						
	hoo	1	d2:						
¥	;		,						

Foo {				
a	9:	Ø		
b	Ø:	1		
d	0:	4		
a	1:	5		
D.	1:	b		
C	1:	8		
<b>d</b> :	1:	9		
a	2:	10		
b	2:	11		
c	2:	13		
d	2:	14		
<b>N</b>				

#### (3) Extremely low information density

#### How big is your cache line?

class Fo	0	
<		
public:		
bool	m_NeedParentUpdate;	
bool	<pre>m_NeedChildUpdate;</pre>	
bool	m_ParentNotNotified;	
Mat4	m_ObjectWorld;	
bool	m_InheritScale;	
bool	m_InheritOrientation;	
>;	—	

#### Foo

m\_NeedParentUpdate: 0
m\_NeedChildUpdate: 1
m\_ParentNotNotified: 2
m\_ObjectWorld: 4
m\_InheritScale: 68
m\_InheritOrientation: 69

## bools in structs...

lass;	att	ribute	<u>}_((</u> _	_packed_	_>>	Foo	l l
public	::						
uint	t8_t	aØ;					
	t16_t	ь0; с0-					
boo	L	dØ;					
uint	t8_t	al;					
uint	t8 t	c1:					
boo	l l	d1;					
		0					
	t8_t ⊦16 +	aZ; b2•					
uint	t8 t	c2:					2
boo	1	d2;					
≻;							

Foo		
م0: b0:	0 1 7	
d0: a1:	4 5	
b1 : c1 : d1 : a2 : b2 : c2 : d2 :	ь 8 9 10 11 13 14	

### (3) Extremely low information density

### How big is your cache line?

### What's the most commonly accessed data?

class Fo	0	Foo
۲.		Κ
public:		m_h
pool	<pre>m_NeedParentUpdate;</pre>	m
pool	<pre>m_NeedChildUpdate;</pre>	m F
bool Mata	m_ParentNotNot1fled;	m í
Mat4	M_VDJectWorld;	
	m_inneritocale,	
7. 1001	M_INMEFICOFIENCALION,	
· •		}

Foo K		
<pre>m_NeedParentUpdate:</pre>	Ø	
m_NeedChildUpdate:	1	
<pre>m ParentNotNotified:</pre>	2	
m_ObjectWorld:	4	
m_InheritScale:	68	
m_InheritOrientation:	69	
3		

64b?

How is it used? What does it generate?

```
int
Definition Foo::Bar( int count )
{
    int value = 0;
    for (int i=0;i<count;i++)
    {
        if ( m_NeedParentUpdate )
        {
            value++;
        }
     }
    return (value);
}</pre>
```

#### (2) Bools and last-minute decision making

#### ?Bar@Foo@@QEAAHH@Z PROC ; Foo::Bar, COMDAT ; 1696 : int value = 0; 00000 33 c0 xor eax, eax ; 1697 : for (int i=0;i<count;i++) 00002 85 d2 edx, edx test jle SHORT \$LN2@Bar 00004 7e 11 ; 1696 : int value = 0; 00006 44 8a 01 r8b, BYTE PTR [rcx] mov 00009 8b ca ecx, edx mov \$LL9@Bar: ; 1698 : { ; 1699 : if ( m NeedParentUpdate ) 0000b 45 84 c0 test r8b, r8b je SHORT \$LN10@Bar 0000e 74 02 ; 1700 : { ; 1701 : value++; 00010 ff c0 inc eax \$LN10@Bar: ; 1697 : for (int i=0;i<count;i++) 00012 48 ff c9 dec ncx 00015 75 f4 jne SHORT \$LL9@Bar \$LN2@Bar: ; 1702 : } ; 1703 : } ; 1704 : return (value); ; 1705 : }

#### MSVC

?Bar@Foo@@QEAAHH@Z PROC	; Foo::Bar, COMDAT	MSVC
; 1696 : int value = 0;		
00000 33 c0 xor	eax, eax	
; 1697 : for (int i=0;i <d< td=""><td>count;i++)</td><td></td></d<>	count;i++)	
00002 85 d2 test 00004 7e 11 jle	edx, edx SHORT \$LN2@Bar	
; 1696 : int value = 0;		
00006 44 8a 01 mov 00009 8b ca mov \$LL9@Bar:	r8b, BYTE PTR [rcx] ecx, edx	Re-read and re-test
; 1698 : { ; 1699 : if ( m_NeedPar	rentUpdate )	
0000b 45 84 c0 test 0000e 74 02 je SH	r8b, r8b HORT \$LN10@Bar	
; 1700 : { : 1701 : value++:		
00010 ff c0 inc \$LN10@Bar:	eax	Increment and loop
; 1697 : for (int i=0;i <d< td=""><td>count;i++)</td><td></td></d<>	count;i++)	
00012 48 ff c9 dec 00015 75 f4 jne \$LN2@Bar:	rcx SHORT \$LL9@Bar	
; 1702 : } ; 1703 : } ; 1704 : return (value); ; 1705 : }		

#### ?Bar@Foo@@QEAAHH@Z PROC ; Foo::Bar, COMDAT ; 1696 : int value = 0; 00000 33 c0 xor eax, eax ; 1697 : for (int i=0;i<count;i++) edx, edx 00002 85 d2 test jle SHORT \$LN2@Bar 00004 7e 11 ; 1696 : int value = 0; r8b, BYTE PTR [rcx] 00006 44 8a 01 mov 00009 8b ca ecx, edx mov \$LL9@Bar: ; 1698 : { if ( m NeedParentUpdate ) ; 1699 : r8b, r8b 0000b 45 84 c0 test 0000e 74 02 je SHORT \$LN10@Bar ; 1700 : : 1701 : value++: 00010 ff c0 inc eax \$LN10@Bar: ; 1697 : for (int i=0;i<count;i++) 00012 48 ff c9 dec rcx 00015 75 f4 ine SHORT \$LL9@Bar \$LN2@Bar: ; 1702 : } ; 1703 : } return (value); ; 1704 : 1705 : }



#### Why?

#### Re-read and re-test...

Super-conservative aliasing rules...? Member value might change?

Increment and loop...

What about something more aggressive...?



clang version 3.4 (tags/RELEASE\_34/final) Target: x86\_64-unknown-linux-gnu Thread model: posix

	.type	ZNJF00JBarE1,@function	1	
_ZN3Foo	3BarEi:		#	@_ZN3Foo3BarEi
	.cfi_sta	artproc		
‡ BB#Ø:				
	xorl	%eax, %eax		
	testl	%esi, %esi		
	jle	.LBB1_2		
‡ BB#1:			#	z.lr.ph
	movzbl	(%rdi), %eax		
	negl	%eax		
	andl	%esi, %eax		
<b>LBB1_2</b>	-			
	ret			
.Ltmp3:				
	.size .cfi_end	_ZN3Foo3BarEi, .Ltmp3 lproc	ZN	3Foo3BarEi

What about something more aggressive...?



clang version 3.4 (tags/RELEASE\_34/final) Target: x86\_64-unknown-linux-gnu Thread model: posix



Test once and return...

Okay, so what about...

```
int
Foo::Bar( int count )
  int value = 0;
  for (int i=0;i(count;i++)
  <
    if ( m_NeedParentUpdate )
    £
     value++;
    }
  >
  return (value);
int
Foo::Baz( int count )
  int value = 0;
 for (int i=0;i(count;i++)
  €
    if ( Bar(count) > 0 )
    ξ.
      value++;
    }
  >
 return (value);
```

clang version 3.4 (tags/RELEASE\_34/final) Target: x86\_64-unknown-linux-gnu Thread model: posix

...well at least it inlined it?

_ZN3Foo3	BazEi:	_245100504211,014000100	# 0	ZN3Foo3BazEi
	.cfi_sta	artproc		
н ввно:		Kony Kony		
	testl	Zesi Zesi		
	ile	LBB2_7		
# BB#1:	0		# 2	:.lr.ph
	xorl	%eax, %eax		
	movl	Zesi, Zr8d		
	andl	5-2, %r8d		
H DDHJ.	Jе	.LBBZ_Z		
# DD#J.	mou l	Zr8d Zecx		
	xorl	zr9d. zr9d		
	.align	16, 0x90		
.LBB2_4:	:		# 2	vector.body
			# =	=>This Inner Loop Header: Depth=1
	movzbl	(%rdi), %edx		
	negl	Xedx		
	testi	Zesi, Zeax		
	mouzhl	Xdl Yedy		
	addl	%edx. %eax		
	addl	%edx, %r9d		
	addl	\$-2, %ecx		
	jne	.LBB2_4		
T D D O .	յան	.LBB2_5		
.LBBZ_Z:	: voul	wwo.d wwo.d		
	XUFI	zngd zngd		
.LBB2 5:	:	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	# 2	middle.block
	addl	%r9d, %eax		
	cmpl	zesi, zr8d		
	je	.LBB2_7		
	.align	16, 0×90	<b>.</b> .	
.LBB2_6:	:		# 2	(_ZNJF003BarE1.exit
	mouzhl	(Yudi) Yecy	# -	-/INIS Inner Loop Header. Depth-1
	neal	Zecx		
	testl	Xesi, Xecx		
	setg	×cl		
	movzbl	%cl, %ecx		
	addl	Xecx, Xeax		
	incl	Xr8d		
	ine	LRR2 6		
LBB2 2:	Jue		# 2	. crit edge
	ret			
.Ltmp4:				

	00000 33	c0	xor	eax, eax
;	1711 :	for (int :	i=0;i≺cou	unt;i++)
	00002 85	42	test	edv edv
	00002 05	25	410	SHOPT \$1 N20P>7
	00004 /e	25	lie	SHOKT \$LN2@ba2
	00006 44	8a 11	mov	r10b, BYTE PTR [rcx]
	00009 44	8b ca	mov	r9d, edx
	0000c 44	8b c2	mov	r8d, edx
\$۱	LL4@Baz:			
	0000f 33	c9	xor	ecx, ecx
	00011 49	8b d1	mov	rdx, r9
\$1	L17@Baz:			
	1699 :	if ( m l	VeedParer	ntUndate )
,	10000			
	00014 45	84 d2	test	r10b, r10b
	00017 74	02	je SHOF	RT \$LN18@Baz
;	1701 :	value	++;	
	00010 55	-1	4.0.0	
¢1	N120P-71	C1	THC	ecx
ЪI	INTOGODAT:			
;	1697 :	for (int i	i=0;i≺cou	unt;i++)
	0001b 48	ff ca	dec	rdx
	0001e 75	f4	jne	SHORT \$LL17@Baz
;	1/13 :	1† ( Bar	r(count)	)
	00020 85	c9	test	ecx, ecx
	00022 74	02	ie SHOP	RT \$LN3@Baz
			Je 5.101	
;	1715 :	value	++;	
0.	00024 ff	60	100	eax
≯l	LIN2@B92:			
:	1711 :	for (int :	i=0:i≺cou	unt;i++)
1				
	00026 49	ff c8	dec	r8
	00029 75	e4	ine	SHORT \$LL4@Baz

di Naca



MSVC doesn't fare any better...

```
int
Foo::Bar( int count )
   int value
                    = 0;
   bool need_update = m_NeedParentUpdate;
   for (int i=0;i<count;i++)
     if ( need_update )
       value++;
   return (value);
 int
Foo::Baz( int count )
   int value
                = 0.
   bool need_update = Bar(count) > 0;
   for (int i=0;i<count;i++)</pre>
     if ( need_update )
       value++;
   return (value);
```

#### (4) Ghost reads and writes

Don't re-read member values or re-call functions when you *already* have the data.

clang version 3.4 (tags/RELEASE\_34/final) Target: x86\_64-unknown-linux-gnu Thread model: posix

### BAM!

_ZN3Foo3BazÉi:		# @_ZN3Foo3BazEi
.cfi_s	tartproc	
# BB#0:	-	
xorl	%eax, %eax	
testl	%esi, %esi	
.ile	.LBB2_2	
# BB#1:		# %.lr.ph
movzb]	l (%rdi), %ecx	-
negl	%ecx	
xorl	%eax, %eax	
testl	%esi, %ecx	
cmovg]	l %esi, %eax	
.LBB2_2:		# %_ZN3Foo3BarEi.exit
ret		
T dama d a		

; 1697 : bool need\_update = m\_NeedParentUpdate; 00000 44 8a 09 mov r9b, BYTE PTR [rcx] 00003 33 c0 xor eax, eax r8d, r8d 00005 45 33 c0 xor ; 1698 : for (int i=0;i<count;i++) 00008 85 d2 test edx, edx ile SHORT \$LN8@Baz 0000a 7e 0f ; 1711 : int value = 0; 0000c 8b ca mov ecx, edx \$LL10@Baz: ; 1700 : if ( need\_update ) 0000e 45 84 c9 test r9b, r9b 00011 74 03 je SHORT \$LN9@Baz 00013 41 ff c0 inc r8d \$LN9@Baz: ; 1698 : for (int i=0;i<count;i++) 00016 48 ff c9 dec rcx 00019 75 f3 ine SHORT \$LL10@Baz \$LN8@Baz: ; 1712 : bool need\_update = Bar(count) > 0; 0001b 45 85 c0 test r8d, r8d 0001e 41 0f 9f c0 setg r8b ; 1713 : for (int i=0;i<count;i++) 00022 85 d2 test edx, edx 00024 7e 0e jle SHORT \$LN2@Baz 00026 8b ca mov ecx, edx \$LL4@Baz: ; 1715 : if ( need\_update ) 00028 45 84 c0 test r8b, r8b 0002b 74 02 je SHORT \$LN3@Baz 0002d ff c0 inc eax \$LN3@Baz: ; 1713 : for (int i=0;i<count;i++) 0002f 48 ff c9 dec rcx 00032 75 f4 ine SHORT \$LL4@Baz \$LN2@Baz:

; 1720 : return (value);



```
int
Foo::Bar( int count )
   int value
                      - 0.
   bool need_update = m_NeedParentUpdate;
    if ( need update )
     for (int i=0;i<count;i++)</pre>
        value++;
   return (value);
 int
Foo::Baz( int count )
       value
   bool need_update = Bar(count) > 0;
    if ( need_update )
     for (int i=0;i<count;i++)</pre>
        value++;
    return (value);
```

#### (4) Ghost reads and writes

Don't re-read member values or re-call functions when you *already* have the data.

Hoist all loop-invariant reads and branches. Even superobvious ones that should already be in registers.

```
; Foo:::Baz, COMDAT
?Baz@Foo@@QEAAHH@Z PROC
; 1711 : int value = 0;
 00000 33 c0
             xor
                        eax, eax
; 1698 : if ( need_update )
 00002 38 01
                        BYTE PTR [rcx], al
             cmp
 00004 74 07 je SHORT $LN3@Baz
; 1699 : {
; 1700 : for (int i=0;i<count;i++)
 00006 85 d2
             test edx, edx
             jle
 00008 7e 03
                        SHORT $LN3@Baz
; 1712 : bool need_update = Bar(count) > 0;
; 1713 : if ( need_update )
 0000a 0f 4f c2 cmovg eax, edx
$LN3@Baz:
; 1714 : {
; 1715 : for (int i=0;i<count;i++)
; 1716 : {
        value++;
; 1717 :
; 1718 :
       }
; 1719 : }
; 1720 : return (value);
; 1721 : }
 0000d c3
           ret
                    0
                                 ; Foo::Baz
?Baz@Foo@@QEAAHH@Z ENDP
```

\_\_\_\_\_



<pre>?Baz@Foo@@QEAAHH@Z PROC ; Foo::Baz, COMDAT ; 1711 : int value = 0;</pre>	<b>N</b> Visual Studio
<pre>00000 33 c0 xor eax, eax ; 1698 : if ( need_update ) 00002 38 01     cmp   BYTE PTR [rcx], al 00004 74 07     je SHORT \$LN3@Baz ; 1699 : {     ; 1700 : for (int i=0;i<count;i++) \$ln3@baz<="" 00006="" 00008="" 03="" 7e="" 85="" d2="" edx="" edx,="" jle="" pre="" short="" test=""></count;i++)></pre>	• )
<pre>; 1712 : bool need_update = Bar(count) &gt; 0; ; 1713 : if ( need_update )</pre>	ranching, but more-or-less equivalent.
0000d c3 ret 0 ?Baz@Foo@@QEAAHH@Z ENDP ; Foo::Baz	

```
int
Foo::Bar( int count )
   int value
                      - 0.
   bool need_update = m_NeedParentUpdate;
    if ( need_update )
     for (int i=0;i<count;i++)</pre>
        value++;
   return (value);
 int
Foo::Baz( int count )
      t value
   bool need_update = Bar(count) > 0;
    if ( need update )
     for (int i=0;i<count;i++)</pre>
        value++;
    return (value);
```

#### (4) Ghost reads and writes

Don't re-read member values or re-call functions when you *already* have the data.

Hoist all loop-invariant reads and branches. Even superobvious ones that should already be in registers.

Applies to any member fields especially. (Not particular to bools)
-4-4		
45 46	namespace Ogre {	The story so far How can you help the compiler?
47	NameGenerator Node::msNameGenerator("Unnamed "):	
48	Node: OueuedUpdates Node: msOueuedUpdates:	
49	//	
50	Node: Node()	
51	:mParent(0).	
52	mNeedParentUndate(false)	
53	mNeedChildUndate(false)	
54	mParentNotified(false)	
55	mOueuedForUpdate(false).	
56	mOrientation(Ougternion::IDENTITY).	
57	mPosition(Vector3::7ER0)	
58	mScale(Vector3::UNIT_SCALE).	
59	mInheritOrientation(true).	
60	mInheritScale(true).	
61	mDerivedOrientation(Ougternion::IDENTITY)	
67	mDerivedPosition(Vector3::7ER0)	
63	mDerivedScale(Vector3::UNTT_SCALE)	
64	mInitial Position(Voctor3::7EPO)	
65	mInitial Oniontation (Oustannion: TDENTITY)	
66	minicial Scale (Vector2: UNIT SCALE)	
00	minitialScale(VectorS::UNIT_SCALE),	
67	mcachealransformoutorbate(true),	
68	mListener(0),	
69	mDebug(0)	
70	1	
/1	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

mespace Ogre {	The story so far How can you help the compiler?
NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
Node: Node()	
<pre>:mParent(0), mNeedParentUpdate(false), mNeedChildUpdate(false), mParentNotified(false), mOueuedForUpdate(false), mOrientation(Quaternion::IDENTITY), mPosition(Vector3::ZERO), mScale(Vector3::UNIT_SCALE), mInheritOrientation(true), mInheritScale(true), mDerivedOrientation(Quaternion::IDENTITY), mDerivedPosition(Vector3::ZERO), mDerivedScale(Vector3::UNIT_SCALE), mInitialPosition(Vector3::ZERO), mInitialOrientation(Quaternion::IDENTITY), mInitialOrientation(Quaternion::IDENTITY), mInitialOrientation(Quaternion::IDENTITY), mInitialOrientation(Quaternion::IDENTITY), mInitialScale(Vector3::UNIT_SCALE), mCachedTransformOutOfDate(true), mlistener(0).</pre>	
mDebug(0)	
<pre>i // Generate a name mName = msNameGenerator.generate(); needUpdate();</pre>	
}	

44 45 46	namespace Ogre {	The story so far How can you help the compiler?
47 48	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
49	//	
50	Node::Node()	
51	:mParent(0),	Arrange memory by probability of access.
52	mNeedParentUpdate(false),	
53	<pre>mNeedChildUpdate(false),</pre>	
54	mParentNotified(false),	
55	mQueuedForUpdate(false),	
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	
59	mInheritOrientation(true),	
60	mInheritScale(true),	
61	<pre>mDerivedOrientation(Quaternion::IDENTITY),</pre>	
62	mDerivedPosition(Vector3::ZERO),	
63	mDerivedScale(Vector3::UNIT_SCALE),	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(0)	
70	{	
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

44		
45	namespace Ogre {	The story so far How can you help the compiler?
47 48 49	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
50	Node: Node()	
51 52 53	<pre>:mParent(0), mNeedParentUpdate(false), mNeedChildUpdate(false),</pre>	(2) Bools and last-minute decision making
54	mParentNotified(false),	
55	mQueuedForUpdate(false),	
56	mUrientation(Quaternion::IDENIIIY),	
58	mPosition(vectors::ZERU),	
59	mInheritOrientation(true).	
60	mInheritScale(true).	
61	mDerivedOrientation(Quaternion::IDENTITY),	
62	mDerivedPosition(Vector3::ZERO),	
63	mDerivedScale(Vector3::UNIT_SCALE),	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(0)	
70	{	
/1	// Generate a name	
72	mName = msNameGenerator.generate();	
73	noodlindato();	
75	needopdate(),	
76	}	

44 45 46	namespace Ogre {	The story so far How can you help the compiler?
47 48	<pre>NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;</pre>	(1) Can't re-arrange memory (much)
49	//	
50	Node::Node()	
51	:mParent(0),	(2) Bools and last-minute decision making
52	mNeedParentUpdate(false),	
53	mNeedChildUpdate(false),	
54	mParentNotified(false),	Hoist decision making to first-opportunity
55	mQueuedForUpdate(false).	holst decision making to mist opportantly.
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	
59	mInheritOrientation(true),	
60	mInheritScale(true),	
61	mDerivedOrientation(Quaternion::IDENTITY),	
62	mDerivedPosition(Vector3::ZERO),	
63	mDerivedScale(Vector3::UNIT_SCALE),	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(0)	
70		
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

44 45 46	namespace Ogre {	The story so far How can you help the compiler?
47 48 49	<pre>NameGenerator Node::msNameGenerator("Unnamed_");     Node::QueuedUpdates Node::msQueuedUpdates;</pre>	(1) Can't re-arrange memory (much)
50	Node::Node()	
51 52	<pre>:mParent(0), mNeedParentUpdate(false), mNeedChildUndate(false)</pre>	(2) Bools and last-minute decision making
54 55	mNeedChildopdate(Taise), mParentNotified(false), mQueuedForUpdate(false),	(3) Extremely low information density
56 57 58 59 60 61 62 63 64 65 66 67 68 69	<pre>mOrientation(Quaternion::IDENTITY), mPosition(Vector3::ZERO), mScale(Vector3::UNIT_SCALE), mInheritOrientation(true), mInheritScale(true), mDerivedOrientation(Quaternion::IDENTITY), mDerivedPosition(Vector3::ZERO), mDerivedScale(Vector3::UNIT_SCALE), mInitialPosition(Vector3::ZERO), mInitialOrientation(Quaternion::IDENTITY), mInitialScale(Vector3::UNIT_SCALE), mInitialScale(Vector3::UNIT_SCALE), mInitialScale(Vector3::UNIT_SCALE), mListener(0), mListener(0), mDebug(0)</pre>	
70 71 72 73 74 75 76	<pre>{     // Generate a name     mName = msNameGenerator.generate();     needUpdate(); }</pre>	

44 45 46	namespace Ogre {	The story so far How can you help the compiler?
47 48	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
49	//	
50	Node::Node()	
51	:mParent(0),	(2) Bools and last-minute decision making
52	mNeedParentUpdate(false),	
53	mNeedChildUpdate(false),	
54	mParentNotified(false),	
55	mQueuedForUpdate(false),	(3) Extremely low information density
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	Maximize memory read value.
59	mInheritOrientation(true),	
60	mInheritScale(true),	
61	mDerivedOrientation(Quaternion::IDENTITY),	
62	mDerivedPosition(Vector3::ZERO),	
63	mDerivedScale(Vector3::UNIT_SCALE),	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(0)	
70	{	
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

44 45 46	namespace Ogre {	The story so far How can you help the compiler?
47 48	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
49	//	
50	Node: Node()	
51	:mParent(∅),	(2) Bools and last-minute decision making
52	mNeedParentUpdate(false),	
53	mNeedChildUpdate(false),	
54	mParentNotified(false),	(2) Extremely low information density
55	mQueuedForUpdate(false),	(3) Extremely low information density
56	murientation(Quaternion::IDENIIIY),	
57	mPosition(Vector3::2ERO),	Maximiza mamany road value
58	mScale(vector5::UNIT_SCALE),	waximize memory read value.
59	minneritorientation(true),	
60	minneritScale(true),	
101	mDerivedOrientation(Quaternion::IDENIII),	How can we measure this?
62	mDerivedPosition(vectors::2ERU),	
64	mDerivedScale(Vector5::UNIT_SCALE),	
04 CE	mInitialPosition(vectors::2ERU),	
60	minitialOrientation(Quaternion::IDENIIT),	
60	minitialScale(Vectors::UNII_SCALE),	
69	multichearransformoutorbate(true),	
00	mListener(0),	
70	Indebug(0)	
70	1/ Compate a nome	
72	mName - msNameConcrator apparate():	
72	minume = msnumedeneracor.generace(),	
74	needIndate():	
75	needopduce(),	
76	3	
10	1	

```
float choose = RandFloat();
bool is_spawn = (choose < m_SpawnChance);
if (is_spawn)
{
    Spawn();
    m_SpawnChance = 0.0f;
}
else
{
    m_SpawnChance += kFooSpawnChanceIncrease;
}
```

```
float choose = RandFloat();
bool is_spawn = (choose < m_SpawnChance);
if (is_epawn) = (choose < m_SpawnChance);
{
    Spawn();
    m_SpawnChance = 0.0f;
}
else
{
    m_SpawnChance += kFooSpawnChanceIncrease;
}
```

What is the information density for is\_spawn over time?

```
float choose = RandFloat();
bool is_spawn = (choose < m_SpawnChance);
printf("%d",is_spawn?1:0);
if (is_spawn)
{
    Spawn();
    m_SpawnChance = 0.0f;
}
else
{
    m_SpawnChance += kFooSpawnChanceIncrease;
}
```

What is the information density for is\_spawn over time?

## The easy way.

Zip the output 10,000 frames = 915 bytes = (915\*8)/10,000 = 0.732 bits/frame 

300000000000000100( 000000100 00000000000000010000000 000100000000000000000 000001000000000010001 000000010( 1001000000 000001001 10000: 000001000( 1000000000 0000000000001000000 000000010 100000000000001 0000001 0000 001000000000 300001000 1000000000 00001000000000000 001 00000000000000000 10001000 ៲៱៲៱៲៱៲៱៲៱៲៱៲៱៲៱ 01,0000000000 1 000000000000 19999919 0000000000 100000100 

Zip the output 10,000 frames = 915 bytes = (915\*8)/10,000 = 0.732 bits/frame

### Alternatively, Calculate Shannon Entropy:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

```
float choose = RandFloat();
bool is_spawn = (choose < m_SpawnChance);
if (is_spawn)
{
    Spawn();
    m_SpawnChance = 0.0f;
}
else
{
    m_SpawnChance += kFooSpawnChanceIncrease;
}
```

What does that tell us?

```
float choose = RandFloat();
bool is_spawn = (choose < m_SpawnChance);
if (is_spawn)
{
    Spawn();
    m_SpawnChance = 0.0f;
}
else
{
    m_SpawnChance += kFooSpawnChanceIncrease;
}
```

What does that tell us?

Figure (~2 L2 misses each frame ) x 10,000 If each cache line = 64b, 128b x 10,000 = 1,280,000 bytes

```
float choose = RandFloat();
bool is_spawn = (choose < m_SpawnChance);
if (is_spawn)
{
    Spawn();
    m_SpawnChance = 0.0f;
}
else
{
    m_SpawnChance += kFooSpawnChanceIncrease;
}
```

What does that tell us?

Figure (~2 L2 misses each frame ) x 10,000 If each cache line = 64b, 128b x 10,000 = 1,280,000 bytes

If avg information content = 0.732bits/frame X 10,000 = 7320 bits / 8 = 915 bytes

```
float choose = RandFloat();
bool is_spawn = (choose < m_SpawnChance);
if (is_spawn)
{
    Spawn();
    m_SpawnChance = 0.0f;
}
else
{
    m_SpawnChance += kFooSpawnChanceIncrease;
}
```

What does that tell us?

Figure (~2 L2 misses each frame ) x 10,000 If each cache line = 64b, 128b x 10,000 = 1,280,000 bytes

If avg information content = 0.732bits/frame X 10,000 = 7320 bits / 8 = 915 bytes

Percentage waste (Noise::Signal) = (1,280,000-915)/1,280,000

0.99928515625

# What're the alternatives?

(1) Per-frame...

1 of 512 (8\*64) bits used...

1 of 512 (8\*64) bits used...

(a) Make same decision x 512

1 of 512 (8\*64) bits used...

(a) Make same decision x 512

(b) Combine with other reads / xforms

1 of 512 (8\*64) bits used...

(a) Make same decision x 512

(b) Combine with other reads / xforms

Generally simplest.

- But things cannot exist in abstract bubble.
- Will require context.

(2) Over-frames...

(2) Over-frames...

i.e. Only read when needed

(2) Over-frames...

i.e. Only read when needed



44 45 46	namespace Ogre {	The story so far How can you help the compiler?
47 48	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
49	//	
50	Node: Node()	
51	:mParent(0),	(2) Bools and last-minute decision making
52	mNeedChildUndstaCfalse),	
55	mNeedChildOpdate(False),	
54	mparentNotified(faise),	(2) Extramoly low information density
55	moniontation(Oustonnion::IDENTITY)	(3) Extremely low information density
57	morition(Vector3::7EPO)	
58	mscale(Vector3: UNIT SCALE)	
59	mInheritOrientation(true)	Maximize memory read value.
60	mInheritScale(true).	
61	mDerivedOrientation(Ouaternion::IDENTITY).	
62	mDerivedPosition(Vector3::ZER0).	How can we measure this?
63	mDerivedScale(Vector3::UNIT_SCALE).	
64	mInitialPosition(Vector3::ZER0),	
65	mInitialOrientation(Quaternion::IDENTITY),	(Irvit.)
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(0)	
70	{	
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

espace Ogre {	The story so far How can you help the compiler?
NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
Node: Node()	
<pre>:mParent(0), mNeedParentUpdate(false),</pre>	(2) Bools and last-minute decision making
mNeedChildUpdate(false), mParentNotified(false), mQueuedForUpdate(false),	(3) Extremely low information density
<pre>mOrientation(Quaternion::IDENTITY), mPosition(Vector3::ZERO), mScale(Vector3::UNIT_SCALE), mInheritOrientation(true), mInheritScale(true), mDerivedOrientation(Quaternion::IDENTITY), mDerivedPosition(Vector3::ZERO), mDerivedScale(Vector3::UNIT_SCALE), mInitialPosition(Vector3::ZERO), mInitialOrientation(Quaternion::IDENTITY), mInitialOrientation(Quaternion::IDENTITY), mInitialScale(Vector3::UNIT_SCALE),</pre>	Maximize memory read value. How can we measure this?
	(Try it.)
<pre>mCachedTransformOutOfDate(true), mListener(0), mDebug(0)</pre>	All these "code smells" can be viewed as symptoms
<pre>{     // Generate a name     mName = msNameGenerator.generate(); }</pre>	of information density problems
needUpdate();	
}	

45 46	namespace Ogre {	The story so far How can you help the compiler?
47 48	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
49	//	
50	marcant(0)	
57	mNeedParentIndate(false)	(2) Bools and last-minute decision making
52	mNeedChildUndate(false)	
54	mParentNotified(false)	
55	mOueuedForUpdate(false).	(3) Extremely low information density
56	mOrientation(Ouaternion::IDENTITY).	(b) Extremely four information activity
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	(1) Chest reads and writes
59	mInheritOrientation(true),	(4) Gnost reads and writes
60	mInheritScale(true),	
61	<pre>mDerivedOrientation(Quaternion::IDENTITY),</pre>	
62	mDerivedPosition(Vector3::ZERO),	
63	mDerivedScale(Vector3::UNIT_SCALE),	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
109 70	mbebug(0)	
70	1 // Conorato a namo	
72	mName = msNameGenerator generate():	
73	intune – instanceener acor igener ace();	
74	needUpdate():	
75		
76	}	

44 45 46	namespace Ogre {	The story so far How can you help the compiler?
47 48	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
49	//	
50	Node: Node()	
51	:mparent(0),	(2) Bools and last-minute decision making
52	mNeedParentUpdate(false),	
55	mNeedChildUpdate(false),	
54	mParentNotified(false),	(2) Extremely low information density
55	moueuearoropaate(Talse),	(S) Extremely low information density
50	mortentation(Quaternion::IDENTIT),	
57	mPOSITION(Vector3::ZERU),	
50	mTnheni+Onientation(true)	(4) Ghost reads and writes
60	mInheritScale(true)	
61	mDerivedOrientation(Ousternion::IDENTITY)	
67	mDerivedPosition(Vector3::7ER0)	Don't re-read member values or re-call functions when
63	mDerivedScale(Vector3::INTT_SCALE)	you already have the data
64	mInitialPosition(Vector3::7ER0)	you <i>ulleuuy</i> have the uata.
65	mInitialOrientation(Ousternion::IDENTITY)	
66	mInitialScale(Vector3::INTT_SCALE)	
67	mCachedTransformOutOfDate(true)	
68	mlistener(0)	
69	mDebua(0)	
70	{	
71	// Generate a name	
72	mName = msNameGenerator.generate();	
73	5	
74	needUpdate();	
75		
76	}	

44 45 46	namespace Ogre {	The story so far How compiler ben provintier?
47 48	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(1) Can't re-arrange memory (much)
49	//	
50	Node::Node()	
51 52	<pre>:mParent(0), mNeedParentUpdate(false),</pre>	(2) Bools and last-minute decision making
53	<pre>mNeedChildUpdate(false),</pre>	
54	mParentNotified(false),	
55	mQueuedForUpdate(false),	(3) Extremely low information density
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	(1) Chost reads and writes
59	mInheritOrientation(true),	(4) Onost reads and writes
60	mInheritScale(true),	
61	<pre>mDerivedOrientation(Quaternion::IDENTITY),</pre>	Don't re read member values or re call functions when
62	<pre>mDerivedPosition(Vector3::ZER0),</pre>	Don't re-read member values of re-call functions when
63	<pre>mDerivedScale(Vector3::UNIT_SCALE),</pre>	you <i>already</i> have the data.
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	Easy to confuse compiler, even within the 10% space
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(0)	
70	{	
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

<pre>Are we done with the constructor Are we done with the constru</pre>	44		
<pre>NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates; //</pre>	45	namespace Ogre {	Are we done with the constructor?
<pre>Armodesistant State ("Unnamed."); Node::QueuedUpdates Node::msQueuedUpdates; //</pre>	46		
<pre>Mode::QueuedUpdates Node::msQueuedUpdates; //</pre>	47	NameGenerator Node::msNameGenerator("Unnamed_");	
<pre>Mode::Node()</pre>	48	Node::QueuedUpdates Node::msQueuedUpdates;	
<pre>Node::Node() imParent(@), imParent(@), mNeedParentUpdate(false), mNeedChildUpdate(false), mParentNotified(false), mOrientation(Quaternion::IDENTITY), mPosition(Vector3::ZERO), mScale(Vector3::UNIT_SCALE), mInheritScale(true), mInheritScale(true), mDerivedPosition(Vector3::ZERO), mDerivedPosition(Vector3::ZERO), mInitialPosition(Vector3::ZERO), mInitialScale(Vector3::UNIT_SCALE), mInitialScale(Vector3::ZERO), mInitialScale(Vector3::ZERO), mInitialScale(Vector3::ZERO), mInitialScale(Vector3::ZERO), mInitialScale(Vector3::ZERO), mInitialScale(Vector3::ZERO), mInitialScale(Vector3::DENTITY), mCachedTransformOutOfDate(true), mIstener(@), mIstener(@), mIstener(@), mName = msNameGenerator.generate(); needUpdate();</pre>	49	//	
<pre>S1 :mParent(0), S2 mNeedParentUpdate(false), mParentNotified(false), s4 mParentNotified(false), s5 mQueuedForUpdate(false), s6 mOrientation(Quaternion::IDENTITY), s7 mPosition(Vector3::ZERO), s8 mScale(Vector3::UNIT_SCALE), 9 mInheritScale(true), 60 mInheritScale(true), 11 mDerivedOrientation(Quaternion::IDENTITY), 62 mDerivedOrientation(Quaternion::IDENTITY), 63 mDerivedScale(Vector3::ZERO), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialOrientation(Quaternion::IDENTITY), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate(); 74 needUpdate(); 75 mScale(Vector3: Vector3: Vector3:</pre>	50	Node::Node()	
<pre>52 mNeedParentUpdate(false), 53 mNeedChildUpdate(false), 54 mParentNotified(false), 55 mQueuedForUpdate(false), 56 mOrientation(Quaternion::IDENTITY), 57 mFosition(Vector3::ZERO), 58 mScale(Vector3::UNIT_SCALE), 59 mInheritScale(true), 60 mInheritScale(true), 61 mDerivedPosition(Vector3::ZERO), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate(); 74 needUpdate(); 75 mInitialScale(Vector3: 75 mInitialScale(Vector3: 76 mInitialScale(Vector3: 77 mScale(Vector3: 78 mName = msNameGenerator.generate(); 79 mEather (State), 70 mScale(Vector3: 71 meather (State), 72 mName = msNameGenerator.generate(); 73 mScale(Vector3: 74 needUpdate(); 75 mScale(Vector3: 75 mScale(Vector3: 76 mScale(Vector3: 77 mScale(Vector3: 78 mScale(Vector3: 78 mScale(Vector3: 79 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 71 mScale(Vector3: 72 mScale(Vector3: 73 mScale(Vector3: 74 mScale(Vector3: 75 mScale(Vector3: 75 mScale(Vector3: 76 mScale(Vector3: 77 mScale(Vector3: 78 mScale(Vector3: 78 mScale(Vector3: 79 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 71 mScale(Vector3: 72 mScale(Vector3: 73 mScale(Vector3: 74 mScale(Vector3: 75 mScale(Vector3: 76 mScale(Vector3: 77 mScale(Vector3: 78 mScale(Vector3: 78 mScale(Vector3: 79 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 71 mScale(Vector3: 72 mScale(Vector3: 73 mScale(Vector3: 73 mScale(Vector3: 74 mScale(Vector3: 75 mScale(Vector3: 75 mScale(Vector3: 76 mScale(Vector3: 77 mScale(Vector3: 78 mScale(Vector3: 78 mScale(Vector3: 79 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 70 mScale(Vector3: 71 mScale(Vector3: 73 mScale(Vector</pre>	51	:mParent(0),	
<pre>53 mNeedChildUpdate(false), 54 mParentNotified(false), 55 mQueueForUpdate(false), 56 mOrientation(Quaternion::IDENTITY), 57 mPosition(Vector3::ZERO), 58 mScale(Vector3::INIT_SCALE), 59 mInheritOrientation(true), 60 mInheritScale(true), 61 mDerivedPosition(Vector3::ZERO), 62 mDerivedPosition(Vector3::ZERO), 63 mDerivedPosition(Vector3::ZERO), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 misitener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate(); 75 // Setting a state of the state of the</pre>	52	<pre>mNeedParentUpdate(false),</pre>	
<pre>54 mParentNotified(false), 55 mQueuedForUpdate(false), 56 mOrientation(Quaternion::IDENTITY), 57 mPosition(Vector3::ZERO), 58 mScale(Vector3::UNIT_SCALE), 59 mInheritOrientation(true), 60 mInheritScale(true), 61 mDerivedPosition(Vector3::ZERO), 62 mDerivedScale(Vector3::UNIT_SCALE), 63 mDerivedScale(Vector3::ZERO), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(@), 69 mDebug(@) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate(); 75</pre>	53	<pre>mNeedChildUpdate(false),</pre>	
<pre>55 mQueuedForUpdate(false), 56 mOrientation(Quaternion::IDENTITY), 57 mPosition(Vector3::ZERO), 58 mInheritOrientation(true), 60 mInheritScale(true), 61 mDerivedOrientation(Quaternion::IDENTITY), 62 mDerivedPosition(Vector3::ZERO), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate(); 75</pre>	54	mParentNotified(false),	
<pre>56 mOrientation(Quaternion::IDENTITY), 57 mPosition(Vector3::ZERO), 58 mScale(Vector3::UNIT_SCALE), 59 mInheritOrientation(True), 60 mInheritScale(true), 61 mDerivedOrientation(Quaternion::IDENTITY), 62 mDerivedPosition(Vector3::ZERO), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	55	mQueuedForUpdate(false),	
<pre>57 mPosition(Vector3::ZERO), 58 mScale(Vector3::UNIT_SCALE), 59 mInheritOrientation(true), 60 mDerivedOrientation(Quaternion::IDENTITY), 61 mDerivedPosition(Vector3::ZERO), 62 mDerivedScale(Vector3::UNIT_SCALE), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialOrientation(Quaternion::IDENTITY), 65 mInitialScale(Vector3::UNIT_SCALE), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate();</pre>	56	mOrientation(Quaternion::IDENTITY),	
<pre>58 mScale(Vector3::UNIT_SCALE), 59 mInheritOrientation(true), 60 mInheritScale(true), 61 mDerivedOrientation(Quaternion::IDENTITY), 62 mDerivedPosition(Vector3::ZERO), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	57	mPosition(Vector3::ZERO),	
<pre>59 mInheritOrientation(true), 60 mInheritScale(true), 61 mDerivedOrientation(Quaternion::IDENTITY), 62 mDerivedPosition(Vector3::ZERO), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialCoale(Vector3::UNIT_SCALE), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate();</pre>	58	mScale(Vector3::UNIT_SCALE),	
<pre>60 mInheritScale(true), 61 mDerivedOrientation(Quaternion::IDENTITY), 62 mDerivedPosition(Vector3::ZERO), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialScale(Vector3::UNIT_SCALE), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	59	mInheritOrientation(true),	
<pre>61 mDerivedOrientation(Quaternion::IDENTITY), 62 mDerivedPosition(Vector3::ZERO), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate();</pre>	60	mInheritScale(true),	
<pre>62 mDerivedPosition(Vector3::ZERO), 63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 needUpdate();</pre>	61	mDerivedOrientation(Quaternion::IDENTITY),	
<pre>63 mDerivedScale(Vector3::UNIT_SCALE), 64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	62	mDerivedPosition(Vector3::ZERO),	
<pre>64 mInitialPosition(Vector3::ZERO), 65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	63	mDerivedScale(Vector3::UNIT_SCALE),	
<pre>65 mInitialOrientation(Quaternion::IDENTITY), 66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	64	mInitialPosition(Vector3::ZERO),	
<pre>66 mInitialScale(Vector3::UNIT_SCALE), 67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	65	mInitialOrientation(Quaternion::IDENTITY),	
<pre>67 mCachedTransformOutOfDate(true), 68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	66	mInitialScale(Vector3::UNIT_SCALE),	
<pre>68 mListener(0), 69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate();</pre>	67	mCachedTransformOutOfDate(true),	
69 mDebug(0) 70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate(); 75	68	mListener(0),	
<pre>70 { 71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate(); 75</pre>	69	mDebug(0)	
<pre>71 // Generate a name 72 mName = msNameGenerator.generate(); 73 74 needUpdate(); 75</pre>	70	{	
<pre>72 mName = msNameGenerator.generate(); 73 74 needUpdate(); 75</pre>	71	// Generate a name	
<pre>73 74 needUpdate(); 75</pre>	72	<pre>mName = msNameGenerator.generate();</pre>	
74 needUpdate();	73		
75	74	needUpdate();	
	75		
76 }	76	}	

45	namespace Ogre {	Are we done with the constructor?
47	NameGenerator Node::msNameGenerator("Unnamed_");	
48	Node::QueuedUpdates Node::msQueuedUpdates;	(5) Over-generalization
49	Node::Nede()	
51	mParent(0).	
52	mNeedParentUpdate(false).	
53	mNeedChildUpdate(false).	
54	mParentNotified(false),	
55	mQueuedForUpdate(false),	
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZER0),	
58	mScale(Vector3::UNIT_SCALE),	
59	mInheritOrientation(true),	
60	mInheritScale(true),	
61	<pre>mDerivedOrientation(Quaternion::IDENTITY),</pre>	
62	mDerivedPosition(Vector3::ZERO),	
63	<pre>mDerivedScale(Vector3::UNIT_SCALE),</pre>	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(0)	
70	{	
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

45	namespace Ogre {	Are we done with the constructor?
40 47 48	NameGenerator Node::msNameGenerator("Unnamed Node::QueuedUpdates Node::msQueuedUpdate	(5) Over-generalization
49 50 51	Node::Node() mParent(0),	
53	mNeedChildUpdate(false).	Complex constructors tend to imply that
54	mParentNotified(false),	
55	mQueuedForUpdate(false),	- Reads are unmanaged (one at a time)
56	mOrientation(Quaternion::IDENTIT	Υ),
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	
59	mInheritOrientation(true),	
60	mInheritScale(true),	
61	mDerivedOrientation(Quaternion::	IDENTITY),
62	mDerivedPosition(Vector3::ZERO),	
63	mDerivedScale(Vector3::UNIT_SCAL	E),
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::	IDENTITY),
66	mInitialScale(Vector3::UNIT_SCAL	E),
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(Ø)	
70	{	
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

45	namespace Ogre {	Are we done with the constructor?
47 48	<pre>NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;</pre>	(5) Over-generalization
48 49 50 51 52 53 54 55 57 58 59 60 61 62 63 64 65 66 67 68	<pre>Node::QueuedUpdates Node::msQueuedUpdates; Node::Node() mParent(@), mNeedParentUpdate(false), mParentNotified(false), mQueuedForUpdate(false), mOrientation(Quaternion::IDENTITY), mPosition(Vector3::ZERO), mScale(Vector3::UNIT_SCALE), mInheritOrientation(true), mInheritScale(true), mDerivedOrientation(Quaternion::IDENTITY), mDerivedPosition(Vector3::ZERO), mDerivedPosition(Vector3::ZERO), mInitialPosition(Vector3::ZERO), mInitialPosition(Vector3::ZERO), mInitialOrientation(Quaternion::IDENTITY), mInitialOrientation(Quaternion::IDENTITY), mInitialOrientation(Quaternion::IDENTITY), mInitialScale(Vector3::UNIT_SCALE), mCachedTransformOutOfDate(true), mListener(@),</pre>	Complex constructors tend to imply that - Reads are unmanaged (one at a time) - Unnecessary reads/writes in destructors
69 70 71 72	<pre>mDebug(0) {     // Generate a name     mName = msNameGenerator.generate();</pre>	
73 74 75 76	<pre>needUpdate(); }</pre>	
45	namespace Ogre {	Are we done with the constructor?
---	---	--
40 47 48	<pre>NameGenerator Node::msNameGenerator("Unnamed_");     Node::QueuedUpdates Node::msQueuedUpdates;</pre>	(5) Over-generalization
49 50 51 52 53 54 55 56 57 58 60 61 62 63 66 66 67 68 970 72 72	<pre>Node::Node()     mParent(@),     mParent(@),     mNeedParentUpdate(false),     mParentNotified(false),     mQueuedForUpdate(false),     mOrientation(Quaternion::IDENTITY),     mPosition(Vector3::ZERO),     mScale(Vector3::UNIT_SCALE),     mInheritScale(true),     mDerivedOrientation(Quaternion::IDENTITY),     mDerivedPosition(Vector3::ZERO),     mDerivedScale(Vector3::UNIT_SCALE),     mInitialPosition(Vector3::ZERO),     mInitialOrientation(Quaternion::IDENTITY),     mInitialScale(Vector3::UNIT_SCALE),     mInitialOrientation(Quaternion::IDENTITY),     mInitialScale(Vector3::UNIT_SCALE),     mInitialScale(Vector3::UNIT_SCALE),     mInitialScale(Vector3::UNIT_SCALE),     mInitialScale(Vector3::UNIT_SCALE),     mInitialScale(Vector3::UNIT_SCALE),     mListener(@),     mDebug(@)     {         // Generate a name         mName = msNameGenerator.generate();         // Generate();         // Generate();</pre>	Complex constructors tend to imply that - Reads are unmanaged (one at a time) - Unnecessary reads/writes in destructors - Unmanaged icache (i.e. virtuals) => unmanaged reads/writes
74 75 76	<pre>needUpdate(); }</pre>	

NameGenerator Node::msNameGenerator("Unnamed\_"); Node::QueuedUpdates Node::msQueuedUpdates;

(5) Over-generalization

Nod	::Node()
	mParent(0), mNeedParentUpdate(false).
	mNeedChildUpdate(false),
	mParentNotified(false),
	mQueuedForUpdate(false),
	mOrientation(Quaternion::IDENTITY),
	mPosition(Vector3::ZERO),
-	mScale(Vector3::UNIT_SCALE),
	mInheritOrientation(true),
	mInheritScale(true),
	<pre>mDerivedOrientation(Quaternion::IDENTITY),</pre>
	<pre>mDerivedPosition(Vector3::ZERO),</pre>
	mDerivedScale(Vector3::UNIT_SCALE),
	mInitialPosition(Vector3::ZERO),
	mInitialOrientation(Quaternion::IDENTITY),
Ľ	mInitialScale(vectors::UNII_SCALE), mCachedTransformOutOfDate(true), mListener(0).
	mDebug(0)
{	<pre>// Generate a name nName = msNameGenerator.generate();</pre>
	needUpdate();

Complex constructors tend to imply that...

- Reads are unmanaged (one at a time...)
- Unnecessary reads/writes in destructors
- Unmanaged icache (i.e. virtuals)
   => unmanaged reads/writes
- Unnecessarily complex state machines (back to bools)
  - E.g. 2^7 states

-1-1

NameGenerator Node::msNameGenerator("Unnamed\_"); Node::QueuedUpdates Node::msQueuedUpdates;

(5) Over-generalization

Node::Node()	
	mParent(0),
n	nNeedParentUpdate(false),
n	nNeedChildUpdate(false),
n	nParentNotified(false),
mQueuedFo	orUpdate(false),
n	nOrientation(Quaternion::IDENTITY),
n	<pre>Position(Vector3::ZER0),</pre>
n	Scale(Vector3::UNIT_SCALE),
mInherit(	Drientation(true),
n	nInheritScale(true),
n	<pre>DerivedOrientation(Quaternion::IDENTITY),</pre>
n	nDerivedPosition(Vector3::ZERO),
n	nDerivedScale(Vector3::UNIT_SCALE),
n	nInitialPosition(Vector3::ZERO),
n	<pre>iInitialOrientation(Quaternion::IDENTITY),</pre>
n	initialScale(vectors::UNII_SCALE),
n	<pre>hCachedTransformOutOfDate(true), hListener(@).</pre>
n	Debug(0)
{	
// Genero	ate a name
mName = n	nsNameGenerator.generate();
needUpdat	:e();

Сс - -	omplex constructors tend to imply that Reads are unmanaged (one at a time) Unnecessary reads/writes in destructors
-	Unmanaged icache (i.e. virtuals) => unmanaged reads/writes
-	Unnecessarily complex state machines (back to bools) - E.g. 2^7 states

Rule of thumb: Store each state type separately Store same states together (No state value needed)

45	namespace Ogre {	Are we done with the constructor?
40 47 48 49	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(5) Over-generalization
50	Node::Node()	
51	:mParent(0),	(6) Undefined or under-defined constraints
52	mNeedParentUpdate(false),	
53	mNeedChildUpdate(false),	
54	mParentNotified(false),	
55	mQueuedForUpdate(false),	
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	
59	mInheritOrientation(true),	
60	mInheritScale(true),	
61	mDerivedOrientation(Quaternion::IDENTITY),	
62	mDerivedPosition(Vector3::ZERO),	
63	mDerivedScale(Vector3::UNIT_SCALE),	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(Ø)	
70		
71	// Generate a name	
72	mixame = msixameGenerator.generate();	
73	noodlindata():	
74	needopdate();	
75	1	
10	1	

45	namespace Ogre {	Are we done with the constructor?
47 48 49	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(5) Over-generalization
50 51 52	Node::Node() :mParent(0), mNeedParentUpdate(false),	(6) Undefined or under-defined constraints
53 54 55	<pre>mNeedChildUpdate(false), mParentNotified(false), mQueuedForUpdate(false),</pre>	
56 57 58	<pre>mOrientation(Quaternion::IDENTITY), mPosition(Vector3::ZERO), mScale(Vector3::UNIT_SCALE)</pre>	Imply more (wasted) reads because pretending you don't know what it could be.
59 60	mInheritOrientation(true),	
61 62	mDerivedPosition(Vector3::ZERO)	
63 64	<pre>mDerivedScale(Vector3::UNIT_SCALE), mInitialPosition(Vector3::ZER0).</pre>	
65 66	<pre>mInitialOrientation(Quaternion::IDENTITY), mInitialScale(Vector3::UNIT_SCALE),</pre>	
67 68	<pre>mCachedTransformOutOfDate(true), mListener(0),</pre>	
69 70	mDebug(0)	
71 72 73	<pre>// Generate a name mName = msNameGenerator.generate();</pre>	
74 75 76	<pre>needUpdate(); }</pre>	

45	namespace Ogre {	Are we done with the constructor?
47 48 49	<pre>NameGenerator Node::msNameGenerator("Unnamed_");     Node::QueuedUpdates Node::msQueuedUpdates;</pre>	(5) Over-generalization
50	Node::Node()	(C) the define of an up down define of a sector inte
52	mNeedParentlindate(false)	(6) Underined of under-defined constraints
53	mNeedChildUpdate(false).	
54	mParentNotified(false).	
55	mQueuedForUpdate(false),	
56	mOrientation(Quaternion::IDENTITY),	Imply more (wasted) reads because pretending you
57	mPosition(Vector3::ZERO),	don't know what it could be
58	mScale(Vector3::UNIT_SCALE),	don't know what it could be.
59	mInheritOrientation(true),	
60	mInheritScale(true),	e.g. Strings, generally, Filenames, in particular
61	mDerivedOrientation(Quaternion::IDENTITY),	
62	mDerivedPosition(Vector3::2ERU),	
64	mDerivedScale(Vectors::UNII_SCALE),	
65	mInitialOpiontation(Vector52ERO),	
66	mInitialScale(Vector3::UNIT_SCALE)	
67	mCachedTransformOutOfDate(true)	
68	mListener(0).	
69	mDebug(0)	
70	{	
71	// Generate a name	
72	<pre>mName = msNameGenerator.generate();</pre>	
73		
74	needUpdate();	
75		
76	}	

45	namespace Ogre {	Are we done with the constructor?
47 48 49	NameGenerator Node::msNameGenerator("Unnamed_"); Node::QueuedUpdates Node::msQueuedUpdates;	(5) Over-generalization
50 51 52	Node::Node() :mParent(0), mNeedParentUpdate(false),	(6) Undefined or under-defined constraints
53 54 55	<pre>mNeedChildUpdate(false), mParentNotified(false), mOueuedForUpdate(false).</pre>	
56 57 58	<pre>mOrientation(Quaternion::IDENTITY), mPosition(Vector3::ZER0), mScale(Vector3::UNIT_SCALE)</pre>	Imply more (wasted) reads because pretending you don't know what it could be.
59 60 61	<pre>mInheritOrientation(true), mInheritScale(true), mDerivedOrientation(Quaternion::IDENTITY),</pre>	e.g. Strings, generally. Filenames, in particular.
62 63	<pre>mDerivedOrtentation(QuaternioniDENTITY), mDerivedPosition(Vector3::ZERO), mDerivedScale(Vector3::UNIT_SCALE), mInitialOrientation(Quaternion::IDENTITY), mInitialScale(Vector3::UNIT_SCALE), mCachedTransformOutOfDate(true),</pre>	
65 66 67		Rule of thumb:
68 69 70	mListener(0), mDebug(0) {	The best code is code that doesn't need to exist. Do it offline. Do it once.
71 72 73	<pre>// Generate a name mName = msNameGenerator.generate();</pre>	
74 75 76	<pre>needUpdate(); }</pre>	

45	namespace Ogre {	Are we done with the constructor?
40 47 48 49	<pre>NameGenerator Node::msNameGenerator("Unnamed_");     Node::QueuedUpdates Node::msQueuedUpdates;</pre>	(5) Over-generalization
50	Node::Node()	(6) Undefined or under-defined constraints
52	mNeedParentUpdate(false),	(b) ondermed of under-defined constraints
53	mNeedChildUpdate(false),	
54	mParentNotified(false),	(7) Over colving (computing too much)
55	mQueuedForUpdate(false),	
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNII_SCALE),	
59	mInheritScale(true)	Compiler doesn't have enough context to know
61	mDerivedOrientation(Ougternion::IDENTITY)	how to simplify your problems for you
62	mDerivedPosition(Vector3::ZERO).	now to simplify your problems for you.
63	mDerivedScale(Vector3::UNIT_SCALE),	
64	mInitialPosition(Vector3::ZERO),	
65	mInitialOrientation(Quaternion::IDENTITY),	
66	mInitialScale(Vector3::UNIT_SCALE),	
67	mCachedTransformOutOfDate(true),	
68	mListener(0),	
69	mDebug(Ø)	
70	1	
72	mlana - mslamaConorator apponato();	
72	minume = msinumedenerator.generate(),	
74	needUpdate();	
76	}	

45	namespace Ogre {	Are we done with the constructor?
40 47 48 49	<pre>NameGenerator Node::msNameGenerator("Unnamed_");     Node::QueuedUpdates Node::msQueuedUpdates;</pre>	(5) Over-generalization
50 51 52	Node::Node() :mParent(0), mNeedParentUpdate(false),	(6) Undefined or under-defined constraints
53 54 55	mNeedChildUpdate(false), mParentNotified(false), mQueuedForUpdate(false), mOniontation(Ougtornion::IDENTITY)	(7) Over-solving (computing too much)
50 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71	<pre>6 mOrientation(Quaternion::IDENTITY), 7 mPosition(Vector3::ZERO), 8 mScale(Vector3::UNIT_SCALE), 9 mInheritOrientation(true), 9 mInheritScale(true), 1 mDerivedOrientation(Quaternion::IDENTITY), 7 mDerivedPosition(Vector3::ZERO), 7 mDerivedScale(Vector3::UNIT_SCALE), 7 mInitialPosition(Vector3::ZERO), 7 mInitialOrientation(Quaternion::IDENTITY), 7 mInitialScale(Vector3::UNIT_SCALE), 7 mInitialScale(Vector3::UNIT_SCALE), 7 mListener(0), 8 mListener(0), 9 mDebug(0) 7 {</pre>	Compiler doesn't have enough context to know how to simplify your problems for you. But you can make simple tools that do - E.g. Premultiply matrices
72 73 74 75 76	<pre>mName = msNameGenerator.generate(); needUpdate(); }</pre>	

45	namespace Ogre {	Are we done with the constructor?
46		
47	NameGenerator Node::msNameGenerator("Unnamed_");	
48	Node::QueuedUpdates Node::msQueuedUpdates;	(5) Over-generalization
49	//	
50	Node::Node()	
51	:mParent(0),	(6) Undefined or under-defined constraints
52	mNeedParentUpdate(false),	
53	mNeedChildUpdate(false),	
54	mParentNotified(false),	(7) Over-solving (computing too much)
55	mQueuedForUpdate(false),	
56	mOrientation(Quaternion::IDENTITY),	
57	mPosition(Vector3::ZERO),	
58	mScale(Vector3::UNIT_SCALE),	
59	mInheritOrientation(true),	Compiler doesn't have enough context to know
60	mInheritScale(true),	Complier doesn't have enough context to know
61	mDerivedOrientation(Quaternion::IDENTITY),	how to simplify your problems for you.
62	mDerivedPosition(Vector3::2ERO),	
63	mDerivedScale(vector3::UNII_SCALE),	Dut us a second second stands to all that do
64	minitialPosition(vector5::2ERU),	But you can make simple tools that do
60	minitialOrientation(Quaternion::IDENTITY),	<ul> <li>E.g. Premultiply matrices</li> </ul>
67	minitialScale(vectors::UNIT_SCALE),	<b>o i</b> <i>i i</i>
68	mistener(0)	
60	mDebug(Q)	Work with the (actual) data you have.
70	s	- E.g. Sparse or affine matrices
71	1/ Cenerate a name	
72	mName = msNameGenerator generate():	
73	invalle = insidanederier acor .generace(),	
74	needIndate():	
75	needopudee();	
76	3	
	11 · · · ·	

### Finish your derivations, please

October 21, 2010

http://fgiesen.wordpress.com/2010/10/21/finish-your-derivations-please/

Is the compiler going to transform this...

```
float alpha = max( acos( dot( v, n ) ), acos( dot( l, n ) ) );
float beta = min( acos( dot( v, n ) ), acos( dot( l, n ) ) );
C = sin(alpha) * tan(beta);
```

Into this... for you?

```
float vdotn = dot(v, n);
```

```
float ldotn = dot(l, n);
```

```
C = sqrt((1.0 - vdotn*vdotn) * (1.0 - ldotn*ldotn))
```

```
/ max(vdotn, ldotn);
```





While we're on the subject... DESIGN PATTERNS:

Design patterns are spoonfeed material for brainless programmers incapable of independent thought, who will be resolved to producing code as mediocre as the design patterns they use to create it.

> http://realtimecollisiondetection.net/blog/?p=81 http://realtimecollisiondetection.net/blog/?p=44

Okay... Now a quick pass through some other functions.

```
439
              void Node::translate(const Vector3& d, TransformSpace relativeTo)
440
441
         {
442
            switch(relativeTo)
443
444
            case TS_LOCAL:
445
                // position is relative to parent so transform downwards
                mPosition += mOrientation * d;
446
447
                    break;
448
            case TS_WORLD:
                // position is relative to parent so transform upwards
449
450
                if (mParent)
451
                {
                   mPosition += (mParent->_getDerivedOrientation().Inverse() * d)
452
                       / mParent->_getDerivedScale();
453
454
                }
455
                else
456
                Ł
457
                   mPosition += d;
458
                }
459
                   break:
460
            case TS_PARENT:
                mPosition += d;
461
462
                break;
463
             }
            needUpdate();
464
465
         }
466
407
         12
```

1.00

```
1.00
439
               (2) Bools and last-minute decision making
         void Node::translate(const Vector3& d, TransformSpace relativeTo)
440
441
         {
             switch(relativeTo)
442
443
             case TS_LOCAL:
444
445
                 // position is relative to parent so transform downwards
                 mPosition += mOrientation * d;
446
447
                     break;
448
             case TS_WORLD:
                 // position is relative to parent so transform upwards
449
450
                 if (mParent)
451
                 Ł
                    mPosition += (mParent->_getDerivedOrientation().Inverse() * d)
452
453
                         / mParent->_getDerivedScale();
454
                 }
455
                 else
456
                 ł
457
                    mPosition += d;
458
                 }
459
                    break:
460
             case TS_PARENT:
                 mPosition += d;
461
462
                 break;
463
             }
             needUpdate();
464
465
466
         }
407
          12
```

```
void
NodesTranslateLocal( Node* nodes, int count, const Vector3& d )
 for (int i=0;i(count;i++)
   Node * node = &nodes[i];
   node-\mbox{m_Position} += \mbox{node} - \mbox{m_Orientation} * d;
 >
void
NodesTranslateWorld( Node* nodes, int count, const Vector3& d )
 for (int i=0;i(count;i++)
  ₹.
   Node + node = &nodes[i];
   if ( node->m_Parent )
      node->m_Position += ( node->m_Parent->_getDerivedOrientation().Inverse() * d >;
                          / node->m_Parent->_getDerivedScale();
    з
   else
    ٢.
      node->m_Position += d;
    З
 >
void
NodesTranslateParent( Node* nodes, int count, const Vector3& d )
 for (int i=0;i(count;i++)
 ۲.
   Node * node = &nodes[i];
   node->m_Position += d;
 >
```

# Separate states so you can reason about them

```
void
NodesTranslateLocal( Node* nodes, int count, const Vector3& d )
 for (int i=0;i(count;i++)
   Node * node = &nodes[i];
   node-\mbox{m_Position} += \mbox{node} - \mbox{m_Orientation} * d;
                                                                              Separate states so you can reason about them
 >
void
NodesTranslateWorld< Node* nodes, int count, const Vector3& d >
                                                                              What are the relative values of each case
 for (int i=0;i(count;i++)
 ۲.
   Node + node = &nodes[i];
   if < node->m_Parent >
     node->m_Position += ( node->m_Parent->_getDerivedOrientation().Inverse() * d );
                          / node->m_Parent->_getDerivedScale();
    з
   else
   ۲
     node->m_Position += d;
   З
 >
void
NodesTranslateParent( Node* nodes, int count, const Vector3& d )
 for (int i=0;i(count;i++)
 ۲
   Node * node = &nodes[i];
   node->m_Position += d;
 >
```

```
void
NodesTranslateLocalEach< Node* nodes, int count, const Vector3* t >
 for (int i=0;i(count;i++)
   Node # node = &nodes[i];
                                                                                 Step 1: organize
   Vec3& d
               = *t[i];
   node \rightarrow m_Position += node \rightarrow m_Orientation * d;
                                                                                 Separate states so you can reason about them
 3
void
                                                                                 Step 2: triage
What are the relative values of each case
 for (int i=0;i(count;i++)
                                                                                            * count
 <
   Node + node = &nodes[i];
   Vec3& d
               = *t[i];
   if ( node->m_Parent )
     node \rightarrow m_Position += \langle node \rightarrow m_Parent \rightarrow getDerivedOrientation \langle \rangle. Inverse \langle \rangle * d \rangle;
                         / node->m_Parent->_getDerivedScale();
   3
   else
     node->m_Position += d;
   3
 >
void
NodesTranslateParentEach( Node* nodes, int count, const Vector3* t )
                                                                                                  e.g. in-game vs. in-editor
 for (int i=0;i(count;i++)
   Node * node = &nodes[i];
   Vec3& d
               = *t[i];
   node->m_Position += d;
 >
```

```
void
NodesTranslateLocal( Node* nodes, int count, const Vector3& d )
 for (int i=0;i(count;i++)
    Node * node = &nodes[i];
   node-\mbox{m_Position} += \mbox{node} - \mbox{m_Orientation} * d;
  3
```

```
void
NodesTranslateWorld( Node* nodes, int count, const Vector3& d )
 for (int i=0;i(count;i++)
 ۲.
   Node + node = &nodes[i];
   if ( node->m_Parent )
     node->m_Position += ( node->m_Parent->_getDerivedOrientation().Inverse() * d >;
                         / node->m_Parent->_getDerivedScale();
   з
                                                                            Step 3: reduce waste
   else
   ٢.
     node->m_Position += d;
   З
 >
void
NodesTranslateParent( Node* nodes, int count, const Vector3& d )
 for (int i=0;i(count;i++)
 ۲
   Node * node = &nodes[i];
   node->m_Position += d;
 >
```

## Separate states so you can reason about them

What are the relative values of each case



(back of the envelope read cost)

~200 cycles x 2 x count

```
void
                                                                   (back of the envelope read cost)
 NodesTranslateLocal( Node* nodes, int count, const Vector3& d )
  for (int i=0;i(count;i++)
                                                                   ~200 cycles x 2 x count
    node->m_Position += m_Orientation * d;
struct NodeTranslate
  Vec3 m_Position;
  Quat m_Orientation;
                                                                                        ~2.28 count per 200 cycles
₿;
                                                                                        =~88
void
NodesTranslateLocal( NodeTranslate* nodes, int count, const Vector3& d )
  for (int i=0;i(count;i++)
  {
   node->m_Position += node->m_Orientation * d;
  >
```





```
void
NodesTranslateWorldEach( Node* nodes, int count, const Vector3* t )
 for (int i=0;i(count;i++)
 ٤
   Node* node = &nodes[i];
   Vec3& d
               = t[i];
   if < node->m_Parent >
   ۲
     node->m_Position += < node->m_Parent->_getDerivedOrientation<>.Inverse<> * d >;
                         / node->m_Parent->_getDerivedScale();
   >
   else
   ۲
     node->m_Position += d;
   >
 >
```



Root or not; Calling function with context can distinguish



Root or not; Calling function with context can distinguish

```
void
NodesTranslateWorldEachRoot( Node* nodes, int count, const Vector3* t )
 for (int i=0;i(count;i++)
 ۲
                                                                     Step 1: organize
    Node * node = &nodes[i];
                                                                     Separate states so you can reason about them
    Vec3& d
                = t[i];
    node->m_Position += d;
 >
void
NodesTranslateWorldEachWithParent( Node* nodes, int count, const Vector3* t )
 for (int i=0;i(count;i++)
 ۲
    Node * node = &nodes[i];
    Vec3& d
                = t[i];
    node \rightarrow m_Position += ( node \rightarrow m_Parent \rightarrow getDerivedOrientation(). Inverse() * d );
                         / node->m_Parent->_getDerivedScale<>;
```

```
void
NodesTranslateWorldEachRoot( Node* nodes, int count, const Vector3* t )
 for (int i=0;i(count;i++)
 ۲
                                                                 Step 1: organize
    Node * node = &nodes[i];
                                                                 Separate states so you can reason about them
   Vec3& d
               = t[i];
   node->m_Position += d;
 3
void
NodesTranslateWorldEachWithParent( Node* nodes, int count, const Vector3* t )
 for (int i=0;i(count;i++)
 ۲
    Node* node = &nodes[i];
    Vec3& d
                  t[i];
               Π
   node->m_Position +=
                         ( node->m_Parent->_getDerivedOrientation(>.Inverse(> * d >;
                         node->m_Parent->_getDerivedScale<>;
```

Can't reason well about the cost from ...

```
NodeParent
              DerivedOrientationInverse
    Quat
    float
              DerivedScale
                                                                      Step 1: organize
    uint32_t ChildCount
              ChildPosition[]
    Vector3
                                                                      Separate states so you can reason about them
void
NodesTranslateWorldEachWithParent( char* nodes, int parentCount, const Vector3* t )
 int k=0;
 for (int i=0;i(parentCount;i++)
 ۲
   Quat*
            derivedOrientationInverse = (Quat*)nodes;
   nodes += sizeof(Quat);
            derivedScale
                                       = *(float*)nodes;
   float
   nodes += sizeof(float);
                                       = *(uint32_t*)nodes;
   uint32_t childCount
   nodes += sizeof(uint32_t)
```

```
for (int j=0;j<childCount;j++,k++)
{
    Vector3& d = t[k];
    Vector3& childPosition = *(Vector3*)nodes;
    nodes += sizeof(Vector3);</pre>
```

>

```
childPosition += (derivedOrientationInverse * d) / derivedScale;
}
```

```
NodeParent
               DerivedOrientationInverse
    Quat
    float
               DerivedScale
                                                                       Step 1: organize
    uint32_t ChildCount
              ChildPosition[]
    Vector3
                                                                       Separate states so you can reason about them
void
NodesTranslateWorldEachWithParent( char* nodes, int parentCount, const Vector3* t )
 int k=0;
 for (int i=0;i(parentCount;i++)
                                                                       Step 2: triage
 ۲
   Quat*
            derivedOrientationInverse = (Quat*)nodes;
                                                                        What are the relative values of each case
   nodes += sizeof(Quat);
                                                                        i.e. p(call) * count
            derivedScale
                                       = *(float*)nodes;
   float
   nodes += sizeof(float);
   uint32_t childCount
                                       = *(uint32_t*)nodes;
   nodes += sizeof(uint32_t)
                                                                        Step 3: reduce waste
   for (int j=0;j(childCount;j++,k++)
                             = t[k];
     Vector3& d
     Vector3& childPosition = *(Vector3*)nodes;
     nodes += sizeof(Vector3);
     childPosition += (derivedOrientationInverse * d) / derivedScale;
   >
 >
```

#### And here...

----

```
000
         509
         void Node::rotate(const Quaternion& q, TransformSpace relativeTo)
510
511
         {
                    // Normalise guaternion to avoid drift
512
513
                    Quaternion qnorm = q;
                    qnorm.normalise();
514
515
516
            switch(relativeTo)
517
             {
518
            case TS_PARENT:
519
                // Rotations are normally relative to local axes, transform up
                mOrientation = qnorm * mOrientation;
520
521
                break:
522
            case TS_WORLD:
                // Rotations are normally relative to local axes, transform up
523
                mOrientation = mOrientation * _getDerivedOrientation().Inverse()
524
525
                    * gnorm * _getDerivedOrientation();
526
                break;
527
            case TS_LOCAL:
                // Note the order of the mult, i.e. g comes after
528
529
                mOrientation = mOrientation * gnorm;
530
                break;
531
             }
532
            needUpdate();
533
         }
```

# Before we close, let's revisit...

```
struct FooUpdateIn {

                                                12 bytes x count(32) = 384 = 64 \times 6
    float m_Velocity[2];
    float m Foo;
                                                 4 bytes x count(32) = 128 = 64 \times 2

struct FooUpdateOut {

    float m Foo;
 };
□ void UpdateFoos(const FooUpdateIn* in, size t count, FooUpdateOut* out, float f)
  {
   for (size t i = 0; i < count; ++i) {</pre>
      float mag = sqrtf(
        in[i].m_Velocity[0] * in[i].m_Velocity[0] +
        in[i].m_Velocity[1] * in[i].m_Velocity[1]);
        out[i].m Foo = in[i].m Foo + mag * f;
   }
```

Good News: Most problems are easy to see.

# Good News: Side-effect of solving the 90% well, compiler can solve the 10% better.

Good News: Organized data makes maintenance, debugging and concurrency much easier
## Bad News: Good programming is hard. Bad programming is easy.

## PS: Let's get more women in tech