

```
ics
& (depth < MAXDEPTH)
{
    inside / inside
    nt = nt / nc;
    cos2t = 1.0f - nt;
    D, N );
}
{
    at a = nt - nc, b = nt;
    at Tr = 1 - (R0 + (1 - R0) *
    Tr) R = (D * nnt - N * (a
    E * diffuse;
    = true;
}
efl + refr)) && (depth < MAXDEPTH)
D, N );
efl * E * diffuse;
= true;
}
MAXDEPTH)
survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following
if;
radiance = SampleLight( &rand, I, &t, &align,
e.x + radiance.y + radiance.z) > 0) && (rand
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Survive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
random walk - done properly, closely following
ive)
;
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

/INFOMOV/

Optimization & Vectorization

J. Bikker - Sep-Nov 2016 - Lecture 14: "Grand Recap"

Welcome!



```
...ics
& (depth < MAXDEPTH)
...
c = inside / (1 + refl);
nt = nt / nc; ddn = ddn / nc;
cos2t = 1.0f - nnt * nnt;
D, N );
)
...
at a = nt - nc, b = nt + nc;
at Tr = 1 - (R0 + (1 - R0) * c);
Tr) R = (D * nnt - N * (a * c + b));
...
E * diffuse;
= true;
...
efl + refr)) && (depth < MAXDEPTH)
D, N );
-efl * E * diffuse;
= true;
...
MAXDEPTH)
survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following
if;
radiance = SampleLight( &rand, I, &t, &light;
e.x + radiance.y + radiance.z) > 0) && (depth <
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Survive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
...
random walk - done properly, closely following
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf;
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

Today's Agenda:

- Exam
- Digest
- Grand Recap
- Now What



Exam

What to Study

1. Slides

2. Literature on the website:

- Designing for Performance, Scalability & Reliability: StarCraft II's Approach
- What Every Programmer Should Know About Memory
- Game Programming Patterns - Data Locality
- Data-Oriented Design (Or Why You Might Be Shooting Yourself in the Foot With OOP)
- The Neglected Art of Fixed Point Arithmetic
- A Survey of General-Purpose Computation on Graphics Hardware

3. Skills you picked up with the practical assignments

```

...
    & (depth < MAXDEPTH)
...
    t = inside / (nc + ndd);
    nt = nt / nc;
    cos2t = 1.0f - nt;
    D, N );
...
    a = nt - nc, b = nt;
    Tr = 1 - (R0 + (1 - R0) * t);
    R = (D * nnt - N * (1 - nnt));
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH)
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
...
survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following
if;
radiance = SampleLight( &rand, I, &t, &align,
e.x + radiance.y + radiance.z) > 0) && (depth <
...
    v = true;
    brdfPdf = EvaluateDiffuse( L, N ) * Recursive
    t3 factor = diffuse * INVPI;
    t3 weight = Mis2( directPdf, brdfPdf );
    t3 cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / directPdf) * (radiance
...
random walk - done properly, closely following
ive)
...
t3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Exam

You may bring a dictionary to the exam.
 You may **not** bring notes to the exam.
 You may bring pizza to the exam.

Example Questions

1. What is ‘false sharing’?
2. A class that consists of 28 bytes of data is padded to 32 bytes and aligned to a cache line boundary. A large array of objects of this type is then accessed in random order. Why is the padding a bad idea in this case?
3. CPUs and GPUs have fundamentally different core strategies for dealing with latencies such as memory access time. What are these strategies?
4. Why is a 'divide and conquer' algorithm in general unsuitable for GPGPU?
5. What is the purpose of the Northbridge in a modern PC architecture?
6. What is a NUMA architecture?
7. What is ‘hot/cold splitting’?
8. Explain what a super-scalar pipeline is.

Expect a limited number (~10) of open questions covering the main topics: CPU-, cache- and GPU architecture, SIMD, Data Oriented Design, fixed point math.

Some questions explicitly require that you read the literature.

You will have plenty of time to answer the questions.



```
...ics
& (depth < MAXDEPTH)
...
c = inside / (1 + refl);
nt = nt / nc; ddn = ddn / nc;
cos2t = 1.0f - nnt * nnt;
D, N );
)
...
at a = nt - nc, b = nt + nc;
at Tr = 1 - (R0 + (1 - R0) * c);
Tr) R = (D * nnt - N * (a *
...
E * diffuse;
= true;
...
efl + refr)) && (depth < MAXDEPTH)
D, N );
-refl * E * diffuse;
= true;
...
MAXDEPTH)
survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following
if;
radiance = SampleLight( &rand, I, &t, &light;
e.x + radiance.y + radiance.z) > 0) && (depth <
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Survive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
...
random walk - done properly, closely following
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf;
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

Today's Agenda:

- Exam
- Digest
- Grand Recap
- Now What



Digest

Patterns: Vectorization

Optimal use of SIMD: independent lanes in parallel, which naturally extends to 8-wide, 16-wide etc.

Optimal use of GPGPU: large number of independent tasks running in parallel.

Similar pitfalls (conditional code, dependencies / concurrency issues).

Successful algorithm conversion can yield linear speedup in number of lanes.

```

...
    & (depth < MAXDEPTH)
...
    t = inside / inside;
    nt = nt / nc;
    cos2t = 1.0f - nt;
    D, N );
...
    at a = nt - nc, b = nt;
    at Tr = 1 - (R0 + (1 - R0) * a);
    Tr) R = (D * nnt - N * (a *
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH)
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following
if;
radiance = SampleLight( @rand, I, @t, @align,
e.x + radiance.y + radiance.z) > 0) && (radiance
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * SurvivalProbability( diffuse );
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
random walk - done properly, closely following
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, @R, @pdf );
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Digest

Patterns: Vectorization

“The only correct SSE code / GPGPU program is one where many scalar threads run concurrently and independently”

(this pretty much rules out auto-vectorization by the compiler – go manual!)

(this requires suitable data structures: typically SoA)

```

...
    & (depth < MAXDEPTH)
...
    inside / inside
    nt = nt / nc;
    cos2t = 1.0f - nt;
    D, N );
    );
...
    at a = nt - nc, b = nt;
    at Tr = 1 - (R0 + (1 - R0)
    Tr) R = (D * nnt - N * (a
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH)
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
...
survive = SurvivalProbability( diffuse );
estimation - doing it properly, close
if;
radiance = SampleLight( @rand, I, &t, &align
e.x + radiance.y + radiance.z) > 0) && (survive
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Survive;
    at3 factor = diffuse * INVPI;
    at weight = Mis2( directPdf, brdfPdf );
    at cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / directPdf) * (radiant
...
random walk - done properly, closely following
ive)
...
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;

```



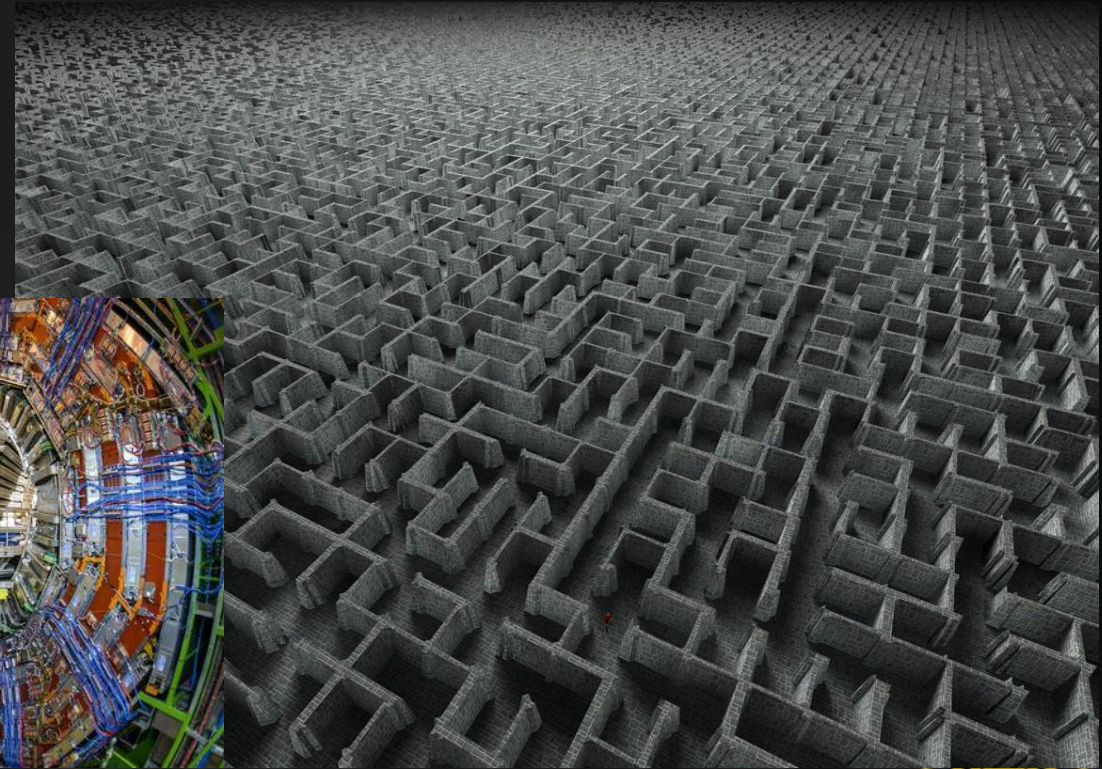
Digest

The Relevance of Low Level

Small gains?

Understanding the hardware


One more percent – Programmer’s Sudoku

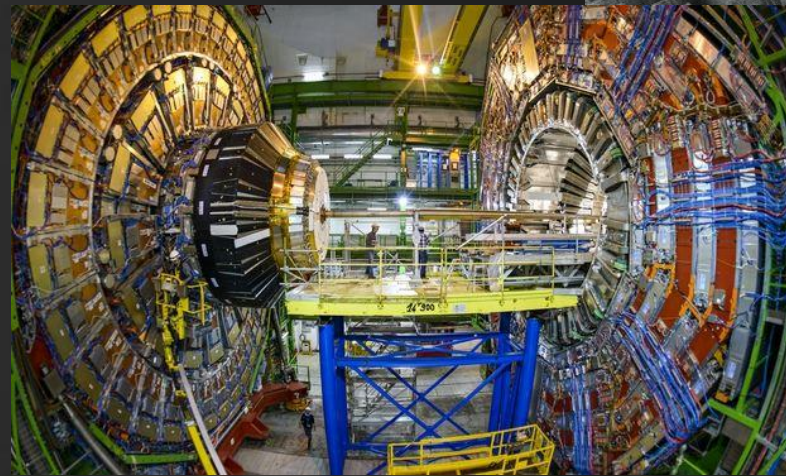


twitter Home Profile Find People Settings Help Sign out

Things that blow your mind (#WA): Why do otherwise intelligent people ignore the low hanging fruit and want to eat the whole tree first? ☆

9:32 AM Aug 6th, 2008 from web

 **avinashkaushik**
Avinash Kaushik



Digest

Multi-threading

Considered ‘trivial’ – but it isn’t

Hard to get linear speedup (typical: 2x on 8 cores...)

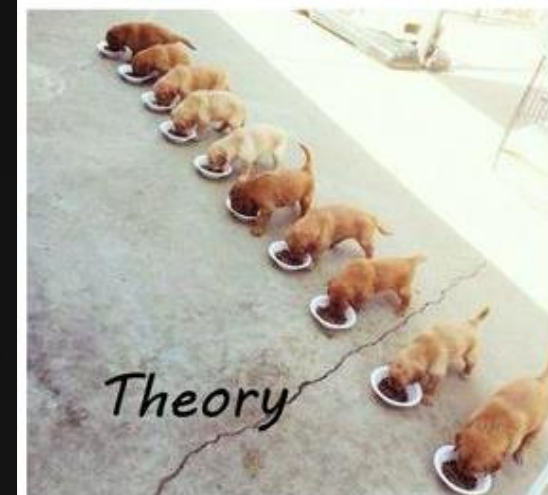
Increasingly relevant

May affect high level optimization greatly

Benefits from vectorization

Covered in other UU courses, e.g. concurrency (next block, but in bachelor).

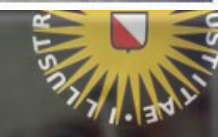
Multithreaded programming



```

...
    & (depth < MAXDEPTH)
...
    c = inside / inside;
    nt = nt / nc;
    cos2t = 1.0f - nt;
    D, N );
    );
...
    at a = nt - nc, b = nt;
    at Tr = 1 - (R0 + (1 - R0)
    Tr) R = (D * nnt - N * (a
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH)
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
...
survive = SurvivalProbability( diffuse,
estimation - doing it properly, close
if;
radiance = SampleLight( @rand, I, R1, Aligned
e.x + radiance.y + radiance.z) > 0) && (survive
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Survive;
    at3 factor = diffuse * INVPI;
    at weight = Mis2( directPdf, brdfPdf );
    at cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / directPdf) * (radiance
...
random walk - done properly, closely following
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, @R, @pdf
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Digest

Automatic Optimization

Compilers:

Not all compilers are equal

Will do a fair bit of optimization for you

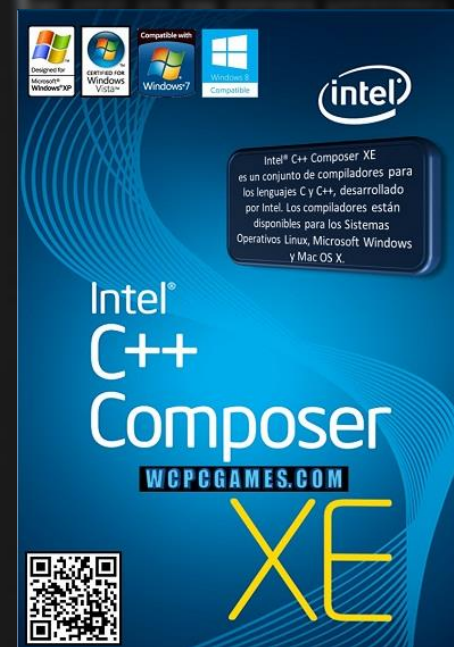
Will tune it to different processors

Will sometimes vectorize for you

But: have to be conservative

Creating optimizing compilers is a job profile

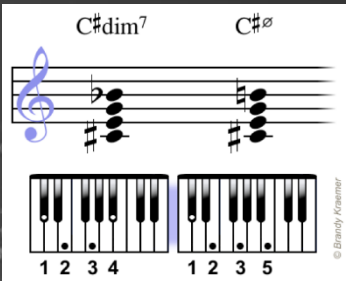
```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Matthew>arm-none-eabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-none-eabi-gcc
COLLECT_LTO_WRAPPER=c:/program files/gnu tools arm embedded/4.6.2012q2/bin/./lto-wrapper.exe
Target: arm-none-eabi
Configured with: /home/build/work/jenkins-daily-build/src/gcc/configure --build=i686-linux-gnu --host=i586-mingw32 --target=arm-none-eabi --prefix=/home/build/work/jenkins-daily-build/install-mingw --enable-languages=c,c++ --disable-decimal-float --disable-libffi --disable-libgomp --disable-libmudflap --disable-libquadmath --disable-libssp --disable-libstdc++-pch --disable-lto --disable-nls --disable-shared --disable-threads --disable-tls --with-gnu-as --with-gnu-ld --with-headers=yes --with-newlib --with-sysroot=/home/build/work/jenkins-daily-build/install-mingw/arm-none-eabi --with-libiconv-prefix=/home/build/work/jenkins-daily-build/build-mingw/host-libs/usr --with-gmp=/home/build/work/jenkins-daily-build/build-mingw/host-libs/usr --with-mpfr=/home/build/work/jenkins-daily-build/build-mingw/host-libs/usr --with-mpc=/home/build/work/jenkins-daily-build/build-mingw/host-libs/usr --with-ppl=/home/build/work/jenkins-daily-build/build-mingw/host-libs/usr --with-cloog=/home/build/work/jenkins-daily-build/build-mingw/host-libs/usr --with-libelf=/home/build/work/jenkins-daily-build/build-mingw/host-libs/usr --with-host-libstdcxx='-static-libgcc -Wl,-Bstatic,-lstdc++,-Bdynamic -lm' --with-pkgversion='GNU Tools for ARM Embedded Processors' --with-extra-multilibs=armv6-m,armv7-m,armv7e-m,armv7-r
Thread model: single
gcc version 4.6.2 20120613 (release) [ARM/embedded-4_6-branch revision 188521] <
GNU Tools for ARM Embedded Processors>
```



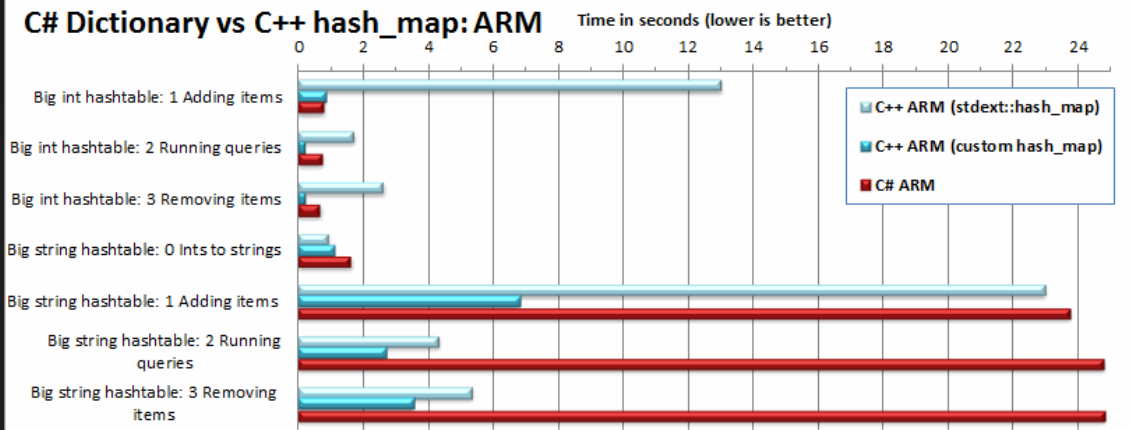
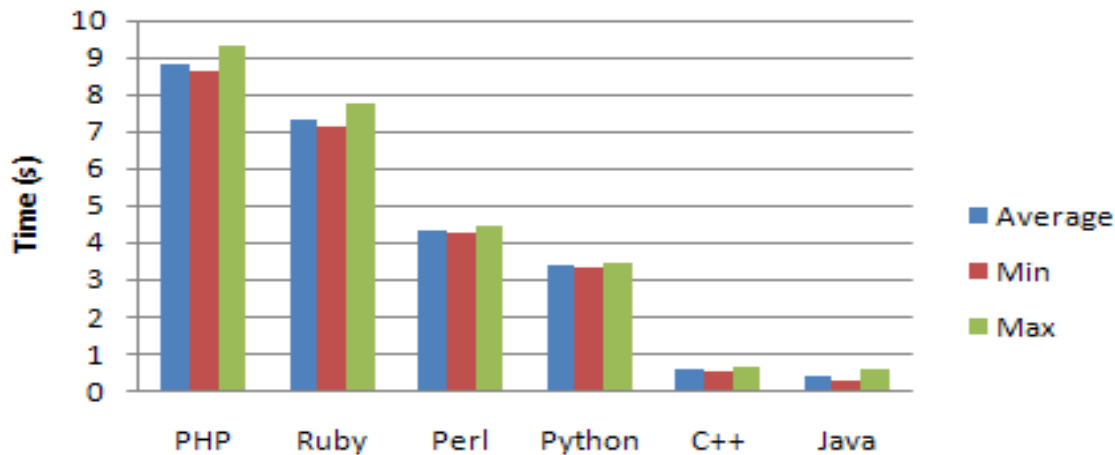
Digest

INFOMOV / C#

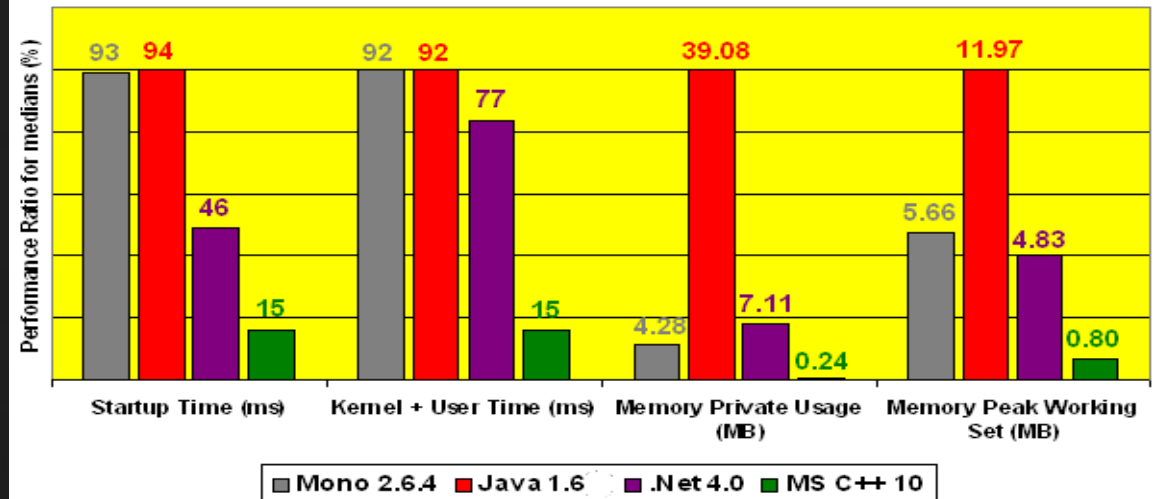
High level still works
 Profiling still works
 Some low level still works
 Performance Basis: C# versus C++



Mergesort Performance



Runtime Performance for warm starts (lower is better)



Digest

Implementation	Lang	sudoku:t	matmul:t	matmul:m
ICC-12.0.3	C	1.0	1.8	31.8
Clang@LLVM-2.9	C	1.0	2.3	31.7
GCC-4.3.2	C	1.0	2.3	31.7
C#@Mono-2.10.1	C#	3.8	8.9	40.6
GDC-0.24	D	1.1	3.3	33.9
LDC-20110428@LLVM-2.9	D	1.1	2.4	31.4
6g-20110424	Go	2.3	3.1	38.2
Java@JRE-1.6.0_25	Java	1.7	2.6	67.1
V8-r8384	Javascript	3.7	2.6	141.6
JaegerMonkey-a95d42642281	Javascript	18.1	16.4	35.8
LuaJIT-2.0.1 (JIT-on)	Lua	3.7	2.5	33.2
Lua-5.1.4	Lua	50.5	68.3	65.4
Perl-5.12.2	Perl	121.2	230.3	225.6
Shedskin-0.9@GCC-4.3.2	Python	4.4	3.7	50.4
IronPython-2.7@Mono-2.10.1	Python	100.9	202.7	190.2
Jython-2.5.2@JRE-1.6.0_25	Python	136.3	731.4	355.6
PyPy-1.4.1	Python	19.5	8.5	84.1
R-2.13.0	R	999.9	1736.3	57.2
JRuby-1.6.1@JRE-1.6.0_25	Ruby	71.1	238.2	342.5
Rubinius-1.2.3	Ruby	135.5	298.1	162.5

sudoku:t: time for solving 20 extremely hard Sudoku’s 50 times.

matmul:t: time (relative to ICC) for multiplying two 1000x1000 matrices (standard $O(N^2)$ algorithm).

matmul:m: memory (in megabytes) for multiplying two 1000x1000 matrices.

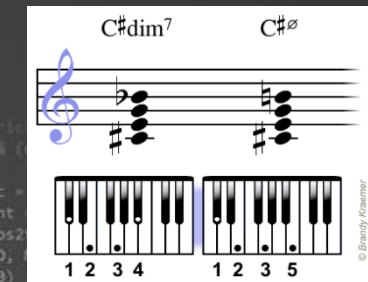
Reference:

Intel C++ compiler version 12.0.3, ‘10;

Java JRE: End of 2011;
Mono 2.1: End of 2010.



Digest



INFOMOV / C#

High level still works

Profiling still works

Some low level still works

Performance Basis: C# versus C++

C#-specific optimization:

<http://www.dotnetperls.com/optimization>

<https://www.udemy.com/csharp-performance-tricks-how-to-radically-optimize-your-code/>

<http://www.c-sharpcorner.com/UploadFile/47fc0a/code-optimization-techniques/>



Digest

The Process

10x and more – proven? (did we use realistic scenarios?)

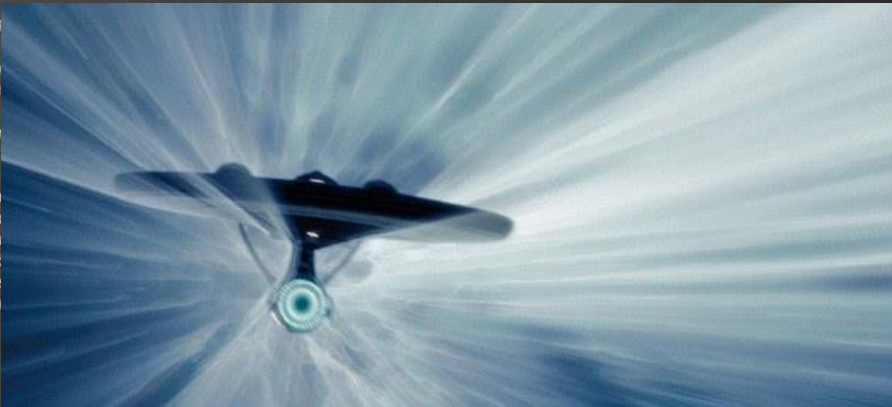
Counter-intuitive steps – attractive square roots

Importance of profiling

Is the process generic?

```

...
    & (depth < MAXDEPTH)
...
    inside / ...
    nt = nt / nc;
    pos2t = 1.0f - nnt;
    D, N );
...
    at a = nt - nc, b = nt;
    at Tr = 1 - (R0 + (1 - R0)
    Tr) R = (D * nnt - N * (1 -
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH)
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEI
survi
esti
if;
radia
e.x +
v = ti
at br
at3 fi
at we;
at co;
E *
...
andom
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, R,
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
    
```



Recap

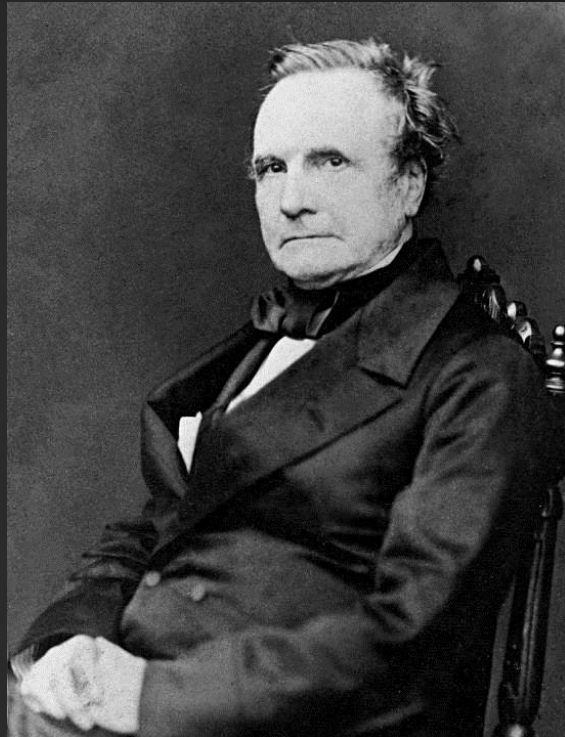
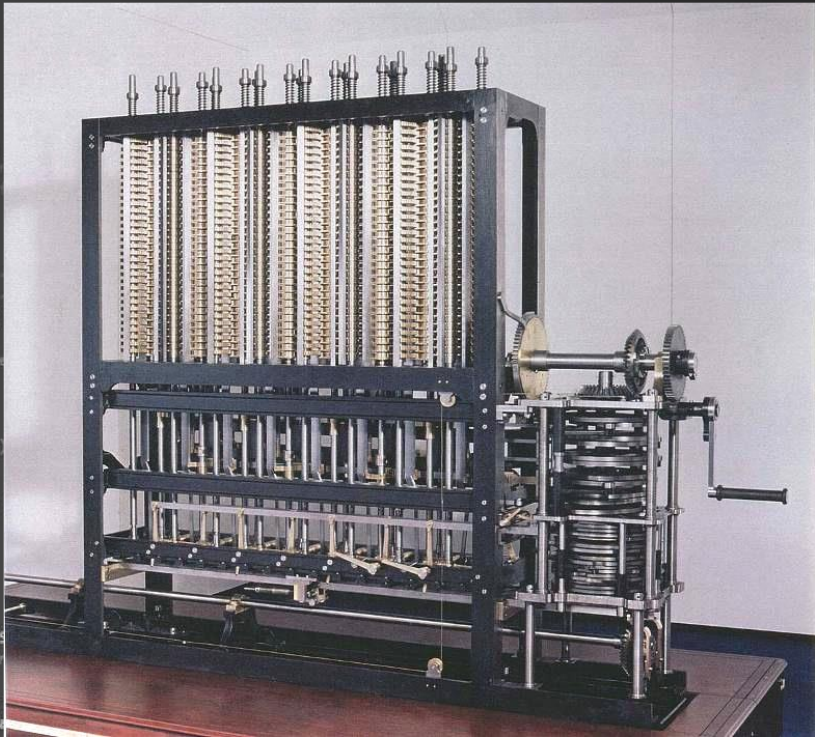
```

...
    & (depth < MAXDEPTH)
...
    c = inside / (inside + outside);
    nt = nt / nc; nct = nc / nc;
    pos2t = 1.0f - nnt;
    D, N );
    )
...
    at a = nt - nc, b = nt;
    at Tr = 1 - (RB + (1 - RB) * c);
    Tr) R = (D * nnt - N * (a + b * c));
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH);
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
...
survive = SurvivalProbability( diffuse );
estimation - doing it properly, close to
if;
radiance = SampleLight( &rand, I, R, R, R );
e.x + radiance.y + radiance.z) > 0) && (depth < MAXDEPTH);
...
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * SurvivalProbability( diffuse );
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * radiance;
...
random walk - done properly, closely following the path of the photon
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Recap – lecture 1

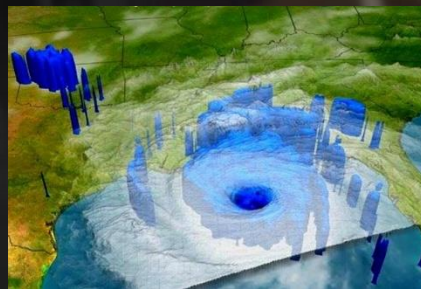


Profiling
High Level
Basic Low Level
Cache & Memory
Data-centric
Compilers
Fixed-point Arithmetic
CPU architecture
SIMD
GPGPU

```

(depth
inside
nt /
os2t = 1.0
D, N );
)
a = nt
at Tr = 1
Tr) R = (D
diffuse
true;
efl + refr
D, N );
efl * E *
= true;
MAXDEPTH)
survive = $
estimation
if;
-radiance =
e.x + radi
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * n
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (PDF
random walk - done properly, closely following
ive)
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Recap – lecture 1



Basic Hotspots Hotspots by CPU Usage viewpoint (change) Intel VTune Amplifier XE 2015

Analysis Target Analysis Type Collection Log Summary Bottom-up Caller/Callee Top-down Tree Tasks

Grouping: Function / Call Stack

Function / Call Stack	Effective Time by Utilization	Spin Time	Overhead Time
FireObject:checkCollision	4.507s	0s	0s
FireObject:ProcessFireCollisionsRange	3.444s	0s	0s
FireObject:FireCollisionCallback	3.025s	0s	0s
FireObject:EmitterParallelCollisionCheck	0.419s	0s	0s
NWaitForSingleObject	0s	3.406s	0s

Data Of Interest (CPU Metrics)

★ Viewing 1 of 49 selected stack(s)

22.8% (1.029s of 4.507s)

SystemProceduralFire...- fireobject.cpp
SystemProceduralFir...fireobject.cpp:1459
SystemProceduralFire...fireobject.cpp:1377
Smoke.exe|Parallel...managertbb.cpp:573
Smoke.exe|TBB paral... parallel_for.h:212
Smoke.exe|tbb:inter... parallel_for.h:150

Called functions

Bottom of Stack

Performance metric: Inclusive Samples %

Very Sleepy CS - C:\Users\Jacco\AppData\Local\Temp\F8M.tmp

Name	Exclu...	Inclusive	% Exclusive	% Inclusive	Module	Source File
TmpB:Game:Simulate	0.47s	5.42s	62.76%	62.76%	water	d:\water\game...
TmpB:Game:SimulateWater	2.01s	2.01s	13.34%	13.34%	water	d:\water\game...
TmpB:Game:DrawTriangle	1.13s	1.13s	8.80%	8.80%	water	d:\water\game...
TmpB:Game:RenderSprites	1.19s	1.19s	7.86%	7.86%	water	d:\water\game...
TmpB:Game:RenderWaterSurface	0.43s	1.76s	2.87%	11.67%	water	d:\water\game...
TmpB:Surface:Clear	0.11s	0.11s	0.75%	0.75%	water	d:\water\game...
TmpB:Game:RenderDebugInfo	0.03s	0.05s	0.19%	0.32%	water	d:\water\game...
TmpB:Surface:Plot	0.02s	0.02s	0.13%	0.13%	water	d:\water\game...
TmpB:Game:DownScale	0.02s	0.02s	0.10%	0.10%	water	d:\water\game...
TmpB:Game:TrendDepth	0.00s	0.00s	0.02%	0.02%	water	d:\water\game...
TmpB:Surface:AddLine	0.00s	0.00s	0.01%	0.01%	water	d:\water\game...
swap	0.00s	0.25s	0.01%	1.66%	water	d:\water\tempa...
[D666C00]	0.00s	0.00s	0.00%	0.01%	water	d:\water\game...
_mainCRTStartup	0.00s	15.08s	0.00%	99.93%	water	f:\sdk\tools\cr...
SDL_main	0.00s	15.03s	0.00%	99.57%	water	d:\water\tempa...
TmpB:Game:Tick	0.00s	13.88s	0.00%	91.96%	water	d:\water\game...
TmpB:Game:DrawBeat	0.00s	0.00s	0.00%	0.01%	water	d:\water\game...
TmpB:Game:GlowLine	0.00s	0.00s	0.00%	0.01%	water	d:\water\game...

Source Log

```

for ( int step = 0; step < 3; step++ )
{
    // simulation step 3a - satisfy constraints - evade other drops
    for ( int j = 1 + i; j < DROPCOUNT; j++ )
    {
        float dist = (drop[i].pos - drop[j].pos).Length();
        if (dist < (DROPRADIUS * 2))
        {
            vec3 direction = (drop[i].pos - drop[j].pos).Normalized();
            drop[i].pos += direction * (DROPRADIUS * 2 - dist) * 0.02f;
            drop[j].pos -= direction * (DROPRADIUS * 2 - dist) * 0.02f;
        }
    }
    // simulation step 3b - satisfy constraints - evade walls
    if (drop[i].pos.y > 20) drop[i].pos.y = 19.99f - drop[i].pos.z * 0.0001f;
    if (drop[i].pos.x < -20) drop[i].pos.x = -19.99f + drop[i].pos.z * 0.0001f;
    if (drop[i].pos.x > 20) drop[i].pos.x = 19.99f - drop[i].pos.z * 0.0001f;
    if (drop[i].pos.z < -20) drop[i].pos.z = -19.99f + drop[i].pos.z * 0.0001f;
    if (drop[i].pos.z > 20) drop[i].pos.z = 19.99f - drop[i].pos.z * 0.0001f;
}

```

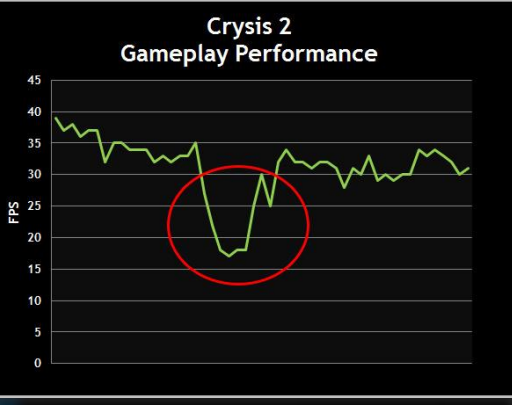
Function Code View

D:\ACTIVE\water\game.cpp

```

// simulation step 1 - move
drop[i].pos += drop[i].pos - prev_pos;
// simulation step 2 - apply gravity
drop[i].pos += gravity * 0.25f;
// simulation step 3 - satisfy constraints
for ( int step = 0; step < 3; step++ )
{
    // simulation step 3a - satisfy constraints - evade other drops
    for ( int j = i + 1; j < DROPCOUNT; j++ )
    {
        float dist = length( drop[i].pos - drop[j].pos );
        if (dist < (DROPRADIUS * 2))
        {
            vec3 direction = normalize( drop[i].pos - drop[j].pos );
            drop[i].pos += direction * (DROPRADIUS * 2 - dist) * 0.02f;
            drop[j].pos -= direction * (DROPRADIUS * 2 - dist) * 0.02f;
        }
    }
    // simulation step 3b - satisfy constraints - evade walls
    if (drop[i].pos.y > 20) drop[i].pos.y = 19.99f - drop[i].pos.z * 0.0001f;
    if (drop[i].pos.x < -20) drop[i].pos.x = -19.99f + drop[i].pos.z * 0.0001f;
    if (drop[i].pos.x > 20) drop[i].pos.x = 19.99f - drop[i].pos.z * 0.0001f;
    if (drop[i].pos.z < -20) drop[i].pos.z = -19.99f + drop[i].pos.z * 0.0001f;
    if (drop[i].pos.z > 20) drop[i].pos.z = 19.99f - drop[i].pos.z * 0.0001f;
}

```



Recap – lecture 2



Red = $u4 \& (255 \ll 16)$;
 Green = $u4 \& (255 \ll 8)$;
 Blue = $u4 \& 255$;

We need to go deeper

```

fldz
xor ecx, ecx
fld dword ptr ds:[405290h]
mov edx, 28929227h
fld dword ptr ds:[40528Ch]
push esi
mov esi, 0C350h = 50000

add ecx, edx
mov eax, 91D2A969h = 2^46 / 28763 (!!)
```

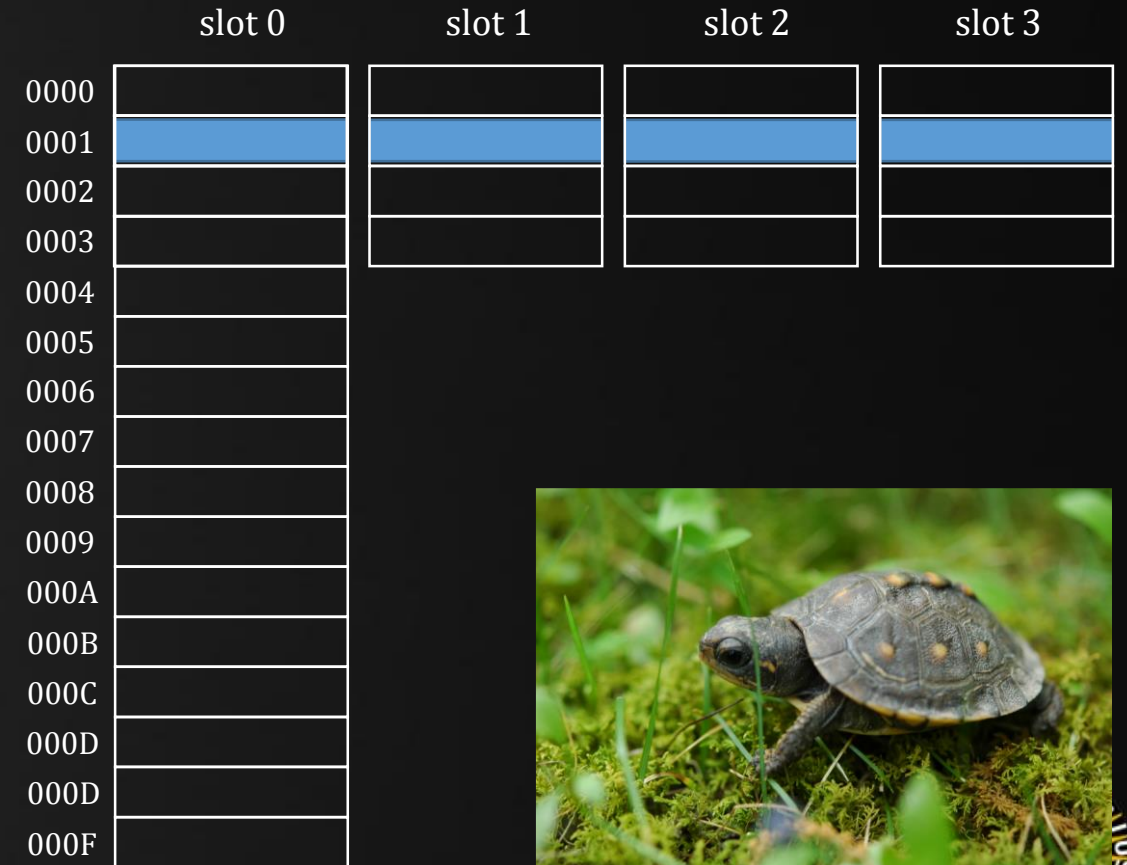
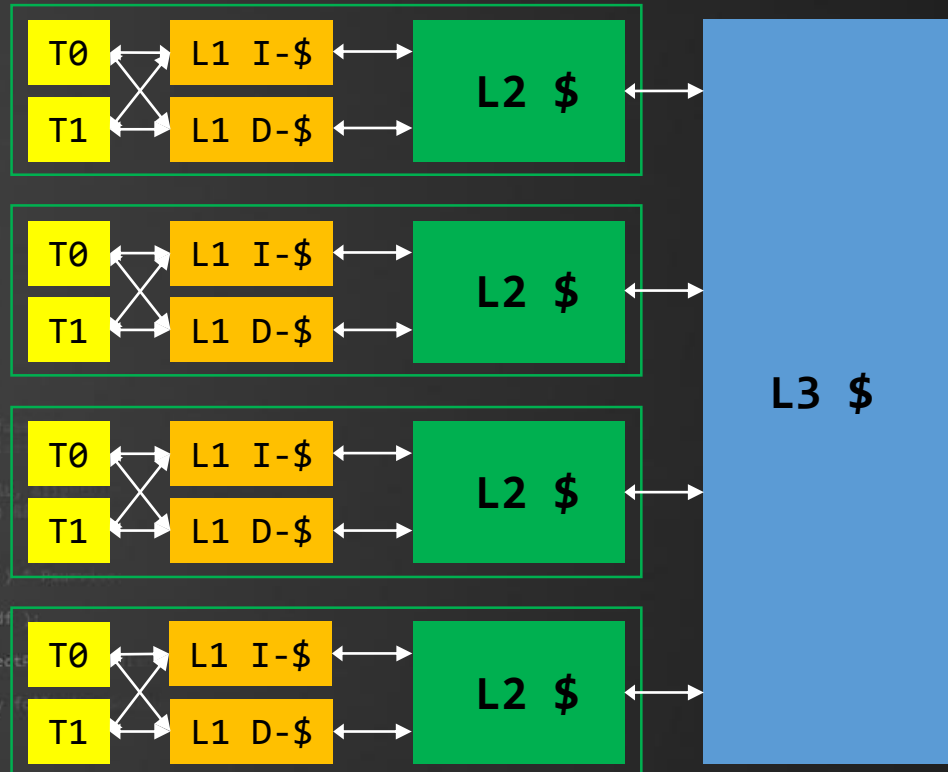
```

xor edx, 17737352h
shr ecx, 1
mul eax, edx
fld st(1)
faddp st(3), st

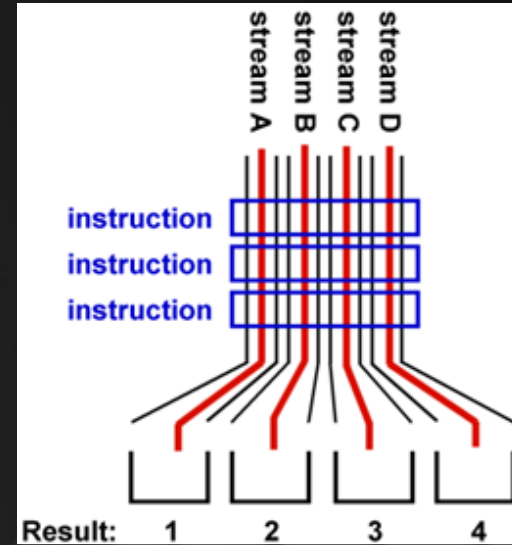
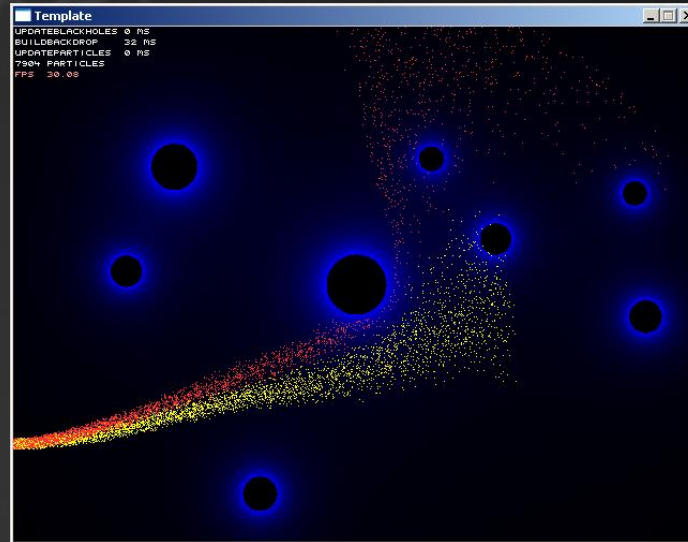
mov eax, 91D2A969h
shr edx, 0Eh
add ecx, edx
fmul st(1), st
xor edx, 17737352h
shr ecx, 1
mul eax, edx
shr edx, 0Eh
dec esi
jne tobetimed<0>+1Fh
    
```



Recap – lecture 3



Recap – lecture 6 & 7

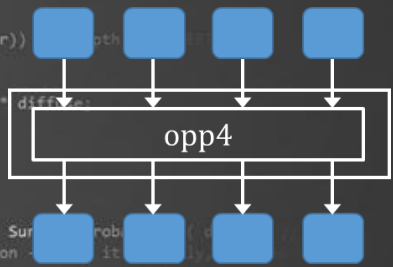


Agner Fog:
 “Automatic vectorization is the easiest way of generating SIMD code, and I would recommend to use this method when it works. Automatic vectorization may fail or produce suboptimal code in the following cases:

- when the algorithm is too complex.
- when data have to be re-arranged in order to fit into vectors and it is not obvious to the compiler how to do this or when other parts of the code needs to be changed to handle the re-arranged data.
- when it is not known to the compiler which data sets are bigger or smaller than the vector size.
- when it is not known to the compiler whether the size of a data set is a multiple of the vector size or not.
- when the algorithm involves calls to functions that are defined elsewhere or cannot be inlined and which are not readily available in vector versions.
- when the algorithm involves many branches that are not easily vectorized.
- when floating point operations have to be reordered or transformed and it is not known to the compiler whether these transformations are permissible with respect to precision, overflow, etc.
- when functions are implemented with lookup tables.

```

    ...
    & (depth + MAXDEPTH)
    ...
    inside / ...
    nt = nt / nc;
    pos2t = 1.0f / nnt;
    D, N );
    ...
    at a = nt - nc, b = nt;
    at Tr = 1 - (R0 + (1 - R0)
    Tr) R = (D * nnt - N * (nc
    ...
    E * diffuse;
    = true;
    ...
    refl + refr))
    D, N );
    refl * E * diffuse;
    = true;
    ...
    MAXDEPTH)
    survive = Sur
    estimation
    if;
    radiance = SampleLight( @rand, I, AL, All
    e.x + rad
    ...
    w = true
    at brdfP
    at3 fact
    at weigh
    at cosTh
    E * ((w
    ...
    random wa
    ve)
    ...
    at3 brdf
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;
    
```



AoS

SoA

SIMD Basics

Other instructions:

```

__m128 c4 = _mm_div_ps( a4, b4 ); // component-wise division
__m128 d4 = _mm_sqrt_ps( a4 ); // four square roots
__m128 d4 = _mm_rcp_ps( a4 ); // four reciprocals
__m128 d4 = _mm_rsqrt_ps( a4 ); // four reciprocal square roots (!)

__m128 d4 = _mm_max_ps( a4, b4 );
__m128 d4 = _mm_min_ps( a4, b4 );
    
```

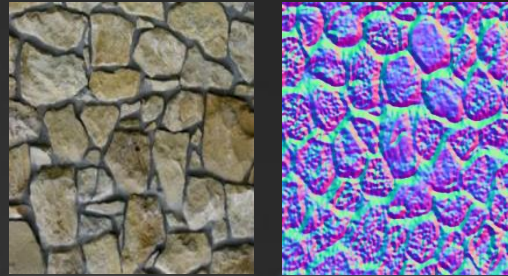
Keep the assembler-like syntax in mind:

```

__m128 d4 = dx4 * dx4 + dy4 * dy4;
    
```



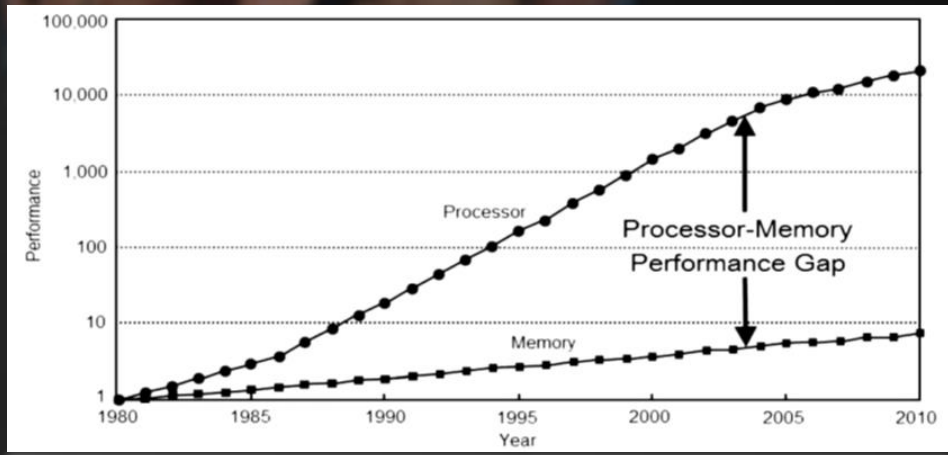
Recap – lecture 8



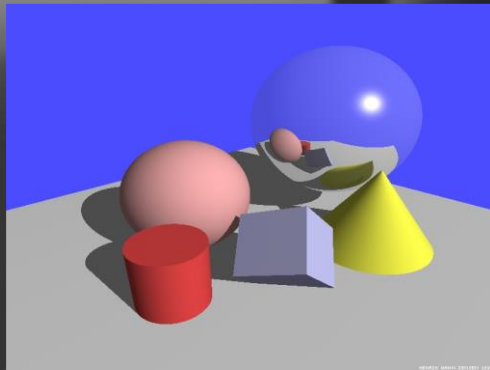
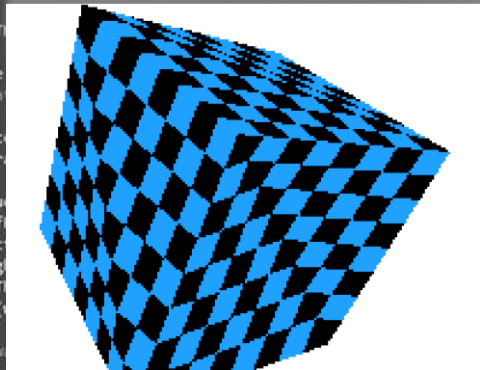
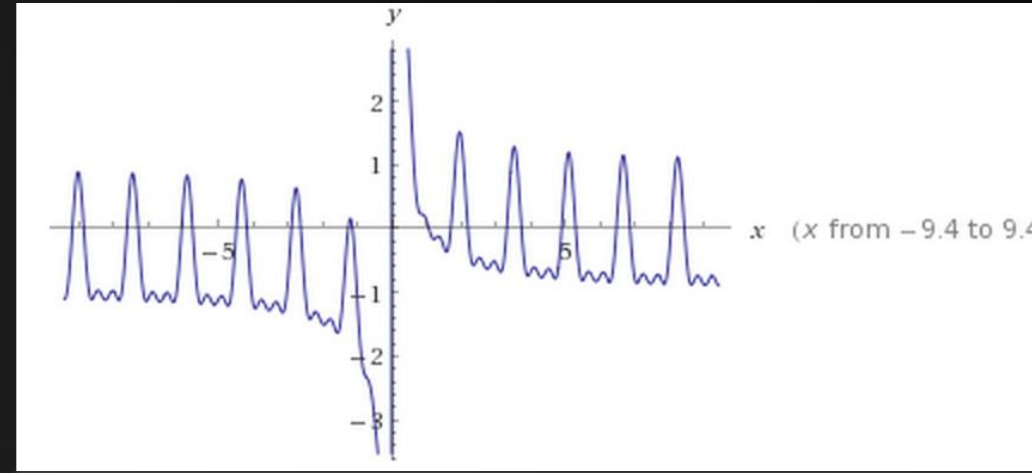
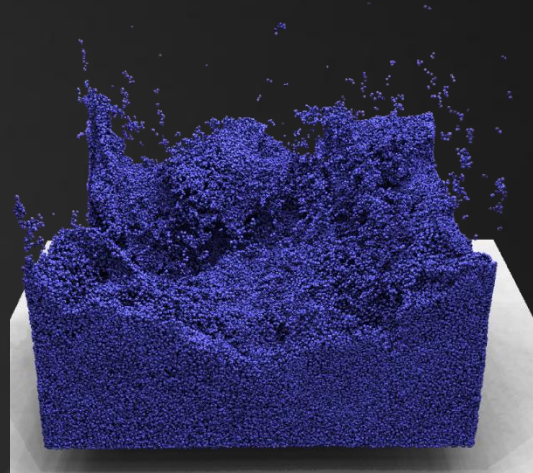
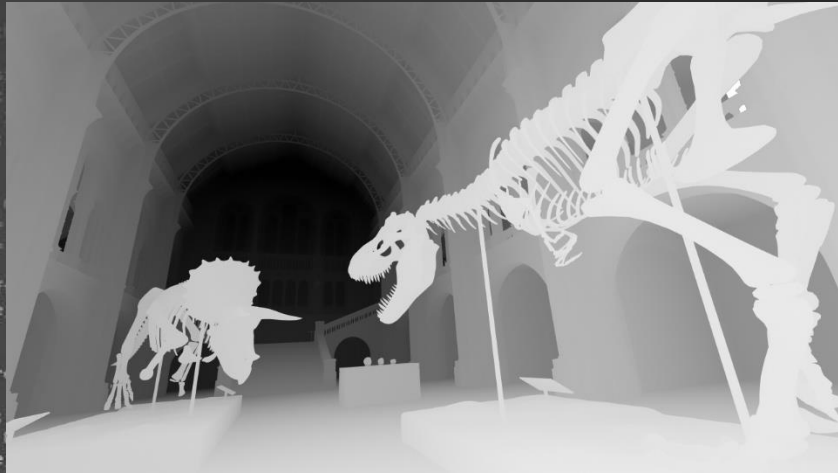
```

...
(depth < MAXDEPTH)
...
inside / ...
nt = nt / nc;
cos2t = 1.0f - nt;
D, N );
...
a = nt - nc; b = nt;
at Tr = 1 - (R0 + (1 - R0) *
Tr) R = (D * nnt - N * (a
...
diffuse;
= true;
...
refl + refr)) && (depth < MAXDEPTH)
D, N );
-refl * E * diffuse;
= true;
...
MAXDEPTH)
survive = SurvivalProbability( diffuse );
estimation - doing it properly, close
if;
radiance = SampleLight( @rand, I, R, Alignment
e.x + radiance.y + radiance.z) > 0) && (survive)
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Radiance
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
...
random walk - done properly, closely following
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, @R, @pdf
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Recap – lecture 9



Recap – lecture 10

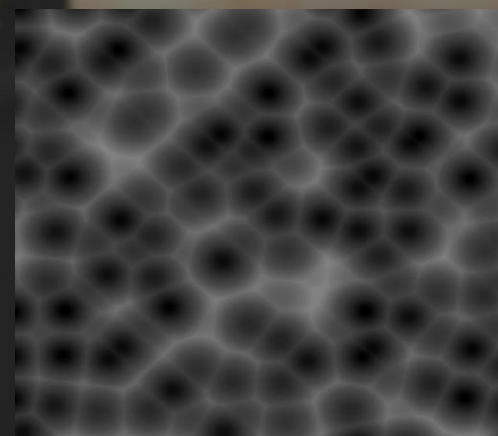
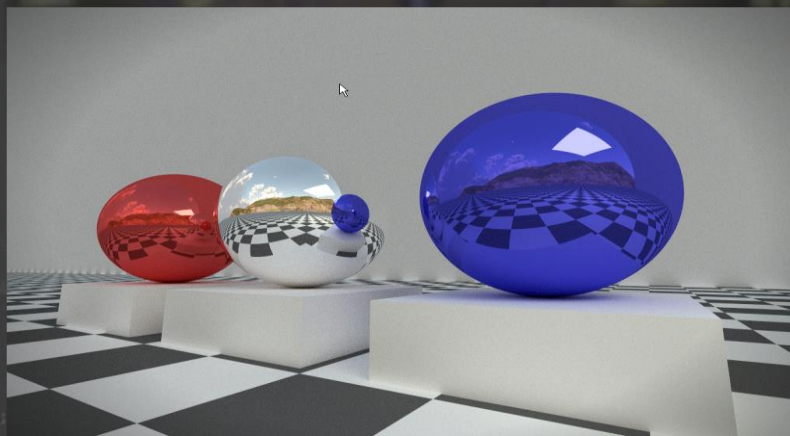
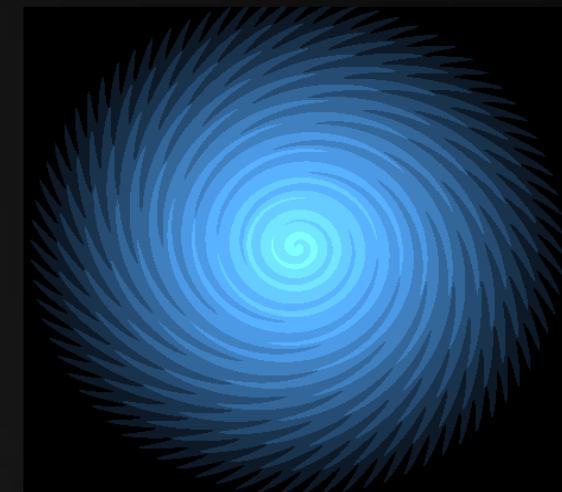


POTATO SALAD

Some good cooks sprinkle grated pimienta cheese on this

- 4 cups diced cooked potatoes
- 1 cup sliced celery
- 3 hard-cooked eggs, cut up
- ½ cup finely cut onion or sliced green onions
- ¼ cup sliced radishes
- 1 cup mayonnaise
- 1 tablespoon vinegar
- 1 teaspoon prepared mustard
- 1½ to 2 teaspoons salt
- ½ teaspoon pepper
- Lettuce

Mix all the ingredients in a bowl. Cover and refrigerate several hours so flavors can blend. Serve on crisp lettuce. Makes 6 servings.

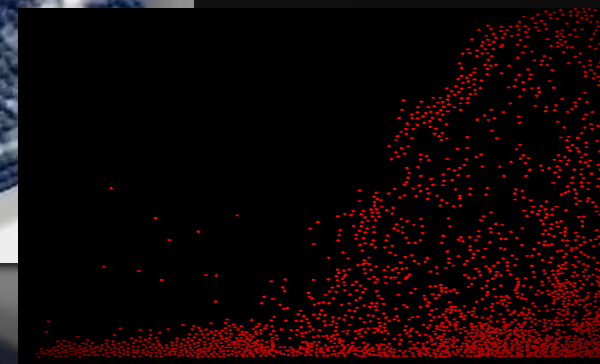
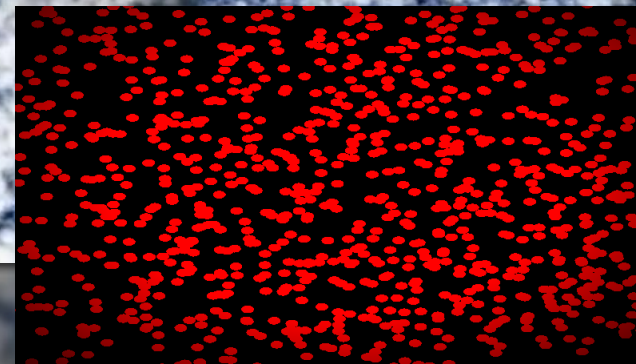
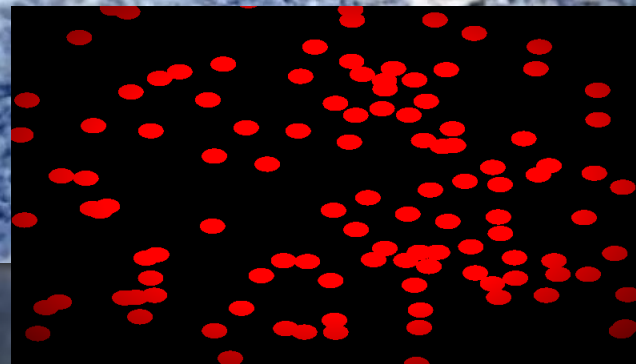
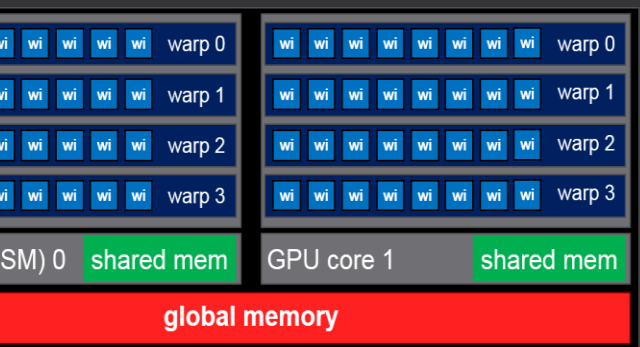
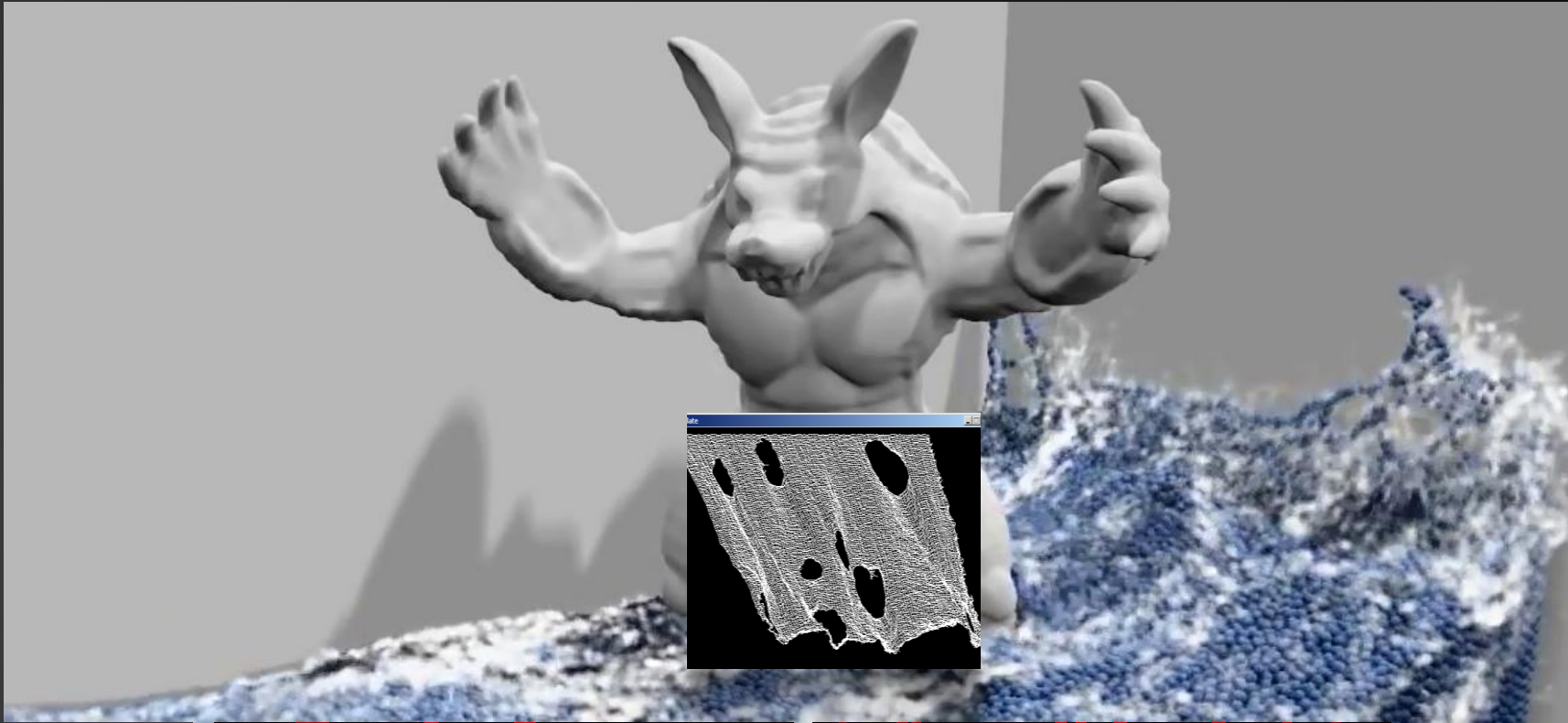


Recap – lecture 11

```

...
(depth < MAXDEPTH)
...
inside / ...
nt = nt / nc;
pos2t = 1.0f - nnt;
D, N );
...
at a = nt - nc; b = nt;
at Tr = 1 - (R0 + (1 - R0) * ...
Tr) R = (D * nnt - N * ...
...
E * diffuse;
= true;
...
refl + refr)) && (depth < MAXDEPTH)
D, N );
-refl * E * diffuse;
= true;
...
MAXDEPTH)
survive = SurvivalProbability( diffuse;
...

```



```

survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```

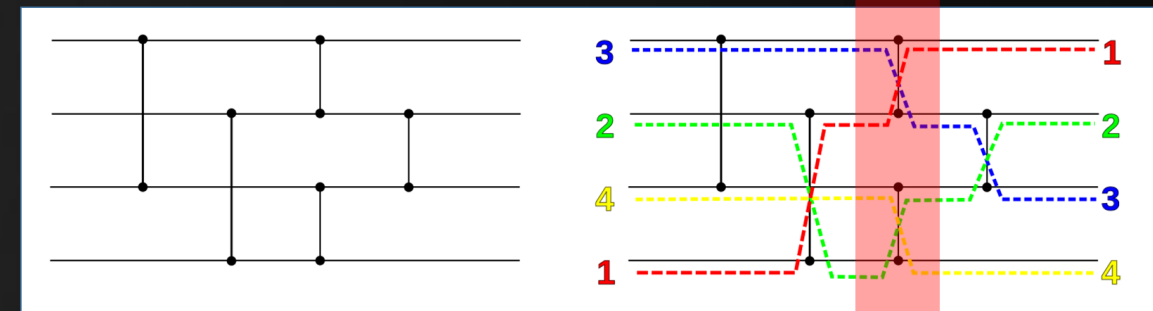
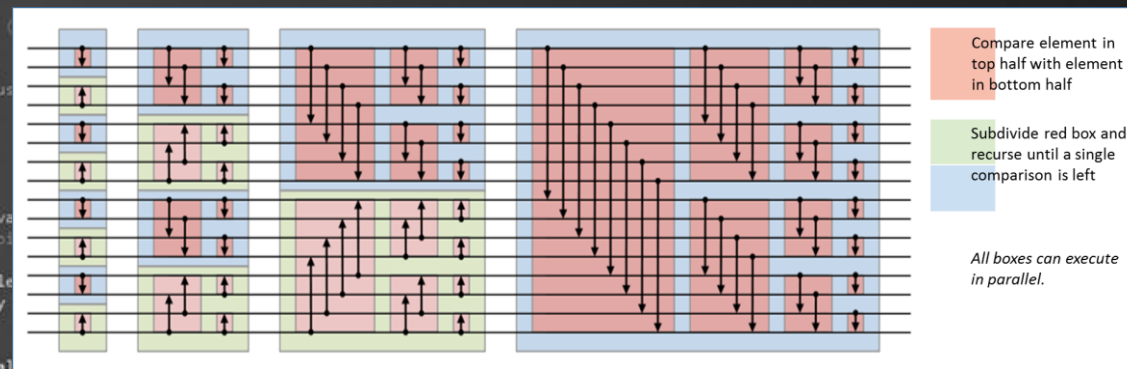
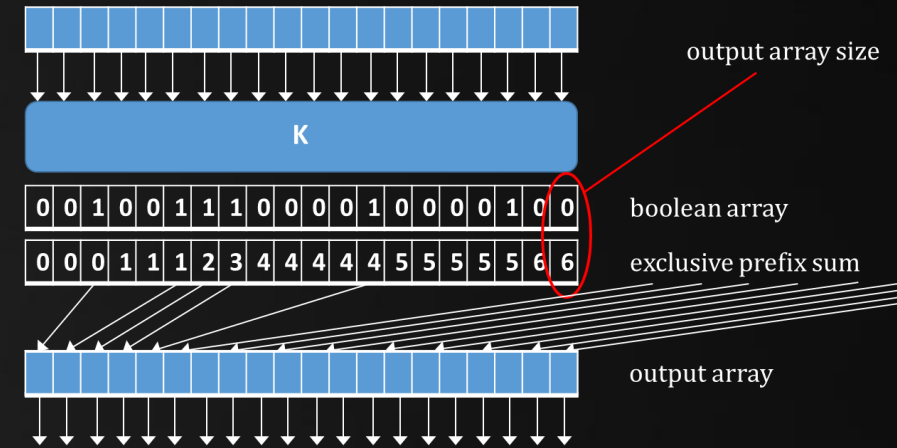


Recap – lecture 12

```

...
    & (depth + MAXDEPTH);
...
    inside / inside;
    nt = nt / nc;
    pos2t = 1.0f - nnt;
    D, N );
...
    at a = nt - nc, b = nt;
    at Tr = 1 - (R0 + (1 - R0));
    Tr) R = (D * nnt - N * (a0
...
    E * diffuse;
    = true;
...
    (refl + refr)) &&
...
    D, N );
    refl * E * diffuse;
    = true;
...
    MAXDEPTH);
    survive = Survive;
    estimation - do;
    if;
    radiance = Sample;
    e.x + radiance.y;
...
    w = true;
    at brdfPdf = Eval;
    at3 factor = diffuse * INVPI;
    at weight = Mis2( directPdf, brdfPdf );
    at cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / directPdf) * (radiance
...
    random walk - done properly, closely following
    (survive)
...
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf);
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;

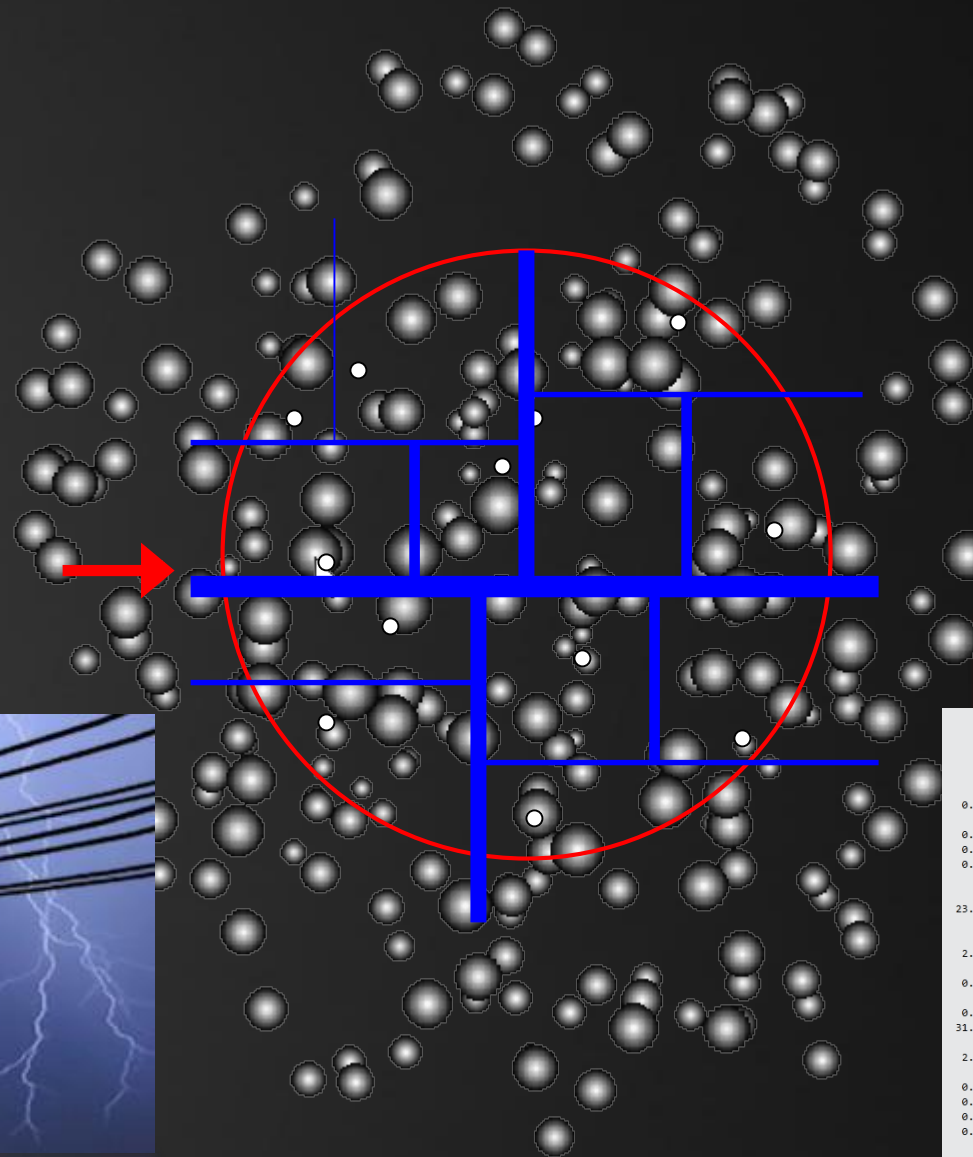
```



Recap – lecture 13

CPU-Z Ver. 1.69.2.x64

CPU			
Caches			
Mainboard			
Memory			
SPD			
Graphics			
About			
Processor			
Name	Intel Core i7 4800MQ		
Code Name	Haswell	Max TDP	47 W
Package	Socket 947 rPGA		
Technology	22 nm	Core Voltage	0.743 V
Specification			
Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz			
Family	6	Model	C
Ext. Family	6	Ext. Model	3C
Stepping	3		
Revision	C0		
Instructions	MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX, AVX2, FMA3		
Clocks (Core #0)			
Core Speed	1498.58 MHz		
Multiplier	x 15.0 (8 - 37)		
Bus Speed	99.91 MHz		
Rated FSB			
Cache			
L1 Data	4 x 32 KBytes	8-way	
L1 Inst.	4 x 32 KBytes	8-way	
Level 2	4 x 256 KBytes 8-way		
Level 3	6 MBytes 12-way		
Selection			
Processor #1	Cores 4 Threads 8		
Tools			
Validate			
OK			



```

void Game::Tick(float _dt)
{
    // fill the grid
    for( int q = 0; q < 20; q++)
    {
        memset( nr, 0, SCRWIDTH / 16 * SCRHEIGHT / 16 * 4 );
        for( int i = 0; i < BALLCOUNT; i++)
        {
            int gx = CLAMP( (int)(pos[i].x / 16 ), 0, SCRWIDTH / 16 - 1 );
            int gy = CLAMP( (int)(pos[i].y / 16 ), 0, SCRHEIGHT / 16 - 1 );
            grid[gy][gx][nr[gy][gx]++] = i;
        }
    }

    screen->Clear( 0 );
    timer t;
    t.reset();
    for( int v = 0; v < SCRHEIGHT / 16; v++) for( int u = 0; u < SCRWIDTH / 16; u++) for( int j = 0; j < nr[v][u]; j++)
    {
        int i = grid[v][u][j];
        // draw ball
        if( i == 0 ) green->Draw( (int)pos[i].x, (int)pos[i].y, screen );
        else ball->Draw( (int)pos[i].x, (int)pos[i].y, screen );
        // update ball position
        pos[i] += vel[i] * speed[i] * 0.05f;
        // screen boundary collisions
        if( pos[i].x < 10 ) vel[i].x = -vel[i].x, pos[i].x = 10;
        if( pos[i].y < 10 ) vel[i].y = -vel[i].y, pos[i].y = 10;
        if( pos[i].x > (SCRWIDTH - 20) ) vel[i].x = -vel[i].x, pos[i].x = SCRWIDTH - 20;
        if( pos[i].y > (SCRHEIGHT - 20) ) vel[i].y = -vel[i].y, pos[i].y = SCRHEIGHT - 20;
        // ball collisions
        int gx1 = MAX( 0, u - 1 ), gx2 = MIN( SCRWIDTH / 16 - 1, u + 1 );
        int gy1 = MAX( 0, v - 1 ), gy2 = MIN( SCRHEIGHT / 16 - 1, v + 1 );
        for( int y = gy1; y <= gy2; y++) for( int x = gx1; x <= gx2; x++) for( int k = 0; k < nr[y][x]; k++)
    }
}

```

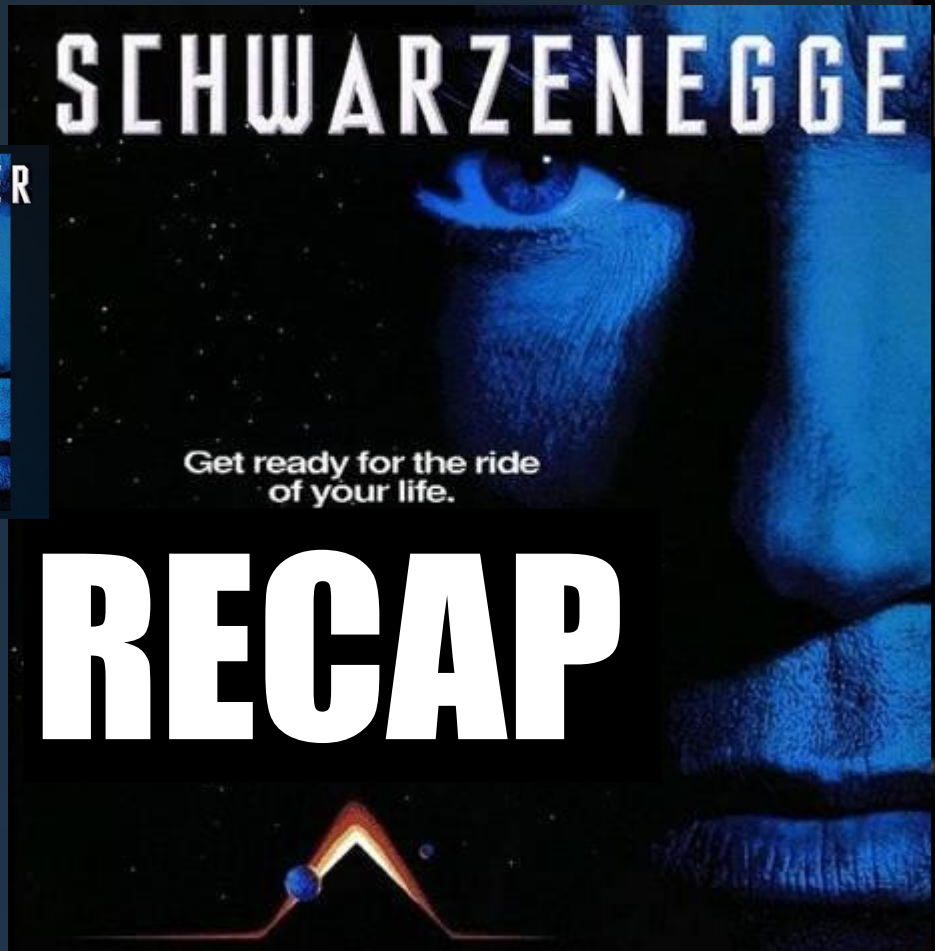


Recap – Lecture 14

```

...
    & (depth < MAXDEPTH)
...
    inside / inside
    nt = nt / nc;
    cos2t = 1.0f - nt;
    D, N );
    )
...
    at a = nt - nc, b = nt;
    at Tr = 1 - (R0 + (1 - R0)
    Tr) R = (D * nnt - N * (a
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH)
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
...
survive = SurvivalProbability( diffuse,
estimation - doing it properly, close
if;
radiance = SampleLight( @rand, I, R, Aligned
e.x + radiance.y + radiance.z) > 0) && (rand
...
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Survive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radia
...
random walk - done properly, closely following
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R,
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



TOTAL RECAP

TOTAL RECAP

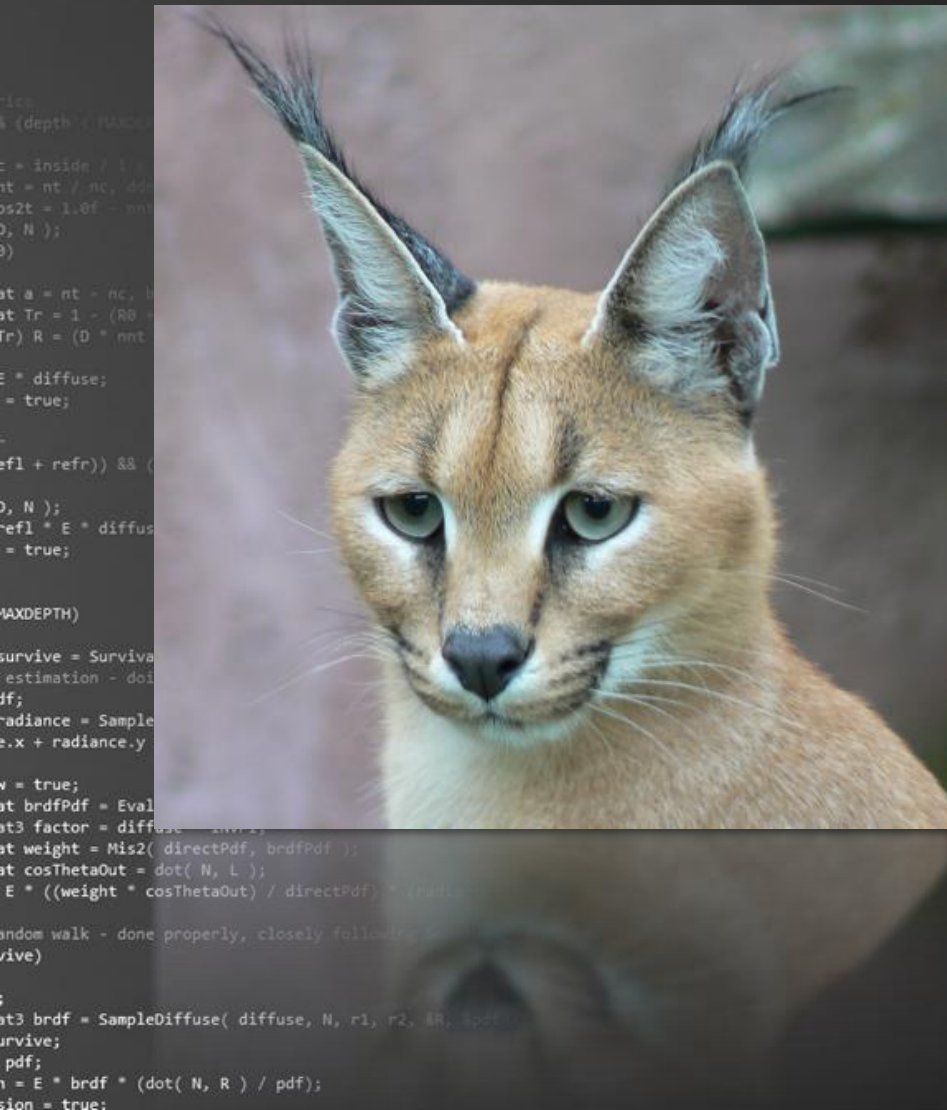
```
...ics
& (depth < MAXDEPTH)
...
c = inside / (1 + refl);
nt = nt / nc; ddn = ddn / nc;
cos2t = 1.0f - nnt * nnt;
D, N );
)
...
at a = nt - nc, b = nt * nc;
at Tr = 1 - (R0 + (1 - R0) * c);
Tr) R = (D * nnt - N * (1 - nnt));
...
E * diffuse;
= true;
...
efl + refr)) && (depth < MAXDEPTH)
D, N );
-efl * E * diffuse;
= true;
...
MAXDEPTH)
survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following
if;
radiance = SampleLight( &rand, I, &t, &align,
e.x + radiance.y + radiance.z) > 0) && (depth <
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Survive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
...
random walk - done properly, closely following
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf);
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

Today's Agenda:

- Exam
- Digest
- Grand Recap
- Now What



Now What



```
ics
& (depth < MAXDEPTH)
{
    t = inside / (inside + outside);
    nt = nt / nc; nct = nct / nc;
    cos2t = 1.0f - nnt * nnt;
    D, N );
}

at a = nt - nc, b = nt + nc;
at Tr = 1 - (R0 + (1 - R0) * a);
Tr) R = (D * nnt - N * nct) * a;

E * diffuse;
= true;

efl + refr)) && (depth < MAXDEPTH)
D, N );
refl * E * diffuse;
= true;

MAXDEPTH)

survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following
if;
radiance = SampleLight( @rand, I, @t, @align, @
e.x + radiance.y + radiance.z) > 0) && (survive);

v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Survive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance

random walk - done properly, closely following
ive)

;
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, @R, @pdf
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```



/INFOMOV/

