

```
ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 0)
    {
        nt = nt / nc; ddn = ddn * ddn;
        cos2t = 1.0f - nnt * ddn;
        D, N );
    }
    at a = nt - nc, b = nt + nc;
    at Tr = 1 - (R0 + (1 - R0) * ddn);
    (Fr) R = (D * nnt - N * (ddn *
    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, &L, &light);
    e.x + radiance.y + radiance.z) > 0) && (abs(radiance
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    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;
```

/INFOMOV/

Optimization & Vectorization

J. Bikker - Sep-Nov 2018 - Lecture 15: "Grand Recap"

Welcome!

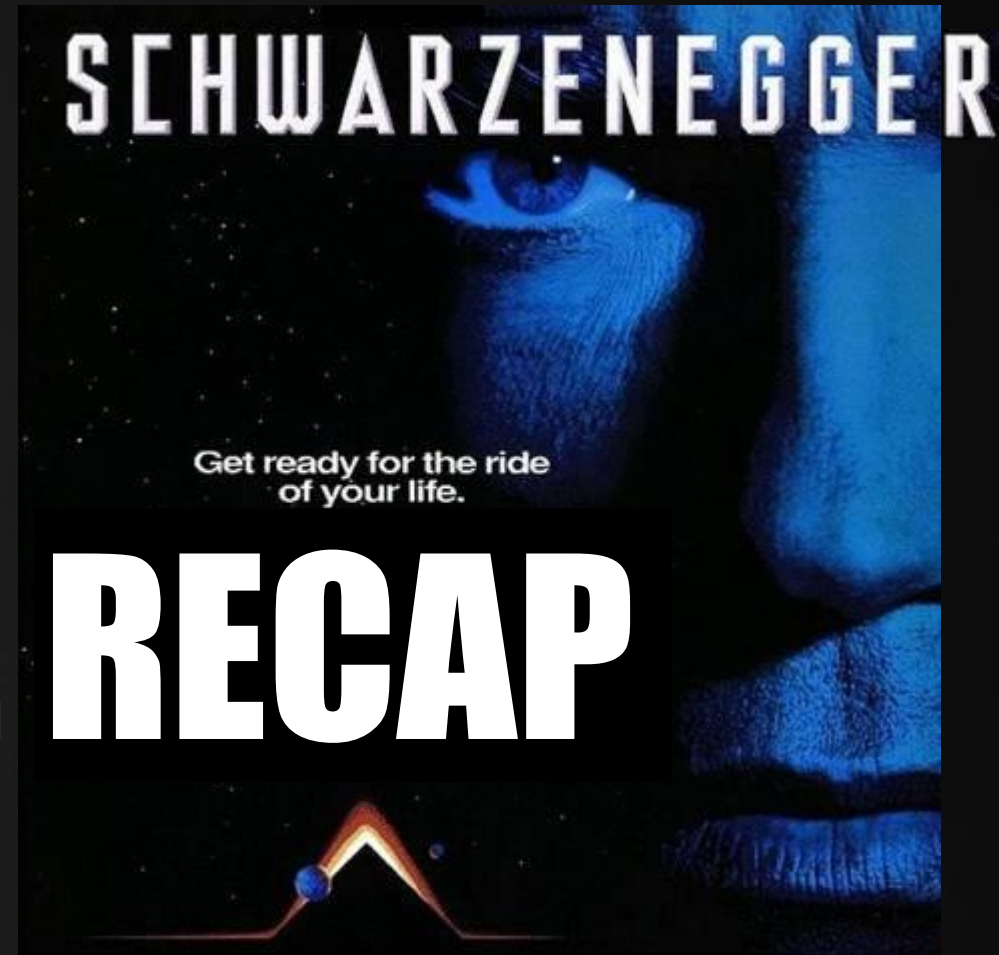



```
ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 0)
    {
        nt = nt / nc; ddn = ddn * ddn;
        float r2 = 1.0f - nnt * nnt;
        float D, N );
        float a = nt - nc, b = nt + nc;
        float Tr = 1 - (R0 + (1 - R0) * r2);
        float R = (D * nnt - N * (ddn * r2));
        E * 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, &L, &light);
        e.x + radiance.y + radiance.z) > 0) && (abs(radiance.x)
        w = true;
        at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
        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;
    }
}
```

Today's Agenda:

- Grand Recap
- Exam
- Now What

TOTAL RECAP



Recap

```

...
    & (depth < MAXDEPTH)
...
    if (inside ? 1 : 0)
        nt = nt / nc, ddn = ddn / n;
        cos2t = 1.0f - nnt * nnt;
        D, N );
    )
...
    at a = nt - nc, b = nt * nc;
    at Tr = 1 - (R0 + (1 - R0) * r);
    Tr) R = (D * nnt - N * (Dd
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEP
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
...
survive = SurvivalProbability( diffuse,
estimation - doing it properly, closely
if;
radiance = SampleLight( &rand, I, &L, &light
e.x + radiance.y + radiance.z) > 0) && (survive
...
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
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 S&S
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Recap

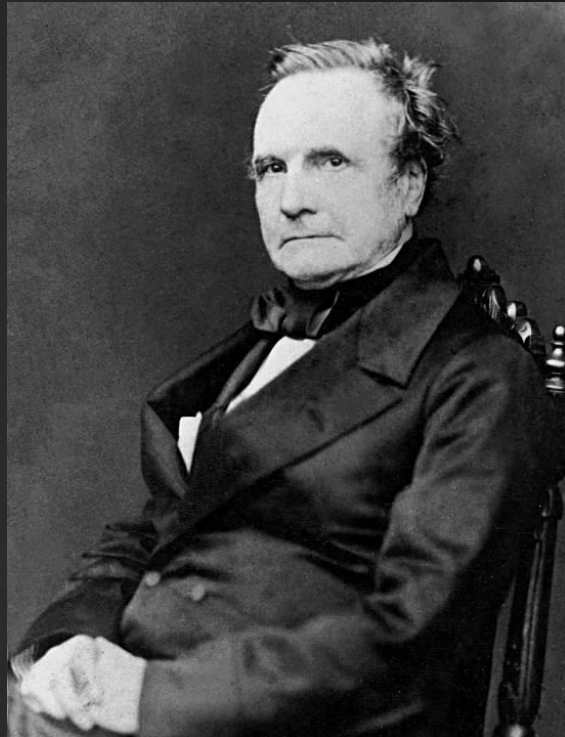
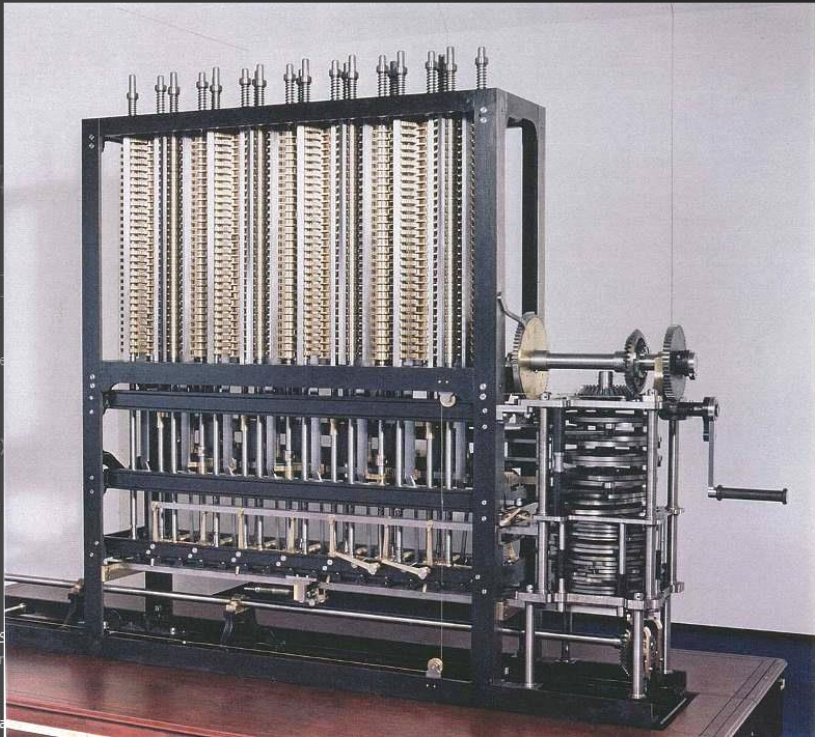
```

...
    & (depth < MAXDEPTH)
...
    if (inside ? 1 : 0) {
        nt = nt / nc, ddn = ddn / n;
        cos2t = 1.0f - nnt * nnt;
        D, N );
    }
...
    at a = nt - nc, b = nt + nc;
    at Tr = 1 - (R0 + (1 - R0) * r);
    Tr) R = (D * nnt - N * (ddn
...
    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, &L, &light
e.x + radiance.y + radiance.z) > 0) && (cosThetaOut
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
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 S&S10.10
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Recap – lecture 1



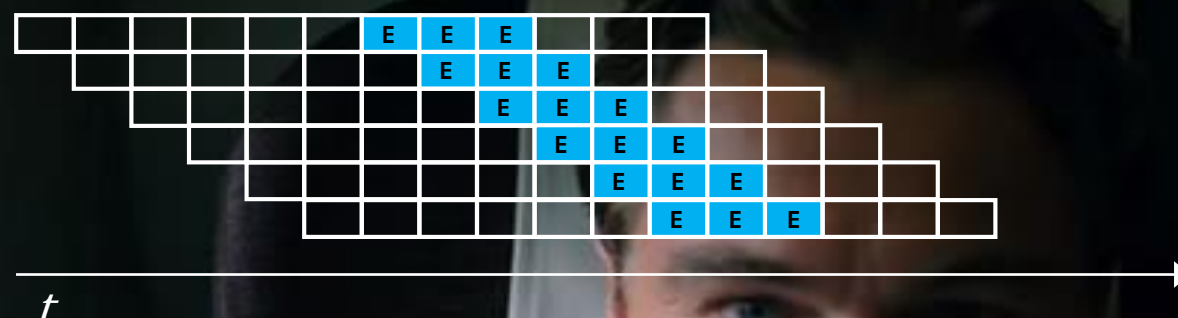
- Profiling
- High Level
- Basic Low Level
- Cache & Memory**
- Data-centric
- Compilers
- Fixed-point Arithmetic
- CPU architecture
- SIMD**
- GPGPU

```

ics
(depth
= inside
nt = nt /
s2t = 1.0
D, N );
)
at a = nt
at Tr = 1
Tr) R = (D
E * diffuse
= true;
efl + refr
D, N );
refl * E *
= true;
MAXDEPTH)
survive =
estimation
df;
-radiance =
e.x + radi
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurf
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (rad
random
ve) properly, closely following
ive)
at3 brdf
mpleDiffuse( diffuse, N, r1, r2, &R, sp
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
    
```



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 (!!)
```

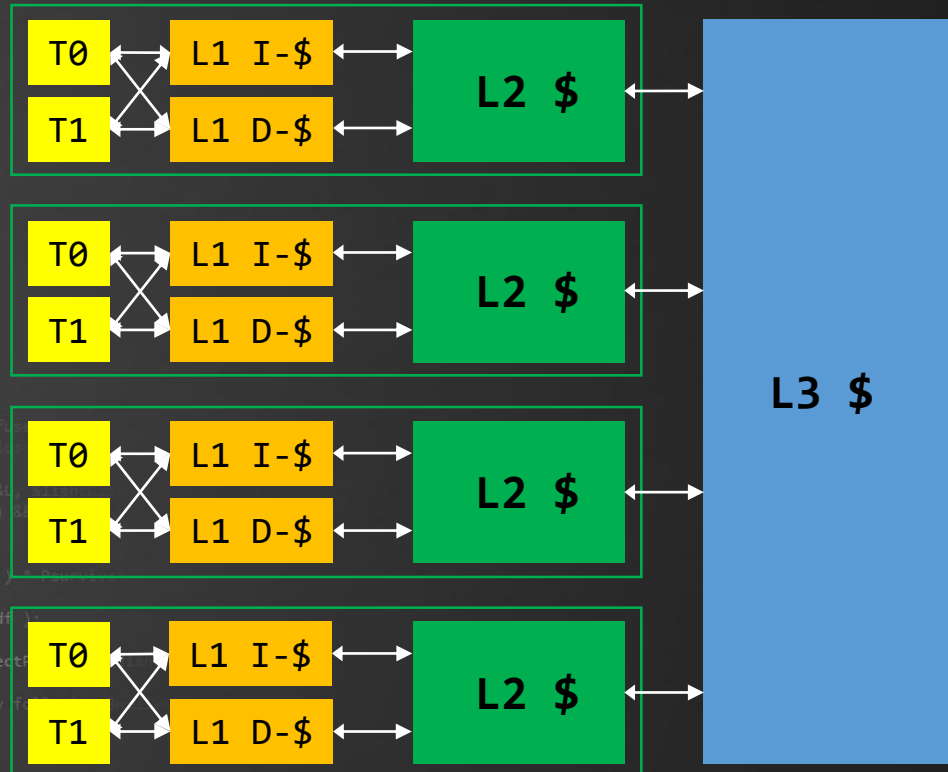
```

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



0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000D
000F

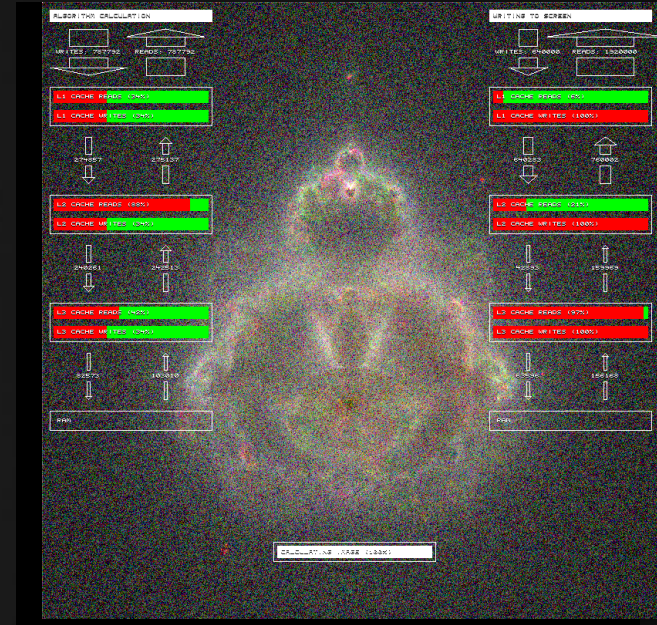
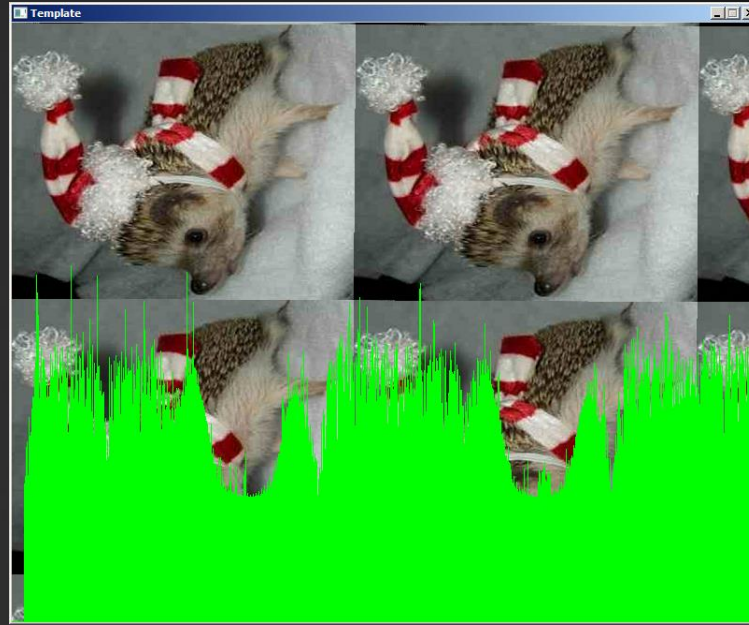


Recap – lecture 4

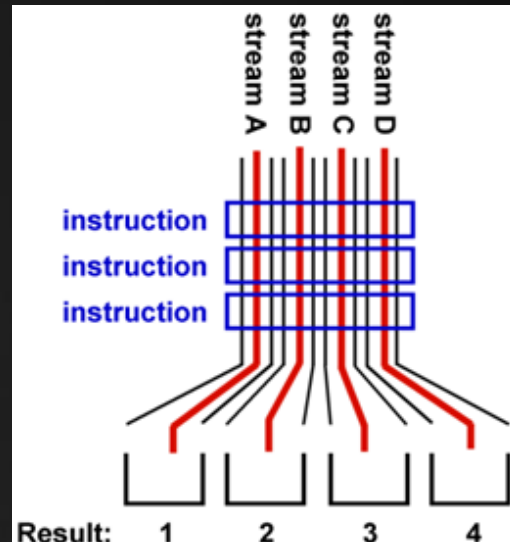
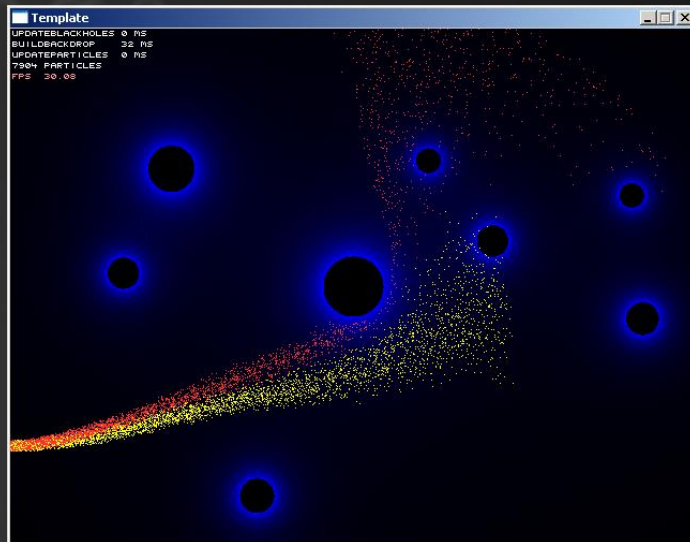
```

ics
& (depth < MAXD
c = inside ? i
nt = nt / nc; do
os2t = 1.0f - nt
D, N );
);
)
a = nt - nc,
at Tr = 1 - (R0
Tr) R = (D * nnt
E * diffuse;
= true;
efl + refr)) &&
D, N );
refl * E * diffu
= true;
MAXDEPTH)
survive = Surviv
estimation - do
df;
radiance = Samplelight( &rand, I, &L, &light
e.x + radiance.y + radiance.z) > 0) &&
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurface
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
random properly, closely following 3-sphere
ive)
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Recap – lecture 5 & 6

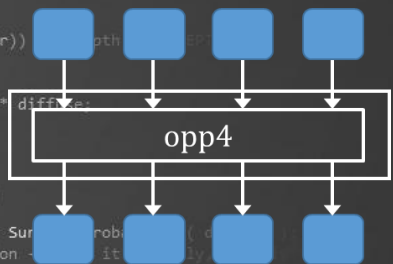


~~Agner Fog: "Automatic vectorization is the easiest way of generating SIMD code, and I would recommend to use this method when it works. Automatic vectorization may fail or produce suboptimal code in the following cases:~~

- when the algorithm is too complex.
- when data has 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 ? 1 : 0;
    nt = nt / nc;
    ps2t = 1.0f * nnt;
    D, N );
    )
...
    at a = nt - nc, b = nt - nc;
    at Tr = 1 - (R0 + (1 - R0));
    Tr) R = (D * nnt - N * (dd);
...
    E * diffuse;
    = true;
...
    refl + refr))
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
...
    survive = Survive;
    estimation - it;
    if;
    radiance = SampleLight( &rand, I, &L, &light;
    e.x + ra;
...
    w = true;
    at brdfP;
    at3 fact;
    at weigh;
    at cosTh;
    E * ((w;
...
    random;
    (ive);
...
    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;
    
```

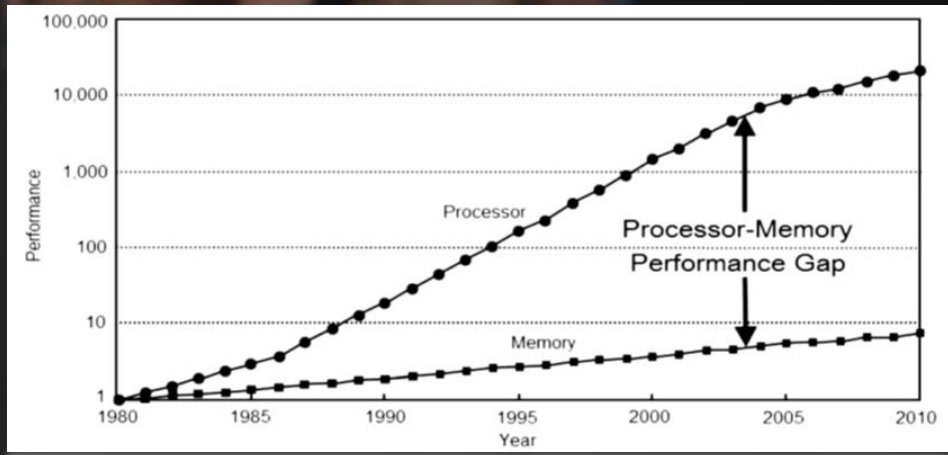
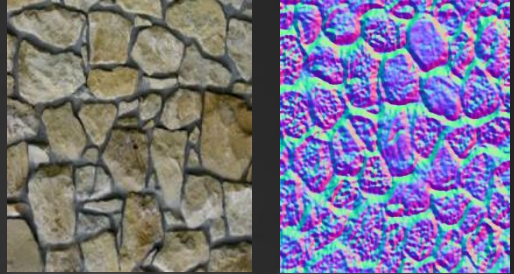


Recap – lecture 7

```

...ics
& (depth < MAXDEPTH)
...
c = inside ? 1 : 0;
nt = nt / nc; ddn = ddn;
...
s2t = 1.0f - nnt;
...
D, N );
...
)
...
at a = nt - nc, b = nt + nc;
at Tr = 1 - (R0 + (1 - R0)
Tr) R = (D * nnt - N * (dd
...
E * diffuse;
= true;
...
refl + refr)) && (depth < MAXDEPTH)
D, N );
refl * E * diffuse;
= true;
...
MAXDEPTH)
survive = SurvivalProbability( di
estimation - doing it properly,
df;
radiance = SampleLight( &rand, I,
e.x + radiance.y + radiance.z ) >
...
w = true;
at brdfPdf = EvaluateDiffuse( L,
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdf
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / di
...
random properly, close
ive)
...
at3 brdf SampleDiffuse( diffuse
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Recap – lecture 8

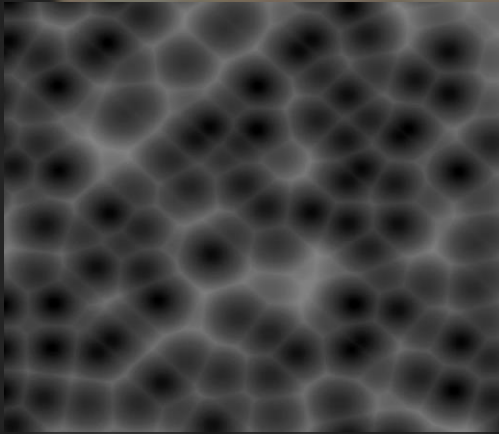
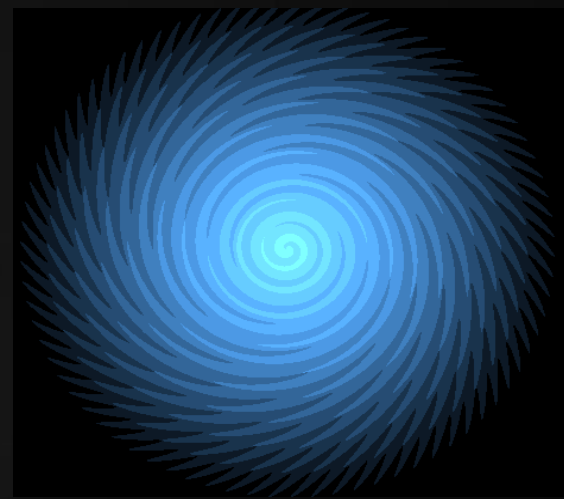


POTATO SALAD

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
- ¼ 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.

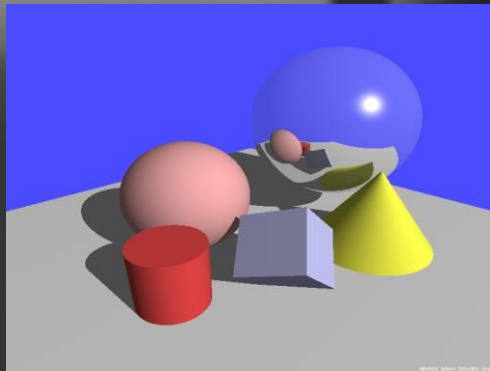
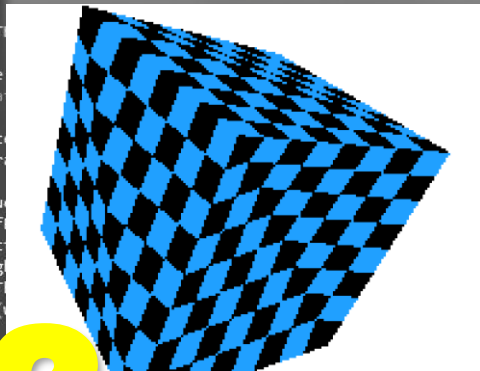
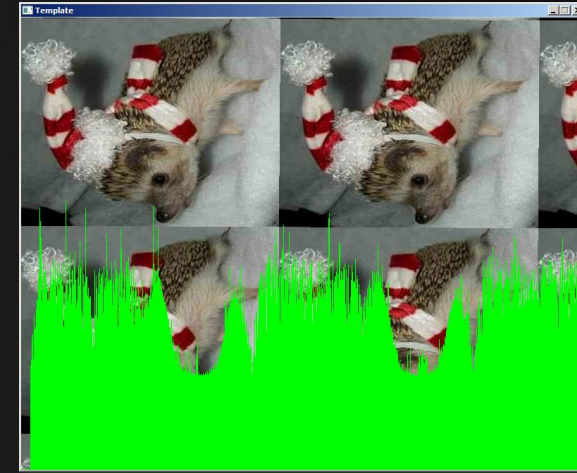
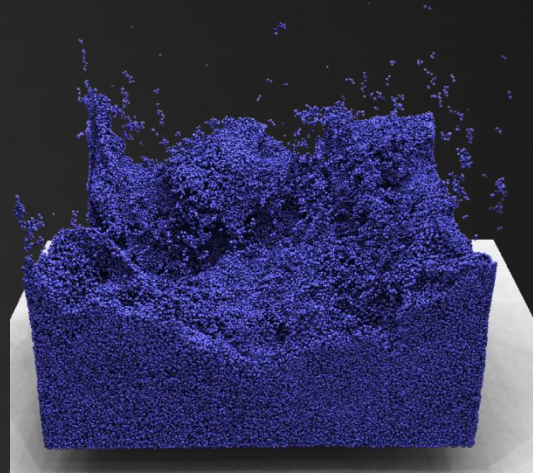


```

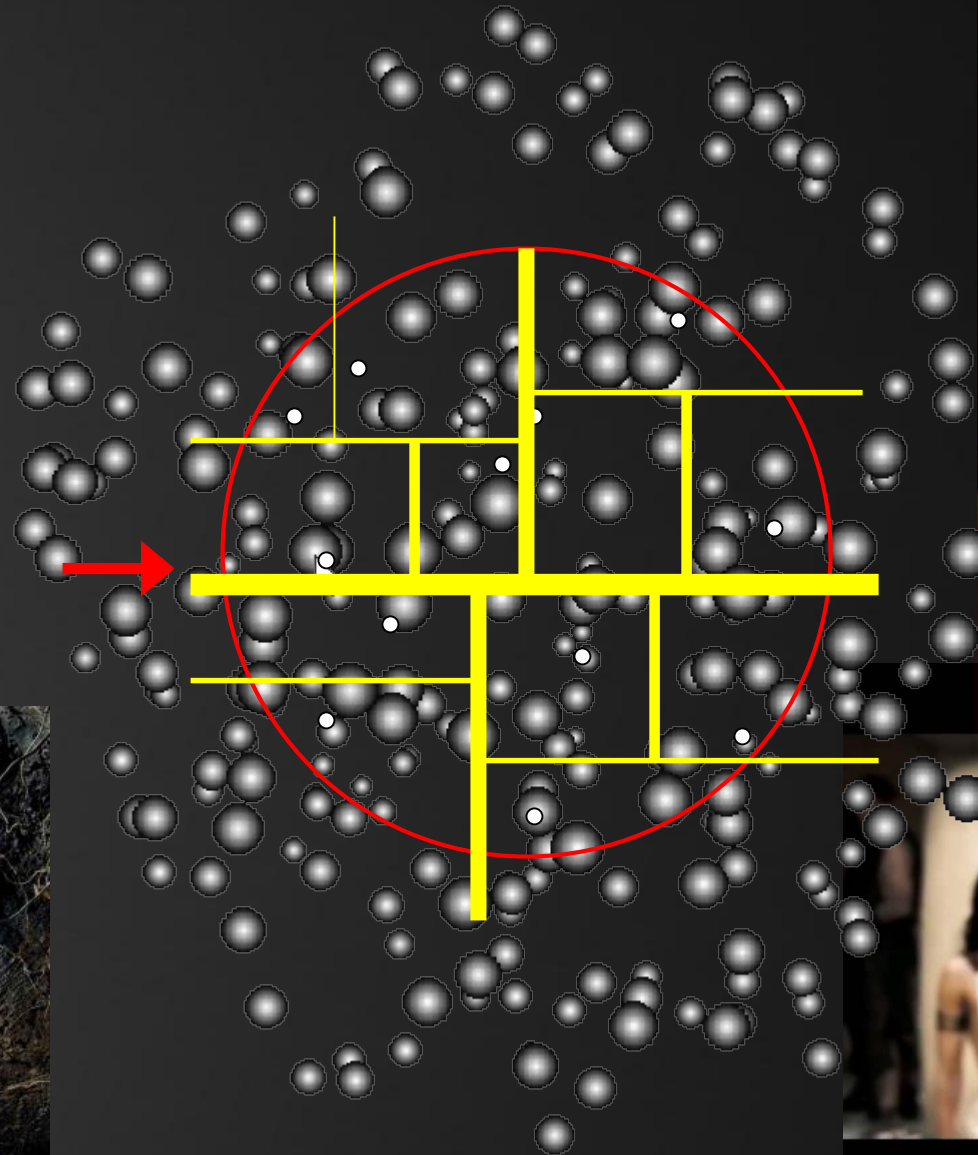
...
    & (depth < MAXDEPTH)
...
    nt = nt / nc; ddn = ddn / nc;
    ns2t = 1.0f - nnt;
    D, N );
...
    at a = nt - nc, b = nt + nc;
    at Tr = 1 - (R0 + (1 - R0) * nnt);
    Tr) R = (D * nnt - N * (D0
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH)
...
    D, N );
    refl * E * diffuse;
    = true;
...
    MAXDEPTH)
...
    survive = SurvivalProbability( diffuse,
    estimation - doing it properly, closely
    f;
    radiance = SampleLight( &rand, I, &L, M);
    e.x + radiance.y + radiance.z) > 0) && (
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Pdf;
    at3 factor = diffuse * INVPI;
    at weight = Mis2( directPdf, brdfPdf );
    at cosThetaOut = dot( N, L );
    E * ((weight * cosThetaOut) / directPdf) * radiance;
...
    properly, closely following a
    (ive)
...
    at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf);
    survive;
    pdf;
    n = E * brdf * (dot( N, R ) / pdf);
    sion = true;
    
```



Recap – lecture 11



Recap – lecture 13

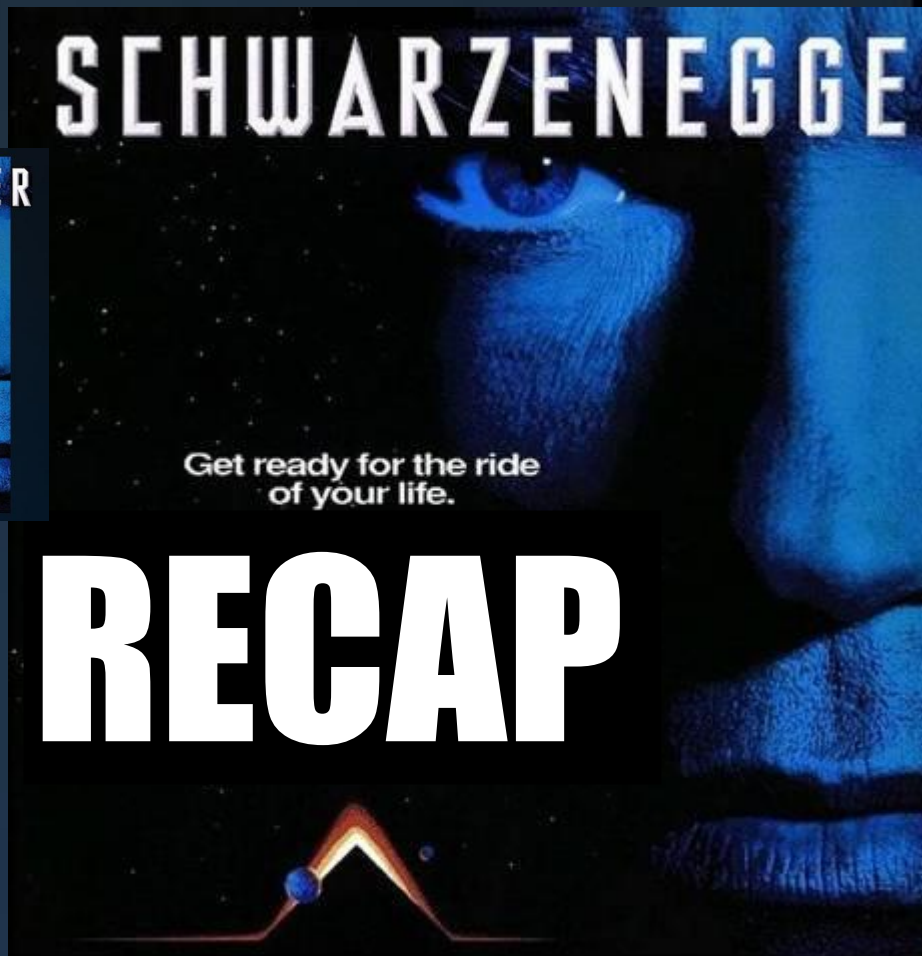


Recap – Lecture 15

```

ics
& (depth < MAXDEPTH)
= inside ? 1 : 0;
nt = nt / nc; ddn = ddn * nt;
s2t = 1.0f - nnt * nnt;
D, N );
)
at a = nt - nc, b = nt * nc;
at Tr = 1 - (R0 + (1 - R0) * s2t);
R = (D * nnt - N * (ddn * s2t + a * b));
E * diffuse;
= true;
efl + refr)) && (depth < MAXDEPTH)
D, N );
refl * E * diffuse;
= true;
MAXDEPTH)
survive = SurvivalProbability( diffuse, r1, r2, &R, &N );
estimation - doing it properly, closely following the
df;
radiance = SampleLight( &rand, I, &L, &light, &N );
e.x + radiance.y + radiance.z) > 0) && (depth < MAXDEPTH)
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance.x + radiance.y + radiance.z);
random walk - done properly, closely following the
ive)
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &N );
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



TOTAL RECAP

TOTAL RECAP

```

ics
& (depth < MAXDEPTH)
    if (inside ? 1 : 0) {
        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) * a);
at R = (D * nnt - N * (ddn * a + b));

E * diffuse;
= true;

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

MAXDEPTH)

survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following Beer's law
if;
radiance = SampleLight( &rand, I, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (depth < MAXDEPTH)
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
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 Beer's law
ive)

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

```

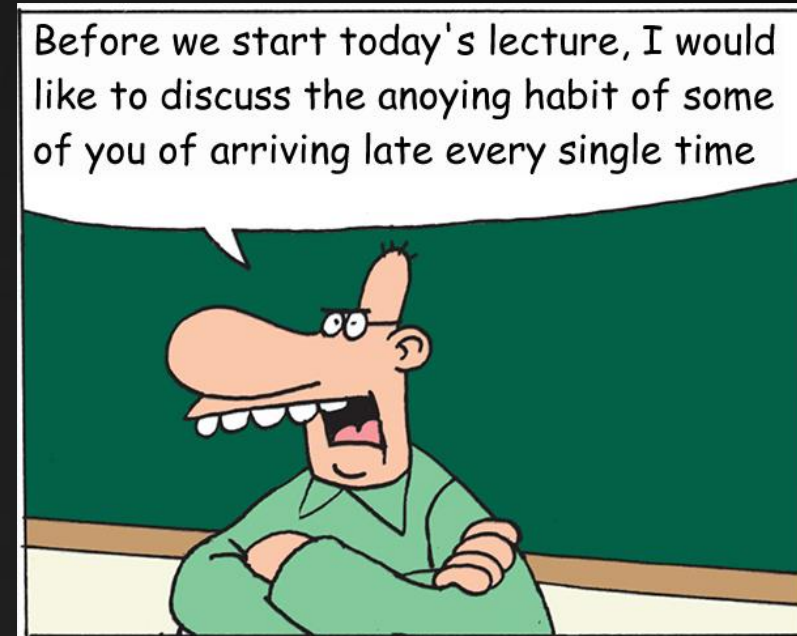
Educational Experiments [®]™



Recap

```

ics
& (depth < MAXDEPTH)
c = inside ? 1.0 : 0.0;
nt = nt / nc; ddn = sqrt(1 - ddn2);
s2t = 1.0f - nnt * nnt;
D, N );
0)
at a = nt - nc, b = nt * nc;
at Tr = 1 - (R0 + (1 - R0) * s2t);
R = (D * nnt - N * (D0 + (1 - D0) * s2t));
E * diffuse;
= true;
efl + refr)) && (depth < MAXDEPTH);
D, N );
refl * E * diffuse;
= true;
MAXDEPTH);
survive = SurvivalProbability(
estimation - doing it properly);
if;
radiance = SampleLight( &R, &D, &N );
e.x + radiance.y + radiance.z;
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance.x + radiance.y + radiance.z);
random walk - done properly, closely following the path of a photon (recursive)
ive);
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
    
```

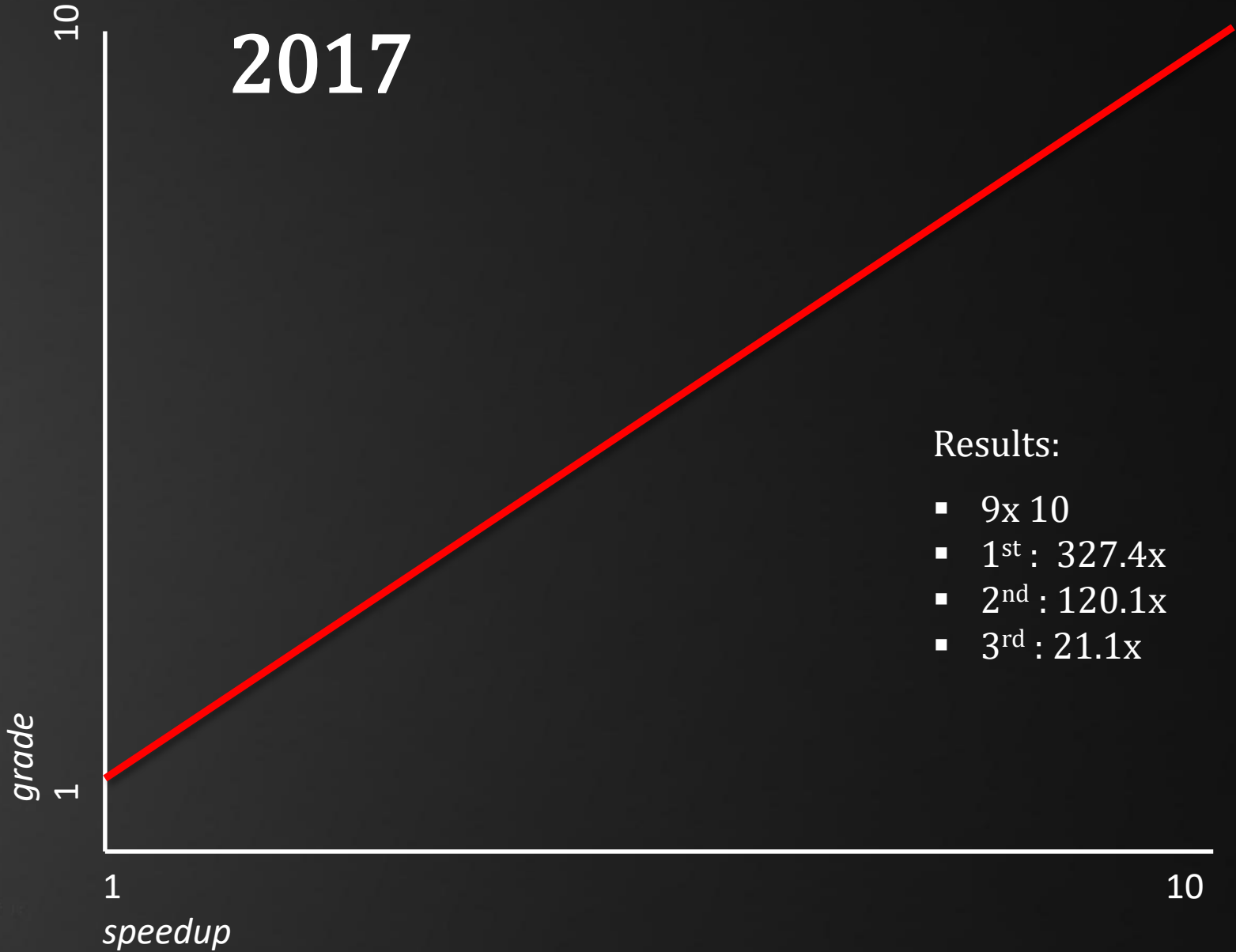


Recap

```

ics
(depth < MAXDEPTH)
inside ? 1 : 0;
nt = nt / nc; ddn = sqrt(1 - nt * nt);
s2t = 1.0f - nnt * nnt;
D, N );
);
at a = nt - nc, b = nt * nc;
at Tr = 1 - (R0 + (1 - R0) * ddn);
Tr) R = (D * nnt - N * ddn);
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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (depth <
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
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;

```



Results:

- 9x 10
- 1st : 327.4x
- 2nd : 120.1x
- 3rd : 21.1x

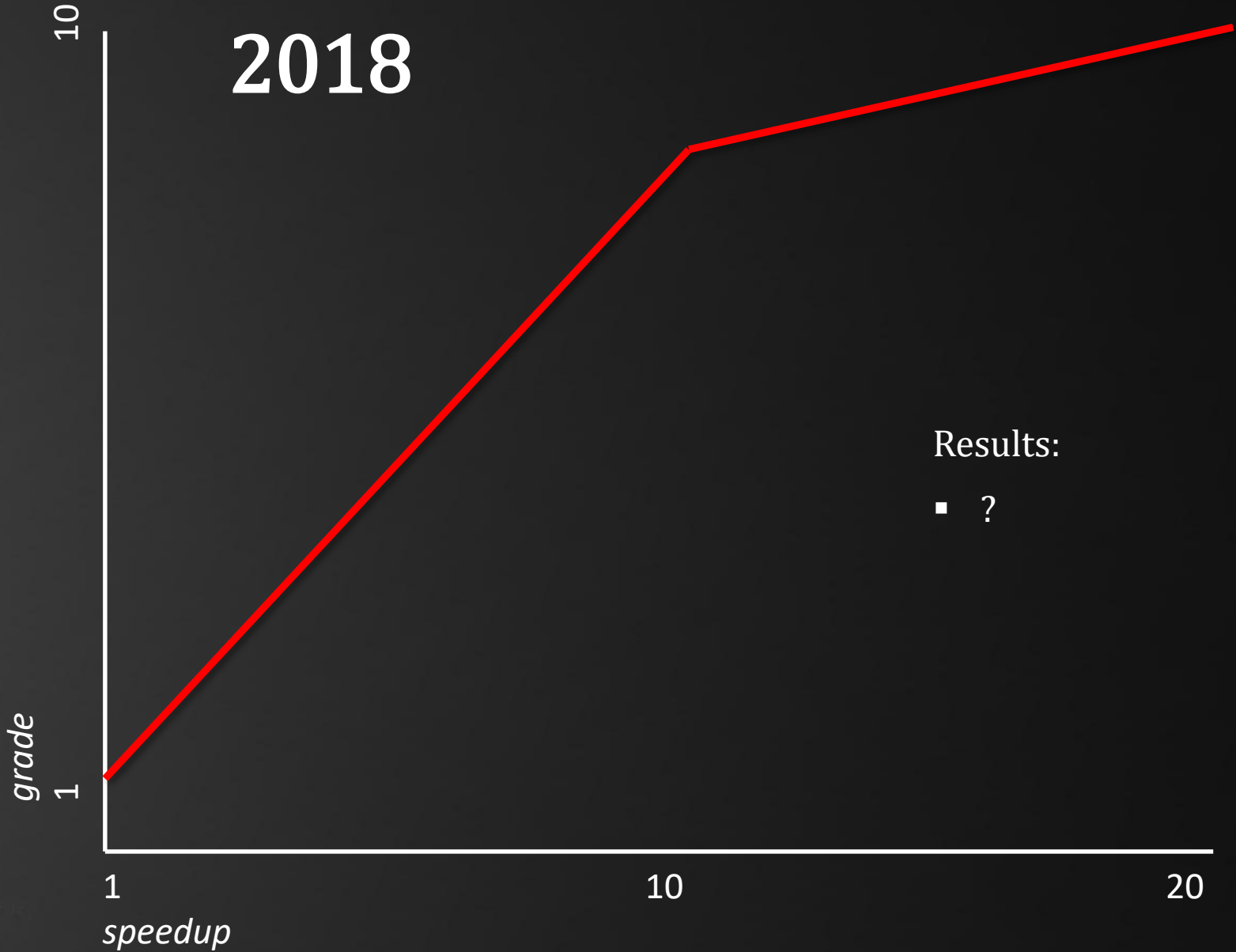


Recap

```

ics
(depth < MAXDEPTH)
inside ? 1 : 0;
nt = nt / nc; ddn = abs(ddn);
s2t = 1.0f - nnt * nnt;
D, N );
)
at a = nt - nc; b = nt + nc;
at Tr = 1 - (R0 + (1 - R0) * ddn);
Tr) R = (D * nnt - N * (ddn > 0 ? 1 : -1));
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, &L, &light;
e.x + radiance.y + radiance.z) > 0) && (depth <
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
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



“Dear Charles,

At the end of this course:

You will know how to speed up critical code by a factor 2x to 10x (and more).

- You will be able to do this to virtually any program*.
- Your understanding of higher level optimization approaches will increase.
- You will be able to apply these principles to new / alien hardware.
- You will have a more intimate relationship with your computer.

In other words:

We will talk a lot about the ‘C’ in $O(N)$.

* disclaimer: ‘that has not been optimized by an expert’.



Exam

What to Study

1. Slides

2. Literature on the website and in the slides:

- Designing for Performance, Scalability & Reliability: StarCraft II's Approach
- Modern Microprocessors: a 90 minute guide, see lecture 2 slides
- 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. 2016/2017 exams

4. Skills you picked up with the practical assignments



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

CPU's and GPU's have fundamentally different core strategies for dealing with latencies such as memory access time. What are these strategies?

```

...
    & (depth < MAXDEPTH)
...
    = inside ? 1.0f : 0.0f;
    nt = nt / nc; ddn = ddn * 2.0f;
    cos2t = 1.0f - nnt * ddn;
    D, N );
    )
...
    at a = nt - nc, b = nt + nc;
    at Tr = 1 - (R0 + (1 - R0) * ddn);
    Tr) R = (D * nnt - N * (ddn *
...
    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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (cosThetaOut > 0)
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    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;

```



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

If you were plowing a field, which would you rather use? Two strong oxen,
 or 1024 chickens?

```

...
    & (depth < MAXDEPTH)
...
    if (inside ? 1 : 0)
        nt = nt / nc; ddn = ddn * 2;
        cos2t = 1.0f - nnt * nnt;
        D, N );
    )
...
    at a = nt - nc, b = nt + nc;
    at Tr = 1 - (R0 + (1 - R0) * ddn);
    Tr) R = (D * nnt - N * (ddn
...
    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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (cosThetaOut > 0)
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    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;
    
```



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

```

...
    & (depth < MAXDEPTH)
...
    if (inside ? 1 : 0)
        nt = nt / nc; ddn = sqrt(1 - nt * nt);
        cos2t = 1.0f - nnt * nnt;
        D, N );
    )
...
    at a = nt - nc, b = nt * nc;
    at Tr = 1 - (R0 + (1 - R0) * ddn);
    Tr) R = (D * nnt - N * (ddn > 0 ? 1 : -1));
...
    E * diffuse;
    = true;
...
    refl + refr)) && (depth < MAXDEPTH)
...
    D, N );
    refl * E * diffuse;
    = true;
...
MAXDEPTH)
...
survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following Serdar's
if;
radiance = SampleLight( &rand, I, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (abs(radiance.x) < 0.001 ||
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance.x > 0 ? 1 : -1);
...
random walk - done properly, closely following Serdar's
ive)
...
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
    
```

Why is the theoretical peak performance of a GPU typically much higher than that of a CPU?



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

Explain the concept of streaming processing.

```

...ics
& (depth < MAXDEPTH)
...
= inside ? 1.0f : 0.0f;
nt = nt / nc; ddn = ddn * 2.0f;
s2t = 1.0f - nnt * nnt;
D, N );
)
...
at a = nt - nc; b = nt + nc;
at Tr = 1 - (R0 + (1 - R0) * ddn);
Tr) R = (D * nnt - N * (ddn > 0.5f) ? 1.0f : 0.0f);
...
E * diffuse;
= true;
...
efl + refr)) && (depth < MAXDEPTH)
D, N );
refl * E * diffuse;
= true;
...
MAXDEPTH)
survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely following the
if;
radiance = SampleLight( &rand, I, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (survive)
...
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
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
vive)
...
at3 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

How does a GPU handle conditional code?

```

...
    & (depth < MAXDEPTH)
...
    if (inside ? 1 : 0) {
        nt = nt / nc; ddn = ddn * ddn;
        cos2t = 1.0f - nnt * ddn;
        D, N );
    }
...
    at a = nt - nc; b = nt + nc;
    at Tr = 1 - (R0 + (1 - R0) * ddn);
    at R = (D * nnt - N * (ddn * ddn));
...
    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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (cosThetaOut > 0)
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    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;
    
```



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

```

...
    & (depth < MAXDEPTH)
...
    inside ? 1.0 : 0.0;
    nt = nt / nc; ddn = ddn * ddn;
    cos2t = 1.0f - nnt * ddn;
    D, N );
    )
...
    at a = nt - nc; b = nt * nc;
    at Tr = 1 - (R0 + (1 - R0) * r);
    Tr) R = (D * nnt - N * (ddn *
...
    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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (cosThetaOut > 0)
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    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;

```

Two kernels executed on the same GPU are executed with different work group sizes: one with 256 threads per work group, one with 512. Why did the programmer not use the same work group size in both cases?



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

What is stream compaction?

```

...
    & (depth < MAXDEPTH)
...
    if (inside ? 1 : 0) {
        nt = nt / nc; ddn = ddn * 2;
        cos2t = 1.0f - nnt * nnt;
        D, N );
    }
...
    at a = nt - nc; b = nt * nc;
    at Tr = 1 - (R0 + (1 - R0) * r);
    at R = (D * nnt - N * (ddn * ...
...
    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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (cosThetaOut > 0)
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    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;
    
```



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

Why does OpenGL have a native_sqrt as well as an sqrtf?

```

...
    & (depth < MAXDEPTH)
...
    if (inside ? 1 : 0)
    {
        nt = nt / nc; ddn = ddn * ddn;
        cos2t = 1.0f - nnt * ddn;
        D, N );
    }

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

    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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (cosThetaOut > 0)
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
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;
    
```



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

Is self-modifying code possible on a modern processor? Under what conditions?

```

...
    & (depth < MAXDEPTH)
...
    = inside ? 1.0f : 0.0f;
    nt = nt / nc; ddn = ddn * 0.5f;
    cos2t = 1.0f - nnt * nnt;
    D, N );
    )
...
    at a = nt - nc, b = nt + nc;
    at Tr = 1 - (R0 + (1 - R0) * r);
    Tr) R = (D * nnt - N * (ddn
...
    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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (cosThetaOut > 0)
...
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    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;

```

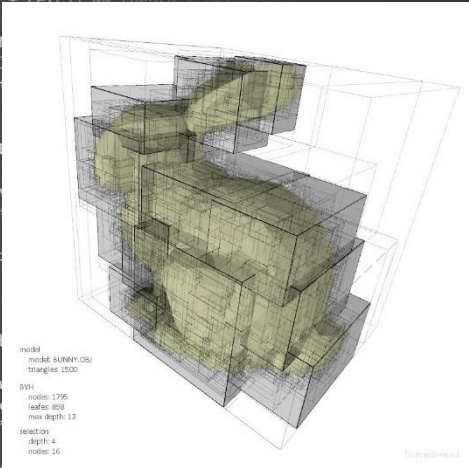


Now What

```

ics
(depth < MAXDEPTH)
& (depth < MAXDEPTH)
c = inside ? 1.0 : 0.0;
nt = nt / nc; ddn = ddn * c;
ps2t = 1.0f - nnt * ddn;
D, N );
0);
at a = nt - nc, b = nt * c;
at Tr = 1 - (R0 + (1 - R0) * c);
Tr) R = (D * nnt - N * (ddn *
E * diffuse;
= true;
fl + refr) * 88 / (depth * 1000);
model SUNNY.001
triangles: 1000
BOX
nodes: 1795
leafes: 809
max depth: 13
selection
depth: 4
nodes: 16
random walk = done properly, closely, and
(ive)
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

```



Now What

```

ics
& (depth < MAXDE
c = inside ? 1 :
nt = nt / nc, ddi
s2t = 1.0f - nnt
D, N );
)
at a = nt - nc, b
at Tr = 1 - (R0 +
Tr) R = (D * nnt
E * diffuse;
= true;
efl + refr)) && (
D, N );
refl * E * diffus
= true;
MAXDEPTH)
survive = Surviva
estimation - doi
df;
radiance = Sample
e.x + radiance.y
w = true;
at brdfPdf = Eval
at3 factor = diffuse
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, spdf
survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;

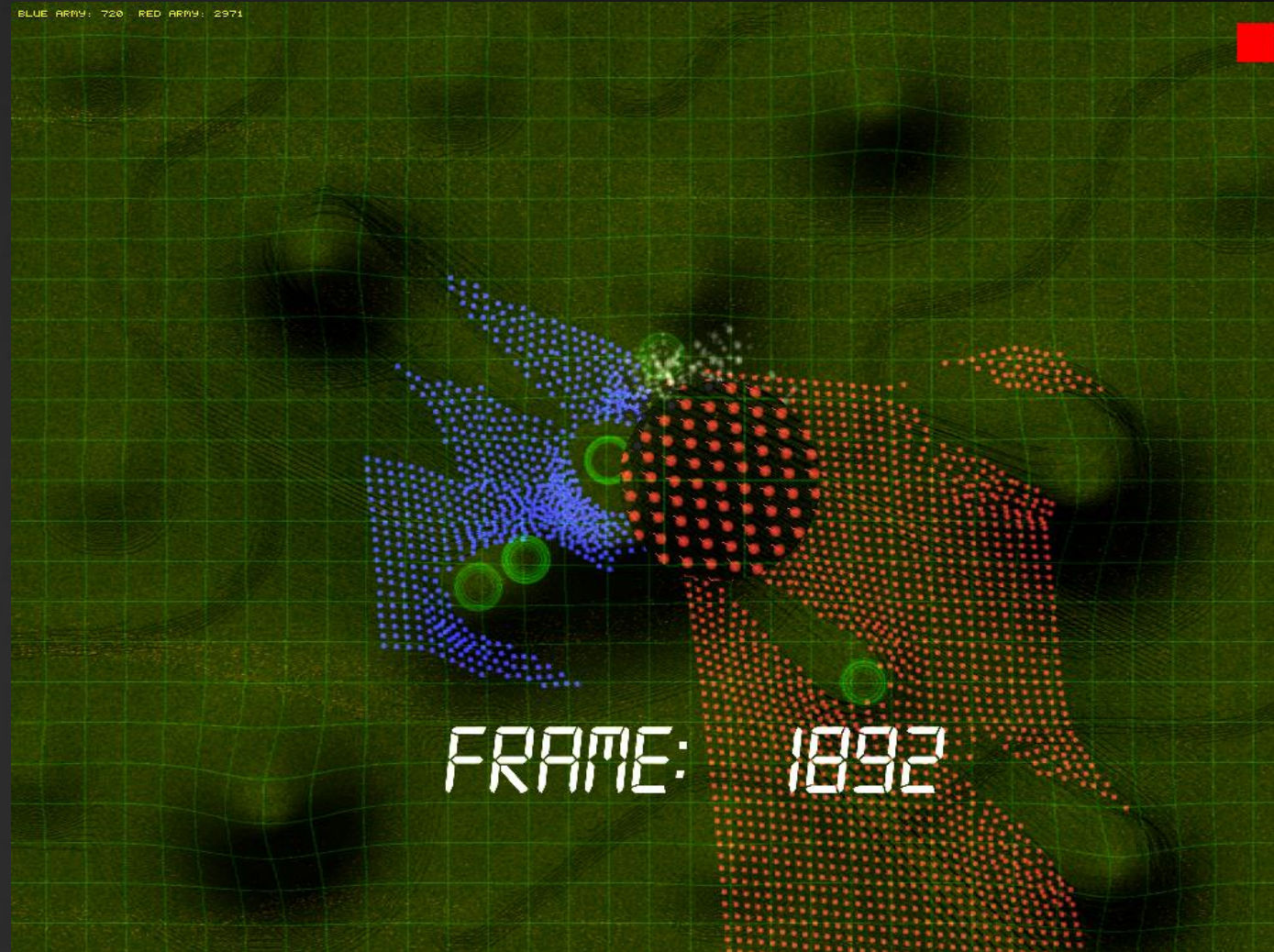
```



Now What

```

ics
& (depth < MAXDEPTH)
    if (inside ? 1 : 0)
        nt = nt / nc; ddn = ddn * ddn;
        cos2t = 1.0f - nnt * nnt;
        D, N );
        0)
        at a = nt - nc, b = nt * nc;
        at Tr = 1 - (R0 + (1 - R0) * ddn);
        Tr) R = (D * nnt - N * (ddn *
        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, &L, &light);
        e.x + radiance.y + radiance.z) > 0) && (depth <
        w = true;
        at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
        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
        vive)
        ;
        at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
        survive;
        pdf;
        n = E * brdf * (dot( N, R ) / pdf);
        sion = true;
    
```



Now What

```

ics
& (depth < MAXDEPTH)
c = inside ? 1 : 0;
nt = nt / nc; ddn = dot(N, L);
cos2t = 1.0f - nnt * ddn;
D, N );
)
at a = nt - nc, b = nt + nc;
at Tr = 1 - (R0 + (1 - R0) * ddn);
Tr) R = (D * nnt - N * (ddn > 0) ? a : b);
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, &L, &light);
e.x + radiance.y + radiance.z) > 0) && (depth <
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
andom 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;

```

COMPUTER PROGRAMMER



What my friends think I do



What my mom thinks I do



What society thinks I do



What my spouse thinks I do



What I think I do



What I actually do



```
ics
& (depth < MAXDEPTH)
{
    if (inside ? 1 : 0)
    {
        nt = nt / nc; ddn = ddn * ddn;
        cos2t = 1.0f - nnt * ddn;
        D, N );
    }
    else
    {
        at a = nt - nc, b = nt + nc;
        at Tr = 1 - (R0 + (1 - R0) * ddn);
        Tr) R = (D * nnt - N * ddn);
    }
    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, &L, &light);
    e.x + radiance.y + radiance.z) > 0) && (depth <
    w = true;
    at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
    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;
}
```



/INFOMOV2018/

