Welcome!
Today's Agenda:

- Grand Recap
- Exam
- Now What
Today's Agenda:

- Grand Recap
- Exam
- Now What
Recap
Recap
Recap – lecture 1

Profiling
High Level
Basic Low Level
Cache & Memory
Data-centric
Compilers
Fixed-point Arithmetic
CPU architecture
SIMD
GPGPU
Recap – lecture 1

Assumptions

- SURVIVE = Survive
- Probability = estimation - doing it properly

- Radiance = SimpleLight &\textbullet;\textbullet;\textbullet;\textbullet;\textbullet;
- radiances.x = radiances.y + radiances.z

- true;

- if (brdf) & (depth < 0.1)
- set factor = diffuse
- weight = MSIS2
direction = dot(N, R)
- E = ((weight * cosThetaOut))
- dir;
Recap – lecture 2

Red = u4 & (255 << 16);
Green = u4 & (255 << 8);
Blue = u4 & 255;

We need to go deeper

Red = u4 & (255 << 16);
Green = u4 & (255 << 8);
Blue = u4 & 255;
Recap – lecture 4
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 need 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 data.
▪ 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 is specific to functions that are defined elsewhere or cannot be inlined which are not readily available in vector versions.
▪ when the algorithm involves many branches that are not easily vectorizable.
▪ when floating point operations have to be reordered or transformed and it is unknown to the compiler whether the transformations are permissible with respect to precision, overflow, etc.
▪ when functions are implemented with lookup tables.

SIMD Basics

Other instructions:

```c
_mm128 c4 = _mm_div_ps( a4, b4 ); // component-wise division
_mm128 d4 = _mm_sqrt_ps( a4 );   // four square roots
_mm128 d4 = _mm_rcp_ps( a4 );    // four reciprocals
_mm128 d4 = _mm_rsqrt_ps( a4 );  // four reciprocal square roots (!)
_mm128 d4 = _mm_max_ps( a4, b4 );
_mm128 d4 = _mm_min_ps( a4, b4 );
```

Keep the assembler-like syntax in mind:

```c
_m128 d4 = dx4 * dx4 + dy4 * dy4;
```
Recap – lecture 7
Recap – lecture 8

Potato Salad
Some good cooks sprinkle grated pimiento cheese on this
4 cups diced cooked potatoes
1 cup sliced celery
3 hard-boiled eggs, cut up
1/4 cup finely cut onion or sliced green onions
1/4 cup sliced radishes
1 cup mayonnaise
1 tablespoon vinegar
1 teaspoon prepared mustard
1/4 teaspoon paprika
1/4 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 9
Recap – lecture 10
Recap – lecture 11
Recap – lecture 12

Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)

- Dual-Core Itanium 2
- Pentium 4
- Pentium
- 386

Transition (GHz)
Clock Speed (GHz)
Power (W)
Perf/Clock (LP)
Recap – lecture 13
Recap – Lecture 15

TOTAL RECAP

Get ready for the ride of your life.

TOTAL RECAP

[Image of Schwarzenegger's face]
INFOMOV – Lecture 15 – “Digest & Recap”
Before we start today's lecture, I would like to discuss the annoying habit of some of you of arriving late every single time.

Perhaps it's best to mention that again in half an hour.

Yeah that might be better.
Recap

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

Results:
- ?
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’.
Today's Agenda:

- Grand Recap
- Exam
- Now What
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
CPU and GPUs have fundamentally different core strategies for dealing with latencies such as memory access time. What are these strategies?
Example Questions

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

**Example Questions**

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

Explain the concept of streaming processing.
Example Questions

How does a GPU handle conditional code?
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

Example Questions

What is stream compaction?
Example Questions

Why does OpenCL have a native_sqrt as well as an sqrtf?
Example Questions

Is self-modifying code possible on a modern processor? Under what conditions?
Today's Agenda:

- Grand Recap
- Exam
- Now What
Now What
Now What
Now What

FRAME: 1892
Now What

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