Optimized C++

Spring 2017

CSC 361

instructor:	Ed Keenan
email:	ekeenan2@cdm.depaul.edu
office hours:	Thursday 3-5pm, 9-10pm in classroom or by email appointment
office:	CDM 830
phone:	(312)362-6747
website:	piazza.com/depaul/spring2017/csc361461 (Preferred communication)
lecture:	Lewis 1509, Thursdays 5:45-9:00pm
Desired to Learn (D2L):	d2l.depaul.edu (Grades, Viewing lectures, Announcements)
Version Control:	perforce: ***140.192.39.61:1666***

Description:

Implementation techniques to improve the performance and resource usage of C++ programs. This class will provide low level understanding of C++'s internal behavior that can be exploited to create faster performing software. Identify existing SISD (single instruction single data) implementation and refactor into SIMD (single instruction multiple data) to improve math centric software segments. Understand a system's dynamic memory usage to design and implement a custom high performance memory system. Topics include: performance enhancements through extended SIMD instruction set, dynamic memory usage, caching, implicit behavior, C++ language extensions, algorithms, streaming and profiling.

Prerequisites:

- Data Structures in Java or C++ (CSC 301 or CSC 383 or CSC 393)
- Computer Systems I (CSC 373)
- or instructor consent

Learning Goals:

- Students will be able to analyze software systems, identifying performance related issues in its design and implementation.
- Students will be able to identify and remedy execution performance issues related to data layout, processor caching, unintended compiler interactions, algorithmic considerations, data containers and supplied subsystems.
- Students will be able to refactor existing SISD (single instruction single data) into SIMD (single instruction multiple data) to improve math centric software segments.
- Students will be able to add performance profiling metrics to software systems to monitor the performance behavior as the software is refactored for optimization.
- Students will be able to apply understanding of dynamic memory knowledge to rewrite a custom memory system with performance exceed supplied systems from the compiler.

Grading:

15 % - C++ Basics (Proficiency) programs
65 % - Programming Assignments
- PA1: C++ Fundamentals – 5%
- PA2: Caching / Data Alignment – 10%
- PA3: Memory – 20%
- PA4: C++ Efficiency – 5%
- PA5: Math Optimizations - 10%
- PA6: File System - 10%
- PA7: Assessment - 5%

10% - Final Exam
10% - Final Project

NOTE: You must pass the final exam (***60% or higher***) to receive a passing grade in the class.
(It's not an easy exam)

Textbooks and printed resources

Required Texts:

- ***The C++ Programming Language:*** Stroustrup
  - ***4th Edition 2013 (new edition) or 3rd Edition (either are acceptable)***
  - Stroustrup Addison-Wesley Longman/Pearson, 2014. ISBN: 978-0321563842
- ***STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*** (2nd Edition) 2001 - Musser, Derge, Saini, ISBN-13: 978-0201379235
- ***Additional material provided in class***
  - Websites and handouts

Optional:
- **Effective C++ (3rd Edition),** 2005, Scott Meyers, ISBN-13: 978-0321334879
- **C++ for the Impatient,** 2013, Brian Overland, ISBN-13: 978-0321888020

Additional Material
- Will be provided by the instructor
- Lectures, links, SDKs and other corresponding material

Software

- ***Microsoft Visual Studio 2015 Enterprise Edition***
  - Visual Studio Enterprise 2015 with update 3 32/64-bit
  - C++ and C# install (future classes)
  - Microsoft Visual Studio 2017 is not used in this class.
  - Microsoft Visual Studio Community edition is not used in this class
- ***Perforce - Visual Client (p4v)***
  - www.perforce.com
- Download and configuration instructions will be provided in class
  - Server address: ***140.192.39.61:1666***

Programming assignments:

Standalone programming assignments to reemphasize the optimized programming concepts.

Program Assignment #1: ***C++ Fundamentals***

- C++ fundamentals
  - Class, methods, linked lists, pointers, references, etc
- Convert from a Java or C# perspective to C++
- Learn how to use version control and process

Program Assignment #2: **Caching / Data Alignment**

- Identifying data layout and alignment for supplied data structures and C++ classes
- Rework several data structures to reduce memory size
- Rework the supplied linked list data structure to a hot / cold data structure

Program Assignment #3: **Memory System in Real-time**

- Create a memory system within a heap
- Overloading new/delete C++ functions
- Placement New
- Create a different memory pools for the supplied C++ classes

Program Assignment #4: **C++ Efficiency**

- Implicit conversions
- Return value optimizations
- Proxy Objects
- Taking advantage of the compilers

Program Assignment #5: **Math optimizations**

- Algorithm optimization
- Refactor several matrices transformations to use SSE vector instruction set, Vector Unit or intrinsic math functions supplied by the compilers.

Program Assignment #6: **File Systems (Reading / Writing files)**

- Read / Write Buffers
- Data Layout

Program Assignment #7: **Programming Assessment**

- Real world C++ robust and optimized assessment
- Can you write optimal code that is also safe?

Final project

Given a particle system that dynamically updates several particles. Every particle and controlling object is dynamically allocated, often in very large blotted data structures. Each particle is controlled per frame by its own unique math transformations that are unoptimized. The memory allocations in the system are slow and fragments memory as the number of particles increase. Some of the system uses STL in a very inefficient manor. Many C++ classes are inefficient and naïve in nature.

Students refactor this system to:

- Maximize the number of particles to be processed with the given memory and performance constraint
    - Keep the frame rate constant to specified
    - Keep the memory within a fixed specified size
- Real working system
    - Program needs to cleanly be created and destroyed with no memory or resource leaks
    - Error free
    - Warning level 4 free or higher
- Dynamic monitoring for development, such as:
    - Total memory consumption
    - Memory / particle ratio
    - Performance cycles
    - Particle stats
- It's a contest
    - See how you can improve to original system.

Summary paper

Defining each major optimization and it relative performance
Expected length 2-3 pages in pdf format.

Final Exam

- A comprehensive final exam, covering the concepts of this class.
- Closed book written exam.
    - 11 week of class
- You must pass the final exam (60% or higher) to receive a passing grade in the class.
    - *It's not an easy exam*

C++ Basics (Proficiency) Programs

- There will be at least 9-10 weekly proficiency exams, validating your C++ fundamental knowledge
- The programs are easy to implement, but is a motivator for those who need encouragement to learn more thoroughly the basics C++ material.
- Topics range from:
    - Overloading, pointers, STL and Templates
    - vTable, inheritance, C-Strings and debugging
- If you already know the material, the assignments are a way to validate that knowledge

Perforce Submissions

- Everyone is expected to submit several submissions to perforce a week.
    - Minimum 5 significant (real) submissions on 3 separate days.
    - To promote incremental development and prevent last day rush.
    - Grade deduction will occur if not followed
- The biggest reason students get into trouble with software design:
    - Not starting the project early
    - Not working on the material frequently enough
    - Taking too large of a bite(byte) of the design
- Both are minimized with this *Perforce RULE*
- Even my simplest programs take 10-20 submissions.
    - For these project assignments my average is 40-400 submissions, so 5 will be no problem.
- Detailed perforce changelist comments are expected

Piazza Discussion forum

- Statistics show: students who participate more and help other students do better!
    - The correlation is ridiculous!
        - Poor understanding / poor participation.
        - Great understanding / Great participation
    - As you master the material, help others learn!
        - Want to be a Master programmer so master it!
- Everyone is ***expected*** and encouraged to participate on the Piazza discussion forum. All class-related discussion here this term.
    - At least one real question or response per week from every student.
- Everyone is expected to keep up with the material on Piazza and are responsible for its content. Critical class updates and directions will be presented there.
    - Not participating or reading the material on Piazza is not an excuse.
- All correspondence that is not personal in nature should be vectored through Piazza
    - Sensitive material, use Piazza private note, not email.

- The quicker you begin asking questions on Piazza (rather than via emails), the quicker you'll benefit from the collective knowledge of your classmates and instructors. I encourage you to ask questions when you are struggling to understand a concept.
- Keep the forum professional and positive, help each other out.
    - Karma really pays off here.
    - Help each other whenever you can.
        - There will be a section where you'll need help (trust me).

NOTE: Do ***NOT*** post until you have watched the entire lecture ***FIRST*** (in class or online)
This will prevent frustration on all sides (members asking or answering questions)

Collaborating together on programming assignments

- You are encouraged to work together
    - Use the Piazza forums heavy
    - Even share your material with others in the common directory
        - Obviously not the answers
- Everyone is 100% responsible for the work they do.
    - If you get help with a section of code,
    - Please refactor the code the ***snot out of it***
        - Comment and understand that material
        - Transform the code to ***make it yours***
    - Be able to answer ***any*** question regarding the code you commit
- System for Detecting Software Plagiarism
    - We will be using MOSS - Measure of Software Similarity (Stanford University)
        - Indicates possible code infringements (plagiarism)
        - MOSS - will detect the similarity independent of naming convention, indentation style or formatting, it compares abstract syntax tree of your code.
    - I will pursue any plagiarism/integrity violations aggressively, arguing for full expulsion from the university for the offenders.
        - Don't put me or you in this scenario
- If you gain significant support / help from another student
    - Fully disclose the support / help you had in a Readme.txt file submitted with your assignments.
        - Disclosing the help, is ***not permission*** for copying the code.
        - Only there to clarify and acknowledge help you were given from a fellow student.
- Modifying any Unit Test to alter the outcome results is also an ***Academic Integrity Violation***
- If you are stuck and find yourself even tempted to plagiarize
    - Ask for help !!!!
        - Use on Piazza -> Visit during offices hours, make an appointment
        - ***Don't ever compromise your integrity!***
- Material was uniquely created for this Class.

    - You indirectly by the process of tuition, "paid" for the contents and material of this class.
        - Do not share this ***copyrighted*** material in any form
        - It is design for your personal use, while enrolled in the Class.
    - Do ***NOT*** post any content or revealing material to any external website or forum outside of this class.
        - The Class Piazza forum is provided for this service, ask questions there, not on the internet (i.e. StackOverflow and other software forums)
- After you leave this class
    - You are expressly ***FORBIDDEN*** to provide or share the content with others.
    - Academic Integrity Violations can still be applied to students who provide material support to other students even after completion of the class.
- Just follow the golden rule:
    - ***"I have neither given, nor received, nor have I tolerated others' use of unauthorized aid."***

Tentative Class Schedule

Date	Lecture	Activity	Due
Week 1	C++ Fundamentals General Optimizations Perforce	PA1 - C++ Class Basics1 - Overloading	Compiler Perforce
Week 2 B	Caching Data Alignment Hot/Cold Data Structures	PA2 - Hot/Cold structures Basics2 - Debugging	PA1 Basics1
Week 3	Pointers Memory System	PA3 - Memory (part A) Basics3 - Pointers	PA2 Basics2
Week 4	Memory System Detail Implementation Testing and Verification	PA3 - Memory (part B) Basics4 - C Strings	PA3 (part A) Basics3
Week 5	Implicit Conversions Proxy Objects Return Value Optimization	PA4 - Proxy, Implicit, RVO Basics5 - Inheritance	PA3 (part B) Basics4
Week 6	Intrinsics - SSE, SIMD Matrix Math	PA5 - SSE Math Basics6 - STL	PA4 Basics5
Week 7	File system Load in Place	PA6 - File System Basics7 - vTable	PA5 Basics6
Week 8	Memory Overloading Refactoring Particle System	PA7 - Assessment Basics8 - Templates ***Particle System*** *- start*	PA6 Basics7
Week 9	Profilers Strings PSE	PA9 – Extra credit Basics9 - Ellipsis	PA7 Basics8
Week 10	C++ 11 / Boost Review	Basics10 - Mystery?	Basics9 PA9
Week 11	Final Exam Performance Contest		***Particle System*** Basics10

March 31, 2017	Last day to add classes to SQ2017 schedule
April 7, 2017	Last day to drop classes with no penalty, Last day to select pass/fail option
April 8, 2017	Grades of "W" assigned for SQ2017 classes dropped on or after this day
April 13, 2017	Last day to select auditor status
May 12, 2017	Last day to withdraw from SQ2017 classes

Course Policies

Changes to Syllabus

This syllabus is subject to change as necessary during the quarter. If a change occurs, it will be thoroughly addressed during class, posted under Announcements in D2L and sent via email.

Online Course Evaluations

Evaluations are a way for students to provide valuable feedback regarding their instructor and the course. Detailed feedback will enable the instructor to continuously tailor teaching methods and course content to meet the learning goals of the course and the academic needs of the students. They are a requirement of the course and are key to continue to provide you with the highest quality of teaching. The evaluations are anonymous; the instructor and administration do not track who entered what responses. A program is used to check if the student completed the evaluations, but the evaluation is completely separate from the student's identity. Since 100% participation is our goal, students are sent periodic reminders over three weeks. Students do not receive reminders once they complete the evaluation. Students complete the evaluation online in CampusConnect.

Academic Integrity and Plagiarism

This course will be subject to the university's academic integrity policy. More information can be found at http://academicintegrity.depaul.edu/. If you have any questions be sure to consult with your professor.

Academic Policies

All students are required to manage their class schedules each term in accordance with the deadlines for enrolling and withdrawing as indicated in the University Academic Calendar. Information on enrollment, withdrawal, grading and incompletes can be found at: cdm.depaul.edu/enrollment.

Students with Disabilities

Students who feel they may need an accommodation based on the impact of a disability should contact the instructor privately to discuss their specific needs. All discussions will remain confidential.
To ensure that you receive the most appropriate accommodation based on your needs, contact the instructor as early as possible in the quarter (preferably within the first week of class), and make sure that you have contacted the Center for Students with Disabilities (CSD) at: csd@depaul.edu.

Lewis Center 1420, 25 East Jackson Blvd.
Phone number: (312)362-8002
Fax: (312)362-6544
TTY: (773)325.7296

Retroactive withdrawal

This policy exists to assist students for whom extenuating circumstances prevented them from meeting the withdrawal deadline. During their college career students may be allowed one medical/personal administrative withdrawal and one college office administrative withdrawal, each for one or more courses in a single term. Repeated requests will not be considered. Submitting an appeal for retroactive withdrawal does not guarantee approval. Information on enrollment, withdrawal, grading and incompletes can be found at: http://www.cdm.depaul.edu/Enrollment-Policies.aspx