Optimized C++

Spring 2019

CSC 461

instructor:	Ed Keenan
email:	ekeenan2@cdm.depaul.edu
office hours:	9-10:30pm in classroom or by email appointment
office:	CDM 830
phone:	(312)362-6747
website:	piazza.com/depaul/spring2019/csc461361 (Preferred communication)
lecture:	CDM 224, Wednesday 5:45-9:00pm
Desired to Learn (D2L):	d2l.depaul.edu (Grades, Viewing lectures, Announcements)
Version Control:	perforce: ***140.192.39.61:1666***

Description:

Analysis and 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. Analysis of existing software to help identify and remedy execution performance issues related to data layout, processor caching, unintended compiler interactions, algorithmic considerations, data containers and supplied subsystems. 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 403)
- Computer Systems II (CSC 407)
- Discrete Math (CSC 400)
- 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.
- Student will be able to model the whole system to optimally configure runtime data layout on hard drives (external media) for speed of execution and to minimize memory spikes.
- Students will be able to configure and analyzed results from a commercial analyzer and profiler to optimize an existing software system.

Grading:

10 % - C++ Basics (Proficiency) programs
55 % - Programming Assignments
{ Drop lowest PA: except PA4 and PA6 }
- PA1: C++ Fundamentals – 5%
- PA2: Caching / Data Alignment – 5%
- PA3: Memory Intro – 5%
- **** PA4: Memory Stress***– 15%
  - Mandatory - does not qualify for drop
- PA5: C++ Efficiency – 5%
- **** PA6: Math Optimizations*** - 10%
  - Mandatory – does not qualify for drop
- PA7: File System (Load in Place) - 5%
- PA8: Assessment - 5%
- PA9: Cow List – 5%

25% - ***Final Exam***
- Minimum score required to pass the class

10% - Final Project

NOTE: You must pass the final exam (***50% 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 2017 Enterprise Edition (not Community)***
  - MSDNAA Depaul – Visual Studio 2017 Enterprise
    - C++ and C# install (future classes)
  - Any other variants are not used in this class
  - Students are responsible keeping their development tools working

- ***Perforce Server***
    - Download and configuration instructions will be provided in class
    - ***Perforce – Helix Visual Client (p4v)***
        - https://www.perforce.com/downloads/helix-visual-client-p4v
        - 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 & #4: **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 #5: **C++ Efficiency**

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

Program Assignment #6: **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 #7: **File Systems (Load in Place)**

- Read / Write Buffers that are prepackaged offline
- Remove the memory allocations on creation
- Deal with big/little endian format
- Data Layout

Program Assignment #8: **Programming Assessment**

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

Program Assignment #9: **Boustrophedonic Lists a.k.a. COW List**

- 2D linked list with a unique topology
- Better at Linked lists

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 not optimized. 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.

Optimization Analysis Paper

- Graduate students must perform an optimization analysis for the particle system before they work on the system.
- 2-3 pdf paper detailing the current performance and a plan to improve the performance plan white paper.

Students refactor this system:

- 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 - *It's not an easy exam*

NOTE: You must pass the final exam (***50% or higher***) to receive a passing grade in the class.
(It's not an easy exam)

Readings

Textbooks are used for references and learning new topics. It is suggested that you research and investigate material and ancillary topics covered in the class through these books as needed. High performance programming requires both breadth and depth knowledge in the C++ language, therefore everyone's needs and research will vary based on their own experience and evolving mastery of the material.

Grading Scale:

93-100: A	87-89: B+	77-79: C+	67-69: D+	0-59: F
90-92: A-	83-86: B	73-76: C	60-66: D	
	80-82: B-	70-72: C-		

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 of ***five*** significant (real) submissions on ***three*** 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 five 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 will 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 time when you will need help from the class (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 or website
    - 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 or Project setting 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.
    - By the process of tuition, you "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."***

Miscellaneous

- ***Late Policies***
  - Due dates and times are verified by the submission record on the Perforce Server
    - No extensions are allowed
  - All assignments need to compile without warnings
    - Failure to compile "as-is" results in a 0 for the grade

- ***Memory Leaking***
  - For assignments that have memory tracking enabled
    - If an assignment is determined that its leaking memory
      - A deduction of 20% is applied to the grade of that assignment
  - Leaking status is provided during development

- ***Crashing***
  - Assignments are expected to work for a set duration
    (long enough to demo all the features)
    - A grade of 0 is given to any project that throws an exception, ends unexpectedly, crashes or hangs (not proceeding forward).
    - Crash – program locking up or quitting unexpectedly

Tentative Class Schedule

Date	Lecture	Activity	Due
Week 1	Course Overview C++ Fundamentals Perforce	PA1 - C++ Class Basics1 - Overloading	Compiler Perforce
Week 2	Caching Data Alignment Hot/Cold Data Structures	PA2 - Hot/Cold structures Basics2 - Inheritance	PA1 Basics1
Week 3	Pointers Memory System	PA3 - Memory Intro Basics3 – Pointers	PA2 Basics2
Week 4	Memory System Detail Implementation Testing and Verification	PA4 - Memory Stress Basics4 - C Strings	PA3 Basics3
Week 5	Implicit Conversions Proxy Objects Return Value Optimization	PA5 - Proxy, Implicit, RVO Basics5 - Debugging	PA4 Basics4
Week 6	Intrinsics - SSE, SIMD Matrix Math	PA6 - SSE Math Basics6 – STL	PA5 Basics5
Week 7	File system Load in Place	PA7 - File System Basics7 - vTable	PA6 Basics6
Week 8	Memory Overloading Particle System/Refactoring Profilers	PA8 - Assessment Basics8 - Templates ***Particle System*** *- start*	PA7 Basics7
Week 9	General Optimizations Strings PSE	PA9 – Cow List Basics9 - Ellipsis	PA8 Basics8 ***Analysis Paper***
Week 10	C++ 11 / Boost Review	Basics10 - Mystery?	PA9 Basics9
Week 11	Final Exam Performance Contest		***Particle System*** Basics10

University Dates (Drop, Withdrawal, Audit, Exam)

- https://academics.depaul.edu/calendar/Pages/default.aspx

University 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. Please see https://resources.depaul.edu/teaching-commons/teaching/Pages/online-teaching-evaluations.aspx for additional information.

Academic Integrity and Plagiarism

This course will be subject to the university's academic integrity policy. More information can be found at https://resources.depaul.edu/teaching-commons/teaching/academic-integrity/Pages/default.aspx.

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:
http://www.cdm.depaul.edu/Current%20Students/Pages/PoliciesandProcedures.aspx

Incomplete Grades

An incomplete grade is a special, temporary grade that may be assigned by an instructor when unforeseeable circumstances prevent a student from completing course requirements by the end of the term and when otherwise the student had a record of satisfactory progress in the course. All incomplete requests must be approved by the instructor of the course and a CDM Associate Dean. Only exceptions cases will receive such approval. Information about the Incomplete Grades policy can be found at
http://www.cdm.depaul.edu/Current%20Students/Pages/Grading-Policies.aspx

Students with Disabilities

Students seeking disability-related accommodations are required to register with DePaul's Center for Students with Disabilities (CSD) enabling them to access accommodations and support services to assist with their success. There are two office locations:

- Loop Campus – Lewis Center #1420 – (312) 362-8002
- Lincoln Park Campus – Student Center #370 – (773) 325-1677

Students who register with the Center for Students with Disabilities are also invited to contact Dr. Gergory Moorhead, Director of the Center, privately to discuss how he may assist in facilitating the accommodations to be used in a course. This is best done early in the term. The conversation will remain confidential to the extent possible.

Please see https://offices.depaul.edu/student-affairs/about/departments/Pages/csd.aspx for Services and Contact Information.

Proctored exams for OL courses (if applicable)

If you are an online learning student living in the Chicagoland area (within 30 miles of Chicago), you will need to come to the Loop campus to take an exam. Online learning students outside of the Chicagoland area are required to locate a proctor at a local library, college or university. You will need to take the exam within the window your instructor gives. Students should examine the course syllabus to find exam dates and the instructor's policy on make-up exams. Detailed information on proctored exams for online learning students can be found at http://www.cdm.depaul.edu/onlinelearning/Pages/Exams.aspx

Online office hours for OL courses (if applicable)

Faculty should be accessible to online students via phone, email and/or Skype.