Optimized C++

CSC 361

instructor:	Ed Keenan
email:	ekeenan2@cdm.depaul.edu
office hours:	REMOTE: 9:15-10:45pm Synchronous Zoom Meeting or by appointment
office:	CDM 830
phone:	(312)362-6747
website:	piazza.com/depaul/winter2021/csc361461 (Preferred communication)
lecture:	REMOTE: 6:00-9:00pm Synchronous Zoom Meeting
Desired to Learn (D2L):	d2l.depaul.edu (Grades, Viewing lectures, Announcements)
Version Control:	perforce: *140.192.39.61:1666*
All Communication:	Piazza for all communication, reply within 24 hrs during school week

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:

10 % - C++ Basics (Proficiency) programs
55 % - Programming Assignments
  { all PA: regrade 1 week later Max 70% }
  - PA1: C++ Fundamentals – 5%
  - PA2: Caching / Data Alignment – 5%
  - PA3: Memory Intro – 5%
  - PA4: Memory Stress– 15%
  - PA5: C++ Efficiency – 5%
  - PA6: Math Optimizations - 10%
  - PA7: File System - 5%
  - PA8: Cow List - 5%

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

10% - Final Project
5% - Extra Credit Assignment

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 2019 Enterprise Edition (not Community)***
  - MSDNAA Depaul – Visual Studio 2019 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*
    - o Download and configuration instructions will be provided in class
    - o *Perforce – Helix Visual Client (p4v)*
        - ▪ https://www.perforce.com/downloads/helix-visual-client-p4v
        - ▪ Server address: *perforce.dpu.depaul.edu:1666*

Programming assignments:

Standalone programming assignments to reemphasize the optimized programming concepts.

Program Assignment #1: *C++ Fundamentals*
- C++ fundamentals
    - o 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 (Reading / Writing files)**
- Read / Write Buffers
- Data Layout
-
Program Assignment #8: **Boustrophedonic Lists a.k.a. COW List**
- 2D linked list with a unique topology
- Better at Linked list

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.

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 take home programming final exam, covering the concepts of this class.
    - You will code several mini – programming tasks

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 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 <u>submissions</u> 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

- Previous classes have highly participated in Piazza and did quite well
    - To promote this secondary learning approach
- Statistics show: students who participate more and help other students <u>do better</u>!
    - 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.
- 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.
    - o 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
        - Any integrity violations will result in a failing grade for the course
- If you gain significant support / help from another student or website
    - o 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
    - o 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.
    - o 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.
    - o 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
    - o You are expressly **_FORBIDDEN_** to provide or share the content with others.
    - o 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:
    - o **_"I have neither given, nor received, nor have I tolerated others' use of unauthorized aid."_**

Miscellaneous

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

- *Memory Leaking*
    - o 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
    - o Leaking status is provided during development
- *Crashing*
    - o 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
- *Integrity Violation*
    - o Any form of integrity violation will receive an "F" letter grade for the course, no exceptions
    - o All material submitted is from this current offering of class, any material from the outside is considered a violation

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 – Cow List Basics8 - Templates *Particle System - start*	PA7 Basics7
Week 9	General Optimizations Strings PSE	Extra Credit Basics9 - Ellipsis	PA8 Basics8
Week 10	C++ 11 / Boost Review		Extra Credit Basics9
Week 11	Final Exam Performance Contest		*Particle System*

University Dates (Drop, Withdrawal, Audit, Exam)
- https://academics.depaul.edu/calendar/Pages/default.aspx

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 (312) 362-8002
• Lincoln Park Campus (773) 325-1677
• Email: csd@depaul.edu

Students who register with the Center for Students with Disabilities are also invited to contact Dr. Gregory 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.

Online office hours

Faculty should be accessible to students using Zoom, Skype or other similar platforms for the duration of the office hours. Faculty must be accessible on the designated platform for the duration of the office hours.