Real-Time Multithreaded Architecture

Spring 2019

CSC 588

instructor:	Ed Keenan
email:	ekeenan2@depaul.edu
office hours:	9-10:30 Classroom or by email appointment
office:	CDM 830
phone:	(312)362-6747
website:	piazza.com/depaul/spring2019/csc388588 (Preferred communication)
lecture:	CDM 216, Tuesday 5:45-9:00pm
Desired to Learn (D2L):	d2l.depaul.edu (Grades, Viewing lectures, Announcements)
Version Control:	perforce: ***140.192.39.61:1666***

Description:

Real-time constrained multithreaded architecture. Topics include multithreaded handle development, inter-thread communication, creating systems for user-defined callbacks, asynchronous loading and streaming of resources, non-blocking threading synchronization, design patterns and data driven messaging with time delivery constraints. Design and implementation of thread safe data management with atomic non-blocking synchronization. Exploration of real-time data driven messaging to allowing the run-time object data to control the behavior an application. Students will design, develop and implement a multithreaded real-time application (i.e. Audio engine) that integrates existing single and multithreaded middleware libraries.

Prerequisites:

- CSC 461 and (SE 456 or SE 450)

Learning Goals:

- Students will design, development and implement a multithreaded application with real-time constraints.
- Students will design and implement custom Handles to control resource contention.
- Students will explore inter-thread communication with safe data queuing.
- Students will design and implement user callbacks passed and tunneled between threads.
- Students will implement asynchronous loading and streaming of runtime resources.
- Students will explore thread safe data management with atomic non-blocking synchronization.
- Students will learn how to create real-time data driven messaging to allowing the run-time object data to control the behavior and flow of an application.

Grading

Programming Assignments

10% - Weekly Code/Video Submissions (approx. 8 assignments)

Week 6

30% - Milestone 1: Prototype: Basics Audio Engine

Week 11

50% - Milestone 2: Finished Audio Engine, Demo, and Documentation

10% - Playlist graduate students differing task

Textbooks and printed resources

Course material will be many supplied through class notes, handouts or online links.

- 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) -*** www.perforce.com
    - Server address: ***140.192.39.61:1666***

Topics will include:

Threads

- Concurrency
- Spawning and coordinating threads in C++
- Process, thread, stack data ownership

Handles

- Inter-process and inter-thread resource contention
- Atomic identification for resource pooling
- Thread safety and exception safety

Inter-Thread communication

- By direction queuing between threads
- Circular queues without conditional wrapping
- Updating and processing request between threads

Commands Passing

- Commands originated on main thread being services on other threads
- Tunneling commands
- Access data and relationship of commands variable output passing in a queue

XAudio2 Drivers

- Microsoft Audio driver
- Multithreaded audio system with its own threads, voices, callbacks, data marshaling
- Audio basics – sample rate, format, IFF file format, etc.

Callbacks

- Audio Driver callbacks interfacing to Main thread
- User define callback architecture – extending base classes
- Marshalling callbacks between threads and thread hopping

Asynchronous loading

- Spawning multiple threads for file loading
- Coordinating loading between threads with signaling
- Zero blocking atomic data structure for loading

Thread Management

- Thread pooling and reuse management
- Controlling data on the stacks
- Ownership issues

Design Patterns

- Complexity added with multithreaded environment on Design Patterns
- Multiple threads interactions for Visitor, Command, Observer, and Singletons

Timer Coordination

- Single coordinating clocked based events on Main thread
- Adaptive scripting with a resolution Audio thread

Debugging / Tracking

- Debugging and tracking with I/O and printing
- Demos and interactive scenario development
- Resource tracking

Milestone 1: Prototype Basics Audio Systems (30%)

Students will design and build simple audio systems with fundamental multithreaded components. The goal of this milestone is to understand the primitive features in the XAudio2 API and building blocks for a more complex full feature multithreaded system (final project). Students will explore the internal features and resource usage of the XAudio2 API such as Audio playback, data management, Callbacks and event sequence synchronization. In addition to the Audio system basics, students will develop multithreaded components such as custom Handles, inter-threading queuing system, thread tracking, and multithreaded testing environment. This milestone is a shake out many mini-demos to prove that the completed audio engine is achievable. Engine in C++, with video and summary of each component.

- Several Mini-demo prototypes:
    - Playback
        - Voices
        - Attributes (Volume, Pan, Pitch, etc.) control
        - Looping
        - Loading/unloading source wave data
        - XAudio2 Callbacks
    - Audio Management
        - Handles
        - Buffer reuse/control/tracking
        - Voice management
    - Scripting
        - Timer events
        - Coordinating between threads
    - Stitching
        - Seamless transitions between voices
- Complete Code base
    - 100% working game stored in perforce
    - Needs to compile and run
    - Not compiling or working (FAIL)
- Feature List
    - Self-grading feature checklist (supplied PDF)
    - Link to Video Demo
- Video Demo
    - YouTube Video Demo
    - Demo of the game
    - Identifying features in the game
    - 5-10 minutes with clear audio commentary

Milestone 2: Finished Audio Engine (50%)

Students will design and build a single multithread runtime audio game engine library written in C++. This system has many complex real-world features and constraints to all single threaded applications to interface with multithreaded engine. Required features of this engine are asynchronous real-time loading/streaming, dynamic preemptive priority system, seamless transitioning between discrete sound calls without application intervention, application define user callbacks, data driven script playback and more. In addition, a compelling demo to show off the newly constructed engine.

- Single integrated C++ library
  - Incorporates the components fully working from Milestone1
  - Static Library or DLL
- Required Features
  - Basics audio features (milestone 1)
  - Priority system
  - User Callbacks
  - Asynchronous file loading / streaming
  - Interpretive Scripting management
  - Queue Stitching
- Compelling Demo
  - Show off your work
- Documentation
  - UML diagramming of the system
  - Description of the features
  - Architecture design document (TBD)
- Complete Code base
  - 100% working game stored in perforce
  - Needs to compile and run
  - Not compiling or working (FAIL)
- Feature List
  - Self-grading feature checklist (supplied PDF)
  - Link to Video Demo
- Video Demo
  - YouTube Video Demo
  - Demo of the game
  - Identifying features in the game
  - 5-10 minutes with clear audio commentary

Playlist Master's requirement (10%)

Playlist sometimes called Sound Definitions - choreographed script generally created by the sound designer. This script controls the playing of audio samples on specific virtual tracks; controls the sample's playback rate, looping, cross-fading, and many other effects.

- Playlist demo
    - A choreographed script generally created by the sound designer. This script controls the playing of audio samples on specific virtual tracks; controls the sample's playback rate, looping, cross-fading, and many other effects. All of the virtual tracks specified are relative to each other, in reality the virtual tracks will be assigned to physical hardware tracks depending on the current state of the audio system. The Playlist may also have markers for beat and loop transitions, signals to the applications for external events, or may receive dynamic input from the user as dynamic input for internal variables of the Playlist.
    - 3 sample playlists will be provided
        - Controlling panning, fading, tracks, etc over time
        - Control of external sound calls
- Demonstration with the milestone2 project

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 be compiling without warnings
    - Failure to compile “as-is” results in a 0 for the grade
- ***Crashing***
  - For assignments that work for a set duration (long enough to demo all the features) but then crash after time.
    - A deduction of 20% is applied to the grade of that assignment
    - Crash – program locking up or quitting unexpectedly

Tentative Class Schedule

Date	Lecture	Activity	Due
Week 1	Course Overview Audio Engine Features	PA1 – Xaudio2 play	Compiler Perforce
Week 2	XAudio2 Threads Sample Game	PA2 – Audio Voices	PA1
Week 3	Handles Exception Safety	PA3 - Handles	PA2
Week 4	Inter-Thread Communication Callbacks	PA4 - XAudio Callbacks	PA3
Week 5	Testing Audio Command Passing	MS1 - Milestone 1	PA4
Week 6	Audio Threads Thread Management	PA5 - Audio Threads	MS1
Week 7	Asynchronous loading Multithreaded Patterns	PA6 - Async Loading	PA5
Week 8	Priority System Scripting systems	PA7 - Priority	PA6
Week 9	Timer Coordination	PA8 - Scripting	PA7
Week 10	Debugging / Tracking	Final Project	PA8
Week 11			Final Project

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. 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