Architecture of Real-Time Systems

Winter 2017

SE 456

instructor:	Ed Keenan
email:	ekeenan2@depaul.edu
office hours:	Thursday 3-5pm Office, 9-10 Classroom or by email appointment
office:	CDM 830
phone:	(312)362-6747
website:	piazza.com/depaul/winter2017/se456 (Preferred communication)
lecture:	CDM 216, Tuesdays, 5:45-9:00pm
Desired to Learn (D2L):	d2l.depaul.edu (Grades, Viewing lectures, Announcements)
Version Control:	perforce: ***140.192.39.61:1666***

Description:

This course discusses the principles, styles, and patterns of real-time software architecture. Trade-offs and ramifications of software architecture with respect to performance, maintainability, and reusability, will be explored. The course will also investigate the design and implementation of real-time behavior and constraints for common Design Patterns such as Observer, Visitor, and Strategy. Finally, the course will demonstrate how creation of real-time Data Driven environment allows the run-time object data to control the behavior and flow of an application. These topics will be discussed in the context of best practices in software engineering such as, iterative development, testing, and continuous integration.

Learn the principles, styles, and patterns of software architecture and framework design in the context of computer game development. Emphasis of the macro/micro design patterns will be incorporated to control the complexity and efficiency of the application.

This is intensive software development course using the C# - windows environment. Students will work on the understanding of the requirements, design, and implementation of real-time individual components of an application. A final project of developing a complete computer game with the emphasis on controlling the real-time tradeoffs and behavior's will be emphasized throughout the class.

Software engineering aspects throughout the class will be emphasized. You will learn software design principles such as design pattern and software architecture styles that can be applied in the design of object-oriented frameworks for game development, i.e., object-oriented game engines. We will explore the trade-offs and ramifications of software architecture and design choices with respect to performance, maintainability, and reusability, etc. We will also discuss how some of the best practices in software engineering such as, iterative development, testing, and continuous integration, can be applied in game development.

Prerequisites:

- Required:
    - CSC 403: Data Structures II
        - Java, C# or C++ language programming ability
        - Basic data structures
        - Introduction to Object Oriented concepts

Learning Goals:

- Students will be able to identify real-time structures and layout.
- Students will be able to develop components such as update loops, interactive graphics, collisions, audio, input, and animation.
- Students will be able to understand and implement real-time behavior and constraints for common Design Patterns such as Factory, Observer, Visitor, Composite, Commander, State, Strategy, Null, Singleton, Flyweight, Iterator, and Memento.
- Students will learn how to create real-time Data Driven environment allowing the run-time object data to control the behavior and flow of an application.
- Students will reinforce their Software Engineering practices such as Iterative development, Testing, continuous integration and documentation.

Grading

Weekly

10% - Code/Video Submissions (code + video)

Week 6

15% - Milestone 1: Prototype: (*Moving animated alien sprites in a grid)*

Week 11 -

10% - Final Exam

15% - Design paper

50% - Milestone 2: Space Invaders game

Textbooks and printed resources

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

- 2 Required Books
    - ***Head First Design Patterns*** by Freeman, Bates, Sierra, Robson, publisher O'Reilly, November 2004
        - ISBN 978-0596007126
    - ***Head First Object-Oriented Analysis and Design*** by McLaughlin, Pollice, West, publisher O'Reilly, Dec 2006
        - ISBN 978-0596008673

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# install and C++ (future classes)
  - Microsoft Visual Studio 2015 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***

Topics will include:

Game Fundamentals

- Game Loops
- Timers
- Audio
- State Machines
- Input

2D Games

- Sprites
- Collision
- Movement
- Images loading / manipulation

Design Patterns

- Singleton
- Factory
- Null Object
- Flyweight
- State
- Strategy
- Observer – Publisher/Subscriber
- Memento
- Data Driven

Project Management

- Iterative Development
- Scheduling

Milestone 2: Space Invaders Game (50%)

Students will design and build a clone of the arcade version of Space Invaders. This cloned game will be written in C# in the AZUL framework using MODERN software engineering principles, including design patterns, iterative based development, robust and orthogonal systems. Minimum of 10 design patterns must be used.

- Stand alone Game application will include but limited to:
  - Select Screens,
  - Game cycling
    - (attract mode->select->enter->game over->select),
    - 2 player mode
  - Score,
  - Player movement and control,
  - Alien Grid,
  - Animation,
  - UFOs,
  - Missiles,
  - Bombs,
  - Erodible shields,
  - Graphics effects
  - Sounds
- 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
  - Indentifying features in the game
  - 5-10 minutes with clear audio commentary

Game Components

Individual game components will be designed and discussed during each class. Each component uses several design patterns. Components are implemented in progressive order, based on complexity and number of design patterns introduced. Features of each component will be discussed and evolved as the class progresses. List of some of the components:

- Input Manager
  - Input recorder / manager
  - Ability to search for specific button combos
  - Clean abstraction and mapping

- Sprite / Animation system
    - Create a sprite system that displays the sprite image
    - Animate the sprite to display an different image at a time
        - Set the series of images to cycle through
        - Set the timing / speed to display the images
    - Ability to reuse sprite images instead having duplicate images loaded at the same time.
    - Should be general enough to display any sprite used in the game
        - Alien Ships, Shields, Player, UFO, missiles
- Collision system
    - Create a collision system that determines collisions between the missiles and collideable objects
        - Such as the Aliens, Player, Ship, UFO, Collision subpart in the shield
    - The collision system should be able to determine the if any missile hit the collision box.
        - Ideally this system should be able to intersect with missile box with the target box
        - Determine the state of the collision, {non-intersection, or intersection}
    - This system should be able to be used with any collision needs in this game:
        - Aliens, Player, Ship, UFO, Collision subpart in the shield
- Shield collision and display system
    - Determine how to have the shield be dissolving and eroding by missiles from the aliens
    - Determine how the shield is dissolved and eroded by missiles from the player
    - Shield as a protector
        - It protects the player, from the alien missiles
    - Drawing mechanism to show the erosion
    - Determine the underlining collision grid and update mechanism
- Sound system
    - Create a system that loads and plays static waves in the game
    - Allow other systems to call single of sound waves for their respective effects
    - Ability to control individual volumes of each sound playing
    - Allow many different sounds to be playing at once,
        - i.e. multiple explosions overlapping
    - Gross sound controls
        - Mute, game overall volume

Design paper (15%)

- Engineering Design Documentation
    - Arrangement of design document (like a book)
        - Overall design / High level view
        - Component Discussions
            - Should discuss every Design Pattern used in detail
            - Min 10 design patterns (must match YOUR code)
        - Post-Mortem
            - Improvements, Commentary
    - Discussion of all the major systems
        - Each system (component)
            - UML of the system
        - Design Patterns Used in component
            - 2+ pages for each Design Pattern
            - Diagramming and discussion

        - Descriptions of the interactions between components
        - Any discussion of trade-offs or problems you overcome
    - In general UML diagrams
        - Structural diagrams
        - Sequence diagrams (if needed)
    - How do you know if your wrote enough
        - This report should give:
            - A good understand of what you did
            - Impress a future employer
        - Should be clear that you are a Software Architect
    - Page length Expectations
        - 25-35 pages in PDF format
            - don't freak - UML diagrams take space

Milestone 1: Prototype: Moving animated alien sprites in a grid (15%)

A prototype milestone drop will happen in the 6th week. This milestone consists of basic alien sprites animating in a grid, moving as a collection back and forth.

Reality Check: minimum systems working to finish the project.

- Graded all or nothing Full Credit or No credit
- Due date doesn't change

Validates critical systems

- Sprite
- Sprite Batch
- Timer
- Animation
- Creation Factory
- Managers

Design Document - Prototype

- Not graded
    - Allows feedback
- Must include critical design patterns used in the Prototype

Complete Code base

- 100% working game stored in perforce
- 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
- Indentifying features in the Prototype
- 2-5 minutes with clear audio commentary

Final Exam (10%)

Final exam covering the concepts and the material of the class

- In class exam
  - Closed BOOK!
- Key concepts and material from class
- Object Oriented Concepts
- Design Patterns
  - Q & A
  - Identification

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're 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)

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

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 Schedule:

Week	Lecture	Component	Patterns	Due
1	Overivew UML C# Language	Linked List	Manager	
2	Design Patterns Architectural Process	Sprites	Singleton Factory	PA1 – Manager Proto
3	Management of Objects	Sprite Batch	Flyweight Null Object *(Manager)*	PA2 – Engine Demo
4	Timed based updates	Timer Animation	Commander *(Priority Queue)*	PA3 – Sprite System
5	Alien Grouping Movement	Alien Grid	Composite Observer	PA4 - Animation
6	Collisions Collision Pairing	Collisions	Visitor Iterator	**Milestone 1 (MS1)** **Prototype**
7	Rendering Effect Coarse to Fine	Shields Bombs	State Strategy	PA5 – Collection Proxy
8	Sound System Handles	Audio	*(Handles)*	PA6 - Collision
9	Game Flow	Game Mode Cycling 2 Player Mode	*(Data Instancing)*	PA7 – State & Strategy
10	Attract Modes Replay	Attract Mode	Momento	
11				**Final Exam** **Milestone 2** **Design Doc**

January 9, 2017	Last day to add classes to WQ2017 schedule
January 15, 2017	Last day to drop classes with no penalty
January 16, 2017	Grades of "W" assigned for WQ2017 classes dropped on or after this day
January 20, 2017	Last day to select auditor status
February 19, 2017	Last day to withdraw from WQ2017 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