Syllabus

Software Development Project - COMP 440/441

8:00 – 8:50 MWF Science 201

Fall 2007

Instructor:	Frank McCown
Contact:	501-279-4826, HU Box 10764, fmccown@harding.edu
Home Page:	http://www.harding.edu/fmccown/
Office Hours:	Science 208: 10 – 11, 3 – 4 MWF and 2 – 5 TR or by appointment

Course Description

The capstone course for Computer Science and Computer Information Systems majors. Students will complete the development of a computer application in a simulated on-the-job environment through the analysis, design, programming, and testing phases of the software life cycle. Prerequisites: All classes required by the major.

Class Objectives

1. Learn the importance of teamwork and communication in the development of software.
2. Simulate as much as possible a competitive, real-world software development situation.
3. Develop software using an organized approach from analysis to implementation.
4. Experience working under pressure and deadlines.
5. Bring glory to God by utilizing the talents He has given you and creating something amazing.

Background

Since 1993, the Computer Science Department has required teams to produce game software for the capstone class. The games produced by these teams have been *tremendous* in all respects. The students have worked extremely hard and their finished software product was something they have taken tremendous pride in. We will continue to develop game software this semester because of the positive feedback from previous years and because the creativity involved in such production will motivate many to their peak performance. This type of development challenges you to reach a higher level of skill as you finish your college years, and the team approach will help you gain valuable interpersonal communication skills which are necessary in the professional world.

Several of the games from previous years are located in `\\cs1\SoftDev\Past Semesters`. Feel free to browse through their source code and documents. You may re-purpose code from previous years as long as you cite where you obtained the code in both your code documentation and your Implementation Summary. You may also install and run the games (if you can get the installation program to work), but **do not distribute** the software to anyone outside of the course- many of the games use copyrighted images, videos, and sound which could lead to litigation if the games were ever distributed widely.

Teams

You will be divided into teams of 3 to 4 people. Each of you will be given the opportunity to indicate one person who you would like to be on your team, and I will do my best to accommodate all requests. You will also be given the opportunity to elect team leaders; I will assign elected leaders to each team. Once the teams are formed, you will come up with a team name and trade contact information. You will also want to assign duties to each team member- someone who will be in charge of graphics, sound, AI, interface, networking, help, etc. I recommend assigning duties early so everyone is clear on what their responsibilities are can get started sooner rather than later.

All teams will meet as a class for the first five weeks, but after that we will just meet once a week for each team to give oral status reports. Team leaders will meet with me every week to give status reports.

Being a team leader is an honor, but it is also a huge responsibility. Team leaders are responsible for evaluating their teammates, assigning tasks, ensuring deadlines are being met, making final decisions on matters of disagreement, and motivating the team to do their best.

Keys to Success

You will undoubtedly experience plenty of pressure. The success of your project will depend on your team meeting deadlines which will be imposed. A software professional must learn to meet their obligations to their bosses and to their clients. Former students who are now in the work force have indicated that this aspect of the class has proven to be very valuable to them.

You will also be exposed to competition. This is something that you will face again in the working world. The best project in this class will be determined during the Acceptance Test; members of the winning team will receive 5% bonus added to their final grade. The winners will be immortalized on the plaque in the trophy case (first floor) and on the department's website at http://www.harding.edu/comp/awards_software.html. They will also have bragging rights at graduation and subsequent Harding alumni reunions.

Here are some winners from previous years:

Year	Game	Leader
1993	Yahtzee	Tyler Cutshall
1994	Superior Spades	Joel DeYoung
1995	Sunday School Software	Brad Choate
…		
2006 Spring	Pawns of the Galaxy	Chris Curry
2006 Fall	Lemmings	Anthony Sofio
2007 Spring	Pentigo Classico & Super Mario Squares	Jose Rodriguez & Daniel Bewley
2007 Fall	**Could your team be here?**	

Consider carefully from among your peers in this class who would serve as a good software development leader. Make no mistake about it: the **key** to successfully completing the project is going to be **great leadership**. And remember that it was Jesus who showed that leadership is about being a servant.

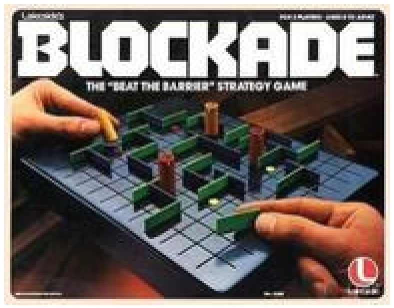
Expect to spend a lot of time in this class. This should not be surprising considering that this is *your* capstone class in *your* major field of study. A good portion of your grade will be determined by the amount of time and effort you spent helping your team successfully complete the project. Get your priorities straight and **get focused** on the task at hand! At the same time, do not put this class in front of all other relationships, especially your relationship with God. Paraphrasing Matthew 16:26: "For what shall it profit a man if he should win the software development project, and yet lose his own soul?"

For your software to be competitive, a dynamic and compelling GUI should be your biggest focus. To the typical user, the user interface is "the thing," and if "the thing" is clumsy and un-intuitive, it becomes the thing in the "trash can." The integration of sound and animation is also a major factor. If a picture is worth a thousand words, then a moving picture is worth millions. And a moving picture with sound has got to be worth even more. Think carefully about whether your interface should be 2D or 3D. Although there is usually an initial "cool" effect of a 3D interface, it usually takes much more time to develop, and sometimes it can hinder rather than help the user during game play. Last semester's winners both used a 2D interface.

Although each team would ideally like to keep what it is doing "under wraps," it's almost impossible. Other teams will eventually see each other's work sometime during the semester. Although it is not wrong to get new ideas from seeing someone else's work, I encourage you to avoid copying each other's ideas and work.

This Semester's Game

This year's project will be the development of a computerized version of the strategy game *Blockade*. *Blockade* is a "the Beat the Barrier" board game for two players, invented in by Mirko Marchesi and published by Lakeside in 1975. Marchesi later produced a game called *Quoridor*, a modification of *Blockade*, which won a Mensa Mind Game award in 1997. While several algorithms and implementations exist on the Internet for *Quoridor*[1], I have not been able to find an implementation of *Blockade*. The rules for *Blockade* are available from the class website and will be given to you in a separate handout.



[1] http://en.wikipedia.org/wiki/Quoridor

Game Requirements

Your implementation must conform strictly to the given rules so that each team's game can effectively play each other in the AI competitions (see below). If there are any ambiguities that you discover in the rules or the requirements listed below, please bring it to the classes' attention immediately so we can resolve it as soon as possible. The following are some general requirements for the *Blockade* software:

- It should conform to the given Blockade rules.
- It should run on a Windows XP machine. Optionally it would be nice if it ran on Vista as well. The application can be web-based (ran from a web browser) or run from an executable installed on the target machine. The XP machines in the Software Project Lab (Sci 203) will be the official machines on which the software must be successfully installed.
- It should require minimal outside instruction as to how to use. Children as young as age 10 should be able to run the software with just a little guidance by a human.
- It should have robust error handling (no hang-ups).
- It should allow user vs. computer from the same computer, user vs. user from the same computer, or user vs. user from two networked computers. Competing games do not have to be capable of playing each other over a network.
- The AI should be configurable by the user to account for different skill levels (beginner vs. expert).
- It should provide software to install/uninstall Blockade and any supporting software required.
- It should provide on-line help.

Development Tools

The tools that you choose to use are completely left up to the decision of the teams. The tools of choice in recent years have been the .NET Framework, the XNA framework, and Flash. There are little if no Harding-supplied graphic/audio tools, but GIMP is a free graphics package that may be useful. Other development tools that you may need will have to be borrowed or obtained as freeware. The department is not able to purchase any tools for you.

You may not purchase any software for this course without the pre-approved permission of all teams. Additionally, anyone who uses **pirated software** to produce any portion of their software will be dropped a letter grade and be assessed a 10% team penalty on the Acceptance Test; pirating software is stealing from your current and future peers, and it will not be tolerated.

Code Reuse

As indicated before, you may reuse code from previous semesters or borrow from open source projects on the Web. But you **must** cite your sources, in both the source code and in the Implementation Plan. Some of the algorithms that have been developed for *Quoridor*, for example, may be useful for developing an algorithm for *Blockade*. The reason I am allowing you to reuse existing code is because code reuse is ideally how software *should* be written, but in practice it is usually very difficult to do.

I encourage you to release any improvements you make to open source projects. You may want to make your own source code open source; keep in mind that many open source software licenses like the GPL have certain requirements about how your software should be made available if it uses other open source software.

Copyright

Copyright is the set of exclusive rights regulating the use of a particular expression of an idea or information. Copyright law protects the owner from others making copies or reproductions of their work. For example, Disney has exclusive rights to Mickey Mouse, and anyone who makes a mouse that looks similar to Mickey without Disney's permission is violating their copyright. On the other hand, the "fair use" doctrine of copyright law allows for some reproductions depending on several factors, including the purpose and character of the use, the nature of the copyrighted work, the amount and proportion of the whole work which was taken, and the effect of the use upon the potential market for or value of the copyrighted work. Therefore a political cartoon using Mickey is protected as fair use.

Many CS software projects in the past have used copyrighted work. Star Wars, Super Mario Bros., and Batman are a few examples of themes that projects have used in recent years. I am allowing teams to use copyrighted work because it usually will save you time, but **any game using copyrighted material may not be posted on the Internet or distributed to anyone outside of the team's immediate family.** I believe this constitutes fair use of copyrighted material. But if you would like to place your game on the Web or distribute it to friends, you should develop your own images, characters, sounds, etc. Keep in mind that when users are deciding which game they like best between a game using copyrighted work and one with original work, when all else is equal, they will usually choose the one with original work.

Grades

Standard letter grades: A = 90-100%, B = 80-89% C = 70-79%, D = 60-69%, F = 0-59%

Your final grade will be 65% *team* oriented and 35% *individual* oriented using these weights:

Requirements Document					5%	
User Interface Presentation					9%	
Implementation Plan					8%	
AI Competitions						
	Human vs. Human Competition				2%	
	AI vs. AI First Competition				2%	
	AI vs. AI Final Competition (AI Super Bowl)				6%	
Acceptance Test					33%	
	User Test	40%				Team
	Technical Test	60%				
			GUI	30%		
			Networking	30%		
			AI	30%		
			Website	5%		
			Help	5%		
			Installation penalty	(up to 5%)		
Bonus (to winner of Acceptance Test)					(5%)	
Mid-term Evaluation					10%	
Final Evaluation					20%	Individual
Computer Science Outcomes Assessment Exam					5%	

1% will be ***subtracted*** for each class meeting that you are absent. Absenteeism will also negatively affect evaluations.

If your final grade is 1% below the next highest grade, Computer Seminar attendance will be evaluated. 3 or fewer absences will grant you the better grade.

Schedule

Aug 20 – Introductions, project overview
22 – Turn in leaders and work mates, review calendar
24 – Establish teams, discuss Requirements Document
27 – Discuss individual evaluation methods, teamwork lecture, web page requirements
29 – User Interface lecture, UI Document requirements and presentation procedure
31 – Team Leader meeting
Sept 3 – Implementation Plan requirements
5 – Requirements of Oral Status Reports, review Acceptance Test
7 – Team Leader meeting; Requirements Document due by 5:00 pm in McCown's office.
10 – Human vs. Human Competition
14 – Oral Status Report #1
21 – Team Leader meeting
28 – Oral Status Report #2
Oct 5 – Team Leader meeting
8-11 – Mid-term individual evaluations
12 – Oral Status Report #3
19 – Team Leader meeting
26 – Oral Status Report #4
31 – First AI Competition
Nov 2 – Team Leader meeting
9 – Oral Status Report #5
16 – Team Leader meeting
30 – Oral Status Report #6
Dec 2 – Projects must be installed by midnight in Software Project Lab and website must be complete
3-8 – Acceptance Testing
5 – Outcomes Assessment exam, 7:00-9:00 am
6 – AI Super Bowl at 7:00 pm
8-10 – Final individual evaluations
9 – Software Showcase at 7:00 pm – winners of Acceptance Test are announced

Assessment

Harding University, since its charter in 1924, has been strongly committed to providing the best resources and environment for the teaching-learning process. The board, administration, faculty, and staff are wholeheartedly committed to full compliance with all criteria of the Higher Learning Commission of the North Central Association of Colleges and Schools. The university values continuous, rigorous assessment at every level for its potential to improve student learning and achievement and for its centrality in fulfilling the stated mission of Harding. Thus, a comprehensive assessment program has been developed that includes both the Academic units and the Administrative and Educational Support (AES) units. Specifically, all academic units will be assessed in reference to the following Expanded Statement of Institutional Purpose: **The University provides programs that enable students to acquire essential knowledge, skills, and dispositions in their academic disciplines for successful careers, advanced studies, and servant leadership.**

You will be given an Outcomes Assessment exam which will count as part of your grade in this course. The two hour, comprehensive exam will cover a variety of questions from these courses: Data Structures (245), Internet Development (250), Software Engineering (301), Operating Systems Concepts (310), Data Communications and Networking (311), Database Concepts and Applications (336), and Object-Oriented Programming (345).

What You Can Expect From the Instructor

Because this class is not like the traditional lecture/exam class, my primary job is to organize and direct this class in a way that simulates a real-world, professional software development working environment. To simulate such an environment, I will have to play several roles: 1) the customer, 2) the program manager, 3) the resident techie, and 4) the human resources arbitrator.

In such a working environment, it is the duty of the **customer** to make sure that the product being developed is the product desired. The customer in our scenario is the company which has contracted out the development of the game which is to be marketed to a given demographic. As the customer, I will be acting on and making suggestions based on what I feel is in the best interest of the target market.

It is the duty of the **project manager** to evaluate the overall effectiveness of each employee (you) based on the quality of work performed by the employee and the overall contribution made to his/her team. A great deal of the project manager's input is received from the direct supervisor (team leader) of the employee. Supervisors are also evaluated based on the quality of the leadership given to his/her team and the ability to foster effective communication and goodwill.

As the local **techie**, I will be available to give some technical advice when needed. My expertise lies primarily with .NET and Web programming. You will likely find other faculty and students more helpful in other areas. Just like the "real world", you will likely need to teach yourself to be successful.

There will inevitably be times when things get a little dicey and some feelings may get hurt. As your **H.R. arbitrator**, it's my duty to help in stilling the waters and dealing with unmotivated or irritated employees. The team leader should first try to work out whatever situation arises, but you may contact me if the leader is unable to resolve the issue. I will be as fair and even-handed as I can possibly be.

If you ever need assistance in this class or with anything else, please don't hesitate to come by during office hours or give me a call.

Students with Disabilities

It is the policy for Harding University to accommodate students with disabilities, pursuant to federal and state law. Therefore, any student with a *documented disability* condition (e.g. physical, learning, psychological, vision, hearing, etc.) who needs to arrange reasonable accommodations, must contact the instructor and the Disabilities Office at the *beginning* of each semester. (If the diagnosis of the disability occurs during the academic year, the student must self-identify with the Disabilities Director *as soon as possible* in order to get academic accommodations in place for the remainder of the semester.) The Disabilities Office is located in Room 102 of the Lee Academic Center, telephone, (501) 279-4019.

"And whatever you do in word or deed, do all in the name of the Lord Jesus,
giving thanks through him to God the Father." **Colossians 3:17**