

Computer Game Programming and Design

“Catalog” Description

Computer Game Programming and Design is a project-based computer science course. Students will learn to critique games, how to use a high-level game engine, and how to extend that game engine. Students will also receive an introduction to game studies, game design, and the game industry. Programming maturity and ability to program in teams is required. Prerequisite: CIS 203.

The term *computer game programming*, while sounding very specialized, is actually overly broad; in the industry there are gameplay programmers, AI programmers, server programmers, and, of course, graphics programmers. This course will provide an overview of what these specialties entail while students learn to use software engineering tools and to work in teams to develop an actual computer game.

Given the level of specialization described above, it is not surprising to hear that computer games are developed by teams of programmers (working with teams of artists; that is not going to be an option for our programs). Computer games are also often built on top of game *engines*.

A game engine is a framework for building a particular type of game. The engine often provides graphics, sound, and input subsystems, sometimes also providing game-oriented networking and user interface parts. The team takes the general capabilities of the engine specializes them to make the game in their heads. Some game engines provide everything, right down to the physics for three-dimensional models to interact with one another, and the programmers are almost left with nothing to do.

Actually, most game designs have something new, something different, something that makes them unique. It is seldom the case that a game engine directly supports unique features (game engine companies make their money by commoditizing the common subset of game features for a given genre) so game programmers almost always have something to do.

Student teams will learn a substantial game engine (as well as a version control system to keep all members of the team in synch as the code changes). They will be required to change the engine in several substantial ways before designing their game. Then, as a team, students will design a game which they will then implement using the game engine. As mentioned above, art assets will be limited

to what the team can produce (many rightly fear the *programmer art!*) or find in the public or common domain.

This course will look at games from different points of view: the “fun” factor, the cultural impact, the mathematical underpinnings, and the narrative power. Part of this will look at games as a means of teaching (it seems obvious that games teach *something*; the question is what and how can the designer shape what is taught) and as a means of story telling.

The “fun” factor is an important starting point in looking at computer games so students will begin playing games with the intent to review them. Students will present three or four game reviews over the semester where they will discuss what works and what fails to work in a particular game. By playing several different genre on several different platforms, students will begin to get a feeling for how to *improve* a game design. That is one of the first steps to being able to create a game design.

Classwork and Grading

Start early! Everything takes longer than you think.

It is important to learn to manage your time. You are expected to spend two or three hours outside class per class credit hour per week. I would bet that writing a digital game takes more effort; any takers? Reading for comprehension takes time and concentration. Programming takes even more time and even more concentration. Class presentations take research, development, and practice. This is not your only class but please make sure that you do not overbook your time.

Consider also that sometimes assignments are hard for you to finish. One of the instructor’s jobs is to help you through those assignments. Unfortunately, contacting the instructor and receiving an answer takes...time. Start early enough that you can seek help if you need it. This is only more true when working with a team; the members of your team depend on you getting your part done so make sure you start in enough time.

| Grading Criteria | |
|-----------------------|-----------------------|
| Assignment Type | Weight in Final Grade |
| Individual Programs | 20 |
| Group Project | 40 |
| Presentations | 20 |
| On-line Participation | 20 |

Individual Programs – Students will complete individual assignments designed to familiarize them with the git version control system, the game engine we are using, navigating large amounts of code and the like. Individual programming assignments will continue while teams are working on their initial designs.

Students code will be graded by the instructor but some projects will be exchanged with other students in the class. This grade include how carefully you evaluate other students' code.

Group Project – Or, as it said on a previous syllabus for this course, *The Big Game*. This is a group project and three-quarters of the grade will be shared by all participating members of the team (anyone not contributing to their team will receive a zero for the project and, in turn, a zero for the course). This grade will be based on standard coding rubrics: quality of documentation, impact of design decisions (to the good or ill), readable and maintainable code, and good use of software engineering tools. A portion of this grade will also be based on the quality and playability of the finished game. Note that this obviously has many dimensions but, in general, pretty games with crappy code will be marked down in comparison to rougher games with better code.

The remaining quarter of the grade will come from team evaluations. Throughout the semester each of you will be asked to grade the contributions of the members of your team. The rubric for this may change over time but it will be based on code quality and how well each student participates in the team: Do they make meetings? Do they finish what they say they will on time? Do they keep the team informed of their progress?

Presentations – The ability to present on technological topics using computer technology is an important skill to develop. There will be at least four individual presentations (and perhaps as many as five) and an equal number of group presentations. You will be evaluated (by the instructor and your peers) on how well you prepare, the structure of your presentation, the content of your presentation, and how well you use technology to support your presentation.

On-line Participation – Regular participation in the on-line class Website is expected. Three to four forum postings per week, including some number of answers to others questions should not take too much of your time. There will also be on-line assignments requiring feedback on other students' work and a game wiki for each development team. That wiki (and any forums you want to create along side it) will be the way your team communicates (with each other and with the artists).

Outcomes

This elective course fits into the learning goals of the Computer Science department by supporting the following outcomes (from CS Outcomes Document):

3. *knowledge of and the ability to apply programming fundamentals in at least two programming languages.*

The term project is designed to stretch each student's knowledge of programming. Students receive and must understand and modify a very large codebase that they did not write. The ability to decipher someone else's code is one large part of understanding how to program.

4. *knowledge of fundamental data structures and algorithms – including analysis of their correctness and complexity – related to various fields of computer science, and the ability to apply this knowledge to problems through the use of appropriate programming languages.*

Computer games use many data structures and algorithms; students must understand them and, importantly, be able to defend the choice of using any particular algorithm in terms of memory and time budgets.

5. *knowledge of computer architecture and organization, computer operating systems, and computer networks, and the ability to apply this knowledge to problems through the use of appropriate programming languages.*

Many if not most modern computer games use the network to communicate so that multiple players on multiple computers can play together. This class will examine network topologies and technologies used to do this efficiently. What happens when things go wrong will also be part of the discussion.

6. *competence and effectiveness in technical oral, written, and visual communication, particularly as they apply to the dissemination of technical information on subjects dealing with computing technology and applications.*

Students must develop a sense of what makes a game fun. Part of this requires hearing about many different computer games, both good and bad. Thus students will be required to review several different games over the course of the semester. This will entail playing the game, researching the game (who produced it, who designed it, when, what are the ratings (E, E10+, M, AO), what were contemporary reviews like), writing a report on the game, and giving an in-class presentation.

Teams will present their designs at the beginning of the project and will present their games at various points through the semester.

7. *knowledge of and skill in applying good practices in software engineering.*

It is necessary that students learn to understand software engineering tools (version control, configuration managers, automatic build and test frameworks)

in order to write a large, team-programmed project. The course project in this class is large and sufficiently complex to motivate the necessity of many of these tools.

8. *the ability to function effectively in teams to accomplish a common goal.*

The course project requires teams of three to five students.

9. *an understanding of professional, ethical, legal, security, and social responsibilities and issues, including an awareness of impact of computing on individuals, organizations and society.*

The introduction of game studies, the study of games as aesthetic cultural artifacts (art), necessarily includes a look at how games impact society (as well as the reverse). This leads to consideration of how stereotypes are reinforced or undermined in game play and presentation. Further, the question of game violence and the role of the game designer and game programmer in promoting it is explored.

Schedule

| Week | Topic | Reading Source |
|-------------|---|--|
| 01 | What is a game? Dimensionality in games. Using the engine. | Rouse, <i>Game Design: Theory and Practice</i> |
| 02 | Introduction to Game Studies. Reading foreign code. | Salen and Zimmerman <i>Rules of Play</i> |
| 03 | Importing models. Game review presentation. | |
| 04 | Violence in games. <i>Starpeace</i> design criteria. | <i>Rules of Play</i> |
| 05 | Project: Game design presentation. Game physics. | Bourg <i>Physics for Game Developers</i> |
| 06 | Interaction design. Interface design. More physics | <i>Physics for Game Developers</i> |
| 07 | Game physics. Introduction to Graphics. Game review presentation. | <i>Physics for Game Developers</i> |
| 08 | Project: Revised Game Design presentation. Graphics: 2D/3D/Combined. User interface | <i>Game Design: Theory and Practice</i> |
| 09 | Different input. Stereotypes in games. | <i>Rules of Play</i> |
| 10 | Game review presentation. Networking in games. | Topology paper. |
| 11 | Project: Alpha presentation. More networking. | |
| 12 | Sound in games. Automated testing. | |
| 13 | Game review presentation. Consoles. | |
| 14 | Project: Beta presentation. Users manual. | |
| 15 | Catch-up (Physics). | |
| Final Exam | Project Gold Master presentation. | |