Multidisciplinary Students And Instructors: A Second-Year Games Course

Nathan R. Sturtevant[1] nathanst@cs.ualberta.ca
H. James Hoover[1] hoover@cs.ualberta.ca
Jonathan Schaeffer[1] jonathan@cs.ualberta.ca
Sean Gouglas[2] sean.gouglas@ualberta.ca
Michael H. Bowling[1] bowling@cs.ualberta.ca

ABSTRACT

Computer games are a multi-billion dollar industry and have become an important part of our private and social lives. It is only natural, then, that the technology used to create games should become part of a computing science curriculum. However, game development is more than a massive programming endeavor. Today's games are largely about generating content within multidisciplinary teams. CMPUT 250 is a new computing science course at the University of Alberta that emphasizes creating games in multidisciplinary teams. This paper describes our experiences with the course, emphasizing the issues of multidisciplinary interactions: teaching, teamwork, and evaluation.

Categories and Subject Descriptors

K.3 [**Computers and Education**]: Miscellaneous

General Terms: design

Keywords: computer games, multidisciplinary students, multidisciplinary teaching

1. INTRODUCTION AND MOTIVATION

It is challenging to keep an academic curriculum relevant, especially in the quickly-evolving field of computing science. The commercial success of the computer games industry is but one of many recent developments in computing that should be reflected in a state-of-the-art computing science curriculum. For over two decades the Department of Computing Science at the University of Alberta has had a strong research group working in artificial intelligence applied to classical games. In 1999, the group began moving their focus towards the artificial intelligence needs of the commercial games industry, establishing strong ties with Electronic Arts, the world's largest games company, and BioWare Corp., a leader in role-playing games.

[1]University of Alberta, Computing Science Department
[2]University of Alberta, Department of History and Classics

It was within this context that our department began to plan a senior-level games-programming course for computing majors. The course would have been centered on the implementation of a large project on top of an established code base. However, over the past four years there has been an alarming decline in computing science (CS) enrollments at the undergraduate level. Introducing a senior undergraduate course that was limited to CS majors was not appealing as adding more course breadth to a declining population did not seem to be a good strategy.

Recommendations from industrial partners convinced us to look into a different approach, which also had the potential to strengthen ties with other departments on campus. Most game-development projects in industry have two notable features that have recently become more pronounced:

• Multidisciplinary teams. Early computer games were developed by teams of computer programmers. Today, the game-development team is composed of members with a multitude of skill sets, including programmers, writers, artists, and musicians. Computing scientists typically comprise about one-quarter of the employees at large games companies, and this percentage is likely to decline.
• Game content. Games originally were seen as almost exclusively a programming effort. Now game programming is largely directed towards the building of tools for the content developers (e.g., writers, artists, musicians) to integrate their work into the game. Building the game content can, in some instances, occupy the majority of work.

Our industry partners made the importance of multidisciplinary teams clear. John Buchanan, a former professor at the University of Alberta and the former director of university relations for Electronic Arts, wrote to us that,

"There are many attempts at building game courses across the academy. The games industry is a unique segment of the software engineering community; unique in that the teams that build the games are multidisciplinary. I receive a lot of requests to review courses in this area. My advice is always the same: make the course cross traditional boundaries; get engineers, artists and designers working together. ... If you have the flexibility to make the course multidisciplinary, then you must."

David Hibbeln, Director of Art for BioWare Corp., emphasized the same point:

"It is one thing to just train students to have a set of skills, but it is much more important to train them to use those skills within the context of an interdisciplinary project. The ability to harmonize with a group is one of the most important employee skills when working for a company such as ours."

This was our motivation for creating the multidisciplinary games course, CMPUT 250: Computers and Games. Several courses in a typical CS curriculum involve team projects, but few curricula involve teamwork with non-CS students. This skill is becoming increasingly important not only in the games industry, but in other fields as well, as computers are pervasive in society. One program that takes a similar approach to ours is at Carnegie Mellon [5]. Our course debuted in September 2005, with continued offerings since. Some of the important features of CMPUT 250 include multidisciplinary teaching, industrial partnerships, multidisciplinary teams for the course project, and a particular approach to project management.

This paper describes our experiences with CMPUT 250, including an overview of the course material, the tools used, and the course project. It assesses the strengths and weaknesses of the course, including lessons learned that might help others in planning and shaping their own program. All course material discussed in this paper is publicly available, including the software tools.

2. COURSE GOALS

CMPUT 250 exposes students to a broad array of topics related to both computer science and the creation of computer games. The pedagogical goals of the course include

• Creating a stimulating, collaborative learning environment for students to explore the theoretical and technical issues involved in the study and creation of computer games.
• Providing non-CS students an opportunity to learn from the intellectual traditions of CS in solving challenging tasks in general, and particularly in resolving issues in computer game development.
• Providing CS students an opportunity to work with and learn from faculty and students in the Arts, specifically with respect to the cultural, social, and economic issues of computer games and new narrative forms.
• Use the game-development cycle to provide students with a real-world learning experience that is complemented with theoretical and historical discussions of the games industry.

These specific pedagogical goals prompted a team-centered, problem-based learning approach. Our decision to replicate the game-design process not only gives students a perspective into how games are created, it also gives them unique skills in project management by working on open-ended projects. As such, the structure of the course requires a focus on the interdisciplinary nature of game design, which includes a vast list of potential topics and challenges.

3. INTERDISCIPLINARY LECTURES

CMPUT 250 has been taught by teams of five instructors, including faculty members from computing science, the humanities, and fine arts, covering the theoretical and technical issues of game design from both a science and a social science/humanities perspective. A pedagogical imperative of the course is to bring non-CS students into contact with the intellectual traditions of computing science while also expanding the cross-department experiences and interactions of the CS students.

Early lectures focus on topics immediately needed for the course project, including team management, game design, and narrative. At the same time, lab tutorials and exercises are used to provide students with the skills needed to begin work on the project (i.e. the tools for building the project).

Members of the CS faculty provide lectures on scripting, game development cycles, artificial intelligence, advances in computer game technology (such as sound and video development), and project management in a team environment. Some themes which are emphasized in these lectures include topics like algorithmic and hardware constraints as well as the history of specialized versus general-purpose hardware. Here, gaming provides a non-threatening introduction into the complexities of computing science.

Faculty from the humanities and social sciences have provided lectures on narrative in traditional and digital environments, the cultural aspects of gaming (including violence, sexuality, and community), and the design process for artwork and sound. These lectures often facilitate extended discussion from students of all backgrounds.

Interspersed with the main lectures are a series of review-like lectures given by the head instructor. These serve to tie together ideas being presented by the different instructors. Game postmortems, taken from Game Developer Magazine, are used to illustrate real-life examples of the things being discussed in the course.

One of the biggest challenges posed by a multidisciplinary class and a multidisciplinary set of instructors is the need to deliver the material at the right level of detail. Consider a CS lecture on artificial intelligence. If there is too much technical detail, the non-CS audience will not have the necessary background to understand the material. If the information is presented at too general a level, the CS students will learn nothing new. Thus, it is important that the instructors continually relate the technical material to the unique challenges of game development (such as strict memory and computational limitations). Even if students have previously studied material covered in the course, the relationship of the material with game design will provide a new perspective and understanding. This is an ongoing challenge for the course instructors.

4. COURSE PROJECT

The central exercise and means of evaluation for the course is a team project, intended to reflect the game development process as far as is practical. As Loren Andruko, former director of programming for BioWare Corp., stated,

"In most courses students work on assignments with well defined goals and deadlines. The problems faced in industry are never so simple and clear-cut, so it is valuable for students to work on projects with more uncertainty including a need to adapt and refine goals as the project progresses."

Teams of four or five students are required to generate a short, self-contained "module" for BioWare's popular game Neverwinter Nights^{TM}(NWN). Each team is regarded as a new development team, building a prototype or proof of concept for a new game. This is consistent with our aim to emphasize content generation and multidisciplinary teamwork, rather than the more technical aspects involved in game engine development.

To foster the kind of environment found in the industry, teams are also restricted in the backgrounds of their members. As far as possible, teams members are required to have different majors and each team was required to have at least one CS student so that all teams will be certain to have at least some technical capability. Teams are also strongly encouraged to attend the same lab sections so they will be sure to have some time when the entire team can be together.

4.1 Project Tools

Because CMPUT 250 is open to students of all backgrounds, we cannot assume that they will have any computing or programming background. This precludes us from using many of the available game engines. NWN has been a popular choice for use in the classroom (eg [1, 4]), as this game was one of the first to include a complete suite of tools to allow users to create their own game story within a game. The BioWare tools, Aurora and NWScript, provide a complete game-authoring package.

To minimize the technical requirements of development, students are encouraged to use the publicly available ScriptEase tool which has been developed at the University of Alberta [3]. ScriptEase presents a graphical interface that allows the user to select from predefined patterns of behavior to describe common scenarios and events in role-playing games such as NWN. For example, opening a chest and having a monster appear is a common pattern in role-playing games that is supported as a ScriptEase menu selection. Based on these patterns, ScriptEase automatically generates BioWare's NWScript that implement the behavior. ScriptEase is suitable for students who are not programmers, as it has even been used in several Grade 10 English classes as part of the short story writing curriculum [2].

These tools are primarily taught through tutorials and other exercises provided in the course lab, which students attend for two hours a week. The teaching assistants main role is to facilitate these lab sections, helping students with the course tools and any issues that arise in their projects. To ensure that all students learn these tools, there are short assignments which can be completed during the lab each week and a lab exam half-way through the course.

Note that a good lab infrastructure is necessary for such a course. Role-playing games can be built on relatively modest platforms using the NWN and ScriptEase toolkits, but extensive design and video work requires a high performance platform. Computers with dual displays are particularly useful for design and testing. The key infrastructure is a well-defined, rapid-response reporting system in place for hardware and software problems, as they can be expected to occur in any lab setting. Finally, game assets can grow to multiple gigabytes and students cannot be expected to backup their own projects, so it is important to have a repository and backup system in place as part of the lab environment.

4.2 Project Management

There are a number of important ways we have worked to help students manage their projects well. Teams are allowed to organize themselves, but must select one member as a "lead designer" and another as an "associate producer". The lead designer ensures that one person has final decision-making authority as regards features of the game itself. The associate producer is responsible for the operation of the team, including the scheduling of the project and ensuring that all course objectives are met. Beyond these two specific roles, teams are allowed to allocate people to tasks as they desire, with consideration to maintaining an equal workload.

A key danger in a project of this kind is that students will be over-ambitious, planning out a far larger game than their resources allow. In our first course offering this was the case, as several of the games developed in the course reflected effort far in excess of a normal course load. To avoid this, in subsequent offerings of the course we adopted three measures to help teams control the scope of their games.

First, games are limited in scope by the addition of a 'budget' system that limits the number of game elements teams can employ. Second, games are required to be playable in 10-15 minutes, assuming that the user knows exactly how to play the game. This requirement, suggested by BioWare, provides additional focus for the teams, encouraging students to build small, rich modules instead of sprawling unfocused ones. Finally, a 'producer' is assigned to each team to advise and critique them. Initially this was one of the lecturing faculty, but for the past year we have hired previous CMPUT250 students. Students work well, as they have experience in the course and can be easier to approach than a professor. We motivate them by rewarding the producer whose group creates the best game. Together, these limitations improve the quality and scope of the projects.

4.3 Project Overview

The project is split into seven milestones. In our 13-week semester, this means that there is a deadline every other week. By using many small deadlines, students gradually complete the work for the course, instead of leaving everything for the last minute.

Team Formation: To facilitate social interactions between students with diverse backgrounds, a pizza party is held during the second week of the course. This gives course members the opportunity to discuss their project ideas and helps the team formation process. By the end of the week, students form teams and submit their team name to the course instructor for approval.

Concept Document: In this document the teams provide a short description of the game they are planning, including details such as the type of narrative that will drive the game and aspects of the game that will be interesting or unique. This deadline ensures that the teams are meeting together to solidify the goals, and is the first opportunity for the head instructor and producers to give feedback on the scope and scale of the game being developed.

Design Document: Within the games industry, the design document is often considered the "bible" of the project, documenting why and how most of the project will be accomplished. For this course, the design document fulfills two roles. First, it serves to provide the complete environment of the game, adding information such as the setting, the main characters, and the obstacles faced during the game. The second and more important role is to document how the team plans to complete the course project. Teams are expected to outline their own milestones and to form budget.

The game development budget concept not only helps control the scope of team projects, it also encourages teams to analyze their game requirements in detail and helps to capture the limited resources and design restrictions found in real-world development. To give teams a sense of freedom in development, they are also allowed to construct a "wish list" of extra items beyond their core budget. Custom music and artwork can greatly enhance a game, but only when the story and other plot elements have been completed.

Game Prototype Walkthrough: At this point in the course, teams are expected to have a skeletal version of their entire game constructed. All play areas should be laid out, although not necessarily fully decorated and refined. Similarly, all major characters and encounters should be in place, although all conversational elements and scripting need not be complete. The aim is to make sure that progress is being made and that there is enough functionality so that producers and the head instructor can see how the game would progress. This milestone originally only required a storyboard, but during the first offering of the course we found that it was too easy for students to show a storyboard without completing any technical work.

Design Issue Presentations: To help groups think critically about the decisions faced in designing their games we devoted one or two lectures to team presentations that describe a "design challenge" faced by the team. This presentation is followed by a discussion with the rest of the class. Aside from the basic pedagogical value in students practicing presentations, this exercise encourages them to think about game design as a problem to be solved as well as a means for creative expression. Many teams do not consider what would happen if a player plays their game in an unexpected way, for instance, killing a key character in the plot. They may view such possibilities as annoyances instead of opportunities for creative design. Thus, these presentations provide a unique opportunity to explore design possibilities.

The Pitch: By this point each team's module is expected to be finished and ready to demonstrate. Teams give ten-minute presentations during class in which they attempt to "sell" their game. In addition to introducing the background, main character(s), and plot, they are required to show screenshots or recorded video demonstrations of the game. Most teams chose to create short game trailers, similar to what is done in industry.

Check-off and Peer Evaluation: When two weeks remain in the course, students must play and evaluate other groups' modules. Groups then have an opportunity to fix any outstanding issues in their own modules. After this period, each team must provide a document that summarizes their game, pointing out any interesting aspects including sophisticated scripts, custom artwork, music, or other features that may not be immediately apparent to the evaluator.

4.4 Project Evaluation

It is very difficult to evaluate open-ended projects. Some teams have members with time to put extra effort into a project, while other teams have more constrained schedules. Thus, we cannot mark projects simply by comparing the resulting games among groups in the class. Additionally, in our first offering of the course some teams created games that were too large to play through quickly without additional guidance.

There are three criteria by which we have been able to effectively evaluate student projects. The first criterion is whether they followed the stated guidelines when creating their games. For instance, is the module playable in 10-15 minutes, given insider knowledge of how to play the module. We have also asked students to create modules that are non-linear, so that the outcome of the game is affected by the choices you make. It is not difficult to evaluate whether these elements are found within a game.

The second criterion is through evaluations of other teams' games. Each student is required to play all other games and turn in an evaluation sheet form. These evaluations do not directly determine the marks for other groups but, they give us a wide range of experience with each game.

The last means we have used to evaluate a course project is to do a walk-through of each module with the team. This allows the team to point out all the different things that they did in the game. Some aspects of the game, such as custom artwork, can be missed, so this ensures that each team can show the creativity they put into their module.

When all of this information is put together, we can establish a mark for each team. We then take an additional step to compensate for the work done by various team members. Each team member must fill out a team evaluation form which asks how the workload was balanced between team members. We use this to determine a "multiplier" for each student within a team. Students that put in extra work receive a bonus, while if other team members consistently indicate that a team member did not contribute their fair share, this team member does not receive the full mark awarded the rest of the team. Students are aware of this marking scheme from the beginning of the semester, but producers also serve to keep the instructor informed about any problems that may crop up, so the instructor can step in, if necessary, to help mitigate any problems. This has worked well, giving us enough information to ascertain the balance of work within a team.

5. INDUSTRY PARTNERSHIPS

The one aspect of this course which cannot be easily duplicated by other universities is the contributions from BioWare Corp. In addition to making software donations for the course, BioWare employees have given lectures and have given feedback on student projects.

BioWare has given lectures on level design, on what makes games fun, and question-and-answer sessions. While this level of participation can only be gained from having good relationships with a company like BioWare, many of the things they discussed in lecture could be gleaned from gaming resources such as the Game Developer Conference, the Gamasutra web site, or Game Developer magazine.

The advice that BioWare has given on students' projects is difficult to replicate, as professionals in the industry have many unique experiences in game development. But, if a strong course is being offered, even remote game studios might be persuaded to help participate in a course. BioWare has reaped one reward from their relationship with our course: four of our former students have now been hired to work at BioWare.

6. COURSE EVALUATION

It is difficult to make a quantifiable evaluation of the effectiveness of a course; asking students their opinion of a course may not tell you whether you have achieved your pedagogical goals. We can look at course reviews to glean at least some information about the effectiveness of a course.

The University of Alberta conducts reviews of all courses at the end of each semester. A summary of the course evaluation can be found in Table 1. There are 15 metrics by which a course is evaluated; we present the average result. For each metric, we also reported how the course compares to other courses on campus.

Table 1: Course Evaluation

Year	Average Evaluation (Out of 5.0)	In Top 25% of all UofA Courses
Fall 2005	3.87	2 of 7[3]
Winter 2006	4.38	6 of 8[3]
Fall 2006	4.47	11 of 15
Winter 2007	4.87	15 of 15

Although we emphasized to students during the first semester that they were taking an experimental offering of the course, they were very upset about several things that did not go smoothly, particularly a lab exam that was too difficult. There was also a project team which fell apart near the end of the course, resulting in a fairly low course evaluation. We were able to fix most of these issues during following offerings of the course.

In addition to formal course evaluations, we have also conducted our own evaluations of the lectures. This is where we have been able to see trends in more detail. Students assess all lectures according to their level of interest in the material, the quality of the delivery, and the amount of knowledge that they felt they learned. For fairly technical topics like artificial intelligence students generally indicated that they learned a lot, but they also had lower comprehension of the material. Another set of lectures which have a strong CS bent, a historical overview of the hardware used to produce games, is one of the highest ranked lectures. The humanities lectures were also highly ranked as they have content which is relevant and interesting to all students in the course.

We used this feedback to refine our lectures; usually by increasing or decreasing detail, but also by eliminating some lecture topics. We do not claim to have solved the lecture content problem, only that a proactive and iterative approach will allow us to help reduce its impact.

7. DEMOGRAPHICS

The experience of teaching any course is contingent on the students enrolled in that course. To provide context on the students enrolled in our course and their backgrounds we provide some statistics about the students and the work they did in the course.

The majority of students taking the course are in their second-year. The course has no prerequisites. Enrollment is limited through an application process so we can balance students' backgrounds. Students have to fill out a short form detailing why they are interested in taking the course as well as their major and other background experience.

Table 2 shows the demographics of the students taking the course. We have generally, but not always, been able to maintain a good balance of CS to non-CS students. This has partially been due to advertising. Because CMPUT 250 was not an official course until fall of 2007, it was difficult to keep students informed about the course. When we switched the head instructor after the first year, there was a gap in our efforts to advertise the course.

One area where we can improve is in boosting the female enrollment in the course. From informal discussions with women eligible to take the course, they often perceive that they would need to be programmers to take the course. While this misconception may clear as more students take the course, we need to modify our advertising to better target female students.

[3]There were not enough evaluations for the University to provide us with statistically significant ranks for all metrics.

Table 2: Course Demographics

Year	Male	Female	CS	Science	Arts
Fall 2005	15	6	6	0	15
Winter 2006	20	4	7	9	8
Fall 2006	14	0	8	4	2
Winter 2007	16	1	11	3	3
Fall 2007	19	6	13	3	9

8. CONCLUSIONS

CMPUT 250 has been a success for the CS department. First, we received positive feedback from most students in the course. Although part of the success is due to the students' passion for games, a surprising number of students felt the best part of the course was the multidisciplinary teamwork. Second, by introducing a multidisciplinary course and engaging the humanities, it has helped to build important relationships between different departments. Third, in times of low computing science enrollments, courses like this act as attractors to students. Finally, the course has strengthened our ties with industry.

The team-based focus provides students with a unique learning opportunity not usually seen at the university. The cooperative, multidisciplinary learning environment brought together students with disparate skill sets which resulted in some remarkably creative efforts.

One challenge which we anticipate over the next few years is what to do when NWN is no longer an appropriate platform for the course. We have a good deal of expertise invested in this game, and so it will take some time to transition the course over to a different platform.

While we continue to work to enhance the learning experience for students taking this course, the course has met our design objectives. It provides an excellent learning environment not only for those who will continue on to work within the games industry, but for all students who will be involved in interdisciplinary or collaborative projects during or after they complete their university education.

9. ACKNOWLEDGMENTS

This work is supported by grants from NSERC, iCORE, and BioWare Corp. We gratefully acknowledge the help of: Loren Andrucko, Kevin Barrett, and Mark Brockington (BioWare); our excellent teaching assistants Matt Bouchard and Hector Perez; and the CS Instructional Support Group.

10. ADDITIONAL AUTHORS

Finnegan Southey, Matthew Bouchard, and Ghassan Zabaneh.

11. REFERENCES

[1] A. Berger. "Neverwinter Nights" in the classroom. *University of Minnesota News*, 2006.

[2] M. Carbonaro and et. al. Interactive story writing in the classroom: Using computer games. *Proceedings of DiGRA*, pages 323–338, 2005.

[3] M. McNaughton and et. al. Scriptease: Generative design patterns for computer role-playing games. *19th Intl Conf. on Automated Soft. Eng.*, pages 88–99, 2004.

[4] J. Robertson and J. Good. Story creation in virtual game worlds. *Commun. ACM*, 48(1):61–65, 2005.

[5] J. Schell. Shaping an entertaining future at Carnegie Mellon. *Computer*, 36(8):96–98, 2003.