The ETH Game Programming Laboratory: A Capstone for Computer Science and Visual Computing

Robert W. Sumner
ETH Zurich

Nils Thuerey
ETH Zurich

Markus Gross
ETH Zurich

ABSTRACT

The Visual Computing bachelors/masters program at ETH Zurich provides an internationally renowned degree in computer science with a specialization track in computer graphics. A new project-based game development course serves as a capstone to the program by reinforcing core computer science concepts and specialized topics in Visual Computing. Additionally, students learn design principles and obtain a better understanding of the interplay between the desires of game design and the realities of technical implementation. Finally, students practice crucial "soft skills" such as team work, effective communication, time management, and leadership. This article details the course goals and structure, presents three case studies of student-made games and the effect of the class on the students, and evaluates the overall class design. We hope that this document presents a compelling argument in favor of game development as a capstone to computer science and also provides useful insights for other academics wishing to incorporate game development into the computer science curriculum.

1. INTRODUCTION

The Visual Computing bachelors/masters program at ETH Zurich, Switzerland's Federal Institute of Technology, provides an internationally renowned degree in computer science with a specialization in computer graphics. We have recently designed the ETH Game Programming Laboratory (ETHGPL) to serve as a capstone for this program. Students work in small teams for the entire semester to design and implement their own video game. The ETHGPL provides a forum in which the cumulative knowledge from the entire program can be applied in a project-oriented setting. In doing so, the class focuses on transfer, the ultimate goal of learning where learned concepts are transferred to novel problems. Additionally, the ETHGPL provides a venue for students to learn design concepts, a rare opportunity at a technical institute. Finally, we use the ETHGPL to instill "soft skills," such as team work, effective communication, time management, and leadership, that are difficult to incorporate elsewhere in the technical curriculum, yet crucial as the students embark on the next phase of their careers.

The ETHGPL has far-reaching benefits. From the students' perspective, it offers a unique opportunity to apply the technical knowledge they have acquired over the course of their studies to the exciting problem of game creation. This experience is of great benefit for nearly every career path, since large-scale collaborative projects are universally valued. The course also provides an important service to ETH. A world-wide trend has seen undergraduate enrollment in computer science drop dramatically in the past few years [7]. The ETHGPL culminates in a public presentation of the student-created games that is widely advertised and highly visible. Due to the huge appeal of video games to today's youth, this presentation provides an ideal opportunity to attract new students to computer science. Finally, the ETHGPL, together with other game courses taught at universities across the world, has the potential to make a profound impact on society by fostering a new "science of games" in which the popularity of gaming is redirected toward serious topics such as education, training, health, public policy, and strategic communications [8]. The first step in this new science is providing education programs that not only teach the technical aspects of computer science but also foster the creativity that engenders novel game designs, game genres, and next-generation technologies.

This article discusses our experience creating the ETHGPL and teaching it for the first time. We summarize the Visual Computing bachelors/masters program and detail the goals and design of the new capstone course. To ground the discussion in concrete examples, we present three case studies of student-made games and describe the effect of the game class on the students. Finally, we evaluate the overall class design and many of our general preconceptions about an academic game course. We hope that this document presents a compelling argument in favor of game development as a capstone to computer science and also provides useful insights for other academics wishing to incorporate game development into the CS curriculum.

2. VISUAL COMPUTING

The computer science program at ETH is a highly technical- and engineering-oriented educational track comprised of a 3 year bachelors followed by a 1.5 year masters. The first two years of the bachelors education cover the foundations of computer science including discrete and continuous mathematics, theory, logic, programming, algorithms, and other topics. Students begin to specialize during the third year, and formally choose a specialization track when they enter the masters program. The possible options are visual com-

puting, theory, computational science, computer systems, distributed systems, information systems, and security.

The specialization track in Visual Computing teaches students in-depth knowledge of computer graphics, computer vision, visual information processing, and machine learning. Each student, mentored by a faculty member, composes her or his own individualized curriculum from a broad compendium of specialized courses. Many of our students pick the overall field of computer graphics and image generation and select specific courses such as computer graphics, geometric modeling, physically-based simulation, scientific visualization, computer vision, and machine learning. Toward the end of the masters program, such students have a strong technical profile in visual computing combined with a solid foundation of object-oriented programming, software engineering, theory, computational algorithms, and computer systems. However, collaborative work on a major project allowing students to apply a broad spectrum of knowledge and expertise has not been offered as part of the curriculum. The master thesis is a highly specialized, individual project and does not serve this purpose either.

This conspicuous lack of a central educational component created the need to design a novel and attractive class format for collaborative project work. To this end, and motivated by many of our students, we developed the ETHGPL as a capstone to the Visual Computing program.

3. ETH GAME PROGRAMMING LAB

In this section, we describe the goals of the ETH Game Programming Laboratory and the course design that we developed to meet these goals.

3.1 Course Goals

The highest form of learning is transfer: the flexible adaptation of learned concepts to new situations [2]. The primary goal of the ETHGPL is to provide a capstone to the computer science curriculum and Visual Computing specialization in which the cumulative knowledge obtained during the entire program can be transferred to the task of creating a video game. The course should serve as a venue for the reinforcement of both core computer science concepts such as software engineering, object-oriented programming, data structures, and algorithms as well specialized topics in Visual Computing, including imaging, surface representations, geometric modeling, physically-based simulation, texture mapping, and many others.

We chose game creation as the capstone course because the topic ties in closely with the overall theme of Visual Computing while allowing us to employ several learning paradigms that have been identified as especially effective by studies in learning science [5]. In particular, game creation provides an ideal setting for:

- **Experiential learning.** The first-hand experience of building a game ensures that the topics involved are internalized.

- **Inquiry-based learning.** Programming a game involves a great deal of experimentation in the design of shaders, levels, character behavior, physical simulations, lighting effects, sound, among many others. This inquiry-based exploration in which the student constantly wonders what happens when a change is made leads to deeper understanding.

- **Team learning.** Students work together in teams to develop their games. Studies on cooperative learning [4] indicate that teamwork leads to enhanced learning.

- **Goal setting.** The planning phase of game creation requires a detailed time line with many intermediate goals. Setting these goals in the first place as well as replanning in case goals are not met requires careful reasoning and leads to increased understanding.

Additionally, the ETHGPL allows us to impart several important skills that are neither addressed in other courses nor easily incorporated into a traditional lecture. ETH is a technical institute and, consequentially, the student experience is somewhat one-sided. The game class provides an opportunity to make this experience more well-rounded since game design is equally as important as technical execution. Students participate fully in both aspects. Thus, they learn important design principles and obtain a better understanding of the interplay between the desires of game design and the realities of technical implementation. Furthermore, students practice team work, time management, effective communication, leadership, giving and accepting criticism, and many other "soft skills" that are essential for success in the next phase of their lives [9].

3.2 Course Design

We designed the ETHGPL to meet the goals outlined in the previous section. The class is primarily project based. Students work together in teams of two or three to design and implement a new game during the 14 week semester. Lectures focus on aspects of game development that students have never learned, such as style and design principles, game prototyping, and understanding fun, and follow two prominent textbooks on game design and development [3, 6].

3.2.1 XNA & Xbox 360

One significant design decision we made was to use Microsoft XNA [10] with games deployed on the Xbox 360 console. Although this choice restricts development to C# with the XNA libraries and Microsoft's development tools, it greatly advances our goals. The first justification for this choice is very practical. XNA removes many of the low-level hurdles that would otherwise overburden the game development process. As a consequence, students concentrate their time on higher-level implementation tasks that better reinforce the core topics of the Visual Computing major.

The second reason for selecting XNA and the Xbox 360 deals with the students' perception of the class. The time commitment for this course is significant and it is crucial that students are motivated to work very hard toward the realization of their game. We believe that developing for a console increases motivation by lending a more professional feel to the class and providing a tangible connection to commercial game development, which is largely dominated by console games. The course gives game players a chance to switch roles and become a game producer by creating their own console game; the experience is not as authentic if development is restricted to the PC. The extra excitement engendered by the real-world hardware increases the time on task, makes the game development more successful, and enhances learning. Although we have no preference for one console over another, the Xbox 360 is currently the only practical choice since no high-level development tools are available to educational institutions for any other model.

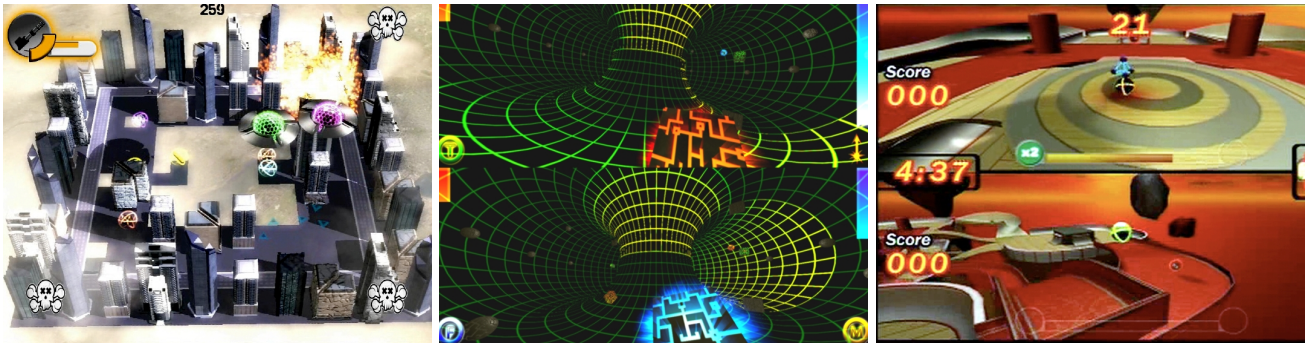

Figure 1: Screen shots of the three game projects discussed as case studies. To the left, the game "Battle Balls" is shown, in the middle "Titor's Equilibrium," and to the right "SPHERES."

3.2.2 *Project structure*

In order to leverage the past experience of others, we used the project structure from the game course offered by Tiffany Barnes at UNC Charlotte [1] as a starting point for our own project structure, since Prof. Barnes's course has already benefited from several years of successful instruction. Our project structure consists of a design and planning phase and a development phase.

Design and planning phase.

When the class began, students immediately started working on the design and planning phase of the course project. Students were given roughly four weeks to develop a formal game proposal detailing their game design and development schedule, along with mock-ups or prototypes to convey their game idea. During this phase, we gave lectures targeting game design concepts and encouraged novel game ideas and non-traditional genres. In this phase, we immediately addressed several of our goals. Each team created a detailed development schedule that reinforced software engineering skills, one of the most important aspects of computer science. Students were required to incorporate some advanced graphics topic such as physical simulation or non-photorealistic rendering into their game, which ensured that ideas from the Visual Computing specialization track would be reinforced. Finally, students took a brief hiatus from their technical education to focus on creativity and design. Although many game development courses center around interdisciplinary groups of pure engineers and pure designers, we feel there is also great benefit in allowing the engineers to test their design skills in order to gain an understanding and appreciation of the more artistic side of game development.

Development phase.

The development phase consumed the remaining 10 weeks of the semester. Progress was tracked with an interim report due at week 9, an "alpha release" due at week 12, a playtesting report due at week 13, and a public presentation and written report due at week 14. In this phase, students utilized concepts learned throughout their computer science education in order to develop their game. Additionally, their software engineering skills were challenged as they tried to follow the development schedule created in the design and planning phase. Team organization, cooperation, communication, presentation, and other aspects of the cooperative development effort enforced soft skills necessary for success in most real-world situations.

4. CASE STUDIES

We discuss three exemplary ETHGPL projects as case studies to illustrate the lecture's capstone nature (Figure 1). Each game was awarded a prize at the public showcase at the end of the course by the audience and a jury of experts. Videos from all games can be found in the teaching section of the ETH Computer Graphics Laboratory's web site (`http://graphics.ethz.ch`).

One of the highlights of the lecture is the multi-player party game "Battle Balls." The game's setting is a fight of alien vehicles in the blocks of a giant city. The matches take place on small hovering city platforms. The last player to stay on the platform wins. The game has a polished and complete look and feel, well balanced power-ups, an intuitive game-play, and a variety of different levels. On the technology side, it features collision detection, physical simulation for crumbling buildings, and advanced shaders and particle systems for smoke, fire, and glowing effects. This team was the only one to include a team member, Matthias Bühlmann, who is both an artist and a computer scientist. The advantage is obvious from the game's attractive visual design.

The gameplay in "Titor's Equilibrium" is based on two player battles with physical interactions of rigid bodies using, among others, forces to push around smaller objects in the level, activate defensive shields, or even slow down and reverse time. The fights are based on the classic "Rock, Paper, Scissors" game, with each player choosing a "host object" that bestows powers having both advantages and weaknesses depending on the other player's choice. Although this game is harder to learn than the others, it exhibits the most novel design and gameplay. In terms of technical contributions, it features a full rigid-body simulation engine. The students were able to transfer the knowledge gained in the physically-based animation lecture and model their gameplay around the capabilities of the rigid body solver. Furthermore, this group remarked in their final report on the success of the "soft skills" of communication and knowledge transfer within their team, with other teams, and with the course organizers. Indeed, Gioacchino Noris from the "Titor's Equilibrium" team helped create a ETHGPL Wiki page with collaborative tips and tricks from all teams. Previous lectures, he says, did not require a comparable amount of team work. He also confirms that the lecture provides practical use of the algorithms learned in other classes. According to him, it is hard to really understand complex algorithms unless they are implemented and used.

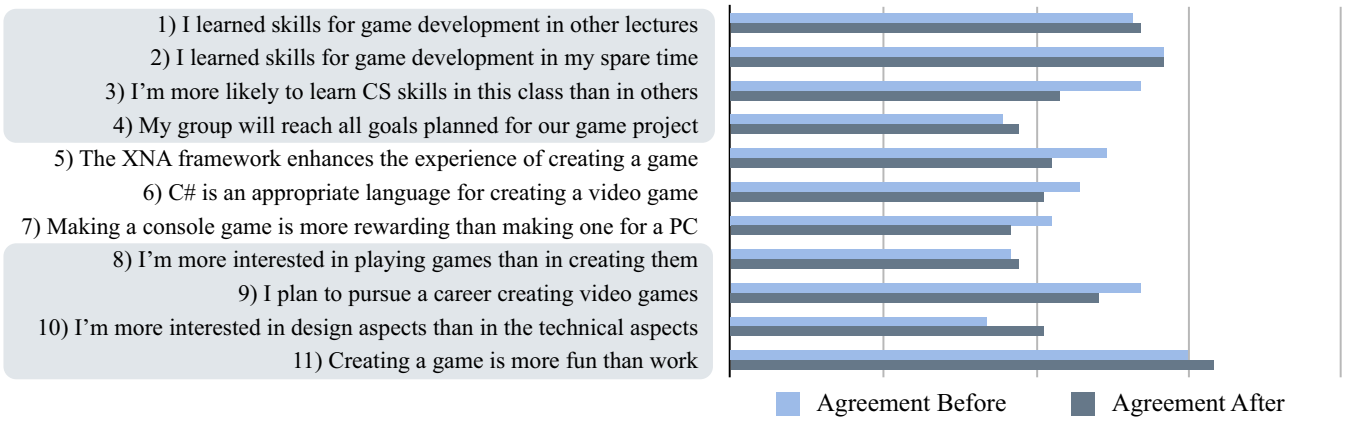

Figure 2: Our survey to evaluate the design of the ETHGPL. Light blue indicates the averaged students' view at the beginning of the course, while dark blue shows the students' agreement at the end of the course.

"SPHERES" is a two-player capture-the-flag game in a roller coaster-like 3D arena. Gameplay is fast-paced and competitive as players take advantage of short-cuts and power-ups that increase the game's strategical variety. The team members implemented a game engine for rendering the complex level complete with soft shadows, an impulse-driven physics model, and collision detection via a BSP tree. Each of these components was taught elsewhere within the Visual Computing program. In terms of software engineering, Thomas Oskam states that their group was forced to organize development by assigning tasks according to each team member's knowledge. This work distribution resulted in the design of clear interfaces and areas of responsibility within the collaborative development. The "SPHERES" group also praised the console option, stating in their final report, "Absolutely the most exciting thing about this course was the possibility to create a game for the Xbox."

Additionally, several students told us that it was excellent to be able to rely on the console's controller. With its analog sticks and buttons it offered interesting possibilities for player interactions that are not readily available on a standard PC. Students also commented that frequent meetings and intermediate presentations helped to motivate them by seeing how the projects of the other groups advanced over time and which new effects another group was able to include since the last version.

5. EVALUATION

In this section, we discuss the results of our lecture survey, and the students' feedback from a debriefing session.

5.1 Course Design Survey

In order to evaluate and validate our course design, we performed an anonymous survey in which the 19 students were were asked to anonymously indicate their agreement with 16 separate statements on a scale from one to five. This was done at the beginning of the lecture, and once more after the students had officially completed their game projects. Although space limitations prevent us from discussing all statements, 11 of them are included in Figure 2, together with the averaged responses. The 11 statements can be separated into three groups: the first targets learning aspects, the second one deals with the console approach using XNA, and the third group evaluates the role of fun and personal involvement. The remaining five statements which are omitted here deal with gender issues and the interplay between games and social interaction.

The first two statements, which deal with learned skills for game development, indicate that the students feel well prepared for their game projects by the lectures they have taken and their spare time activities. There is, however, a slight tendency of the students to learn the necessary skills for game development outside of the normal lectures. Both statements are consistent before and after the lecture. On the other hand, the question of whether computer science skills are better learned in a gaming project than in a normal lecture has a significant gap between the pre- and post-lecture evaluation. On average, the students are less confident in learning these skills in the game laboratory. This result fits with our overall design of the game lab as a forum for reinforcing computer science concepts learned throughout the bachelors/masters program.

The second group of statements, dealing with the XNA framework, C#, and the console deployment, interestingly shows a consistent trend toward a more negative evaluation at the end of the lecture. According to discussions with the students, several sources of difficulties contribute to this disparity. First of all, many students realized that console development has a high "coolness" factor, but also imposes stricter limitations than PC development. The students, thus, might have originally had an idealized view of the console development process. Another factor was the status of the XNA framework. During the course of the semester, several issues with file formats, performance, lack of documentation, and the absence of introductory books appeared. Many of these issues are likely growing pains that will be resolved with future XNA releases. Moreover, the C# compiler caused unexpected slowdowns for some numerically intensive procedures. This speed problem probably results from limited optimization capabilities compared to current C and C++ compilers. Considering the active development of both C# and XNA, we hope most of these issues will have been alleviated when the ETHGPL is taught again in 2008. Overall, the students agreed that the quality of the game projects would not have been possible without the underlying features of the XNA framework.

The third block of statements targets the fun aspect and personal involvement of the students with game develop-

ment. While they, in general, show interest both in playing and developing games, some students seem to be less eager to enter the game industry after the lecture. There is, however, still a notably positive interest in game development as indicated by the post lecture survey. Additionally, more students are interested in the technical aspects of computer games after the course. And, luckily, there is even a more positive trend after the lecture according to the statement that computer game development is more fun than work. This result is especially good to hear, considering the large amount of work most of the teams invested during the course of their project.

In a second part of our course evaluation we asked the students which skills they considered to be most important for game creation. They could choose, among others, from topics such as linear algebra, programming, hardware knowledge, and sound creation. The three most important skills according to the students are 1) programming, 2) computer graphics and 3) three-dimensional modeling. Although this order did not change from the pre- to the post-evaluation, the "programming" skill votes increased by more than 50% after the lecture. This change indicates that the students underestimated the programming skills necessary to develop a complete game using a sophisticated new programming framework such as XNA.

5.2 Debriefing

During the final course lecture, we conducted a "debriefing" session in which we asked the students about their experience taking the game class, the biggest challenges and technical difficulties they encountered, and any suggestions for improvement. Although space limitations prevent a detailed discussion of all feedback, we highlight two prominent comments here as an aide to others developing similar courses.

Undoubtedly, software engineering presented the most significant challenge. For all students, this project was the largest they had ever attempted and, for many, it was their first significant team effort. With the project structure (Section 3.2.2), we enforced basic software engineering practices. Students admitted that this requisite planning was annoying at first, but payed off in the end. Many even suggested placing more emphasis on good software engineering as some teams "wasted" time implementing features that never made it into the game. All teams noted that they drastically underestimated the time and effort required for individual game components, with some taking as much as four times longer than anticipated. In response to this feedback, in future instances of this course we will help students be more realistic about what can be accomplished and monitor their progress more closely.

From a technical standpoint, all groups highlighted the XNA framework as the most significant source of difficulty. While students noted that its overall structure is intuitive, the technical details present many problems. There are dozens of plausible ways to approach every individual aspect of game creation, but only a few are viable solutions given other software and hardware dependencies. Finding the appropriate solution for each task was quite time consuming for the students, with hours spent searching the web and reading forum posts. Students requested more examples of large-scale, working games as well as in-depth technical tutorials about XNA. An obvious conclusion drawn from this feedback is the necessity of an assistant with a deep and solid understanding of the technical details of XNA. When the course is taught again, we plan to hire students from the previous instantiation as expert helpers to provide the needed technical assistance.

6. CONCLUSION

Our primary goal with the ETH Game Programming Laboratory was pedagogical: reinforce core computer science concepts as well as specialized topics in Visual Computing, incorporate design principles into an otherwise all-technical major, and teach "soft skills" that are crucial yet often overlooked. In this article, we have presented an argument for game design as the ideal venue for such an effort. We detailed our course design, described our experience teaching it for the first time, and evaluated the results. We feel the course was rewarding both for the students who took it as well as for the course organizers. The public presentation of the student-created games filled one of our largest auditoriums and received very positive feedback. We hope the ETHGPL public presentation will become an ETH institution and help to improve the image of computer science, increasing enrollment and leading to more highly qualified computer science graduates. Perhaps the most convincing evidence of the game class's success comes from a comment written by one group in their final report: "This course really gave us one of the best experiences in our academic careers. What could be more fun than learning through games?"

7. ACKNOWLEDGEMENTS

We wish to thank Simon Heinzle for his help as the course assistant, and we are grateful to Marc-Alain Steinemann from Microsoft Switzerland, Lars Lipper, Thomas Stowasser, and Dirk Primbs from Microsoft Germany, and Dave Mitchell from Microsoft in Seattle for sponsoring some Xboxes and providing XNA Creators Club accounts for the class.

8. REFERENCES

[1] T. Barnes. Computer game design and development. UNC Charlotte Course ITCS 4230/5230, Fall 2006. http://www.cs.uncc.edu/~tbarnes2/GameDesign/.

[2] J. D. Barnsford, A. L. Brown, and R. R. Cocking, editors. *How people learn: Brain, mind, experience, and school*. National Academy Press, 2000.

[3] T. Fullerton, C. Swain, and S. Hoffman. *Game Design Workshop: Designing, Prototyping, and Playtesting Games*. CMP Books, 2004.

[4] D. W. Johnson, G. Maruyama, R. Johnson, D. Nelson, and L. Skon. Effects of cooperative, competitive, and individualistic goal structures on achievement: A meta-analysis. *Psych. Bulletin*, 89(1):47–62, 1981.

[5] M. J. Mayo. Games for science and engineering education. *Commun. ACM*, 50(7):30–35, 2007.

[6] S. Rabin. *Introduction To Game Development*. Charles River Media, Inc., Rockland, MA, USA, 2005.

[7] J. Vegso. Continued drop in CS bachelor's degree production and enrollments as the number of new majors stabilizes. *Comp. Research News*, 19(2), 2007.

[8] M. Zyda. Creating a science of games. *Commun. ACM*, 50(7):26–29, 2007.

[9] Many graduates 'lack soft skills'. BBC News Article, January 2007. http://news.bbc.co.uk/go/pr/fr/-/1/hi/education/6311161.stm.

[10] XNA developer center. Website accessed 9 October 2007. http://msdn.microsoft.com/xna.