Bringing a pioneer games project to the next level

Rafael Bidarra
Delft University of Technology
Mekelweg 4
NL-2628 CD Delft
The Netherlands
r.bidarra@ewi.tudelft.nl

Jerke Boers Jeroen Dobbe Remco Huijser
Cannibal Game Studios
Rotterdamseweg 145
NL-2628 AL Delft
The Netherlands
{j.boers, j.dobbe, r.huijser}@cannibalgamestudios.com

ABSTRACT

Many universities with a computer science (CS) curriculum nowadays offer a game development course in a variety of flavors. However, it is not always clear what is the fundamental standpoint that leads their particular course design. Delft University of Technology introduced project-based education in its CS curriculum five years ago, including a second year *games project*. Initially designed as little more than a companion to the computer graphics course, the *games project* matured into a large project integrating a broad range of computer science topics. More importantly, though, the current games project brings CS students for the first time to work in a realistic and interdisciplinary game development team, involving students pursuing a Game Design and Development degree at the Utrecht School of the Arts. We believe that the key to the huge success of our games project lies in the consistent combination of this careful interdisciplinary organization with the deployment of professional technology and working environment specifically crafted for an educational environment. We also conclude that a streamlined collaboration among students of related disciplines works as a very powerful catalyst in their personal and academic development.

GDCSE'08, February 28–March 3, 2008, Miami, FL, USA.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education – *Collaborative learning*
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education, Curriculum*
K.8.0 [**Personal Computing**]: General – *Games*

General Terms

Design, Experimentation, Human Factors.

Keywords

Games education, interdisciplinary education, game development

1. INTRODUCTION

Five years ago, Delft University of Technology introduced project-based education in the computer science (CS) curriculum. One of the new project-based courses was the second year *games project* [2]. This games project was initially associated with an introductory course on computer graphics (CG) and as such the primary goal was to have students apply computer graphics techniques in a practical setting.

While first running the games project, it soon became apparent that it had more potential and a bigger scope than was initially envisioned. Not only was CG covered, it naturally required students to learn more about other game related aspects as well.

Because of the potential of the project and the enthusiastic reaction from the students, the project has been actively improved over the years. Combined with valuable industry input, the games project has now matured into a multidisciplinary course covering all aspects of game development, and therefore better reflecting real-life game development environments.

In this paper we describe and motivate the evolution over the past five years, from a pioneer to a professional games project, and we do this from both an academic and an industry perspective. We start by summarizing the project organization (Section 2), followed by a discussion about the working environment provided and the technology deployed (Section 3). Throughout the years, more and more external partners got involved in the games project (Section 4). We conclude with an evaluation of the project run this year, 2007 (Section 5), and with some general conclusions about running such a games project (Section 6).

2. PROJECT ORGANIZATION

Project-based education very much responds to the basic concepts behind *constructive alignment* [3], a rather influential stream, in particular in higher education, which advocates among other things that 'students construct meaning from what they do to learn'. In line with this, an advantage of including such projects in a curriculum is that the acquisition of knowledge is strongly motivated by its immediate application in a practical environment. In addition, it encourages students to actively learn to value and promote the teamwork process, instead of focusing exclusively on the final product.

Characteristic of CS project courses is that students have to work in groups on a more or less open assignment [11]. In our case, they design and implement a computer game from scratch, using the technology provided and working in a team (see Section 3). This section describes how we organized the project as a whole.

2.1 Course goals

At first, the games project had a focus on teaching students to apply Computer Graphics techniques in practice. However, as game development involves more than just CG, we wanted the students to be able to focus not only on computer graphics, but also on software engineering, artificial intelligence, modeling, user interaction and other areas involved in game development.

As we strived to continuously improve the project and better prepare students for work after their study, we also wanted to make sure that students learned how to work within the context of a realistic software project, while, at the same, learning how to cope with the challenges of interdisciplinary collaboration. Considering this over the past few years, we gradually expanded the course goals to comprise a wider range of games-related issues, eventually leading to the current set of learning objectives. We say they have been achieved when the student has demonstrated proficiency in:

1. applying media and programming techniques within the context of computer games, and in relating them to particular game effects;
2. striving for the balance between the effectiveness of a programming technique and the desired quality of a game effect;
3. describing the main modules of a game engine and purposefully use their functionality;
4. deepening object-oriented programming skills while building a complex and large software system in an agile context;
5. developing and contrasting teamwork skills within the context of a realistic interdisciplinary team.

2.2 Teams

For several years, this project has been run in groups of about 5-7 CS sophomores, who had to handle alone *both* game design *and* implementation. The former not being part of their educational curriculum or goals, distracted from their work as programmers and did not allow them to really focus their efforts on the course goals stated above.

Therefore, in the Spring of 2007 the project entered a new phase: we started a pilot collaboration with the Utrecht School of the Arts (HKU), which offers a bachelor degree on Game Design and Development. Their second year students also have a one-semester project, focusing exactly on the game design process as a whole. Integrating their game design project with our game development project led to one large multidisciplinary project. In this integrated project, groups consisted of 4 CS students and 5 game design (GD) students. The CS students were mainly responsible for the implementation of the game, while the HKU students were in charge of game design and artwork/content creation; in doing this, they worked as two departments of 'one single company', with lead programmer and lead designer roles, respectively, assigned among them.

Integrating these two projects brought much more realism and power to the project: realism, because it more closely matches the actual team composition in real-world game developers; power, because this interdisciplinary collaboration promotes that each team member contributes with his/her best skills to the project. In other words, we fully confirmed the value of the splendid advice recently given by Randy Pausch: "(…) not to turn artists into engineers or vice versa, but to teach students how to work in teams that utilize the disparate talents of their members" [9].

These mixed groups, though having clear advantages over traditional uniform groups, also had some disadvantages; for example, more time was spent on communication, traveling and appointments. In particular, everyone in these groups vividly experienced the additional challenges brought about by communicating with people from outside your own discipline, which requires a rather different way of thinking and explaining.

Significantly, after this interdisciplinary pilot experiment, which although being facultative, was chosen by the vast majority of the students, all of them were unanimous to recommend that next year we make it obligatory to work in such mixed teams.

2.3 Project planning

In line with other project courses in the CS curriculum, the games project at first consisted of three phases: *analysis*, *design* and *implementation*, where the implementation phase was by far the largest and most complicated phase. However, students did not perceive the analysis and design phases as very useful, which can be explained by looking at game development projects in practice.

Key to designing and implementing a successful game is having an approach in which you strive to have a playable and working version of the game as soon as possible: the so called *first playable*. After this version is established different gameplay elements can be tried out and changes can be made to the initial

version. This usually occurs in multiple iterations leading to a more agile development process [6].

To better accommodate for this process and to make the project more interesting for students we decided to drop the classical waterfall style of development. We introduced new phases of a more iterative nature: *spikes*, *first playable*, *beta* and *release*.

At the beginning of the project, students have no experience with the technology and, as they have no prior experience developing games, no knowledge of what developing a game entails. To smoothly introduce students to actual game development and the technology involved, the spikes phase offers room to try out different concepts and technical solutions, gaining more insight into important aspects of their game. The first playable phase is aimed at gaining the first playable by integrating all relevant spike solutions into one product. Both the beta and release phase are aimed at refining the previous versions and completing the game.

For the interdisciplinary groups this turned out to be a very important methodology that enabled both parties to work on their game together. It not only provided them with the necessary development cycle that allowed them to continuously (re)design, develop, evaluate and discuss a "tangible" prototype. It also allowed the programmers to better cope with the frequently changing requirements that the creative process of game design and development brings forth.

2.4 Deliverables

To monitor the progress of the teams and to steer them along an effective development process, students had to hand-in three distinct deliverables at the end of each phase:

1. the implementation of the game (working source code);
2. a simple game design document (containing an explanation of the game and its key features);
3. a technical document (linking the explanation of the game to the implementation).

As is to be expected from an iterative approach, each of these deliverables started from a basic version and evolved into the final product. These deliverables and the team progress also served as a valuable basis for the final assessment (see Subsection 2.6).

2.5 Focus on requirements

To provide students with a clear direction and a tangible approach to fulfill the course, a list of requirements was set up. Where this list initially only contained some general requirements and computer graphics techniques, the list has been expanded to also include AI techniques and a number of other game-related requirements. Students had to make a selection from among these different requirements, as long as they incorporated all of the general requirements, two graphics and two AI techniques and implemented another technique/requirement of choice. These requirements ensured that students build a 3D game involving interesting technical challenges.

By offering a wide choice among many game-related techniques we guarantee that there are always challenging aspects for every student to explore. This, in turn, encourages students to remain motivated, to delve deeper into whatever study subjects required, and to exceed themselves in the implementation of the techniques of their choice.

2.6 Assessment

Several aspects are important when it comes to determining how to assess the students work. From the course goals it is apparent that we not only have to assess the final product, but also the process. This led to both a product mark and a process mark:

$$\textbf{final grade} = \frac{6 * \text{product mark} + 4 * \text{process mark}}{10}$$

The product mark takes into account, among other things, the quality of the game (various aspects of gameplay), the quality of the software (e.g. architecture, modularity, clarity, choice of technical solutions), technical realization of the different requirements and the quality of the project documentation. Placing a large emphasis on the technical realization supports the focus on the requirements.

The process mark takes into account the collaboration between team members (e.g. use of working environment, tasks, communication with GD colleagues) and the individual contribution of each group member in the whole development process (e.g. dedication, initiative, leadership, performance).

To assist the tutors in performing the assessment of individual contribution and collaboration, the students performed several peer-evaluations throughout the semester, in which they anonymously assess each of their group members. Our experience has steadily confirmed that this peer assessment provides very valuable, reliable and effective learning elements to each student [5], in addition to assisting the tutors in their coaching and assessment responsibilities.

Adding the collaboration component and the quality of the game into the equation, stimulated students to also focus on collaboration and get as much out of the group as they could. Including peer assessment assured that students would be motivated to cooperate with this collaborative process, thus avoiding negative peer-reviews.

3. WORKING ENVIRONMENT

An important part of any project is the choice of the supporting technology to work with. In our games project, this ended up requiring also support for collaboration within the teams.

3.1 Game technology

At first the project was supported by the open source graphics engine OGRE [8]. This engine is written in C++, which is considered the industry standard. But in the context of "inexperienced" students who have to develop a large and complex software product, C++ becomes a problem. Students become more focused on mastering the programming language than on the development of their game. Furthermore this engine was an open source graphics engine which, at that time, was hard to install and lacked quality support and the functionality of a complete game engine.

To overcome these difficulties, we formed a development group, and started to work on our own game engine, called *Cannibal*. A key aspect of this new engine was that it should favor usability

over raw performance. This should make it not only easier to use, but also more manageable, both leading to increased productivity and a better focus on developing the game.

This was also our motivation to choose C# [1] as the programming language. Although it is not an industry standard in game development, C# offers good performance and has proven to be very easy to learn and work with. Complementary to the choice of C#, the XNA Framework [12] was chosen as the underlying platform for the Cannibal Engine. XNA not only enables cross platform development using C# on both Windows and Xbox360, but it also comes with a convenient development environment: Game Studio Express.

Together, the Cannibal Engine, a managed language like C# and the XNA Framework take away a lot of the technical details so students can focus more on the design of the system and the project requirements. A better focus on design is especially convenient for game development projects because they tend to grow exponentially in complexity as they grow in size.

In 2006, the developers of Cannibal Engine started their own company, called Cannibal Game Studios, with the main goal of turning their technology and experience into a professional product. This product, called Cannibal Experience, is directly aimed at higher-education institutions, supporting them to use game development as a means to teach their curriculum. Cannibal Experience mainly consists of two components, a *Game Development Platform* and an *Online Collaboration Platform*, and it was designed to facilitate most technological aspects of running such a project, as well as to provide information on game development and game education, thus enabling teaching personnel to concentrate on the learning objectives.

3.2 Collaboration

In early editions of the games project, only a marginal working environment was provided to the students, leaving the collaboration up to them. In this way students experienced how hard collaboration is, but did not specifically learn how they could go about improving this situation.

As the project evolved into an interdisciplinary project which included more game development related aspects, collaboration within each team became even more important. To allow the students to learn the most about the actual collaboration process, an integrated working environment was provided in which they had access to a number of collaboration tools, the most important of which were a Wiki [4], a Subversion repository [10] and a bug/task tracking system. The Wiki system allowed for easy, fast and collaborative editing of documentation for the game and communication. Students were motivated to keep their Wiki up-to-date throughout the project. This not only improves the collaboration between team members but also makes it very easy to produce their deliverables, by simply extracting document data from the Wiki. The subversion system was used to share code and assets among the group and to record the changes and different versions of the game. The bug/task tracking system also supported planning and was used to keep track of the project progress. The team was encouraged to create a milestone for each phase of the project, and to fill them with tasks assigned to each team member.

The tools presented here did not only provide added value to the teams, but they are also very useful for the tutors. By supplying these tools to students, tutors can meticulously follow the development process of the teams. This provides them with valuable insight and overview of the course, and helps them decide when and where to focus their guidance.

4. INDUSTRY INVOLVEMENT

From the beginning of this project we have actively tried to involve a variety of partners related to game development. Involvement of real stakeholders from the games industry has been an important success factor for the project because it strongly stimulates and motivates students. Furthermore, these parties enrich the project with game development experience and technical expertise. For example, we always schedule a number of guest lectures in which experts from renowned Dutch game developers (e.g. Streamline Studios, Triumph Studios, W!Games and Playlogic), tell about their experiences with developing games, from a wide variety of viewpoints.

Another way of getting the industry involved has been to invite companies to sponsor the *Game of the Year* competition, an exciting contest 'unofficially organized' every year in our faculty among the participating teams: the basic idea is that the sponsoring company provides both a jury member and a prize for the winning team. This scheme not only gets the companies to promote their games, but above all it helps them get acquainted with the best skills of our best students.

In 2007 the project has been sponsored by Microsoft Netherlands. Because the Cannibal Engine is based on XNA, Microsoft Netherlands donated a number of Xbox360 consoles to the faculty for use in this project, giving a significant boost to the enthusiasm of all students. This year was also unique as the students were given the opportunity to present the games they had created at the Microsoft DevDays event, in Amsterdam.

5. PROJECT EVALUATION

This year, for the first time, *all* students of *all* participating groups successfully finished the project. As might be expected, the games developed by the six interdisciplinary groups were significantly more creative, consistent and appealing than the game of the single group working alone; however, all games, although considerably simple, were recognized to be a remarkable result for a one-semester design and development cycle. Please refer to the course website [7] for the description and sources of each of the games produced. See also some screenshots on the title page of this paper.

From the organization point of view, we very much profited from the accumulated experience, the biggest challenges having to do with the novel cooperation with the HKU colleagues, e.g. appointments, traveling time, language and culture clashes, etc. However, learning to cope with this diversity was precisely one of the main reasons for the initial choice, and the general consensus was that that had been very effectively achieved.

The working environment (see Section 3) was generally acclaimed as rather helpful and pleasant to most tasks. The Cannibal engine was, this year, considered as especially accessible, easy to use and attractive, among other things, due to the Xbox 360 compatibility. The assistance and supervision tasks were now more directed towards architecture and conceptual issues, rather than having to concentrate on technical programming problems. In addition, quite some extra time had to

be dedicated to the coordination of the interdisciplinary groups, in order to avoid or overcome conflicts at hand.

First, and most importantly, the five project goals mentioned in Subsection 2.1 were largely achieved. Indeed, most students acknowledged having attained a much deeper insight on a variety of subjects. When asked to indicate the three areas most improved upon, students mostly indicated media and programming techniques, ranging from mathematical foundations (55%) to computer graphics (64%) and AI (45%). Programming and software design proficiency were mentioned the most (90%). Although apparently most creative work had been left to the GD students, CS students quickly realized that they had plenty of room left to exercise their own creativity, getting the most out of the engine, e.g. programming many gameplay, physics and control elements of the game, and overcoming the limited experience of OO-programming at project start-up. Finally, all groups recognized that carefully watching over their teamwork process had made it possible to achieve their successful results.

In Table 1 we summarize several other results of the survey, highlighting some more concrete, interesting aspects of the project realization. The table indicates, for each statement, the percentage of students who subscribed to it. Not surprisingly, every year many students point out that they would have liked to spend even more time in order to "get their product really satisfying", an interesting conclusion that remarkably matches the reality of many game developer companies.

Table 1 – Summary of survey results

My dedication was (very) great	71%
We were given an interesting assignment	93%
I experienced the powerful capabilities of teamwork	92%
I am satisfied with the product delivered	63%
I learned more from this project than from any other in the curriculum	75%
The project was more fun than any other in the curriculum	100%

6. CONCLUSIONS

Five years after the introduction of project-based Computer Science education at Delft University of Technology, we can safely conclude that its highly instructive and motivating potential has been more than confirmed, so much so that various Departments and Faculties started following the same approach. Initiated as a pioneer project on computer graphics [2], the *games project*, as it is known on campus, has now gained a prominent role as the integrator course *par excellence* of the Computer Science BSc curriculum.

In its current form and organization, including the input from the game development industry as described in this paper, the project has achieved a substantial maturity, deploying a professional game engine, a fine-tuned working environment and very experienced tutoring assistance. It goes without saying that by now numerous former CS students of this project have graduated from Delft and either have found their career in one of the various Dutch game developer companies, or established their own start-up companies in the field, as is the case of most authors of this paper. Furthermore, the increasing reputation and popularity of the games project is being very effectively exploited by the Faculty for the urgent purposes of recruiting new CS students.

We believe that deploying adequate game technology, professionally crafted for this purpose within a carefully set up working environment, is crucial for the academic success of any integrated games project as the one described here. Finally, we can conclude that a streamlined collaboration among students of related disciplines is a powerful catalyst that can significantly raise the levels of knowledge, experience and teamwork skills achieved by the students.

7. ACKNOWLEDGMENTS

The authors are very grateful to all their (former) students for all their patient and invaluable feedback throughout the years, and to all colleagues who contributed to the success of this project with their constructive ideas and criticism. Special thanks go to Natasha Tatarchuk and Alpana Kaulgud, from ATI Inc., for generously equipping our CG Lab, and to Maarten-Jan Vermeulen, from Microsoft Netherlands, for his enthusiast and supportive involvement in our work.

8. REFERENCES

[1] Bates B (2004) C# as a first language: a comparison with C++. Journal of Computing Sciences in Colleges, 19 (3): 89-95

[2] Bidarra R, van Dalen R, van Zwieten J (2003) A Computer Graphics pioneer project on computer games. Proceedings of CGME 2003 - Workshop on Computer Graphics, Multimedia and Education, 8 October, Porto, Portugal, pp. 61-65

[3] Biggs J (1999) Teaching for Quality Learning at University. SRHE and Open University Press, Buckingham

[4] Leuf B, Cunningham W (2001) The Wiki Way. Quick Collaboration on the Web, Addison-Wesley, Boston

[5] Liu, N-F, Carless, D (2006) Peer feedback: the learning element of peer assessment. Teaching in Higher Education, 11(3): 279-290

[6] Martin RC (2003) Agile Software Development: Principles, Patterns and Practices. Prentice Hall, Upper Saddle River, NJ, USA

[7] MKT4 project website, Delft University of Technology. http://graphics.tudelft.nl/~mkt4/

[8] OGRE, http://www.ogre3d.org/

[9] Pausch R, Marinelli D, (2007) Carnegie Mellon's Entertainment Technology Center: combining the left and right brain. Communications of the ACM, 50 (7): 50-57

[10] Pilato M (2004) Version Control With Subversion. O'Reilly & Associates, Inc., Sebastopol, CA, USA

[11] Schaefer S, Warren J (2004) Teaching computer game design and construction. Computer-Aided Design 36 (2004): 1501–1510

[12] XNA, http://msdn.microsoft.com/xna/