From a Light CG Framework to a strong Cannibal Experience

Jerke Boers[2], Jeroen Dobbe[2], Remco Huijser[2] and Rafael Bidarra[1]

[1]Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, The Netherlands
[2]Cannibal Game Studios, Delft, The Netherlands

Abstract

Game development courses are being more and more deployed within computer graphics (CG) curricula. A fundamental element in the pedagogical effectiveness of such courses is the quality of the development framework provided to the students. We discuss the most important challenges faced throughout the years while using, configuring and improving the framework for our games project, and describe the solutions we came up with to resolve those issues. We conclude that a carefully designed development framework, including all underlying technology, course material and quality support, significantly determine the quality of a project-based game development course. In addition, when the teams in such projects have an interdisciplinary character, providing an effective collaboration environment is crucial for the success of team members. We believe that the key to the huge success of our games project lies, to a great extent, in the deployment of a professional working environment specifically crafted for an educational setting.

Categories and Subject Descriptors (according to ACM CCS): K.3.2 [Computers and Education]: Computer and Information Science Education – *Computer science education, Curriculum*, K.8.0 [Personal Computing]: General – *Games*

1. Introduction

Delft University of Technology introduced project-based education in its computer science (CS) curriculum five years ago, including a second year games project. Initially designed as little more than a companion project to the computer graphics course, our *games project* has matured into a large project integrating a broad range of computer science topics, and bringing our CS students to work for the first time in a realistic and interdisciplinary game development team, involving other students from game design studies. In such projects, a variety of facilities, or development framework, has to be provided to the students, in order to assist them concentrating on their project specific tasks, without dispersing their attention in many sideline tracks.

An important lesson we learned in these five years is that the quality of the deployed framework, including the whole development environment, from software libraries to documentation and technical support provided, plays a crucial role in achieving the desired CS, and in particular computer graphics (CG), learning objectives.

The pedagogical background and organisational setup of this project have been extensively described in [BBDH08a]. In this paper, we limit ourselves to motivate the evolution of a successful framework for our games project throughout the years: we discuss the most important challenges faced while continuously using, configuring and improving the framework, and describe the solutions we came up with to resolve those issues. At the end, some conclusions are drawn that should be useful to anyone running game design and development courses.

2. First lesson: focus on game, not on technology !

After the first running of the project, it became soon apparent that a few students were somehow frustrated and disappointed with the results achieved. Initial enthusiasm was taken down quite a bit due to the fact that they had not been able to concentrate on making their game, as they had to overcome many difficulties related to the programming language and support technology. Once students had finally learned how to work with the framework provided, they had very little time left for creating their game, which should have been the focus of the whole course.

As games are created using a lot of technology, an important part of any games project is the choice of the technology to work with. In this phase, the project was supported by the OGRE rendering engine [Ogr07]. This engine is written in C++, which was considered the industry standard. But in the context of "inexperienced" undergraduate 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 OGRE, an open source engine, was back then not easy to install and lacked quality support and documentation. On top of that, we needed the functionality of a complete game engine, which includes a lot more than just graphics. Students basically had to first set up OGRE, then choose a sound library, set that up to work and share the right details with OGRE. In some cases even a physics and collision engine had to be integrated. This all in a language they felt uncomfortable with, before they could even begin work on their actual assignment: the game.

All in all, the most important lesson learned was that the choice of supporting technology for your course or project should really be carefully made and aligned in straight relation to the learning objectives. In order to get students to focus on the actual course goals (e.g. applying computer graphics techniques and concepts in practice), you will have to carefully tune the choice of tools, language and libraries in line with those goals.

3. Seeing the light

During the second year this project was run (Spring 2004), we started to realize how we could improve the above situation. So a couple of students, who were particularly keen with games and their technology, formed a development group called Cannibal, and started to work on a new game engine. In order to adopt this new technology in our games project, we had set as requirements that (i) it was intuitive to use, (ii) it enabled students to realize their dreams and (iii) it was fun to work with.

Key aspect of this new engine was that it favored usability (ease of use) over raw performance. This made it easier to use, but also more manageable, both leading to increased productivity and a better focus on creating a game. This motivation also led to the choice of C# as the main programming language. Although C# is not an industry standard in game development, it has proven itself to be very easy to learn and work with [BAT04] and offers good performance.

Complementary to the choice of C#, Managed DirectX [MIL03] was also chosen as a managed graphics framework. Together, the Cannibal Engine and a managed framework based on C# and Managed DirectX, took away a lot of the technical details, so that students could 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.

After that first year using the Cannibal Engine in our course, a survey was conducted among the students. Results of this survey showed that almost 90% of the students found it very easy to install and update the Cannibal Engine. This was a major improvement, looking at the problems with OGRE in the previous years. Also, 100% of the students could get along with the new programming environment offered by C#. Regarding the structure of Cannibal, 80% thought it was very clean and very intuitive. Considering the problems and frustrations users experienced during previous runs of the project, and the promises made by the Cannibal Engine, these results confirmed the improvement.

During the project, student enthusiasm grew as they increased their programming speed and were able to accomplish a lot in a short amount of time. This led to them requesting lots of new features to accommodate their new found wishes, hardly leaving us any time for testing, which allowed for a number of issues to creep in. Although some small imperfections were still latent in the Cannibal Engine, and some new ones were introduced with every new feature being released, most of the students found that the problems were handled swiftly by the teaching staff.

All in all it seemed that the focus on usability and support, and not so much on features and performance turned out to be a very good and effective choice. Students could get right into creating their games, instead of having to first struggle with all the 'complicated' technology and language issues.

4. The next notch: supporting multidisciplinary work

Considering the immediate success of the previous years, 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.

The Cannibal team continued to improve their engine on usability, a number of new features and also on performance. With the addition of multidisciplinary teamwork and bigger groups, something had to be done about collaboration support.

4.1 Evolving the engine

The evolution of the Cannibal framework has been a continuous process driven by the eagerness of the Cannibal team to improve their technology. In this process, improving usability has always remained the focus point of the development taking user feedback into account. In addition, the team has also worked on new techniques like collision detection methods and different dynamic lighting models. The latter one has resulted in a version of the engine that used deferred lighting, theoretically allowing an unlimited number of dynamic light sources.

In this phase, one of the most important developments of the Cannibal framework has been the adoption of Microsoft's XNA [Xna07] . XNA not only enables Cannibal to provide cross platform development using C# on both Windows and Xbox360, but it also comes with a convenient development environment called Game Studio Express, further increasing the usability of the engine and the focus on students' support.

4.2 Collaborative environment

In early editions of the games project, only a marginal working environment was provided to the students, leaving the collaboration up to them. Students realized that collaboration was hard, but were handed no tools to cope with this. To this end the Cannibal team set up a working environment which allowed students to manage their documentation, code and assets and planning online.

With a change of curriculum at Delft University of Technology, the games project was expanded to one semester and, for the first time, it was combined with a Game Design (GD) project at the Utrecht School of the Arts, leading to a multidisciplinary, and therefore much richer, course [BBDH08b]. With this change, collaboration within each team became even more important as the project involved a larger, rather diverse group of students. More and more game development aspects were included in the process, so planning became more important with geographically dispersed groups.

To enable students to learn effectively from this collaboration process, a completely integrated working environment was provided in which they had access to a number of different collaboration tools.

First we wanted students to be able to share information and documentation about the product with each other in a flexible way. A Wiki system [CL01], specifically designed with collaborative editing in mind, is particularly suited for this purpose. The wiki system was also used quite often for discussion and communication.

To enable students to do effect planning with tasks and milestones, a ticketing (or task tracking [SC05]) system was also provided. This allowed students to discuss and create a planning and task division. Students could then check the status of their project online and share information on their progress with each other, leading to more insight in the overall team progress.

An integrated Subversion system [Pil04] was also provided, which allowed students to store their source code and game assets. As it was integrated into the rest of the collaboration environment, it allowed students to easily link their source to specific tickets and wiki pages.

As an extra tool a forum was provided by the HKU partners and this seemed to be a much desired functionality on their part.

4.3 Even more lessons learned

As students were moving faster and faster through course material and their requirements, the need for more and more information on more advanced game development techniques arose. Instead of having to explain techniques and concepts every once in a while, teachers and assistants were now overloaded with new questions.

As it turned out, collaboration between different disciplines is a lot more difficult than between like-minded people, even with the right tools available. Suddenly, artists had to understand technical people and vice versa; content created by GD students now had to be adapted to suit the needs of CS students and the capabilities of the development platform. Since these different disciplines cause problems also faced by many established game development studios in the past [RF98], some clashes were to be expected. In exceptional cases, mostly due to inexperience, this would lead to an 'over the fence' culture, where GD students would create content, throw it over the fence and blame the CS colleagues for it not working. These, in turn, would point the finger back at the GD students, leading to undesirable and non-productive situations. However, learning through experience, all teams were eventually able to work out their differences and become rather productive.

5. The Cannibal Experience

After the success of last year, the Cannibal staff realized the potential of their solution for educational settings, and started to work on their first professional product using the developed technology. 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 consists mainly of three components, each of which is considered critical for the success of any game related course, with an emphasis on project-based education.

5.1 Game Development Platform

As the Game Engine is now part of a professional product, it has been carefully revised and more rigorously tested. More features have been added and a lot of flexibility has cautiously been added, without compromising the usability and cleanliness of the API.

During development of the new version of the engine, special attention has been paid to allow students to work with the engine on their knowledge level. Students using game technology or the engine for the first time can use the engine at a very high-level, using only top-level features. When students want to delve deeper into the material they can start extending and modifying the behavior of the engine at any level they feel fit. They can for instance add new event triggers, add new sources of textures (e.g. a webcam) or even implement new input devices (e.g. the Wii controller [Thi07]).

Since most engines are specifically built for professional use, they are usually more difficult to start with. The shorter learning of Cannibal curve allows students to get started more easily.

5.2 Online Collaboration Environment

As discussed before the online collaboration environment contains several tools for the students to share work and collaborate. After careful evaluation a (simplified) discussion forum has been added to better facilitate online discussion between students at different geographical locations. This also allows students to work at different times, since a forum is by definition an asynchronous communication tool.

Having gone professional, Cannibal staff will usually be less involved in future educational projects than has been the case with the original games project described in this paper. Therefore different tools were provided for teachers to manage their course, enroll students, set up teams, and monitor progress. These new functionalities provide them with valuable insight, and helps them decide when and where to focus their guidance.

Besides supporting project planning and team collaboration, a complete community environment has also been integrated. Students and teaching staff alike can come together online and hold their discussions using Cannibal Experience. In addition, teaching staff have some private forums where they can discuss course setup and other educational or tutoring aspects.

The community environment also provides a way for all its members to communicate directly with the Cannibal staff, by means of a forum, for feature requests, bug reporting, submitting suggestions, etc. Cannibal, in turn, continuously provides the Cannibal Experience community with knowledge about the environment and its components. Creating a community and actively participating in it allows Cannibal and teaching staff to cope with the continuous requests for information and feedback from students.

5.3 Game Environment

All the aforementioned adjustments greatly contribute to the improvement of the framework. But there is still one element missing for students to get started right away: a collection of tutorials that students can study and build upon. These tutorials range from getting to start up the game to collision detection algorithms, character AI and working with content like textures, sounds, models and animations. For students to get started with the concepts and techniques even faster than before we created the Game Environment.

The Game Environment comprises a fully prepared virtual world where all these elements are covered. The game environment comes with a full set of game content items like textures, sounds, various shaders and different static and animated models.

Another aspect of the Game Environment is a tool chain that comprises a world editor and a game object editor. These tools can be used by non-technical team members to verify and configure their content in the game environment. Configuring a game object can be understood as assigning materials to models and setting material properties like textures etc. The tool chain is an indispensable component when working in an inter disciplinary team. With the tools provided, artistic team members can work independently on their content items and know that it won't give complications when they hand over their content to the technical team members.

For all aspects of the Game Environment short and comprehensible tutorials are provided for students to find and learn from. Having at your disposal all information on developing a game, is a significant part of the whole process of gaining experience.

5.4 Comparison to other Frameworks

Since the latest edition of the framework has not been extensively scrutinized and tested in practice, we will compare the framework with existing solutions in use.

At TU Delft, there were (and are) some systems in use to support the teaching of courses and projects. Among which are BSCW [BHST96] and Blackboard [Bla08]. Both systems are mainly used for sharing documents and managing their versions. The workflow of those system is to download a document, edit it locally and upload a new version on the system. This functionality is now offered by the Wiki system.

To accommodate the need to share and merge code, a Subversion is provided by the lab. Downside to this system is that the whole team will only get *one* login, making it impossible to track who did what, an essential part of teamwork.

A system for course monitoring, developed at the TU Delft, for the lack of a good system, is called CPM (Complete Project Monitoring) [CPM06]. Which allows teachers to manage their course, students, milestones and deliverables; something very comparable to the course management and course monitoring of Cannibal Experience.

All systems above work perfectly for the job they were designed to do. However, the biggest downside of the systems mentioned is that they are not all integrated. Cannibal Experience does incorporate all functionality as one integrated whole, in addition to providing the engine and matching documentation and support infrastructure. This is of course the advantage of a system explicitly designed for the purpose of teaching games in a curriculum.

Another system that also integrated a number of features, specifically aimed at education is Moodle [Moo07]. Downside of this system for games education is that the teacher has to setup the infrastructure and modules to use, which takes time and is not specifically aimed at teamwork. The advantage of such an approach is that it can be used in a great variety of situations and courses. However, there is no functionality to do planning, such as tickets, milestones and a roadmap or pre-provided content and materials for teaching a course, both essential for games education.

6. Educational Impact

Since the first time the project was ran [BDZ03], a lot has changed to the project and the curriculum. The games project has been given a rather prominent role in the curriculum as an integrator project, and its results are recognized and used to promote the curriculum to new and existing students. With the latest edition even establishing useful relations between educational institutes for interdisciplinary education.

We feel the framework has been an important factor in enabling students to get the learning experience and results

they achieved. It allows them to focus on creating their game using the examples and material provided. It also facilitated the collaboration so that students could get the most out of their teamwork. Without the results and experience achieved by the students, the project would not have become this important.

7. Future work

Now that our framework has grown into a professional product, time has come to use it in a wider range of projects and gain more feedback and knowledge in relation to different types of projects. Incorporating this feedback and knowledge will continue to improve the Cannibal Experience. We realize that as the computer industry, and the games industry in particular, keep moving, also the solutions will have to be continuously adapted.

Given the multidisciplinary nature of game development, the framework could also be used to support courses not directly related to creating a complete game. Interest is rising from other professors to incorporate game development in more specialized courses as well. For example a course on artificial intelligence, or more advanced computer graphics could also be taught using a game framework. Parts of a preprogrammed game could be left out, having the students fill in the blanks. The approach could also easily be expanded into creative areas, where e.g. the game has been programmed, but the content is missing.

8. Conclusions

The most relevant feature of the Cannibal framework described here, from its early steps to its current commercial Cannibal Experience, is that it has evolved from the accumulated experience of years of practical application in a rather successful educational project.

Our experience at Delft University of Technology confirms that a carefully designed development framework, including all underlying technology, course material and quality support, significantly determine the quality of a project-based game development course. In addition, when the teams in such projects have an interdisciplinary character, providing an effective collaboration environment is crucial for the success of team members. We believe that the key to the huge success of our games project lies, to a great extent, in the deployment of a professional working environment specifically crafted for an educational setting.

Games are, and have always been, all about fun. In pretty much the same way, our experience is that getting students in the position of making games can be even more fun. But the most fortunate of them are those who realize how much they have learned in that process.

9. References

[BAT04] BATES B. C# as a first language: a comparison with C++. *Journal of Computing Sciences in Colleges*, 19 (3): 89-95.

[BBDH08a] BIDARRA R., BOERS J., DOBBE J., HUIJSER R. The making of an interdisciplinary games project. *Journal of Game Development*, **3**(2). (forthcoming in March 2008).

[BBDH08b] BIDARRA R., BOERS J., DOBBE J., HUIJSER R. Bringing a pioneer games project to the next level. In: Proceedings of Third Annual Microsoft Academic Days Conference on Game Development in Computer Science Education, February 28-March 3, 2008.

[BDZ03] BIDARRA R., VAN DALEN R., VAN ZWIETEN J. 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

[BHST96] BENTLEY R. AND HORSTMANN T. SIKKEL K., TREVOR L. Supporting Collaborative Information Sharing with the WWW: The BSCW Shared Workspace System. In: Proceedings of Fifth International World Wide Web Conference, O'Reilly & Associates, Inc., 1996, pp. 63-73.

[Bla08] Blackboard Academic Suite. http://www.blackboard.com/

[CL01] CUNNINGHAM W., LEUF B. The Wiki Way. Quick Collaboration on the Web, *Addison-Wesley*, Boston (2001)

[CPM06] https://cpm.ewi.tudelft.nl/

[MIL03] Miller T. Managed DirectX 9: Graphics and Game Programming, *Sams.* Indianapolis, IN, USA (2003)

[Moo07] Moodle. http://moodle.org/

[Pil04] PILATO M. Version Control With Subversion, *O'Reilly & Associates, Inc.*, Sebastopol, CA, USA (2004)

[RF98] ROATHE L., FREGIEN C. CGDC '98 Roundtable Report 1998. http://www.gamasutra.com/features/gdc_reports/cgdc_98/roathe_fregien.htm

[SC05] SERRANO N., CIORDIA I. Bugzilla, ITracker and Other Bug Trackers, *IEEE Software.* **22**(2): 11-13 (2005)

[Ogr07] OGRE Team. OGRE website. http://www.ogre3d.org/

[Thi07] THIBAULT R.W. Wii Console. http://www.uweb.ucsb.edu/~rwthibault/Tech_Report.pdf

[Xna07] Microsoft Corporation. XNA Website. http://msdn.microsoft.com/xna/