Cannibal Collaboration Platform

BSc-project IN3700

PUBLIC VERSION

Project Team:
M.N.W. Smit 1097709
X.H. Tai 1234447
D. Terra 1220683
J.L. van Velthoven 1156020
Delft University of Technology,
Faculty EECMS

Cannibal Game Studios:
J.T. Dobbe
R.F. Huijser

Delft University of Technology:
R. Bidarra
M. Sepers

June 25, 2007

Foreword

This document describes the process of our work on the Collaboration Platform for Cannibal Game Studios (CGS). The idea for this project sprung from the minds of the CGS founders and became the subject of our bachelor internship. Our team consists of four Media and Knowledge Engineering students from Delft University of Technology; namely Mark Smit, Xiao-Hu Tai, Daan Terra and Lieven van Velthoven.

Besides working on the Collaboration Platform, we were involved in some additional activities. Most notably, as a welcome diversion from the more abstract work on the platform, we created several Graphical User Interface elements inside the Cannibal Engine to accommodate the needs of the project. Furthermore, every week an afternoon was reserved for self-development, meaning that we had time to read a book, website or white paper of interest to us. Obviously, on condition that the subject had some connection to the domain of the project. Physical labour was also part of the internship, as we were forced to move our office twice. Lastly, the company visited Microsoft DevDays 2007 in the Amsterdam RAI, where we manned a booth to send word about the current projects at Cannibal Game Studios.

Summary

The demands for the quality and scale of videogames have become so high in the last few years, that game studios are having increasing difficulties meeting them, unless the development process is changed. The goals of this project are to create a platform aimed at supporting a new iterative game development paradigm, improving collaboration within studios, automating trivial tasks, supporting project management and improving the overall workflow.

The solution we came up with is a framework that has a modular design and was built with usability in mind. Different components are created to facilitate the different goals for this platform. Together, these components form a solid framework to further build the Cannibal Game Development Platform on.

Contents

Chapter 1

Introduction

1.1 Motive

Currently in the game industry there are some problems that developers are coping with. General practices are based too much upon old software development principles. Game development however is a much more turbulent process and needs to accommodate many changes as requirements change during production. Furthermore a game development team consists of people from greatly varying disciplines having to work together tightly.

Nowadays, studios are creating next-generation games that are faster, have better graphics and are more immersive than the previous generations. The next-generation of computer games is both a blessing and a curse. The blessing is that the sky is the limit and stunning games can be made. The curse is that because of this, new games must be of the highest quality before they even stand out in the crowd.

Because of the high demands of customers, increasingly more work is needed to create games. Yet they need to be produced in approximately the same amount of time. This creates an enormous pressure on the development process since every delay can be costly and detrimental to the quality of the final product.

Cannibal Game Studios believes these problems can be solved by, among other things, providing a framework to support collaboration between everyone in the production process of games.

1.2 Goal

This collaboration framework creates a new game development paradigm. Game development teams mostly consist of people from different disciplines and backgrounds. The platform aims at increasing the collaboration between individual team members by supporting them in their communication. The main benefits of the platform are:

- To provide a structure for agile development that facilitates rapid prototyping and a large number of iterations in the design and implementation of the game.

- More automation in the development process, so developers can focus on the important things.
- Faster time to market, the ability to produce games faster.
- Increase collaboration/integration, better collaboration makes the process more efficient and more manageable.
- Teams can be more flexible
- Better quality; there is more time to actually work on the game because less time is needed to spend on the trivial processes involved in the creation of the game.

The objective of this project is to identify the needs of game studios and to design, integrate and implement the functionality needed to provide such a collaboration platform. Examples of basic collaborative software are Wiki, project management software, bug/task tracking, source code control, asset management, etc.

The goal of the internship is to provide a 'Proof of Concept', design and implementation of the basic structure for the framework. All decisions are made from a user-centered perspective. Extensive research on existing processes and systems will help in designing the platform.

1.3 Document Structure

The structure of this document is as follows, the problem description and analysis is given in chapter 2, followed by the project plan (3). The conclusion and recommendations are given in chapters 5 and 6, respectively and finally the project evaluation can be found in chapter 7.

Chapter 2

Problem Description and Analysis

Cannibal Game Studios did extensive research in a broad section of the Dutch Gaming Industry. Their conclusion was that numerous flaws can be pointed out that severely slow down the process and eventually lead to lower quality products.

Most companies in the industry work with some form of issue tracking system using tickets or tasks, however these are usually not integrated with the tools developers use. Therefore a lot of overhead is inherent with controlling the workflow. Currently, managing tickets is a cumbersome process involving a lot of clicking, assigning and changing states. Also, current issue tracking systems are very inflexible, allowing little or no change in established workflow.

Currently, exchanging files between team members of different disciplines requires manually changing the format of the file with some sort of utility program. Also, code files are usually under version control, but content files like Textures never.

Because everybody has a different role in the company, they often perceive the same items differently. For example, consider a texture; the artist probably wants to see it flattened, but programmers and modelers are more interested in seeing the texture projected on the models that use it. The project manager and creative director are probably only interested in whether the texture is completed or not and perhaps its quality.

Many people are involved in the creation of computer games. They all work simultaneously and often on the same general parts of the product. Furthermore, specialized companies exist to which parts of the product are outsourced.

Cannibal Game Studios believes that the gaming industry can significantly improve its efficiency to keep up with the next-gen demands. They feel this can be achieved by introducing a revolutionary *Collaboration Platform*.

Chapter 3

Project Plan

The domain of this project, automation of collaboration in the game development process, is very young and current solutions are still very basic. For such an immature domain the agile approach of software development is an appropriate method, because the path to the best solution is unclear and many detours will have to be taken. In the agile process creating throwaway prototypes to explore potential solutions is customary; a process called *spiking*. In this project the *eXtreme Programming* agile methodology will be used which includes pair programming, iterative development, continuous integration and design improvement.

Twelve weeks of time are available, which will be divided into multiple iterations per component. The outline of these iterations are very much alike and contain the following steps: research, design, implementation, testing and documenting.

3.1 Research

For the first iteration fairly extensive research will be done, because many ideas and concepts are expected to be found in existing software. The project members will also come up with ideas of their own. In the second and third iteration additional or more specific research will be done to fill up any gaps found in the preceding iteration.

3.2 Design

With the concepts found a design can be constructed, usually in the form of diagrams. The design will be discussed among the team members and with the superiors. The design will take a considerable part of every iteration.

3.3 Implementation

With a fairly complete design, coding will take place in pairs. Two pairs will work on separate small modules, which are integrated at the end of every day.

3.4 Testing

With the continuously changing design of the platform, a form of incremental testing is chosen instead of writing unit tests up front. This means everytime functionality is added, new tests will be created or existing ones have to be altered. Also some functional testing will be done by trying out the application.

3.5 Documentation

Research, designs and decisions will be documented on a private wiki server at the end of each iteration and also to smaller extent during the course of the process.

Chapter 4

Confidential

Chapter 5

Conclusion

The videogame industry struggles to keep up with the increasing demands on the quality and scope of next-generation games. After twelve weeks of research and development, a lot of progress has been made towards a collaboration platform aiming at solving these problems. This platform will save a lot of time and resources by increasing the efficiency within game studios.

The framework for the platform is a flexible system based on the server-client architecture, including an access control component, proprietary data types, and a Graphical User Interface system. The architecture was carefully thought through, leading to a clean and extendable design. Also, usability was an important consideration at all stages of the project. The result is a framework that allows multiple users to cooperatively work on the game, and integrates communication and project management naturally in the workflow.

Since the scope of this project is very large, it has yet to be finished. Even though there is now a fundamental basis for the platform, one cannot yet guarantee that it will be advantageous. However, the many new concepts introduced in this system caused the team to really believe in the power of the Collaboration Platform for increasing the productivity of its users.

Chapter 6

Recommendations

The Collaboration Platform presented here is only a proof-of-concept. It is by no means a complete and professional product. Therefore some things are left for future work as recommendations to Cannibal Game Studios.

Consistency

Some components of the implementation of the Collaboration Platform are currently not using each other's services. In the future the consistency of the implementation can be improved by doing so. This would also automatically assure that certain facilitaties are applied uniformly. The transparency of the implementation is also improved to provide future developers with a better understanding of the system.

Usability

In order to ship the product to customers in a complete, friendly and usable way, a number of back-end applications need to be implemented. These allow the user to manage the collaboration platform and all its settings. Users can then start working with the system in an easier fashion, allowing them to use the system to its full potential.

Optimization

The last important aspect of the system that has to be improved, especially when the platform is deployed at a large company are any and all forms of optimalization. One could think of client- and/or server-side caching and reducing the strain on the network.

Chapter 7

Project Evaluation

As an introduction to the Cannibal Game Studios company and its existing software, the team had an workshop on the creation of games using the Cannibal Engine. This was a pleasant first day experience that was very useful for the next item on the agenda, creating a 3D minigame. First, our knowledge of the concepts of ‘Agile Development’ and ‘eXtreme programming’ was refreshed during a 4h session of reading up and discussing the aspects of these principles. It seemed and turned out to be a good software engineering approach. Furthermore, in the first week the team created a small minigame with the purpose of getting to know each other, which was fun to do.

When we started the actual project, it was very unclear what the product should look like in terms of both functionality and design. However, Cannibal Game Studios had some ideas on several parts of the design. This allowed us to do a lot of research and brainstorming on a variety of domains. It was inspiring to be very free in the design process, because there are infinitely many ways to work out the Collaboration Platform. The two founders of the company were at all times highly involved in the process, which was very pleasant, since they were the most familiar with the workflow inside game studios.

Before we started coding, we had to narrow down the scope of the Platform and divide the work into separate components, so we could start experimenting with some of the possible solutions. This *spiking* was an important point throughout the project, because very often one can find the best solution by testing available options, to see which works best.

During the implementation there were many heated discussions in which weaknesses in a design would often surface. Even though these sessions were often very time-consuming, they were without a doubt essential for the creation of a better overall design.

Every week an afternoon was reserved for self-development, meaning that we all could read a book, website or white paper that interested us, which was a nice diversion from the full-time process.

Two weeks of development time was put into creating Graphical User Interface elements for the Cannibal Engine, because they were necessary for the Platform. Although this might seem like slightly unrelated work, it was a nice interlude because the nature of this work was far less abstract than the Platform. This abstract nature was one of the downsides of the project, because it is virtually impossible to tell or show others what you have done. Another downside was the gigantic scope of the project, as a result of which the product is not finished, even though this was never planned nor expected. From the start this project was intended to be a proof of concept and its desired outcome was a rough design. We think we have accomplished both.

All in all, it was a good experience.

Appendix A

Research

In the first few days the project team did some literature studies. A list of information sources that we looked into can be found below:

- Shark 3D by Spinor GmbH (http://www.shark3d.com/)
- RunIT (http://mmi.tudelft.nl/mkt5/2006-2007/index.php/Team2/Implementation1)
- RunIT SVN (https://cpsvn.twi.tudelft.nl/svn/RunIT)
- Subversion Hook Scripts (http://subversion.tigris.org/tools_contrib.html#hook_scripts)
- The Trac Project (http://trac.edgewall.org/)
- XNA Team Blog: The XNA Framework Content Pipeline (http://blogs.msdn.com/xna/archive/2006/08/29/730168.aspx)
- XNA Diaries: XNA Framework Content Pipeline (http://blogs.msdn.com/garykac/archive/2006/08/29/730464.aspx)
- MSDN XNA Game Studio (http://msdn2.microsoft.com/en-us/directx/aa937794.aspx)
- Alienbrain: Asset Management for Creative Teams (http://www.alienbrain.com/)
- Perforce: The Fast Software Configuration Management System (http://www.perforce.com/)
- Wikipedia: Intentional Programming (http://en.wikipedia.org/wiki/Intentional_Programming)
- Wikipedia: Semantic-Oriented Programming (http://en.wikipedia.org/wiki/Semantic-oriented_programming)
- Charles Simonyi (http://en.wikipedia.org/wiki/Charles_Simonyi)

- Papers by Lutz Roeder (http://www.aisto.com/roeder/Paper/)
- Intentional Software (http://www.intentionalsoftware.com/)
- Intentional Software by Charles Simonyi et al. (http://www.intentionalsoftware.com/technology/IS_OOPSLA_2006_paper.pdf)
- Intentional Software presentation (Charles Simonyi et al.) (http://www.intentionalsoftware.com/technology/IS_OOPSLA_2006_pres.pdf)
- CSCW (Computer Support for Collaborative Work)
  - (39) Designing Task Visualizations to Support the Coordination of Work in Software Development
  - (49) CVS Integration with Notification and Chat: Lightweight Software Team Collaboration
  - (89) FeedMe: A Collaborative Alert Filtering System
  - (139) A Lightweight Approach to Transparent Sharing of Familiar Single-User Editors
  - (181) tagging, communities, vocabulary, evolution
  - (195) Shared Waypoints and Social Tagging to Support Collaboration in Software Development
  - (239) The Wheel of Collaboration Tools: A Typology for Analysis within a Holistic Framework
  - (353) Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools
  - (259) Data Consistency for P2P Collaborative Editing
  - (373) The Mystery of the Missing Referent: Objects, Procedures, and the Problem of the Instruction Follower