Strings Fighter: A millennial history of struggle between elements of writing

André Luis de Lima Martins[1], Anderson Szalai[2], Silvia Maria Farani Costa[3]
[1]andre@intentor.com.br
[2]anderson@askm.com.br
[3]silviafarani@yahoo.com.br

FATEC Carapicuíba
Avenida Francisco Pignatari, 650
Vila Gustavo Correia, Carapicuíba/SP Brazil

***Abstract*— Strings Fighter is a fighting game created in Java which uses elements of communication for composition of its history and gameplay. The objective of this work is to show how the game was engineered and the problems faced during its development.**

***Keywords*— game design, software engineering, Slick, Java, Lua**

I. Introduction

Fighting games, despite their names, are not the most violent type of games. Their violence is often cartoonish and stylized, being only a mean to provide a deep, complex and rich gameplay in a way that any other game style can't offer.

In gaming jargon, it's quite common to call a battle on a fighting game as a piece of art. Given that players have to master many different aspects and commands, each round can be a unique portrait of skills and beautiful movements presented by a multitude of lines and colors.

Strings Fighter, being a fighting game settled in the communication world, is a project that wishes to be as fun as it is graceful and singular on its genre. Presenting letters and symbols struggling with unique gameplay and stylish features, having on its characters the most charismatic and entertaining element of its presentation, Strings Fighter is a fresh attempt to create a different approach in the so repetitive universe of fighting games.

II. Objectives

The project, despite being a college work, was made focusing in the use of common gaming industry technologies and techniques, having on the franchises Street Fighter, Guilty Gear, BlazBlue Clay Fighters and The King of Fighters its source of inspiration.

College Project

As a college project, Strings Fighter had some rules to be followed and clear objects in its delivery.

The main goal of the project was to use all the knowledge achieved on classes to create a local/online multiplayer game with language options in Portuguese and English, with a certain date to be presented to an examination board, who would analyze the work, technologies involved, documents created and presentation approach of the group.

At college, there were two main classes which were sole dedicated to the composition of the game, and another 8 hours of extra-class activities for each member of the project's group.

Audience

The game was planned focusing on players that are either accustomed to fighting games or new to the genre, using RPG elements in a simple way to create a consistent score and battle system, organizing the gameplay in a way that it can be as comfortable as possible for the hardcore gamer and for anyone that are not used to play this type of game. The use of charismatic characters is also used to draw attention of people that are not so familiar with the fighting game genre.

Technology

As a game project, technology plays an important role in its conception. Being the game itself a software, the platform, programming language, game engine and support applications chosen can define the success or failure of the project.

Below are enumerated all technologies used on the project. The choices were based on their applicability, direct benefits offered, continuous evolution of the technology, knowledge base available, money saving and market recognition through real cases.

First of all, Slick 2D Game Library [1] based on Java and OpenGL was chosen to develop the main core architecture of the game, based on the power and consolidation offered by the Java platform through its object-oriented design, multi-platform operation, game libraries availability and integration with third other technologies. Another reason to choose a game engine based on Java was the small learning curve offered by the platform, despite some performance leak, and the easiness to change the baseline of the project, in comparison with a C or C++ approach, despite these two languages being the natural choice for a game development based on performance.

To retrieve information from the multiplayer server, web services were chosen to offer a simple way to deliver lightweight information through web using SOAP protocol. To achieve this, ASP.Net Web Services in C# language were chosen, coupled to a multi-tier solution linked to an ORM (Object-relational mapping) utility, Yamapper [2]. This choice was made on account of prior knowledge and experience of the team with the platform.

Slick and ASP.Net together are sole responsible for all the core aspects of the project, like renderizations, user input, multiplayer communication and collision system interpretation. Lua [2] scripting language appears on project to solve one of the critical points in a game development (and also on every software development): the constant changes the project receives during its development phase to tune all its features. If it was needed to recompile the engine every time a change had to be made by an artist, critical time would be waste that could be used to other important activities for the project. Lua gave the power to test changes on the fly, uprising the game development performance.

Characters design, scenarios, movements, illumination, sprites creation and many others visual aspects of the game were created using Blender [3], Gimp [4] and InkScape [5]. All of these applications are open source solutions with huge communities and knowledge support databases.

Blender is a well-known open source software for 3D modeling and animation that was used to design and animate all the characters and scenarios. Gimp is an application for image manipulation and processing, compared in features with Adobe Photoshop, which was used for all image treatments throughout the project. InkScape is a vector graphics editor, such as Corel Draw, that was used to create and organize sprites for the game, [6].

So, thinking on a hierarchy of game elements and technologies involved, Blender allows the creation of animations, GIMP aids the enhance of images quality, InkScape ensure organization and creation of sprites that will be parsed by scripts in Lua which were interpreted by Java in Slick that communicates with an ASP.Net Web Service to gather all the required information about multiplayer games.

III. Project Definiton

Strings Fighter was designed to comply with some game industry standards to present a game that is lightweight to be executed, faster on its characters movements, beautiful on its presentation and easy to expand if it’s needed.

History

The history takes place nowadays, in an ancient struggle between Symbols and Letters for their own beliefs. However, it started a long time ago, more than 6000 years from now.

The symbols ruled the communication since the dawn of Man, allowing human ancestors to express their feelings and eternalize their history and traditions.

After the discovery of writing, symbols begin to feel threatened; fearing that the creation of a consistent writing system could led them to oblivion.

And, in about 3200 B.C., with the creation of hieroglyphs by the Egyptians, the Symbols started a period of time which became known as Demotic Wars.

Believing they were the personification of the gods on Earth and in a frustrated attempt to exterminate the first Letters and subjugate humanity, preventing the development of writing and returning Man to their gods and traditions, the Symbols took advantage of their use in mystical rituals and created a series of parchments whose contents gave magical powers to the carriers symbols. On the other hand, the Letters, considering themselves as servants of humanity and also being a form of symbol, used the same magical powers of the parchments, matching symbols during battles – and the Demotic Wars ended with the victory of the first letters and subjugation of symbols, who were banned from the writing universe and relegated to ritualistic and occult practices.

Nonetheless, Symbols remained still for centuries plotting their triumphant revenge against Letters, silently establishing themselves in all areas of communication, just awaiting for the right moment of their return…

And this return is the moment when the game takes place.

Architecture

Using a game engine like Slick and an object-oriented language like Java helped too much on planning and creation of the project.

The architecture focused on letting all the animation, character structures, scenes and scenarios data and all the language texts outside the engine, using Lua as a tool to process many aspects of the game. This approach allowed artists to define animations and apply sprites without needing to know how the game really handles them, turning the creation of scenes and gameplay features easier and faster.

The collision system was also created using Lua, in which all the information about hitboxes is processed. And, because the collision system uses many mathematical aspects that could be tough to be understood by artists and hard to create by hand by programmers, a simple application in C# was developed, with an easy-to-use graphical interface that made easier to configure hitboxes for each frame of each animation of the game.

For the multiplayer side of the game, an architecture internally called Socket Processor Architecture was created, in which all the user input is handled using an interface that is implemented by either a local input processor (for the player who is in the local machine) or a network input processor (for the player who is in a remote machine). These processors are bound to a server (the host game) or a client (the client game) processor which manage and translate all the requests/responses during the game.

Additionally, to store all data from the players on the MySQL database, a web service in ASP.Net using ORM Yamapper was created, in which a proxy class in Java was responsible for access.

Methodology

Even not using an agile framework/methodology, agile engineering techniques were used to help and to ensure definition of activities and theirs deadlines. Given that many of the tasks during the project were divided in a way that they could be created with minimal dependencies, each member of the team could work on its part of the project without affecting other members if for some reason a delayed happened.

To manage source code, a centralized Subversion repository on the Internet was created, in which all the source code, gaming assets, graphical files, scripts and documents were stored.

In terms of documentation, besides the mandatory document that describes the project and that had to be delivered to the examination board, a few other documents were created, being the game design document and architecture diagrams the most important of them. The first was used to define and share throughout the team the history, gameplay, goals and mathematical formulas used in the game. This document also helped artists on creating characters based on its motivations, beliefs and fighting styles. The diagrams were mainly used to help development team communicate ideas and possibilities that could conduct to prototypes of components and structures which could or could not be used on the game.

IV. Development

Developing a game is not easy. And for a fighting game, in which the response of user input should be immediate and the collision system fair and well settled, the difficulties just increase.

And, despite the problems of collisions and gameplay, creating a consistent multiplayer experience without slowdowns was a hard and painful development challenge.

The experience of team members varied from high 3D modeling and handcraft skills to years of working on corporate applications. However, despite the first game created for college, a version of naval battle with simple IA and some classical game design concepts, most of the members of the team had not worked on a game project before. And although it could led the project to a decrease of overall quality, the commitment and efforts of the entire team allowed the game to be the best possible within the time we had available.

What Went Right

The technologies chosen helped the team focus on ideas rather than implementations. The organization of the team, with people that had complementary knowledge and full understanding of the concepts about what had to be done, assured the self-management of each member, which was crucial during development because of the short time available.

The use of Lua was also one of the most right choices the team had made. This simple script language allowed the project to be easily configurable without recompilation of the engine.

During planning and initial development of the game, the choice to not send all the multiplayer information to the server throughout the time of a game led to a creation of a better and efficient client-server communication through sockets, involving only the player's games.

Although the game lost many of its initial planned features because of time issues, the team could focus on a few set of aspects, making them the best as possible.

What Went Wrong

Short development time (3 months) and concurrent college works were the biggest problems faced, which led to a huge cut of features and simplification of early game design decisions.

The creation of an efficient multiplayer system was also a huge challenge for the development team. Even though following the communication protocol created was simple, ensure the synchronization of both players was more complicated than expected.

Conclusion

Despite of all team members experience and knowledge on enterprise application development, games have a different approach to be achieved. There are many new layers of project to be considerate that goes far from the basic formula analysis, design, implementation, tests and deploy, and many aspects of architecture and player interaction that became critical points of decision during the development.

Choosing a technology is not just a simple layer of requirement, but an important aspect that leads a game project to success or failure in the end. Support libraries, easy-to-implement interfaces and a fast rendering system are not just technical aspects of a game project: they are requirements that could be the difference between a well-designed game, with good performance and state-of-the-art gameplay, and a poor project, even if the game has all the initial design requirements implemented.

The adjustment of technologies according to the relation scope and time availability is also a critical point of decision. Working with heterogeneous technologies, although complex at first glimpse, proved to be the right approach to give all the technology necessary to create a well designed game with good performance and on schedule.

Game design is an art based on understanding the critical points of the genre of the game that is in development and choose and adjust the necessary technologies to achieve these points. It can be hard and painful sometimes make the right choices. But, in the end, it's the player's entertainment that counts.

References

[1] Slick 2D Game Library. Available: http://slick.cokeandcode.com/ [Accessed: May 1, 2010]

[2] Yamapper Object-relational mapping component for Microsoft .Net. Available: http://intentor.com.br/projects/yamapper/ [Accessed: May 1, 2010]

[3] Lua, a lightweight scripting language. Available: http://www.lua.org/ [Accessed: May 1, 2010]

[4] Blender, free open source 3D content creation suite. Available: http://www.blender.org/ [Accessed: May 1, 2010]

[5] Gimp Image Manipulation Program. Available: http://www.gimp.org/ [Accessed: May 1, 2010]

[6] InkScape, an open source vector graphics editor. Available: http://www.inkscape.org/ [Accessed: May 1, 2010]