Departament of Information Science and Technology

A Collaborative Framework for Browser Games Development

Jorge Pena

A dissertation presented in partial fullfilment of the Requirements for the Degree of Master in Open Source Software

Supervisor:
Dr. Carlos J. Costa, Ph.D., ISCTE-IUL

April, 2012

Abstract

This dissertation describes a conceptual model and prototype for a collaborative framework for browser games development using open source and open content. There is an extensive literature review exploring several areas like game development, modding, open source software development, open content and creative commons. The most relevant ideas about game development and collaboration are then used in defining the conceptual model of the framework with the objective of facilitating community creation and collaboration. Finally the implementation of prototype is explained in detail and the practical difficulties in implementing the conceptual model are addressed. This research shows that a collaboration framework for creating open source and open content browser games is possible and paves way for future studies about the community creation in this type of collaborative systems.

Keywords: Games, Browser Games, Open Source, Open Content, Creative Commons, Collaboration

Resumo

Esta dissertação descreve um modelo conceptual e um protótipo de um sistema colaborativo para o desenvolvimento de jogos no browser utilizando código aberto e conteúdos abertos. É feita uma revisão extensiva da literatura em várias áreas como desenvolvimento de jogos, modding, desenvolvimento de software open source, conteúdos abertos e Creative Commons. As ideias mais importantes acerca do desenvolvimento de jogos, conteúdos e colaboração são usadas para definir um modelo conceptual do sistema com o objectivo de facilitar a colaboração e criação de uma comunidade forte. Finalmente a implementação de um protótipo do sistema é explicada em detalhe e são referidas as principais dificuldades práticas na implementação do modelo conceptual. Esta pesquisa mostra que é possível criar um sistema colaborativo para a criação de jogos como código e conteúdos abertos e abre caminho para futuros estudos sobre a criação de comunidades neste tipo de sistemas colaborativos.

Palavras-chave: Jogos, Jogos no Browser, Código Aberto, Conteúdo Aberto, Creative Commons, Colaboração

Acknowledgments

First I would like to express my gratitude to my supervisor, Dr. Carlos J. Costa, for his insights and constant help while doing this work. The topic of this dissertation came about when we found we both had similar ideas and interests about the topic of games and online collaboration. I am grateful for the long talks we had about the subject and they helped some of the ideas contained here mature and grow.

To my wife Cláudia, and my daughters Matilde, Madalena and Mariana I would like to thank them for the support they gave me and the absences they endured. They are my inspiration in life. Also I would like to thank my parents and family for always teaching me that education is important and encouraging me to always try to better myself.

Finally I would like to acknowledge the help of my classmates in the Masters in Open Source Software for the encouragement, ideas and companionship.

Contents

1 Introduction **11**
1.1 Motivation . . . 11
1.2 Research Question . . . 11
1.3 Methodology . . . 12
1.4 Document Structure . . . 12

2 Literature Review **13**
2.1 Game development process . . . 13
2.1.1 Overview of the game development process . . . 13
2.1.2 Frameworks and methodologies . . . 14
2.2 Game modding . . . 15
2.2.1 Types of mods . . . 16
2.2.2 Licenses . . . 17
2.2.3 Infrastructure and development tools . . . 17
2.2.4 Community practices . . . 18
2.3 FOSS games development . . . 18
2.3.1 Requirements analysis and specification . . . 18
2.3.2 Coordinated version control, system build, and staged incremental release . . . 18
2.3.3 Maintenance as evolutionary redevelopment, reinvention, and redistribution . . . 19
2.3.4 Project management and career development . . . 19
2.3.5 Software technology transfer and licensing . . . 20
2.4 Open Content and Creative Commons . . . 20
2.4.1 Motivations for Open Content collaboration . . . 21
2.4.2 What is Creative Commons . . . 21
2.4.3 Commons based peer production . . . 21
2.4.4 Community of practice . . . 22

3 Conceptual Model **23**
3.1 Usage and roles . . . 23

3.2 Generic framework concepts . . . 24
3.3 Media library . . . 25
3.4 Game engine . . . 26
3.5 Game creation tool . . . 26
3.6 Game portal interface . . . 28

4 Prototype Implementation **31**
4.1 Overview . . . 31
4.2 Tools And Technologies . . . 32
4.2.1 Drupal . . . 32
4.2.2 HTML5 (Canvas) . . . 33
4.2.3 Javascript . . . 33
4.3 Games Portal Prototype . . . 34
4.3.1 Drupal content types . . . 34
4.3.2 Core modules . . . 34
4.3.3 Third party modules . . . 36
4.3.4 Forum . . . 38
4.3.5 Media Library . . . 38
4.3.6 Game content type . . . 39
4.4 Game engine . . . 40
4.4.1 Overview . . . 40
4.4.2 User Interface initialization . . . 40
4.4.3 Assets loading . . . 40
4.4.4 Viewport initialization . . . 42
4.4.5 Game loop . . . 43
4.4.6 Collision detection . . . 43

5 Results Analysis and Discussion **45**
5.1 Conceptual Framework . . . 45
5.2 Open Source Browser Games . . . 45
5.3 Prototype Limitations . . . 46
5.3.1 Content *Forking* . . . 46
5.3.2 Awards and contests . . . 46
5.3.3 Sound content . . . 46

6 Conclusion **47**

A Game Engine Source Code **51**
A.1 game.js . . . 51
A.2 game.css . . . 56

B Prototype usage & installation **59**
B.1 Accessing and using the prototype 59
B.2 Installing and running prototype 59
B.2.1 Common requirements 59
B.2.2 Snapshot installation 60
B.2.3 Clean installation 60

List of Figures

2.1 Game development activities 14
2.2 Collaboration contributing factors 16
2.3 Pyramid meritocracy (Scacchi, 2004) 19

3.1 Framework basic components 23
3.2 Framework main use cases 24
3.3 Collaborative Features 25
3.4 Media Library Use Cases 26
3.5 Conceptual game engine 27
3.6 Game Creation Use Cases 27
3.7 Game Creation Process 28
3.8 Portal Use Cases 29

4.1 Prototype Portal Home 31
4.2 Canvas Example 33
4.3 Content type administration in Drupal 34
4.4 Taxonomy organization interface 36
4.5 Fivestar voting field configuration 37
4.6 Basic forum organization 38
4.7 Media library top content 39
4.8 Game node view with play button 41
4.9 Game playing interface 41
4.10 Game engine activity diagram 42
4.11 Game over 43

Chapter 1

Introduction

1.1 Motivation

Computer games are an increasingly popular application of computer technology (Jones, 2000) , and a very important part of today's entertainment culture.

Game development is a complex and demanding task that typically needs teams with various skills and expertises to create an interesting and viable game(Bethke, 2003)(Costa and Aparício, 2006b). In the context of Free and Open Source Software (FOSS) games, the need to have programmers, designers, graphical artists, musicians and others on the team make the barriers to entry somewhat high.

Various researchers have studied the software development process applied to games development, and its particulars both in commercial and FOSS settings.

In a related movement to FOSS, Open Content and Creative Commons allows for collaboration and sharing in various fields not related software development, like writing, arts, music, and others.

1.2 Research Question

The main purpose of this work is to design and prototype an on-line collaborative framework for browser games development. Users will be able to upload graphic and sound content to the library, define game rules and objectives, and then to play their created games and share them with other people. While there are other on-line game creation frameworks, to my knowledge none combines both open source technologies and an open content library.

Ultimately the purpose of this research is to access the feasibility of an open collaborative development model in the specific area of browser games.

This is also related to work being developed by a team of researchers in ISCTE-IUL (Costa and Aparício, 2006a,b).

1.3 Methodology

The research methodology used in this work according to Järvinen (2004) taxonomy of research methods fall into two different but complementary approaches.

The conceptual model of the framework presented is an example of a conceptual-analytical approach because its concepts are derived from the review of the available literature and studies in related topics.

The prototype framework developed falls in the category of artifact-building approaches. The prototype validates the feasibility of the conceptual model and of the premise that a framework for open source collaborative browser games development can be built.

The two approaches are complementary in this case because the design principles of the prototype are guided by the conceptual framework devised by the analysis of various bodies of research pertinent to the proposed problem.

1.4 Document Structure

This document is divided into three main topics: literature review (chapter 2), conceptual model (chapter 3), and prototype implementation (chapter 4).

In chapter 2 (Literature Review) various research areas relevant to the objectives of this research are reviewed. The development process of games is analyzed to find the differences from software development in general and specific problems and processes that may be relevant to the conceptual model. Next game modding and its community approach to game content creation and modification is reviewed to gather data about game specific communities and approaches to collaborative game development. Free and Open Source development and games development is also reviewed to find how disperse and diverse groups of developers collaborate in successful development projects. Finally the Creative Commons and Open Content are analyzed to extract information about open content licensing and open content creation communities.

In chapter 3 (Conceptual Model) the insights and concepts gathered in the previous chapter are streamlined into a coherent conceptual guide for building an open source and open content game creation framework. The main characteristics and parts of the framework are described and the interaction between the various agents is also conceptualized.

Chapter 4 (Prototype Implementation) describes the implementation of the framework prototype using a simple browser game engine. The tools and technologies used are also described and the difficulties in implementing the conceptual framework in a real world case are also addressed.

Chapters 5 and 6 (Results and Conclusion) describe the results of this research and identify possible avenues for expanding and continuing it in the future.

Chapter 2

Literature Review

I am interested in reviewing literature related to the study of the software development process applied to games. I will look at commercial game development in one hand and open source game development. There is also a field tangential to game development that is game modding and mod development. The game modding communities are a lot like FOSS communities and are an interesting research topic in this area.

Of interest to the problem is also the study of the processes used in the game modding community. I also will look at some examples of collaborative content creation from others fields to find out some practices that can be used to help answer the research question.

2.1 Game development process

Commercial game development is a multidisciplinary team effort, requiring specialized knowledge and skills. The game development industry has been growing and is a very important business today. Games marketing is in every medium and some games have production values and revenues comparable to the movie industry.

2.1.1 Overview of the game development process

Having an idea for a game is not enough. It must be elaborated allowing for a team to start working. These teams can include a lot of people with different roles (Costa and Aparício, 2006a; Tran and Biddle, 2009):

- programmers
- graphical artists
- musicians and sound experts
- designers
- managers

- producers

Costa and Aparício (2006a) propose a framework that divides game development in several activities:

- Conception
- Game specification
- Storyboard production
- Analysis and Design
- Programming and production
- System Test

These activities can then be engaged by the team using various approaches and the end result of this process a playable and fun game (hopefully) that can be played and enjoyed.

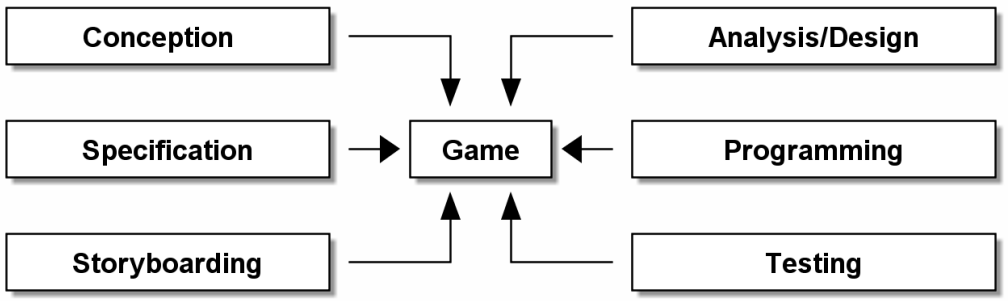

Figure 2.1: Game development activities

2.1.2 Frameworks and methodologies

There are some research papers studying the game development process and proposing frameworks and tools to improve the process. In this section I'll make an overview of these frameworks and tools.

In the process proposed by Costa and Aparício (2006a) they adopt an iterative waterfall development model encompassing all the different activities that contribute to the creation of a game. In the conception activity the team engages in brainstorming meetings to generate ideas for the game. In the game specification phase the product manger writes a document that describes the game from an end user point of view. The document

specifies what the game is about, explains game play, can have concept art, and describes the features and characteristics of the game. Next in the storyboard production phase a series of sketches and mock-ups are created detailing the scenes and sequences of the game and user interaction. Art style is also defined in this phase. In the analysis and design phase software engineers create the technical architecture of the game using UML. Also the tools and platform for the game are defined. In the implementation phase all the documents generated previously are put to use by the various teams to create the game art, and a functional program that compiled into a function game. In the testing phase the game is tested and feedback is generated and sent back to the teams that will take the appropriate measures to correct the problem. This process goes on until the game is deemed ready to be sold to the public. In another case study by the same researchers (Costa and Aparício, 2006b) they found that the application of this process in a real life situation may be very chaotic.

Tran and Biddle (2009) did an ethnographic study of collaboration in a game development team. In this study they collected data on a team developing two games for business training. The team members had the following roles: game programmer, lead graphic artist, graphic artist, graphical user interface designer, usability expert and a manager. The researchers analyzed the type of work done, the collaboration in the development process, the office space, social atmosphere, role differentiation, information sharing, product goals and other variables. The authors identified three core factors that contributed the most to a successful collaboration in the team: role respect, short iteration cycles and shared vision. Role respect means that each member of the team has a clearly defined understanding of what he contributes to the game. Team members acknowledge the roles are complementary and everyone is important to the end goal. Short iteration cycles means that the game is changed and tested almost daily leading to a continuous evaluation and discussion of the product by all team members who keep 'up-to-date' with the development process. Shared vision means that there is an understanding and agreement about the end product. There are no conflicting views of how the game should be. According to the authors this leads to a strong collaborative spirit in the team and means the team works by default in collaborative mode.

2.2 Game modding

Modding is a slang expression derived from the term *modify*. It refers to the act of modifying a piece of software or hardware to perform a function not intended or designed by the original author. In the context of games it refers to the modification of game software and hardware in order to extend or change the subject (Flew and Humphreys, 2005; Kücklich, 2005).

Game modding and its community have been an example of community driven inno-

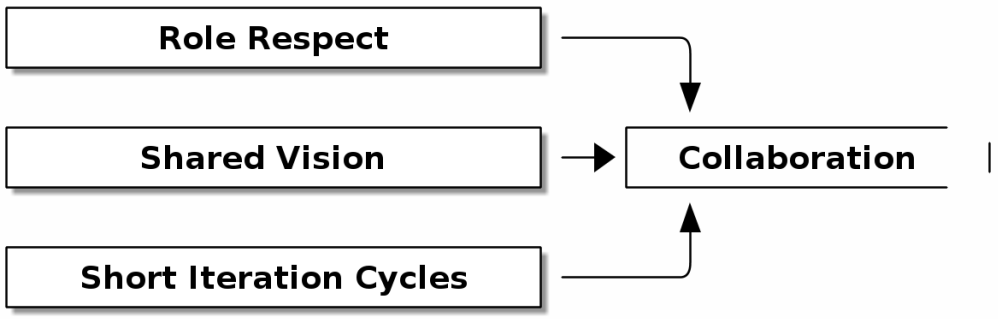

Figure 2.2: Collaboration contributing factors

vation and collaborative development (Scacchi, 2010). Game users acquire the games and then go on to modify, create and remix the game content with or without the encouragement of the game developers themselves. Questions arise about content ownership and the relationships between modders and game developers and companies. The author argues that a web of *affordances* allows games to modified, and that these affordances shape the very modding community governing who mods what, where, when, how and why.

2.2.1 Types of mods

Game mods can mean different things to different people. So I will try to categorize the various types of modding going on and their significance for this work (Table 2.1). All kinds of modding are a kind of content remixing and the way this is done is important in determining the best strategies to facilitate this. Scacchi (2010) identifies the following relevant types: user interface customizations, game conversion, art mods and console hacking.

- User interface customizations
  In this type of modding the focus is on changing the visual interface of the game and is the more widely supported by game studios as way of increasing user satisfaction (Pine and Davis, 1993). The kind of changes allowed is different from game to game and range from purely visual customization (textures, models) to game play enhancing information displayed on the screen.

- Game conversion
  This type of modding imply access to the game engine (normally with the blessing of the creators) and to the tools necessary to change and tweak every aspect of the games. Notorious games which support this are Half-Life, Neverwinter Nights. In

this type of modifications modders can tweak and change game content, add new content, and even create a completely new game using the provided tools.

- Machinima and Art Mods
  Machinima is the creation of movies using in-game scenes and content. Some movies just use the default game content and art but others can use the conversion tools to adapt the game engine to their needs. In machinima game play experience is used for the purpose of story telling and or gameplay demonstration.

- Game console hacking (home-brew apps)
  This kind of modding is the usage of the consoles hardware for ends outside the creators of the consoles intended usage. Usually this entails the exploitation of some kind of software or hardware bug to run user created software on the console. The usages range from installation of general purpose operating systems (Linux), user created games (called home-brew), and other software intended to broaden the utility of the hardware (media players, chat, etc.).

Table 2.1: Game Modding Types and Characteristics

Type	Interface	Graphics	Game Rules	Game Engine	Hardware
Interface	✓				
Game Conversion	✓	✓	✓	✓	
Machinima		✓			
Homebrew					✓

2.2.2 Licenses

In the modding scene the form of licensing used is very important. Some games only allow user interface modding and no engine reverse engineering or customization. Others allow the usage of the engine but not of the game content or artwork. Some games encourage the free redistribution of modded content as a means of increasing the game popularity and appeal for new new gamers.

It is no surprise that the most thriving modding communities exist where the game studios have the more liberal licenses and even encourage the creation of mods.

2.2.3 Infrastructure and development tools

The quality and availability of the tools that enable game modification also plays a major part in the success of the modding communities. Games that are built on modular engines with good modification tools are more likely to be modded. Some studios release to the public the same tools they used to create the game, allowing for mods that have the same

technical quality as the main game. Clearly this is also an important enabler for the creation of a strong community and quality mods.

2.2.4 Community practices

Unlike game programmers modders do not get paid to create mods for commercial games (Kücklich, 2005). Sometimes cash prizes can be awarded at game modding competitions (Sotamaa, 2007), but this an exception. The main incentive for game modders, aside from the possibility to enhance their own game playing experience, is the ability to learn about game development and maybe use those skills later to find a job in a game studio. These skills and knowledge are complemented by gaining status among peers, community recognition and reputation. It can be said the modders are exchanging their effort for some kind of social capital that will benefit them later on (Portes, 1998).

2.3 FOSS games development

According to Scacchi (2004) FOSS development communities do not use traditional software development methodologies and are contributing to a new view of how complex software can be developed. Regarding FOSS games development Scacchi (2004) identifies five software development processes being used across FOSS development communities:

2.3.1 Requirements analysis and specification

The author did not find evidence in any of the projects of formal requirements and specification documents as prescribed in software engineering methodologies. Apparently all requirements and features are discussed at length by the community in threaded message lists (mailing lists) and summarized in web pages after the implementation.

2.3.2 Coordinated version control, system build, and staged incremental release

FOSS projects rely heavily on version control tools like CVS, SVN and more recently Git, Mercurial and Bazaar. These distributed version control tools allow the distributed and collaborative development effort. Control over what features and patches are accepted into the project is also enforced through these tools allowing for the core developer to decide what is accepted. Some project have a more formal decision process using voting while other use more informal and ad-hoc methods. These projects also rely on mailing lists (threaded messages) to discuss and document the proposing and accepting of contributions. The combination of version control and communication tools allows the management of system builds and releases.

2.3.3 Maintenance as evolutionary redevelopment, reinvention, and redistribution

The maintenance of FOSS projects by the adding and or removing of features, debugging, translation, optimization and multiple other tasks, is a widespread and recurring part of FOSS software development. But is such a dynamic process that can be characterized as reinvention. The projects are constantly evolving and changing to suit their users and developers by accumulating lots of very small changes across very small life cycles. Projects can have daily builds and alpha, beta and release candidate versions evolving to a stable version that begins the process anew.

This culture of reinvention allows even communities to take over old and historical closed source or proprietary projects and recreate them as FOSS projects that are sustained by the motivation and interest of their members to keep alive and develop these old systems. The author gives the example of the MAME (Multiple Arcade Machine Emulator) project that has brought to life thousands of old almost lost arcade games.

It's noteworthy that FOSS projects evolve along with their communities and after acquiring a critical mass of 10-15 core developer they can grow almost exponentially defying the old laws of software evolution.

2.3.4 Project management and career development

Typically a FOSS project is organized as a pyramid meritocracy (Scacchi, 2004). In this form of organization authority, trust and respect are awarded to the most experienced and influential contributors to the project who assume positions of more responsibility in the project. Normally there are more than one person in this decision making position and there is shared decision making.

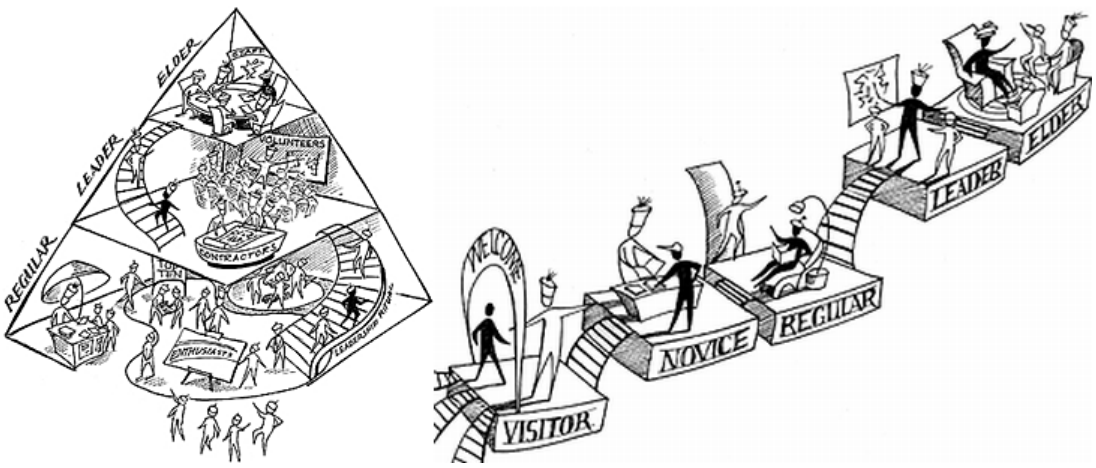

Figure 2.3: Pyramid meritocracy (Scacchi, 2004)

These core teams seem to prefer incremental changes and innovations and if a developer or group wants to push for a radical change this core team must be convinced or otherwise

the proposers may fork the project. In the long run few forks are successful and so this reinforces the incremental innovation mindset.

The structure of FOSS communities tends to be logically centralized and physically distributed and supported by tools like CVS and discussion lists. This leads to a kind o virtual project management where there is a process of community decision making that is directed and controlled by the core team which encourages certain patterns of development and social control.

As for the motivation of the developers: some just participate for the fun and a personal sense of accomplishment, others use FOSS development to exercise and hone their professional skills, and other still thrive on the social capital gained through networking in the community and peer recognition. All of this motivations may be present in one FOSS developer albeit in different degrees.

2.3.5 Software technology transfer and licensing

While software technology transfer is neglected in other areas, in FOSS the diffusion, adoption and usage of other FOSS software systems is an widely adopted practice. This practice is in the core of the rapid evolution of FOSS software systems. Tools like control version systems, development tools (compilers, ides, debuggers), database management servers, web servers, content management systems, operating systems, and communication tools that are themselves FOSS projects are routinely used, adapted, and built upon by other FOSS projects. This socio-technical process enables FOSS to exist with no centrally planned and managed software development centers.

The one thing that facilitates the above process is the FOSS software licensing (Elliott and Scacchi, 2004). The GPL and other similar licenses reinforce the beliefs and practice of sharing, studying, modifying and redistributing software, and are a kind of glue that maintains the FOSS community and fosters it's growth and expansion.

2.4 Open Content and Creative Commons

Open content as defined by Cedergren (2003) is content created for others to use, improve and redistribute, often in a collaborative environment. Also usually this content is produced with no commercial gain in perspective. Open content products have an open content license inspired by open source licenses. From a systems perspective open content benefits from virtual communities connected to the internet, easily available production tools, and easy content distribution.

There are lots of examples of collaborative content creation on the internet. Wikipedia is the most visible example, but innumerable other wikis, portals, forums attests to the widespread adoption of these ideas. Researchers are also begining to study these phe-

nomenon. Portuguese researchers conducted an experiment in on-line brainstorming and art creation to show the power of this kind of tools(Duarte, Costa, and Costa, 2008). Other sources of of studies about collaborative content creation include the analysis of a music creation community (Cheliotis, 2009).

2.4.1 Motivations for Open Content collaboration

According to Cedergren (2003) the main motivations for creating open content are the following:

- People find stimulating working together for a common goal.
- Learning and developing new skills.
- Possibility of feedback.
- Intrinsic motivation (passion for doing something).
- Altruism (giving to the community).
- Publicity.
- Indirect revenue.

Many of this motivations are in accord with the motivations found in open-source programmers and companies (Bonaccorsi and Rossi, 2006). This also brings in to focus an underlying motivation to be free and independent from the big corporations, that is present in open-source and also in open content communities.

2.4.2 What is Creative Commons

Creative Commons is an non-profit organization that promotes free accessible creative works for sharing and building upon. They created a series of licenses for creative works commonly known as Creative Commons Licenses. These licenses enforce the attribution of the works while enforcing also the ability to share them freely and allow for others to use them and remix them into their own creations (Lessig, 2005). While open source licenses apply to software, creative common licenses aim to be their equal for creative and artistic works. The creative common licenses offer a legal and copyright support for the open content creation and collaboration.

2.4.3 Commons based peer production

This term was coined by Benkler (2006) the describes a new mode of organizing production. A loosely connected and highly distributed network of individuals cooperate without

market signals or centralized management producing something by sharing resources and outputs. This mode is radically decentralized, collaborative and non-proprietary.

Reuse and mixing of aesthetic/cultural products has limited capabilities to generate new works, but according to Cheliotis and Yew (2009) can lead to doubling the output of the community production. He also concludes that content and mixing contest can be a form of attracting new authors to the community but in practice is more helpful to generate publicity for the products of the community.

While these communities are loosely connected and widely distributed, they favor the creation of strong bonds between a core of peers who steer the community and create most of the products.

2.4.4 Community of practice

Communities of Practice is another concept that can be helpful to understand these online communities. The term was coined in 1991 in a work by Lave and Wenger (1991) about learning and participation. Since then various authors have used it in different contexts and with different meanings (Cox, 2005).

For this analysis it is assumed that a Community of Practice is a group of people brought together by common interests and shared goals and try to achieve those goals by a process of social learning and community membership (Cheliotis and Yew, 2009). This definition of the term puts a focus on the membership and learning aspect of the community, and it seems to be a common thread both to open source communities and open content production communities.

Chapter 3

Conceptual Model

After studying the literature about game development, modding, open-source development and open content, the ideas are in place to design the conceptual model of the framework in a way that maximizes the appeal to the kind of person who would be a member of a game creation community.

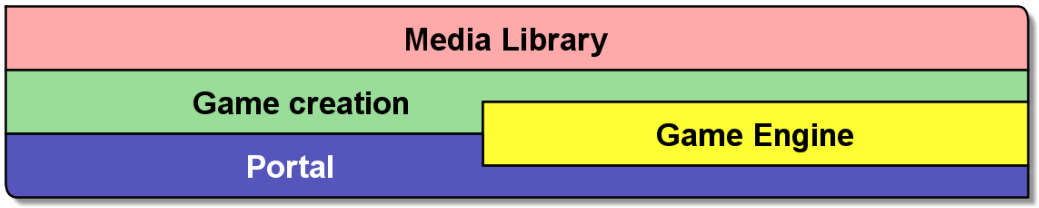

Figure 3.1: Framework basic components

3.1 Usage and roles

First the main roles needed for successful game development must be identified. The framework will have graphics designers, sound designers, game designers and game players. One member of the community can have one or more of this roles according to the kind of content he contributes and uses. Figure 3.2 shows the main use cases and actors for the framework.

Graphic designers authors of graphic content in the media library.

Game designer author and creator of games using the tools in the site and assets from the media library.

Sound designers authors of sound content in the media library.

Player players are community members who play and rate the games and assets in the media library.

The framework will have 4 main areas: the media library, the game creation tool, and the games portal interface (see Fig. 3.1).

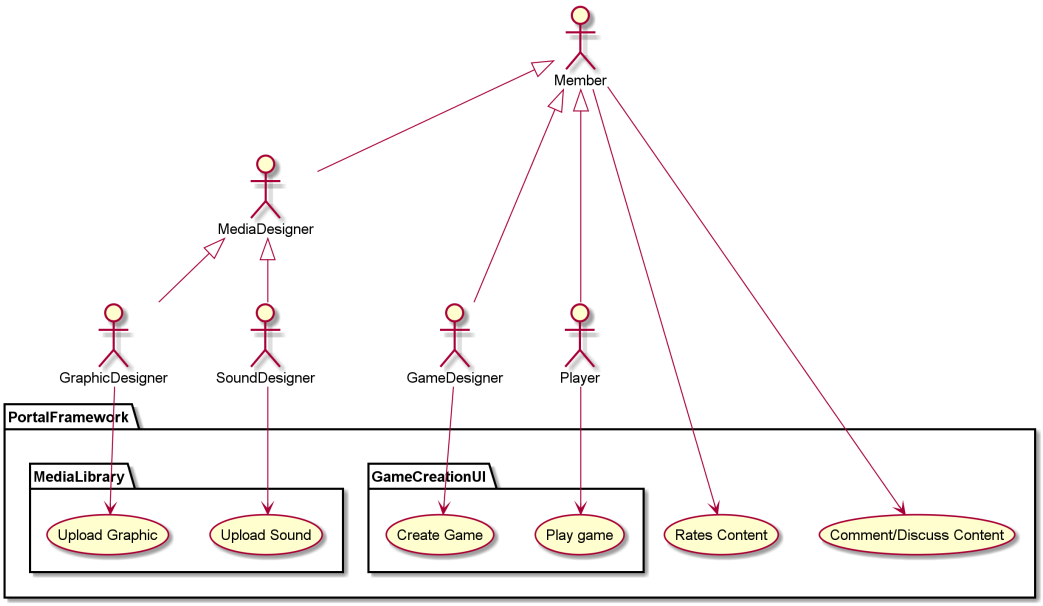

Figure 3.2: Framework main roles and use cases

3.2 Generic framework concepts

All areas of the framework will share some concepts designed to improve the community participation.

1. All pieces of content will have authorship attribution and all authors credited in a piece of content will also be credited in derivative pieces of content.
2. All content will be rated by the members of the community.
3. There will be a comments system associated with every piece of content and members will be able to comment and discuss content.
4. There will be rankings for content listing the highest rated pieces of content, the most viewed, and the most used (played in the case of games).
5. There will also be a ranking for authors based on the ratings and number of their creations.

6. Whenever possible there will be previews of the content available for easy identification.

7. There will be a system of awards for authors using the rankings, that will highlight the winner for a period of time in the community. For example an Author of the Month award every month.

8. All content in the framework will be licensed under a Creative Commons or Open Source license (whichever is more appropriate).

9. Whenever someone intends to create an improved version of a piece of content they should *fork* it and then make the changes to allow the original author to be credited. There will be no way to enforce this automatically but we expect authors to respect this and report any violations. There will have to be content managers that will arbitrate and enforce these rules.

10. There will be a forum associated with the framework with topics like: technical help, game discussion, framework discussion, etc.

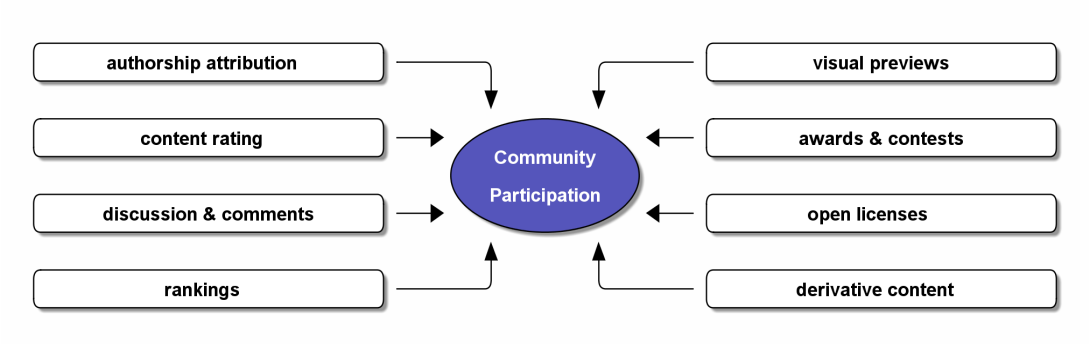

Figure 3.3: Collaborative Features

3.3 Media library

The media library will be primarily used by the graphics and sound designer to upload their content and review (and possibly reuse) other authors content. Game designers will also use the media library to browse the content and add it to their game.

All the content in the media library will be identified by type and tagged to allow for easy navigation and search. Also there will be the possibility of creating content packs grouping atomic pieces of content into sets related by any criteria defined by the authors.

The types of content available in the library will be:

Graphic files images suitable to the type of games allowed by the game engine (backgrounds, sprites, and other).

Sounds sound effects for game actions and background music.

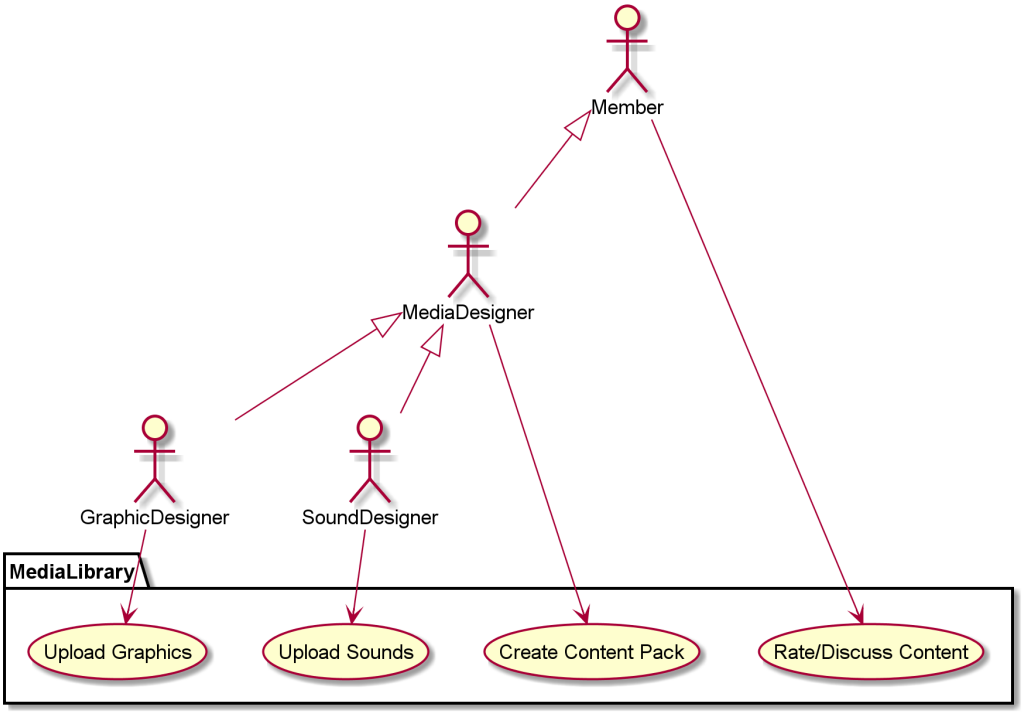

Figure 3.4: Media Library Use Cases

3.4 Game engine

The game engine (or engines) used in the framework assumes that the games can be created from a set of media objects (graphics, sounds), a set of rules and configuration values. From these inputs the game engine can create the game and allow users to play it. These generic characteristics of the game engine allow for a lot of diversity in the kinds of games they can power. Regarding the rules and configuration, the engine can be more generic allowing for greater configuration or more specialized embedding all the rules and most of the configuration values.

Since the engine is a modular part of the framework there can be eventually more than one. The only restriction is that not all types of media objects will be usable in all the engines and maybe some categorization would have to be done in the media library.

3.5 Game creation tool

This area will allow the game designers to use the assets in the media library and the HTML5/Javascript engine to create games that can be enjoyed by others members of the

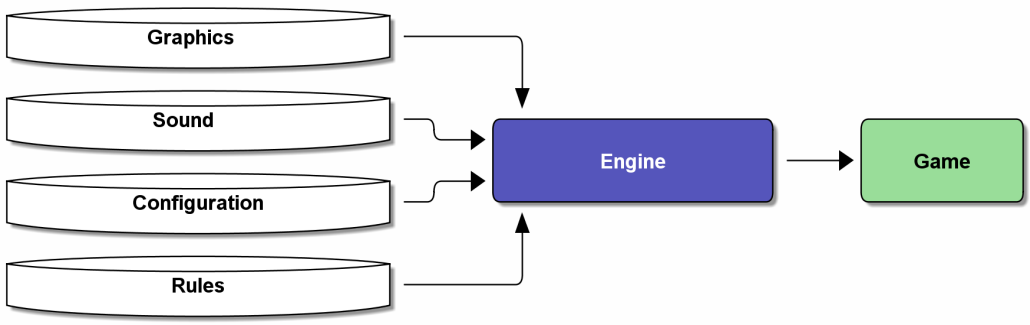

Figure 3.5: Conceptual game engine

community. The authors of all assets used on the game will be credited in the game in their respective role.

The main tasks that can be performed in this area are the following:

- Select and or build the game graphical components using graphics from the library.
- Select and assign the sound components of the game from the library.
- Define the game rules like winning and losing conditions and other rules dependent on the engine.
- Edit the engine configuration values for the specific game to be created.
- Edit game information like title, description, story, screen-shots and other information that will be used to promote the game in the portal.

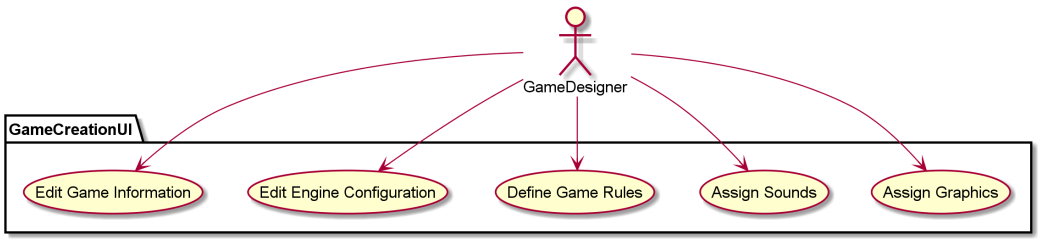

Figure 3.6: Game Creation Use Cases

Besides the standard game creating process described in Figure 3.7 authors will be able to *fork* an existing game and edit all aspects of the game creating a new derivative game (that will credit the original author also).

Games will also have a tagging system to allow game categorization and easier browsing and search as discussed in section 3.2.

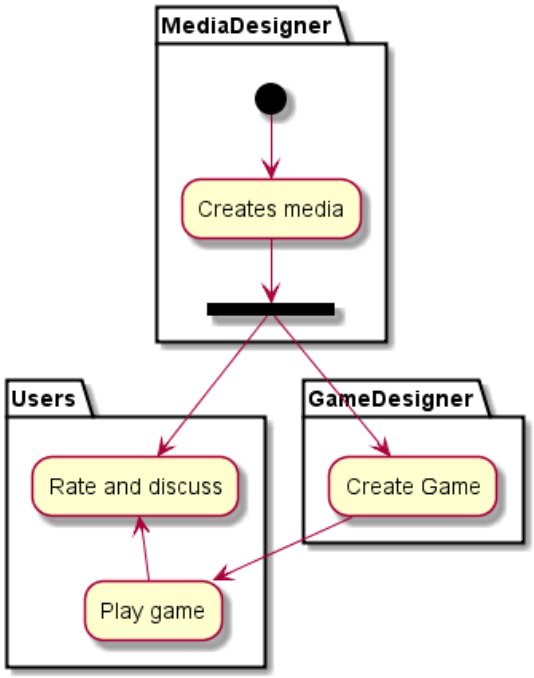

Figure 3.7: Game Creation Process

3.6 Game portal interface

This will be the more publicly visible part of the framework. Members of the community and anonymous users will be able to see and play the games. Here we'll be able to browse and search the games, play them and rate them (only for members).

The games will be displayed using the information provided by the author with focus on visual information (screen-shots). Games with high play rates and or high ratings will be highlighted in the front page were the game and authors rankings will also appear.

A comment system will be associated with the games allowing for users to give feedback to the game creators.

The portal will also have a section for articles (tutorials, technical and others) written by members of the community and other invited writers. This will reinforce the learning focus of the community and allow more senior members to share their experiences with newer ones.

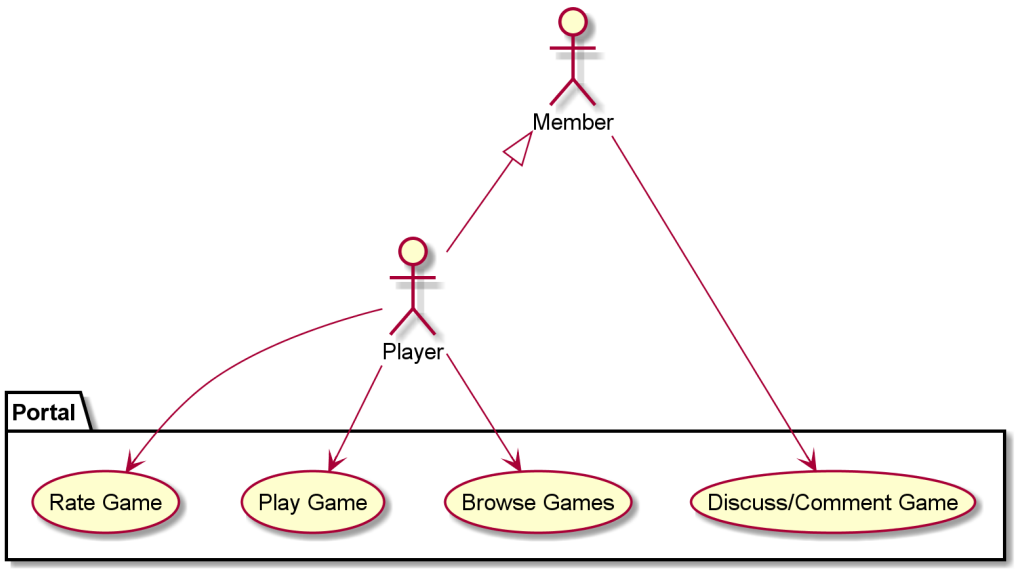

Figure 3.8: Portal Use Cases

Chapter 4

Prototype Implementation

4.1 Overview

The work included in this thesis is only the first step in the creation and testing of the collaborative games creation framework. This is the groundwork needed to start with the building of a prototype web application that will flesh out these ideas and give us some metrics on the applicability of these ideas and their success in the field.

The prototype described here implements the conceptual model described previously using a simple maze game engine integrated into a content management system configured as a game playing and media library portal.

Figure 4.1: Prototype Portal Home

The games supported by the prototype have a graphical maze where the player character will have to move through the maze without touching any of the walls and find the maze exit. The character will have a finite amount of life energy which will be depleted

each time he touches a wall. If the energy reaches zero the game will be lost. The game is controlled by the keyboard.

In this implementation there is no sound in the games and therefore no sound media library.

4.2 Tools And Technologies

4.2.1 Drupal

Drupal is a free and open source content management system implemented in PHP and MySQL, created by Dries Buytaert in 2001. The software is used in many sites around the Internet and is mature and stable (Reid, Tomlinson, and VanDyk, 2010; Townsend, 2010). In the prototype version 7 of Drupal (7.12) was used.

Drupal was chosen because it has several features that simplify the creation of community and user oriented applications. Its core modules support user registration and management, a role based permissions system, blogs and forums. Using standard core tools the media library could also be created easily.

Because of its modularity Drupal has a very active set of third party modules ready to integrate into the application and provide extended features. The rating system, presentation and linking of data needed for this project were all implemented using external drupal modules.

The interface customization of Drupal projects via modular *themes* was also taken advantage of allowing to integrate the HTML5/Javascript game engine in a simple and clean way.

- Drush
  Drush (Townsend, 2010, p. 263) is a tool implemented in PHP which allow the management and configuration of Drupal installations via a command line interface, streamlining the installation of modules, user management, database and site backups, updates and other administrative tasks. This allows the installation and configuration of Drupal to be scripted and replicated in different servers.

  It was very useful in the implementation of the prototype allowing the synchronization of different development versions and the deployment of new features on the production prototype.

- GIT
  Drupal development is managed with GIT (Swicegood, 2008) as a distributed version control system. This allowed to track the changes and history of the development and was taken advantage of in the multiple site development of the prototype, allowing code updates to be propagated between different installations.

4.2.2 HTML5 (Canvas)

For the browser game engine development the HTML5 canvas implementation was used for graphics and animation rendering. The 2D API of the canvas element is powerful enough in current generation browsers for game development (Hawkes, 2011). In the prototype the simplified game engine used wasn't difficult to implement.

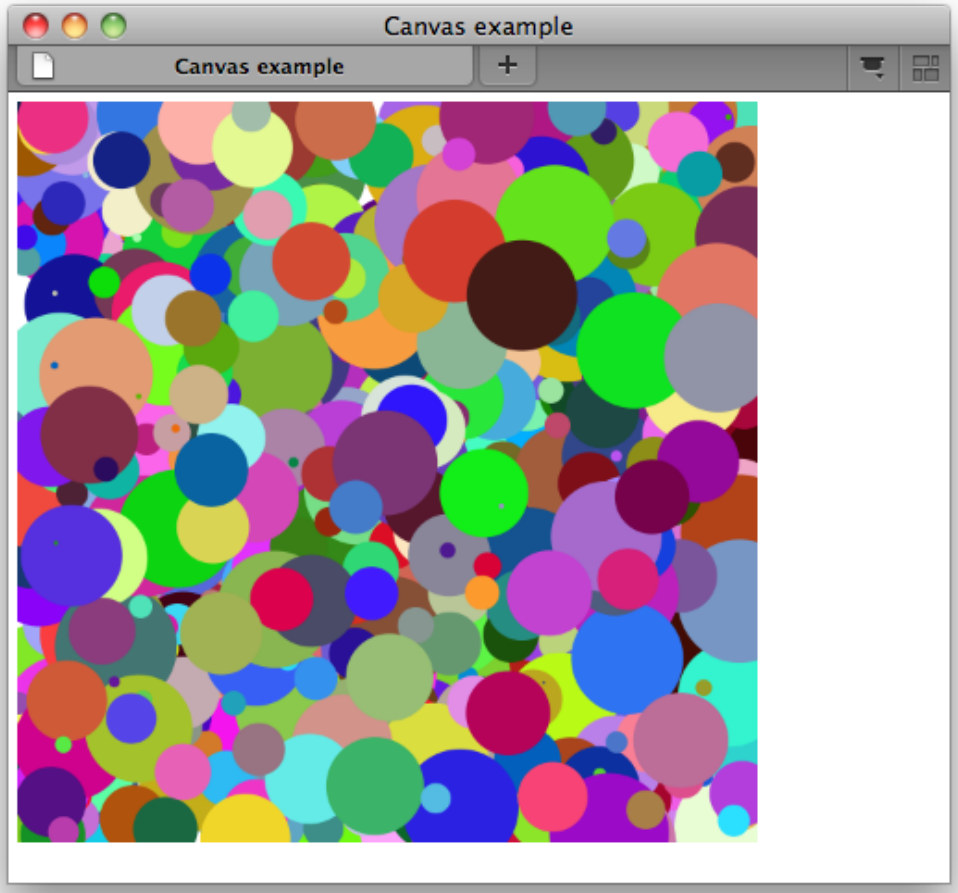

Figure 4.2: Canvas Example

The canvas 2D API being similar to other raster based API's allows the use of standard game graphics algorithms, and mature and tested game engine approaches.

4.2.3 Javascript

For dynamic DOM manipulation in the game engine, and input processing in the browser environment Javascript is the only option (Flanagan, 2006). As such the possible choice was between using pure Javascript or one of the several libraries and frameworks available for interacting with the browser and DOM API.

JQuery was adopted because it is a mature, well documented and widely used framework (Chaffer and Swedberg, 2009), and also because it is the default library in Drupal, which allowed for easier integration into the prototype development.

4.3 Games Portal Prototype

4.3.1 Drupal content types

The ability to create and configure new content types in Drupal was used extensively to create the main components of the media library and the games configuration interface.

Drupal allows the creation of new types of content by combining various predefined field types (see Table 4.1). Additional field types can be added by third party modules.

In Drupal the display of the content type nodes can be managed by context. For example it can display different fields when displaying the full node content in a page and when displaying a list of nodes in a summary or search page. Drupal introduces the concept of view modes to differentiate the contexts where the content will be shown.

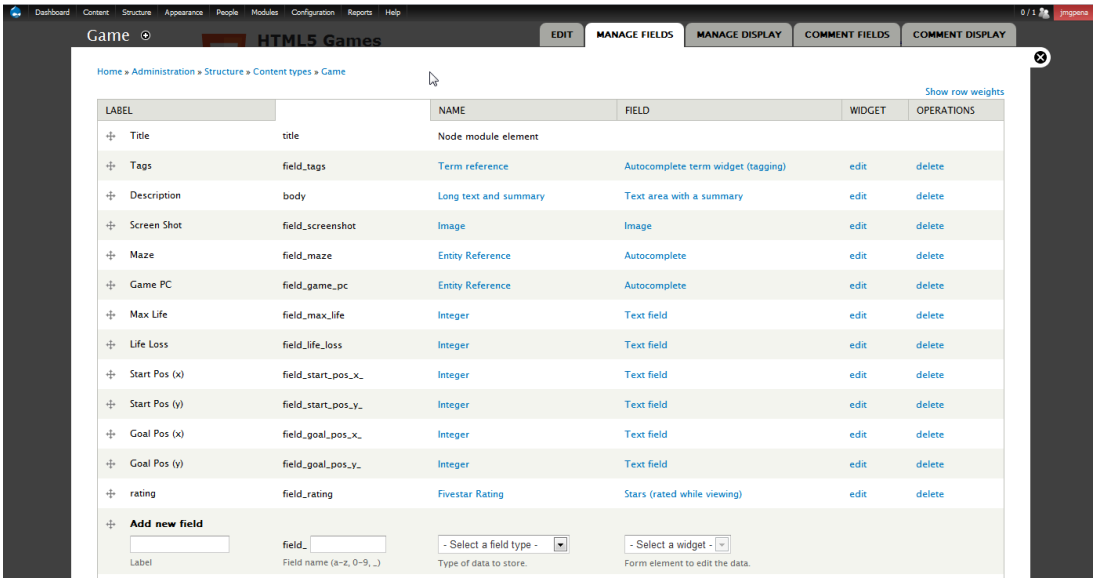

Figure 4.3: Content type administration in Drupal

Furthermore it also attaches CSS classes to the HTML elements displayed on a page to allow even more customization in the Drupal theme used.

4.3.2 Core modules

The standard profile installation of Drupal comes with some core modules already enabled while others are left for the site creator to choose. These core modules are part of Drupal and are maintained by the Drupal core team.

Table 4.1: Drupal predefined field types

Field type	**Description**
Boolean	Boolean values
Integer	Integer values
Decimal	Decimal values
Float	Float values
File	Upload different file types
Image	Upload images
List	Present the user with a list of options
Text	Text content
Long Text	Longer text content
Term Reference	Reference taxonomy terms

The core modules include the standard content types *Article* and *Basic Page*. *Articles* can represent news and other time sensitive items, while *Basic Pages* can be used for static and information content. The comment system for content nodes also is a part of a default Drupal installation.

Other modules from core are not enabled by default but some of them provided needed features for the collaborative games creation portal. The most revelant are the following:

- Forums

  The *Forums* module enables a new content type called *Forum Topic* and a framework for the organization of these nodes into a hierarchy that provides containers and forums for topic organization. This module was enabled on the prototype because the discussion forum is a component of the framework.

  This part of the prototype is self-contained and does not interact directly with other modules and content types. Further on the forums configuration used on the prototype will be explained in detail.

- Menu

  This module allows the site administrators to create and change the navigation menus. Menus can be created as hierarchies of links, and can be placed in various places. Permissions can also be configured giving access to some menu items to certain roles and excluding others. This provides a very flexible way to create the site navigation and provides custom menus and links to specific user roles.

- Taxonomy

  The taxonomy module in Drupal implements the categorization of content using vocabularies and terms. Drupal can create vocabularies of terms and organize the terms using relationships between them. Terms can be nested and this relationships are used to find content. If there is a need to search content categorized with

a specific term, then also content categorized with its children terms would be returned. In the prototype this feature is used to assign tags to content and later those tags (created freely by the users) can be organized and relationships created to help in content organization.

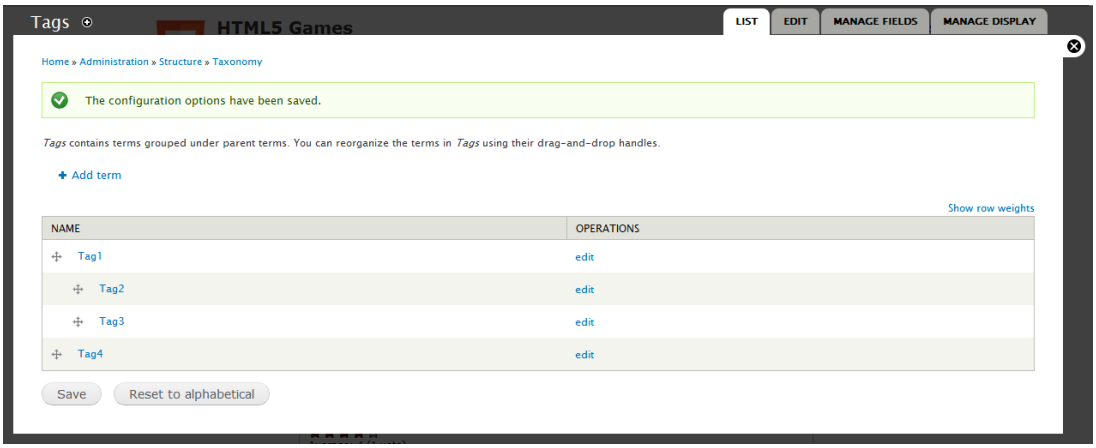

Figure 4.4: Taxonomy organization interface

4.3.3 Third party modules

- Display Suite
  The *Display Suite* module provides the means to control how content is displayed in a Drupal site. The standard Drupal content type display configuration does not give access to all content fields and only allows to hide or show the fields and change their order.

  Drupal allows the site administrator to define how content should be displayed in different contexts (view modes): full page, teaser, rss feeds, etc. The *Display Suite* modules adds a layer on top of this feature greatly enhancing the standard functionality.

  With this module, the display of a node content can be configured in a drag and drop interface and allow much more flexibility in the choice of fields and layouts. Besides the standard layouts included it allows to easily create new layouts if you are using a custom theme. This facility was taken advantage of while creating the game nodes display.

- Entity Reference
  The *Entity Reference* module allows fields to reference content nodes in a Drupal installation. It implements the generic node reference field and the widgets to choose the nodes in the content editing. It also provides formatters for showing the referenced content in nodes, as a label or a "rendered entity".

This functionality was very useful because it allowed to reference the graphics objects needed for a game in the game content editing interface in a natural way. It also allowed to easily show the graphics content when displaying the game content and to pass the information to the javascript game engine.

- Fivestar
  The *Fivestar* module uses a voting API module and a special field to allow the adding of a rating system to content. The module is based on the concept of user rating made popular by Amazon and other online shops using a five star rating system. The field can be configured to specify if anonymous users can vote, if the same user can vote more than once, and other characteristics. The displaying of the rating can also be controlled and can show the current rating based on the sum of all the individual ratings and relevant info like the number of votes.

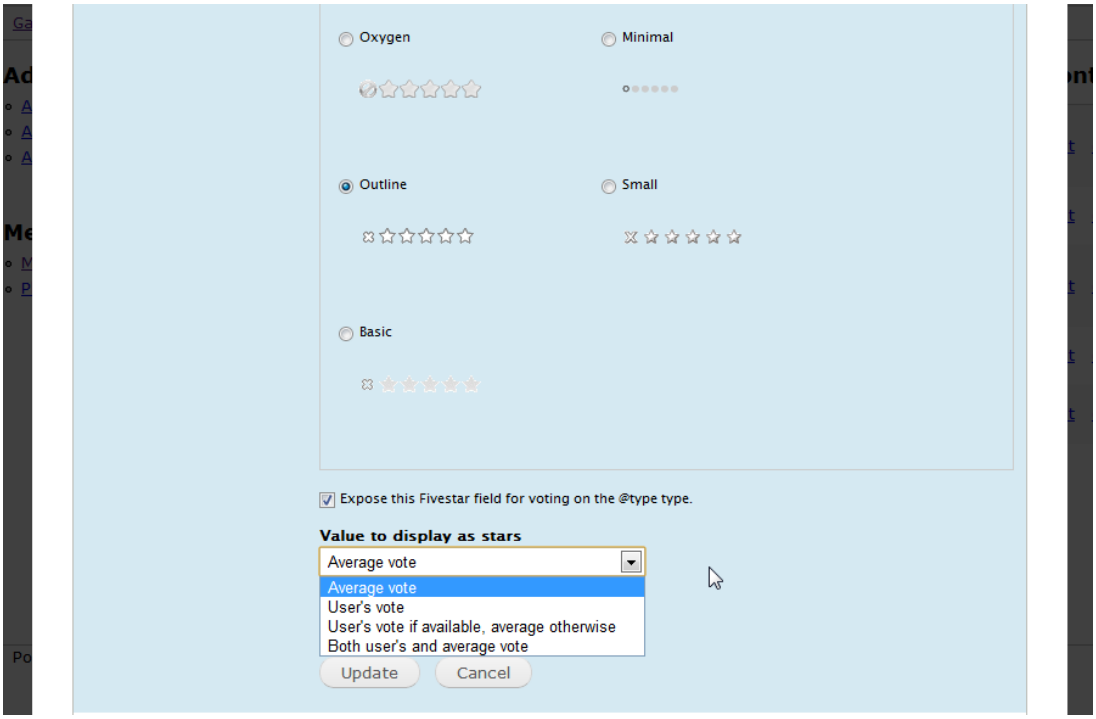

Figure 4.5: Fivestar voting field configuration

- Views
  The *Views* module provides functionality to create and show customized lists and tables of content and other data, and control how it is presented. It allows the creation of queries over content and site data and display the results in a flexible way. It can list the results in blocks and pages. *Views* can generate reports, summaries and display sets of images and other content.

  In the prototype implementation it is used to provide ordered lists of relevant content like games, graphics, users, etc. It allows great flexibility in the query definitions and in formatting the display of the results.

4.3.4 Forum

Drupal comes with a forum module in its core modules so this feature was somewhat easy to implement. The forum was configured to have different areas for games, technical and general discussions. Eventually more areas and divisions can be created.

The forum module in Drupal allow the organization of content into *forums* which contain discussion topics (the actual messages), and the grouping of the forums in *containers*. Both *forums* and *containers* can be organized in a tree like structure where a *forum* can have child forums and containers.

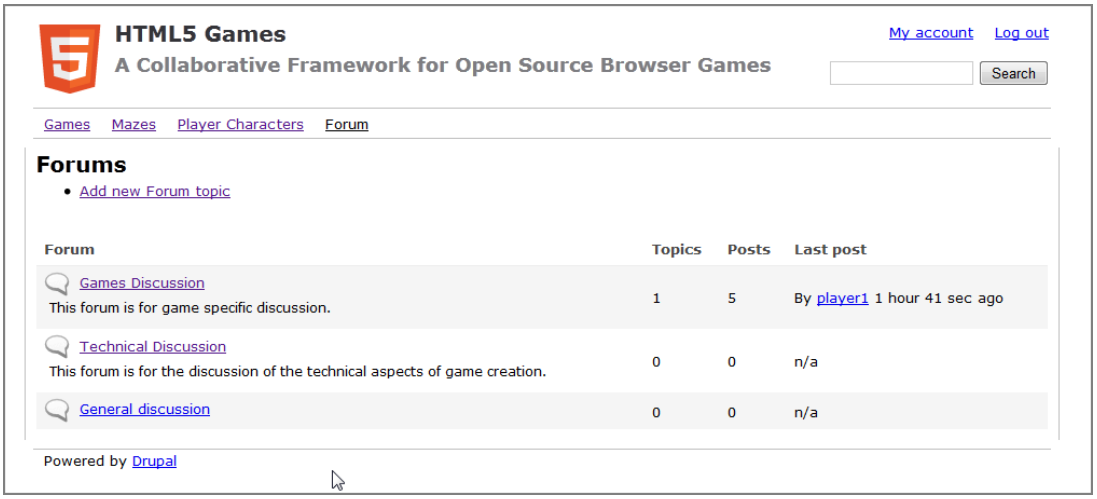

Figure 4.6: Basic forum organization

4.3.5 Media Library

The media library in the prototype only has two types of media: *Mazes* and *Player Characters*. These are the graphical components needed for the game engine being used. To build the media library two new Drupal content types were created matching the two types of graphics needed.

- Maze content type
  The maze content type contains three fields: *Title*, *Description* and *Maze*. The *Title* contains a name to identify the maze (given by its author), the *Description* contains information deemed important for the maze and the *Maze* field contains the image file of the actual maze.

  The maze has a maximum size limit of 640x480 pixels and a minimum size of 320x200 pixels. The image must be in *png* format with the walkable area of the maze as the transparent color.

- Player Character content type
  This content type is very similar to the above but has a smaller image field (named

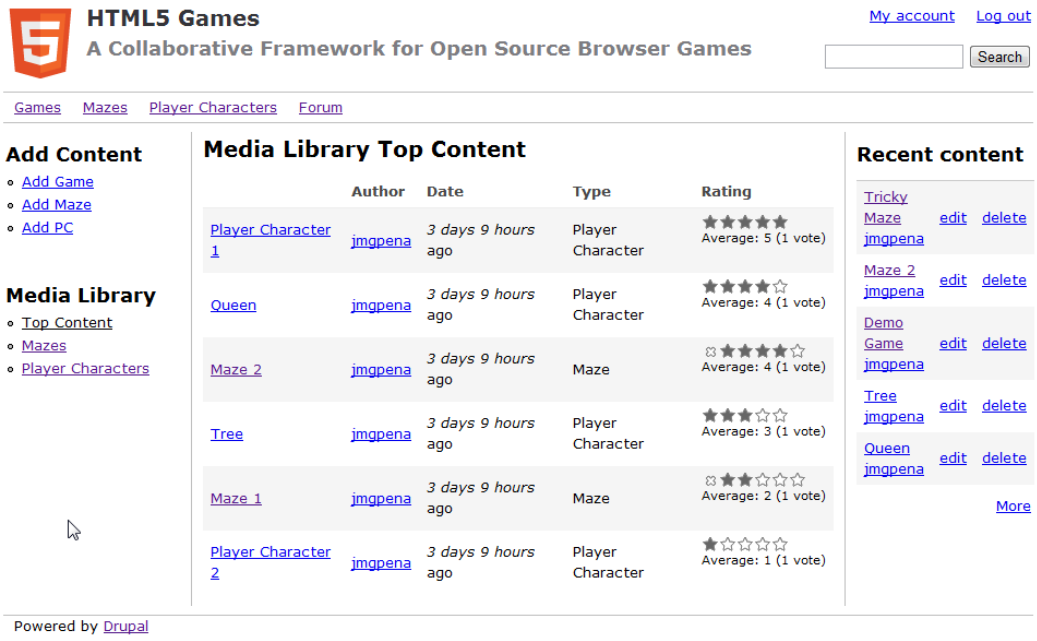

Figure 4.7: Media library top content

PC) with a maximum size of 128x128 pixels and a minimum size of 32x32 pixels. The type of the image must also be *png* and the *player character* can have complex forms because the transparent areas of the image are ignored in the collision detection algorithm.

4.3.6 Game content type

For describing the game rules and components a new content type named *Game* was created. The fields in this content type (Table 4.2) define the name of the game and describe the game parameters and the graphical components used.

The game content type also ties in with the custom theme using the *Display Suite* module discussed above to include custom HTML, CSS and the game engine Javascript when the game node is displayed. Using this technique it is possible to completely control the display of the fields in this content type and add a button (link) to play the game.

When the *play game* link is followed the game configuration is passed to the game engine that controls fully the display and playing of the game. The game playing interface creates an overlay over the current page and creates manipulates the DOM adding and removing dynamically the HTML elements needed for the game interface.

The separation between the game configuration and editing and the game engine provides us with the flexibility to use this same framework to play different kinds of games by creating different engines and linking them to different games content types.

Table 4.2: Game content type fields

Field	Type	Notes
Title	Text	Name of the game
Screenshot	Image	Game screenshot
Description	Long Text	Description of the game
Maze	Entity Reference	Maze graphics to use
PC	Entity Reference	PC graphics to use
Max Life	Integer	PC energy at start of game
Life Loss	Integer	Amount of energy lost on collision with maze walls
Start Pos (x)	Integer	PC starting x position in the maze
Start Pos (y)	Integer	PC starting y position in the maze
Goal Pos (x)	Integer	PC goal x position
Goal Pos (y)	Integer	PC goal y position
Rating	Fivestar Rating	Game rating
Tags	Term Reference	Content tagging

4.4 Game engine

4.4.1 Overview

The game engine was created using Javascript for DOM manipulation (via JQuery) and the HTML canvas element for the actual game graphics rendering.

The game interface is implemented as a JQuery plugin called *modalPanel*. It creates a layer above the current page (hidding the page content) and adds the HTML `<canvas>` element needed for the game rendering.

The game engine is implemented as a Javascript object prototype with the game methods and data. After the *modalPanel* plugin is initialized it calls the game object method *run*. The game engine can be broken in to several main components described bellow.

4.4.2 User Interface initialization

This component of the game engine is very simple and just adds to the DOM the dynamic user interface elements needed to display game information. In this simple maze game there is only need to display the life energy of the character and optionally the current position in the maze.

4.4.3 Assets loading

This part of the engine deals with the loading of the maze and player character (PC) images from the web server. In this case and because of the way the browsers implement

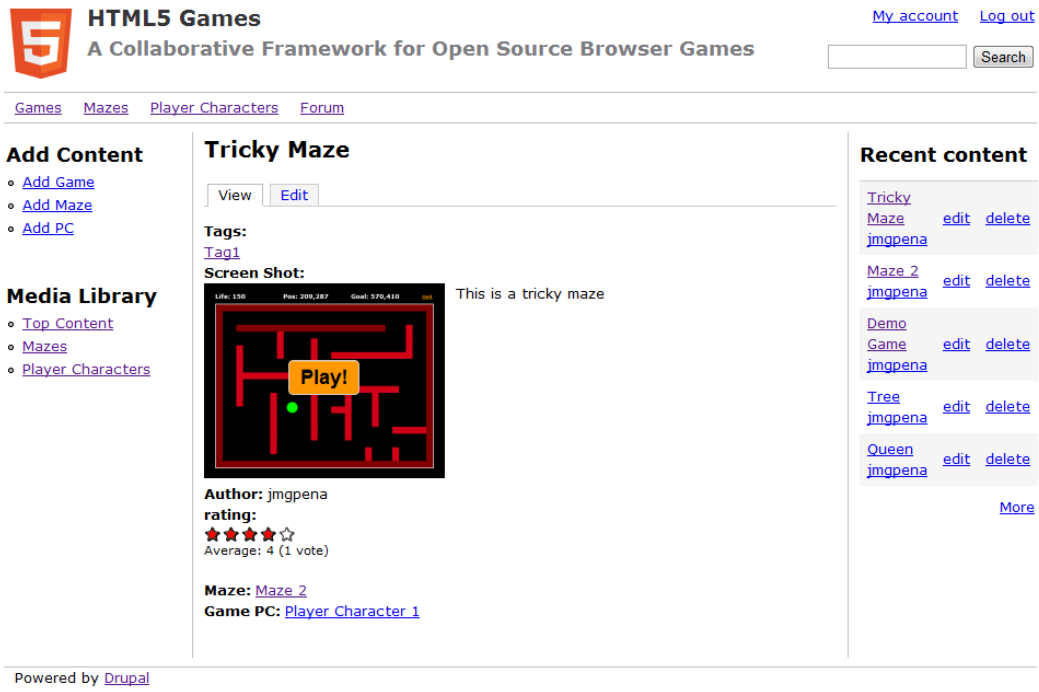

Figure 4.8: Game node view with play button

image caching the code must assure the images are loaded to be possible to render them into the canvas element.

Some browsers implementations don't fire the *onLoad* event for images if they already are on the cache in certain conditions making difficult to access if they are loaded or not. The technique used only adds the image URL to the image object after the attaching of the event assuring the event is triggered in all situations.

Before the assets can be used (images in this case) they must all be loaded, so a scheme was devised to decrement a counter for each of the needed assets and only calling the viewport initialization method when the counter reaches zero.

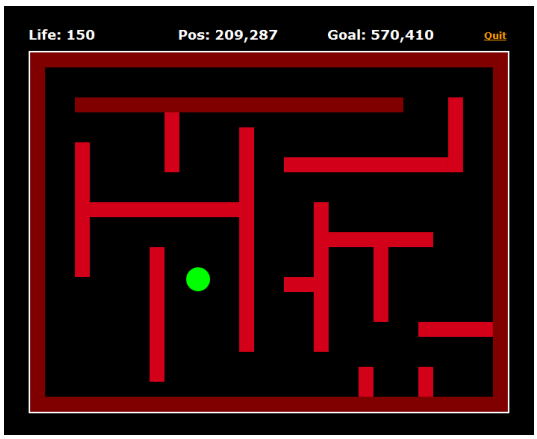

Figure 4.9: Game playing interface

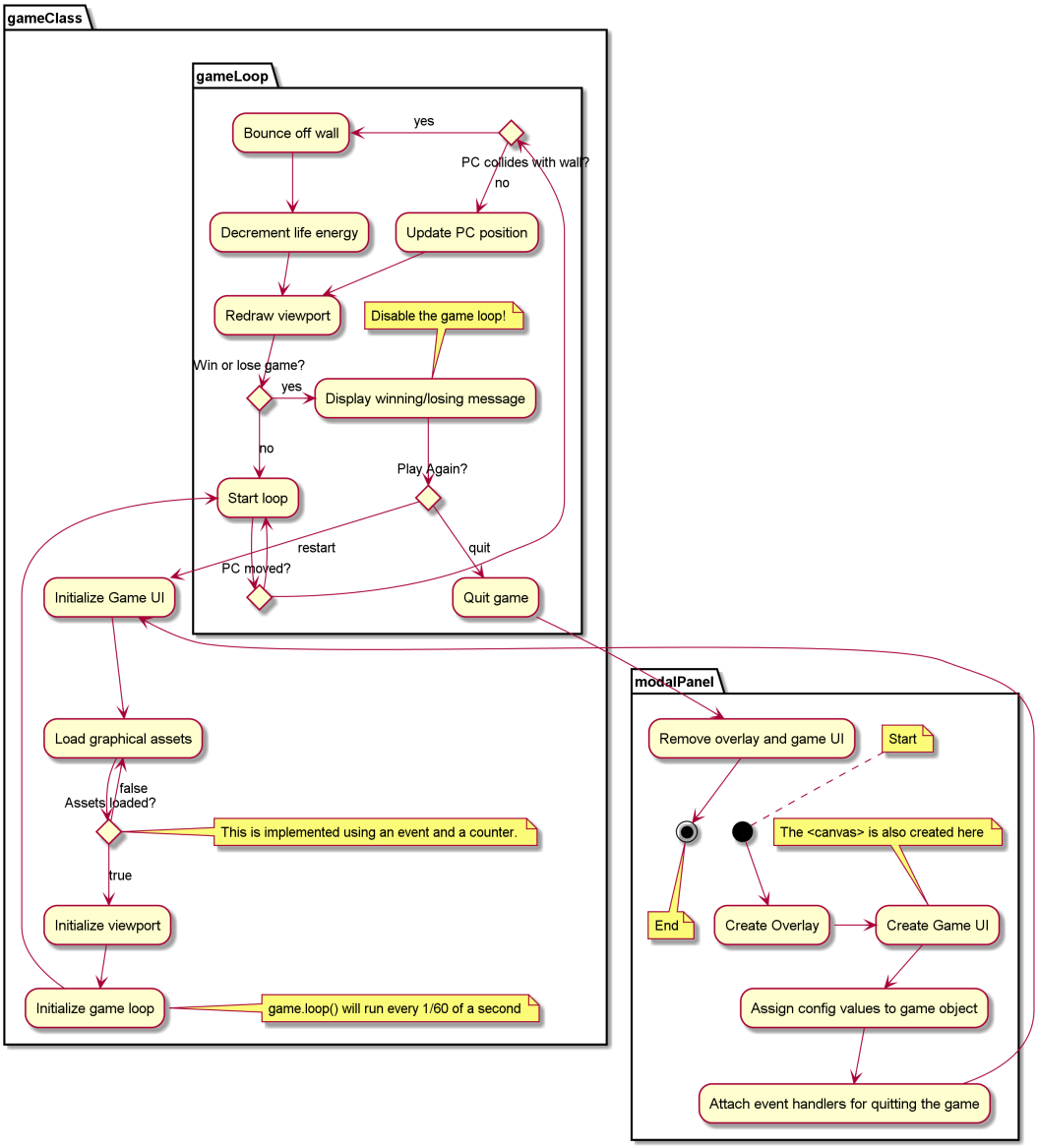

Figure 4.10: Game engine activity diagram

4.4.4 Viewport initialization

In the *initVP* method of the game engine the assets (graphics) loaded before are prepared and rendered into the HTML canvas element.

The player character object (*pc*) is initialized with its graphic image and all the needed variables.

The *drawpc* method is attached to the *viewport* object and will be called when necessary in the main game loop. This method redraws the maze and the PC every time there is movement.

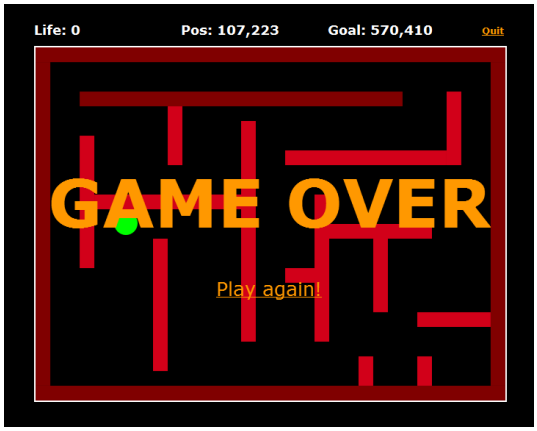

Figure 4.11: Game over

4.4.5 Game loop

The *loop* method is scheduled to run every 1/60th of a second and implements the main game logic. It checks for PC movement, redraws the canvas if necessary and also checks for the winning and losing conditions of the game. If the game ended it stops the scheduling and adds an interface element to allow the restart of the game. The game is won if the PC is over the goal position and lost if the life energy reaches zero (see Listing 1).

The movement detection method (*movepc*) checks if there are any keys pressed and moves the PC accordingly. It calls the collision detection method (described bellow) and moves the PC back to the original position if the movement resulted in collision. In the case of collision this method also decreases the life energy of the PC. When a collision happens the buffer holding the keys pressed is cleared to stop the movement even if the player does note release the keys immediately.

4.4.6 Collision detection

The collision detection method (Listing 2)uses the pixel image data from both the maze and the PC to detect if there is a non transparent pixel from both images in the same canvas position. This method implies the scanning of a whole rectangle of both images, and as such the rectangle searched is limited to the size of the PC which is smaller than the maze. The algorithm could be optimized by selecting a smaller rectangle depending on the shape of the PC and the movement direction, but the scanning of the whole PC image is fast enough for an accurate collision detection in the engine.

```
game.loop = function() {
    game.movepc();
    if (game.redraw) {

        // losing condition
        if (game.pc.life <= 0) {
            game.pc.life = 0;
            game.lost = true;
        }

        // winning condition:
        if ((game.goal_pos_x >= game.pc.x && game.goal_pos_x <= (game.pc.x + game.pc.w)) &&
            (game.goal_pos_y >= game.pc.y && game.goal_pos_y <= (game.pc.y + game.pc.h))) {
            game.won = true;
        }

        game.viewport.drawpc(game.pc);
        game.display('life','Life: '+game.pc.life);
        game.redraw = false;
    }
    if (game.lost) {
        clearInterval(game.int);
        $("#modal-overlay").append("<h1 class='bigmessage'>GAME OVER</h1>");
    }
    if (game.won) {
        clearInterval(game.int);
        $("#modal-overlay").append("<h1 class='bigmessage'>YOU WON!!!</h1>");
    }

    if (game.won || game.lost) {
        $("#modal-overlay")
            .append("<a class='again' href='#'>Play again!</a>")
            .click(function (ev) {
                $(".bigmessage").remove();
                $("a.again").remove();
                ev.preventDefault();
                game.pc.life = game.max_life;
                game.run();
            })
    }
}
```

Listing 1: Game loop method.

```
game.testCollision = function(x,y) {
    // this.pc.w, this.pc.h, this.pc.image
    var xMin = x, yMin = y
    , xMax = x+this.pc.w, yMax = y+this.pc.h
    , w = this.mapW, w2 = this.pc.w
    , pixels1 = this.mapData
    , pixels2 = this.pc.imgData;

    // loop
    for (var pixelX = xMin; pixelX < xMax; pixelX++)
        for (var pixelY = yMin; pixelY < yMax; pixelY++) {
            var px1 = ((pixelX) + (pixelY) * w)*4 + 3;
            // pixel in maze relative to x,y
            var px2 = ((pixelX-x) + (pixelY-y) * w2)*4 + 3;
            // pixel in PC relative to 0,0
            if ( pixels1[px1] !== 0 && pixels2[px2] !== 0) {
                return true;
            }
        }

    return false;
}
```

Listing 2: Collision Detection Method.

Chapter 5

Results Analysis and Discussion

5.1 Conceptual Framework

The main goal of this work was to create a conceptual framework for the collaborative creation of browser games using only Open Source technologies. The conceptual framework devised is based on a solid foundation of concepts taken from successful collaborative work in several areas.

The first positive result of this research is that the conceptual framework can be implemented in the real world using Open Source tools and technologies. The prototype developed does not implement fully all the framework concepts but it can assumed that with sufficient time and effort those concepts could be implemented in the prototype.

5.2 Open Source Browser Games

The creation of browser games using only Open Source technologies is a reality and even though the prototype created in this research is a very simple, it allows the creation of games in a collaborative environment. There can be different people contributing to the creation of the games in a completely open environment.

Other aspect relevant to this research was the concept of collaborative game design and creation for artists and non programmers. The prototype developed shows that this is possible for a simple game engine, but it can be safely assumed that it will work for more complex games and that with sufficient time and effort the technical barriers can be removed and more complex and interesting games implemented.

Even in the context of the maze game engine, the creation of the mazes could be enhanced with tile based mazes, the addition of NPC's (Non Player Characters), sound effects, background music, among other ideas.

5.3 Prototype Limitations

The prototype developed in the context of this dissertation has a few limitations regarding the features discussed in the conceptual framework. Some things were left out because they would take more time to implement, and others could not be easily implemented using the chosen technologies.

5.3.1 Content *Forking*

In the conceptual framework description one of the features mentioned is the ability to reuse content from other author by making a copy and from there making changes and adding or removing something to create a new version. To implement this kind of tracking of changes and trace a piece of content back to the original author, the Drupal content management back-end would have to be substantially changed and it would be big effort. This feature was found to be most technically challenging to implement in a real work content management system.

5.3.2 Awards and contests

This feature can be easily implemented but needs a minimal community already in place. Then it would require more of a community management process than a technological approach. This could be easily implemented using the current prototype implementation.

5.3.3 Sound content

The addition of sound content to the media library and the game engine is feasible and can be implemented in time. This feature was left out by lack of time and the necessity to have a working prototype to deliver with the dissertation.

Chapter 6

Conclusion

Browser games use mostly closed technologies. It is possible to create good quality games using Open Source tools and open collaborative frameworks. Using open licenses both for code and art resources on games allows the creation of a pool of resources that enriches all futures games based on this technology. This approach also allows creators with good game ideas but poor artistic skills to engage in game creation. The opposite is also true, because an artist can take a good game with poor graphics and make a version with better ones.

The ability to remix content to make new things allowed recent revolutions in areas like music, video and software development. Games make use of these things but not in an integrated manner, and tools to allow this can be very useful to make it accessible to more people.

The reviewed literature allowed the creation of a conceptual model of the framework that incorporated the most important ideas of how on-line collaborative efforts work and create successful communities. Also the study of the methodologies for software and game development guided the division of roles and tasks between the various parts of the framework.

Finally the framework was implemented in a prototype on-line game creation portal, showing that the approach proposed is not only feasible but also shows promise for future developments. The most important features of the framework were implemented and the prototype will be open to external users to access if the conceptual framework ideas for community enabling are also sound.

Bibliography

Benkler, Y. (2006). *The wealth of networks: How social production transforms markets and freedom*. Yale Univ Pr.

Bethke, E. (2003). *Game development and production*. Wordware Publishing. ISBN: 1556229518.

Bonaccorsi, A. and C. Rossi (2006). "Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business". In: *Knowledge, Technology & Policy* 18.4, pp. 40–64.

Cedergren, Magnus (Aug. 2003). "Open Content and Value Creation". In: *First Monday* Volume 8.Number 8. URL: `http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/1071/991`.

Chaffer, J. and K. Swedberg (2009). *Learning jQuery*. Packt Publ.

Cheliotis, G. (2009). "From open source to open content: Organization, licensing and decision processes in open cultural production". In: *Decision Support Systems* 47.3, 229–244. ISSN: 0167-9236.

Cheliotis, G. and J. Yew (2009). "An analysis of the social structure of remix culture". In: *Proceedings of the fourth international conference on Communities and technologies*, 165–174.

Costa, C.J. and M. Aparício (2006a). "Computer Game–Discussing Development Process". In: *Proceedings of IRIS*. Vol. 29. Citeseer.

— (2006b). "Computer Games Development Process–Producing an Education Games for The Web IADIS conference WWW/Internet 2006 Múrcia, Espanha". In: *Outubro*, pp. 5–8.

Cox, A. (2005). "What are communities of practice? A comparative review of four seminal works". In: *Journal of Information Science* 31.6, p. 527.

Duarte, P., C. J Costa, and P. Costa (2008). "Webstorm: mixing brainstorming in the web to produce art". In: *Proceedings of the 26th annual ACM international conference on Design of communication*, 267–268.

Elliott, M. and W. Scacchi (2004). "Mobilization of software developers: The free software movement". In: *Information, Technology and People*.

Flanagan, D. (2006). *JavaScript: the definitive guide*. O'Reilly Media.

Flew, T. and S. Humphreys (2005). "Games: technology, industry, culture". In:

Hawkes, R. (2011). *Foundation HTML5 Canvas: For Games and Entertainment*. friends of ED.

Järvinen, P. (2004). "Research Questions Guiding Selection of an Appropriate Research Method". In: *Department of Computer Science, University of Tampere, Finland* 9.

Jones, R.M. (2000). "Design and implementation of computer games: a capstone course for undergraduate computer science education". In: *ACM SIGCSE Bulletin* 32.1, p. 264. ISSN: 0097-8418.

Kücklich, J. (2005). "Precarious playbour: Modders and the digital games industry". In: *Fibreculture* 5.

Lave, J. and E. Wenger (1991). *Situated learning: Legitimate peripheral participation*. Cambridge university press.

Lessig, L. (2005). *Free culture: The nature and future of creativity*. Penguin Group USA. ISBN: 0143034650.

Pine, B.J. and S. Davis (1993). *Mass customization*. Harvard Business School Press.

Portes, Alejandro (1998). "Social capital: Its origins and application in modern sociology". In: *Annual Review of Sociology* 24, 1–24.

Reid, D., T. Tomlinson, and J. VanDyk (2010). *Pro Drupal 7 Development*. Springer.

Scacchi, W. (2004). "Free and open source development practices in the game community". In: *Software, IEEE* 21.1, pp. 59–66. ISSN: 0740-7459. DOI: 10.1109/MS.2004.1259221.

Scacchi, Walt (May 2010). "Computer game mods, modders, modding, and the mod scene". In: *First Monday* 15.5. URL: http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/2965/2526.

Sotamaa, O. (2007). "On modder labour, commodification of play, and mod competitions". In: *First Monday* 12.9.

Swicegood, T. (2008). *Pragmatic version control using Git*. Pragmatic Bookshelf.

Townsend, R.J. (2010). *Foundation Drupal 7: Learn how to Use the Drupal Framework to Quickly Build Feature-rich Web Sites*. friends of ED.

Tran, M. Q and R. Biddle (2009). "An ethnographic study of collaboration in a game development team". In: *Loading...* 3.5. ISSN: 1923-2691.

Appendix A

Game Engine Source Code

Here the most relevant source files to understand the game engine will be presented.

A.1 game.js

```
// by Jorge Pena <jorge@jmgpena.net>

(function($) {
    // fix warnings in webkit browsers (chrome)
    $.event.props = $.event.props.join('|')
        .replace('layerX|layerY|', '').split('|');

    // game object definition
    var game = new Object();

    game.keysDown = {};

    $(document).keydown(function(event) {
        game.keysDown[event.keyCode] = true;
        //event.preventDefault();
    });
    $(document).keyup(function(event) {
        delete game.keysDown[event.keyCode]
        //event.preventDefault();
    });

    game.loadAssets = function() {
        var that = this;

        this.assets = new Object();
        this.assets.countLoaded = function() {
            this.count--;
            if (this.count == 0) { game.initVP(); };
        }
        this.assets.count = 2;
        this.assets.pc = new Image();
        //$(this.assets.pc).one('load',this.assets.countLoaded());
        $(this.assets.pc).one('load', function() {
            game.assets.pc = this;
            game.assets.countLoaded();
```

```
36          })
37          this.assets.pc.src = this.pc_url;
38          this.assets.maze = new Image();
39          $(this.assets.maze).one('load', function() {
40              game.assets.maze = this;
41              game.assets.countLoaded();
42          })
43          this.assets.maze.src = this.maze_url;
44      }
45
46      game.initUI = function() {
47          $('#game-ui').append('<h2 class="display" id="pc-pos"></h2>');
48          $('#game-ui').append('<h2 class="display" id="life"></h2>');
49          $('#game-ui').append('<h2 class="display" id="goal"></h2>');
50      }
51
52      game.display = function(id,value) {
53          $('#'+id).text(value);
54      }
55
56      // game viewport (screen)
57      game.initVP = function() {
58          // pc object
59          this.pc = new Object();
60          this.pc.x = parseInt(this.start_pos_x);
61          this.pc.y = parseInt(this.start_pos_y);
62          this.pc.vel = 2;
63          this.pc.image = this.assets.pc;
64          this.pc.w = this.pc.image.width;
65          this.pc.h = this.pc.image.height;
66          this.pc.life = parseInt(this.max_life);
67          game.display('life','Life: '+game.pc.life);
68          game.display('goal','Goal: '+this.goal_pos_x+','+this.goal_pos_y);
69
70          this.mapW = 640;
71          this.mapH = 480;
72
73          this.viewport = new Object();
74          this.viewport.width = 640;
75          this.viewport.height = 480;
76          this.viewport.canvas = document.getElementById('game-window').getContext("2d");
77
78          game.viewport.drawpc = function(pc) {
79              this.canvas.clearRect(0,0,640,480);
80              this.canvas.drawImage(game.assets.maze,0,0);
81              game.mapData = this.canvas.getImageData(0,0,game.mapW,game.mapH).data;
82              this.canvas.drawImage(pc.image, pc.x, pc.y);
83              game.pc.imgData = this.canvas.getImageData(pc.x, pc.y, pc.w, pc.h).data;
84          }
85          this.viewport.drawpc(this.pc);
86          this.int = setInterval(game.loop, 16);
87      }
88
89      game.testCollision = function(x,y) {
90          // this.pc.w, this.pc.h, this.pc.image
91          var xMin = x, yMin = y
92          , xMax = x+this.pc.w, yMax = y+this.pc.h
93          , w = this.mapW, w2 = this.pc.w
94          , pixels1 = this.mapData
```

```
95          , pixels2 = this.pc.imgData;
96
97          // loop
98          for (var pixelX = xMin; pixelX < xMax; pixelX++)
99              for (var pixelY = yMin; pixelY < yMax; pixelY++) {
100                 var px1 = ((pixelX) + (pixelY) * w)*4 + 3; // pixel in maze relative to x,y
101                 var px2 = ((pixelX-x) + (pixelY-y) * w2)*4 + 3; // pixel in PC relative to 0,0
102                 if ( pixels1[px1] !== 0 && pixels2[px2] !== 0) {
103                     return true;
104                 }
105             }
106
107         return false;
108     }
109
110     game.movepc = function () {
111
112         var getColor = function(x,y) {
113             return game.viewport.canvas.getImageData(newX,newY,1,1).data;
114         }
115
116         var oldX = this.pc.x, oldY = this.pc.y;
117         var dx = 0, dy = 0;
118
119         if (38 in game.keysDown) dy = -this.pc.vel // cima
120         if (40 in game.keysDown) dy = this.pc.vel // baixo
121         if (37 in game.keysDown) dx = -this.pc.vel // esquerda
122         if (39 in game.keysDown) dx = this.pc.vel // direita
123
124         if (dx != 0 || dy != 0) {
125             var newX = this.pc.x + dx;
126             var newY = this.pc.y + dy;
127
128
129             if ((newX > 0) && ((newX + this.pc.w) < this.mapW)) {
130                 this.pc.x = newX;
131                 game.redraw = true;
132             }
133
134             if ((newY > 0) && ((newY + this.pc.h) < this.mapH)) {
135                 this.pc.y = newY;
136                 game.redraw = true;
137             }
138
139             if (game.redraw == true) {
140                 if (game.testCollision(newX,newY)) {
141                     game.pc.x = oldX - dx;
142                     game.pc.y = oldY - dy;
143                     game.pc.life -= (game.life_loss/2);
144                     game.keysDown.length = 0; // clear the keys buffer to avoid continuous energy loss
145                 }
146             }
147         }
148
149         this.display('pc-pos',"Pos: "+this.pc.x+","+this.pc.y);
150     }
151
152     game.run = function() {
153         this.won = false;
```

```
154         this.lost = false;
155         this.initUI();
156         this.loadAssets();
157     }
158 
159     game.loop = function() {
160         game.movepc();
161         if (game.redraw) {
162 
163             // losing condition
164             if (game.pc.life <= 0) {
165                 game.pc.life = 0;
166                 game.lost = true;
167             }
168 
169             // winning condition:
170             if ((game.goal_pos_x >= game.pc.x && game.goal_pos_x <= (game.pc.x + game.pc.w)) &&
171                 (game.goal_pos_y >= game.pc.y && game.goal_pos_y <= (game.pc.y + game.pc.h))) {
172                 game.won = true;
173             }
174 
175             game.viewport.drawpc(game.pc);
176             game.display('life','Life: '+game.pc.life);
177             game.redraw = false;
178         }
179         if (game.lost) {
180             clearInterval(game.int);
181             $("#game-ui").append("<h1 class='bigmessage'>GAME OVER</h1>");
182         }
183         if (game.won) {
184             clearInterval(game.int);
185             $("#game-ui").append("<h1 class='bigmessage'>YOU WON!!!</h1>");
186         }
187 
188         if (game.won || game.lost) {
189             $("#modal-overlay")
190                 .append("<a class='again' href='#'>Play again!</a>")
191                 .click(function (ev) {
192                     $(".bigmessage").remove();
193                     $("a.again").remove();
194                     ev.preventDefault();
195                     game.pc.life = game.max_life;
196                     game.run();
197                 })
198         }
199     }
200 
201     // modal panel
202     $.fn.extend({
203         modalPanel: function() {
204 
205             //Create our overlay object
206             var overlay = $("<div id='modal-overlay'></div>");
207             var gameWindow = $("<div id='game-ui'><a id='quit' href='#'>Quit</a>"+
208                                "<canvas id='game-window' class ='comedic'"+
209                                " height='480' width='640'></canvas></div>");
210 
211             return this.each(function() {
212 
```

```
213                 //Listen for clicks on objects passed to the plugin
214                 $(this).click(function(e) {
215 
216                     //Append the overlay to the document body
217                     $("body").append(overlay);
218                     //Set the css and fade in our overlay
219                     overlay.css("opacity", 1.0);
220                     overlay.fadeIn(150);
221 
222                     overlay.append(gameWindow);
223                     gameWindow.fadeIn();
224                     $("#quit").click(function() {
225                         modalHide();
226                         return false;
227                     });
228 
229                     //Prevent the anchor link from loading
230                     e.preventDefault();
231 
232                     //Activate a listener
233                     $(document).keydown(handleEscape);
234 
235                     game.maze_url = $(this).attr('maze');
236                     game.pc_url = $(this).attr('pc');
237 
238                     game.max_life = parseInt($(this).attr('max_life'));
239                     game.life_loss= parseInt($(this).attr('life_loss'));
240                     game.start_pos_x = parseInt($(this).attr('start_pos_x'));
241                     game.start_pos_y = parseInt($(this).attr('start_pos_y'));
242                     game.goal_pos_x = parseInt($(this).attr('goal_pos_x'));
243                     game.goal_pos_y = parseInt($(this).attr('goal_pos_y'));
244 
245                     // run the game code
246                     game.run();
247                 });
248             });
249 
250             //Our function for hiding the modalbox
251             function modalHide() {
252 
253                 $(document).unbind("keydown", handleEscape)
254                 var remove = function() {
255                     clearInterval(game.int);
256                     $(this).empty().remove();
257                 }
258                 //gameWindow.fadeOut(remove);
259 
260                 overlay.fadeOut(remove);
261             }
262 
263             //Our function that listens for escape key.
264             function handleEscape(e) {
265 
266                 if (e.keyCode == 27) {
267                     modalHide();
268                 }
269             }
270         }
271     }); // END modal window functions
```

```
272
273
274     // call the modal pane
275     Drupal.behaviors.playgame = {
276         attach: function (context) {
277             $('#playgame', context).modalPanel();
278         }
279     };
280
281 })(jQuery);
```

A.2 game.css

```
1  .ds-center {
2  }
3
4  #game-window {
5      position: absolute;
6      left: 0;
7      top: 60px;
8      border: 2px solid white
9  }
10
11 #game-ui {
12     width: 640px;
13     height: 540px;
14     position: absolute;
15     z-index: 110;
16     left: 50%;
17     top: 50%;
18     margin-left: -320px;
19     margin-top: -270px;
20 }
21
22 #modal-overlay {
23     position: fixed;
24     z-index: 100;
25     top: 0px;
26     left: 0px;
27     height: 100%;
28     width: 100%;
29     background: #000;
30     display: none;
31 }
32
33 .display {
34     color: white;
35 }
36
37 #life {
38     display: block;
39     position: absolute;
40     top: 25px;
41     left: 0;
42 }
43
```

```
44  #pc-pos {
45      display: block;
46      position: absolute;
47      top: 25px;
48      left: 200px;
49  }
50
51  #goal {
52      display: block;
53      position: absolute;
54      top: 25px;
55      left: 400px;
56  }
57
58
59  #quit {
60      color: #ff9800;
61      display: block;
62      position: absolute;
63      top: 30px;
64      right: 0px;
65      font-weight: bold;
66  }
67
68  .bigmessage {
69      display: block;
70      font-size: 7em;
71      color: #ff9800;
72      position: absolute;
73      z-index: 200;
74      top: 50%;
75      left: 50%;
76      margin-top: -68px;
77      margin-left: -301px;
78  }
79
80  a.again {
81      display: block;
82      font-size: 2em;
83      color: #ff9800;
84      position: absolute;
85      z-index: 200;
86      top: 50%;
87      left: 50%;
88      margin-top: 100px;
89      margin-left: -72px;
90  }
91
92  #playgame {
93      position: absolute;
94      top: 125px;
95      left: 85px;
96      color: black;
97      font-weight: bold;
98      font-size: large;
99      background-color: #ff9800;
100     text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
101     display: inline-block;
102     padding: 4px 10px 4px;
```

```
103     text-align: center;
104     vertical-align: middle;
105     cursor: pointer;
106     border: 1px solid #CCC;
107     -webkit-border-radius: 4px;
108     border-radius: 4px;
109     -moz-border-radius: 4px;
110     -webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);
111     box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);
112 }
113
114 .field-name-field-screenshot .field-items {
115     float: left;
116     margin-right: 10px;
117 }
118
119 .field-name-author {
120     clear: both;
121 }
```

Appendix B

Prototype usage & installation

B.1 Accessing and using the prototype

There is an on-line available installation of the prototype which is ready for use and testing. The link is the following:

http://games.jmgpena.net

The portal is fully functional and users can submit a new registration (which will have to be approved by an administrator). Any questions, or bug reports should be forwarded to 'jorge@jmgpena.net'.

B.2 Installing and running prototype

If one wants to install a local version of the prototype for testing there are two ways. The first is to use a snapshot of the code and the database provided in the CD, and the second is to install a clean copy of Drupal and add the theme and modules necessary for a brand new install of the games creation portal.

B.2.1 Common requirements

To install and test the prototype the following software should be installed and running beforehand:

- Apache HTTP server (http://apache.org).
- PHP integrated with apache (module or CGI) (http://www.php.net).
- MySQL database server (http://www.mysql.com).

For an easy way to install all these servers on Windows, Linux or Mac OSX checkout XAMPP at http://www.apachefriends.org/en/index.html.

B.2.2 Snapshot installation

For this type of install you will need to decompress the Drupal installation files in the CD (collabgamedev-files.tar.gz) with the command:

```
$ tar xzvf collabgamedev-files.tar.gz
```

The files should be unpacked into a directory that can be served by the Apache web server. After this you must unpack the database dump file (collabgamedev-db.sql.gz)

```
$ gunzip collabgamedev-db.sql.gz
```

Then you must import this file into MySQL. Next you should create user in MySQL with permission to create and use tables in the database. Finally you should check the file in 'sites/default/settings.php' in the Drupal installation and update the user and password for database access.

If every thing went well if you point your browser to your Apache server you should experience the prototype as it was when the snapshots were made. The administrator user and password are: admin/admin.

After successful login you should flush all the caches to get rid clear the database of cached content pertinent to the original installation.

B.2.3 Clean installation

ATTENTION This installation is complex and requires considerable technical skills and familiarization Drupal installation and administration.

These are the necessary steps to finish the install:

1. For this installation I recommend the use of Drush (the Drupal command-line shell). Drush can be found at http://www.drush.org/.

2. For this install first you should start by downloading and installing a fresh copy of Drupal from (http://drupal.org). The instructions for this can be found at the same location.

3. After Drupal is installed and running you must install the following modules and their dependencies:

   - Views (http://drupal.org/project/views)
   - Display Suite (http://drupal.org/project/ds)
   - Features (http://drupal.org/project/features)
   - Entity Reference (http://drupal.org/project/entityreference)

- Fivestar (http://drupal.org/project/fivestar)
- AdaptiveTheme (http://drupal.org/project/adaptivetheme)

4. Check your Drupal installation to see if everything is ok and the modules are enabled.
5. Add the 'media-library-7.x-1.0.tar' module to the 'sites/default/modules/' folder of your installation (or use Drush). This module was exported using the *features* module and contains all the content types, views, and configurations needed to run the game creation portal.
6. Finally install and enable the custom theme provided in the file 'h5games-theme.tar.gz' (to the folder 'sites/default/themes/').
7. Test your installation by uploading content, creating a game and then playing it.
8. Congratulations.